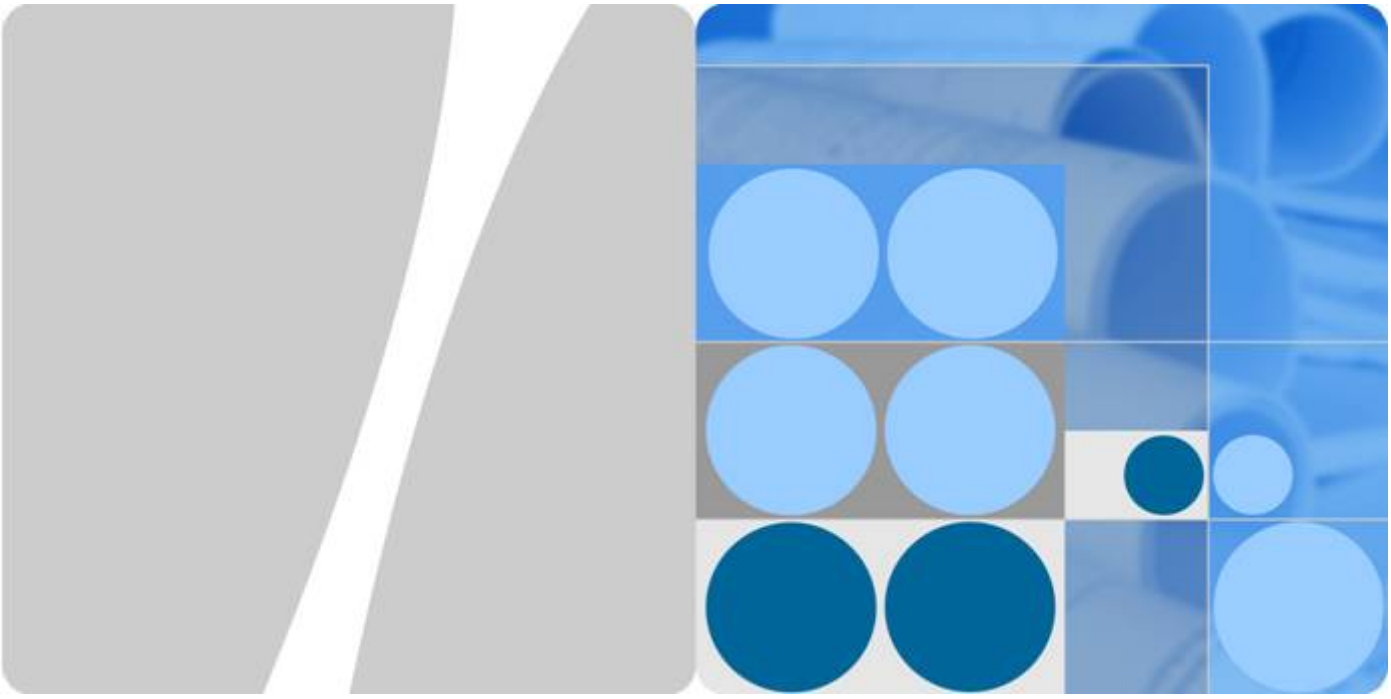


ModelArts

用户指南

发布日期　2024-04-30

目 录

1 产品介绍

1.1 图解 ModelArts

1.1.1 初识 ModelArts


初识ModelArts
更快的普惠AI开发平台
AI开发当前最大的挑战是什么？
训练耗时长
资源紧张
华为云ModelArts产品优势
ModelArts是面向AI开发者的一站式开发平台，提供海量数据预处理及半自动化标注、大规模分布式训练、自动化模型生成，以及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期AI工作流。
01
数据准备效率百倍提升
视频
图片集
标注
02
模型训练耗时降低一半
算法优化
快
简化调参
简
1000规模集群，训练加速比0.8
03
模型一键部署到云、边、端
AI模型
部署
边缘推理
在线推理
批量推理
04
用 AI 方式加速 AI 开发过程 - 自动学习
UI向导
自适应训练
05
快亦有道 - 匠心打造全流程管理
开发流程的自动可视化
训练断点重启
训练结果轻松比对
06
AI 共享 - 帮开发者实现 AI 资源复用
企业内共享
AI共享平台
外部市场
效率提升
数据
模型
应用
开放生态
应用场景
视频分析
图像识别
产品推荐
异常检测
语音识别

1.2 什么是 ModelArts

ModelArts是面向AI开发者的一站式开发平台，提供海量数据预处理及半自动化标注、大规模分布式训练、自动化模型生成及模型按需部署能力，帮助用户快速创建和部署模型，管理全周期AI工作流。

“一站式”是指AI开发的各个环节，包括数据处理、算法开发、模型训练、模型部署都可以在ModelArts上完成。从技术上看，ModelArts底层支持各种异构计算资源，开发者可以根据需要灵活选择使用，而不需要关心底层的技术。同时，ModelArts支持Tensorflow、PyTorch、MindSpore等主流开源的AI开发框架，也支持开发者使用自研的算法框架，匹配您的使用习惯。

ModelArts的理念就是让AI开发变得更简单、更方便。

产品优势

- **一站式**

  开“箱”即用，涵盖AI开发全流程，包含数据处理、模型开发、训练、管理、部署功能，可灵活使用其中一个或多个功能。
- **易上手**
  - 模型超参自动优化，简单快速。
  - 零代码开发，简单操作训练出自己的模型。
- **高性能**
  - 可生成在Ascend芯片上运行的模型，实现高效端边推理。
- **灵活**
  - 支持多种主流开源框架(TensorFlow、PyTorch、MindSpore等)。
  - 支持专属资源独享使用。
  - 支持自定义镜像满足自定义框架及算子需求。

1.3 功能介绍

繁多的AI工具安装配置、数据准备、模型训练慢等是困扰AI工程师的诸多难题。为解决这个难题，将一站式的AI开发平台（ModelArts）提供给开发者，从数据准备到算法开发、模型训练，最后把模型部署起来，集成到生产环境。一站式完成所有任务。

图 1-1 功能总览

ModelArts特色功能如下所示：

- **数据治理**

  支持数据筛选、标注等数据处理，提供数据集版本管理，特别是深度学习的大数据集，让训练结果可重现。
- **极“快”致“简”模型训练**

  自研的MoXing深度学习框架，更高效更易用，有效提升训练速度。
- **多场景部署**

  支持模型部署到多种生产环境，可部署为云端在线推理和批量推理。
- **自动学习**

  支持多种自动学习能力，通过“自动学习”训练模型，用户不需编写代码即可完成自动建模、一键部署。
- **AI Hub**

  预置常用算法和常用数据集，支持模型在企业内部共享或者公开共享。

1.4 基础知识

1.4.1 AI 开发基本流程介绍

什么是 AI

AI（人工智能）是通过机器来模拟人类认识能力的一种科技能力。AI最核心的能力就是根据给定的输入做出判断或预测。

AI 开发的目的是什么

AI开发的目的是将隐藏在一大批数据背后的信息集中处理并进行提炼，从而总结得到研究对象的内在规律。

对数据进行分析，一般通过使用适当的统计、机器学习、深度学习等方法，对收集的大量数据进行计算、分析、汇总和整理，以求最大化地开发数据价值，发挥数据作用。

AI 开发的基本流程

AI开发的基本流程通常可以归纳为几个步骤：确定目的、准备数据、训练模型、评估模型、部署模型。

图 1-2 AI 开发流程

步骤1　确定目的

在开始AI开发之前，必须明确要分析什么？要解决什么问题？商业目的是什么？基于商业的理解，整理AI开发框架和思路。例如，图像分类、物体检测等等。不同的项目对数据的要求，使用的AI开发手段也是不一样的。

步骤2 准备数据

数据准备主要是指收集和预处理数据的过程。

按照确定的分析目的，有目的性的收集、整合相关数据，数据准备是AI开发的一个基础。此时最重要的是保证获取数据的真实可靠性。而事实上，不能一次性将所有数据都采集全，因此，在数据标注阶段你可能会发现还缺少某一部分数据源，反复调整优化。

步骤3 训练模型

俗称“建模”，指通过分析手段、方法和技巧对准备好的数据进行探索分析，从中发现因果关系、内部联系和业务规律，为商业目的提供决策参考。训练模型的结果通常是一个或多个机器学习或深度学习模型，模型可以应用到新的数据中，得到预测、评价等结果。

业界主流的AI引擎有TensorFlow、PyTorch、MindSpore等，大量的开发者基于主流AI引擎，开发并训练其业务所需的模型。

步骤4 评估模型

训练得到模型之后，整个开发过程还不算结束，需要对模型进行评估和考察。经常不能一次性获得一个满意的模型，需要反复的调整算法参数、数据，不断评估训练生成的模型。

一些常用的指标，如准确率、召回率、AUC等，能帮助您有效的评估，最终获得一个满意的模型。

步骤5 部署模型

模型的开发训练，是基于之前的已有数据（有可能是测试数据），而在得到一个满意的模型之后，需要将其应用到正式的实际数据或新产生数据中，进行预测、评价、或以可视化和报表的形式把数据中的高价值信息以精辟易懂的形式提供给决策人员，帮助其制定更加正确的商业策略。

----结束

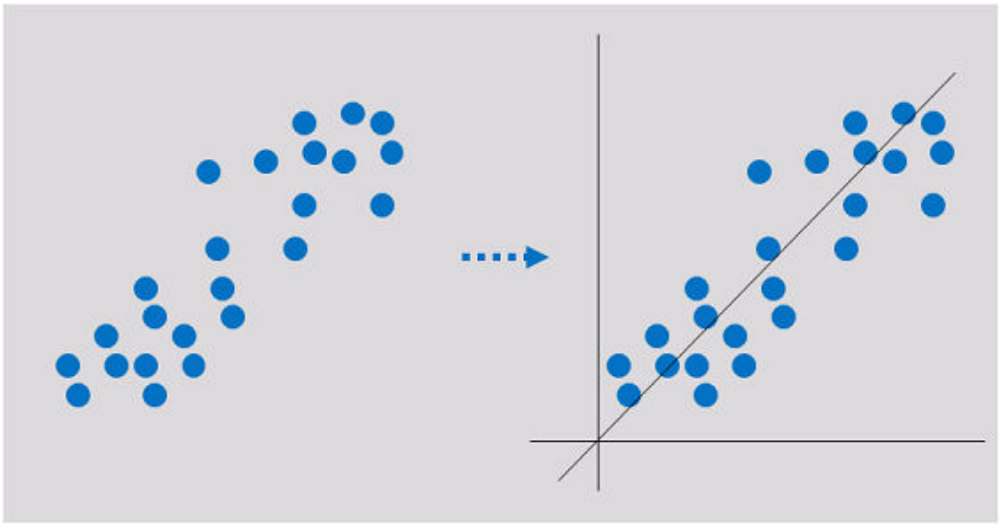
1.4.2 AI 开发基本概念

机器学习常见的分类有3种：

- 监督学习：利用一组已知类别的样本调整分类器的参数，使其达到所要求性能的过程，也称为监督训练或有教师学习。常见的有回归和分类。
- 非监督学习：在未加标签的数据中，试图找到隐藏的结构。常见的有聚类。
- 强化学习：智能系统从环境到行为映射的学习，以使奖励信号（强化信号）函数值最大。

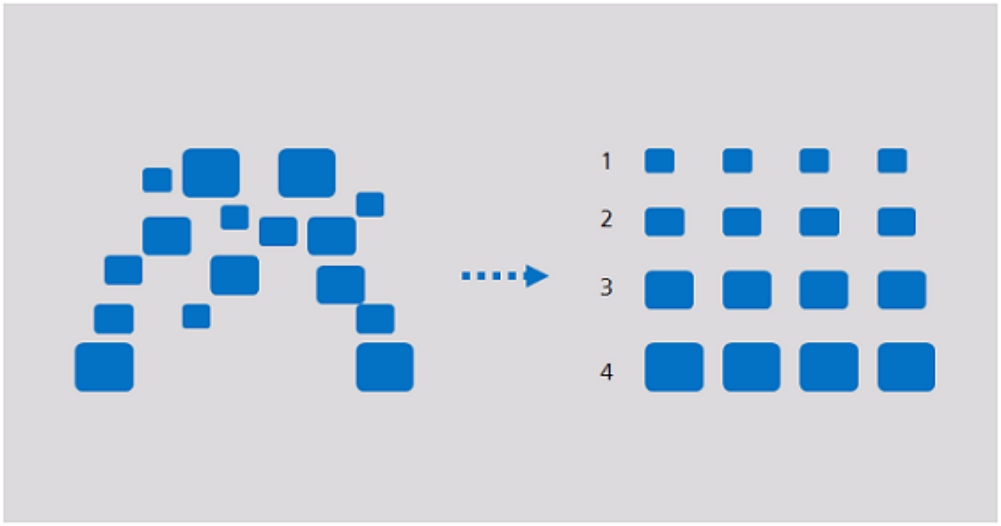
回归

回归反映的是数据属性值在时间上的特征，产生一个将数据项映射到一个实值预测变量的函数，发现变量或属性间的依赖关系，其主要研究问题包括数据序列的趋势特征、数据序列的预测以及数据间的关系等。它可以应用到市场营销的各个方面，如客户寻求、保持和预防客户流失活动、产品生命周期分析、销售趋势预测及有针对性的促销活动等。



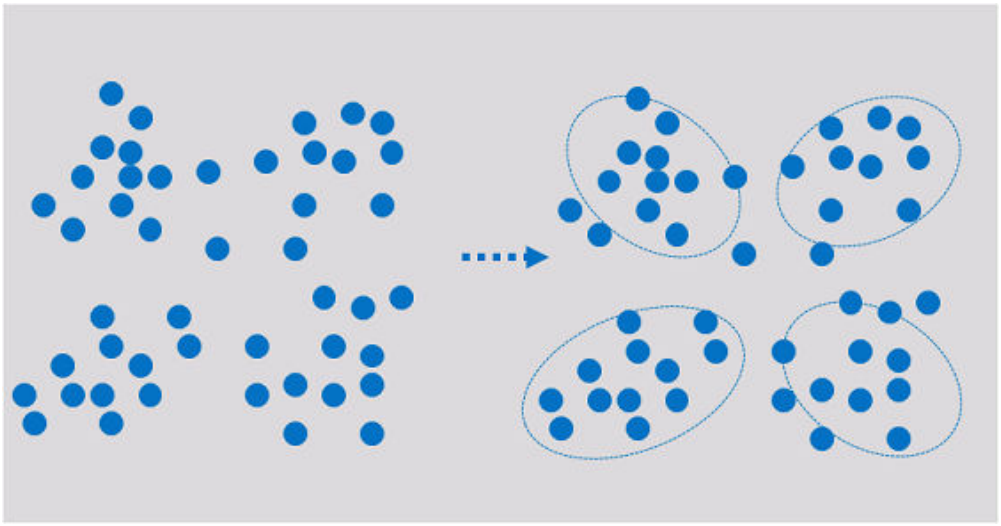
分类

分类是找出一组数据对象的共同特点并按照分类模式将其划分为不同的类，其目的是通过分类模型，将数据项映射到某个给定的类别。它可以应用到客户的分类、客户的属性和特征分析、客户满意度分析、客户的购买趋势预测等。



聚类

聚类是把一组数据按照相似性和差异性分为几个类别，其目的是使得属于同一类别的数据间的相似性尽可能大，不同类别中的数据间的相似性尽可能小。它可以应用到客户群体的分类、客户背景分析、客户购买趋势预测、市场的细分等。



与分类不同，聚类分析数据对象，而不考虑已知的类标号（一般训练数据中不提供类标号）。聚类可以产生这种标号。对象根据最大化类内的相似性、最小化类间的相似性的原则进行聚类或分组。对象的聚类是这样形成的，使得在一个聚类中的对象具有很高的相似性，而与其他聚类中的对象很不相似。

1.4.3 ModelArts 中常用概念

自动学习

自动学习功能可以根据标注数据自动设计模型、自动调参、自动训练、自动压缩和部署模型，不需要代码编写和模型开发经验。只需三步，标注数据、自动训练、部署模型，即可完成模型构建。

推理

指按某种策略由已知判断推出新判断的思维过程。人工智能领域下，由机器模拟人类智能，使用构建的神经网络完成推理过程。

在线推理

在线推理是对每一个推理请求同步给出推理结果的在线服务（Web Service）。

批量推理

批量推理是对批量数据进行推理的批量作业。

Ascend 芯片

Ascend芯片是高计算力低功耗的AI芯片。

资源池

ModelArts提供的大规模计算集群，可应用于模型开发、训练和部署。支持公共资源池和专属资源池两种，分别为共享资源池和独享资源池。ModelArts默认提供公共资源池。专属资源池需单独创建，专属使用，不与其他用户共享。

AI Hub

预置常用模型和算法，您可以直接获取使用。您也可以将自己开发的模型、算法或数据集分享至市场，共享给个人或者公开共享。

MoXing

MoXing是ModelArts自研的组件，是一种轻型的分布式框架，构建于TensorFlow、PyTorch、MXNet、MindSpore等深度学习引擎之上，使得这些计算引擎分布式性能更高，同时易用性更好。MoXing包含很多组件，其中MoXing Framework模块是一个基础公共组件，可用于访问OBS服务，和具体的AI引擎解耦，在ModelArts支持的所有AI引擎(TensorFlow、MXNet、PyTorch、MindSpore等)下均可以使用。

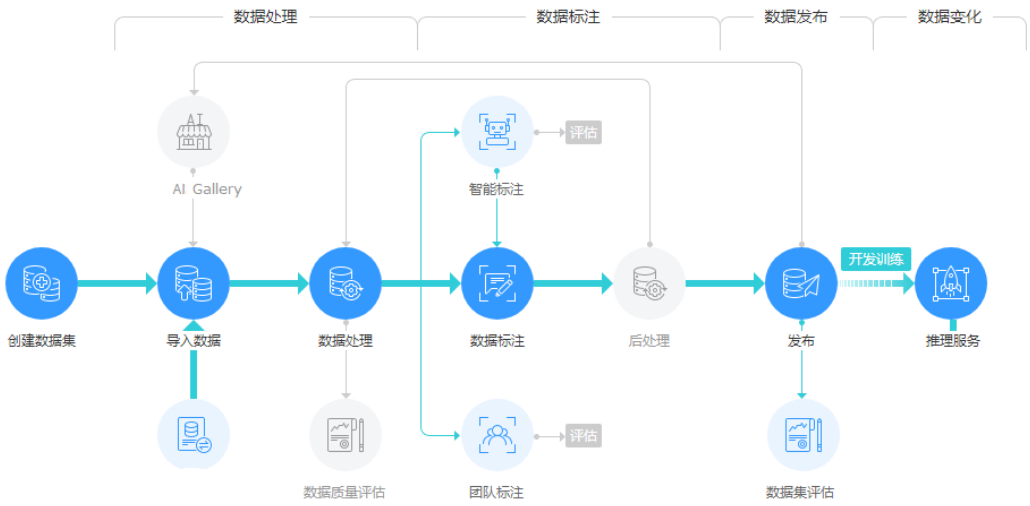
MoXing Framework模块提供了OBS中常见的数据文件操作，如读写、列举、创建文件夹、查询、移动、复制、删除等。

在ModelArts Notebook中使用MoXing接口时，可直接调用接口，无需下载或安装SDK，使用限制比ModelArts SDK和OBS SDK少，非常便捷。

1.4.4 数据管理

AI开发过程中经常需要处理海量数据，数据准备与标注耗费整体开发一半以上时间。ModelArts数据管理提供了一套高效便捷的管理和标注数据框架。不仅支持图片、文本、语音、视频等多种数据类型，涵盖图像分类、目标检测、音频分割、文本分类等多个标注场景，可适用于各种AI项目，如计算机视觉、自然语言处理、音视频分析等；同时提供数据筛选、数据分析、数据处理、团队标注以及版本管理等功能，AI开发者可基于该框架实现数据标注全流程处理。如图1-3所示。

图 1-3 数据标注全流程



数据管理平台提供了聚类分析、数据特征分析、数据清洗、数据校验、数据增强、数据选择等分析处理能力，可帮助开发者进一步理解数据和挖掘数据，从而准备出一份满足开发目标或项目要求的高价值数据。

开发者在数据管理平台可以在线完成图像分类、目标检测、音频分割、文本三元组、视频分类等各种标注场景，同时也可以使用ModelArts智能标注方案，通过预置算法或自定义算法代替人工完成数据标注，提升标注效率。

针对大规模协同标注场景，数据管理平台还提供了强大的团队标注，支持标注团队管理、人员管理、角色管理等，实现从项目的创建、数据分配、进度把控、标注、审核、验收全流程。为用户带来标注效率提升的同时，又最小化项目管理开销。

此外，数据管理平台时刻保障用户数据的安全性和隐私性，确保用户数据仅在授权范围内使用。

新版数据管理中将数据集和数据标注功能解耦，更方便用户使用。

1.4.5 开发环境

软件开发的历史，就是一部降低开发者成本，提升开发体验的历史。在AI开发阶段，ModelArts也致力于提升AI开发体验，降低开发门槛。ModelArts开发环境，以云原生的资源使用和开发工具链的集成，目标为不同类型AI开发、探索、教学用户，提供更好云化AI开发体验。

ModelArts Notebook云上云下，无缝协同

- 代码开发与调测。云化JupyterLab使用，本地IDE+ModelArts插件远程开发能力，贴近开发人员使用习惯
- 云上开发环境，包含AI计算资源，云上存储，预置AI引擎
- 运行环境自定义，将开发环境直接保存成为镜像，供训练、推理使用

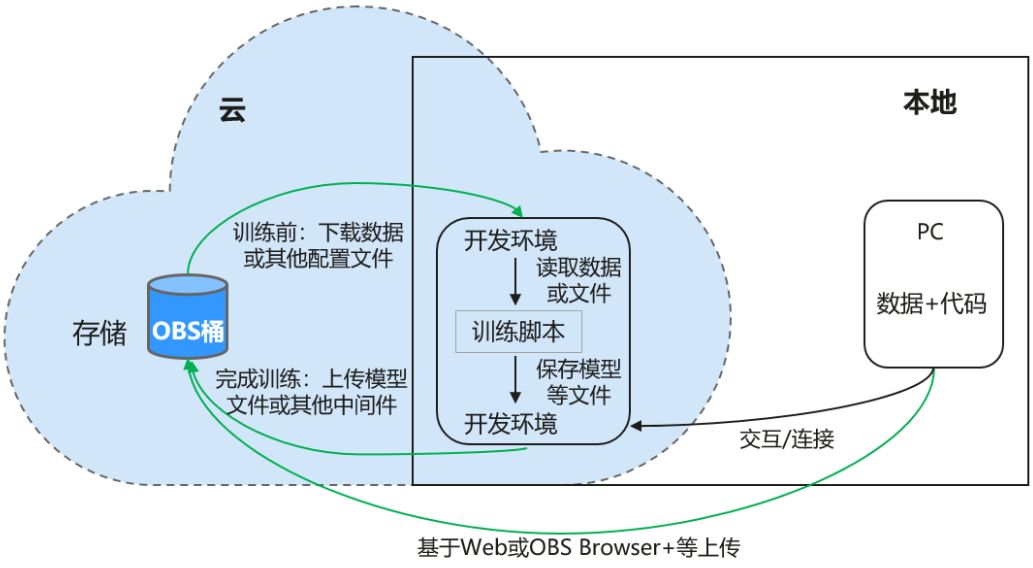
亮点特性 1：远程开发 - 支持本地 IDE 远程访问 Notebook

Notebook提供了远程开发功能，通过开启SSH连接，用户本地IDE可以远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

对于使用本地IDE的开发者，由于本地资源限制，运行和调试环境大多使用团队公共搭建的资源服务器，并且是多人共用，这带来一定的环境搭建和维护成本。

而ModelArts的Notebook的优势是即开即用，它预先装好了不同的AI引擎，并且提供了非常多的可选规格，用户可以独占一个容器环境，不受其他人的干扰。只需简单配置，用户即可通过本地IDE连接到该环境进行运行和调试。

图 1-4 本地 IDE 远程访问 Notebook 开发环境



ModelArts的Notebook可以视作是本地PC的延伸，均视作本地开发环境，其读取数据、训练、保存文件等操作与常规的本地训练一致。

对于习惯使用本地IDE的开发者，使用远程开发方式，不影响用户的编码习惯，并且可以方便快捷的使用云上的Notebook开发环境。

本地IDE当前支持VS Code、PyCharm、SSH工具。还有专门的插件PyCharm Toolkit和VS Code Toolkit，方便将云上资源作为本地的一个扩展。

亮点特性 2：开发环境保存 - 支持一键镜像保存

ModelArts的新版Notebook提供了镜像保存功能。支持一键将运行中的Notebook实例保存为镜像，将准备好的环境保存下来，可以作为自定义镜像，方便后续使用，并且方便进行分享。

保存镜像时，安装的依赖包（pip包）不丢失，VS Code远程开发场景下，在Server端安装的插件不丢失。

亮点特性 3：预置镜像 - 即开即用，优化配置，支持主流 AI 引擎

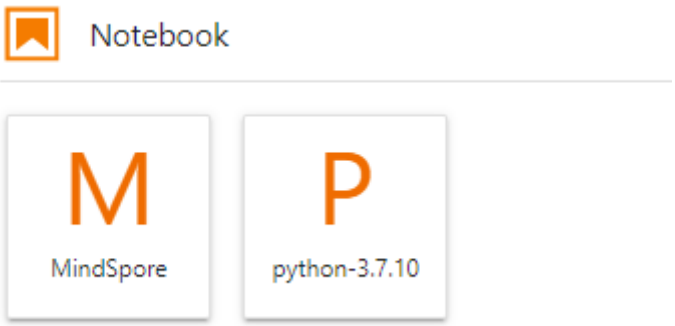
每个镜像预置的AI引擎和版本是固定的，在创建Notebook实例时明确AI引擎和版本，包括适配的芯片。

ModelArts开发环境给用户提供了一组预置镜像，主要包括PyTorch、Tensorflow、MindSpore系列。用户可以直接使用预置镜像启动Notebook实例，在实例中开发完成后，直接提交到ModelArts训练作业进行训练，而不需要做适配。

ModelArts开发环境提供的预置镜像版本是依据用户反馈和版本稳定性决定的。当用户的功能开发基于ModelArts提供的版本能够满足的时候，建议用户使用预置镜像，这些镜像经过充分的功能验证，并且已经预置了很多常用的安装包，用户无需花费过多的时间来配置环境即可使用。

ModelArts开发环境提供的预置镜像主要包含：

- 常用预置包，基于标准的Conda环境，预置了常用的AI引擎，例如PyTorch、MindSpore；常用的数据分析软件包，例如Pandas、Numpy等；常用的工具软件，例如cuda、cudnn等，满足AI开发常用需求。
- 预置Conda环境：每个预置镜像都会创建一个相对应的Conda环境和一个基础Conda环境python（不包含任何AI引擎），如预置Mindspore所对应的Conda环境如下：



  用户可以根据是否使用AI引擎参与功能调试，并选择不同的Conda环境。
- Notebook：是一款Web应用，能够使用户在界面编写代码，并且将代码、数学方程和可视化内容组合到一个文档中。

- JupyterLab插件：插件包括规格切换，停止实例等，提升用户体验。
- 支持SSH远程连接功能，通过SSH连接启动实例，在本地调试就可以操作实例，方便调试。

说明

- 为了简化操作，ModelArts的新版Notebook，同一个Notebook实例中不支持不同引擎之间的切换。
- 不同Region支持的AI引擎不一样，请以控制台实际界面为准。

亮点特性 4：提供在线的交互式开发调试工具 JupyterLab

ModelArts集成了基于开源的JupyterLab，可为您提供在线的交互式开发调试。您无需关注安装配置，在ModelArts管理控制台直接使用Notebook，编写和调测模型训练代码，然后基于该代码进行模型的训练。

JupyterLab是一个交互式的开发环境，是Jupyter Notebook的下一代产品，可以使用它编写Notebook、操作终端、编辑MarkDown文本、打开交互模式、查看csv文件及图片等功能。

1.5 ModelArts 支持的 AI 框架

ModelArts的开发环境Notebook、训练作业、模型推理（即AI应用管理和部署上线）支持的AI框架及其版本，不同模块的呈现方式存在细微差异，各模块支持的AI框架请参见如下描述。

统一镜像列表

ModelArts提供了ARM+Ascend规格的统一镜像，包括MindSpore、PyTorch。适用于开发环境，模型训练，服务部署，请参考**统一镜像列表**。

表 1-1 MindSpore

预置镜像	适配芯片	适用范围
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook、训练、推理部署
mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook、训练、推理部署
mindspore_2.2.10-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Ascend snt9c	Notebook、训练、推理部署

表 1-2 PyTorch

预置镜像	适配芯片	适用范围
pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook、训练、推理部署

预置镜像	适配芯片	适用范围
pytorch_2.1.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Ascend snt9c	Notebook、训练、推理部署
pytorch_1.11.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Ascend snt9c	Notebook、训练、推理部署

开发环境 Notebook

开发环境的Notebook，根据不同的工作环境，对应支持的镜像和版本有所不同。

表 1-3 新版 Notebook 支持的镜像

镜像名称	镜像描述	适配芯片	支持SSH远程开发访问	支持在线JupyterLab访问
pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是
pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine	Ascend	是	是
mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置MindSpore	Ascend	是	是
pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是

镜像名称	镜像描述	适配芯片	支持SSH远程开发访问	支持在线JupyterLab访问
mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是
mindstudio5.0.rc1-ascend-cann5.1.rc1-euler2.8.3-aarch64	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	否
mindspore1.8.0-cann5.1.2-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是
tensorflow1.15-cann5.1.0-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. TensorFlow are preset in the AI engine.	Ascend	是	是
mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3	Ascend+ARM算法开发和训练基础镜像，AI引擎预置MindSpore	Ascend	是	是
pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是

镜像名称	镜像描述	适配芯片	支持SSH远程开发访问	支持在线JupyterLab访问
tensorflow1.15-mindspore1.7.0-cann5.1.0-euler2.8-aarch64	Ascend+ARM algorithm development and training. TensorFlow and MindSpore are preset in the AI engine.	Ascend	是	是
tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是
tensorflow1.15.0-cann5.1.2-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是

训练作业

创建训练作业时，训练支持的AI引擎及对应版本如下所示。

预置引擎命名格式如下：

```
<训练引擎名称_版本号>-[cpu | <cuda_版本号 | cann_版本号 >]-<py_版本号>-<操作系统名称_版本号>-<x86_64 | aarch64>
```

表 1-4 训练作业支持的 AI 引擎

工作环境	系统架构	系统版本	AI引擎与版本	支持的cuda或Ascend版本
Ascend-Powered-Engine	aarch64	Euler2.8	mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0

工作环境	系统架构	系统版本	AI引擎与版本	支持的cuda或Ascend版本
PyTorch	aarch64	Euler2.8	pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0
TensorFlow	aarch64	Euler2.8	tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0

说明

不同区域支持的AI引擎有差异，请以实际环境为准。

推理支持的 AI 引擎

在ModelArts创建AI应用时，若使用预置镜像“从模板中选择”或“从OBS中选择”导入模型，则支持如下常用引擎及版本的模型包。

说明

- 标注“推荐”的Runtime来源于统一镜像，后续统一镜像将作为主流的推理基础镜像。
- 推荐将旧版镜像切换为统一镜像，旧版镜像后续将会逐渐下线。
- 待下线的基本镜像不再维护。
- 统一镜像Runtime的命名规范：<AI引擎名字及版本> - <硬件及版本：cpu或cuda或cann> - <python版本> - <操作系统版本> - <CPU架构>

表 1-5 支持的常用引擎及其 Runtime

模型使用的引擎类型	支持的运行环境（Runtime）
TensorFlow	tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64
MindSpore	mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64
Pytorch	pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64

1.6 与其他服务的关系

与统一身份认证服务的关系

ModelArts使用统一身份认证服务（Identity and Access Management，简称IAM）实现认证功能。IAM的更多信息请参见《统一身份认证服务用户指南》。

与对象存储服务的关系

ModelArts使用对象存储服务（Object Storage Service，简称OBS）存储数据和模型，实现安全、高可靠和低成本的存储需求。OBS的更多信息请参见《对象存储服务控制台指南》。

与云硬盘的关系

ModelArts使用云硬盘服务（Elastic Volume Service，简称EVS）存储创建的Notebook实例。

与云容器引擎的关系

ModelArts使用云容器引擎（Cloud Container Engine，简称CCE）部署模型为在线服务，支持服务的高并发和弹性伸缩需求。CCE的更多信息请参见《云容器引擎用户指南》。

与容器镜像服务的关系

当使用ModelArts不支持的AI框架构建模型时，可通过构建的自定义镜像导入ModelArts进行训练或推理。您可以通过容器镜像服务（Software Repository for Container，简称SWR）制作并上传自定义镜像，然后再通过容器镜像服务导入ModelArts。SWR的更多信息请参见《容器镜像服务（SWR）用户指南》。

与云监控的关系

ModelArts使用云监控服务（Cloud Eye Service， 简称CES）监控在线服务和对应模型负载，执行自动实时监控、告警和通知操作。CES的更多信息请参见《云监控服务用户指南》。

与云审计的关系

ModelArts使用云审计服务（Cloud Trace Service，简称CTS）记录ModelArts相关的操作事件，便于日后的查询、审计和回溯。CTS的更多信息请参见《云审计服务用户指南》。

1.7 如何访问 ModelArts

- **管理控制台方式**

  ModelArts Standard提供了简洁易用的管理控制台，包含自动学习、数据管理、开发环境、模型训练、AI应用管理、部署上线、AI Hub等功能，您可以在管理控制台端到端完成您的AI开发。

- **API方式**

  如果您需要将ModelArts集成到第三方系统，用于二次开发，请使用API方式访问ModelArts，具体操作和API详细描述，请参见《ModelArts API参考》。

2 准备工作

2.1 配置访问授权（全局配置）

场景描述

ModelArts与其他服务类似，对外暴露的每个功能，都通过IAM的权限来进行控制。比如，用户（此处指IAM子用户，而非租户）希望在ModelArts创建训练作业，则该用户必须拥有 "modelarts:trainJob:create" 的权限才可以完成操作（无论界面操作还是API调用）。

而ModelArts还有一个特殊的地方在于，为了完成AI计算的各种操作，AI平台在任务执行过程中需要访问用户的其他服务，典型的就是训练过程中，需要访问OBS读取用户的训练数据。在这个过程中，就出现了ModelArts“代表”用户去访问其他云服务的情形。从安全角度出发，ModelArts代表用户访问任何云服务之前，均需要先获得用户的授权，而这个动作就是一个“委托”的过程。用户授权ModelArts再代表自己访问特定的云服务，以完成其在ModelArts平台上执行的AI计算任务。

ModelArts提供了一键式自动授权功能，用户可以在ModelArts的全局配置功能中，快速完成委托授权，由ModelArts为用户自动创建委托并配置到ModelArts服务中。

一键式自动授权方式为保证使用业务过程中有足够的权限，基于依赖服务的预置系统策略指定授权范围，创建的委托的权限比较大，基本覆盖了依赖服务的全部权限。如果您需要对委托授权的权限范围进行精确控制，请使用定制化委托授权。更多权限控制的内容请参见**权限管理**章节。

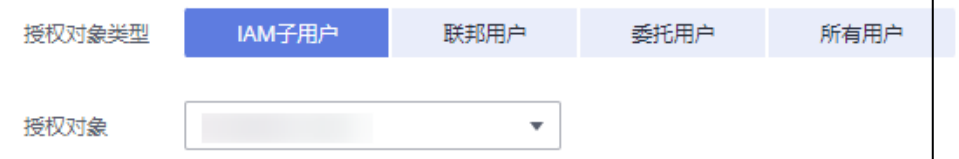
本章节主要介绍一键式自动授权方式。一键式自动授权方式支持给IAM子用户、联邦用户（虚拟IAM用户）、委托用户和所有用户授权。

添加授权

1. 登录ModelArts管理控制台，在左侧导航栏选择“全局配置”，进入“全局配置”页面。
2. 单击“添加授权”，进入“访问授权”配置页面，根据参数说明进行配置。

表 2-1 参数说明

参数	说明
“授权对象类型”	包括IAM子用户、联邦用户、委托用户和所有用户。 ● IAM子用户：由主账号在IAM中创建的用户，是服务的使用人员，具有独立的身份凭证（密码和访问密钥），根据账号授予的权限使用资源。 ● 联邦用户：又称企业虚拟用户。 ● 委托用户：IAM中创建的一个委托。I ● 所有用户：该选项表示会将委托的权限授权到当前账号下的所有子账号、包括未来创建的子账号，授权范围较大，需谨慎使用。个人用户选择“所有用户”即可。

参数	说明
“授权对象”	“授权对象类型”选择“所有用户”时不涉及此参数。 • IAM子用户：选择指定的IAM子用户，给指定的IAM子用户配置委托授权。 **图 2-1** 选择 IAM 子用户  • 联邦用户：输入联邦用户的用户名或用户ID。 **图 2-2** 选择联邦用户  • 委托用户：选择委托名称。使用账号A创建一个权限委托，在此处将该委托授权给账号B拥有的委托。在使用账号B登录控制台时，可以在控制台右上角的个人账号切换角色到账号A，使用账号A的委托权限。 **图 2-3** 委托用户切换角色 
“委托选择”	• 已有委托：列表中如果已有委托选项，则直接选择一个可用的委托为上述选择的用户授权。单击委托名称查看该委托的权限详情。 • 新增委托：如果没有委托可选，可以在新增委托中创建委托权限。对于首次使用ModelArts的用户，需要新增委托。
“新增委托 > 委托名称”	系统自动创建委托名称，用户可以手动修改。

参数	说明
“新增委托 > 授权方式”	• 角色授权：IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。该机制以服务为粒度，提供有限的服务相关角色用于授权。由于各服务之间存在业务依赖关系，因此给用户授予角色时，可能需要一并授予依赖的其他角色，才能正确完成业务。角色并不能满足用户对精细化授权的要求，无法完全达到企业对权限最小化的安全管控要求。 • 策略授权：IAM最新提供的一种细粒度授权的能力，可以精确到具体服务的操作、资源以及请求条件等。基于策略的授权是一种更加灵活的授权方式，能够满足企业对权限最小化的安全管控要求。
“新增委托 > 权限配置 > 普通用户”	普通用户包括用户使用ModelArts完成AI开发的所有必要功能权限，如数据的访问、训练任务的创建和管理等。一般用户选择此项即可。 可以单击“查看权限列表”，查看普通用户权限。
“新增委托 > 权限配置 > 自定义”	如用户有精细化权限管理的需求，可使用自定义模式灵活按需配置ModelArts创建的委托权限。可以根据实际需要在权限列表中勾选要配置的权限。

3. 单击“创建”，即可完成委托配置。

查看授权的权限列表

用户可以在“全局配置”页面的授权列表中，查看已经配置的委托授权内容。单击授权内容列的“查看权限”，可以查看该授权的权限详情。

图 2-4 查看权限

图 2-5 普通用户权限列表

权限名称	使用模块	描述
OBS Administrator	数据管理 | 开发环境 | 训练管理 | AI应用管理 | 部署上线 | AI ...	对象存储服务管理员
DLI FullAccess	数据管理	数据湖探索的所有执行权限
MRS Administrator	数据管理	MapReduce服务（MRS）管理员，拥有该服务下的所有权限
DWS Administrator	数据管理	数据仓库服务（DWS）管理员，拥有该服务下的所有权限
VPC Administrator	训练管理 | AI应用管理 | 部署上线 | 专属资源池	虚拟私有云管理员
CES ReadOnlyAccess	AI应用管理 | 部署上线	云监控服务只读权限
SMN Administrator	训练管理 | AI应用管理 | 部署上线	消息通知服务管理员
EPS FullAccess	训练管理 | AI应用管理 | 部署上线	企业项目管理服务所有权限
CTS Administrator	数据管理 | 开发环境 | 训练管理 | AI应用管理 | 部署上线 | AI ...	云审计服务（CTS）管理员，拥有该服务下的所有权限
SFS ReadOnlyAccess	训练管理	弹性文件服务只读权限
LTS FullAccess	AI应用管理 | 部署上线	云日志服务所有权限
ModelArts CommonOperations	数据管理 | 开发环境 | 训练管理 | AI应用管理 | 部署上线 | AI ...	ModelArts服务普通用户权限（不包括创建、更新、删除专...

修改授权权限范围

1. 在查看授权详情时，如果想要修改授权范围，可以在权限详情页单击“去IAM修改委托权限”。

图 2-6 去 IAM 修改委托权限

2. 进入IAM控制台的委托页面。找到对应的委托信息，修改该委托的基本信息，主要是持续时间。“持续时间”可以选择永久、1天，或者自定义天数，例如 30 天。

图 2-7 手动创建的委托

3. 在授权记录页面单击“授权”，勾选要配置的策略，单击下一步设置最小授权范围，单击确定，完成授权修改。

   设置最小授权范围时，可以选择指定的区域，也可以选择所有区域，即不设置范围。

删除授权

为了更好的管理您的授权，您可以删除某一IAM用户的授权，也可批量清空所有用户的授权。

- **删除某一用户的授权**

在“全局配置”页面，展示当前账号下为其IAM用户配置的授权列表，针对某一用户，您可以单击“操作”列的“删除”，输入“DELETE”后单击“确认”，可删除此用户的授权。删除生效后，此用户将无法继续使用ModelArts的相关功能。

- **批量清空所有授权**

  在“全局配置”页面，单击授权列表上方的“清空授权”，输入“DELETE”后单击“确认”，可删除当前账号下的所有授权。删除生效后，此账号及其所有IAM子用户将无法继续使用ModelArts的相关功能。

常见问题

1. 首次使用ModelArts如何配置授权?

   直接选择“新增委托”中的“普通用户”权限即可，普通用户包括用户使用ModelArts完成AI开发的所有必要功能权限，如数据的访问、训练任务的创建和管理等。一般用户选择此项即可。

2. 访问密钥授权哪去了?

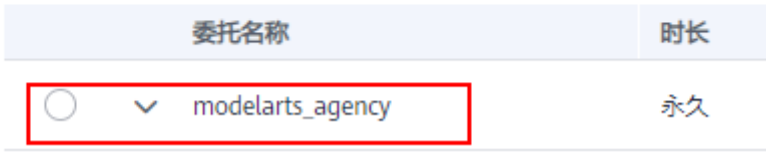
   全局配置中使用密钥委托授权功能已下线，对于之前使用访问密钥授权的老用户，建议修改为委托授权方式。在全局配置页面，一键“清空授权”，然后再使用委托授权完成授权配置。

3. 如何获取访问密钥AK/SK?

   如果在其他功能（例如PyCharmtoolKit/VSCode登录，访问在线服务等）中使用到访问密钥AK/SK认证，获取AK/SK方式请参考**如何获取访问密钥?** 章节。

4. 如何删除已有委托列表下面的委托名称?

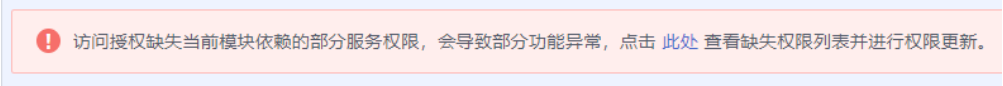
   

   需要前往统一身份认证服务IAM控制台的委托页面删除。

5. 进入ModelArts控制台的某个页面时，为什么会提示权限不足?

   图 2-8 页面提示权限不足

   可能原因是用户委托权限配置不足或模块能力升级，需要更新授权信息。根据界面操作提示追加授权即可。

2.2 创建 OBS 桶

ModelArts使用对象存储服务（Object Storage Service，简称OBS）进行数据存储以及模型的备份和快照，实现安全、高可靠和低成本的存储需求。因此，建议您在使用ModelArts之前先创建一个OBS桶，然后在OBS桶中创建文件夹用于存放数据。

OBS 简介

对象存储服务OBS是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力。对象存储服务OBS的基本组成是桶和对象。桶是OBS中存储对象的容器，每个桶都有自己的存储类别、访问权限、所属区域等属性，用户在互联网上通过桶的访问域名来定位桶。对象是OBS中数据存储的基本单位。

对ModelArts来说，OBS服务是一个数据存储中心，因为ModelArts本身目前没有数据存储的功能。AI开发过程中的输入数据、输出数据、中间缓存数据都可以在OBS桶中进行存储、读取。

因此，在使用ModelArts之前您需要创建一个OBS桶，然后在OBS桶中创建文件夹用于存放数据。

操作步骤

1. 登录OBS管理控制台，在桶列表页面右上角单击“创建桶”，创建OBS桶。例如，创建名称为“c-flowers”的OBS桶。

   说明

   创建桶的区域需要与ModelArts所在的区域一致。

   请勿开启桶加密，ModelArts不支持加密的OBS桶，会导致ModelArts读取OBS中的数据失败。

2. 在桶列表页面，单击桶名称，进入该桶的概览页面。
3. 单击左侧导航的“对象”，在对象页面单击新建文件夹，创建OBS文件夹。例如，在已创建的OBS桶“c-flowers”中新建一个文件夹“flowers”。具体请参见“新建文件夹”章节。

图 2-9 新建文件夹

3 自动学习

3.1 自动学习简介

自动学习功能介绍

ModelArts自动学习是帮助人们实现AI应用的低门槛、高灵活、零代码的定制化模型开发工具。自动学习功能根据标注数据自动设计模型、自动调参、自动训练、自动压缩和部署模型。开发者无需专业的开发基础和编码能力，只需上传数据，通过自动学习界面引导和简单操作即可完成模型训练和部署。

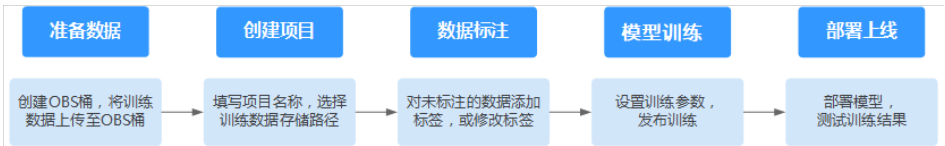
当前自动学习支持快速创建图像分类、物体检测模型的定制化开发。可广泛应用在工业、零售安防等领域。

- 图像分类：识别图片中物体的类别。
- 物体检测：识别出图片中每个物体的位置和类别。

自动学习流程介绍

使用ModelArts自动学习开发AI模型无需编写代码，您只需上传数据、创建项目、完成数据标注、发布训练、然后将训练的模型部署上线。具体流程请参见图3-1。新版自动学习中，该流程可完全由Workflow进行承载。开发者可以通过Workflow进行有向无环图（Directed Acyclic Graph，DAG）的开发，整个DAG的执行就是有序的任务执行模板，依次执行从数据标注、数据集版本发布、模型训练、模型注册到服务部署环节。。

图 3-1 自动学习操作流程



项目类型介绍

- **图像分类**

  图像分类项目，是对图像进行分类。需要添加图片并对图像进行分类标注，完成图片标注后开始模型训练，即可快速生成图像分类模型。可应用于商品的自动分

类、运输车辆种类识别和残次品的自动分类等。例如质量检查的场景，则可以上传产品图片，将图片标注“合格”、“不合格”，通过训练部署模型，实现产品的质检。

- **物体检测**

  物体检测项目，是检测图片中物体的类别与位置。需要添加图片，用合适的框标注物体作为训练集，进行训练输出模型。适用于一张图片中要识别多个物体或者物体的计数等。可应用于园区人员穿戴规范检测和物品摆放的无人巡检。

3.2 图像分类

3.2.1 准备数据

使用ModelArts自动学习构建模型时，您需要将数据上传至对象存储服务（OBS）中。

数据集要求

- 保证图片质量：不能有损坏的图片，目前支持的格式包括jpg、jpeg、bmp、png。
- 不要把明显不同的多个任务数据放在同一个数据集内。
- 每一类数据尽量多，尽量均衡。期望获得良好效果，图像分类项目中，至少有两种以上的分类，每种分类的样本不少于20张。
- 为了保证模型的预测准确度，训练样本跟真实使用场景尽量相似。
- 为保证模型的泛化能力，数据集尽量覆盖可能出现的各种场景。

数据上传至 OBS

在本文档中，采用通过OBS管理控制台将数据上传至OBS桶。

上传OBS的文件规范：

- 文件名规范：不能有+、空格、制表符。
- 如不需要提前上传训练数据，请创建一个空文件夹用于存放工程后期生成的文件。如：“/bucketName/data-cat”。
- 如需要提前上传待标注的图片，请创建一个空文件夹，然后将图片文件保存在该文件夹下，图片的目录结构如：“/bucketName/data-cat/cat.jpg”。
- 如您将已标注好的图片上传至OBS桶，请按照如下规范上传。
  - 图像分类数据集要求将标注对象和标注文件存储在同一目录，并且一一对应，例如标注对象文件名为“10.jpg”，那么标注文件的文件名应为“10.txt”。

    数据文件存储示例：

    ```
    ├─<dataset-import-path>
    │       10.jpg
    │       10.txt
    │       11.jpg
    │       11.txt
    │       12.jpg
    │       12.txt
    ```

  - 只支持JPG、JPEG、PNG、BMP格式的图片。在OBS管理控制台上传时，单张图片的大小不能超过5MB，单次上传的图片总大小不能超过8MB，数据量大时推荐使用OBS Browser+上传 。

- 标签名是由大小写字母、数字、中划线或下划线组成，且不超过32位的字符串。
- 图像分类标签“.txt”规范如下。

  一行一个标签：

  ```
  flower
  book
  ...
  ```

上传OBS操作步骤：

执行如下操作，将数据上传到OBS中，以便用于模型训练和构建。

1. 登录OBS管理控制台，创建桶。
2. 参考上传文件，将本地数据上传至OBS桶中。如果您的数据较多，推荐OBS Browser+上传数据或上传文件夹。上传的数据需满足此类型自动学习项目的数据集要求。

说明

在上传数据时，请选择非加密桶进行上传，否则会由于加密桶无法解密导致后期的训练失败。

创建数据集

数据准备完成后，需要创建相应项目支持的类型的数据集，具体操作请参考**创建数据集**。

3.2.2 创建项目

ModelArts自动学习，包括图像分类、物体检测。您可以根据业务需求选择创建合适的项目。您需要执行如下操作来创建自动学习项目。

创建项目

1. 登录ModelArts管理控制台，在左侧导航栏单击“自动学习”，进入新版自动学习页面。
2. 在您需要的自动学习项目列表中，单击“创建项目”，进入创建自动学习项目界面。

图 3-2 创建项目

3. 在创建自动学习项目页面，参考**表3-1**填写相应参数。

表 3-1 参数说明

参数	说明
“名称”	项目的名称。 ● 名称只能包含数字、字母、下划线和中划线，长度不能超过64位且不能为空。 ● 名称请以字母开头。 ● 名称不允许重复。
“描述”	对项目的简要描述。
“数据集”	可在右侧下拉框选择已有数据集，或单击“创建数据集”前往新建数据集。 ● 已有数据集：在“数据集”右侧的下拉框中选择，仅展示同类型的数据集供选择。 ● 创建数据集：前往创建数据集页面创建一个新的数据集。具体可参考**创建数据集**。
“输出路径”	选择自动学习数据输出的统一OBS路径。 **说明** “输出路径”是存储自动学习在运行过程中所有产物的路径。
“训练规格”	选择自动学习训练节点所使用的资源规格，以实际界面显示为准，将会根据不同的规格计费。 **说明** ● 若您购买了套餐包，可优先选择您对应规格的套餐包，在“配置费用”处会显示您的套餐余量，以及超出的部分如何计费，请您关注，避免造成不必要的资源浪费。

4. 单击“创建项目”，图像分类项目创建成功后页面自动跳转到“自动学习工作流”。
5. 图像分类项目的工作流，将依次运行如下节点：
   a. 数据标注：对您的数据标注情况进行确认。
   b. 数据集版本发布：将已完成标注的数据进行版本发布。
   c. 数据校验：对您的数据集的数据进行校验，是否存在数据异常。
   d. 图像分类：将发布好的数据集版本进行训练，生成对应的模型。
   e. 模型注册：将训练后的结果注册到模型管理中。
   f. 服务部署：将生成的模型部署为在线服务。

快速查找创建好的项目

在自动学习总览页，您可以通过搜索框，根据自动学习的属性类型（项目名称）快速搜索过滤到相应的工作流，可节省您的时间。

1. 登录ModelArts管理控制台，在左侧导航栏选择自动学习，进入自动学习总览页面。
2. 在自动学习列表上方的搜索框中，根据您需要的属性类型，例如，名称、状态、项目类型、当前节点、标签等，过滤出相应的工作流。

图 3-3 属性类型

3. 单击搜索框右侧的按钮，可选择自动学习的基础设置，需要的显示列。

   表格内容折行：默认为关闭状态，启用此能力可让表格内容自动折行，禁用此功能可截断文本。

   操作列：默认为开启状态，启用此能力可让操作列固定在最后一列永久可见。

   自定义显示列：默认所有显示项全部勾选，您可以根据实际需要定义您的显示列。

图 3-4 表格显示设置

4. 单击“确定”即可按照设置好的显示列进行显示。
5. 同时可支持对自动学习项目显示页进行排序，单击表头中的箭头，就可对该列进行排序。

3.2.3 数据标注

由于模型训练过程需要大量有标签的图片数据，因此在模型训练之前需对没有标签的图片添加标签。通过ModelArts您可对图片进行一键式批量添加标签，快速完成对图片的标注操作，也可以对已标注图片修改或删除标签进行重新标注。

说明

请确保数据集中已标注的图片不低于100张，否则会导致数据集校验环节不通过，影响您的模型训练。

项目创建完成后，将会自动跳转至新版自动学习页面，并开始运行。单击“数据标注”节点，当状态变为“等待操作”时，需要手动进行确认数据集中的数据标注情况，也可以对数据集中的数据进行标签的修改，数据的增加或删减。

图 3-5 数据标注节点状态

图片标注

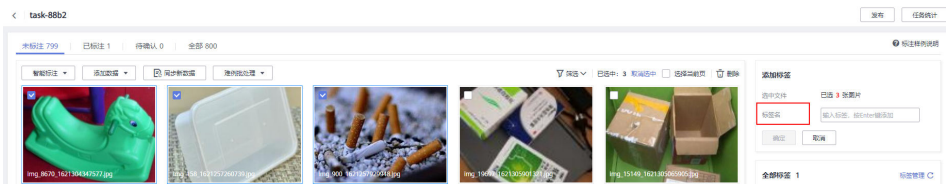
1. 在新版自动学习页面的数据标注节点单击“实例详情”按钮，前往数据标注页面。

   图 3-6 单击实例详情

2. 依次勾选待标注的图片，或勾选“选择当前页”选中该页面所有图片，在页面右侧进行图片标注。

   图 3-7 图片标注

3. 选中图片后，在页面右侧“添加标签”，输入“标签名”或从下拉列表中选择已添加的标签。单击“确定”，完成选中图片的标注操作。例如，您可以选择多张图片，按照花朵种类将图片标注为“tulips”。同样选择其他未标注分类图片，将

其标注为“sunflowers”、“roses”等。标注完成后，图片将存储至“已标注”页签下。

a. 图片标注支持多标签，即一张图片可添加多个标签。

b. 标签名是由大小写字母、数字、中划线或下划线组成。

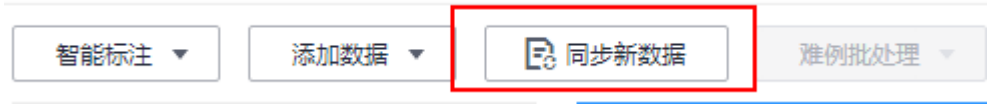
4. 当图片目录中所有图片都完成标注后，您可以在“已标注”页签下查看已完成标注的图片，或者通过右侧的“全部标签”列表，了解当前已完成的标签名称和标签数量。

同步或添加图片

在“数据标注”节点单击“实例详情”进入数据标注页面，数据标注的图片来源有两种，通过本地添加图片和同步OBS中的图片数据。

图 3-8 添加本地图片

图 3-9 同步 OBS 图片数据



- **添加数据**：您可以将本地图片快速添加到ModelArts，同时自动上传至创建项目时所选择的OBS路径中。单击“添加数据”，根据弹出的对话框的引导，输入正确的数据并添加。
- **同步新数据**：将图片数据上传至创建项目时指定的OBS目录，然后单击“同步新数据”，快速将原OBS目录中的新数据添加到ModelArts数据集。
- **删除图片**：您可以依次单击选中图片进行删除，也可以勾选“选择当前页”对该页面所有图片进行删除。

说明

所有的删除操作均不可恢复，请谨慎操作。

修改标注

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

- **基于图片修改**

  在数据标注页面，单击“已标注”页签，然后在图片列表中选中待修改的图片（选择一个或多个）。在右侧标签信息区域中对图片信息进行修改。

  - 添加标签：在“标签名”右侧文本框中，选择已有标签或输入新的标签名，然后单击，为选中图片增加标签。
  - 修改标签：在“选中文件标签”区域中，单击操作列的编辑图标，然后在文本框中输入正确的标签名，然后单击确定图标完成修改。
  - 删除标签：在“选中文件标签”区域中，单击操作列的删除该标签。

- **基于标签修改**

  在数据标注概览页，单击右侧的“标签管理”，即可显示全部标签的信息。

  - 修改标签：在需要修改的标签的“操作”列，单击“修改”，输入修改后的标签，单击“确定”即可。
  - 删除标签：选择对应的标签，单击操作列的“删除”，在弹出的“删除标签”对话框中单击“确定”即可删除对应的标签。

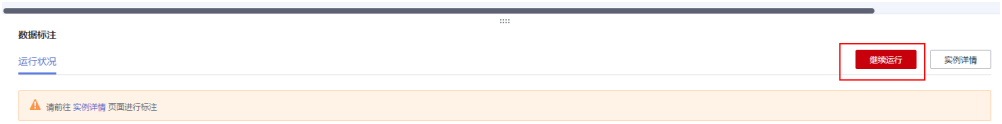
    说明

    删除后无法再恢复，请谨慎操作。

继续运行

完成数据的确认之后，返回新版自动学习的页面，在数据标注节点单击“继续运行”，工作流将会继续依次运行直到所有节点运行成功。

图 3-10 继续运行



3.2.4 模型训练

完成图片标注后，可进行模型的训练。模型训练的目的是得到满足需求的图像分类模型。请参考**前提条件**确保已标注的图片符合要求，否则数据集校验将会不通过。

前提条件

1. 请确保您的数据集中的已标注的图片不低于100张。
2. 请确保您的数据集中至少存在2种以上的图片分类，且每种分类的图片不少于5张。

操作步骤

1. 参考**数据标注**章节，确保您的数据已全部完成标注。

图 3-11 完成数据标注

2. 在新版自动学习页面，单击数据标注节点的“继续运行”按钮，然后等待工作流按顺序进入训练节点即可。
3. 模型将会自动进入训练，无需人工介入，训练时间相对较长，建议您耐心等待。如果关闭或退出此页面，系统仍然在执行训练操作。
4. 在“图像分类”节点中，待训练状态由“运行中”变为“运行成功”，即完成了模型的自动训练。
5. 训练完成后，您可以单击“图像分类”节点上方的按钮，查看相关指标信息，如“准确率”、“评估结果”等。评估结果参数说明请参见表3-2。

表 3-2 评估结果参数说明

参数名称	参数含义	说明
recall	召回率	被用户标注为某个分类的所有样本中，模型正确预测为该分类的样本比率，反映模型对正样本的识别能力。
precision	精确率	被模型预测为某个分类的所有样本中，模型正确预测的样本比率，反映模型对负样本的区分能力。
accuracy	准确率	所有样本中，模型正确预测的样本比率，反映模型对样本整体的识别能力。
f1	F1值	F1值是模型精确率和召回率的加权调和平均，用于评价模型的好坏，当F1较高时说明模型效果较好。

说明

同一个自动学习项目可以训练多次，每次训练会注册一个新的AI应用版本。如第一次训练版本号为“0.0.1”，下一个版本为“0.0.2”。基于训练版本可以对训练模型进行管理。当训练的模型达到目标后，再执行部署上线的操作。

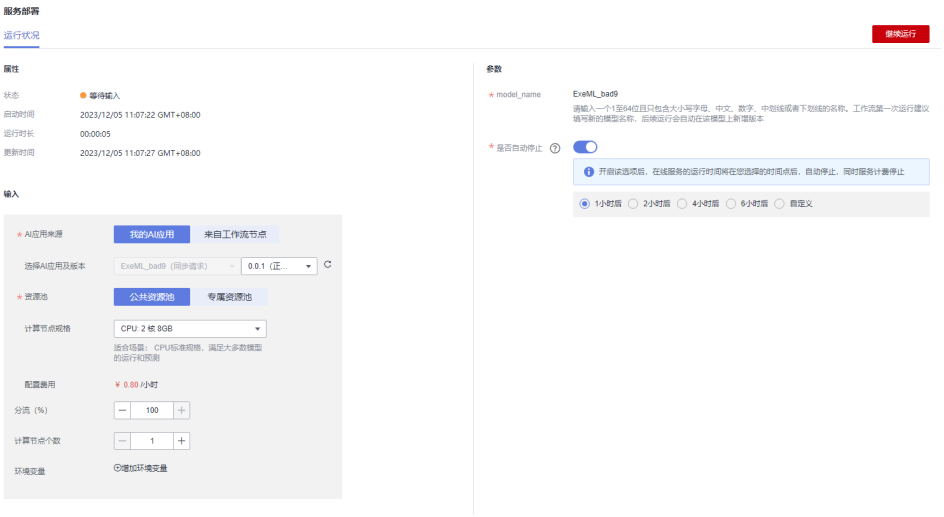
3.2.5 部署上线

部署上线

部署上线操作即将模型部署为在线服务，并且提供在线的测试UI与监控能力。完成模型训练后，可选择准确率理想且训练状态为“运行成功”的版本部署上线。具体操作步骤如下。

1. 在“运行节点”页面中，待服务部署节点的状态变为“等待输入”时，双击“服务部署”进入配置详情页，完成资源的参数配置操作。
2. 在服务部署页面，选择部署上线使用的资源规格。

图 3-12 资源规格



- AI应用来源：默认为生成的AI应用。
- 选择AI应用及版本：自动匹配当前使用的AI应用版本，支持选择版本。
- 资源池：默认公共资源池。
- 分流：默认为100，输入值必须是0-100之间。
- 计算节点规格：请根据界面显示的列表，选择可用的规格，置灰的规格表示当前环境无法使用。如果公共资源池下规格为空数据，表示当前环境无公共资源。建议使用专属资源池，或者联系系统管理员创建公共资源池。
- 计算节点个数：默认为1，输入值必须是1-5之间的整数。
- 是否自动停止：启用该参数并设置时间后，服务将在指定时间后自动停止。如果不启用此参数，在线服务将一直运行。默认开启自动停止功能，且默认值为“1小时后”。

  目前支持设置为“1小时后”、“2小时后”、“4小时后”、“6小时后”、“自定义”。如果选择“自定义”的模式，可在右侧输入框中输入1~24范围内的任意整数。

  说明

  若您购买了套餐包，计算节点规格可选择您的套餐包，同时在“配置费用”页签还可查看您的套餐包余量以及超出部分的计费方式，请您务必关注，避免造成不必要的资源浪费。

3. 完成资源配置后，单击“继续运行”，服务部署节点将继续运行，直至状态变为“运行成功”，至此，已将AI应用部署为在线服务。

服务测试

- 服务部署节点运行成功后，单击“实例详情”可跳转至对应的在线服务详情页面。单击“预测”页签，进行服务测试。

  图 3-13 服务测试

- 您也可以在“部署上线>在线服务”页面，选择对应的服务类型，进入“在线服务”页面，单击目标服务“操作”列的“预测”，进行服务测试，测试方法和下方陈述操作步骤一致。具体操作请参见**测试服务**。
- 您也可以通过调用代码对服务进行测试，根据部署服务类型的不同，具体操作详情参见访问在线服务。
- 下面的测试，是您在自动学习图像分类项目页面将模型部署上线之后进行服务测试的操作步骤。

  a. 模型部署完成后，“在服务部署”节点，单击“实例详情”按钮，进入服务预测界面，在“预测”页签单击“上传”，选择本地图片进行测试。

  b. 单击“预测”进行测试，预测完成后，右侧“预测结果”区域输出标签名称“sunflowers”和检测的评分。如模型准确率不满足预期，可在“数据标注”页签中添加图片并进行标注，重新进行模型训练及部署上线。预测结果中的参数说明请参见**表3-3**。如果您对模型预测结果满意，可根据界面提示调用接口访问在线服务，操作指导请参见访问在线服务。

  目前只支持jpg、jpeg、bmp、png格式的图片。

图 3-14 预测结果

表 3-3 预测结果中的参数说明

参数	说明
predicted_label	表示图片预测的标签。
scores	表示Top5标签的预测置信度。

说明

- 由于“运行中”的在线服务将持续耗费资源，如果不需再使用此在线服务，建议在“在线服务”的操作列单击“更多>停止”。如果需要继续使用此服务，可单击“启动”恢复。

3.3 物体检测

3.3.1 准备数据

使用ModelArts自动学习构建模型时，您需要将数据上传至对象存储服务（OBS）中。

数据集要求

- 保证图片质量：不能有损坏的图片；目前支持的格式包括jpg、jpeg、bmp、png。
- 不要把明显不同的多个任务数据放在同一个数据集内。
- 为了保证模型的预测准确度，训练样本跟真实使用场景尽量相似。
- 为保证模型的泛化能力，数据集尽量覆盖可能出现的各种场景。
- 物体检测数据集中，如果标注框坐标超过图片，将无法识别该图片为已标注图片。

数据上传至 OBS

在本文档中，采用通过OBS管理控制台将数据上传至OBS桶。

上传OBS的文件规范：

- 文件名规范，不能有+、空格、制表符。
- 如不需要提前上传训练数据，请创建一个空文件夹用于存放工程后期生成的文件。如：“/bucketName/data-cat”。
- 如需要提前上传待标注的图片，请创建一个空文件夹，然后将图片文件保存在该文件夹下，图片的目录结构如：“/bucketName/data-cat/cat.jpg”。
- 如您将已标注好的图片上传至OBS桶，请按照如下规范上传。
  - 物体检测数据集要求用户将标注对象和标注文件存储在同一目录，并且一一对应。例如标注对象文件名为“IMG_20180919_114745.jpg”，那么标注文件的文件名应为“IMG_20180919_114745.xml”。

    物体检测的标注文件需要满足PASCAL VOC格式，格式详细说明请参见 **表3-4**。

    数据存储示例：

    ```
    ├─<dataset-import-path>
    │       IMG_20180919_114732.jpg
    │       IMG_20180919_114732.xml
    │       IMG_20180919_114745.jpg
    │       IMG_20180919_114745.xml
    │       IMG_20180919_114945.jpg
    │       IMG_20180919_114945.xml
    ```
  - 只支持JPG、JPEG、PNG、BMP格式的图片，在OBS管理控制台上传时，单张图片的大小不能超过5MB，单次上传的图片总大小不能超过8MB，数据量大时推荐使用OBS Browser+上传 。
  - 标签名是由大小写字母、数字、中划线或下划线组成，且不超过32位的字符串。

表 3-4 PASCAL VOC 格式说明

字段	是否必选	说明
folder	是	表示数据源所在目录。
filename	是	被标注文件的文件名。
size	是	表示图像的像素信息。 ● width：必选字段，图片的宽度。 ● height：必选字段，图片的高度。 ● depth：必选字段，图片的通道数。
segmented	是	表示是否用于分割。

字段	是否必选	说明
object	是	表示物体检测信息，多个物体标注会有多个object体。 ● name：必选字段，标注内容的类别。 ● pose：必选字段，标注内容的拍摄角度。 ● truncated：必选字段，标注内容是否被截断（0表示完整）。 ● occluded：必选字段，标注内容是否被遮挡（0表示未遮挡） ● difficult：必选字段，标注目标是否难以识别（0表示容易识别）。 ● confidence：可选字段，标注目标的置信度，取值范围0-1之间。 ● bndbox：必选字段，标注框的类型，标注信息请参见 表3-5 。

表 3-5 标注框类型描述

type	形状	标注信息
bndbox	矩形框	左上和右下两个点坐标。 <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymax>200<ymax>

标注文件示例：

```
<annotation>
  <folder>test_data</folder>
  <filename>260730932.jpg</filename>
  <size>
     <width>767</width>
     <height>959</height>
     <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
     <name>bag</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <bndbox>
        <xmin>108</xmin>
        <ymin>101</ymin>
        <xmax>251</xmax>
        <ymax>238</ymax>
     </bndbox>
```

```
  </object>
</annotation>
```

上传OBS的操作步骤：

执行如下操作，将数据导入到您的数据集中，以便用于模型训练和构建。

1. 登录OBS管理控制台，创建桶。
2. 参考上传文件，将本地数据上传至OBS桶中。如果您的数据较多，推荐OBS Browser+上传数据或上传文件夹。上传的数据需满足此类型自动学习项目的数据集要求。

说明

- 在上传数据时，请选择非加密桶进行上传，否则会由于加密桶无法解密导致后期的训练失败。
- 用于训练的图片，至少有1种以上的分类，每种分类的图片数不少50张。

创建数据集

数据准备完成后，需要创建相应项目支持的类型的数据集，具体操作请参考**创建数据集**。

3.3.2 创建项目

ModelArts自动学习，包括图像分类、物体检测。您可以根据业务需求选择创建合适的项目。您需要执行如下操作来创建自动学习项目。

创建项目

1. 登录ModelArts管理控制台，在左侧导航栏单击“自动学习”，进入新版自动学习页面。
2. 在您需要的自动学习项目列表中，单击“创建项目”，进入创建自动学习项目界面。

图 3-15 创建项目

3. 在创建自动学习项目页面，参考**表3-6**填写相应参数。

表 3-6 参数说明

参数	说明
“名称”	项目的名称。 ● 名称只能包含数字、字母、下划线和中划线，长度不能超过64位且不能为空。 ● 名称请以字母开头。 ● 名称不允许重复。
“描述”	对项目的简要描述。
“数据集”	可在右侧下拉框选择已有数据集，或单击“创建数据集”前往新建数据集。 ● 已有数据集：在“数据集”右侧的下拉框中选择，仅展示同类型的数据集供选择。 ● 创建数据集：前往创建数据集页面创建一个新的数据集。具体可参考。
“输出路径”	选择自动学习数据输出的统一OBS路径。 **说明** “输出路径”是存储自动学习在运行过程中所有产物的路径。
“训练规格”	选择自动学习训练节点所使用的资源规格，以实际界面显示为准，将会根据不同的规格计费。 **说明** ● 若您购买了套餐包，可优先选择您对应规格的套餐包，在“配置费用”处会显示您的套餐余量，以及超出的部分如何计费，请您关注，避免造成不必要的资源浪费。

4. 单击“创建项目”，图像分类项目创建成功后页面自动跳转到“自动学习工作流”。
5. 物体检测项目的工作流，将依次运行如下节点：
   a. 数据标注：对您的数据进行标注情况确认。
   b. 数据集版本发布：将已完成标注的数据进行版本发布。
   c. 数据校验：对您的数据集的数据进行校验，是否存在数据异常。
   d. 物体检测：将发布好的数据集版本进行训练，生成对应的模型。
   e. 模型注册：将训练后的结果注册到模型管理中。
   f. 服务部署：将生成的模型部署为在线服务。

快速查找创建好的项目

在自动学习总览页，您可以通过搜索框，根据自动学习的属性类型（项目名称）快速搜索过滤到相应的工作流，可节省您的时间。

1. 登录ModelArts管理控制台，在左侧导航栏选择自动学习，进入自动学习总览页面。
2. 在自动学习列表上方的搜索框中，根据您需要的属性类型，例如，名称、状态、项目类型、当前节点、标签等，过滤出相应的工作流。

图 3-16 属性类型

3. 单击搜索框右侧的按钮，可选择自动学习的基础设置，需要的显示列。

   表格内容折行：默认为关闭状态，启用此能力可让表格内容自动折行，禁用此功能可截断文本。

   操作列：默认为开启状态，启用此能力可让操作列固定在最后一列永久可见。

   自定义显示列：默认所有显示项全部勾选，您可以根据实际需要定义您的显示列。

图 3-17 表格显示设置

4. 单击“确定”即可按照设置好的显示列进行显示。
5. 同时可支持对自动学习项目显示页进行排序，单击表头中的箭头，就可对该列进行排序。

3.3.3 数据标注

物体检测之前，首先需考虑如何设计标签，标签设计需要对应所检测图片的明显特征，并且选择的标签比较容易识别（画面主体物与背景区分度较高），每个标签就是对所检测图片期望识别的全部结果。物体的标签设计完成之后，基于设计好的标签准备该图片的数据，每种需识别出的标签，建议应在所有图片个数相加超过100张，若某些图片的标签具有相似性，则需要更多的图片。用于训练的图片，至少有1种以上的分类，每种分类的图片数不少50张。

- 标注时，类内方差尽量要小。即相同类别的标注，尽量近似；不同类别的标注，尽量保持差距较大。
- 标记的每个标签尽量和背景有较大的区分度。
- 物体检测标注，需要保证目标框内物体的完整性；针对图片中存在多个物体的情形，做到不重标、不漏标。

项目创建完成后，将会自动跳转至新版自动学习页面，并开始运行，当数据标注节点的状态变为“等待操作”时，需要手动进行确认数据集中的数据标注情况，也可以对数据集中的数据进行标签的修改，数据的增加或删减。

图 3-18 数据标注节点状态

图片标注

1. 在新版自动学习页面单击“实例详情”按钮，前往数据标注页面。单击任意一张图片，进入图片标注界面。

2. 用鼠标框选图片中的物体所在区域，然后在弹出的对话框中选择标签颜色，输入标签名称，例如此示例中的“yunbao”，按“Enter”键完成此标签的添加。标注完成后，左侧图片目录中此图片的状态将显示为“已标注”。

   数据标注的更多说明：

   - 您可以在图片上方或下方单击左右切换键，或者按键盘的左右方向键，选择其他图片，重复上述操作继续进行图片标注。如果一张图片有多个物体，您可以标注多处。
   - 同一个物体检测自动学习项目内，可以增加多个标签，且标签可选择不同颜色，方便识别。使用鼠标完成物体框选后，在弹出的对话框中，选择新的颜色，输入新的标签名称，即可添加一个新的标签。
   - 自动学习项目中，物体检测仅支持矩形标注框。在“数据管理”功能中，物体检测类型的数据集，支持更多类型的标注框。
   - 在标注窗口中，您可以滚动鼠标，放大或缩小图片，方便您快速定位到物体位置。

说明

“物体检测”类型的数据集，在标注时，支持在一张图片中添加多个标注框以及标签。需注意的是，标注框不能超过图片边缘。

3. 当图片目录中所有图片都完成标注后，返回“自动学习工作流”页面，单击“继续运行”按钮，工作流将会自动发布数据标注版本，并进行下一步训练步骤。

同步或添加图片

在“数据标注”节点单击“实例详情”进入数据标注页面，数据标注的图片来源有两种，通过本地添加图片和同步OBS中的图片数据。

图 3-19 添加本地图片

图 3-20 同步 OBS 图片数据

- **添加数据**：您可以将本地图片快速添加到ModelArts，同时自动上传至创建项目时所选择的OBS路径中。单击“添加数据”，根据弹出的对话框的引导，输入正确的数据并添加。
- **同步新数据**：将图片数据上传至创建项目时指定的OBS目录，然后单击“同步新数据”，快速将原OBS目录中的新数据添加到ModelArts数据集。
- **删除图片**：您可以依次单击选中图片进行删除，也可以勾选“选择当前页”对该页面所有图片进行删除。

说明

所有的删除操作均不可恢复，请谨慎操作。

修改标注

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

- **基于图片修改**

在数据集详情页面，单击“已标注”页签，然后在图片列表中选中待修改的图片，在右侧“标注”区域中对图片信息进行修改。

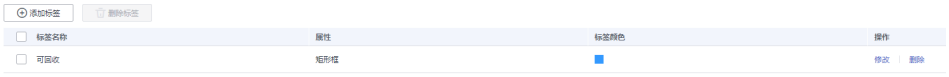
- 修改标签：“标注”区域中，单击编辑按钮，在文本框中输入正确的标签名，然后单击确定按钮完成修改。标签颜色不支持修改。
- 删除标签：在“标注”区域中，单击删除按钮，即可删除此图片中的标签。

  标签删除后，单击页面左上角的项目名称离开标注页面。该图片会重新回到“未标注”页签。

- **基于标签修改**

  在数据标注作业概览页，单击右侧的“标签管理”，进入标签管理页面，标签管理页展示所有标签信息。

  图 3-21 标签管理页

  - 修改标签：单击操作列的“修改”按钮，然后在弹出的对话框中输入修改后的标签名，然后单击“确定”完成修改。修改后，之前添加了此标签的图片，都将被标注为新的标签名称。
  - 删除标签：单击操作列的“删除”按钮，在弹出的对话框中，根据界面提示选择删除对象，然后单击“确定”。

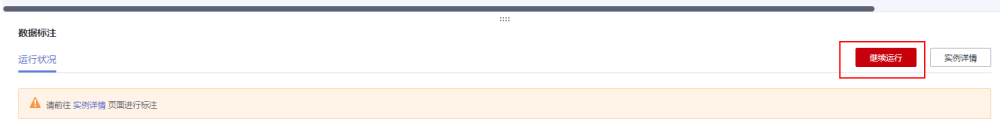
    说明

    删除后的标签无法再恢复，请谨慎操作。

继续运行

完成数据的确认之后，返回新版自动学习的页面，在数据标注节点单击“继续运行”，工作流将会继续依次运行直到所有节点运行成功。

图 3-22 继续运行



3.3.4 模型训练

自动学习物体检测项目，在图片标注完成后，通过模型训练得到合适的模型版本。

操作步骤

1. 在新版自动学习页面，单击项目名称进入运行总览页面，单击“数据标注”节点的“实例详情”进入数据标注页面，完成数据标注。

图 3-23 完成数据标注

2. 返回新版自动学习页面，单击数据标注节点的“继续运行”，然后等待工作流按顺序进入训练节点。
3. 模型将会自动进入训练，无需人工介入，训练时间相对较长，建议您耐心等待。如果关闭或退出此页面，系统仍然在执行训练操作。
4. 在“物体检测”节点中，待训练状态由“运行中”变为“运行成功”，即完成模型的自动训练。
5. 训练完成后，您可以单击物体检测节点上方的  按钮，查看相关指标信息，如“准确率”、“评估结果”等。评估结果参数说明请参见表3-7。

表 3-7 评估结果参数说明

参数	说明
recall：召回率	被用户标注为某个分类的所有样本中，模型正确预测为该分类的样本比率，反映模型对正样本的识别能力。
precision：精确率	被模型预测为某个分类的所有样本中，模型正确预测的样本比率，反映模型对负样本的区分能力。
accuracy：准确率	所有样本中，模型正确预测的样本比率，反映模型对样本整体的识别能力。
f1：F1值	F1值是模型精确率和召回率的加权调和平均，用于评价模型的好坏，当F1较高时说明模型效果较好。

说明

同一个自动学习项目可以训练多次，每次训练会注册一个新的AI应用一个版本。如第一次训练版本号为“0.0.1”，下一个版本为“0.0.2”。基于训练版本可以对训练模型进行管理。当训练的模型达到目标后，再执行部署上线的操作。

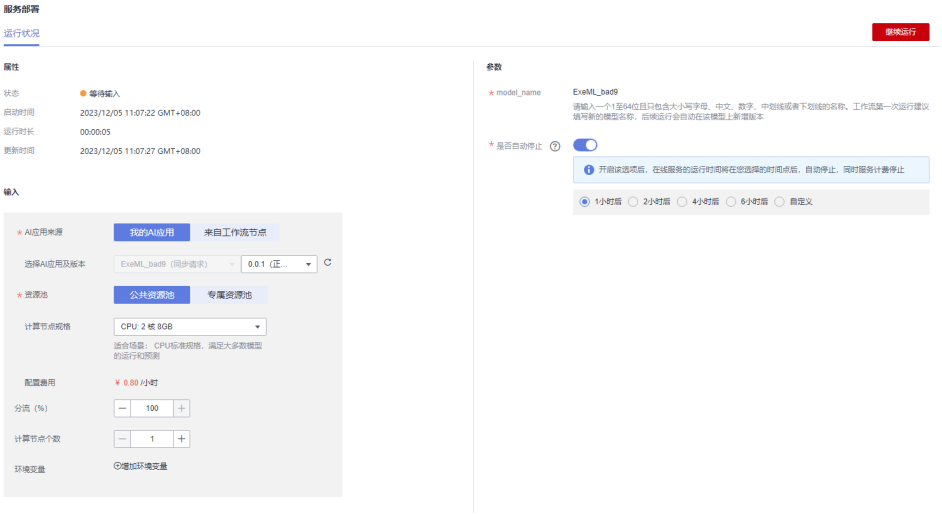
3.3.5 部署上线

部署上线

部署上线操作即将模型部署为在线服务，并且提供在线的测试UI与监控能力。完成模型训练后，可选择准确率理想且训练状态为“运行成功”的版本部署上线。具体操作步骤如下。

1. 在“运行节点”页面中，待服务部署节点的状态变为“等待输入”时，双击“服务部署”进入配置详情页，完成资源的参数配置操作。
2. 在服务部署页面，选择部署上线使用的资源规格。

图 3-24 资源规格



- AI应用来源：默认为生成的AI应用。
- 选择AI应用及版本：自动匹配当前使用的AI应用版本，支持选择版本。
- 资源池：默认公共资源池。
- 分流：默认为100，输入值必须是0-100之间。
- 计算节点规格：请根据界面显示的列表，选择可用的规格，置灰的规格表示当前环境无法使用。如果公共资源池下规格为空数据，表示当前环境无公共资源。建议使用专属资源池，或者联系系统管理员创建公共资源池。
- 计算节点个数：默认为1，输入值必须是1-5之间的整数。
- 是否自动停止：启用该参数并设置时间后，服务将在指定时间后自动停止。如果不启用此参数，在线服务将一直运行。默认开启自动停止功能，且默认值为“1小时后”。

  目前支持设置为“1小时后”、“2小时后”、“4小时后”、“6小时后”、“自定义”。如果选择“自定义”的模式，可在右侧输入框中输入1~24范围内的任意整数。

  说明

  若您购买了套餐包，计算节点规格可选择您的套餐包，同时在“配置费用”页签还可查看您的套餐包余量以及超出部分的计费方式，请您务必关注，避免造成不必要的资源浪费。

3. 完成资源配置后，单击“继续运行”，服务部署节点将继续运行，直至状态变为“运行成功”，至此，已将AI应用部署为在线服务。

服务测试

- 服务部署节点运行成功后，单击“实例详情”可跳转至对应的在线服务详情页面。单击“预测”页签，进行服务测试。

图 3-25 服务测试

- 您也可以在“部署上线>在线服务”页面，选择对应的服务类型，进入“在线服务”页面，单击目标服务“操作”列的“预测”，进行服务测试，测试方法和下方陈述操作步骤一致。具体操作请参见**测试服务**。
- 您也可以通过调用代码对服务进行测试，根据部署服务类型的不同，具体操作详情参见访问在线服务。
- 下面的测试，是您在自动学习物体检测项目页面将模型部署上线之后进行服务测试的操作步骤。

  a. 模型部署完成后，“服务部署”节点，单击“实例详情”按钮，进入服务预测界面，在“预测”页签单击“上传”，选择本地图片进行测试。

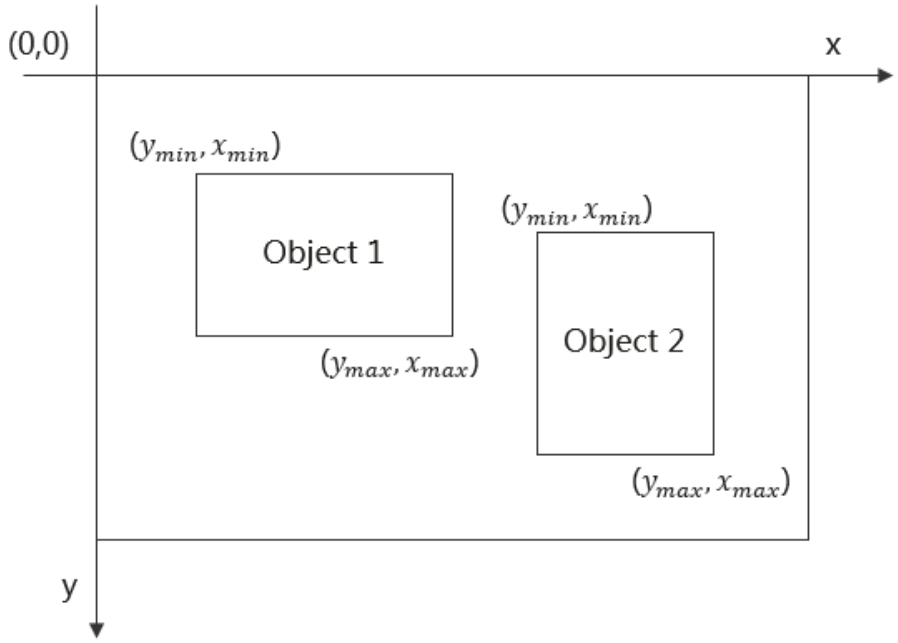
  b. 单击“预测”进行测试，预测完成后，右侧“预测结果”区域输出结果。如模型准确率不满足预期，可在“数据标注”页签中添加图片并进行标注，重新进行模型训练及部署上线。预测结果中的参数说明请参见**表3-8**。如果您对模型预测结果满意，可根据界面提示调用接口访问在线服务，操作指导请参见访问在线服务。

  目前只支持jpg、jpeg、bmp、png格式的图片。

表 3-8 预测结果中的参数说明

参数	说明
detection_classes	每个检测框的标签。
detection_boxes	每个检测框的四点坐标（y_min,x_min,y_max,x_max），如**图3-26**所示。
detection_scores	每个检测框的置信度。

图 3-26 检测框的四点坐标示意图



说明

- 由于“运行中”的在线服务将持续耗费资源，如果不需再使用此在线服务，建议在版本管理区域，单击“停止”，即可停止在线服务的部署。如果需要继续使用此服务，可单击“启动”恢复。

3.4 预测分析

3.4.1 准备数据

使用ModelArts自动学习构建预测分析模型时，您需要将数据上传至对象存储服务（OBS）中。

数据集要求

预测分析项目中需要使用到的数据集为表格数据集，数据格式支持csv格式。表格数据集的具体介绍请参见**表格数据集简介**。

说明

将原始.xlsx格式的数据转换为.csv格式的数据的方法如下：

将原始表格数据（.xlsx）另存。单击“文件>另存为”，选择本地地址后，下拉选择“保存类型”为“CSV (逗号分隔)(*.csv)”单击“保存”，在弹窗中，单击“确定”后就可以将.xlsx格式数据集转换为.csv格式。

表格数据集对训练数据的要求：

- 训练数据列数一致，总数据量不少于100条不同数据（有一个特征取值不同，即视为不同数据）。

- 训练数据列内容不能有时间戳格式（如：yy-mm-dd、yyyy-mm-dd等）的数据。
- 如果某一列的取值只有一种，会被视为无效列。请确保标签列的取值至少有两个且无数据缺失。

  说明

  标签列指的是在训练任务中被指定为训练目标的列，即最终通过该数据集训练得到模型时的输出（预测项）。
- 除标签列外数据集中至少还应包含两个有效特征列（列的取值至少有两个且数据缺失比例低于10%）。
- 当前由于特征筛选算法限制，预测数据列建议放在数据集最后一列，否则可能导致训练失败。

表格数据集示例：

以银行存款预测数据集为例：根据预测人的年龄、工作类型、婚姻状况、文化程度、是否有房贷和是否有个人贷款。

表 3-9 数据源的具体字段及意义

字段名	含义	类型	描述
attr_1	年龄	Int	表示客户的年龄。
attr_2	职业	String	表示客户所从事的职业。
attr_3	婚姻情况	String	表示客户是否结婚或已离异。
attr_4	教育情况	String	表示客户受教育的程度。
attr_5	房产情况	String	表示客户名下是否有房产。
attr_6	贷款情况	String	表示客户名下是否有贷款。
attr_7	存款情况	String	表示客户名下是否有存款。

表 3-10 数据集样本数据

attr_1	attr_2	attr_3	attr_4	attr_5	attr_6	attr_7
31	blue-collar	married	secondary	yes	no	no
41	management	married	tertiary	yes	yes	no
38	technician	single	secondary	yes	no	no
39	technician	single	secondary	yes	no	yes
39	blue-collar	married	secondary	yes	no	no

attr_1	attr_2	attr_3	attr_4	attr_5	attr_6	attr_7
39	services	single	unknown	yes	no	no

数据上传至 OBS

在本文档中，采用通过OBS管理控制台将数据上传至OBS桶。

上传OBS的文件规范：

预测分析项目的OBS数据路径需符合以下规则：

- 输入数据的OBS路径应指向数据文件，且文件不能直接放在OBS桶的根目录下，应该存放在OBS桶的文件夹内。如：“/obs-xxx/data/input.csv”。
- 输入数据的格式必须为csv格式，有效数据行数必须大于100行。列数必须小于200列，数据总大小不能超过100MB。

上传OBS操作步骤：

执行如下操作，将数据导入到您的数据集中，以便用于模型训练和构建。

1. 登录OBS管理控制台，创建桶。
2. 参考上传文件，将本地数据上传至OBS桶中。如果您的数据较多，推荐OBS Browser+上传数据或上传文件夹。上传的数据需满足此类型自动学习项目的数据集要求。

说明

在上传数据时，请选择非加密桶进行上传，否则会由于加密桶无法解密导致后期的训练失败。

创建数据集

数据准备完成后，需要创建预测分析项目支持的类型的表格数据集，具体操作请参考**创建数据集**。

常见问题

使用从OBS选择的数据创建表格数据集如何处理Schema信息?

Schema信息代表表格的列名和对应类型，需要跟导入数据的列数保持一致。

- 若您的原始表格中已包含表头，需要开启“导入是否包含表头”开关，系统会导入文件的第一行（表头）作为列名，无需再手动修改Schema信息。
- 若您的原始表格中没有表头，需关闭“导入是否包含表头”开关，从OBS选择数据后，Schema信息的列名默认为表格中的第一行数据，请更改Schema信息中的“列名”为attr_1、attr_2、……、attr_n，其中attr_n为最后一列，代表预测列。

3.4.2 创建项目

ModelArts自动学习，包括图像分类、物体检测、。您可以根据业务需求选择创建合适的项目。您需要执行如下操作来创建自动学习项目。

创建项目

1. 登录ModelArts管理控制台，在左侧导航栏单击“自动学习”，进入新版自动学习页面。
2. 在您需要的自动学习项目列表中。例如选择预测分析项目，单击“创建项目”，进入创建自动学习项目界面。

图 3-27 创建项目（1）

3. 在创建自动学习项目页面，参考**表3-11**填写相应参数。

图 3-28 创建项目（2）

表 3-11 参数说明

参数	说明
“名称”	项目的名称。 ● 名称只能包含数字、字母、下划线和中划线，长度不能超过64位且不能为空。 ● 名称请以字母开头。 ● 名称不允许重复。
“描述”	对项目的简要描述。
“数据集”	可在右侧下拉框选择已有数据集，或单击“创建数据集”前往新建数据集。 ● 已有数据集：在“数据集”右侧的下拉框中选择，仅展示同类型的数据集供选择。 ● 创建数据集：前往创建数据集页面创建一个新的数据集。具体可参考**创建数据集**。
“标签列”	可自行选择您需要预测的列名。 标签列是预测模型的输出。模型训练步骤将使用全部信息训练预测模型，该模型以其他列的数据为输入，以标签列的预测值为输出。部署上线步骤将使用预测模型发布在线预测服务。
“输出路径”	选择自动学习数据输出的统一OBS路径。 **说明** “输出路径”是存储自动学习在运行过程中所有产物的路径。
“训练规格”	选择自动学习训练节点所使用的资源规格，以实际界面显示为准，将会根据不同的规格计费。 **说明** ● 若您购买了套餐包，可优先选择您对应规格的套餐包，在“配置费用”处会显示您的套餐余量，以及超出的部分如何计费，请您关注，避免造成不必要的资源浪费。

4. 单击“创建项目”，预测分析项目创建成功后页面自动跳转到“自动学习工作流”。
5. 预测分析项目的工作流，将依次运行如下节点：
   a. 数据集版本发布：将已完成确认的数据进行版本发布。
   b. 数据校验：对您的数据集的数据进行校验，是否存在数据异常。
   c. 预测分析：将发布好的数据集版本进行训练，生成对应的模型。
   d. 模型注册：将训练后的结果注册到模型管理中。
   e. 服务部署：将生成的模型部署为在线服务。

快速查找创建好的项目

在自动学习总览页，您可以通过搜索框，根据自动学习的属性类型（项目名称）快速搜索过滤到相应的工作流，可节省您的时间。

1. 登录ModelArts管理控制台，在左侧导航栏选择自动学习，进入自动学习总览页面。

2. 在自动学习列表上方的搜索框中，根据您需要的属性类型，例如，名称、状态、项目类型、当前节点、标签等，过滤出相应的工作流。

图 3-29 属性类型

3. 单击搜索框右侧的  按钮，可选择自动学习的基础设置，需要的显示列。

表格内容折行：默认为关闭状态，启用此能力可让表格内容自动折行，禁用此功能可截断文本。

操作列：默认为开启状态，启用此能力可让操作列固定在最后一列永久可见。

自定义显示列：默认所有显示项全部勾选，您可以根据实际需要定义您的显示列。

图 3-30 表格显示设置

4. 单击“确定”即可按照设置好的显示列进行显示。

5. 同时可支持对自动学习项目显示页进行排序，单击表头中的箭头  ，就可对该列进行排序。

3.4.3 模型训练

创建自动学习后，将会进行模型的训练，得到预测分析的模型。部署上线步骤将使用预测模型发布在线预测服务。

操作步骤

1. 在新版自动学习页面，单击创建成功的项目名称，查看当前工作流的执行情况。
2. 在“预测分析”节点中，待节点状态由“运行中”变为“运行成功”，即完成了模型的自动训练。
3. 训练完成后，您可以在预测分析节点中单击  查看训练详情，如“标签列”和“标签列数据类型”、“准确率”、“评估结果”等。

   该示例为二分类的离散型数值，评估效果参数说明请参见**表3-12**。

   不同类型标签列数据产生的评估结果说明请参见**评估结果说明**。

   说明

   同一个自动学习项目可以训练多次，每次训练会注册一个新的AI应用一个版本。如第一次训练版本号为“0.0.1”，下一个版本为“0.0.2”。基于训练版本可以对训练模型进行管理。当训练的模型达到目标后，再执行部署上线的操作。

评估结果说明

根据训练数据类的不同评估结果会包含不同的指标。

- 离散值评估结果

  包含评估指标为召回率（Recall）、精确率（Precision）、准确率（Accuracy）与F1值（F1 Score）。下表为具体说明：

  表 3-12 离散值评估结果包含指标说明

参数	说明
recall：召回率	被用户标注为某个分类的所有样本中，模型正确预测为该分类的样本比率，反映模型对正样本的识别能力。
precision：精确率	被模型预测为某个分类的所有样本中，模型正确预测的样本比率，反映模型对负样本的区分能力。
accuracy：准确率	所有样本中，模型正确预测的样本比率，反映模型对样本整体的识别能力。
f1：F1值	F1值是模型精确率和召回率的加权调和平均，用于评价模型的好坏，当F1较高时说明模型效果较好。

- 连续数值评估结果

  包含评估指标为平均绝对误差（Mean Absolute Error）、均方误差（Mean Squared Error）与均方根误差（Root Mean Squared Error）。三个误差值能够

表征真实值和预测值之间的差距。在多次建模的过程中，每一次建模结果都会产生一组误差值，评判一个模型好坏的方法就是看这三个误差值是否变小或者变大，误差值越小表示模型越好。

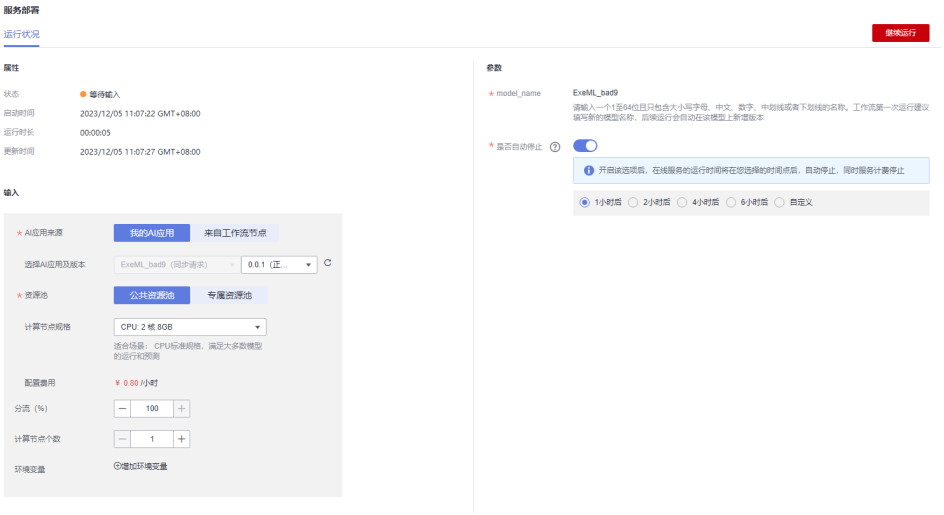
3.4.4 部署上线

部署上线

部署上线操作即将模型部署为在线服务，并且提供在线的测试UI与监控能力。完成模型训练后，可选择准确率理想且训练状态为“运行成功”的版本部署上线。具体操作步骤如下。

1. 在“运行节点”页面中，待训练状态变为“等待输入”，双击“服务部署”节点，完成相关参数配置。
2. 在服务部署页面，选择部署上线使用的资源规格。

图 3-31 资源规格



- AI应用来源：默认为生成的AI应用。
- 选择AI应用及版本：自动匹配当前使用的AI应用版本，支持选择版本。
- 资源池：默认公共资源池。
- 分流：默认为100，输入值必须是0-100之间。
- 计算节点规格：请根据界面显示的列表，选择可用的规格，置灰的规格表示当前环境无法使用。如果公共资源池下规格为空数据，表示当前环境无公共资源。建议使用专属资源池，或者联系系统管理员创建公共资源池。
- 计算节点个数：默认为1，输入值必须是1-5之间的整数。
- 是否自动停止：启用该参数并设置时间后，服务将在指定时间后自动停止。如果不启用此参数，在线服务将一直运行。默认开启自动停止功能，且默认值为“1小时后”。

  目前支持设置为“1小时后”、“2小时后”、“4小时后”、“6小时后”、“自定义”。如果选择“自定义”的模式，可在右侧输入框中输入1~24范围内的任意整数。

> **说明**
>
> 若您购买了套餐包，计算节点规格可选择您的套餐包，同时在“配置费用”页签还可查看您的套餐包余量以及超出部分的计费方式，请您务必关注，避免造成不必要的资源浪费。

3. 完成资源配置后，单击“继续运行”，在弹框中确认继续运行后，服务部署节点将继续运行，直至状态变为“运行成功”，至此，已将AI应用部署为在线服务。

服务测试

- 服务部署节点运行成功后，单击“实例详情”可跳转至对应的在线服务详情页面。单击“预测”页签，进行服务测试。

图 3-32 服务测试

- 您也可以在“部署上线>在线服务”页面，选择对应的服务类型，进入“在线服务”页面，单击目标服务“操作”列的“预测”，进行服务测试，测试方法和下方陈述操作步骤一致。具体操作请参见测试服务。
- 您也可以通过调用代码对服务进行测试，根据部署服务类型的不同，具体操作详情参见访问在线服务。
- 下面的测试，是您在自动学习预测分析项目页面将模型部署上线之后进行服务测试的操作步骤。

  a. 模型部署完成后，您可输入代码进行测试。在“自动学习”页面，在服务部署节点，单击“实例详情”进入“在线服务”界面，在“预测”页签的“预测代码”区域，输入调试代码。

  b. 单击“预测”进行测试，预测完成后，右侧“返回结果”区域输出测试结果。如模型准确率不满足预期，可在“数据标注”页签，重新进行模型训练及部署上线。如果您对模型预测结果满意，可根据界面提示调用接口访问在线服务，操作指导请参见访问在线服务。

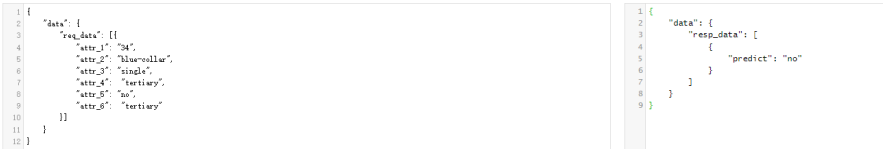
  - 输入代码：其中预测分析要求数据集中数据的预测列名称为class，否则会导致预测失败。

    ```
    {
      "data": {
    ```

```
    "req_data": [{
        "attr_1": "34",
        "attr_2": "blue-collar",
        "attr_3": "single",
        "attr_4":  "tertiary",
        "attr_5": "no",
        "attr_6":  "tertiary"
    }]
  }
}
```

- 返回结果：predict为目标列的预测结果。

图 3-33 预测结果



说明

- 由于“运行中”的在线服务将持续耗费资源，如果不需再使用此在线服务，建议在“在线服务”的操作列单击“更多>停止”，即可停止在线服务的部署。如果需要继续使用此服务，可单击“启动”恢复。
- 如果您启用了自动停止功能，服务将在指定时间后自动停止，不再产生费用。

3.5 使用窍门

3.5.1 创建项目时，如何快速创建 OBS 桶及文件夹?

在创建项目时需要选择训练数据路径，本章节将指导您如何在选择训练数据路径时，快速创建OBS桶和OBS文件夹。

1. 在创建自动学习项目页面，单击数据集输入位置右侧的“”按钮，进入“数据集输入位置”对话框。
2. 单击“新建对象存储服务（OBS）桶”，进入创建桶页面，具体请参见《对象存储服务控制台指南》中的创建桶章节。

图 3-34 快速创建 OBS 桶

3. 桶创建完成后，选择对应桶名称，单击“新建文件夹”，在“新建文件夹”对话框中，填写文件夹“名称”，单击“确定”完成创建，选择创建的文件夹。
   - 文件夹名称不能包含以下字符：\/:*?"<>|。

- 文件夹名称不能以英文句号（.）或斜杠（/）开头或结尾。
- 文件夹的绝对路径总长度不能超过1023字符。
- 任何单个斜杠（/）表示分隔并创建多层级的文件夹。

3.5.2 自动学习生成的模型，存储在哪里？支持哪些其他操作？

模型统一管理

针对自动学习项目，当模型训练完成后，其生成的模型，将自动进入“AI应用管理 > AI应用”页面。模型名称由系统自动命名，前缀与自动学习项目的名称一致，方便辨识。

⚠ 注意

自动学习生成的模型，不支持下载使用。

自动学习生成的模型，支持哪些其他操作

- **支持部署为在线服务、批量服务。**

  在自动学习页面中，仅支持部署为在线服务，如需部署为批量服务，可在“AI应用管理 > AI应用”页面中直接部署。
- **支持创建新版本**

  创建新版本，仅支持从ModelArts训练作业、OBS、模型模板、或自定义镜像中选择元模型。无法从原自动学习项目中，创建新版本。
- **支持删除模型或其模型版本**

4 Workflow

4.1 MLOps 简介

什么是 MLOps

MLOps(Machine Learning Operation)是“机器学习”（Machine Learning）和“DevOps”（Development and Operations）的组合实践。随着机器学习的发展，人们对它的期待不仅仅是学术研究方面的领先突破，更希望这些技术能够系统化地落地到各个场景中。但技术的真实落地和学术研究还是有比较大的差别的。在学术研究中，一个AI算法的开发是面向固定的数据集（公共数据集或者某个特定场景固定数据集），基于单个数据集，不断做算法的迭代与优化。面向场景的AI系统化开发的过程中，除了模型的开发，还有整套系统的开发，于是软件系统开发中成功经验“DevOps”被自然地引入进来。但是，在人工智能时代，传统的DevOps已经不能完全覆盖一个人工智能系统开发的全流程了。

DevOps

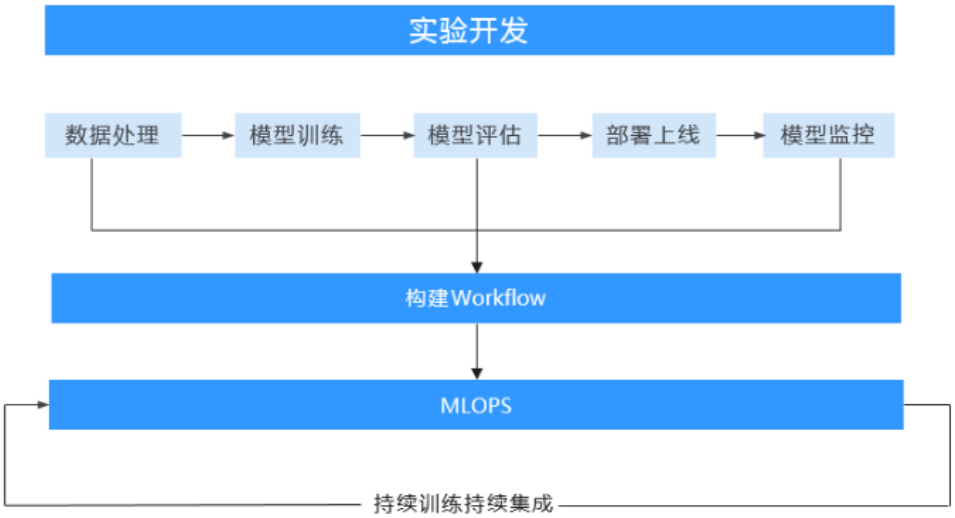
DevOps，即Development and Operations，是一组过程、方法与系统的统称，用于促进软件开发、运维和质量保障部门之间的沟通、协作与整合。在大型的软件系统开发中，DevOps被验证是一个非常成功的方法。DevOps不仅可以加快业务与开发之间的互动与迭代，还可以解决开发与运维之间的冲突。开发侧很快，运维侧太稳，这个就是常说的开发与运维之间固有的、根因的冲突。在AI应用落地的过程中，也有类似的冲突。AI应用的开发门槛较高，需要有一定的算法基础，而且算法需要快速高效地迭代。专业的运维人员追求的更多是稳定、安全和可靠；专业知识也和AI算法大相径庭。运维人员需要去理解算法人员的设计与思路才能保障服务，这对于运维人员来说，门槛更高了。在这种情况下，更多时候可能需要一个算法人员去端到端负责，这样一来，人力成本就会过高。这种模式在少量模型应用的场景是可行的，但是当规模化落地AI应用时，人力问题将会成为瓶颈。

MLOps 功能介绍

机器学习开发流程主要可以定义为四个步骤：项目设计、数据工程、模型构建、部署落地。AI开发并不是一个单向的流水线作业，在开发的过程中，会根据数据和模型结果进行多轮的实验迭代。算法工程师会根据数据特征以及数据的标签做多样化的数据处理以及多种模型优化，以获得在已有的数据集上更好的模型效果。传统的AI应用交付会直接在实验迭代结束后以输出的模型为终点。当应用上线后，随着时间的推移，

会出现模型漂移的问题。新的数据和新的特征在已有的模型上表现会越来越差。在MLOps中，实验迭代的产物将会是一条固化下来的流水线，这条流水线将会包含数据工程、模型算法、训练配置等。用户将会使用这条流水线在持续产生的数据中持续迭代训练，确保这条流水线生产出来的模型的AI应用始终维持在一个较好的状态。

图 4-1 MLOps



MLOps的整条链路需要有一个工具去承载，MLOps打通了算法开发到交付运维的全流程。和以往的开发交付不同，以往的开发与交付过程是分离的，算法工程师开发完的模型，一般都需要交付给下游系统工程师。MLOps和以往的开发交付不同，在这个过程中，算法工程师参与度还是非常高的。企业内部一般都是有一个交付配合的机制。从项目管理角度上需要增加一个AI项目的工作流程机制管理，流程管理不是一个简单的流水线构建管理，它是一个任务管理体系。

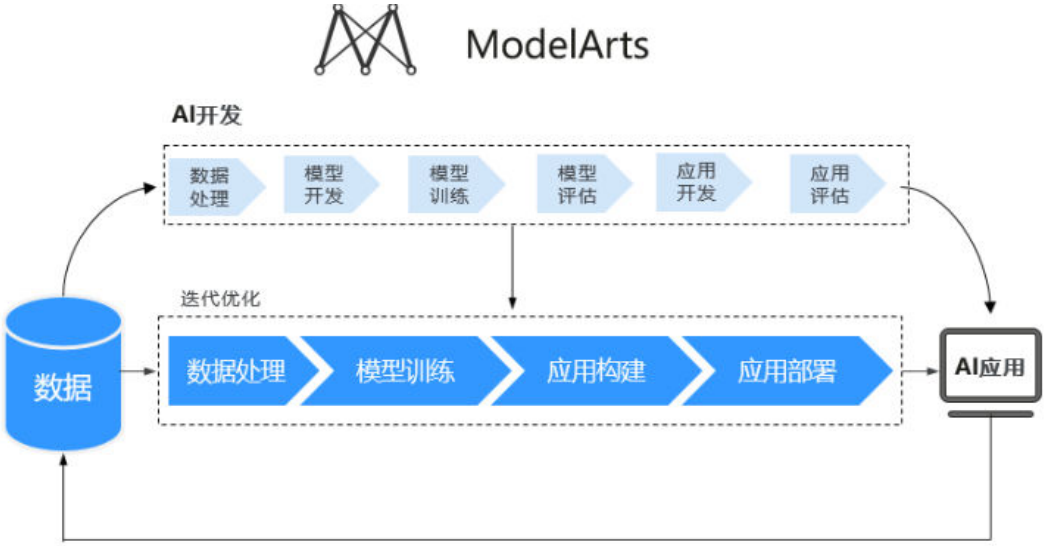
这个工具需要具备以下的能力：

- 流程分析：沉淀行业样例流水线，帮助用户能快速进行AI项目的参考设计，启动快速的AI项目流程设计。
- 流程定义与重定义：以流水线作为承载项，用户能快速定义AI项目，实现训练+推理上线的工作流设计。
- 资源分配：支持账号管理机制给流水线中的参与人员（包含开发者和运维人员）分配相应的资源配额与权限，并查看相应的资源使用情况等。
- 时间安排：围绕子流水线配置相应的子任务安排，并加以通知机制，实现流程执行过程之间配合的运转高效管理。
- 流程质量与效率测评：提供流水线的任务执行过程视图，增加不同的检查点，如数据评估、模型评估、性能评估等，让AI项目管理者能很方便的查看流水线执行过程的质量与效率。
- 流程优化：围绕流水线每一次迭代，用户可以自定义输出相关的核心指标，并获取相应的问题数据与原因等，从而基于这些指标，快速决定下一轮迭代的执行优化。

4.2 什么是 Workflow

Workflow（也称工作流，下文中均可使用工作流进行描述）本质是开发者基于实际业务场景开发用于部署模型或应用的流水线工具。在机器学习的场景中，流水线可能会覆盖数据标注、数据处理、模型开发/训练、模型评估、应用开发、应用评估等步骤。

图 4-2 Workflow



区别于传统的机器学习模型构建，开发者可以使用Workflow开发生产流水线。基于MLOps的概念，Workflow会提供运行记录、监控、持续运行等功能。根据角色的分工与概念，产品上将工作流的开发和持续迭代分开。

一条流水线由多个节点组成，Workflow SDK提供了流水线需要覆盖的功能以及功能需要的参数描述。用户在开发流水线的时候，使用SDK对节点以及节点之间串联的关系进行描述。对流水线的开发操作在Workflow中统称为Workflow的开发态。当确定好整条流水线后，开发者可以将流水线固化下来，提供给其他人使用。使用者无需关注流水线中包含什么算法，也不需要关注流水线是如何实现的。使用者只需要关注流水线生产出来的模型或者应用是否符合上线要求，如果不符合，是否需要调整数据和参数重新迭代。这种使用固化下来的流水线的状态，在Workflow中统称为运行态。

总的来说，Workflow有两种形态。

- 开发态：使用Workflow的Python SDK开发和调测流水线。
- 运行态：可视化配置运行生产好的流水线。

Workflow基于对当前ModelArts已有能力的编排，基于DevOps原则和实践，应用于AI开发过程中，提升了AI应用开发与落地效率，更快的进行模型实验和开发，并更快的将模型部署到生产环境。

工作流的开发态和运行态分别实现了不同的功能。

开发态-开发工作流

开发者结合实际业务的需求，通过Workflow提供的Python SDK，将ModelArts的能力封装成流水线中的一个个步骤。对于AI开发者来说是非常熟悉的开发模式，而且灵活度极高。Python SDK主要提供以下能力。

- 调测：部分运行、全部运行、debug。
- 发布：发布到运行态。
- 实验记录：实验的持久化及管理。

运行态-运行工作流

Workflow提供了可视化的工作流运行方式。使用者只需要关注一些简单的参数配置、模型是否需要重新训练和模型当前的部署情况。

运行态工作流的来源为：通过开发态发布，或者通过AI Hub订阅。

运行态主要提供以下能力。

- 统一配置管理：管理工作流需要配置的参数及使用的资源等。
- 操作工作流：启动、停止、复制、删除工作流。
- 查看运行记录：查看工作流历史运行的参数以及状态记录。

Workflow 的构成

工作流是对一个有向无环图的描述。开发者可以通过 Workflow 进行有向无环图（Directed Acyclic Graph，DAG）的开发。一个DAG是由节点和节点之间的关系描述组成的。开发者通过定义节点的执行内容和节点的执行顺序定义DAG。绿色的矩形表示为一个节点，节点与节点之间的连线则是节点的关系描述。整个DAG的执行其实就是有序的任务执行模板。

图 4-3 工作流

Workflow 提供的样例

ModelArts提供了丰富的基于场景的工作流样例，用户可以。

从 AI Hub 订阅的 Workflow 如何使用

1. 从AI Hub的Workflow资产页面，选择并订阅一个Workflow。
2. 订阅完成后，单击“运行”后跳转到ModelArts控制台界面，选择**资产版本、Workflow名称、云服务区域以及工作空间**，单击“导入”，进入该Workflow的详情页面。

图 4-4 从 AI Gallery 导入工作流

3. 单击右上角的“配置”后进入配置页面，根据您所订阅的工作流，配置Workflow需要的部分输入项和参数，参数配置完成后，单击右上角的“保存配置”。
4. 保存成功后，单击右上角的“启动”，启动Workflow。
5. Workflow进入运行页面，等待Workflow运行。
6. 每一个节点运行状况页面的“状态”为此节点的运行状态，运行成功会自动执行下一个节点的运行，直至所有节点运行成功，代表Workflow完成运行。

图 4-5 完成运行

4.3 如何使用 Workflow

4.3.1 从 AI Hub 订阅的 Workflow 如何使用

1. 从AI Hub的Workflow资产页面，选择并订阅一个Workflow。

2. 订阅完成后，单击“运行”后跳转到ModelArts控制台界面，选择**资产版本**、**Workflow名称**、**云服务区域**以及**工作空间**，单击“导入”，进入该Workflow的详情页面。
3. 单击右上角的“配置”后进入配置页面，根据您所订阅的工作流，配置Workflow需要的部分输入项和参数，参数配置完成后，单击右上角的“保存配置”。
4. 保存成功后，单击右上角的“启动”，启动Workflow。
5. Workflow进入运行页面，等待Workflow运行。

   每一个节点运行状况页面的“状态”为此节点的运行状态，运行成功会自动执行下一个节点的运行，直至所有节点运行成功，代表Workflow完成运行。

图 4-6 完成运行

4.3.2 配置 Workflow

4.3.2.1 配置入口

在运行工作流之前或运行过程中，需要设置工作流相应的参数和使用的资源等内容，这些内容是工作流的开发者根据业务设计呈现给使用者的。用户在获取该条工作流后，可根据自己的需求修改配置，使生成的模型或应用更贴合业务场景。

Workflow的配置包括运行前的配置和运行中的配置。

运行前配置

登录ModelArts控制台，在左侧导航栏选择**Workflow**，进入工作流列表页，有如下两个入口用于运行前的工作流配置。

- 单击Workflow列表页操作栏的“配置”，跳转到工作流配置页。

图 4-7 配置工作流

- 在工作流列表页，单击某条工作流的名称，进入工作流详情页，单击右上角的“配置”，跳转到工作流配置页。

图 4-8 配置工作流

运行中配置

工作流可能存在运行中的配置项，运行到该节点会暂停，等待用户输入操作。

在Workflow总览页面，右方查看待办事项，单击您的工作流名称，可直接跳转至该工作流对应的需要输入的节点，在该节点完成相关参数填写后，单击“继续运行”即可。

图 4-9 Workflow 待办项

4.3.2.2 运行配置

Workflow可以针对工作目录做统一管理，根目录的配置在运行配置页签中完成。

1. 在Workflow列表页，单击某条工作流名称，进入工作流详情页面。
2. 单击页面右上角的“配置”。进入工作流配置详情页。
3. 在“Workflow配置”页签，完成“运行配置”。

图 4-10 运行配置

4.3.2.3 资源配置

一条Workflow中有不止一个节点可以进行资源的配置，当前支持对不同节点配置不同的规格。训练资源规格配置，默认使用公共资源池。

图 4-11 资源配置

当您需要使用专属资源池时，将“是否使用专属资源池”的开关打开即可。

工作流运行到服务部署节点时，需要手动输入推理的资源规格：

1. 待工作流运行至服务部署节点，状态为“等待输入”
2. 在“输入”区域选择推理需要使用到的资源规格。
3. 完成后选择“继续运行”。

4.3.2.4 标签配置

Workflow支持通过标签快速识别对应的工作流，用户在使用的过程中可通过标签筛选出匹配的工作流，实现了工作流分类分块的功能，极大程度上节省了用户的时间。

配置标签

1. 在ModelArts管理控制台，左侧菜单栏单击“Workflow”。进入Workflow列表页。
2. 在列表页根据Workflow工作流名称，找到需要打标签的工作流，单击工作流名称，进入工作流详情页。
3. 在工作流详情页，单击左上角编辑按钮 。
4. 在弹出的编辑Workflow弹窗中，在标签框中输入相应的标签后，单击“新增标签”，新生成的标签会展示在标签行的下方，您可以同时增加多个标签。标签增加完成后，单击“确定”，标签即可生成。

图 4-12 编辑

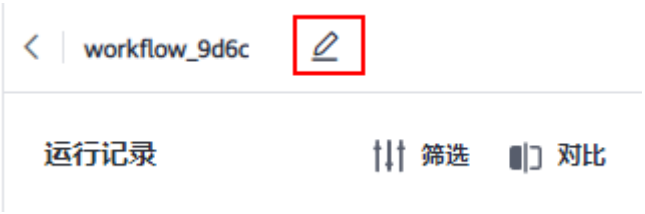


图 4-13 新增标签

通过标签搜索工作流

生成了标签的Workflow，支持在搜索框中按照标签筛选对应的Workflow。

1. 在Workflow列表页，上方搜索栏中，属性类型选择“标签”。

图 4-14 标签搜索

2. 系统会关联出当前页面中所有的Workflow包含的标签，再选择您需要的标签筛选条件后，Workflow列表页所展示的Workflow为您所需要的工作流。

4.3.2.5 输入与输出配置

输入参数与输出参数的配置，可在配置页面进行配置，也可在运行的过程中进行配置。

当运行Workflow时，节点处于“等待输入”状态，也可进行相关参数的配置。

输入配置

输入配置需要填写的参数如下表所示。

表 4-1 输入参数填写说明

输入配置参数	填写说明
数据集	选择您的数据集，或者重新创建数据集。
OBS位置	选择您的OBS路径。
标注任务	选择您的数据集下的标注任务。
运行服务	选择您已部署的在线服务。
容器镜像	选择您注册模型需要的镜像存储路径。

输出配置

这里的输出选择您所创建的OBS的路径，用来存放输出数据。单击“选择”，选择您的OBS路径。

图 4-15 输出配置

4.3.2.6 节点参数配置

对每个节点，开发者都可以选择暴露不同的接口，包含类型：枚举、整数、浮点数、布尔值。

图 4-16 节点参数配置

4.3.2.7 保存配置

通过“配置”入口进入配置页面，所有的配置参数编辑完成之后，单击界面右上角的“保存配置”按钮。

图 4-17 保存配置

完成配置并保存成功后，单击界面右上角的“启动”按钮，出现启动Workflow的弹窗，单击“确定”，工作流就会启动并进入运行页面。

4.3.3 启动/停止/查找/复制/删除 Workflow

启动

当工作流当前状态为非运行态时，可以通过单击“启动”按钮运行工作流，有3种操作方式。

- 工作流列表页：单击操作栏的“启动”按钮，出现启动Workflow询问弹窗，单击“确定”。
- 工作流运行页面：单击右上角的“启动”按钮，出现启动Workflow询问弹窗，单击“确定”。
- 工作流参数配置页面：单击右上角的“启动”按钮，出现启动Workflow询问弹窗，单击“确定”。

查找

在Workflow列表页，您可以通过搜索框，根据工作流的属性类型快速搜索过滤到相应的工作流，可节省您的时间。

1. 登录ModelArts管理控制台，在左侧导航栏选择Workflow，进入Workflow总览页面。
2. 在工作流列表上方的搜索框中，根据您需要的属性类型，例如：名称、状态、当前节点、启动时间、运行时长或标签等，过滤出相应的工作流。

图 4-18 属性类型

3. 单击搜索框右侧的按钮，可设置Workfklow列表页需要展示的内容和展示效果。
   - 表格内容折行：默认为关闭状态。启用此功能可以让Workfklow列表页中的内容在显示时自动换行。禁用此功能可截断文本，Workfklow列表页中仅显示部分内容。
   - 操作列：默认为开启状态，启用此能力可让操作列固定在最后一列永久可见。
   - 自定义显示列：默认所有显示项全部勾选，您可以根据实际需要定义您的显示列。

图 4-19 设置

4. 设置完成后，单击“确定”即可。
5. 同时可支持对Workflow显示列进行排序，单击表头中的箭头，就可对该列进行排序。

停止

可以通过“停止”按钮，主动停止正在运行的工作流，有2种操作方式：

- 工作流列表页：

  当工作流处于“运行中”时，操作栏会出现“停止”按钮。单击“停止”，出现停止Workflow询问弹窗，单击“确定”。
- 进入某条运行中的工作流，单击右上角的“停止”按钮，出现停止Workflow询问弹窗，单击确定。

  说明

  只有处于“运行中”状态的工作流，才会出现“停止”按钮。

  停止Workflow后，关联的训练作业和在线服务也会停止。

复制

某条工作流，目前只能存在一个正在运行的实例，如果用户想要使同一个工作流同时运行多次，可以使用复制工作流的功能。单击列表页的操作栏“更多”，选择“复制”，出现复制Workflow弹窗，新名称会自动生成（生成规则：原工作流名称 + '_copy'）。

用户也可以自行修改新工作流名称，但会有校验规则验证新名称是否符合要求。

说明

新的Workflow名称，必须为1~64位只包含英文、数字、下划线（_）和中划线（-）且以英文开头的名称。

删除

删除工作流有2种方式

- 工作流列表页
  a. 单击操作栏的“更多”，选择“删除”，出现删除Workflow询问弹窗。
  b. 输入“delete”，单击“确定”，删除Workflow。

  图 4-20 确认删除

- 工作流运行页面
  单击界面右上角的“删除”，出现删除Workflow弹窗，输入“DELETE”，单击“确定”，删除Workflow。

  说明

  - 删除后的Workflow无法恢复，请谨慎操作。
  - 删除Workflow后，对应的训练作业和在线服务不会随之被删除，需要分别在“训练管理>训练作业”和“部署上线>在线服务”页面中手动删除任务。

4.3.4 查看 Workflow 运行记录

运行记录是展示某条工作流所有运行状态数据的地方。

1. 在Workflow列表页，单击某条工作流的名称，进入该工作流的详情页面。
2. 在工作流的详情页，左侧区域即为该条工作流的所有运行记录。

图 4-21 查看运行记录

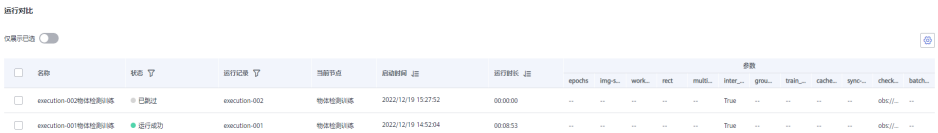
3. 您可以对当前工作流的所有运行记录，进行删除、编辑以及重新运行的操作。
   - 删除：若该条运行记录不再需要，您可以单击“删除”，在弹出的确认框中单击“确定”即可完成运行记录的删除。
   - 编辑：若您想对您当前的工作流下的所有运行记录进行区分，您可以单击“编辑”，对每一条运行记录添加相应的标签予以区分。
   - 重新运行：可以单击“重新运行”直接在某条记录上运行该工作流。
4. 您可以对该条工作流的所有运行记录进行筛选和对比。
   - 筛选：该功能支持您对所有运行记录按照“运行状态”和“运行标签”进行筛选。

     图 4-22 筛选

   - 对比：针对某条工作流的所有运行记录，按照状态、运行记录、启动时间、运行时长、参数等进行对比。

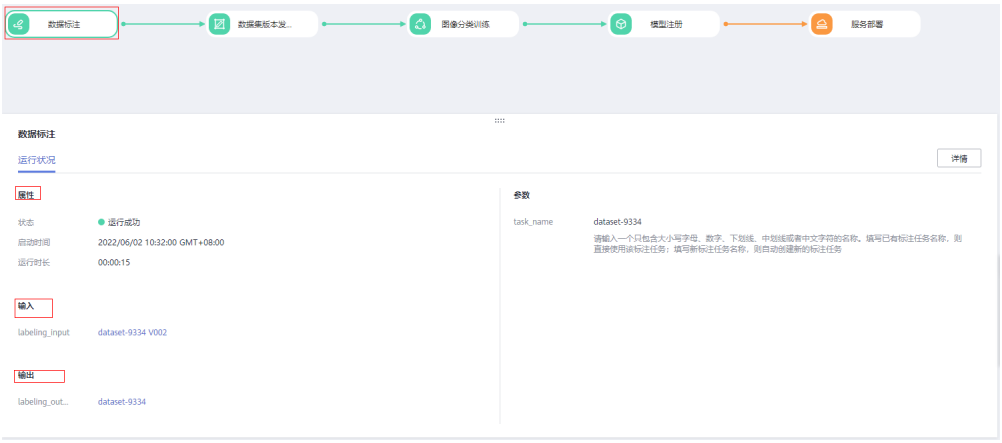
     图 4-23 对比

当单击“启动”运行工作流时，运行记录列表会自动刷新，并更新至最新一条的执行记录数据，并与DAG图和总览数据面板双向联动更新数据。每次启动后都会新增一条运行记录。

用户可以单击Workflow详情页中任一节点查询节点运行状况。包括节点的属性（节点的运行状态、启动时间以及运行时长）

图 4-24 节点运行情况查看



4.3.5 重试/停止/继续运行节点

- 重试

  当单个节点运行失败时，用户可以通过重试按钮重新执行当前节点，无需重新启动工作流。在当前节点的运行状况页面，单击“重试”。在重试之前您也可以前往全局配置页面修改配置，节点重试启动后新修改的配置信息可以在当前执行中立即生效。

- 停止

  单击指定节点查看详情，可以对运行中的节点进行停止操作。

- 继续运行

  对于单个节点中设置了需要运行中配置的参数时，节点运行会处于“等待操作”状态，用户完成相关数据的配置后，可单击“继续运行”按钮并确认继续执行当前节点。

4.3.6 部分运行

针对大型、复杂的Workflow，为节省重复运行消耗的时间，在运行业务场景时，用户可以选择其中的部分节点作为业务场景运行，工作流在执行时将会按顺序执行部分运行节点。

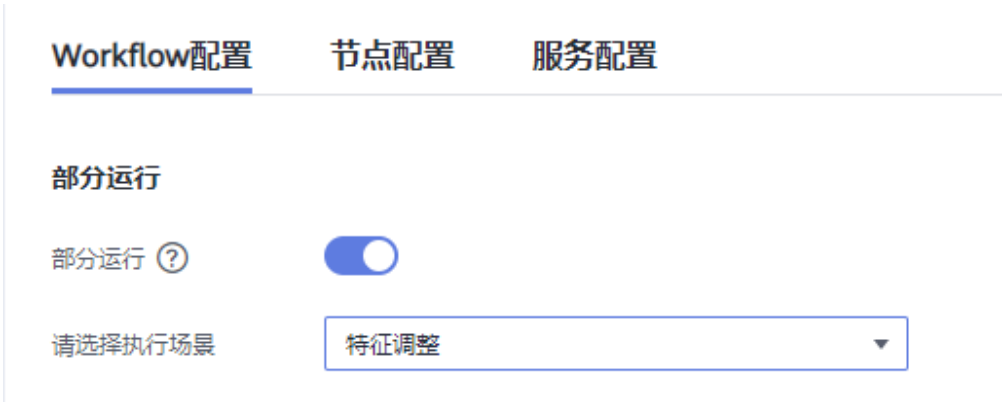
- 创建

  通过SDK创建工作流时，预先定义好部分运行场景。

- 配置

  在配置工作流时，打开“部分运行”开关，选择需要执行的部分运行场景，并填写完善相关节点的参数。

图 4-25 部分运行



- 启动
  保存上一步的配置后，单击“启动”按钮即可启动部分运行场景。

5 数据管理

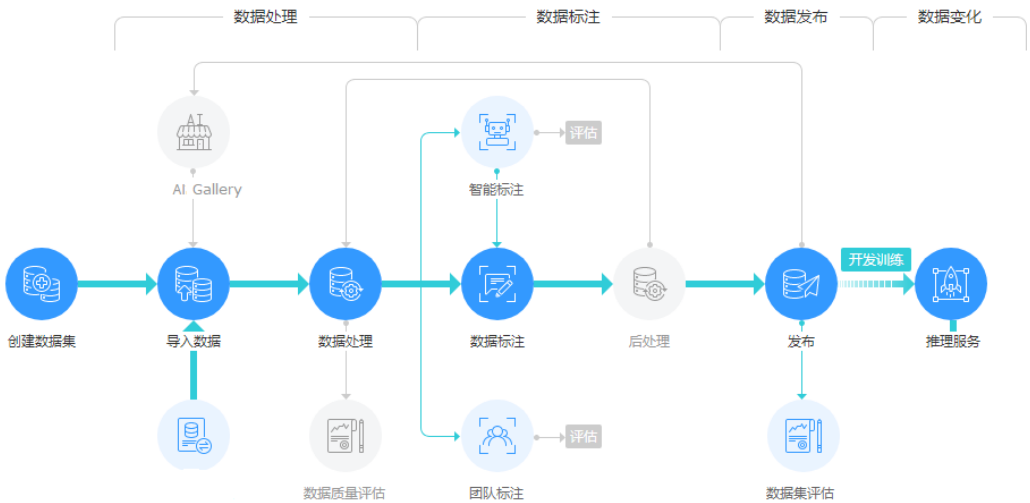
5.1 数据准备与分析

5.1.1 数据准备简介

通常来讲，AI人工智能三要素包括数据、算法和算力。数据的质量会影响模型的精度，一般来说，大量高质量的数据更有可能训练出高精度AI模型。现在很多算法使用常规数据能将准确率做到85%或者90%，而商业化应用往往要求更高，如果将要模型精度提升至96%甚至99%，则需要大量高质量的数据，这个时候也会要求数据更加精细化、场景化、专业化，这往往也成为了AI模型突破瓶颈的关键性条件。如何快速准备大量高质量的数据已经成为AI开发过程中一个极具挑战性的问题。

ModelArts是面向AI开发者的一站式开发平台，能够支撑开发者从数据到AI应用的全流程开发过程，包含数据处理、算法开发、模型训练、模型部署等操作。并且提供AI Hub功能，能够在市场内与其他开发者分享数据、算法、模型等。为了能帮用户快速准备大量高质量的数据，ModelArts数据管理提供了全流程的数据准备、数据处理和数据标注能力。

图 5-1 ModelArts 数据准备全流程



ModelArts数据管理为用户准备高质量的AI数据提供了以下主要能力：

- 解决用户获取数据的问题。
  - 提供多种数据接入方式，支持用户从OBS，MRS，DLI以及DWS等服务导入用户的数据。
  - 提供18+数据增强算子，帮助用户扩增数据，增加训练用的数据量。
- 帮助用户提高数据的质量。
  - 提供图像、文本、音频、视频等多种格式数据的预览，帮助用户识别数据质量。
  - 提供对数据进行多维筛选的能力，用户可以根据样本属性、标注信息等进行样本筛选。
  - 提供12+标注工具，方便用户进行精细化、场景化和专业化的数据标注。
  - 提供基于样本和标注结果进行特征分析，帮助用户整体了解数据的质量。
- 提升用户数据准备的效率。
  - 提供数据版本管理能力，帮助用户提升数据管理的效率。
  - 提供数据校验、数据选择、数据清洗等多种数据处理算子，帮助用户快速处理数据。
  - 提供交互式标注、智能标注等能力，提升用户数据标注的效率。
  - 提供团队标注以及团队标注流程管理能力，帮助用户提升大批量数据标注的能力。

5.1.2 入门教程

本节以准备训练物体检测模型的数据为例，介绍如何针对样例数据，进行数据分析、数据标注等操作，完成数据准备工作。在实际业务开发过程中，可以根据业务需求选择数据管理的一种或多种功能完成数据准备。此次操作分为以下流程：

- **准备工作**
- **创建数据集**
- **数据分析**
- **数据标注**
- **数据发布**
- **数据导出**

准备工作

在使用ModelArts数据管理的功能前，需要先完成以下准备工作。

用户在使用数据管理的过程中，ModelArts需要访问用户的OBS等依赖服务，需要用户进行在“全局配置”页面中进行委托授权。具体操作参考**配置访问授权（全局配置）**。

创建数据集

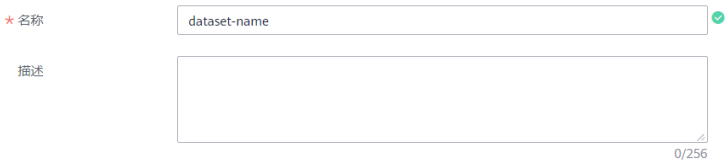
本示例使用OBS中的数据作为数据集的输入目录创建数据集。参考如下操作创建一个物体检测类型的数据集，并将数据导入到数据集中。

步骤1 登录，在左侧菜单栏中选择“数据管理 > 数据集”，进入“数据集”管理页面。

步骤2 单击“创建数据集”，进入“创建数据集”页面，根据数据类型以及数据标注要求，选择创建不同类型的数据集。

1. 填写数据集基本信息，数据集的“名称”和“描述”。

图 5-2 数据集基本信息

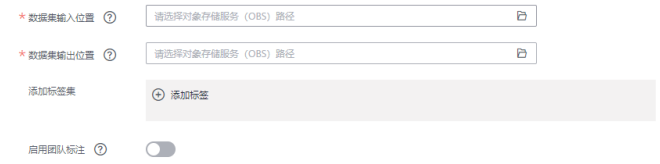


2. 选择“标注场景”和“标注类型”，本案例中分别选择“图片”和“物体检测”。

图 5-3 数据集标注场景和标注类型

3. 选择OBS中的数据目录作为“数据集输入位置”，选择不同的OBS目录作为“数据集输出位置”。

图 5-4 数据集的输入位置和输出位置



4. 参数填写无误后，单击页面右下角“创建”，即可完成数据集的创建。

----结束

数据分析

数据集创建完成后，可以基于图片各项特征，如模糊度、亮度等进行分析，帮助用户更好的分析数据集的数据质量，判断数据集是否满足自己的算法和模型要求。

1. 创建特征分析任务
   a. 在执行特征分析前，需先发布一个数据集版本。在“数据集概览”页单击右上角的“发布”，为数据集发布一个新版本。
   b. 版本发布完成后，进入数据集概览页。选择“数据特征”页签，单击“特征分析”，在弹窗中选择刚才发布的数据集版本，并单击“确定”，启动特征分析任务。

图 5-5 启动特征分析

c. 查看任务进度

任务执行过程中，可以单击“任务历史”，查看任务进度。当任务状态变为“成功”时，表示任务执行完成。

图 5-6 特征分析任务进度

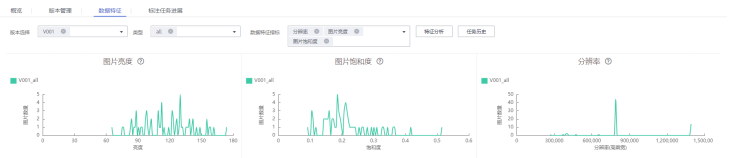
数据集版本	任务ID	创建时间	运行时间（hh:...	状态
V001	QFnWQpFyqi...	2021/08/25 2...	00:01:02	运行中

2. 查看特征分析结果

特征分析任务执行完成后，可以在“数据特征”页签下，选择“数据集版本”、“类型”和“数据特征指标”，页面将自动呈现您选择对应版本及其指标数据，您可以根据呈现的图表了解数据分布情况，帮助您更好的理解您的数据。

- “版本选择”：根据实际情况选择已执行过特征任务的版本，可以选多个进行对比，也可以只选择一个。
- “类型”：根据需要分析的类型选择。支持“all”、“train”、“eval”和“inference”。分别表示所有、训练、评估和推理类型。
- “数据特征指标”：选择您需要展示的指标。详细指标解释，可参见【**特征分析指标列表**】。

图 5-7 查看特征分析结果



在特征分析结果中，例如图片亮度指标，数据分布中，分布不均匀，缺少某一种亮度的图片，而此指标对模型训练非常关键。此时可选择增加对应亮度的图片，让数据更均衡，为后续模型构建做准备。

数据标注

- **人工标注**

a. 在“未标注”页签图片列表中，单击图片，自动跳转到标注页面。

b. 在标注页面的工具栏中选择合适的标注工具，本示例使用矩形框进行标注。

图 5-8 标注工具

c. 使用标注工具选中目标区域，在弹出的标签文本框中，直接输入新的标签名。如果已存在标签，从下拉列表中选择已有的标签。单击“添加”完成标注。

d. 单击页面上方“返回数据标注预览”查看标注信息，在弹框中单击“确定”保存当前标注并离开标注页面。选中的图片被自动移动至“已标注”页签，且在“未标注”和“全部”页签中，标签的信息也将随着标注步骤进行更新，如增加的标签名称、标签对应的图片数量。

- **智能标注**

  通过人工标注完成少量数据标注后，可以通过智能标注对剩下的数据进行自动标注，提高标注的效率。

  a. 在数据集详情页面，单击右上角“启动智能标注”。

  b. 在“启动智能标注”窗口中，填写如下参数，然后单击“提交”。

     - **智能标注类型**：主动学习
     - **算法类型**：快速型

     其他参数采用默认值。

     图 5-9 启动智能标注任务

  c. 查看智能标注任务进度

     智能标注任务启动后，可以在“待确认”页签下查看智能标注任务进度。当任务完成后，即可在“待确认”页签下查看自动标注好的数据。

     图 5-10 查看智能标注任务进度

  d. 确认智能标注结果

在智能标注任务完成后，在“待确认”页签下，单击具体图片进入标注详情页面，可以查看或修改智能标注的结果。

如果智能标注的数据无误，可单击右侧的“确认标注”完成标注，如果标注信息有误，可直接删除错误标注框，然后重新标注，以纠正标注信息。针对物体检测任务，需一张一张确认。确保所有图片已完成确认，然后执行下一步操作。

数据发布

ModelArts训练管理模块支持通过ModelArts数据集或者OBS目录中的文件创建训练作业。如果选择通过数据集作为训练任务的数据源，则需要指定数据集及特定的版本。因此，用户需要为准备好的数据发布一个版本，具体操作参考**发布数据版本**。

说明

为了便于后期的模型构建和开发，对同一数据源来说，将其不同时间对数据的处理和标注按照版本来进行区分，按照需求选择指定的版本使用。

图 5-11 创建训练任务的数据来源

数据导出

ModelArts训练管理模块支持通过ModelArts数据集或者OBS目录中的文件创建训练作业。如果选择通过OBS目录的方式创建训练任务，用户需要将数据集中准备好的数据导出到OBS中。

1. 导出数据到OBS
   a. 在数据集详情页面中，选中需要导出的数据或筛选出需要导出的数据，然后单击右上角“导出”。
   b. 导出方式选择“OBS”，填写相关信息，然后单击“确定”，开始执行导出操作。

      “保存路径”：即导出数据存储的路径。建议不要将数据存储至当前数据集所在的输入路径或输出路径。

      图 5-12 导出至 OBS

c. 数据导出成功后，您可以前往您设置的保存路径，查看到存储的数据。

2. 查看任务历史

当您导出数据后，可以通过任务历史查看导出任务明细。

a. 在数据集详情页面中，单击右上角“任务历史 ”。

b. 在弹出的“任务历史”对话框中，可以查看该数据集之前的导出任务历史。包括“任务ID”、“创建时间”、“导出方式”、“导出路径”、“导出样本总数”和“导出状态”。

图 5-13 导出任务历史

任务ID	创建时间	导出方式	导出路径	导出样...	导出状态
jY4jwr6dODI91cJcJkj	2021/08/25 23:19:27...	OBS	/[redacted]edestria...	66	成功

5.1.3 创建数据集

在ModelArts进行数据准备，首先需要先创建一个数据集，后续的操作如数据导入、数据分析、数据标注等，都是基于数据集来进行的。

5.1.3.1 数据集简介

数据集的类型

当前ModelArts支持如下格式的数据集。

- 图片：对图像类数据进行处理，支持 .jpg、.png、.jpeg、.bmp四种图像格式，支持用户进行图像分类、物体检测、图像分割类型的标注。
- 音频：对音频类数据进行处理，支持.wav格式，支持用户进行声音分类、语音内容、语音分割三种类型的标注。
- 文本：对文本类数据进行处理，支持.txt、.csv格式，支持用户进行文本分类、命名实体、文本三元组三种类型的标注。
- 视频：对视频类数据进行处理，支持.mp4格式，支持用户进行视频标注。
- 自由格式：管理的数据可以为任意格式，目前不支持标注，适用于无需标注或开发者自行定义标注的场景。如果您的数据集需存在多种格式数据，或者您的数据格式不符合其他类型数据集时，可选择自由格式的数据集。

图 5-14 自由格式数据集示例

概览 | 版本管理

文件名称	文件大小	格式	上传时间	文件路径
Images54325efjfowpqwerwtwegregweggergfewfwgweqwfeghwerwyjhnjffefewg...	4.19 KB	图片	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory1/directory11/directory111/
text20_1577686629356.txt	991 B	文本	2019/12/30 14:27:37 GMT+08:00	/obs-sjy-test/data-mix/
Images4355.bmp	7.44 KB	图片	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory1/directory12/
gratisography_mountains_scenic_horizon_600x364.jpg	37.48 KB	图片	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory1/directory12/
7597654747876969845252356263676wh860.gif	4.39 MB	自由格式	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory1/directory11/
reofejofe.exe	--	自由格式	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory2/
7573453675wh300_1576467990491.gif	1.41 MB	自由格式	2019/12/30 14:27:35 GMT+08:00	/obs-sjy-test/data-mix/
images5235.png	5.80 KB	图片	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory1/directory12/
Images358768.png	7.11 KB	图片	2019/12/30 14:27:36 GMT+08:00	/obs-sjy-test/data-mix/directory2/
987576.png	8.25 KB	图片	2019/12/30 14:27:35 GMT+08:00	/obs-sjy-test/data-mix/

不同类型数据集支持的功能列表

其中，不同类型的数据集支持不同的功能，如智能标注、团队标注等。详细信息参考表5-1。

表 5-1 不同类型的数据集支持的功能

数据集类型	标注类型	创建数据集	导入数据	导出数据	发布数据集	修改数据集	管理版本	自动分组	数据特征
图片	图像分类	支持	支持	支持	支持	支持	支持	支持	支持
	物体检测	支持	支持	支持	支持	支持	支持	支持	支持
	图像分割	支持	支持	支持	支持	支持	支持	支持	-
音频	声音分类	支持	支持	-	支持	支持	支持	-	-
	语音内容	支持	支持	-	支持	支持	支持	-	-
	语音分割	支持	支持	-	支持	支持	支持	-	-
文本	文本分类	支持	支持	-	支持	支持	支持	-	-
	命名实体	支持	支持	-	支持	支持	支持	-	-
	文本三元组	支持	支持	-	支持	支持	支持	-	-
视频	视频	支持	支持	-	支持	支持	支持	-	-
自由格式	自由格式	支持	-	_	支持	支持	支持	-	-
表格	表格	支持	支持	-	支持	支持	支持	-	-

规格限制

- 除表格类型之外的数据集（如视频、文本、音频等），单个数据集的最大样本数量限制：1000000，最大标签数量限制：10000。
- 除图片类型之外的数据集（如视频、文本、音频等），单个样本大小限制：5GB。
- 针对图片类数据集（物体检测、图像分类、图像分割），单个图片大小限制：25MB。

- 单个manifest文件大小限制：5GB。
- 文本文件单行大小限制：100KB。
- 数据管理标注结果文件大小限制：100MB。

5.1.3.2 创建数据集

在ModelArts进行数据管理时，首先您需要创建一个数据集，后续的操作，如标注数据、导入数据、数据集发布等，都是基于您创建的数据集。本章节按照非表格类型（图片、音频、文本、视频、自由格式）与表格类型的数据分别介绍如何创建数据集。

前提条件

- 数据管理功能需要获取访问OBS权限，在未进行委托授权之前，无法使用此功能。在使用数据管理功能之前，请前往“全局配置”页面，使用委托完成访问授权。
- 已创建用于存储数据的OBS桶及文件夹。并且，数据存储的OBS桶与ModelArts在同一区域。当前不支持OBS并行文件系统，请选择OBS对象存储。
- ModelArts不支持加密的OBS桶，创建OBS桶时，请勿开启桶加密。

图片、音频、文本、视频、自由格式

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据集”，进入数据集管理页面。
2. 单击“创建数据集”，进入“创建数据集”页面，根据数据类型以及数据标注要求，选择创建不同类型的数据集。填写数据集基本信息。

图 5-15 参数填写

- 名称：数据集的名称，可自定义您的数据集。
- 描述：该数据集的详情信息。
- 数据类型：根据实际需求，选择对应的数据类型。
- 数据来源：
  i. OBS导入数据
  用户在OBS中有准备好的数据时，选择“OBS”，“导入路径”、“数据标注状态”、和数据“标注格式”（当数据标注状态选择“已标注”

时，需要填写该参数）。针对不同类型的数据集，数据输入支持的标注格式不同，ModelArts目前支持的标注格式及其说明请参见 **数据接入简介**。

图 5-16 从 OBS 中导入数据

ii. 从本地上传数据。

当用户没有在OBS存储数据，且AI Gallery中无法下载到所需要的数据时，ModelArts可支持本地上传。本地上传时选择“上传数据存储路径”和“数据标注状态”。单击“文件上传”，上传您本地的数据。并选择“标注格式”（当数据标注状态为“已标注”时，需要关注该参数）。针对不同类型的数据集，数据输入支持的标注格式不同，ModelArts目前支持的标注格式及其说明请参见**数据接入简介**。

- 更多参数填写请参见**表5-2**。

表 5-2 数据集的详细参数

参数名称	说明
导入路径	选择需要导入数据的OBS路径，此位置会作为数据集的数据存储路径。 **说明** “导入路径”不支持OBS并行文件系统下的路径，请选择OBS对象桶。 创建数据集时，此OBS路径下的数据会导入数据集，后续若直接在OBS中修改数据，会造成数据集的数据与OBS的数据不一致，可能导致部分数据不可用。如果需要在数据集中修改数据，建议使用**同步数据源**或4章节**OBS目录导入操作**功能。 超出数据集的样本和标签配额，会导致数据无法正常导入。

参数名称	说明
数据标注状态	选择数据的标注状态，分为“未标注”和“已标注”。 选择“已标注”时，需指定标注格式，并保证数据文件满足相应的格式规范，否则可能存在导入失败的情况。 仅图片（物体检测、图像分类、图像分割）、音频（声音分类）、文本（文本分类）类型的标注任务支持导入已标注数据。
数据集输出位置	选择数据集输出位置的OBS路径，此位置会存放输出的标注信息等文件。 **说明** • 请确保您的OBS路径以字母、数字、下划线命名，不能包含特殊字符，例如：~'@#$%^&*{}[]:;+=<>/以及空格。 • “数据集输出位置”不能与“数据输入路径”为同一路径，且不能是“数据输入路径”的子目录。 • “数据集输出位置”建议选择一个空目录。 • “数据集输出位置”不支持OBS并行文件系统下的路径，请选择OBS对象桶。

3. 参数填写完成，单击“提交”，即可完成数据集的创建。

表格

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据集”，进入数据集管理页面。
2. 单击“创建数据集”，进入“创建数据集”页面，根据数据类型以及数据标注要求，选择创建表格类型的数据集。填写数据集基本信息。

图 5-17 表格类型的参数

- 名称：数据集的名称，可自定义您的数据集。
- 描述：该数据集的详情信息。
- 数据类型：根据实际需求，选择对应的数据类型。
- 更多参数填写请参考**表5-3**。

表 5-3 数据集的详细参数

参数名称	说明
本地上传	上传数据存储路径：选择对应的数据存储的OBS路径。
Schema信息	表格的列名和对应类型，需要跟导入数据的列数保持一致。请根据您导入的数据输入“列名”，同时选择此列的“类型”。其中支持的类型见**表5-4**。 单击“添加Schema信息”，即可增加一行列。创建数据集时必须指定schema，且一旦创建不支持修改。 从OBS数据源导入数据，会自动获取文件路径下csv文件的schema，如果多个csv文件的schema不一致会报错。 **说明** 从OBS选择数据后，Schema信息的列名会自动带出，且默认为表格中的第一行数据。为确保预测代码的正确性，请您手动更改Schema信息中的“列名”为attr_1、attr_2、……、attr_n，其中attr_n为最后一列，代表预测列。
数据集输出位置	选择表格数据存储路径（OBS路径），此位置会存放由数据源导入的数据。此位置不能和OBS数据源中的文件路径相同或为其子目录。 创建表格数据集后，在存储路径下会自动生成以下4个目录。 • annotation：版本发布目录，每次发布版本，会在此目录下生成和版本名称相同的子目录。 • data：数据存放目录，导入的数据会放在此目录。 • logs：日志存放目录。 • temp：临时工作目录。

表 5-4 Schema 数据类型说明

类型	描述	存储空间	范围
String	字符串	-	-
Short	有符号整数	2字节	-32768-32767
Int	有符号整数	4字节	-2147483648～2147483647
Long	有符号整数	8字节	-9223372036854775808～9223372036854775807

类型	描述	存储空间	范围
Double	双精度浮点型	8字节	-
Float	单精度浮点型	4字节	-
Byte	有符号整数	1字节	-128-127
Date	日期类型，描述了特定的年月日，格式：yyyy-MM-dd，例如2014-05-29	-	-
Timestamp	时间戳，表示日期和时间。格式：yyyy-MM-dd HH:mm:ss	-	-
Boolean	布尔类型	1字节	TRUE/FALSE

说明

使用CSV文件时，需要注意以下两点：

- 当数据类型选择String时，默认会把双引号内的数据当作一条，所以同一行数据需要保证双引号闭环，否则会导致数据过大，无法显示。
- 当CSV文件的某一行的列数与定义的Schema不同，则会忽略当前行。

3. 参数填写完成后，单击“提交”，即可完成数据集的创建。

5.1.3.3 修改数据集

对于已创建的数据集，您可以修改数据集的基本信息以匹配业务变化。

前提条件

已存在创建完成的数据集。

修改数据集基本信息

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集列表中，单击操作列的“更多>修改”。
3. 参考表5-5修改数据集基本信息，然后单击“确定”完成修改。

图 5-18 修改数据集

表 5-5 参数说明

参数	说明
名称	数据集的名称，支持1~64位可见字符。名称只能是字母、数字、下划线或者中划线组成的合法字符串。且只能以字母开头。
描述	数据集的简要描述。

5.1.4 数据接入

5.1.4.1 数据接入简介

数据集创建完成后，您还可以通过导入数据的操作，接入更多数据。ModelArts支持从不同数据源导入数据。

- **从OBS导入数据**
- **从本地上传数据**

ModelArts的AIGallery中预置了大量的数据集，文件型和表格型都有，用户可以从AIGallery下载使用预置的数据集。您也可以将您自己的数据导入到ModelArts中。

文件型数据来源

除了从AIGallery下载预置数据集外，文件型数据集还支持从两种数据源导入数据：“OBS”和“本地上传”。导入后，导入目录下的数据会复制至数据集的数据源路径下。

- OBS：又分为从OBS目录或从Manifest文件两种导入方式，需要将导入的数据或Manifest文件提前存储至OBS目录中。
- 本地上传：将本地数据直接通过Internet上传至OBS指定目录后，再导入数据集。

表格型数据来源

除了从AIGallery下载预置数据集外，表格数据集还支持从5种数据源导入数据，分别为对象存储服务（OBS）、数据仓库服务（DWS）、数据湖探索服务（DLI）、MapReduce服务（MRS）和本地上传。

数据集中的数据导入入口

数据集中的数据导入有5个入口。

- 创建数据集时直接从设置的数据导入路径中自动同步数据。

  图 5-19 创建数据集时导入数据

- 创建完数据集后，在数据集列表页面的操作栏单击“导入”，导入数据。

  图 5-20 在数据集列表页导入数据

- 在数据集列表页面，单击某个数据集的名称，进入数据集详情页中，单击“导入>导入”，导入数据。

  图 5-21 在数据集详情页中导入数据

- 在数据集列表页面，单击某个数据集的名称，进入数据集详情页中，单击“同步数据源”，同步OBS中的数据。

图 5-22 在数据集详情页中同步数据源

- 在数据标注的标注作业详情中添加数据。

图 5-23 标注作业详情中添加数据

5.1.4.2 从 OBS 导入数据

5.1.4.2.1 OBS 导入数据简介

导入方式

OBS导入数据方式分为“OBS目录”和“Manifest文件”两种。

- OBS目录：指需要导入的数据集已提前存储至OBS目录中。此时需选择用户具备权限的OBS路径，且OBS路径内的目录结构需满足规范，详细规范请参见**OBS目录导入数据规范说明**。当前只有“图像分类”、“物体检测”、“表格”、“文本分类”和“声音分类”类型的数据集，支持从OBS目录导入数据。其他类型只支持Manifest文件导入数据集的方式。
- Manifest文件：指数据集为Manifest文件格式，Manifest文件定义标注对象和标注内容的对应关系，且Manifest文件已上传至OBS中。Manifest文件的规范请参见**Manifest文件导入规范说明**。

说明

导入“物体检测”类型数据集前，您需要保证标注文件的标注范围不超过原始图片大小，否则可能存在导入失败的情况。

表 5-6 不同类型数据集支持的导入方式

数据集类型	标注类型	OBS目录导入	Manifest文件导入
图片	图像分类	支持 可以导入未标注或已标注数据 已标注数据格式规范：**图像分类**	支持 可以导入未标注或已标注数据 已标注数据格式规范：**图像分类**
	物体检测	支持 可以导入未标注或已标注数据 已标注数据格式规范：**物体检测**	支持 可以导入未标注或已标注数据 已标注数据格式规范：**物体检测**
	图像分割	支持 可以导入未标注或已标注数据 已标注数据格式规范：**图像分割**	支持 可以导入未标注或已标注数据 已标注数据格式规范：**图像分割**
音频	声音分类	支持 导入的是未标注或已标注数据 格式规范：**声音分类**	支持 可以导入未标注或已标注数据 已标注数据格式规范：**声音分类**
	语音内容	支持 导入的是未标注数据	支持 可以导入未标注或已标注数据 已标注数据格式规范：**语音内容**
	语音分割	支持 导入的是未标注数据	支持 可以导入未标注或已标注数据 已标注数据格式规范：**语音分割**
文本	文本分类	支持 导入的是未标注或已标注数据 已标注数据格式规范：**文本分类**	支持 可以导入未标注或已标注数据 已标注数据格式规范：**文本分类**
	命名实体	支持 导入的是未标注数据	支持 可以导入未标注或已标注数据 已标注数据格式规范：**文本命名实体**
	文本三元组	支持 导入的是未标注数据	支持 可以导入未标注或已标注数据 已标注数据格式规范：**文本三元组**

数据集类型	标注类型	OBS目录导入	Manifest文件导入
视频	视频	支持 导入的是未标注数据	支持 可以导入未标注或已标注数据 已标注数据格式规范：**视频标注**
其他	自由格式	支持 导入的是未标注数据	-
表格	表格	支持 格式规范：**表格**	-

5.1.4.2.2 OBS 目录导入操作

前提条件

- 已存在创建完成的数据集。
- 需导入的数据，已存储至OBS中。Manifest文件也需要存储至OBS。
- 确保数据存储的OBS桶与ModelArts在同一区域，并确保用户具有OBS桶的操作权限。

文件型数据从 OBS 目录导入操作

不同类型的数据集，导入操作界面的示意图存在区别，请参考界面信息了解当前类型数据集的示意图。当前操作指导以图像分类的数据集为例。

1. 登录，在左侧菜单栏中选择“数据管理 >数据集”，进入“数据集”管理页面。
2. 在数据集所在行，单击操作列的“导入”。或者，您可以单击数据集名称，进入数据集“概览”页，在页面右上角单击“导入”。
3. 在“导入”对话框中，参考如下说明填写参数，然后单击“确定”。
   - “数据来源”：“OBS”
   - “导入方式”：“目录”。
   - “导入路径”：数据存储的OBS路径。
   - “数据标注状态”：已标注。
   - “高级特征选项 ”：默认关闭，可通过勾选高级选项提供增强功能。

     如“按标签导入”：系统将自动获取此数据集的标签，您可以单击“添加标签”添加相应的标签。此字段为可选字段，您也可以在导入数据集后，在标注数据操作时，添加或删除标签。

图 5-24 导入数据集-OBS

导入成功后，数据将自动同步到数据集中。您可以在“数据集”页面，单击数据集的名称，查看详细数据，并可以通过创建标注任务进行数据标注。

文件型数据标注状态

数据标注状态分为“未标注”和“已标注”。

- 未标注：仅导入标注对象（指待标注的图片，文本等），不导入标注内容（指标注结果信息）。
- 已标注：同时导入标注对象和标注内容，当前“自由格式”的数据集不支持导入标注内容。

  为了确保能够正确读取标注内容，要求用户严格按照规范存放数据：

  导入方式选择目录时，需要用户选择“标注格式”，并按照标注格式的要求存放数据，详细规范请参见**标注格式**章节。

  导入方式选择manifest时，需要满足manifest文件的规范。

说明

- 数据标注状态选择“已标注”，您需要保证目录或manifest文件满足相应的格式规范，否则可能存在导入失败的情况。
- 导入已标注的文件，导入完成后，请检查您导入的数据是否为已标注状态。

表格数据集从 OBS 导入操作

ModelArts支持从OBS导入表格数据，即csv文件。

表格数据集导入说明：

- 导入成功的前提是，数据源的schema需要与创建数据集指定的schema保持一致。其中schema指表格的列名和类型，创建数据集时一旦指定，不支持修改。
- 从OBS导入csv文件，不会校验数据类型，但是列数需要跟数据集的schema保持一致。如果数据格式不合法，会将数据置为null，详见表5-4。
- 导入的csv文件要求如下：需要选择文件所在目录，其中csv文件的列数需要跟数据集schema一致。支持自动获取csv文件的schema。

```
├─dataset-import-example
│      table_import_1.csv
│      table_import_2.csv
│      table_import_3.csv
│      table_import_4.csv
```

5.1.4.2.3 OBS 目录导入数据规范说明

导入数据集时，使用存储在OBS的数据时，数据的存储目录以及文件名称需满足ModelArts的规范要求。

当前只有“图像分类”、“物体检测”、“图像分割”、“文本分类”和“声音分类”标注类型支持按标注格式导入。

说明

- 从OBS目录导入数据时，当前操作用户需具备此OBS路径的读取权限。
- 同时确保数据存储的OBS桶与ModelArts在同一区域。

图像分类

图像分类的数据支持两种格式：

1）ModelArts imageNet 1.0：目录方式，只支持单标签

- 相同标签的图片放在一个目录里，并且目录名字即为标签名。当存在多层目录时，则以最后一层目录为标签名。

  示例如下所示，其中Cat和Dog分别为标签名。

```
dataset-import-example
├─Cat
│      10.jpg
│      11.jpg
│      12.jpg
│
└─Dog
       1.jpg
       2.jpg
       3.jpg
```

2）ModelArts image classification 1.0：txt标签文件，支持多标签

- 当目录下存在对应的txt文件时，以txt文件内容作为图像的标签。

  示例如下所示，import-dir-1和imort-dir-2为导入子目录。

```
dataset-import-example
├─import-dir-1
│      10.jpg
│      10.txt
│      11.jpg
│      11.txt
│      12.jpg
│      12.txt
└─import-dir-2
       1.jpg
       1.txt
       2.jpg
       2.txt
```

  单标签的标签文件示例，如1.txt文件内容如下所示：

```
Cat
```

  多标签的标签文件示例，如2.txt文件内容如下所示：

```
Cat
Dog
```

- 只支持JPG、JPEG、PNG、BMP格式的图片。单张图片大小不能超过5MB，且单次上传的图片总大小不能超过8MB。

物体检测

支持两种格式：

1)ModelArts PASCAL VOC 1.0

- 物体检测的简易模式要求用户将标注对象和标注文件存储在同一目录，并且一一对应，如标注对象文件名为“IMG_20180919_114745.jpg”，那么标注文件的文件名应为“IMG_20180919_114745.xml”。

  物体检测的标注文件需要满足PASCAL VOC格式，格式详细说明请参见**表5-14**。

  示例：

```
├─dataset-import-example
│      IMG_20180919_114732.jpg
│      IMG_20180919_114732.xml
│      IMG_20180919_114745.jpg
│      IMG_20180919_114745.xml
│      IMG_20180919_114945.jpg
│      IMG_20180919_114945.xml
```

  标注文件的示例如下所示：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
    <folder>NA</folder>
    <filename>bike_1_1593531469339.png</filename>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>554</width>
        <height>606</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>Dog</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>279</xmin>
            <ymin>52</ymin>
            <xmax>474</xmax>
            <ymax>278</ymax>
        </bndbox>
    </object>
    <object>
        <name>Cat</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>279</xmin>
            <ymin>198</ymin>
            <xmax>456</xmax>
            <ymax>421</ymax>
        </bndbox>
    </object>
</annotation>
```

- 只支持JPG、JPEG、PNG、BMP格式的图片，单张图片大小不能超过5MB，且单次上传的图片总大小不能超过8MB。

2）YOLO：

- YOLO数据集目录应该遵循以下结构，格式不规范将导致导入任务失败。

```
└─ yolo_dataset/
    |
    ├── obj.names  # 标签集文件
    ├── obj.data   # 记录数据集信息的文件及路径信息(相对路径)
    ├── train.txt  # 训练集中各图片路径信息(相对路径)
    ├── valid.txt  # 验证集中各图片路径信息(相对路径)
    |
    ├── obj_train_data/  # 训练集的图片与对应的标注文件所在目录
    |   ├── image1.txt  # image1对应的带标签bbox列表
    |   ├── image1.jpg
    |   ├── image2.txt
    |   ├── image2.jpg
    |   ├── ...
    |
    ├── obj_valid_data/  # 验证集的图片与对应的标注文件所在目录
    |   ├── image101.txt
    |   ├── image101.jpg
    |   ├── image102.txt
    |   ├── image102.jpg
    |   ├── ...
```

YOLO数据集只支持train和valid子集。如果导入的数据集包括除了上述之外的子集，这些其他子集将被忽略。

- obj.data应包含以下内容，train和valid子集必须至少有一个，其中文件路径均为相对路径。

```
classes = 5 # 可选
names = <path/to/obj.names>#例如 obj.names
train = <path/to/train.txt>#例如 train.txt
valid = <path/to/valid.txt>#可选， 例如valid.txt
backup = backup/ # 可选
```

- obj.names文件记录标签列表。每一行的行标作为该标签的index。

```
label1  # label1的index：  0
label2  # label2的index：  1
label3
...
```

- train.txt和valid.txt文件结构如下，其中文件路径均为相对路径，且文件列表与目录下的文件需一一对应：

```
<path/to/image1.jpg>#例如 obj_train_data/image.jpg
<path/to/image2.jpg>#例如 obj_train_data/image.jpg
...
```

- obj_train_data/ 和obj_valid_data/目录下的.txt文件应该包含对应图像的bbox标签信息，每行代表一个bbox标注。

```
# image1.txt:
# <label_index> <x_center> <y_center> <width> <height>
0 0.250000 0.400000 0.300000 0.400000
3 0.600000 0.400000 0.400000 0.266667
```

  其中x_center, y_center, width, and height分别表示归一化后的目标框中心点x坐标、归一化后的目标框中心点y坐标、归一化后的目标框宽度、归一化后的目标框高度。

- 只支持JPG、JPEG、PNG、BMP格式的图片，单张图片大小不能超过5MB，且单次上传的图片总大小不能超过8MB。

图像分割

ModelArts image segmentation 1.0：

- 要求用户将标注对象和标注文件存储在同一目录，并且一一对应，如标注对象文件名为“IMG_20180919_114746.jpg”，那么标注文件的文件名应为“IMG_20180919_114746.xml”。

  图像分割的标注文件基于PASCAL VOC格式增加了字段mask_source和mask_color，格式详细说明请参见**表5-10**。

  示例：

```
├─dataset-import-example
│      IMG_20180919_114732.jpg
│      IMG_20180919_114732.xml
│      IMG_20180919_114745.jpg
│      IMG_20180919_114745.xml
│      IMG_20180919_114945.jpg
│      IMG_20180919_114945.xml
```

  标注文件的示例如下所示：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
   <folder>NA</folder>
   <filename>image_0006.jpg</filename>
   <source>
      <database>Unknown</database>
   </source>
   <size>
      <width>230</width>
      <height>300</height>
      <depth>3</depth>
```

```
    </size>
    <segmented>1</segmented>
    <mask_source>obs://xianao/out/dataset-8153-Jmf5ylLjRmSacj9KevS/annotation/V001/
segmentationClassRaw/image_0006.png</mask_source>
    <object>
        <name>bike</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <mask_color>193,243,53</mask_color>
        <occluded>0</occluded>
        <polygon>
            <x1>71</x1>
            <y1>48</y1>
            <x2>75</x2>
            <y2>73</y2>
            <x3>49</x3>
            <y3>69</y3>
            <x4>68</x4>
            <y4>92</y4>
            <x5>90</x5>
            <y5>101</y5>
            <x6>45</x6>
            <y6>110</y6>
            <x7>71</x7>
            <y7>48</y7>
        </polygon>
    </object>
</annotation>
```

文本分类

文本分类支持导入“txt”和“csv”两种文件类型，文本的编码格式支持“UTF-8”和“GBK”。

文本分类的标注对象和标注文件有2种存放模式。

- ModelArts text classfication combine 1.0：文本和标注合并，文本分类的标注对象和标注内容在一个文本文件内，标注对象与标注内容之间，多个标注内容之间可分别指定分隔符。

  例如，文本文件的内容如下所示。标注对象与标注内容之间采用tab键分隔。

  ```
  手感很好，反应速度很快，不知道以后怎样	positive
  三个月前买了一个用的非常好果断把旧手机替换下来尤其在待机方面表现得尤为明显	positive
  没充一会电源怎么也会发热呢音量健不好用回弹不好	negative
  算是给自己的父亲节礼物吧物流很快下单不到24小时就到货了耳机更赞有些低音炮的感觉入耳很紧不会掉棒棒哒	positive
  ```

- ModelArts text classfication 1.0：文本和标注分离，文本分类的标注对象和标注文件均为文本文件，并且以行数进行对应，如标注文件中的第一行表示的是标注对象文件中的第一行的标注。

  例如，标注对象“COMMENTS_20180919_114745.txt”的内容如下所示。

  ```
  手感很好，反应速度很快，不知道以后怎样
  三个月前买了一个用的非常好果断把旧手机替换下来尤其在待机方面性能好
  没充一会电源怎么也会发热呢音量健不好用回弹不好
  算是给自己的父亲节礼物吧物流很快下单不到24小时就到货了耳机更赞有些低音炮的感觉入耳很紧不会掉棒棒哒
  ```

  标注文件“COMMENTS_20180919_114745_result.txt”的内容。

  ```
  positive
  negative
  negative
  positive
  ```

此数据格式要求将标注对象和标注文件存储在同一目录，并且一一对应，如标注对象文件名为“COMMENTS_20180919_114745.txt”，那么标注文件名为“COMMENTS _20180919_114745_result.txt”。

数据文件存储示例：

```
├─dataset-import-example
│      COMMENTS_20180919_114732.txt
│      COMMENTS _20180919_114732_result.txt
│      COMMENTS _20180919_114745.txt
│      COMMENTS _20180919_114745_result.txt
│      COMMENTS _20180919_114945.txt
│      COMMENTS _20180919_114945_result.txt
```

声音分类

ModelArts audio classfication dir 1.0：要求用户将相同标签的声音文件放在一个目录里，并且目录名字即为标签名。

示例：

```
dataset-import-example
├─Cat
│      10.wav
│      11.wav
│      12.wav
│
└─Dog
       1.wav
       2.wav
       3.wav
```

表格

支持从OBS导入csv文件，需要选择文件所在目录，其中csv文件的列数需要跟数据集schema一致。支持自动获取csv文件的schema。

```
├─dataset-import-example
│      table_import_1.csv
│      table_import_2.csv
│      table_import_3.csv
│      table_import_4.csv
```

5.1.4.2.4 Manifest 文件导入操作

前提条件

- 已存在创建完成的数据集。
- 需导入的数据，已存储至OBS中。Manifest文件也需要存储至OBS。
- 确保数据存储的OBS桶与ModelArts在同一区域，并确保用户具有OBS桶的操作权限。

文件型数据从 Manifest 导入操作

不同类型的数据集，导入操作界面的示意图存在区别，请参考界面信息了解当前类型数据集的示意图。当前操作指导以图片数据集为例。

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集所在行，单击操作列的“导入”。或者，您可以单击数据集名称，进入数据集“概览”页，在页面右上角单击“导入”。

3. 在“导入”对话框中，参考如下说明填写参数，然后单击“确定”。
   - “数据来源”：“OBS”
   - “导入方式”：“manifest”。
   - “Manifest文件”：存储Manifest文件的OBS路径。
   - “数据标注状态”：已标注。
   - “高级特征选项 ”：默认关闭，可通过勾选高级选项提供增强功能。

     “按标签导入”：系统将自动获取此数据集的标签，您可以单击“添加标签”添加。此字段为可选字段，您也可以在导入数据集后，在标注数据操作时，添加或删除标签。

     “只导入难例”：难例指manifest文件中的“hard”属性，勾选此参数，表示此导入操作，只导入manifest文件“hard”属性中数据信息。

   图 5-25 导入 mainfest 文件

   导入成功后，数据将自动同步到数据集中。您可以在“数据集”页面，单击数据集的名称，查看详细数据，并可以通过创建标注任务进行数据标注。

文件型数据标注状态

数据标注状态分为“未标注”和“已标注”。

- 未标注：仅导入标注对象（指待标注的图片，文本等），不导入标注内容（指标注结果信息）。
- 已标注：同时导入标注对象和标注内容，当前“自由格式”的数据集不支持导入标注内容。

  为了确保能够正确读取标注内容，要求用户严格按照规范存放数据：

  导入方式选择目录时，需要用户选择“标注格式”，并按照标注格式的要求存放数据。

导入方式选择manifest时，需要满足manifest文件的规范，详细规范请参见**标注格式**章节。

说明

数据标注状态选择“已标注”，您需要保证目录或manifest文件满足相应的格式规范，否则可能存在导入失败的情况。

5.1.4.2.5 Manifest 文件导入规范说明

Manifest文件中定义了标注对象和标注内容的对应关系。此导入方式是指导入数据集时，使用Manifest文件。选择导入Manifest文件时，可以从OBS导入。当从OBS导入Manifest文件时，需确保当前用户具备Manifest文件所在OBS路径的权限。

说明

Manifest文件编写规范要求较多，推荐使用OBS目录导入方式导入新数据。一般此功能常用于不同区域或不同账号下ModelArts的数据迁移，即当您已在某一区域使用ModelArts完成数据标注，发布后的数据集可从输出路径下获得其对应的Manifest文件。在获取此Manifest文件后，可将此数据集导入其他区域或者其他账号的ModelArts中，导入后的数据已携带标注信息，无需重复标注，提升开发效率。

Manifest文件描述的是原始文件和标注信息，可用于标注、训练、推理场景。Manifest文件中也可以只有原始文件信息，没有标注信息，如用于推理场景，或用于生成未标注的数据集。Manifest文件需满足如下要求：

- Manifest文件使用UTF-8编码。
- Manifest文件使用json lines格式（jsonlines.org），一行一个json对象。

```
{"source": "/path/to/image1.jpg", "annotation": … }
{"source": "/path/to/image2.jpg", "annotation": … }
{"source": "/path/to/image3.jpg", "annotation": … }
```

为了说明方便，下面的Manifest例子格式化为多行的json对象。

- Manifest文件可以由用户、第三方工具或ModelArts数据标注生成，其文件名没有特殊要求，可以为任意合法文件名。为了ModelArts系统内部使用方便，ModelArts数据标注功能生成的文件名由如下字符串组成：“DatasetName-VersionName.manifest”。例如，“animal-v20190123113030412 3.manifest”。

图像分类

```
{
  "source":"s3://path/to/image1.jpg",
  "usage":"TRAIN",
  "hard":"true",
  "hard-coefficient":0.8,
  "id":"0162005993f8065ef47eefb59d1e4970",
  "annotation": [
    {
      "type": "modelarts/image_classification",
      "name": "cat",
      "property": {
        "color":"white",
        "kind":"Persian cat"
      },
      "hard":"true",
      "hard-coefficient":0.8,
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
```

```
            "type": "modelarts/image_classification",
            "name":"animal",
            "annotated-by":"modelarts/active-learning",
            "confidence": 0.8,
            "creation-time":"2019-01-23 11:30:30"
        }],
    "inference-loc":"/path/to/inference-output"
}
```

表 5-7 字段说明

字段	是否必选	说明
source	是	被标注对象的URI。数据来源的类型及示例请参考表5-8。
usage	否	默认为空，取值范围： ● TRAIN：指明该对象用于训练。 ● EVAL：指明该对象用于评估。 ● TEST：指明该对象用于测试。 ● INFERENCE：指明该对象用于推理。 如果没有给出该字段，则使用者自行决定如何使用该对象。
id	否	此参数为系统导出的样本id，导入时可以不用填写。
annotation	否	如果不设置，则表示未标注对象。annotation值为一个对象列表，详细参数请参见表5-9。
inference-loc	否	当此文件由推理服务生成时会有该字段，表示推理输出的结果文件位置。

表 5-8 数据来源类型

类型	示例
OBS	“source”：“s3://path-to-jpg”
Content	“source”：“content://I love machine learning”

表 5-9 annotation 对象说明

字段	是否必选	说明
type	是	标签类型。取值范围为： ● image_classification：图像分类 ● text_classification：文本分类 ● text_entity：文本命名实体 ● object_detection：对象检测 ● audio_classification：声音分类 ● audio_content：声音内容 ● audio_segmentation：声音起止点
name	是/否	对于分类是必选字段，对于其他类型为可选字段，本示例为图片分类名称。
id	是/否	标签ID。对于三元组是必选字段，对于其他类型为可选字段。三元组的实体标签ID格式为“E+数字”，比如“E1”、“E2”，三元组的关系标签ID格式为“R+数字”，例如“R1”、“R2”。
property	否	包含对标注的属性，例如本示例中猫有两个属性，颜色（color）和品种（kind）。
hard	否	表示是否是难例。“True”表示该标注是难例，“False”表示该标注不是难例。
annotated-by	否	默认为“human”，表示人工标注。 ● human
creation-time	否	创建该标注的时间。是用户写入标注的时间，不是Manifest生成时间。
confidence	否	表示机器标注的置信度。范围为0～1。

图像分割

```
{
  "annotation": [{
    "annotation-format": "PASCAL VOC",
    "type": "modelarts/image_segmentation",
    "annotation-loc": "s3://path/to/annotation/image1.xml",
    "creation-time": "2020-12-16 21:36:27",
    "annotated-by": "human"
  }],
  "usage": "train",
  "source": "s3://path/to/image1.jpg",
  "id": "16d196c19bf61994d7deccafa435398c",
  "sample-type": 0
}
```

- “source”、“usage”、“annotation”等参数说明与图像分类一致，详细说明请参见表5-7。
- “annotation-loc”：对于图像分割、物体检测是必选字段，对于其他类型是可选字段，标注文件的存储路径。

- “annotation-format”：描述标注文件的格式，可选字段，默认为“PASCAL VOC”。目前只支持“PASCAL VOC”。
- “**sample-type**”： 样本格式，0表示图片，1表示文本，2表示语音，4表示表格，6表示视频

表 5-10 PASCAL VOC 格式说明

字段	是否必选	说明
folder	是	表示数据源所在目录。
filename	是	被标注文件的文件名。
size	是	表示图像的像素信息。 • width：必选字段，图片的宽度。 • height：必选字段，图片的高度。 • depth：必选字段，图片的通道数。
segmented	是	表示是否用于分割。
mask_source	否	表示图像分割保存的mask路径
object	是	表示物体检测信息，多个物体标注会有多个object体。 • name：必选字段，标注内容的类别。 • pose：必选字段，标注内容的拍摄角度。 • truncated：必选字段，标注内容是否被截断（0表示完整）。 • occluded：必选字段，标注内容是否被遮挡（0表示未遮挡） • difficult：必选字段，标注目标是否难以识别（0表示容易识别）。 • confidence：可选字段，标注目标的置信度，取值范围0-1之间。 • bndbox：必选字段，标注框的类型，可选值请参见**表5-11**。 • mask_color：必选字段，标签的颜色，以RGB值表示

表 5-11 标注框类型描述

type	形状	标注信息
polygon	多边形	各点坐标。 <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6> <x7>100<x7> <y7>100<y7>

示例：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
    <folder>NA</folder>
    <filename>image_0006.jpg</filename>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>230</width>
        <height>300</height>
        <depth>3</depth>
    </size>
    <segmented>1</segmented>
    <mask_source>obs://xianao/out/dataset-8153-Jmf5ylLjRmSacj9KevS/annotation/V001/
segmentationClassRaw/image_0006.png</mask_source>
    <object>
        <name>bike</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <mask_color>193,243,53</mask_color>
        <occluded>0</occluded>
        <polygon>
            <x1>71</x1>
            <y1>48</y1>
            <x2>75</x2>
            <y2>73</y2>
            <x3>49</x3>
            <y3>69</y3>
            <x4>68</x4>
            <y4>92</y4>
            <x5>90</x5>
            <y5>101</y5>
            <x6>45</x6>
            <y6>110</y6>
```

```
            <x7>71</x7>
            <y7>48</y7>
        </polygon>
    </object>
</annotation>
```

文本分类

```
{
  "source": "content://I like this product ",
  "id":"XGDVGS",
  "annotation": [
    {
      "type": "modelarts/text_classification",
      "name": " positive",
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    } ]
}
```

content字段是指被标注的文本，其他参数解释与**图像分类**相同，请参见**表5-7**。

文本命名实体

```
{
  "source":"content://Michael Jordan is the most famous basketball player in the world.",
  "usage":"TRAIN",
  "annotation":[
    {
      "type":"modelarts/text_entity",
      "name":"Person",
      "property":{
        "@modelarts:start_index":0,
        "@modelarts:end_index":14
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_entity",
      "name":"Category",
      "property":{
        "@modelarts:start_index":34,
        "@modelarts:end_index":44
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

“source”、“usage”、“annotation”等参数说明与**图像分类**一致，详细说明请参见**表5-7**。

其中，property的参数解释如**表5-12**所示。例如，当“"source":"content://Michael Jordan"”时，如果要提取“Michael”，则对应的“start_index”为“0”，“end_index”为“7”。

表 5-12 property 参数说明

参数名	数据类型	说明
@modelarts:start_index	Integer	文本的起始位置，值从0开始，包括start_index所指的字符。

参数名	数据类型	说明
@modelarts:end_index	Integer	文本的结束位置，但不包括end_index所指的字符。

文本三元组

```
{
  "source":"content://"Three Body" is a series of long science fiction novels created by Liu Cix.",
  "usage":"TRAIN",
  "annotation":[
    {
      "type":"modelarts/text_entity",
      "name":"Person",
      "id":"E1",
      "property":{
        "@modelarts:start_index":67,
        "@modelarts:end_index":74
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_entity",
      "name":"Book",
      "id":"E2",
      "property":{
        "@modelarts:start_index":0,
        "@modelarts:end_index":12
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_triplet",
      "name":"Author",
      "id":"R1",
      "property":{
        "@modelarts:from":"E1",
        "@modelarts:to":"E2"
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_triplet",
      "name":"Works",
      "id":"R2",
      "property":{
        "@modelarts:from":"E2",
        "@modelarts:to":"E1"
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

“source”、“usage”、“annotation”等参数说明与**图像分类**一致，详细说明请参见**表5-7**。

其中，property的参数解释如**表5 property参数说明**所示。其中，“@modelarts:start_index”和“@modelarts:end_index”和文本命名实体的参数说明一致。例如，当“source”： "content://"Three Body" is a series of long science

fiction novels created by Liu Cix."时，“Liu Cix”是实体Person（人物），“Three Body”是实体Book（书籍），Person指向Book的关系是Author（作者），Book指向Person的关系是Works（作品）。

表 5-13 property 参数说明

参数名	数据类型	说明
@modelarts:start_index	Integer	三元组实体的起始位置，值从0开始，包括start_index所指的字符。
@modelarts:end_index	Integer	三元组实体的结束位置，但不包括end_index所指的字符。
@modelarts:from	String	三元组关系的起始实体id
@modelarts:to	String	三元组关系的指向实体id

物体检测

```
{
  "source":"s3://path/to/image1.jpg",
  "usage":"TRAIN",
  "hard":"true",
  "hard-coefficient":0.8,
  "annotation": [
    {
      "type":"modelarts/object_detection",
      "annotation-loc": "s3://path/to/annotation1.xml",
      "annotation-format":"PASCAL VOC",
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }]
}
```

- “source”、“usage”、“annotation”等参数说明与**图像分类**一致，详细说明请参见**表5-7**。
- “annotation-loc”：对于物体检测、图像分割是必选字段，对于其他类型是可选字段，标注文件的存储路径。
- “annotation-format”：描述标注文件的格式，可选字段，默认为“PASCAL VOC”。目前只支持“PASCAL VOC”。

表 5-14 PASCAL VOC 格式说明

字段	是否必选	说明
folder	是	表示数据源所在目录。
filename	是	被标注文件的文件名。
size	是	表示图像的像素信息。 • width：必选字段，图片的宽度。 • height：必选字段，图片的高度。 • depth：必选字段，图片的通道数。

字段	是否必选	说明
segmented	是	表示是否用于分割。
object	是	表示物体检测信息，多个物体标注会有多个object体。 • name：必选字段，标注内容的类别。 • pose：必选字段，标注内容的拍摄角度。 • truncated：必选字段，标注内容是否被截断（0表示完整）。 • occluded：必选字段，标注内容是否被遮挡（0表示未遮挡） • difficult：必选字段，标注目标是否难以识别（0表示容易识别）。 • confidence：可选字段，标注目标的置信度，取值范围0-1之间。 • bndbox：必选字段，标注框的类型，可选值请参见**表5-15**。

表 5-15 标注框类型描述

type	形状	标注信息
point	点	点的坐标。 <x>100<x> <y>100<y>
line	线	各点坐标。 <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>
bndbox	矩形框	左上和右下两个点坐标。 <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymax>200<ymax>

type	形状	标注信息
polygon	多边形	各点坐标。 <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	圆形	圆心坐标和半径。 <cx>100<cx> <cy>100<cy> <r>50<r>

示例：

```
<annotation>
  <folder>test_data</folder>
  <filename>260730932.jpg</filename>
  <size>
     <width>767</width>
     <height>959</height>
     <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
     <name>point</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <point>
        <x1>456</x1>
        <y1>596</y1>
     </point>
  </object>
  <object>
     <name>line</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <line>
        <x1>133</x1>
        <y1>651</y1>
        <x2>229</x2>
        <y2>561</y2>
     </line>
  </object>
```

```
  <object>
     <name>bag</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <bndbox>
        <xmin>108</xmin>
        <ymin>101</ymin>
        <xmax>251</xmax>
        <ymax>238</ymax>
     </bndbox>
  </object>
  <object>
     <name>boots</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <hard-coefficient>0.8</hard-coefficient>
     <polygon>
        <x1>373</x1>
        <y1>264</y1>
        <x2>500</x2>
        <y2>198</y2>
        <x3>437</x3>
        <y3>76</y3>
        <x4>310</x4>
        <y4>142</y4>
     </polygon>
  </object>
  <object>
     <name>circle</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <circle>
        <cx>405</cx>
        <cy>170</cy>
        <r>100<r>
     </circle>
  </object>
</annotation>
```

声音分类

```
{
"source":
"s3://path/to/pets.wav",
   "annotation": [
      {
         "type": "modelarts/audio_classification",
         "name":"cat",
         "annotated-by":"human",
         "creation-time":"2019-01-23 11:30:30"
      }
   ]
}
```

“source”、“usage”、“annotation”等参数说明与**图像分类**一致，详细说明请参见**表5-7**。

语音内容

```
{
   "source":"s3://path/to/audio1.wav",
   "annotation":[
```

```
    {
      "type":"modelarts/audio_content",
      "property":{
        "@modelarts:content":"Today is a good day."
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

- “source”、“usage”、“annotation”等参数说明与**图像分类**一致，详细说明请参见**表5-7**。
- “property”中的“@modelarts:content”参数，数据类型为“String”，表示语音内容。

语音分割

```
{
  "source":"s3://path/to/audio1.wav",
  "usage":"TRAIN",
  "annotation":[
    {

"type":"modelarts/audio_segmentation",
      "property":{
        "@modelarts:start_time":"00:01:10.123",
        "@modelarts:end_time":"00:01:15.456",

        "@modelarts:source":"Tom",

        "@modelarts:content":"How are you?"
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/audio_segmentation",
      "property":{
        "@modelarts:start_time":"00:01:22.754",
        "@modelarts:end_time":"00:01:24.145",
        "@modelarts:source":"Jerry",
        "@modelarts:content":"I'm fine, thank you."
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

- “source”、“usage”、“annotation”等参数说明与**图像分类**一致，详细说明请参见**表5-7**。
- “property”的参数解释如**表5-16**所示。

表 5-16 “property”参数说明

参数名	数据类型	描述
@modelarts:start_time	String	声音的起始时间，格式为“hh:mm:ss.SSS”。 其中“hh”表示小时，“mm”表示分钟，“ss”表示秒，“SSS”表示毫秒。

参数名	数据类型	描述
@modelarts:end_time	String	声音的结束时间，格式为“hh:mm:ss.SSS”。 其中“hh”表示小时，“mm”表示分钟，“ss”表示秒，“SSS”表示毫秒。
@modelarts:source	String	声音来源。
@modelarts:content	String	声音内容。

视频标注

```
{
  "annotation": [{
    "annotation-format": "PASCAL VOC",
    "type": "modelarts/object_detection",
    "annotation-loc": "s3://path/to/annotation1_t1.473722.xml",
    "creation-time": "2020-10-09 14:08:24",
    "annotated-by": "human"
  }],
  "usage": "train",
  "property": {
    "@modelarts:parent_duration": 8,
    "@modelarts:parent_source": "s3://path/to/annotation1.mp4",
    "@modelarts:time_in_video": 1.473722
  },
  "source": "s3://input/path/to/annotation1_t1.473722.jpg",
  "id": "43d88677c1e9a971eeb692a80534b5d5",
  "sample-type": 0
}
```

- “source”、“usage”、“annotation”等参数说明与图像分类一致，详细说明请参见表5-7。
- “annotation-loc”：对于物体检测、是必选字段，对于其他类型是可选字段，标注文件的存储路径。
- “annotation-format”：描述标注文件的格式，可选字段，默认为“PASCAL VOC”。目前只支持“PASCAL VOC”。
- “**sample-type**”：样本格式，0表示图片，1表示文本，2表示语音，4表示表格，6表示视频。

表 5-17 property 参数说明

参数名	数据类型	说明
@modelarts:parent_duration	Double	标注视频的时长，单位：秒。
@modelarts:time_in_video	Double	标注的视频帧的时间戳，单位：秒。
@modelarts:parent_source	String	标注视频的OBS路径。

表 5-18 PASCAL VOC 格式说明

字段	是否必选	说明
folder	是	表示数据源所在目录。
filename	是	被标注文件的文件名。
size	是	表示图像的像素信息。 • width：必选字段，图片的宽度。 • height：必选字段，图片的高度。 • depth：必选字段，图片的通道数。
segmented	是	表示是否用于分割。
object	是	表示物体检测信息，多个物体标注会有多个object体。 • name：必选字段，标注内容的类别。 • pose：必选字段，标注内容的拍摄角度。 • truncated：必选字段，标注内容是否被截断（0表示完整）。 • occluded：必选字段，标注内容是否被遮挡（0表示未遮挡） • difficult：必选字段，标注目标是否难以识别（0表示容易识别）。 • confidence：可选字段，标注目标的置信度，取值范围0-1之间。 • bndbox：必选字段，标注框的类型，可选值请参见表5-19。

表 5-19 标注框类型描述

type	形状	标注信息
point	点	点的坐标。 <x>100<x> <y>100<y>
line	线	各点坐标。 <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>

type	形状	标注信息
bndbox	矩形框	左上和右下两个点坐标。 <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymax>200<ymax>
polygon	多边形	各点坐标。 <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	圆形	圆心坐标和半径。 <cx>100<cx> <cy>100<cy> <r>50<r>

示例：

```
<annotation>
  <folder>test_data</folder>
  <filename>260730932_t1.473722.jpg.jpg</filename>
  <size>
     <width>767</width>
     <height>959</height>
     <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
     <name>point</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <occluded>0</occluded>
     <difficult>0</difficult>
     <point>
        <x1>456</x1>
        <y1>596</y1>
     </point>
  </object>
  <object>
     <name>line</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
```

```
    <occluded>0</occluded>
    <difficult>0</difficult>
    <line>
      <x1>133</x1>
      <y1>651</y1>
      <x2>229</x2>
      <y2>561</y2>
    </line>
  </object>
  <object>
    <name>bag</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <bndbox>
      <xmin>108</xmin>
      <ymin>101</ymin>
      <xmax>251</xmax>
      <ymax>238</ymax>
    </bndbox>
  </object>
  <object>
    <name>boots</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <hard-coefficient>0.8</hard-coefficient>
    <polygon>
      <x1>373</x1>
      <y1>264</y1>
      <x2>500</x2>
      <y2>198</y2>
      <x3>437</x3>
      <y3>76</y3>
      <x4>310</x4>
      <y4>142</y4>
    </polygon>
  </object>
  <object>
    <name>circle</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <circle>
      <cx>405</cx>
      <cy>170</cy>
      <r>100<r>
    </circle>
  </object>
</annotation>
```

5.1.4.3 从本地上传数据

前提条件

- 已存在创建完成的数据集。
- 创建一个空的OBS桶，OBS桶与ModelArts在同一区域，并确保用户具有OBS桶的操作权限。

本地上传

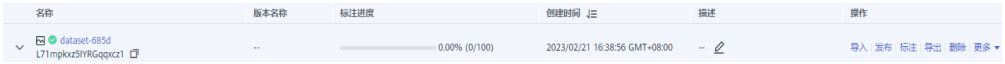
文件型和表格型数据均支持从本地上传。从本地上传的数据存储在OBS目录中，请先提前创建OBS桶。

从本地上传的数据单次最多支持100个文件同时上传，总大小不超过5GB。

不同类型的数据集，导入操作界面的示意图存在区别，请参考界面信息了解当前类型数据集的示意图。当前操作指导以图像分类的数据集为例。

1. 登录，在左侧菜单栏中选择“数据管理 >数据集”，进入“数据集”管理页面。
2. 在数据集所在行，单击操作列的“导入”。

图 5-26 导入数据



或者，您可以单击数据集名称，进入数据集“概览”页，在页面右上角单击“导入”。

3. 在“导入”对话框中，参考如下说明填写参数，然后单击“确定”。
   - “数据来源”：“本地上传”
   - “上传数据存储路径”：数据存储的OBS路径。
   - “上传数据”：单击“文件上传”，上传本地的数据，单击“确定”。

图 5-27 从本地上传数据

5.1.5 数据分析与预览

用户的原始数据的质量一般无法满足训练的要求，如存在不合法的数据、重复数据等。为了帮助用户提高数据的质量，ModelArts提供了多种能力：

- **自动分组**：通过聚类对数据进行预分类，用户可以根据预分类结果进行标注，有助于均衡不同类别的数据标注数量。
- **数据筛选**：用户可以根据样本属性，自动分组结果等进行数据筛选，帮助用户过滤数据。

- **数据特征分析**：分析数据或者标注结果的特征分布，如图像亮度分布、标注框的分布等，帮助用户分析数据的均衡性，从而提升模型训练的效果。

5.1.5.1 数据处理

当数据采集和接入之后，数据一般是不能直接满足训练要求的。为了保障数据质量，以免对后续操作（如数据标注、模型训练等）带来负面影响，开发过程通常需要进行数据处理。ModelArts提供了数据处理的功能，目的是帮助用户从大量的、杂乱无章的、难以理解的数据中抽取或者生成对某些特定的人们来说是有价值、有意义的数据。

ModelArts提供了四种基本的数据处理功能：

- 数据校验：帮助AI开发者提前识别数据中的不合法数据，如已损坏数据、不合格数据等，有效防止数据噪声造成的算法精度下降或者训练失败问题。
- 数据清洗：在数据校验的基础上，对数据进行一致性检查，处理一些无效值。
- 数据选择：在AI开发过程中，采集的数据可能存在大量重复数据，重复数据对模型精度提升并没有太大作用，反而需要花费很多时间对其进行标注。使用数据选择进行数据预处理，对采集到的数据去重，根据相似度删除一些重复度比较高的数据。
- 数据增强：数据增强的目的是帮助用户增加数据量。

5.1.5.2 自动分组

为了提升智能标注算法精度，可以均衡标注多个类别，有助于提升智能标注算法精度。ModelArts内置了分组算法，您可以针对您选中的数据，执行自动分组，提升您的数据标注效率。

自动分组可以理解为数据标注的预处理，先使用聚类算法对未标注图片进行聚类，再根据聚类结果进行处理，可以分组打标或者清洗图片。

例如，用户通过搜索引擎搜索XX，将相关图片下载并上传到数据集，然后再使用自动分组，可以将XX图片分类，比如论文、宣传海报、确认为XX的图片、其他。用户可以根据分组结果，快速剔除掉不想要的，或者将某一类直接全选后添加标签。

说明

目前只有“图像分类”、“物体检测”和“图像分割”类型的数据集支持自动分组功能。

启动自动分组任务

1. 登录，在左侧菜单栏中选择“数据管理>数据标注”，进入“数据标注”管理页面。
2. 在标注作业列表中，选择“物体检测”或“图像分类”类型的标注作业，单击标注作业名称进入“标注作业详情页”。
3. 在数据集详情页的“全部”页签中，单击“自动分组 > 启动任务”。

   说明

   只能在“全部”页签下启动自动分组任务或查看任务历史。
4. 在弹出的“自动分组”对话框中，填写参数信息，然后单击“确定”。
   - “分组数”：填写2~200之间的整数，指将图片分为多少组。
   - “结果处理方式”：“更新属性到当前样本中”，或者“保存到对象存储服务（OBS）”。

- “属性名称”：当选择“更新属性到当前样本中”时，需输入一个属性名称。
- “结果存储目录”：当选择“保存到对象存储服务（OBS）”时，需指定一个用于存储的OBS路径。
- “高级特征选项”：启用此功能后，可选择“清晰度”、“亮度”、“图像色彩”等维度为自动分组功能增加选项，使得分组着重于图片亮度、色彩和清晰度等特征进行分组。支持多选。

5. 启动任务提交成功后，界面右上角显示此任务的进度。等待任务执行完成后，您可以查看自动分组任务的历史记录，了解任务状态。

查看自动分组结果

在数据集详情页面的“全部”页签中，展开“筛选条件”，将“样本属性”设置为自动分组任务中的“属性名称”，并通过设置样本属性值，筛选出分组结果。

图 5-28 查看自动分组结果

查看自动分组的历史任务

在数据集详情页面的“全部”页签中，单击“自动分组 > 任务历史”。在弹出的“任务历史”对话框中，展示当前数据集之前执行的自动分组任务的基本信息。

图 5-29 自动分组任务历史

5.1.5.3 数据筛选

在数据概览页中，默认展示数据集的概览情况。在界面右上方，单击“开始标注”，进入数据集的详细数据页面，默认展示数据集中全部数据。在“全部”、“未标注”或“已标注”页签下，您可以在筛选条件区域，添加筛选条件，快速过滤出您想要查看的数据。

支持的筛选条件如下所示，您可以设置一个或多个选项进行筛选。

- 难例集：难例或非难例。
- 标签：您可以选择全部标签，或者基于您指定的标签，选中其中一个或多个。
- 样本创建时间：1个月内、1天内或自定义，如果选择自定义，可以在时间框中指定明确时间范围。
- 文件名或目录：根据文件名称或者文件存储目录筛选。
- 标注人：选择执行标注操作的账号名称。

5.1.5.4 数据特征分析

基于图片或目标框对图片的各项特征，如模糊度、亮度进行分析，并绘制可视化曲线，帮助处理数据集。

您还可以选择数据集的多个版本，查看其可视化曲线，进行对比分析。

背景信息

- 只有“图片”的数据集，且版本标注类型为“物体检测”和“图像分类”的数据集版本支持数据特征分析。
- 只有发布后的数据集支持数据特征分析。发布后的Default格式数据集版本支持数据特征分析。
- 数据特征分析的数据范围，不同类型的数据集，选取范围不同：
  - 对于标注任务类型为“物体检测”的数据集版本，当已标注样本数为0时，发布版本后，数据特征页签版本置灰不可选，无法显示数据特征。否则，显示已标注的图片的数据特征。
  - 对于标注任务类型为“图像分类”的数据集版本，当已标注样本数为0时，发布版本后，数据特征页签版本置灰不可选，无法显示数据特征。否则，显示全部的图片的数据特征。
- 数据集中的图片数量要达到一定量级才会具有意义，一般来说，需要有大约1000+的图片。
- “图像分类”支持分析指标有：“分辨率”、“图片高宽比”、“图片亮度”、“图片饱和度”、“清晰度”和“图像色彩的丰富程度”。“物体检测”支持所有的分析指标。目前ModelArts支持的所有分析指标请参见**支持分析指标及其说明**。

数据特征分析

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 选择对应的数据集，单击操作列的“数据特征”，进入数据集概览页的数据特征页面。

   您也可以在单击数据集名称进入数据集概览页后，单击“数据特征”页签进入。
3. 由于发布后的数据集不会默认启动数据特征分析，针对数据集的各个版本，需手动启动特征分析任务。在数据特征页签下，单击“特征分析”。

图 5-30 选择特征分析

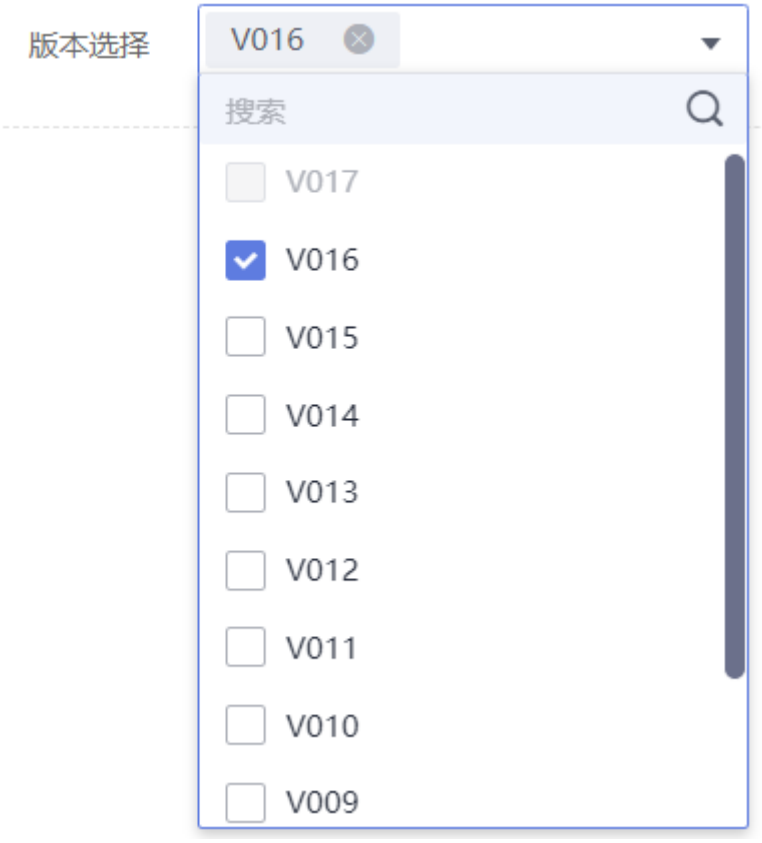
4. 在弹出的对话框中配置需要进行特征分析的数据集版本，然后单击“确定”启动分析。

   “版本选择”，即选择当前数据集的已发布版本。

图 5-31 启动数据特征分析任务

5. 数据特征分析任务启动后，需执行一段时间，根据数据量不同等待时间不同，请耐心等待。当您选择分析的版本出现在“版本选择”列表下，且可勾选时，即表示分析已完成。

图 5-32 可选择已执行特征分析的版本



6. 查看数据特征分析结果。

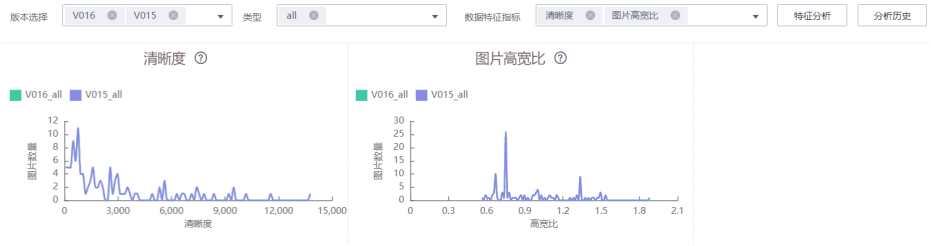
“版本选择”：在右侧下拉框中选择进行对比的版本。也可以只选择一个版本。

“类型”：选择需要分析的类型。支持“all”、“train”、“eval”和“inference”。

“数据特征指标”：在右侧下拉框中勾选需要展示的指标。详细指标说明请参见**支持分析指标及其说明**。

选择完成后，页面将自动呈现您选择对应版本及其指标数据，如**图5-33**所示，您可以根据呈现的图表了解数据分布情况，帮助您更好的处理您的数据。

图 5-33 数据特征分析



7. 查看分析任务的历史记录。

在数据特征分析后，您可以在“数据特征”页签下，单击右侧“任务历史”，可在弹出对话框中查看历史分析任务及其状态。

图 5-34 任务历史

任务历史

数据集版本	任务ID	创建时间	运行时间（h...	状态
V016	rzGaEY2lQDZ...	2020/06/01 1...	00:00:19	成功
V015	fOPPZbgwdY...	2020/06/01 1...	00:00:17	成功
V014	hfRjPLx03w3...	2020/06/01 1...	00:00:15	成功
V013	xwSatuRsHLu...	2020/06/01 1...	00:00:16	成功
V012	ARQfHizkrGR...	2020/06/01 1...	00:00:13	成功
V011	fsDmMsPrtv...	2020/06/01 1...	00:00:18	成功
V010	uocIdNmR2B...	2020/06/01 1...	00:00:13	成功
V009	EFSJPawWlzu...	2020/06/01 1...	00:00:19	成功
V008	QBBflfFszv5...	2020/06/01 1...	00:00:23	成功
V006	LvNT9UKBx8...	2020/06/01 1...	00:00:27	成功

10 总条数: 10 1

支持分析指标及其说明

表 5-20 分析指标列表

名称	说明	分析说明
分辨率 Resolution	图像分辨率。此处使用面积值作为统计值。	通过指标分析结果查看是否有偏移点。如果存在偏移点，可以对偏移点做resize操作或直接删除。
图片高宽比 Aspect Ratio	图像高宽比，即图片的高度/图片的宽度。	一般呈正态分布，一般用于比较训练集和真实场景数据集的差异。
图片亮度 Brightness	图片亮度，值越大代表观感上亮度越高。	一般呈正态分布，可根据分布中心判断数据集整体偏亮还是偏暗。可根据使用场景调整，比如使用场景是夜晚，图片整体应该偏暗。
图片饱和度 Saturation	图片的色彩饱和度，值越大表示图片整体色彩越容易分辨。	一般呈正态分布，一般用于比较训练集和真实场景数据集的差异。
清晰度 Clarity	图片清晰程度，使用拉普拉斯算子计算所得，值越大代表边缘越清晰，图片整体越清晰。	可根据使用场景判断清晰度是否满足需要。比如使用场景的数据采集来自高清摄像头，那么清晰度对应的需要高一些。可通过对数据集做锐化或模糊操作，添加噪声对清晰度做调整。
图像色彩的丰富程度 Colorfulness	横坐标：图像的色彩丰富程度，值越大代表色彩越丰富。 纵坐标：图片数量。	是观感上的色彩丰富程度，一般用于比较训练集和真实场景数据集的差异。
按单张图片中框的个数统计图片分布 Bounding Box Quantity	横坐标：单张图片中框的个数。 纵坐标：图片数量。	对模型而言一张图片的框个数越多越难检测，需要越多的这种数据用作训练。
按单张图片中框的面积标准差统计图片分布 Standard Deviation of Bounding Boxes Per Image	横坐标：单张图片中框的标准差。单张图片只有一个框时，标准差为0。标准差的值越大，表示图片中框大小不一程度越高。 纵坐标：图片数量。	对模型而言一张图中框如果比较多且大小不一，是比较难检测的，可以根据场景添加数据用作训练，或者实际使用没有这种场景可直接删除。

名称	说明	分析说明
按高宽比统计框数量的分布 Aspect Ratio of Bounding Boxes	横坐标：目标框的高宽比。 纵坐标：框数量（统计所有图片中的框）。	一般呈泊松分布，但与使用场景强相关。多用于比较训练集和验证集的差异，如训练集都是长方形框的情况下，验证集如果是接近正方形的框会有比较大影响。
按面积占比统计框数量的分布 Area Ratio of Bounding Boxes	横坐标：目标框的面积占比，即目标框的面积占整个图片面积的比例，越大表示物体在图片中的占比越大。 纵坐标：框数量（统计所有图片中的框）。	主要判断模型中使用的anchor的分布，如果目标框普遍较大，anchor就可以选择较大。
按边缘化程度统计框数量的分布 Marginalization Value of Bounding Boxes	横坐标：边缘化程度，即目标框中心点距离图片中心点的距离占图片总距离的比值，值越大表示物体越靠近边缘。（图片总距离表示以图片中心点为起点画一条经过标注框中心点的射线，该射线与图片边界交点到图片中心点的距离）。 纵坐标：框数量（统计所有图片中的框）。	一般呈正态分布。用于判断物体是否处于图片边缘，有一些只露出一部分的边缘物体，可根据需要添加数据集或不标注。
按堆叠度统计框数量的分布 Overlap Score of Bounding Boxes	横坐标：堆叠度，单个框被其他的框重叠的部分，取值范围为0～1，值越大表示被其他框覆盖的越多。 纵坐标：框数量（统计所有图片中的框）。	主要用于判断待检测物体的堆叠程度，堆叠物体一般对于检测难度较高，可根据实际使用需要添加数据集或不标注部分物体。
按亮度统计框数量的分布 Brightness of Bounding Boxes	横坐标：目标框的图片亮度，值越大表示越亮。 纵坐标：框数量（统计所有图片中的框）。	一般呈正态分布。主要用于判断待检测物体的亮度。在一些特殊场景中只有物体的部分亮度较暗，可以看是否满足要求。
按清晰度统计框数量的分布 Clarity of Bounding Boxes	横坐标：目标框的清晰度，值越大表示越清晰。 纵坐标：框数量（统计所有图片中的框）。	主要用于判断待检测物体是否存在模糊的情况。比如运动中的物体在采集中可能变得模糊，需要重新采集。

5.1.6 数据标注

由于模型训练过程需要大量有标签的数据，因此在模型训练之前需对没有标签的数据添加标签。您可以通过创建单人标注作业或团队标注作业对数据进行手工标注，或对任务启动智能标注添加标签，快速完成对图片的标注操作，也可以对已标注图片修改或删除标签进行重新标注。

- 人工标注：用户创建单人标注作业，对数据进行手工标注。
- 智能标注：在标注一定量的数据情况下，用户可以通过启动智能标注任务对数据进行自动标注，提高标注的效率。
- 团队标注：对于大批量的数据，用户可以通过创建团队标注作业，进行多人协同标注。

关于数据标注的详细信息，请参考。

5.1.7 数据发布

5.1.7.1 数据发布简介

ModelArts在数据准备过程中，针对同一数据源的数据，对不同时间处理或标注后的数据，按照版本进行区分方便后续模型构建和开发时选择对应的数据集版本进行使用。

关于数据集版本

- 针对刚创建的数据集（未发布前），无数据集版本信息，必须执行发布操作后，才能应用于模型开发或训练。
- 数据集版本，默认按V001、V002递增规则进行命名，您也可以在发布时自定义设置。
- 您可以将任意一个版本设置为当前目录，即表示数据集列表中进入的数据集详情，为此版本的数据及标注信息。
- 针对每一个数据集版本，您可以通过“存储路径”参数，获得此版本对应的Manifest文件格式的数据集。可用于导入数据或难例筛选操作。
- 表格数据集暂不支持切换版本。

5.1.7.2 发布数据版本

1. 登录，在左侧菜单栏中选择“数据管理> 数据集”，进入“数据集”管理页面
2. 在数据集列表中，单击操作列的“发布”。或者，您可以单击数据集名称，进入数据集“概览”页，在页面右上角单击“发布”。
3. 在“发布新版本”弹出框中，填写发布数据集的相关参数，然后单击“确定”。

图 5-35 发布数据集版本

表 5-21 发布数据集的参数说明

参数	描述
“版本名称”	默认按V001、V002递增规则进行命名，您也可以自定义版本名称。版本名称只能包含字母、数字、中划线或下划线。
“版本格式”	仅“表格”类型数据集支持设置版本格式，支持“CSV”和“CarbonData”两种。 **说明** 如果导出的CSV文件中存在以“=”“+”“-”和“@”开头的命令时，为了安全考虑，ModelArts会自动加上Tab键，并对双引号进行转义处理。
“数据切分”	仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。 默认不启用。启用后，需设置对应的训练验证比例。 输入“训练集比例”，数值只能是0~1区间内的数。设置好“训练集比例”后，“验证集比例”自动填充。“训练集比例”加“验证集比例”等于1。 **说明** 为确保训练模型的精度，建议将训练集比例设置为0.8或者0.9。 “训练集比例”即用于训练模型的样本数据比例；“验证集比例”即用于验证模型的样本数据比例。“训练验证比例”会影响训练模板的性能。
“描述”	针对当前发布的数据集版本的描述信息。

参数	描述
“开启难例属性”	仅“图像分类”和“物体检测”类型数据集支持难例属性。 默认不开启。启用后，会将此数据集的难例属性等信息写入对应的Manifest文件中。

数据集版本文件目录结构

由于数据集是基于OBS目录管理的，发布为新版本后，对应的数据集输出位置，也将基于新版本生成目录。

以图像分类为例，数据集发布后，对应OBS路径下生成，其相关文件的目录如下所示。

```
|-- user-specified-output-path
   |-- DatasetName-datasetId
      |-- annotation
         |-- VersionMame1
            |-- VersionMame1.manifest
         |-- VersionMame2
            ...
         |-- ...
```

以物体检测为例，如果数据集导入的是Manifest文件，在数据集发布后，其相关文件的目录结构如下。

```
|-- user-specified-output-path
   |-- DatasetName-datasetId
      |-- annotation
         |-- VersionMame1
            |-- VersionMame1.manifest
            |-- annotation
               |-- file1.xml
         |-- VersionMame2
            ...
         |-- ...
```

以视频标注为例，在数据集发布后，标注结果将标注结果文件（XML）存放在数据集输出目录下。

```
|-- user-specified-output-path
    |-- DatasetName-datasetId
       |-- annotation
          |-- VersionMame1
             |-- VersionMame1.manifest
             |-- annotations
               |-- images
                  |-- videoName1
                     |-- videoName1.timestamp.xml
                  |-- videoName2
                     |-- videoName2.timestamp.xml
         |-- VersionMame2
            ...
         |-- ...
```

视频标注的关键帧存在数据集的输入目录下。

```
|-- user-specified-input-path
    |-- images
      |-- videoName1
         |-- videoName1.timestamp.jpg
       |-- videoName2
         |-- videoName2.timestamp.jpg
```

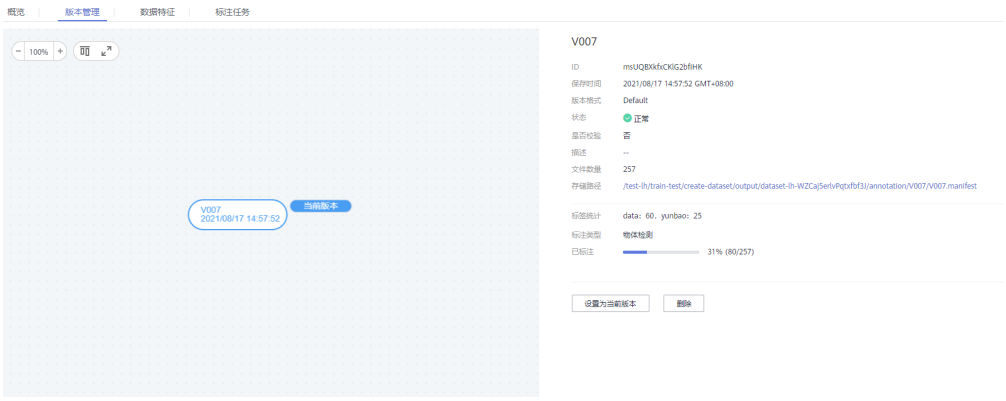
5.1.7.3 管理数据版本

在数据准备的过程中，用户可以将数据发布成多个版本对数据集进行管理。针对已发布生成的数据集版本，用户可以通过查看数据集的演进过程、切换版本、删除版本等操作，对数据集进行管理。

查看数据集演进过程

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集列表中，单击操作列的“更多 > 版本管理”，进入数据集“版本管理”页面。
   您可以查看数据集的基本信息，并在左侧查看版本演进信息及其发布时间。

图 5-36 查看数据集版本



设置当前版本

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集列表中，单击操作列的“更多 > 版本管理”，进入数据集“版本管理”页面。
3. 在“版本管理”页面中，选择对应的数据集版本，在数据集版本基本信息区域，单击“设置为当前版本”。设置完成后，版本名称右侧将显示为“当前版本”。

说明

只有状态为“正常”的版本，才能被设置为当前版本。

删除数据集版本

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集列表中，单击操作列的“更多 > 版本管理”，进入数据集“版本管理”页面。
3. 选择需删除的版本所在行，单击操作列的“删除”。在弹出的对话框中确认信息，然后单击“确定”完成删除操作。

说明

删除数据集版本不会删除原始数据，数据及其标注信息仍存在于对应的OBS目录下。但是，执行删除操作后，无法在ModelArts管理控制台清晰的管理数据集版本，请谨慎操作。

5.1.8 数据导出

5.1.8.1 数据导出简介

针对数据集中的数据，用户可以选中部分数据或者通过条件筛选出需要的数据，导出成新的数据集，或者将数据导出到指定的OBS目录下。用户可以通过任务历史查看数据导出的历史记录。

目前只有“图像分类”、“物体检测”、“图像分割”类型的数据集支持导出功能。

- “图像分类”只支持导出txt格式的标注文件。
- “物体检测”只支持导出Pascal VOC格式的XML标注文件。
- “图像分割”只支持导出Pascal VOC格式的XML标注文件以及Mask图像。

5.1.8.2 导出数据为新数据集

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集列表中，选择“图片”类型的数据集，单击数据集名称进入“数据集概览页”。
3. 在“数据集概览页”，单击右上角“导出 ”。在弹出的“导出”对话框中，填写相关信息，然后单击“确定”，开始执行导出操作。

   “导出方式”：选择新数据集。

   “名称”：新数据集名称。

   “保存路径”：表示新数据集的输入路径，即当前数据导出后存储的OBS路径。

   “输出路径”：表示新数据集的输出路径，即新数据集在完成标注后输出的路径。“输出路径”不能与“保存路径”为同一路径，且“输出路径”不能是“保存路径”的子目录。
4. 数据导出成功后，您可以前往您设置的保存路径，查看到存储的数据。当导出方式选择为新数据集时，在导出成功后，您可以前往“数据集”列表中，查看到新的数据集。
5. 在“数据集概览页”，单击右上角“导出历史”，在弹出的“任务历史”对话框中，可以查看该数据集之前的导出任务历史。

5.1.8.3 导出数据到 OBS

1. 登录，在左侧菜单栏中选择“数据管理>数据集”，进入“数据集”管理页面。
2. 在数据集列表中，选择“图片”类型的数据集，单击数据集名称进入“数据集概览页”。
3. 在“数据集概览页”，单击右上角“导出 ”。在弹出的“导出”对话框中，填写相关信息，然后单击“确定”，开始执行导出操作。

   “导出方式”：选择OBS。

   “保存路径”：即导出数据存储的路径。建议不要将数据存储至当前数据集所在的输入路径或输出路径。

图 5-37 导出到 OBS

4. 数据导出成功后，您可以前往您设置的保存路径，查看到存储的数据。
5. 在“数据集概览页”，单击右上角“导出历史”，在弹出的“任务历史”对话框中，可以查看该数据集之前的导出任务历史。

5.2 数据标注

5.2.1 数据标注简介

说明

数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

模型训练过程中需要大量已标注的数据，因此在模型训练之前需要进行数据标注作业。ModelArts为用户提供了标注数据的能力：

- 人工标注：用户创建单人标注作业，对数据进行手工标注。
- 智能标注：在标注一定量的数据情况下，用户可以通过启动智能标注任务对数据进行自动标注，提高标注的效率。
- 团队标注：对于大批量的数据，用户可以通过创建团队标注作业，进行多人协同标注。

人工标注

对于不同类型的数据，用户可以选择不同的标注类型。当前ModelArts支持如下类型的标注作业：

- 图片
  - 图像分类：识别一张图片中是否包含某种物体。
  - 物体检测：识别出图片中每个物体的位置及类别。
  - 图像分割：根据图片中的物体划分出不同区域。
- 音频
  - 声音分类：对声音进行分类。
  - 语音内容：对语音内容进行标注。

  - 语音分割：对语音进行分段标注。
- 文本
  - 文本分类：对文本的内容按照标签进行分类处理。
  - 命名实体：针对文本中的实体片段进行标注，如“时间”、“地点”等。
  - 文本三元组：针对文本中的实体片段和实体之间的关系进行标注。
- 视频

  视频标注：识别出视频中每个物体的位置及分类。目前仅支持mp4格式。

智能标注

除了人工标注外，ModelArts还提供了智能标注功能，快速完成数据标注，为您节省70%以上的标注时间。智能标注是指基于当前标注阶段的标签及图片学习训练，选中系统中已有的模型进行智能标注，快速完成剩余图片的标注操作。

目前只有“图像分类”和“物体检测”类型的数据集支持智能标注功能。

团队标注

数据标注任务中，一般由一个人完成，但是针对数据集较大时，需要多人协助完成。ModelArts提供了团队标注功能，可以由多人组成一个标注团队，针对同一个数据集进行标注管理。

团队标注功能当前仅支持“图像分类”、“物体检测”、“文本分类”、“命名实体”、“文本三元组”、“语音分割”类型的数据集。

不同类型数据集支持的功能列表

其中，不同类型的数据集，支持不同的功能，详细信息请参见表5-22。

表 5-22 不同类型数据集支持的功能

数据集类型	标注类型	人工标注	智能标注	团队标注
图片	图像分类	支持	支持	支持
	物体检测	支持	支持	支持
	图像分割	支持	-	-
音频	声音分类	支持	-	-
	语音内容	支持	-	-
	语音分割	支持	-	支持
文本	文本分类	支持	-	支持
	命名实体	支持	-	支持
	文本三元组	支持	-	支持
视频	视频标注	支持	-	-

数据集类型	标注类型	人工标注	智能标注	团队标注
自由格式	-	-	-	-
表格	-	-	-	-

5.2.2 人工标注

5.2.2.1 创建标注作业

由于模型训练过程需要大量有标签的数据，因此在模型训练之前需对没有标签的数据添加标签。您可以通过创建单人标注作业或团队标注作业对数据进行手工标注，或对任务启动智能标注添加标签，快速完成对图片的标注操作，也可以对已标注图片修改或删除标签进行重新标注。

标注作业支持的数据类型

对于不同类型的数据集，用户可以选择不同的标注任务，当前ModelArts支持如下类型的标注任务。

- 图片
  - 图像分类：识别一张图片中是否包含某种物体。
  - 物体检测：识别出图片中每个物体的位置及类别。
  - 图像分割：根据图片中的物体划分出不同区域。
- 音频
  - 声音分类：对声音进行分类。
  - 语音内容：对语音内容进行标注。
  - 语音分割：对语音进行分段标注。
- 文本
  - 文本分类：对文本的内容按照标签进行分类处理。
  - 命名实体：针对文本中的实体片段进行标注，如“时间”、“地点”等。
  - 文本三元组：针对文本中的实体片段和实体之间的关系进行标注。
- 视频

  视频标注：识别出视频中每个物体的位置及分类。目前仅支持mp4格式。

前提条件

在进行数据标注前，需要创建相应类型的数据集。

操作步骤

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

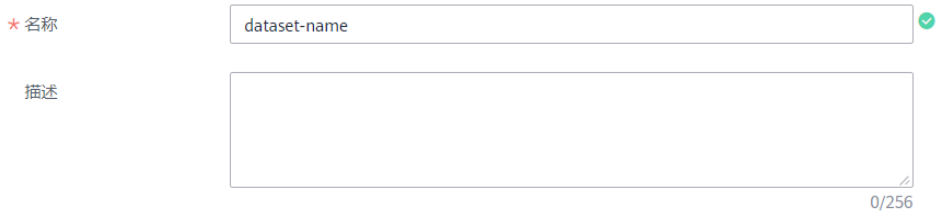
说明

数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在数据标注管理页面，单击页面右上角“创建标注作业”，进入“创建标注作业”页面，根据需求创建不同类型的标注作业。

   a. 填写标注作业基本信息，标注作业的“名称”和“描述”。

   图 5-38 标注作业基本信息

   b. 根据您的需求，选择“标注场景”和“标注类型”。

   图 5-39 选择标注场景和标注类型

   c. 针对不同类型的标注作业，需填写参数不同，请参考如下类型标注作业对应的参数介绍。

   - 图片（图像分类、物体检测、图像分割）
   - 音频（声音分类、语音内容、语音分割）
   - 文本（文本分类、命名实体、文本三元组）
   - 视频

   d. 参数填写无误后，单击页面右下角“创建”。

   标注作业创建完成后，系统自动跳转至数据标注管理页面，针对创建好的标注作业，您可以执行智能标注、发布、修改和删除等操作。

图片（图像分类、物体检测、图像分割）

图 5-40 图像分类和物体检测类型的参数

表 5-23 图片类型标注作业的详细参数

参数名称	说明
数据集名称	选择支持当前标注类型的数据集。
添加标签集	• **设置标签名称**：在标签名称文本框中，输入标签名称。长度为1～1024字符。 • **添加标签**：单击“添加标签”可增加多个标签。 • **设置标签颜色**：“物体检测”和“图像分割”类型标注作业需设置此参数。在每个标签右侧的标签颜色区域下，可在色板中选择颜色，或者直接输入十六进制颜色码进行设置。 • **设置标签属性**：针对“物体检测”类型标注作业，在设置完标签颜色后，可在右侧单击加号，增加对应的标签属性。标签属性用于区分同一标签物体的不同属性。例如，黄色小猫、黑色小猫。标签为cat，颜色为不同的标签属性。
启用团队标注	选择是否启用团队标注。图像分割暂不支持团队标注，当选择图像分割类型时，界面不显示此参数。 启用团队标注功能，需填写对应的团队标注任务“类型”，同时选择对应的“标注团队”及参与标注的“团队成员”。参数详细介绍请参见**创建团队标注任务**。 在启用“团队标注”前，需确保您已经在“标注团队”管理页面，添加相应的团队以及成员。如果没有标注团队，可直接从界面链接跳转至“标注团队”页面，添加您的团队并为其添加成员。详细指导请参见**添加团队**。 启用团队标注功能的数据集，在创建完成后，可以在“标注类型”中看到“团队标注”的标识。

音频（声音分类、语音内容、语音分割）

图 5-41 声音分类、语音内容、语音分割类型的参数

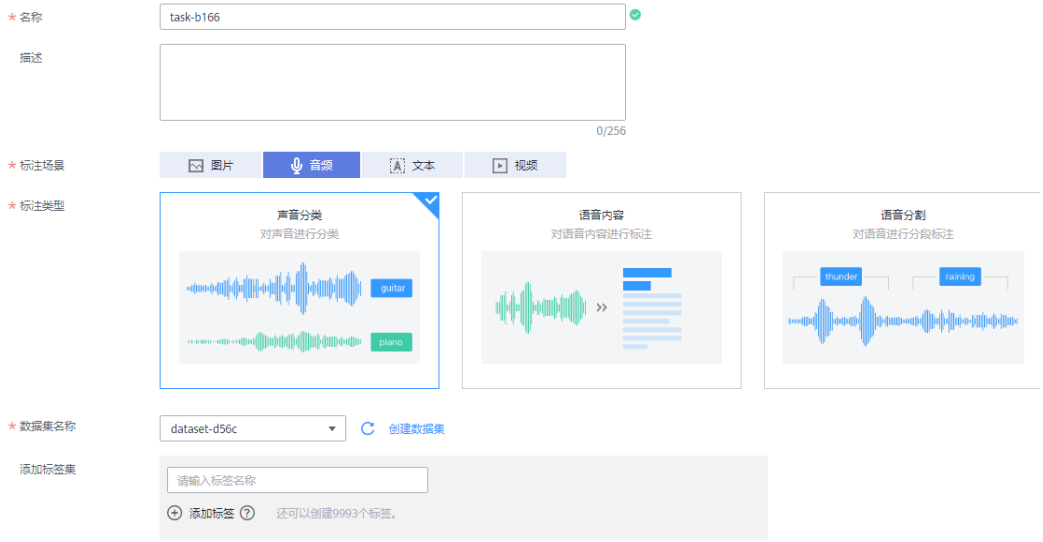


表 5-24 音频类型标注作业的详细参数

参数名称	说明
数据集名称	选择支持当前标注类型的数据集。
添加标签集（声音分类）	“声音分类”类型的标注作业可以添加标签集。 • **设置标签名称**：在标签名称文本框中，长度为1～1024字符。 • **添加标签**：单击“添加标签”可增加多个标签。
标签管理（语音分割）	“语音分割”类型的标注作业，支持标签管理。 • **单标签** 单标签适用于一段音频标注只有一种类别的音频，通常标注一个标签。 – **设置标签名称**：在“标签名”列输入标签名称。长度为1～1024字符。 – **设置标签颜色**：在“标签颜色”列设置标签颜色。可在色板中选择颜色，或者直接输入十六进制颜色码进行设置。 • **多标签** 多标签适用于多维度标注，例如在一段音频标注噪音与人说话的声音两种类别，其中说话的声音还可以标注为不同人的声音。单击“新建标签类别”可添加多个标签类别，一个标签类别可以包含多个标签。“标签类别”和“标签名”只能是中文、字母、数字、英文句号、下划线或中划线组成的合法字符串。长度为1～256字符。 – **设置标签类别**：在“标签类别”输入标签类别的名称。 – **设置标签名称**：在“标签名”输入标签名称。 – **添加标签**：单击“添加标签”可增加多个标签。

参数名称	说明
启用语音内容标注（语音分割）	仅“语音分割”类型数据集支持设置，默认关闭。如果启用此功能，支持针对语音内容进行标注。
启用团队标注（语音分割）	仅“语音分割”类型支持团队标注，因此选择创建语音分割类型时，支持设置是否启用团队标注。 启用团队标注功能，需填写对应的团队标注任务“类型”，同时选择对应的“标注团队”及参与标注的“团队成员”。参数详细介绍请参见**创建团队标注任务**。 在启用“团队标注”前，需确保您已经在“标注团队”管理页面，添加相应的团队以及成员。如果没有标注团队，可直接从界面链接跳转至“标注团队”页面，添加您的团队并为其添加成员。详细指导请参见**添加团队**。 启用团队标注功能的数据集，在创建完成后，可以在“标注类型”中看到“团队标注”的标识。

文本（文本分类、命名实体、文本三元组）

图 5-42 文本分类、命名实体、文本三元组类型的参数

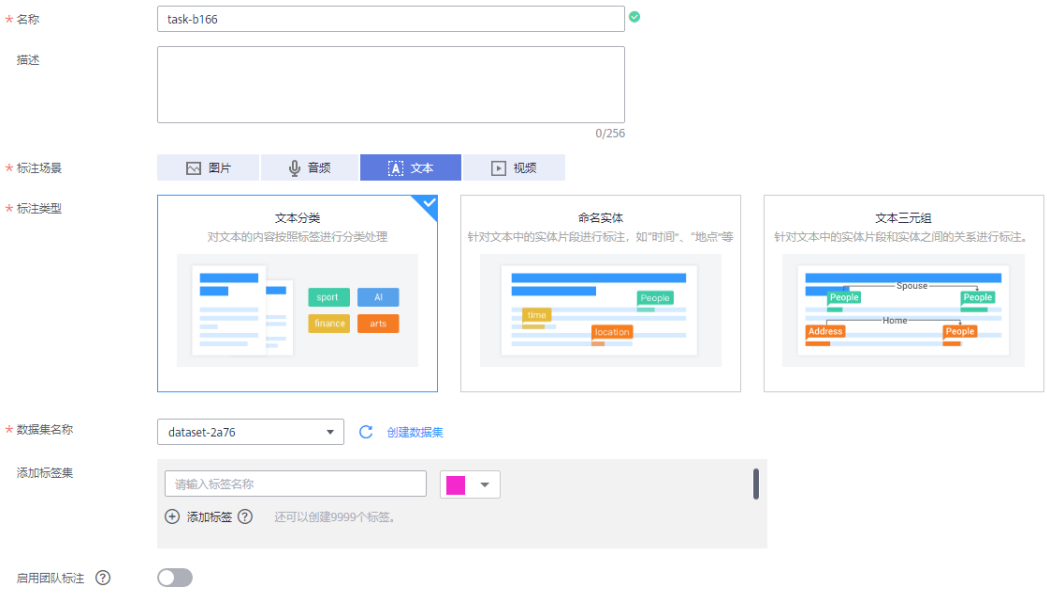
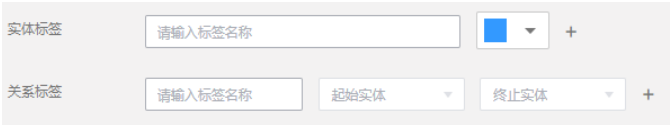


表 5-25 文本类型标注作业的详细参数

参数名称	说明
数据集名称	选择支持当前标注类型的数据集。

参数名称	说明
添加标签集（文本分类、命名实体）	• **设置标签名称**：在标签名称文本框中，输入标签名称。长度为1～1024字符。 • **添加标签**：单击“添加标签”可增加多个标签。 • **设置标签颜色**：在每个标签右侧的标签颜色区域下，可在色板中选择颜色，或者直接输入十六进制颜色码进行设置。
添加标签集（文本三元组）	针对“文本三元组”类型的数据集，需要设置实体标签和关系标签。 • **实体标签**：需设置标签名以及标签颜色。可在颜色区域右侧单击加号增加多个标签。 • **关系标签**：关系标签为两个实体之间的关系。需设置起始实体和终止实体，您需要先添加至少2个实体标签后，再添加关系标签。 
启用团队标注	选择是否启用团队标注。 启用团队标注功能，需填写对应的团队标注任务“类型”，同时选择对应的“标注团队”及参与标注的“团队成员”。参数详细介绍请参见**创建团队标注任务**。 在启用“团队标注”前，需确保您已经在“标注团队”管理页面，添加相应的团队以及成员。如果没有标注团队，可直接从界面链接跳转至“标注团队”页面，添加您的团队并为其添加成员。详细指导请参见**添加团队**。 启用团队标注功能的数据集，在创建完成后，可以在“标注类型”中看到“团队标注”的标识。

视频

图 5-43 视频类型的参数

表 5-26 视频类型标注作业的详细参数

参数名称	说明
数据集名称	选择支持当前标注类型的数据集。
添加标签集	• **设置标签名称**：在标签名称文本框中，输入标签名称。长度为1～1024字符。 • **添加标签**：单击“添加标签”可增加多个标签。 • **设置标签颜色**：在每个标签右侧的标签颜色区域下，可在色板中选择颜色，或者直接输入十六进制颜色码进行设置。

5.2.2.2 图片标注

5.2.2.2.1 图像分类

由于模型训练过程需要大量有标签的图片数据，因此在模型训练之前需对没有标签的图片添加标签。您可以通过手工标注或智能一键标注的方式添加标签，快速完成对图片的标注操作，也可以对已标注图片修改或删除标签进行重新标注。

针对图像分类场景，开始标注前，您需要了解：

- 图片标注支持多标签，即一张图片可添加多个标签。
- 标签名是由中文、大小写字母、数字、中划线或下划线组成，且不超过1024位的字符串。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型。基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

   图 5-44 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“全部”、“未标注”或“已标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

筛选数据

在标注作业详情页面，默认展示作业中全部未标注的数据，您可以在“全部”、“未标注”或“已标注”页签下，在筛选条件区域，单击 ，添加筛选条件，快速过滤出您想要查看的数据。

支持的筛选条件如下所示，您可以设置一个或多个选项进行筛选。

- 难例集：难例或非难例。
- 标签：您可以选择全部标签，或者基于您指定的标签，选中其中一个或多个。
- 文件名或目录：根据文件名称或者文件存储目录筛选。
- 标注人：选择执行标注操作的账号名称。

标注图片（手工标注）

在标注作业详情页中，展示了此数据集中“全部”、“未标注”和“已标注”的图片，默认显示“未标注”的图片列表。单击图片，即可进行图片的预览，对于已标注图片，预览页面下方会显示该图片的标签信息。

1. 在“未标注”页签，勾选需进行标注的图片。
   - 手工点选：在图片列表中，单击勾选图片左上角的选择框，进入选择模式，表示图片已勾选。可勾选同类别的多个图片，一起添加标签。
   - 批量选中：如果图片列表的当前页，所有图片属于一种类型，可以在图片列表的右上角单击“选择当前页”，则当前页面所有的图片将选中。
2. 为选中图片添加标签。
   a. 在右侧的“添加标签”区域中，单击“标签名”右侧的文本框中设置标签。

      单击“标签名”右侧的文本框，然后从下拉列表中选择已有的标签。如果已有标签无法满足要求时，直接在文本框中添加新标签。

   b. 单击“确定”。此时，选中的图片将被自动移动至“已标注”页签，且在“未标注”和“全部”页签中，标签的信息也将随着标注步骤进行更新，如增加的标签名称、各标签对应的图片数量。

说明

如果您还不太清楚如何进行标注，可参考数据集详情页面的“标注样例说明”完成标注。

1. 登录ModelArts管理控制台，选择数据管理>数据标注进入数据标注页。
2. 在“我创建的”或“我参与的”页签下，找到您需要标注的数据集。
3. 单击数据集名称，进入标注详情页。（默认直接进入“未标注”页签）。
4. 在标注详情页右上角单击“标注样例说明”。

图 5-45 标注样例说明

查看已标注图片

在标注任务详情页，单击“已标注”页签，您可以查看已完成标注的图片列表。图片缩略图下方默认呈现其对应的标签，您也可以勾选图片，在右侧的“选中文件标签”中了解当前图片的标签信息。

修改标注

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

- **基于图片修改**

  在标注作业详情页面，单击“已标注”页签，然后在图片列表中选中待修改的图片（选择一个或多个）。在右侧标签信息区域中对图片信息进行修改。

  修改标签：在“选中文件标签”区域中，单击操作列的编辑图标，然后在文本框中输入正确的标签名，然后单击确定图标完成修改。

  删除标签：在“选中文件标签”区域中，单击操作列的删除图标删除该标签。此操作仅删除选中图片中的标签。

  图 5-46 编辑标签

- **基于标签修改**
  - 在标注作业详情页，单击右侧区域的“标签管理”，显示全部标签列表。
    - 修改标签：单击操作列的“修改”，然后在弹出的对话框中输入修改后的标签名，然后单击“确定”完成修改。修改后，之前添加了此标签的图片，都将被标注为新的标签名称。
    - 删除标签：单击操作列“删除”，之前添加了此标签的图片，都将删除此标签。

图 5-47 标签管理

图 5-48 全部标签的信息

- 单击标注作业操作列的“标签”，可跳转至标签管理页。
  - 单击操作列的“修改”，即可完成标签的修改。
  - 单击操作列的“删除”，即可删除该标签。

添加数据

除了同步数据集中的新数据外，您还可以在标注作业中，直接添加图片，用于数据标注。添加的数据将先导入至标注任务关联的数据集中，然后标注任务会自动同步数据集中最新的数据。

1. 在标注作业详情页面，单击“全部”、“已标注”或“未标注”页签，然后单击左上角“添加数据”，选择添加数据。

图 5-49 添加数据

2. 在弹出的导入对话框中，选择数据来源和导入方式，选择导入的数据路径和数据标注状态。

3. 在导入对话框中，单击“确定”，完成添加数据的操作。

   您添加的图片将自动呈现在“全部”的图片列表中，也可单击“添加数据>查看历史记录”，进入“任务历史”界面，可查看相应的导入历史。

图 5-50 查看历史数据

删除图片

通过数据删除操作，可将需要丢弃的图片数据快速删除。

在“全部”、“未标注”或“已标注”页面中，依次选中需要删除的图片，或者选择“选择当前页”选中该页面所有图片，然后单击“删除”。在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成图片删除操作。

图 5-51 删除图片

其中，被选中的图片，其左上角将显示为勾选状态。如果当前页面无选中图片时，“删除”按钮为灰色，无法执行删除操作。

说明

如果勾选了“同时删除OBS源文件”，删除图片操作将删除对应OBS目录下存储的图片，此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

标注人员管理

如果您创建的标注作业，开启了团队标注，“标注人员管理”页面中可查看团队标注作业的标注详情。添加、修改或删除标注成员。

1. 登录“数据管理>数据标注”，在“我创建的”页签下可查看所有的标注作业列表。
2. 在作业列表的“名称”列，根据标注作业名称找到对应的团队标注作业。（团队标注作业的名称后带有标识。）
3. 单击作业操作列的“更多>标注人员管理”。或单击作业名称进入作业详情，继续单击右上角“团队标注>标注人员管理”，进入成员管理页面。

图 5-52 进入标注人员管理页（1）

图 5-53 进入标注人员管理页（2）

- 添加成员：

  单击页面“添加成员”，选择成员名称，单击确定。

  在操作列，选择“发送邮件”，可将该标注任务以邮件的方式发送至该标注成员。
- 修改成员信息：

单击操作列的“修改”，可修改该成员的角色。

- 删除标注成员：
  单击操作列的“删除”可删除该标注成员的所有信息。

5.2.2.2.2 物体检测

由于模型训练过程需要大量有标签的图片数据，因此在模型训练之前需对没有标签的图片添加标签。您可以通过手工标注或智能一键标注的方式添加标签，快速完成对图片的标注操作，也可以对已标注图片修改或删除标签进行重新标注。

针对物体检测场景，开始标注前，您需要了解：

- 图片中所有目标物体都要标注。
- 目标物体清晰无遮挡的，必须画框。
- 画框仅包含整个物体。框内包含整个物体的全部，画框边缘不可与待标注的物体的边缘轮廓相交，在此基础之上确保边缘和待标注物体间不要留着空隙，避免背景对模型训练造成干扰。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

   图 5-54 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“全部”、“未标注”或“已标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

筛选数据

在标注作业详情页面，默认展示作业中“未标注”的数据，您可以在“全部”、“未标注”或页签下，在筛选条件区域，单击“ ”，添加筛选条件，快速过滤出您想要查看的数据。

支持的筛选条件如下所示，您可以设置一个或多个选项进行筛选。

- 难例集：难例或非难例。
- 标签：您可以选择全部标签，或者基于您指定的标签，选中其中一个或多个。
- 文件名或目录：根据文件名称或者文件存储目录筛选。
- 标注人：选择执行标注操作的账号名称。

标注图片（手工标注）

标注作业详情页中，展示了此数据集中“全部”“未标注”和“已标注”的图片，默认显示“未标注”的图片列表。

1. 在“未标注”页签图片列表中，单击图片，自动跳转到标注页面。在标注页面，常用按钮的使用可参见**表5-28**。
2. 在页面上方工具栏选择合适的标注图形，系统默认的标注图形为矩形。本示例使用矩形工具进行标注。

说明

页面左侧可以选择多种形状对图片进行标注。标注第一张图片时，一旦选择其中一种，其他图片默认使用此形状进行标注，用户可以根据自己需求再进行切换。

表 5-27 支持的标注框

图标	使用说明
	矩形。也可使用快捷键【1】。鼠标单击标注对象左上角边缘位置，界面将出现矩形框，移动鼠标使得矩形框覆盖标注对象，然后单击完成标注。

图标	使用说明
	多边形。也可使用快捷键【2】。在标注对象所在范围内，鼠标左键单击完成一个点的标注，沿着物体的形状边缘，通过鼠标指定多个点，最终单击到第一个点的位置，由所有的点组成一个多边形形状。使得需标注的对象在此标注框内。
	圆形。也可使用快捷键【3】。在标注对象中，选择物体的中心点位置，单击鼠标确定圆心，然后移动鼠标，使得圆形框覆盖标注对象，然后再单击鼠标完成标注。
	直线。也可使用快捷键【4】。在标注对象中，选择物体的起始点，单击鼠标确定直线的起始点，然后使得直线覆盖标注对象，然后再单击鼠标完成标注。
	虚线。也可使用快捷键【5】。在标注对象中，选择物体的起始点，单击鼠标确定虚线的起始点，然后使得虚线覆盖标注对象，然后再单击鼠标完成标注。
	点。也可使用快捷键【6】。单击图片中的物体所在位置，即可完成点的标注。

3. 在弹出的添加标签文本框中，直接输入新的标签名，在文本框前面选中标签颜色，然后单击“添加”。如果已存在标签，从下拉列表中选择已有的标签，单击“添加”。

   逐步标注图片中所有物体所在位置，一张图片可添加多个标签。完成一张图片标注后，可单击图片右上角来切换下一张（也可使用快捷键【D】直接切换），快速选中其他未标注的图片，然后在标注页面中执行标注操作。

4. 单击页面上方“返回数据标注预览”查看标注信息，在弹框中单击“确定”保存当前标注并离开标注页面。

   选中的图片被自动移动至“已标注”页签，且在“未标注”和“全部”页签中，标签的信息也将随着标注步骤进行更新，如增加的标签名称、标签对应的图片数量。

表 5-28 标注界面的常用按钮

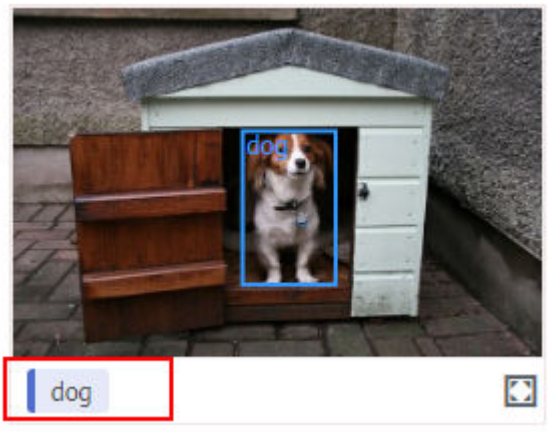
按钮图标	功能说明
	撤销上一个操作。也可使用快捷键【Ctrl+Z】
	重做上一个操作。也可使用快捷键【Ctrl+Shift+Z】
	放大图片。也可以使用滚轮进行放大。
	缩小图片。也可以使用滚轮进行缩小。
	删除当前图片中的所有标注框。也可使用快捷键【Shift+Delete】
	显示或隐藏标注框。只有在已标注图片中可使用此操作。也可使用快捷键【Shift+H】

按钮图标	功能说明
	拖动，可将标注好的框拖动至其他位置，也可以选择框的边缘，更改框的大小。也可使用【X+鼠标左键】
	复位，与上方拖动为同组操作，当执行了拖动后，可以单击复位按钮快速将标注框恢复为拖动前的形状和位置。也可使用快捷键【Esc】

查看已标注图片

在标注作业详情页，单击“已标注”页签，您可以查看已完成标注的图片列表。可在每张图片的下方显示当前图片的标签信息。

图 5-55 标签信息

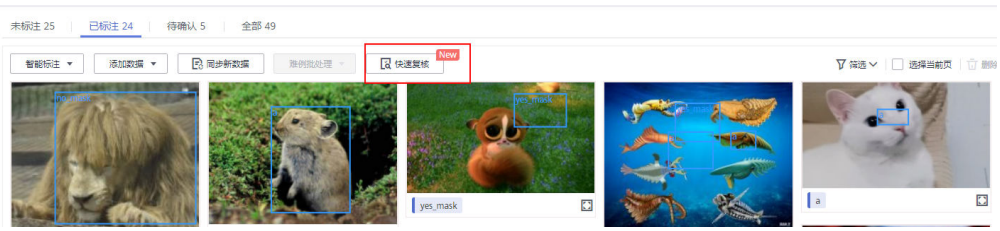


快速复核

当前的标注作业无法实现批量复核，如果有某一样本的标签修改或者删除，只能进入到标注页面详情进行，操作繁琐。为了简化用户操作，实现此功能，用户可以批量进行标注信息的审核或者修改，提升用户效率。

1. 登录ModelArts管理控制台，在总览页选择“数据管理>数据标注”，进入“我创建的”页签，在右上方的作业类型中下拉选择对应类型的标注作业。（仅物体检测与图像分割支持快速复核功能）
2. 在物体检测类型的标注作业列表，单击标注作业名称，进入标注详情页。
3. 单击“已标注”页签的“快速复核”，进入复核页面，对标注结果进行确认。

图 5-56 进入快速复核



4. 快速复核，支持您按照标签批量复核。
   a. 在复核页面，单击“按照标签过滤”，选择需要复核的标签类型图片。
   b. 在当前页面，您可以选择对当前的标签类型的图片，按照标注面积排序，或按照宽高比排序。
   c. 依次单击需要复核的图片，在标注页面拖动图片的标注框，即可重新完成标注。（修改后的图片会带有“已修改”的信息。）
   d. 您也可以选中需要删除标签的图片，单击右上方的 ，删除原始的标注信息。（删除后的图片会带有“已删除”的信息）

图 5-57 已修改

图 5-58 已删除

   e. 您也可以对当前已标注的图片标签信息进行修改。
      i. 选中待复核的图片，单击右侧的“全部标签”区域的 按钮。
      ii. 输入新的标签，单击“确定”。

图 5-59 全部标签

图 5-60 添加标签

5. 标注页面和标签都修改完成后，单击“应用所有修改”，在弹出的对话框单击“确定”，自动返回至标注概览页，同时会覆盖原始的标注数据。

图 5-61 应用所有修改

6. 如果您对修改后的数据不满意，也可以单击“放弃修改”选择放弃本次修改，保持原有的标注数据。

图 5-62 放弃修改

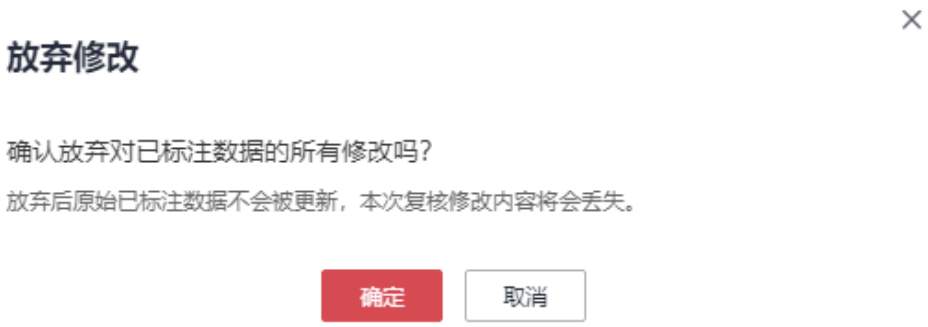


表 5-29 快速复核界面的常用按钮

按钮图标	功能说明
	删除原有的标注数据，删除后可重新标注。
	还原本页所有操作至未复核页面。
	撤销上一步操作。
	重做上一步操作。

修改标注

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

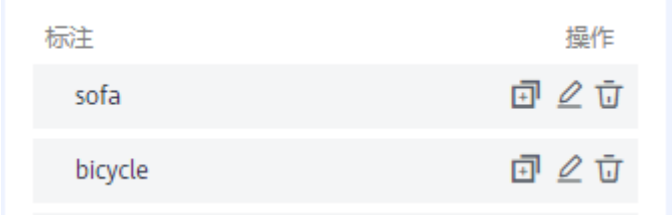
- **基于图片修改**

  在标注作业详情页面，单击选择“已标注”页签，然后在图片列表中选中待修改的图片，单击图片跳转到标注页面，在右侧标签信息区域中对图片信息进行修改。

  - 修改标签：“标注”区域中，单击编辑图标，在文本框中输入正确的标签名，然后单击确定图标完成修改。也可以单击标签，在图片标注区域，调整标注框的位置和大小，完成调整后，右键选择“修改”，填入修改后的标签名称，单击“修改”即可保存修改。
  - 删除标签：在“标注”区域中，单击删除图标即可删除此图片中的标签。

    标签删除后，单击页面左上角的“返回数据标注预览”离开标注页面，在弹出对话框中保存标注。图标的标签全部删除后，该图片会重新回到“未标注”页签。

图 5-63 编辑物体检测标签



- **基于标签修改**
  - 在标注作业详情页中，单击右侧的“标签管理”页签，在标签管理详情页，显示全部标签的信息。
    - 修改标签：单击操作列的“修改”，然后在弹出的对话框中输入修改后的标签和颜色，然后单击“确定”完成修改。修改后，之前添加了此标签的图片，都将被标注为新的标签名称。
    - 删除标签：单击操作列的“删除”，或选中需要删除的标签名称，单击上方的“删除”可删除此标签。

图 5-64 进入标签管理

图 5-65 全部标签

  - 单击标注作业操作列的“标签”，也可跳转至标签管理页。

图 5-66 从标注作业列表进入标签管理

- 单击操作列的“修改”，即可完成标签的修改。
- 单击操作列的“删除”，即可删除该标签。

添加数据

除了同步数据集中的新数据外，您还可以在标注作业中，直接添加图片，用于数据标注。添加的数据将先导入至标注任务关联的数据集中，然后标注任务会自动同步数据集中最新的数据。

1. 在标注作业详情页面，单击“全部”、“已标注”或“未标注”页签，然后单击左上角“添加数据”，选择添加数据。

图 5-67 添加数据

2. 在弹出的导入对话框中，选择数据来源和导入方式，选择导入的数据路径和数据标注状态。

图 5-68 添加图片

3. 在导入对话框中，单击“确定”，完成添加数据的操作。

   您添加的图片将自动呈现在“全部”的图片列表中，也可单击“添加数据>查看历史记录”，进入“任务历史”界面，可查看相应的导入历史。

图 5-69 查看历史数据

删除图片

通过数据删除操作，可将需要丢弃的图片数据快速删除。

在“全部”、“未标注”或“已标注”页面中，依次选中需要删除的图片，或者选择“选择当前页”选中该页面所有图片，然后单击“删除”。在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成图片删除操作。

图 5-70 删除图片

其中，被选中的图片，其左上角将显示为勾选状态。如果当前页面无选中图片时，“删除”按钮为灰色，无法执行删除操作。

说明

如果勾选了“同时删除OBS源文件”，删除图片操作将删除对应OBS目录下存储的图片，此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

标注人员管理

如果您创建的标注作业，开启了团队标注，“标注人员管理”页面中可查看团队标注作业的标注详情。添加、修改或删除标注成员。

1. 登录“数据管理>数据标注”，在“我创建的”页签下可查看所有的标注作业列表。

2. 在作业列表的“名称”列，根据标注作业名称找到对应的团队标注作业。（团队标注作业的名称后带有 标识。）
3. 单击作业操作列的“更多>标注人员管理”。或单击作业名称进入作业详情，继续单击右上角“团队标注>标注人员管理”，进入成员管理页面。

图 5-71 进入标注人员管理页（1）

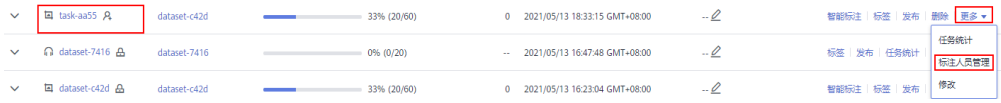


图 5-72 进入标注人员管理页（2）

- 添加成员：

  单击页面“添加成员”，选择成员名称，单击确定。

  在操作列，选择“发送邮件”，可将该标注任务以邮件的方式发送至该标注成员。
- 修改成员信息：

  单击操作列的“修改”，可修改该成员的角色。
- 删除标注成员：

  单击操作列的“删除”可删除该标注成员的所有信息。

5.2.2.2.3 图像分割

由于模型训练过程需要大量有标签的图片数据，因此在模型训练之前需对没有标签的图片添加标签。您可以通过在ModelArts控制台进行标注，也可以对已标注图片修改或删除标签进行重新标注。

针对图像分割场景，开始标注前，您需要了解：

- 图片中需要提取轮廓的物体都要标注。
- 支持使用多边形标注。
  - 多边形标注，根据目标物体的轮廓绘制多边形。
- 多边形标注时，标注框必须在图片范围内，超出图片将导致后续作业异常。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型。基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

   图 5-73 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“全部”、“未标注”或“已标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

筛选数据

在标注作业详情页面，默认展示作业中“未标注”数据，您可以在“全部”、“未标注”或页签下，在筛选条件区域，单击 ，添加筛选条件，快速过滤出您想要查看的数据。

支持的筛选条件如下所示，您可以设置一个或多个选项进行筛选。

- 难例集：难例或非难例。
- 标签：您可以选择全部标签，或者基于您指定的标签，选中其中一个或多个。
- 文件名或目录：根据文件名称或者文件存储目录筛选。
- 标注人：选择执行标注操作的账号名称。

标注图片（手工标注）

标注作业详情页中，展示了此标注作业中**“全部”**、“未标注”和“已标注”的图片，默认显示“未标注”的图片列表。

1. 在“未标注”页签图片列表中，单击图片，自动跳转到标注页面。在标注页面，常用按钮的使用可参见**表5-31**。
2. 选择标注方式。

   在标注页面，上方工具栏提供了常用的**标注方式**及**常用按钮**，系统默认的标注方式为多边形标注。

   说明

   标注第一张图片时，一旦选择其中一种，其他所有图片都需要使用此方式进行标注。

图 5-74 工具栏

表 5-30 标注方式

图标	使用说明
	多边形。在标注对象所在范围内，鼠标左键单击完成一个点的标注，沿着物体的形状边缘，通过鼠标指定多个点，最终单击到第一个点的位置，由所有的点组成一个多边形形状。使得需标注的对象在此标注框内。

表 5-31 工具栏常用按钮

按钮图标	功能说明
	撤销上一个操作。
	重做上一个操作。

按钮图标	功能说明
	放大图片。
	缩小图片。
	删除当前图片中的所有标注框。
	显示或隐藏标注框。只有在已标注图片中可使用此操作。
	拖动，可将标注好的框拖动至其他位置，也可以选择框的边缘，更改框的大小。
	复位，与上方拖动为同组操作，当执行了拖动后，可以单击复位按钮快速将标注框恢复为拖动前的形状和位置。
	全屏显示标注的图片。

3. 标注物体。

   完成一张图片标注后，可单击图片下方  展开缩略图，查看图片列表，快速选中其他未标注的图片，然后在标注页面中执行标注操作。

4. 单击页面上方“返回数据标注预览”查看标注信息，在弹框中单击“确定”保存当前标注并离开标注页面。

   选中的图片被自动移动至“已标注”页签，且在“未标注”和“全部”页签中，标签的信息也将随着标注步骤进行更新，如增加的标签名称、标签对应的图片数量。

查看已标注图片

在标注作业详情页中，单击“已标注”页签，您可以查看已完成标注的图片列表。单击图片进入图片标注详情，可在右侧的“当前文件标签”中了解当前图片的标签信息。

快速复核

当前的标注作业无法实现批量复核，如果有某一样本的标签修改或者删除，只能进入到标注页面详情进行，操作繁琐。为了简化用户操作，实现此功能，用户可以批量进行标注信息的审核或者修改，提升用户效率。

1. 登录ModelArts管理控制台，在总览页选择“数据管理>数据标注”，进入“我创建的”页签，在右上方的作业类型中下拉选择对应类型的标注作业。（仅物体检测与图像分割支持快速复核功能）
2. 在物体检测类型的标注作业列表，单击标注作业名称，进入标注详情页。
3. 单击“已标注”页签的“快速复核”，进入复核页面，对标注结果进行确认。

图 5-75 进入快速复核

4. 快速复核，支持您按照标签批量复核。

   a. 在复核页面，单击“按照标签过滤”，选择需要复核的标签类型图片。

   b. 在当前页面，您可以选择对当前的标签类型的图片，按照标注面积排序，或按照宽高比排序。

   c. 依次单击需要复核的图片，在标注页面拖动图片的标注框，即可重新完成标注。（修改后的图片会带有“已修改”的信息。）

   d. 您也可以选中需要删除标签的图片，单击右上方的 ，删除原始的标注信息。（删除后的图片会带有“已删除”的信息）

图 5-76 已修改

图 5-77 已删除

   e. 您也可以对当前已标注的图片标签信息进行修改。

      i. 选中待复核的图片，单击右侧的“全部标签”区域的 按钮。

      ii. 输入新的标签，单击“确定”。

图 5-78 全部标签

图 5-79 添加标签

5. 标注页面和标签都修改完成后，单击“应用所有修改”，在弹出的对话框单击“确定”，自动返回至标注概览页，同时会覆盖原始的标注数据。

图 5-80 应用所有修改

6. 如果您对修改后的数据不满意，也可以单击“放弃修改”选择放弃本次修改，保持原有的标注数据。

图 5-81 放弃修改

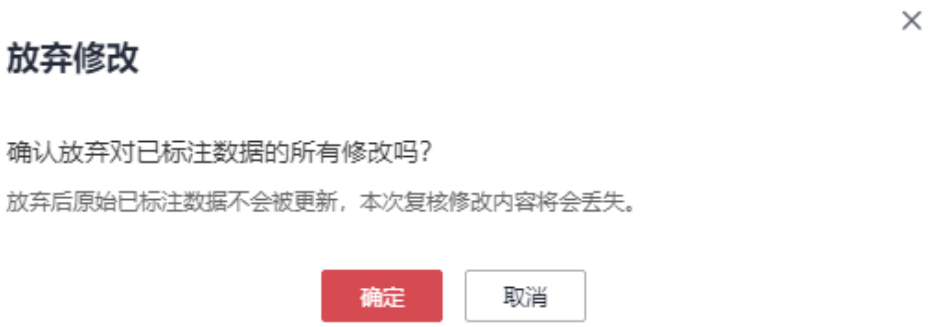


表 5-32 快速复核界面的常用按钮

按钮图标	功能说明
	删除原有的标注数据，删除后可重新标注。
	还原本页所有操作至未复核页面。
	撤销上一步操作。
	重做上一步操作。

修改标注信息

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

在数据标注详情页面，单击“已标注”页签，然后在图片列表中选中待修改的图片，单击图片跳转到标注页面，在右侧标签信息区域中单击此图片已添加的标注信息。

- 修改标签：“标注”区域中，单击编辑图标，在弹出框中输入正确的标签名或标签颜色，然后单击 完成修改。也可以单击标签，在图片标注区域，调整标注框的位置和大小，完成调整后，单击其他标签即可保存修改。
- 删除标签：在“标注”区域中，单击删除图标即可删除此图片中的标签。图片的标签全部删除后，该图片会重新回到“未标注”页签。

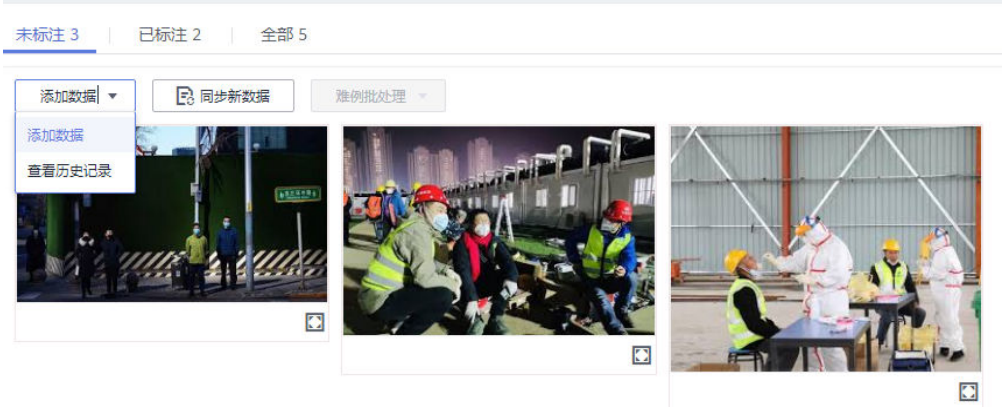
标注信息修改后，单击页面左上角的“返回数据标注预览”离开标注页面，在弹出对话框中单击“确定”保存修改。

添加数据

除了同步数据集中的新数据外，您还可以在标注作业中，直接添加图片，用于数据标注。添加的数据将先导入至标注任务关联的数据集中，然后标注任务会自动同步数据集中最新的数据。

1. 在标注作业详情页面，单击“全部”、“已标注”或“未标注”页签，然后单击左上角“添加数据”，选择添加数据。

图 5-82 添加数据



2. 在弹出的导入对话框中，选择数据来源和导入方式，选择导入的数据路径和数据标注状态。

图 5-83 添加图片

3. 在导入对话框中，单击“确定”，完成添加数据的操作。

   您添加的图片将自动呈现在“全部”的图片列表中，也可单击“添加数据>查看历史记录”，进入“任务历史”界面，可查看相应的导入历史。

图 5-84 查看历史数据

删除图片

通过数据删除操作，可将需要丢弃的图片数据快速删除。

在“全部”、“未标注”或“已标注”页面中，依次选中需要删除的图片，或者选择“选择当前页”选中该页面所有图片，然后单击左上角“删除图片”。在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成图片删除操作。

其中，被选中的图片，其左上角将显示为勾选状态。如果当前页面无选中图片时，“删除图片”按钮为灰色，无法执行删除操作。

说明

如果勾选了“同时删除OBS源文件”，删除图片操作将删除对应OBS目录下存储的图片，此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

5.2.2.3 文本标注

5.2.2.3.1 文本分类

由于模型训练过程需要大量有标签的数据，因此在模型训练之前需对没有标签的文本添加标签。您也可以对已标注文本进行修改、删除和重新标注。

针对文本分类场景，是对文本的内容按照标签进行分类处理，开始标注前，您需要了解：

- 文本标注支持多标签，即一个标注对象可添加多个标签。
- 标签名是由中文、大小写字母、数字、中划线、下划线或特殊符号组成，且不超过1024位的字符串。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

图 5-85 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“未标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

标注文本

标注作业详情页中，展示了此标注作业中“未标注”和“已标注”的文本，默认显示“未标注”的文本列表。

1. 在“未标注”页签文本列表中，页面左侧罗列“标注对象列表”。在列表中单击需标注的文本对象，选择右侧“标签集”中的标签进行标注。一个标注对象可添加多个标签。

   以此类推，不断选中标注对象，并为其添加标签。

图 5-86 文本分类标注

2. 当所有的标注对象都已完成标注，单击页面下方“保存当前页”，完成“未标注”列表的文本标注。

添加标签

- 在“未标注”页签添加：单击页面中标签集右侧的加号，然后在弹出的“新增标签”页中，添加标签名称，选择标签颜色，单击“确定”完成标签的新增。

图 5-87 添加标签（1）

- 在“已标注”页签添加：单击页面中标签集右侧的加号，然后在弹出的“新增标签”页中，添加标签名称，选择标签颜色，单击“确定”完成标签的新增。

图 5-88 添加标签（2）

查看已标注文本

在标注作业详情页，单击“已标注”页签，您可以查看已完成标注的文本列表。您也可以在右侧的“全部标签”中了解当前标注作业支持的所有标签信息。

修改标注

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

- **基于文本修改**

在标注作业详情页，单击“已标注”页签，然后在文本列表中选中待修改的文本。

在文本列表中，单击文本，当文本背景变为蓝色时，表示已选择。当文本有多个标签时，可以单击文本标签上方的×删除单个标签。

- **基于标签修改**

  在标注作业详情页，单击“已标注”页签，在图片列表右侧，显示全部标签的信息。

  - 批量修改：在“全部标签”区域中，单击操作列的编辑图标，然后在文本框中修改标签名称，选择标签颜色，单击“确定”完成修改。
  - 批量删除：在“全部标签”区域中，单击操作列的删除图标，在弹出对话框中，可选择“仅删除标签”或“删除标签及仅包含此标签的标注对象”，然后单击“确定”。

添加文件

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

   导入数据的详细操作介绍请参见导入操作简介。

图 5-89 导入数据

删除文件

通过数据删除操作，可将需要丢弃的文件数据快速删除。

- 在“未标注”页面中，单击选中需要删除的文本对象，然后单击左上角“删除”，即可完成文本的删除操作。

- 在“已标注”页面中，选中待删除的文本对象，然后单击“删除”，删除单个文本。或者选择“选择当前页”选中该页面所有文本，然后单击左上角“删除”，即可完成当前页所有文本的删除操作。

其中，被选中的文本，其背景将显示为蓝色。

标注人员管理

如果您创建的标注作业，开启了团队标注，“标注人员管理”页面中可查看团队标注作业的标注详情。添加、修改或删除标注成员。

1. 登录“数据管理>数据标注”，在“我创建的”页签下可查看所有的标注作业列表。
2. 在作业列表的“名称”列，根据标注作业名称找到对应的团队标注作业。（团队标注作业的名称后带有标识。）
3. 单击作业操作列的“更多>标注人员管理”。或单击作业名称进入作业详情，继续单击右上角“团队标注>标注人员管理”，进入成员管理页面。

图 5-90 进入标注人员管理页（1）

图 5-91 进入标注人员管理页（2）

- 添加成员：

  单击页面“添加成员”，选择成员名称，单击确定。

  在操作列，选择“发送邮件”，可将该标注任务以邮件的方式发送至该标注成员。
- 修改成员信息：

  单击操作列的“修改”，可修改该成员的角色。
- 删除标注成员：

  单击操作列的“删除”可删除该标注成员的所有信息。

5.2.2.3.2 命名实体

命名实体场景，是针对文本中的实体片段进行标注，如“时间”、“地点”等。开始标注前，您需要了解：

实体命名标签名是由中文、大小写字母、数字、中划线、下划线或特殊符号组成，且不超过1024位的字符串。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

   图 5-92 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“未标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

标注文本

标注作业详情页中，展示了此标注作业中“未标注”和“已标注”的文本，默认显示“未标注”的文本列表。

1. 在“未标注”页签文本列表中，页面左侧罗列“标注对象列表”。在列表中单击需标注的文本对象，在右侧标签集下显示的文本内容中选中需要标注的部分，然后选择右侧“标签集”中的标签进行标注。

   以此类推，不断选中标注对象，并为其添加标签。
2. 单击页面下方“保存当前页”完成文本标注。

添加标签

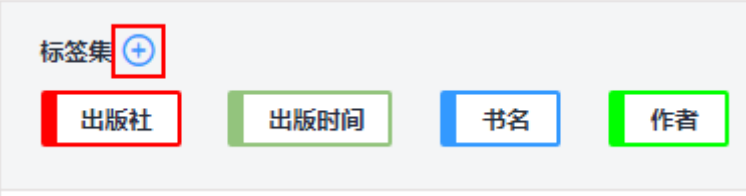
- 在“未标注”页签添加：单击页面中标签集右侧的加号，然后在弹出的“新增标签”页中，添加标签名称，选择标签颜色，单击“确定”完成标签的新增。

  图 5-93 添加命名实体标签（1）

- 在“已标注”页签添加：单击页面中标签集右侧的加号，然后在弹出的“新增标签”页中，添加标签名称，选择标签颜色，单击“确定”完成标签的新增。

  图 5-94 添加命名实体标签（2）

查看已标注文本

在数据集详情页，单击“已标注”页签，您可以查看已完成标注的文本列表。您也可以在右侧的“全部标签”中了解当前数据集支持的所有标签信息。

修改标注

当数据完成标注后，您还可以进入“已标注”页签，对已标注的数据进行修改。

在数据集详情页，单击“已标注”页签，在右侧标签信息区域中对文本信息进行修改。

- **基于文本修改**

  在数据集详情页，单击“已标注”页签，然后在文本列表中选中待修改的文本。

  手工点选删除：在文本列表中，单击文本，当文本背景变为蓝色时，表示已选择。在页面右侧，单击文本标签上方的×删除单个标签。

- **基于标签修改**

  在数据集详情页，单击“已标注”页签，在图片列表右侧，显示全部标签的信息。

  - 批量修改：在“全部标签”区域中，单击操作列的编辑按钮，然后在文本框中添加标签名称，选择标签颜色，单击“确定”完成修改。
  - 批量删除：在“全部标签”区域中，单击操作列的删除按钮，在弹出对话框中，可选择“仅删除标签”或“删除标签及仅包含此标签的标注对象”，然后单击“确定”。

添加文件

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

   导入数据的详细操作介绍请参见导入操作简介。

图 5-95 导入数据

删除文件

通过数据删除操作，可将需要丢弃的文件数据快速删除。

- 在“未标注”页面中，单击选中需要删除的文本对象，然后单击左上角“删除”，即可完成文本的删除操作。
- 在“已标注”页面中，选中待删除的文本对象，然后单击“删除”，删除单个文本。或者选择“选择当前页”选中该页面所有文本，然后单击左上角“删除”，即可完成当前页所有文本的删除操作。

其中，被选中的文本，其背景将显示为蓝色。

标注人员管理

如果您创建的标注作业，开启了团队标注，“标注人员管理”页面中可查看团队标注作业的标注详情。添加、修改或删除标注成员。

1. 登录“数据管理>数据标注”，在“我创建的”页签下可查看所有的标注作业列表。
2. 在作业列表的“名称”列，根据标注作业名称找到对应的团队标注作业。（团队标注作业的名称后带有标识。）
3. 单击作业操作列的“更多>标注人员管理”。或单击作业名称进入作业详情，继续单击右上角“团队标注>标注人员管理”，进入成员管理页面。

图 5-96 进入标注人员管理页（1）

图 5-97 进入标注人员管理页（2）

- 添加成员：

  单击页面“添加成员”，选择成员名称，单击确定。

  在操作列，选择“发送邮件”，可将该标注任务以邮件的方式发送至该标注成员。
- 修改成员信息：

  单击操作列的“修改”，可修改该成员的角色。
- 删除标注成员：

  单击操作列的“删除”可删除该标注成员的所有信息。

5.2.2.3.3 文本三元组

三元组标注适用于标注出语句当中形如（主语/Subject，谓词/Predicate，宾语/Object）结构化知识的场景，标注时不但可以标注出语句当中的实体，还可以标注出实体之间的关系，其在依存句法分析、信息抽取等自然语言处理任务中经常用到。

文本三元组类型的数据标注，需要关注两种标签，“实体标签”和“关系标签”。“关系标签”需设置对应的“起始实体”和“终止实体”。

- 支持设置多个“实体标签”和“关系标签”。一个文本数据中，也可以标注多个“实体标签”和“关系标签”

- 创建数据集时定义的“实体标签”，不支持删除。

注意事项

在开始标注之前，需确保标注作业对应的“实体标签”和“关系标签”已定义好。“关系标签”需设置对应的“起始实体”和“终止实体”。“关系标签”只能添加至其设置好的“起始实体”和“终止实体”之间。

例如，当两个文本都被标注为“地点”，那么针对这两个实体，无法添加本示例中的任意一个关系标签。当无法添加某个关系标签时，界面将显示一个红色的叉号。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

   图 5-98 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“未标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

标注文本

标注作业详情页中，展示了此标注作业中“未标注”和“已标注”的文本，默认显示“未标注”的文本列表。

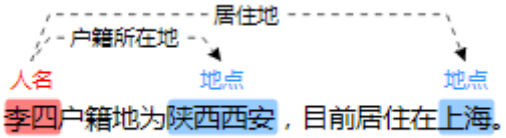
1. 在“未标注”页签文本列表中，页面左侧罗列“标注对象列表”。在列表中单击需标注的文本对象，选中相应文本内容，在页面呈现的实体类型列表中选择实体名称，完成实体标注。

   图 5-99 实体标注

2. 在完成多个实体标注后，鼠标左键依次单击起始实体和终止实体，在呈现的关系类型列表中选择一个对应的关系类型，完成关系标注。

   图 5-100 关系标注

3. 当所有的标注对象都已完成标注，单击页面下方“保存当前页”完成“未标注”列表的文本标注。

说明

“文本三元组”类型的数据集，不支持在标注页面修改标签，需要进入“标签管理”页面，修改“实体标签”和“关系标签”。

修改标注

当数据完成标注后，您还可以进入已标注页签，对已标注的数据进行修改。

在标注作业详情页，单击“已标注”页签，在左侧文本列表中选中一行文本，右侧区域显示具体的标注信息。将鼠标移动至对应的实体标签或关系类型，单击鼠标右键，

可删除此标注。单击鼠标左键，依次单击连接起始实体和终止实体，可增加关系类型，增加关系标注。

您也可以在单击页面下方的“删除当前项标签”按钮，删除选中文本对象中的所有标签。

添加文件

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

   导入数据的详细操作介绍请参见数据接入。

图 5-101 导入数据

删除文件

通过数据删除操作，可将需要丢弃的文件数据快速删除。

- 在“未标注”页面中，单击选中需要删除的文本，然后单击左上角“删除”，即可完成文本的删除操作。
- 在“已标注”页面中，选中待删除的文本，然后单击“删除”，删除单个文本。或者勾选“选择当前页”选中该页面所有文本，然后单击左上角“删除”，即可完成当前页所有文本的删除操作。

其中，被选中的文本，其背景将显示为蓝色。如果当前页面无选中文本时，“删除”按钮为灰色，无法执行删除操作。

标注人员管理

如果您创建的标注作业，开启了团队标注，“标注人员管理”页面中可查看团队标注作业的标注详情。添加、修改或删除标注成员。

1. 登录“数据管理>数据标注”，在“我创建的”页签下可查看所有的标注作业列表。
2. 在作业列表的“名称”列，根据标注作业名称找到对应的团队标注作业。（团队标注作业的名称后带有标识。）
3. 单击作业操作列的“更多>标注人员管理”。或单击作业名称进入作业详情，继续单击右上角“团队标注>标注人员管理”，进入成员管理页面。

图 5-102 进入标注人员管理页（1）

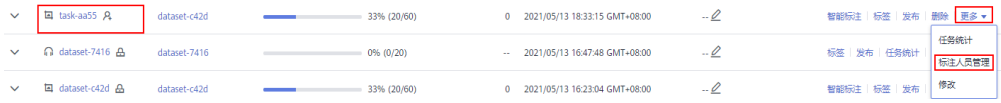


图 5-103 进入标注人员管理页（2）

- 添加成员：

  单击页面“添加成员”，选择成员名称，单击确定。

  在操作列，选择“发送邮件”，可将该标注任务以邮件的方式发送至该标注成员。
- 修改成员信息：

  单击操作列的“修改”，可修改该成员的角色。
- 删除标注成员：

  单击操作列的“删除”可删除该标注成员的所有信息。

5.2.2.4 音频标注

5.2.2.4.1 声音分类

声音分类是对声音进行分类。

由于模型训练过程需要大量有标签的音频数据，因此在模型训练之前需对没有标签的音频添加标签。通过ModelArts您可对音频进行一键式批量添加标签，快速完成对音频的标注操作，也可以对已标注音频修改或删除标签进行重新标注。

音频标注涉及到的标注标签和声音内容只支持中文和英文，不支持小语种。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

   图 5-104 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“未标注”和“已标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

标注音频

标注作业详情页中，展示了此标注作业中“未标注”和“已标注”的音频，默认显示“未标注”的音频列表。单击音频左侧，即可进行音频的试听。

1. 在“未标注”页签，勾选需进行标注的音频。
   - 手工点选：在音频列表中，单击音频，当右上角出现蓝色勾选框时，表示已勾选。可勾选同类别的多个音频，一起添加标签。
   - 批量选中：如果音频列表的当前页，所有音频属于一种类型，可以在列表的右上角单击“选择当前页”，则当前页面所有的音频将选中。
2. 添加标签。
   a. 在右侧的“添加标签”区域中，单击“标签”下侧的文本框设置标签。

      方式一（已存在标签）：单击“标签”下方的文本框，在快捷键下拉列表中选择快捷键，然后在标签文本输入框中选择已有的标签名称，然后单击“确定”。

      方式二（新增标签）：在“标签”下方的文本框中，在快捷键下拉列表中选择快捷键，然后在标签文本输入框中输入新的标签名称，然后单击“确定”。

   b. 选中的音频将被自动移动至“已标注”页签，且在“未标注”页签中，标签的信息也将随着标注步骤进行更新，如增加的标签名称、各标签对应的音频数量。

说明

快捷键的使用说明：为标签指定快捷键后，当您选择一段音频后，在键盘中按一下快捷键，即可为此音频增加为此快捷键对应的标签。例如“aa”标签对应的快捷键是“1”，在数据标注过程中，选中1个或多个文件，按“1”，界面将提示是否需要将此文件标注为“aa”标签，单击确认即可完成标注。

快捷键对应的是标签，1个标签对应1个快捷键。不同的标签，不能指定为同一个快捷键。快捷键的使用，可以大大提升标注效率。

图 5-105 添加音频标签

查看已标注音频

在标注作业详情页，单击“已标注”页签，您可以查看已完成标注的音频列表。单击音频，可在右侧的“选中文件标签”中了解当前音频的标签信息。

修改标注

当数据完成标注后，您还可以进入“已标注”页签，对已标注的数据进行修改。

- **基于音频修改**

  在标注作业详情页面，单击“已标注”页签，然后在音频列表中选中待修改的音频（选择一个或多个）。在右侧标签信息区域中对标签进行修改。

  - 修改标签：在“选中文件标签”区域中，单击操作列的编辑图标，然后在文本框中输入正确的标签名，然后单击确定图标完成修改。
  - 删除标签：在“选中文件标签”区域中，单击操作列的删除图标删除该标签。

- **基于标签修改**

  在标注作业详情页面，单击“已标注”页签，在音频列表右侧，显示全部标签的信息。

  图 5-106 全部标签信息

  - 修改标签：单击操作列的编辑图标，然后在弹出的对话框中输入修改后的标签名，然后单击“确定”完成修改。修改后，之前添加了此标签的音频，都将被标注为新的标签名称。
  - 删除标签：单击操作列的删除图标，在弹出的对话框中，根据提示框选择需要删除的对象，然后单击“确定”完成删除。

添加音频

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”或“已标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

   导入数据的详细操作介绍请参见导入操作简介。

图 5-107 导入数据

删除音频

通过数据删除操作，可将需要丢弃的音频数据快速删除。

在“未标注”或“已标注”页面中，选中需要删除的音频，或者选择“选择当前页”选中该页面所有音频，然后单击左上角“删除音频”，在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成音频删除操作。

其中，被选中的音频，其右上角将显示为勾选状态。如果当前页面无选中音频时，“删除音频”按钮为灰色，无法执行删除操作。

说明

如果勾选了“同时删除OBS源文件”，删除音频操作是将删除对应OBS目录下存储的音频。此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

5.2.2.4.2 语音内容

语音内容是对语音内容进行标注。

由于模型训练过程需要大量有标签的音频数据，因此在模型训练之前需对没有标签的音频添加标签。通过ModelArts您可对音频进行一键式批量添加标签，快速完成对音频的标注操作，也可以对已标注音频修改或删除标签进行重新标注。

音频标注涉及到的标注标签和声音内容只支持中文和英文，不支持小语种。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

说明

数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

图 5-108 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步新数据

ModelArts会自动将数据集中新增的数据同步至标注作业，包含数据及当前标注作业支持的标注信息。

为了快速获取数据集中最新数据，可在标注作业详情页的“未标注”页签中，单击“同步新数据”，快速将数据集中的数据添加到标注作业中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

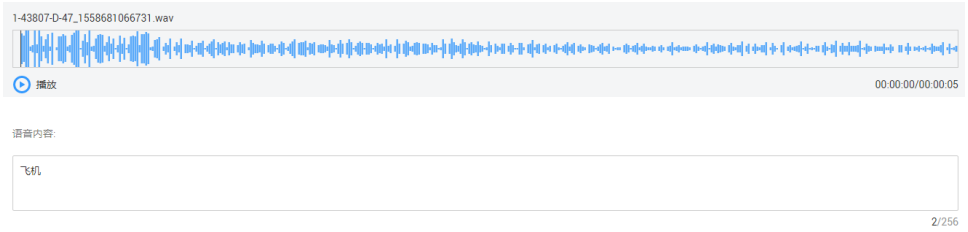
标注音频

标注作业详情页中，展示了此数据集中“未标注”和“已标注”的音频，默认显示“未标注”的音频列表。

1. 在“未标注”页签左侧音频列表中，单击目标音频文件，在右侧的区域中出现音频，单击音频下方，即可进行音频播放。
2. 根据播放内容，在下方“语音内容”文本框中填写音频内容。

3. 输入内容后单击下方的“确认标注”按钮完成标注。音频将被自动移动至“已标注”页签。

图 5-109 语音内容音频标注



查看已标注音频

在标注作业详情页，单击“已标注”页签，您可以查看已完成标注的音频列表。单击音频，可在右侧的“语音内容”文本框中了解当前音频的内容信息。

修改标注

当数据完成标注后，您还可以进入“已标注”页签，对已标注的数据进行修改。

在标注作业详情页，单击“已标注”页签，然后在音频列表中选中待修改的音频。在右侧标签信息区域中修改“语音内容”文本框中的内容，单击下方的“确认标注”按钮完成修改。

添加音频

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

   导入数据的详细操作介绍请参见接入简介。

图 5-110 导入数据

删除音频

通过数据删除操作，可将需要丢弃的音频数据快速删除。

在“未标注”或“已标注”页面中，选中需要删除的音频，然后单击左上角“删除音频”，在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成音频删除操作。

说明

如果勾选了“同时删除源OBS文件”，删除音频操作是将删除对应OBS目录下存储的音频。此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

5.2.2.4.3 语音分割

语音分割是对语音进行分段标注。

由于模型训练过程需要大量有标签的音频数据，因此在模型训练之前需对没有标签的音频添加标签。通过ModelArts您可对音频添加标签，快速完成对音频的标注操作，也可以对已标注音频修改或删除标签进行重新标注。

音频标注涉及到的标注标签和声音内容只支持中文和英文，不支持小语种。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

说明

数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

图 5-111 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步数据源

ModelArts会自动从数据集输入位置同步数据至数据集详情页，包含数据及标注信息。

为了快速获取OBS桶中最新数据，可在数据集详情页的“未标注”页签中，单击“同步数据源”，快速将通过OBS上传的数据添加到数据集中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

标注音频

标注作业详情页中，展示了此标注作业中“未标注”和“已标注”的音频，默认显示“未标注”的音频列表。

1. 在“未标注”页签左侧音频列表中，单击目标音频文件，在右侧的区域中出现音频，单击音频下方，即可进行音频播放。
2. 根据播放内容，选取合适的音频段，在下方“语音内容”文本框中填写音频标签和内容。

图 5-112 语音标签音频标注

3. 输入内容后单击下方的“确认标注”按钮完成标注。音频将被自动移动至“已标注”页签。

查看已标注音频

在标注作业详情页，单击“已标注”页签，您可以查看已完成标注的音频列表。单击音频，可在右侧的对应的文本框中了解当前音频的内容信息。

修改标注

当数据完成标注后，您还可以进入“已标注”页签，对已标注的数据进行修改。

- 修改标签：在数据标注详情页，单击“已标注”页签，然后在音频列表中选中待修改的音频。在右侧下方的标签信息区域中修改语音内容中的“标签”和，单击下方的“确认标注”按钮完成修改。
- 删除标签：单击目标编号操作列的“删除”，删除该段音频的标注。您也可以单击标注音频文件上方的删除标注，然后单击“确认标注”。

添加音频

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

图 5-113 导入数据

删除音频

通过数据删除操作，可将需要丢弃的音频数据快速删除。

在“未标注”或“已标注”页面中，选中需要删除的音频，然后单击左上角“删除音频”，在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成音频删除操作。

说明

如果勾选了“同时删除OBS源文件”，删除音频操作是将删除对应OBS目录下存储的音频。此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

标注人员管理

如果您创建的标注作业，开启了团队标注，“标注人员管理”页面中可查看团队标注作业的标注详情。添加、修改或删除标注成员。

1. 登录“数据管理>数据标注”，在“我创建的”页签下可查看所有的标注作业列表。
2. 在作业列表的“名称”列，根据标注作业名称找到对应的团队标注作业。（团队标注作业的名称后带有标识。）
3. 单击作业操作列的“更多>标注人员管理”。或单击作业名称进入作业详情，继续单击右上角“团队标注>标注人员管理”，进入成员管理页面。

图 5-114 进入标注人员管理页（1）

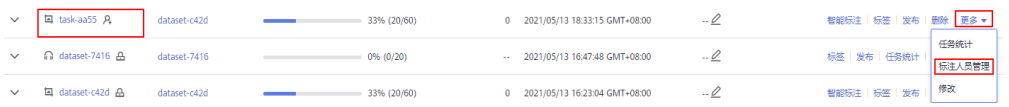


图 5-115 进入标注人员管理页（2）

- 添加成员：
  单击页面“添加成员”，选择成员名称，单击确定。
  在操作列，选择“发送邮件”，可将该标注任务以邮件的方式发送至该标注成员。
- 修改成员信息：
  单击操作列的“修改”，可修改该成员的角色。
- 删除标注成员：
  单击操作列的“删除”可删除该标注成员的所有信息。

5.2.2.5 视频标注

由于模型训练过程需要大量有标签的视频数据，因此在模型训练之前需对没有标签的视频添加标签。通过ModelArts您可对视频添加标签，快速完成对视频的标注操作，也可以对已标注视频修改或删除标签进行重新标注。

说明

视频标注仅针对视频帧进行标注。

开始标注

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理> 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表右侧“所有类型”页签下拉选择标注类型，基于“标注类型”选择需要进行标注的标注作业，单击标注作业名称进入标注作业标注详情页。

图 5-116 下拉选择标注类型

3. 在标注作业标注详情中，展示此标注作业下全部数据。

同步数据源

ModelArts会自动从数据集输入位置同步数据至数据集详情页，包含数据及标注信息。

为了快速获取OBS桶中最新数据，可在数据集详情页的“已标注”或“未标注”页签中，单击“同步数据源”，快速将通过OBS上传的数据添加到数据集中。

说明

问题现象：

将已标注好的数据上传至OBS，同步数据后，显示为未标注。

原因分析：

可能是OBS桶设置了自动加密导致此问题。

解决方法：

需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

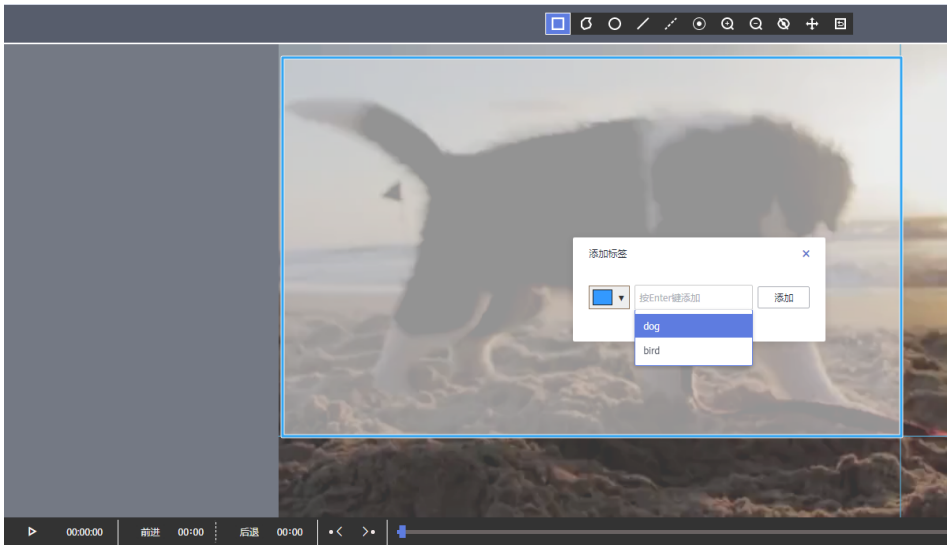
视频标注

标注作业详情页中，展示了此数据集中“未标注”、“**已标注**”和“全部”的视频。

1. 在“未标注”页签左侧视频列表中，单击目标视频文件，打开标注页面。
2. 在标注页面中，播放视频，当视频播放至待标注时间时，单击进度条左侧的暂停按钮，将视频暂停至某一帧对应的画面。
3. 在上方区域选择标注框，默认为矩形框。使用鼠标在视频画面中框出目标，然后在弹出的添加标签文本框中，直接输入新的标签名，在文本框前面选中标签颜色，单击“添加”完成1个物体的标注。如果已存在标签，从下拉列表中选择已有的标签，然后单击“添加”完成标注。逐步此画面中所有物体所在位置，一帧对应的画面可添加多个标签。

   支持的标注框与“物体检测”类型一致，详细描述请参见物体检测章节的**表2 标注界面的常用按钮**。

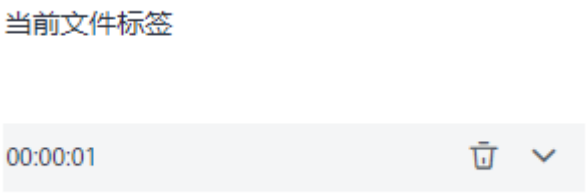
图 5-117 视频标注



4. 上一帧对应的画面标注完成后，在进度条处单击播放按钮继续播放，在需要标注处暂停，然后重复执行步骤3完成整个视频的标注。

   单击界面右上角的“标注列表”，在“当前文件标签”的详情页将呈现当前视频带标注的时间点。

图 5-118 当前文件标签信息



5. 单击页面左上角“返回数据标注预览”，页面将自动返回标注作业详情页面，同时，标注好的视频将呈现在“已标注”页签下。

常见问题

Q：视频数据集无法显示或者无法播放视频?

A：如果无法显示和播放视频，请检查视频格式类型，目前只支持MP4格式。

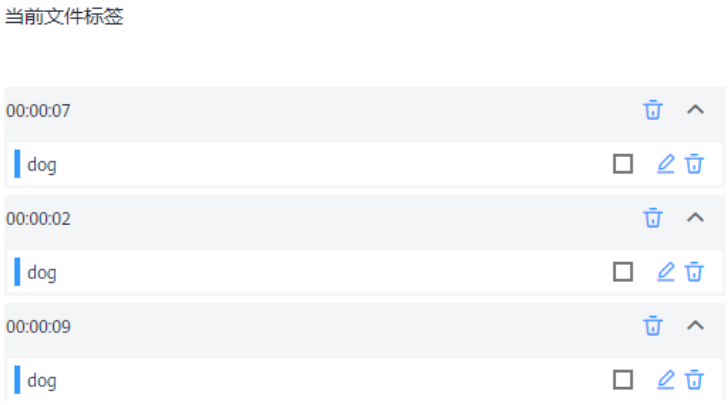
修改标注

当数据完成标注后，您还可以进入“已标注”页签，修改标注数据。

- 在“已标注”页签下，单击目标视频文件，在标注页面的右上方选择“标注列表”进入“当前文件标签”详情页面，可单击时间点右侧小三角展开详情，您可以修改或删除标签。
- 修改标签：在“当前文件标签”详情页，单击标签右侧的编辑按钮，标签名称可进行修改。

- 删除标签：在“当前文件标签”详情页，单击标签右侧的删除按钮，将直接删除此标签。如果单击画面时间右侧的删除按钮，将删除此画面下的所有标签。

图 5-119 修改标注

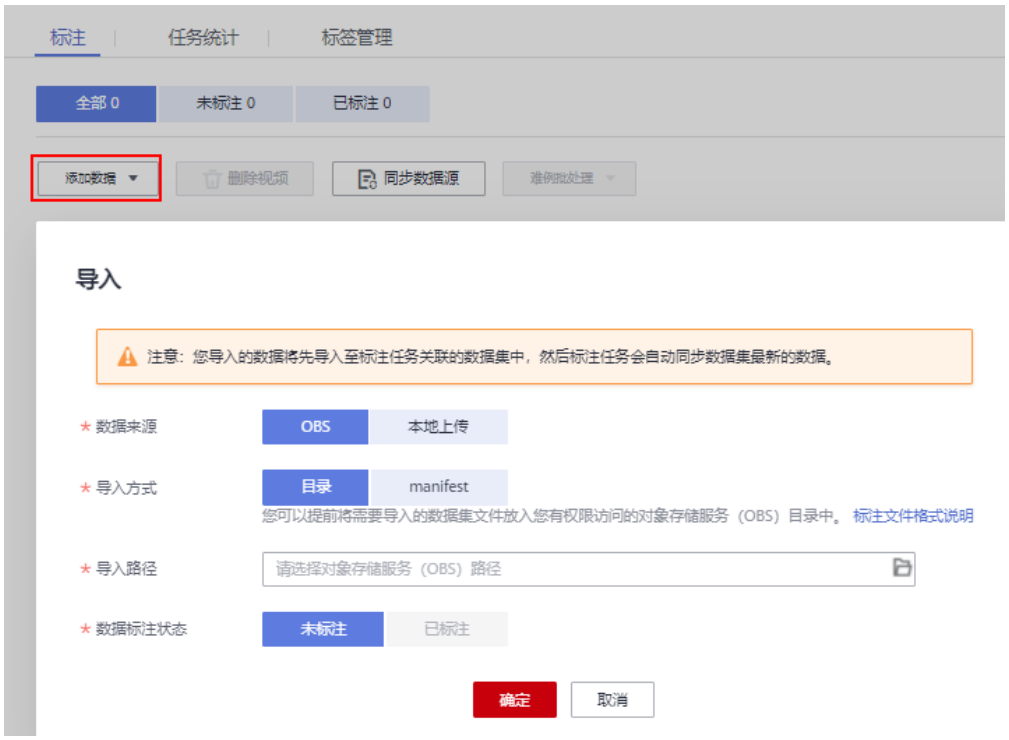


添加视频

除了同步新数据外，您还可以在标注详情页面中，直接添加数据，用于数据标注。

1. 在标注作业详情页面，单击“未标注”或“已标注”页签，然后单击左上角“添加数据”。
2. 在弹出的导入对话框中，选择数据来源、导入方式、导入路径等参数，导入数据。单击确定。

图 5-120 导入数据



删除视频

通过数据删除操作，可将需要丢弃的视频数据快速删除。

在“全部”、“未标注”或“已标注”页面中，依次选中需要删除的视频，或者选择“选择当前页”选中该页面所有视频，然后单击上边的“删除”。在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成视频删除操作。

其中，被选中的视频，其左上角将显示为勾选状态。如果当前页面无选中视频时，“删除视频”按钮为灰色，无法执行删除操作。

说明

如果勾选了“同时删除OBS源文件”，删除视频操作将删除对应OBS目录下存储的视频，此操作可能会影响已使用此源文件的其他数据集或数据集版本，有可能导致展示异常或训练/推理异常。删除后，数据将无法恢复，请谨慎操作。

5.2.2.6 查看标注作业

5.2.2.6.1 查看创建的作业

在ModelArts数据标注页面可查看用户自己创建的标注作业。

操作步骤

1. 登录ModelArts管理控制台，在左侧菜单栏选择“数据管理>数据标注”，进入数据标注页面。
2. 在数据标注详情页选择“我创建的”，即可展示所有自己创建的标注作业。用户可查看自己创建的标注作业的相关信息，

图 5-121 我创建的

标注作业复制

1. 登录ModelArts管理控制台，在左侧菜单栏选择“数据管理>数据标注”，进入数据标注页面。
2. 在数据标注列表页，“我创建的”页签下，选择需要复制的标注任务。
3. 单击作业操作列的“更多>复制”。
4. 在标注任务复制的弹窗中，填写作业描述，作业名称task-xxxx-copy-***xxxx***，其中***xxxx***为系统生成的随机码，用来区分新作业与被复制作业。也可以修改新生成的作业名称。单击“确定”。

5. 复制完成后，在标注作业列表页即可查询新的标注任务，拷贝标注作业信息包含，拷贝标注任务的样本、标签、团队标注信息。

5.2.2.6.2 查看参与标注的作业

在ModelArts数据标注页面可查看用户参与的标注作业。

前提条件

创建标注作业时，启用了团队标注。

操作步骤

1. 登录ModelArts管理控制台。在左侧菜单栏选择“数据管理>数据标注”，进入数据标注详情页面。
2. 在数据标注详情页选择“我参与的”，即可展示所有参与过标注的标注作业，用户可查看自己参与过的标注作业的详情。包括标注团队的成员、标注进展等情况。

5.2.3 智能标注

5.2.3.1 创建智能标注作业

除了人工标注外，ModelArts还提供了智能标注功能，快速完成数据标注，为您节省70%以上的标注时间。智能标注是指基于当前标注阶段的标签及图片学习训练，选中系统中已有的模型进行智能标注，快速完成剩余图片的标注操作。

背景信息

- 目前只有“图像分类”和“物体检测”类型的标注作业支持智能标注功能。
- 启动智能标注时，需标注作业存在至少2种标签，且每种标签已标注的图片不少于5张。
- 启动智能标注时，必须存在未标注图片。
- 启动智能标注前，保证当前系统中不存在正在进行中的智能标注任务。
- 检查用于标注的图片数据，确保您的图片数据中，不存在RGBA四通道图片。如果存在四通道图片，智能标注任务将运行失败，因此，请从数据集中删除四通道图片后，再启动智能标注。

启动智能标注作业

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理 > 数据标注”，进入“数据标注”管理页面。

   说明

   数据管理模块在重构升级中，对未使用过数据管理的用户不可见。如果要使用数据管理相关功能，建议提交工单开通权限。

2. 在标注作业列表中，选择“物体检测”或“图像分类”类型的标注作业，单击操作列的“智能标注”启动智能标注作业。
3. 在弹出的“启动智能标注”对话框中，选择智能标注类型，可选“主动学习”或者“预标注”，详见表5-33和表5-34。

表 5-33 主动学习

参数	说明
智能标注类型	“主动学习”。“主动学习”表示系统将自动使用半监督学习、难例筛选等多种手段进行智能标注，降低人工标注量，帮助用户找到难例。
算法类型	针对“图像分类”类型的数据集，您需要选择以下参数。 “快速型”：仅使用已标注的样本进行训练。
计算节点规格	即智能标注任务使用的资源规格。
计算节点个数	默认为1，表示单机模式。目前仅支持此参数值。

表 5-34 预标注

参数	说明
智能标注类型	“预标注”。“预标注”表示选择用户模型管理里面的模型，选择模型时需要注意模型类型和数据集的标注类型相匹配。预标注结束后，如果标注结果符合平台定义的标准标注格式，系统将进行难例筛选，该步骤不影响预标注结果。

参数	说明
选择模型及版本	• “我的AI应用”。您可以根据实际需求选择您的模型。您需要在目标AI应用的左侧单击下拉三角标，选择合适的版本。您的模型导入参见**创建AI应用**。

说明

针对“物体检测”类型的标注作业，选择“主动学习”时，只支持识别和标注矩形框。

图 5-122 启动智能标注（预标注）

4. 完成参数设置后，单击“提交”，即可启动智能标注。
5. 在标注作业列表中，单击标注作业名称进入“标注作业详情”页。
6. 在“数据集概览页标注作业详情页”，选择“标注”页签，单击“待确认”页签，即可查看智能标注进度。

   您也可以在该页签，“启动智能标注”或者查看“智能标注历史”

图 5-123 标注进度

说明

当系统中智能标注任务过多时，因免费资源有限，可能会出现排队的情况，导致作业一直处于“标注中”的状态。请您耐心等待，为确保您的标注作业能顺利进行，建议您避开高峰期使用。

7. 智能标注完成后，“待确认”页面将呈现所有标注后的图片列表。
   - 图像分类标注作业

     在“待确认”页面查看标签是否准确，勾选标注准确的图片，然后单击“确认”完成智能标注结果的确认。确认完成后的图片将被归类至“已标注”页面下。
   - 物体检测标注作业

     在“待确认”页面，单击图片查看标注详情，查看标签及目标框是否准确，针对标注准确的图片单击“确认标注”完成智能标注结果的确认。确认完成后的图片将被归类至“已标注”页面下。

相关问题

- **智能标注失败，如何处理?**

  当前智能标注为免费使用阶段，当系统的标注任务过多时，因免费资源有限，导致任务失败，请您重新创建智能标注任务或建议您避开高峰期使用。
- **智能标注时间过长，如何处理?**

  当前智能标注为免费使用阶段，当系统的标注任务过多时，因免费资源有限，需要排队，您的标注任务会长时间处于“标注中”状态，请您耐心等待。或建议您避开高峰期使用。

5.2.4 团队标注

5.2.4.1 团队标注简介

数据标注任务中，一般由一个人完成，但是针对数据集较大时，需要多人协助完成。ModelArts提供了团队标注功能，可以由多人组成一个标注团队，针对同一个数据集进行标注管理。

说明

团队标注功能当前仅支持“图像分类”、“物体检测”、“文本分类”、“命名实体”、“文本三元组”、“语音分割”类型的数据集。

针对启用团队标注功能的数据标注任务，支持创建团队标注任务，将标注任务指派给不同的团队，由多人完成标注任务。同时，在成员进行数据标注过程中，支持发起验收、继续验收以及查看验收报告等功能。

团队标注功能是以团队为单位进行管理，数据集启用团队标注功能时，必须指定一个团队。一个团队可以添加多个成员。

- 一个账号最多可添加10个团队。
- 如果数据集需要启用团队标注功能，当前账号至少拥有一个团队。如果没有，请执行**添加团队**操作添加。

5.2.4.2 创建和管理团队

5.2.4.2.1 管理团队

团队标注功能是以团队为单位进行管理，数据集启用团队标注功能时，必须指定一个团队。一个团队可以添加多个成员。

背景说明

- 一个账号最多可添加10个团队。
- 如果数据集需要启用团队标注功能，当前账号至少拥有一个团队。如果没有，请执行**添加团队**操作添加。

添加团队

1. 在ModelArts管理控制台左侧导航栏中，选择“数据管理>标注团队”，进入“标注团队”管理页面。
2. 在“标注团队”管理页面，单击“添加团队”。
3. 在弹出的“添加团队”对话框中，填写团队“名称”和“描述”，然后单击“确定”。完成标注团队的添加。

   团队添加完成后，“标注团队”管理页面呈现新添加的团队，在页面右侧区域，可以查看团队详情。新添加的团队，其成员列表为空，请参考**添加成员**操作，为您的团队添加成员。

删除团队

当已有的团队不再使用，您可以执行删除操作。

在“标注团队”管理页面中，选中需删除的团队，然后单击“删除”。在弹出的对话框中，确认信息无误后，单击“确定”完成团队删除。

5.2.4.2.2 管理成员

新添加的团队，其成员列表为空。您需要根据实际情况添加即将参与标注任务的成员信息。

一个团队最多支持添加100个成员，当超过100时，建议分为多个团队进行管理。

添加成员

1. 在ModelArts管理控制台左侧导航栏中，选择“数据管理>标注团队”，进入“标注团队”管理页面。
2. 在“标注团队”管理页面，从左侧团队列表中选择一个团队，单击团队，其右侧区域将呈现“团队详情”。
3. 在“团队详情”区域，单击“添加成员”。
4. 邮箱作为团队管理中的唯一标识，不同成员不能使用同一个邮箱。您填写的邮箱地址将被记录并保存在ModelArts中，仅用于ModelArts团队标注功能，当成员删除后，其填写的邮箱信息也将被一并删除。

   其中，“角色”支持“Labeler”、“Reviewer”和“Team Manager”，“Team Manager”只能设置为一个人。

   需要注意的是：目前不支持从标注任务中删除labeler。labeler的标注必须通过审核后，才能同步到最终结果，不支持单独分离操作。

   成员添加完成后，团队详情区域中将呈现此成员的信息。

修改成员信息

团队中的成员，当其信息发生变化时，可以编辑其基本情况。

1. 在“团队详情”区域，选择需修改的成员。
2. 在成员所在行的“操作”列，单击“修改”。在弹出的对话框中，修改其“描述”或“角色”。

   成员的“邮箱”无法修改，如果需要修改邮箱地址，建议先删除此成员，然后再基于新的邮箱地址添加新成员。

   “角色”支持“Labeler”、“Reviewer”和“Team Manager”，“Team Manager”只能设置为一个人。

删除成员

- **删除单个成员**

  在“团队详情”区域，选择需要删除的成员， 单击“操作”列的“删除”。在弹出的对话框中，确认信息无误后，单击“确定”完成删除操作。

- **批量删除**

  在“团队详情”区域，勾选需删除的成员，然后单击“删除”。在弹出的对话框中，确认信息无误后，单击“确定”完成多个成员的删除操作。

5.2.4.3 创建团队标注任务

如果您在创建标注作业时，即启用团队标注，且指派了某一团队负责标注，系统将默认基于此团队创建一个标注任务。您可以在创建数据标注任务后，在“我创建的”页面查看此任务。

您还可以重新创建一个团队标注任务，指派给同一团队的不同成员，或者指派给其他标注团队。

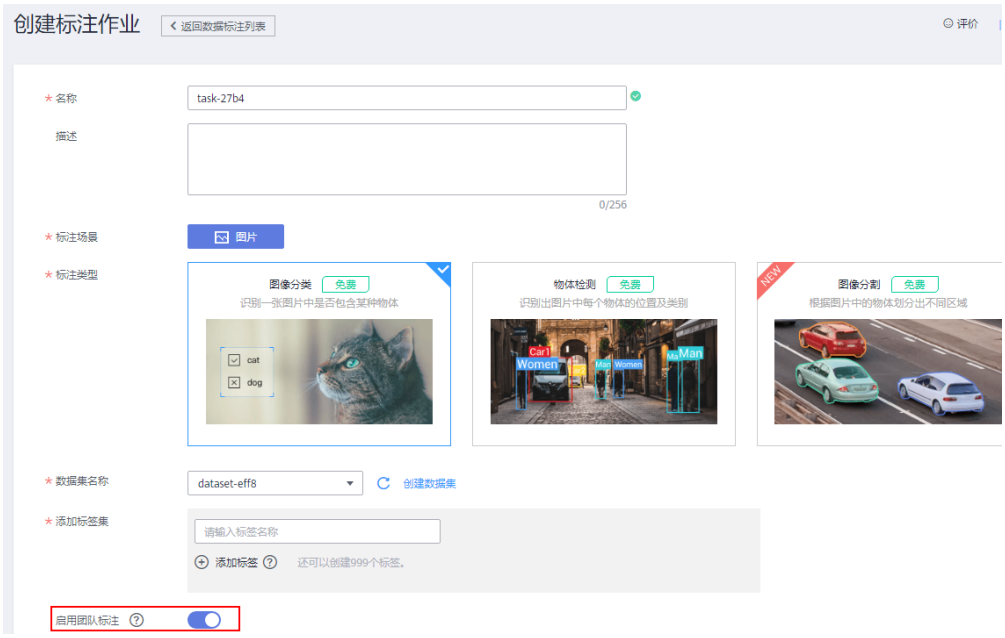
团队标注作业的创建方式

- 从控制台的“数据管理 >数据标注”页面进入，创建标注作业时，打开“启用团队标注”开关，同时指定一个标注团队，或者指定标注管理员。

图 5-124 创建团队标注作业

- 从控制台的“数据管理 >数据集”进入数据集页面，在需要进行团队标注的数据集的操作列，单击“标注”，进入创建标注作业页面，打开“启用团队标注”开关。对于同一个数据集，可以创建多个团队标注任务。

图 5-125 打开启用团队标注



说明

- 只有当创建团队标注任务时，标注人员才会收到邮件。创建标注团队及添加标注团队的成员并不会发送邮件。此外，当所有样本都是已标注状态时，创建团队标注任务也不会收到邮件。
- 标注任务创建完成后，会将所有未标注状态的样本分配给标注人员。分配采用随机均分的策略，不支持重复分配。

创建团队标注任务

同一个数据集，支持创建多个团队标注作业，指派给同一团队的不同成员，或者指派给其他标注团队。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“数据管理 >数据集”，打开数据集列表。
2. 在数据集列表中，选择支持团队标注的数据集，单击数据集名称进入数据集概览页。
3. 在数据集概览页页面，右侧的“标注任务”区域，可查看此数据集已有的标注任务。单击“新建标注任务”开始创建新任务。

   或者也可以从“数据管理 >数据标注”页面进入，单击“创建标注作业”进入创建标注作业页面。
4. 在弹出的“创建标注作业”页面中，填写相关参数，然后单击“确定”，完成任务创建。
   - “名称”：设置此任务的名称。
   - **“标注场景”**：选择标注作业的任务类型。
   - “标签集”：展示当前数据集已有的标签及标签属性。
   - **“启用团队标注”**：选择打开，并配置如下团队标注相关参数。
     - “类型”：设置任务类型，支持“指定标注团队”或“指定标注管理员”。
     - “选择标注团队”：任务类型设置为“指定标注团队”，需在此参数中指定一个团队，同时勾选此团队中某几个成员负责标注。下拉框中将罗列当前账号下创建的标注团队及其成员。
     - “选择标注接口人”：任务类型设置为“指定标注管理员”，需在所有团队的“Team Manager”中选择一人作为管理员。
     - “自动将新增图片同步给标注团队”：根据需要选择是否将任务中新增的数据自动同步给标注人员。
     - “团队标注的图片自动加载智能标注结果”：根据需要选择是否将任务中智能标注待确认的结果自动同步给标注人员。

       说明

       团队标注加载智能标注结果的处理步骤：
       - 如果类型选择"指定标注团队"，需要先创建团队标注任务，然后执行智能标注任务。
       - 如果类型选择"指定标注管理员"，在“我参与的”页签下选择团队标注任务，单击"分配任务"。

任务创建完成后，您可以在“我创建的”页签下看到新建的任务。

5.2.4.4 登录 ModelArts-Console

在ModelArts中，一般用户使用数据标注功能，直接是在“数据管理”模块操作，此模块包含数据管理所有能力，数据集管理、数据标注、数据导入导出、智能标注、团队标注和管理等。团队标注任务创建成功后，团队成员登录ModelArts-Console查看相关任务。

1. 团队标注任务创建成功后，团队成员收到标注任务的邮件。

图 5-126 任务邮件

2. 单击任务邮件中的标注任务地址，跳转至ModelArts数据管理>数据标注页面的“我参与的”页签。如果未登录控制台，请先登录。
3. 在“我参与的”页签下，可查看您的标注任务。

图 5-127 标注任务

如果团队成员绑定了邮箱，可以收到任务通知邮件，成员也可以通过邮件中给出的地址访问ModelArts-Console标注地址。

登录后，仅显示当前用户（此邮箱用户）相关的团队标注任务及其相关数据。

5.2.4.5 启动团队标注任务

登录到console标注页面后在“我参与的”页签下，可查看到分配的标注任务，单击任务名称，可进入标注页面。不同类型的标注作业，标注方式不同，详细请参见：

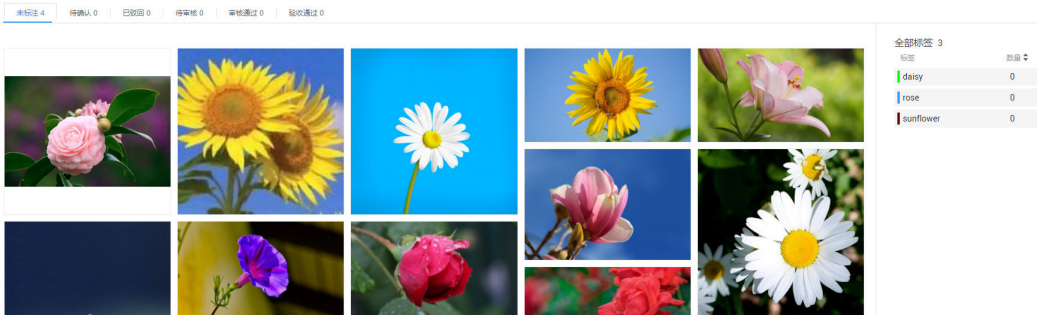
- 图像分类
- 物体检测
- 文本分类
- 命名实体

- 文本三元组
- 语音分割

在标注页面中，每个成员可查看“未标注”、“待确认”、“已驳回”、“待审核”、“审核通过”、“验收通过”的图片信息。请及时关注管理员驳回以及待修正的图片。

当团队标注任务中，分配了Reviewer角色，则需要对标注结果进行审核，审核完成后，再提交给管理员验收。

图 5-128 成员标注平台



5.2.4.6 审核团队标注任务结果

团队标注成员完成后，团队审核者可以对标注结果进行审核。

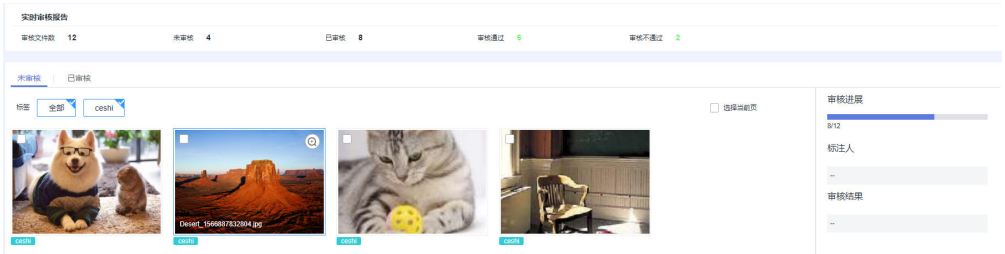
1. 登录ModelArts管理控制台，左侧菜单栏选择“数据管理>数据标注”，在数据标注页面选择“我参与的”，在任务列表“操作”列单击“审核”，发起审核。

   图 5-129 发起审核

2. 在审核页面中，审核人员可以查看“未审核”、“已审核”、“审核通过”、“审核不通过”的样本。

   图 5-130 标注结果审核

3. 审核人员可以在审核页面的右侧选择“审核结果”（“通过”或“不通过”）。

   当选择审核结果为“通过”时，需设置“验收评分”（分“A”、“B”、“C”、“D”四个选项，“A”表示最高分），如图5-131所示。当选择审核结果为“不通过”时，可以在文本框中写明驳回原因，如图5-132所示。

图 5-131 审核通过

图 5-132 审核不通过

5.2.4.7 验收团队标注任务结果

任务验收（管理员）

- **发起验收**

  当团队的成员已完成数据标注，标注作业的创建者可发起验收，对标注结果进行抽验。只有当标注成员存在标注完成的数据时，才可以发起验收，否则发起验收按钮为灰色。

a. 录ModelArts管理控制台，在左侧菜单栏中选择“数据管理 >数据标注”，打开数据标注管理页。

b. 选择“我参与的”，选择团队标注作业，单击作业名称进入“标注作业详情页”，单击右上角“团队标注>验收”，发起验收。

图 5-133 发起验收

c. 在弹出的对话框中，设置“抽样策略”，可设置为“按百分比”，也可以设置为“按数量”。设置好参数值后，单击“确定”启动验收。

“按百分比”：按待验收图片总数的一定比例进行抽样验收。

“按数量”：按一定数量进行抽样验收。

d. 验收启动后，界面将展示实时验收报告，您可以在右侧选择“验收结果”（“通过”或“不通过”）。

当选择验收结果为“通过”时，需设置“验收评分”（分“A”、“B”、“C”、“D”四个选项，“A”表示最高分）。当选择验收结果为“不通过”时，可以在文本框中写明驳回原因。

- **继续验收**

针对未完成验收的任务，可以继续验收。针对未发起过验收流程的任务，不支持“继续验收”，按钮为灰色。

在“任务统计>标注进展”页签中，针对需继续验收的任务，单击“继续验收”。系统直接进入“实时验收报告”页面，您可以继续验收未验收的图片，设置其“验收结果”。

- **完成验收**

继续验收完成后，单击右上角“完成验收”在完成验收窗口，您可以查看本标注作业的验收情况，如抽样文件数等，同时设置如下参数，然后进行验收。只有完成验收，标注信息才会同步到标注作业的已标注页面中。

一旦标注数据完成验收，团队成员无法再修改标注信息，只有数据集创建者可修改。

表 5-35 完成验收的参数设置

参数	说明
对已标注数据修改	• 不覆盖：针对同一个数据，不使用当前团队标注的结果覆盖已有数据。 • 覆盖：针对同一个数据，使用当前团队标注的结果覆盖已有数据。覆盖后无法恢复，请谨慎操作。
验收范围	• 全部通过：被驳回的样本，也会通过。 • 全部驳回：已经通过的样本，需要重新标注，下次验收时重新进行审核。 • 剩余全部通过：已经驳回的会驳回，其余会自动验收通过。 • 剩余全部驳回：样本抽中的通过的，不需要标注了，未通过和样本未抽中的需要重新标注验收。

查看验收报告

针对进行中或已完成的标注任务，都可以查看其验收报告。登录管理控制台，选择“数据管理>数据标注”，在数据标注页选择“我创建的”，并单击某条团队标注的任务名称，进入标注详情页。在右上角单击“验收报告”，即可在弹出的“验收报告”对话框中查看详情。

删除标注任务

验收结束后，针对不再使用的标注任务，您可单击任务所在行的删除。任务删除后，未验收的标注详情将丢失，请谨慎操作。但是数据集中的原始数据以及完成验收的标注数据仍然存储在对应的OBS桶中。

6 开发环境

6.1 开发环境介绍

说明

本手册基于新版开发环境Notebook功能展开介绍。

软件开发的历史，就是一部降低开发者成本，提升开发体验的历史。在AI开发阶段，ModelArts也致力于提升AI开发体验，降低开发门槛。ModelArts开发环境，以云原生的资源使用和开发工具链的集成，目标为不同类型AI开发、探索、教学用户，提供更好云化AI开发体验。

ModelArts Notebook云上云下，无缝协同

- 代码开发与调测。云化JupyterLab使用，本地IDE+ModelArts插件远程开发能力，贴近开发人员使用习惯
- 云上开发环境，包含AI计算资源，云上存储，预置AI引擎
- 运行环境自定义，将开发环境直接保存成为镜像，供训练、推理使用

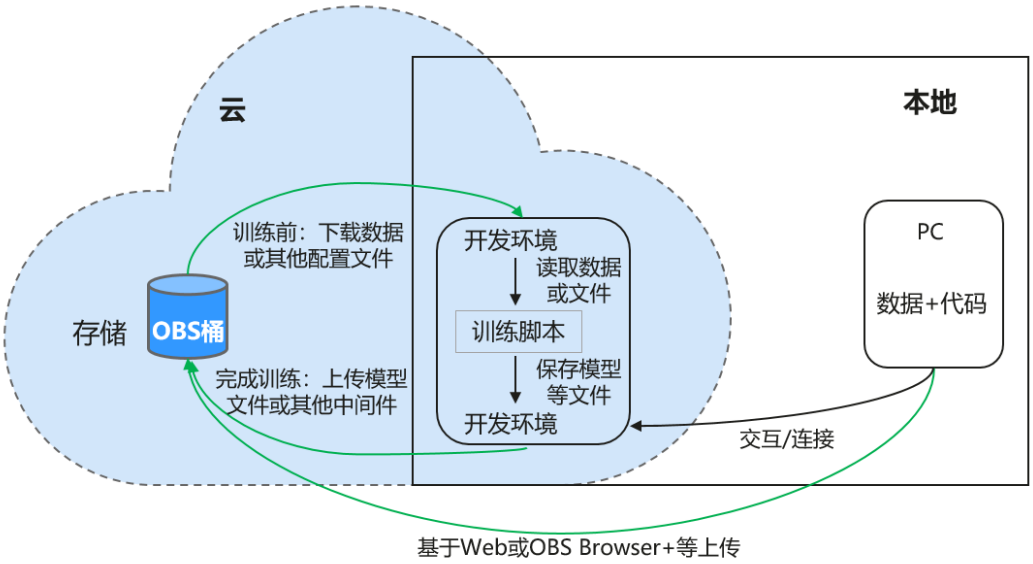
亮点特性 1：远程开发 - 支持本地 IDE 远程访问 Notebook

新版Notebook提供了远程开发功能，通过开启SSH连接，用户本地IDE可以远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

对于使用本地IDE的开发者，由于本地资源限制，运行和调试环境大多使用团队公共搭建的服务器，并且是多人共用，这带来一定的环境搭建和维护成本。

而ModelArts的Notebook的优势是即开即用，它预先装好了不同的AI引擎，并且提供了非常多的可选规格，用户可以独占一个容器环境，不受其他人的干扰。只需简单配置，用户即可通过本地IDE连接到该环境进行运行和调试。

图 6-1 本地 IDE 远程访问 Notebook 开发环境



ModelArts的Notebook可以视作是本地PC的延伸，均视作本地开发环境，其读取数据、训练、保存文件等操作与常规的本地训练一致。

对于习惯使用本地IDE的开发者，使用远程开发方式，不影响用户的编码习惯，并且可以方便快捷的使用云上的Notebook开发环境。

本地IDE当前支持VS Code、PyCharm、SSH工具。还有专门的插件PyCharm Toolkit和VS Code Toolkit，方便将云上资源作为本地的一个扩展。

亮点特性 2：开发环境保存 - 支持一键镜像保存

ModelArts的新版Notebook提供了镜像保存功能。支持一键将运行中的Notebook实例保存为镜像，将准备好的环境保存下来，可以作为自定义镜像，方便后续使用，并且方便进行分享。

保存镜像时，安装的依赖包（pip包）不丢失，VS Code远程开发场景下，在Server端安装的插件不丢失。

亮点特性 3：预置镜像 - 即开即用，优化配置，支持主流 AI 引擎

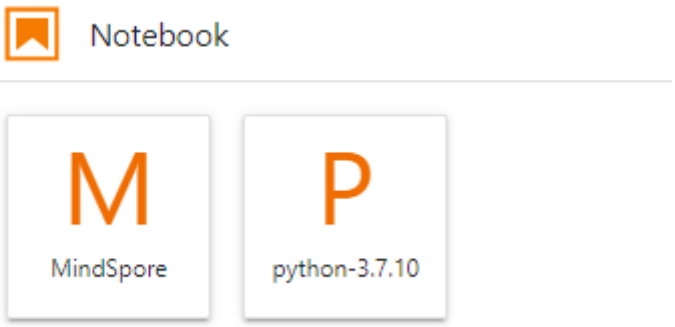
每个镜像预置的AI引擎和版本是固定的，在创建Notebook实例时明确AI引擎和版本，包括适配的芯片。

ModelArts开发环境给用户提供了一组预置镜像，主要包括PyTorch、TensorFlow、MindSpore系列。用户可以直接使用预置镜像启动Notebook实例，在实例中开发完成后，直接提交到ModelArts训练作业进行训练，而不需要做适配。

ModelArts开发环境提供的预置镜像版本是依据用户反馈和版本稳定性决定的。当用户的功能开发基于ModelArts提供的版本能够满足的时候，建议用户使用预置镜像，这些镜像经过充分的功能验证，并且已经预置了很多常用的安装包，用户无需花费过多的时间来配置环境即可使用。

ModelArts开发环境提供的预置镜像主要包含：

- 常用预置包，基于标准的Conda环境，预置了常用的AI引擎，例如PyTorch、MindSpore；常用的数据分析软件包，例如Pandas、Numpy等；常用的工具软件，例如cuda、cudnn等，满足AI开发常用需求。
- 预置Conda环境：每个预置镜像都会创建一个相对应的Conda环境和一个基础Conda环境python（不包含任何AI引擎），如预置MindSpore所对应的Conda环境如下：



用户可以根据是否使用AI引擎参与功能调试，选择不同的Conda环境。

- Notebook：是一款Web应用，能够使用户在界面编写代码，并且将代码、数学方程和可视化内容组合到一个文档中。
- JupyterLab插件：插件包括规格切换，停止实例等，提升用户体验。
- 支持SSH远程连接功能，通过SSH连接启动实例，在本地调试就可以操作实例，方便调试。

说明

- 为了简化操作，ModelArts的新版Notebook，同一个Notebook实例中支持不同引擎之间的切换。
- 不同Region支持的AI引擎不一样，请以控制台实际界面为准。

亮点特性 4：提供在线的交互式开发调试工具 JupyterLab

ModelArts集成了基于开源的JupyterLab，可为您提供在线的交互式开发调试。您无需关注安装配置，在ModelArts管理控制台直接使用Notebook，编写和调测模型训练代码，然后基于该代码进行模型的训练。

JupyterLab是一个交互式的开发环境，是Jupyter Notebook的下一代产品，可以使用它编写Notebook、操作终端、编辑MarkDown文本、打开交互模式、查看csv文件及图片等功能。

6.2 使用场景

ModelArts提供灵活开放的开发环境，您可以根据实际情况选择。

- ModelArts提供了云化版本的Notebook，无需关注安装配置，即开即用，具体参见**JupyterLab简介及常用操作**。
- ModelArts也提供了本地IDE的方式开发模型，通过开启SSH连接，用户本地IDE可以远程连接到ModelArts的Notebook开发环境中，调试和运行代码。本地IDE方式不影响用户的编码习惯，并且可以方便快捷的使用云上的Notebook开发环境。

本地IDE当前支持VS Code、PyCharm、SSH工具。PyCharm和VS Code还分别有专门的插件PyCharm Toolkit、VS Code Toolkit，让远程连接操作更便捷。具体参见。

6.3 管理 Notebook 实例

6.3.1 创建 Notebook 实例

在开始进行模型开发前，您需要创建Notebook实例，并打开Notebook进行编码。

约束与限制

- 只有处于“运行中”状态的Notebook，才可以执行打开、停止操作。
- 一个账户最多创建10个Notebook。

创建 Notebook 实例

1. 登录ModelArts管理控制台，在左侧导航栏中选择“全局配置”，检查是否配置了访问授权。若未配置，请先配置访问授权。参考使用委托授权完成操作。
2. 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。
3. 单击右上角“创建”，进入“创建Notebook”页面，请参见如下说明填写参数。
   a. 填写Notebook基本信息，包含名称、描述、是否自动停止，详细参数请参见表6-1。

   表 6-1 基本信息的参数描述

参数名称	说明
“名称”	Notebook的名称。系统会自动生成一个名称，您可以根据业务需求重新命名，命名规则如下：只能包含数字、大小写字母、下划线和中划线，长度不能超过128位且不能为空。
“描述”	对Notebook的简要描述。
“自动停止”	默认开启，且默认值为“1小时”，表示该Notebook实例将在运行1小时之后自动停止。可选择“1小时”、“2小时”、“4小时”、“6小时”或“自定义”几种模式。选择“自定义”模式时，可指定1~24小时范围内任意整数。 ● 定时停止：开启定时停止功能后，该Notebook实例将在运行时长超出您所选择的时长后，自动停止。 **说明** 出于对用户任务进度的保护，在您设置的自动停止时间到达后，Notebook不会立即自动停止，可能会有2-5分钟的延迟，方便您进行续约。

   b. 填写Notebook详细参数，如镜像、资源规格等，详细参数请参见表6-2。

表 6-2 Notebook 实例的详细参数说明

参数名称	说明
“镜像”	支持公共镜像和自定义镜像。 ● 公共镜像：即预置在ModelArts内部的AI引擎。 ● 自定义镜像：可以将基于公共镜像创建的实例保存下来，作为自定义镜像使用。 一个镜像对应支持一种AI引擎，创建Notebook实例时选择好了对应AI引擎的镜像。用户可以根据需要选择镜像。在右侧搜索框中输入镜像名称关键字，可快速查找镜像。 Notebook运行停止后，可以在同一个Notebook实例中变更镜像。
“资源类型”	根据实际使用需要选择资源池。
“类型”	芯片类型包括CPU、GPU和ASCEND类型。 不同的镜像支持的芯片类型不同，根据实际需要选择。
“规格”	根据选择的芯片类型不同，可选资源规格也不同。请根据界面实际情况和需要选择。

参数名称	说明
“存储配置”	根据选择的资源类型和规格不同，存储配置也不同，请根据界面实际情况和需要选择。 • 选择“对象存储服务OBS”或“并行文件系统PFS”作为存储位置。 选择“存储位置”：设置用于存储Notebook数据的OBS路径。如果想直接使用已有的文件或数据，可将数据提前上传至对应的OBS路径下。“存储位置”不能设置为OBS桶的根目录，需设置为对应OBS桶下的具体目录。 选择“凭据”：选择已有的凭据或单击右侧的“立即创建”，跳转至数据加密控制台**创建凭据**，凭据键/值填写用户的AK、SK信息（“键”分别填写“accessKeyId”，“secretAccessKey”；“值”在控制台个人账号下“我的凭证>访问密钥”获取AK、SK）。 **图 6-2** 设置凭据值  “云硬盘EVS”和“弹性文件服务SFS”的存储路径挂载在/home/ma-user/work目录下。用户在Notebook实例中的所有文件读写操作都是针对该存储目录下的内容操作，与OBS对象存储（OBS对象桶）无关。 停止或重启Notebook实例时，存储的内容会被保留，不丢失。 删除Notebook实例时，EVS存储会一起释放，存储的内容不保留。SFS可以重新挂载到新的Notebook，可以保留数据。

参数名称	说明
“扩展存储配置”	**说明** “扩展存储配置”功能是白名单功能，如果有试用需求，请提工单申请权限。 如果有多个数据存储路径，可以单击“增加扩展存储配置”，增加用户指定的存储挂载目录。支持增加的存储类型有“存储桶OBS”、“并行文件系统PFS”、“弹性文件服务SFS”。 约束限制： ● 每种存储类型最多支持挂载5个。 ● 扩展存储挂载目录不允许重复，不允许挂载到黑名单目录，允许嵌套挂载。不允许挂载的黑名单目录为以下前缀匹配的目录：/data/、/cache/、/dev/、/etc/、/bin/、/lib/、/sbin/、/modelarts/、/train-worker1-log/、/var/、/resource_info/、/usr/、/sys/、/run/、/tmp/、/infer/、/opt/ 添加扩展存储后，可进入Notebook实例详情页，单击“存储配置 > 扩展存储”，查看或编辑扩展存储信息。在存储个数未达到最大个数时，也可在右侧单击“添加扩展存储”。
“SSH远程开发”	● 开启此功能后，用户可以在本地开发环境中远程接入Notebook实例的开发环境。 ● 实例在停止状态时，用户可以在Notebook详情页中更新SSH的配置信息。 **说明** 开启此功能的实例中会预置VS Code插件（python、jupyter等）以及VS Code Server包，会占用约1G左右的持久化存储空间。
“密钥对”	开启“SSH远程开发”功能后，需要设置此参数。 可以选择已有密钥对。 也可以单击密钥对右侧的“立即创建”，跳转到数据加密控制台，在“密钥对管理 > 私有密钥对”页面，单击“创建密钥对”。 创建完Notebook后，可以在Notebook详情页中修改密钥对。 **注意** 创建好的密钥对，请下载并妥善保存，使用本地IDE远程连接云上Notebook开发环境时，需要用到密钥对进行鉴权认证。

参数名称	说明
“远程访问白名单”	可选，开启“SSH远程开发”功能后，可以设置此参数。 设置为允许远程接入访问这个Notebook的IP地址（例如本地PC的IP地址或者访问机器的外网IP地址，最多配置5个，用英文逗号隔开），不设置则表示无接入IP地址限制。 如果用户使用的访问机器和ModelArts服务的网络有隔离，则访问机器的外网地址需要在主流搜索引擎中搜索“IP地址查询”获取，而不是使用**ipconfig**或**ifconfig/ip**命令在本地查询。 **图 6-3** 查询外网 IP 地址  创建完Notebook后，可以在Notebook详情页中修改白名单IP地址。

4. 参数填写完成后，单击“立即创建”进行规格确认。
5. 参数确认无误后，单击“提交”，完成Notebook的创建操作。

   进入Notebook列表，正在创建中的Notebook状态为“创建中”，创建过程需要几分钟，请耐心等待。当Notebook状态变为“运行中”时，表示Notebook已创建并启动完成。
6. 在Notebook列表，单击实例名称，进入实例详情页，查看Notebook实例配置信息。

   “SSH远程开发”功能开启时，在“白名单”右侧单击修改，可以修改允许远程访问的白名单IP地址。实例在停止状态时，在“认证”右侧单击修改，用户可以更新密钥对。

   如果存储使用的是云硬盘EVS，单击存储容量右侧的“扩容”，可以动态扩充云硬盘EVS的容量，具体操作参见**动态扩充云硬盘EVS容量**。

6.3.2 打开 Notebook 实例

针对创建好的Notebook实例（即状态为“运行中”的实例），可以打开Notebook并在开发环境中启动编码。

基于不同AI引擎创建的Notebook实例，打开方式不一样。

- 本地IDE使用PyCharm/VS Code/SSH工具，远程连接访问，具体参见**VS Code一键连接Notebook**。
- 在线JupyterLab访问，具体参见**JupyterLab简介及常用操作**。

创建实例，持久化存储挂载路径为/home/ma-user/work目录。

```
sh-4.4$pwd
/home/ma-user
sh-4.4$cd work/
sh-4.4$pwd
/home/ma-user/work
sh-4.4$
```

存放在work目录的内容，在实例停止、重新启动后依然保留，其他目录下的内容不会保留，使用开发环境时建议将需要持久化的数据放在/home/ma-user/work目录。

6.3.3 查找/启动/停止/删除实例

查找实例

Notebook页面展示了所有创建的实例。若需要查找特定的实例，可根据筛选条件快速查找。单击搜索框，可按照属性类型选择单个条件来筛选，或者同时选择多个筛选条件筛选。

- 打开“查看所有”开关，可以看到IAM项目下所有子用户创建的Notebook实例。
- 按实例名称、实例ID、实例状态、使用的镜像、实例规格、实例描述、创建时间等单个筛选或组合筛选。

表格显示设置

单击  按钮可自定义设置需要显示在表格中的列。

启动/停止实例

由于运行中的Notebook将一直耗费资源，您可以通过停止操作，停止资源消耗。对于停止状态的Notebook，可通过启动操作重新使用Notebook。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook管理页面。
2. 执行如下操作启动或停止Notebook。
   - **启动Notebook**：单击“操作”列的“启动”。只有处于“停止”状态的Notebook可以执行启动操作。
   - **停止Notebook**：单击“操作”列的“停止”。只有处于“运行中”状态的Notebook可以执行停止操作。

⚠ 注意

Notebook停止后：

- /home/ma-user/work目录下的数据会保存，其余目录下内容会被清理。例如：用户在开发环境中的其他目录下安装的外部依赖包等，在Notebook停止后会被清理。

删除实例

针对不再使用的Notebook实例，可以删除以释放资源。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境>Notebook”，进入Notebook页面。
2. 在Notebook列表中，单击操作列的“删除”，在弹出的确认对话框中，确认信息无误，然后输入“DELETE”，单击“确定”，完成删除操作。

> **⚠ 注意**
>
> Notebook删除后不可恢复，请谨慎操作。实例删除后，挂载目录下的数据也将一并删除，请谨慎操作。

6.3.4 变更 Notebook 实例镜像

ModelArts允许用户在同一个Notebook实例中切换镜像，方便用户灵活调整实例的AI引擎。

约束限制

Notebook实例状态必须在“停止”中。

变更实例镜像操作

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 在Notebook列表，单击某个Notebook实例操作栏的“更多 > 变更镜像”，在变更镜像窗口选择新的镜像，单击“确定”。
3. 在镜像窗口选择新的镜像，单击“确定”，变更成功后，在Notebook列表页的镜像栏，可以查看到变更后的镜像。

6.3.5 变更 Notebook 实例运行规格

ModelArts允许用户在同一个Notebook实例中切换节点运行规格，方便用户灵活调整规格资源。

约束限制

只有处于“停止”、“运行中”和“启动失败”的Notebook实例才能变更规格。

变更实例规格操作

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 在Notebook列表，单击某个Notebook实例操作列的“更多 > 变更规格”，在弹出的“变更规格”对话框中选择对应规格资源。

6.3.6 开发环境中如何选择存储

不同存储的实现都不同，在性能、易用性、成本的权衡中可以有不同的选择，没有一个存储可以覆盖所有场景，了解下云上开发环境中各种存储使用场景说明，更能提高使用效率。

说明

仅支持挂载同一区域下的OBS并行文件系统（PFS）和OBS对象存储。

表 6-3 云上开发环境中各种存储使用场景说明

存储类型	建议使用场景	优点	缺点
云硬盘EVS	比较适合只在开发环境中做数据、算法探索，性能较好。	块存储SSD，可以理解为一个磁盘，整体IO性能比NFS要好，可以动态扩充，最大可以到4096GB。 云硬盘EVS作为持久化存储挂载在/home/ma-user/work目录下，该目录下的内容在实例停止后会被保留，存储支持在线按需扩容。	缺点是只能在单个开发环境中使用 。
并行文件系统PFS	**说明** 并行文件系统PFS为白名单功能，如需使用，请联系技术支持开通。 适合直接使用PFS桶作为持久化存储进行AI开发和探索场景。 1. 数据集的存储。将数据集直接挂载到Notebook进行浏览和数据处理，在训练时直接使用。本地数据上传请参考**如何上传文件到OBS?** 2. 代码的存储。在Notebook调测完成，可以直接指定对应的对象存储路径作为启动训练的代码路径，方便临时修改。 3. 训练观测。可以将训练日志等输出路径进行挂载，在Notebook中实时查看和观测，特别是利用TensorBoard，Notebook功能完成对训练输出的分析。	PFS是一种经过优化的高性能对象存储文件系统，存储成本低，吞吐量大，能够快速处理高性能计算（HPC）工作负载。在需要使用对象存储服务场景下，推荐使用PFS挂载。 **说明** 建议上传时按照128MB或者64MB打包或者切分，使用时边下载边解压后在本地存储读取，以获取更好的读写与吞吐性能。	缺点是小文件频繁读写性能较差，例如直接作为存储用于模型重型训练，大文件解压等场景慎用。 **说明** PFS挂载需要用户对当前桶授权给ModelArts完整读写权限，Notebook删除后，此权限策略不会被删除。

存储类型	建议使用场景	优点	缺点
对象存储服务OBS	**说明** OBS对象存储为白名单功能，如需使用，请联系技术支持开通。 在开发环境中做大规模的数据上传下载时，可以通过OBS桶做中转。	存储成本低，吞吐量大，但是小文件读写较弱。建议上传时按照128MB或者64MB打包或者切分，使用时边下载边解压后在本地读取。	对象存储语义，和Posix语义有区别，需要进一步理解。
弹性文件服务SFS	目前只支持在专属资源池中使用；针对探索、实验等非正式生产场景，建议使用这种。开发环境和训练环境可以同时挂载一块SFS存储，省去了每次训练作业下载数据的要求，一般来说重IO读写模型，超过32卡的大规模训练不适合。	实现为NFS，可以在多个开发环境、开发环境和训练之间共享，如果不需要重型分布式训练作业，特别是启动训练作业时，不需要额外再对数据进行下载，这种存储便利性可以作为首选。	缺点性能比EVS云硬盘块存储低。
本地存储	重型训练任务首选	运行所在虚拟机或者裸金属机器上自带的SSD高性能存储，文件读写的吞吐量大，建议对于重型训练任务先将数据准备到对应目录再启动训练。 默认在容器/cache目录下进行挂载，/cache目录可用空间请参考**开发环境中不同Notebook规格资源“/cache”目录的大小**。	缺点是存储生命周期和容器生命周期绑定，每次训练都要下载数据。

如何使用

在开发环境中如何使用云硬盘EVS块存储?

例如，在**创建Notebook实例**时选择云硬盘EVS存储小容量，Notebook运行过程中如果发现存储容量不够，可以扩容，请参考**动态扩充云硬盘EVS容量**。

6.3.7 动态扩充云硬盘 EVS 容量

什么是动态扩容 EVS

存储配置采用云硬盘EVS的Notebook实例， 存储盘是挂载至容器/home/ma-user/work/目录下， 可以在实例运行中的状态下，动态扩充存储盘容量，单次最大动态扩容100GB。

动态扩容 EVS 适用于哪些使用场景

在Notebook开发过程中，初期存储使用量较小时， 创建Notebook可以选择小容量EVS， 比如5G大小； 开发完成后，需要大规模数据集训练，此时再将存储容量扩容至当前阶段所需容量，可以节约成本。

动态扩容 EVS 有什么限制

- Notebook实例的存储配置采用的是云硬盘EVS。
- 单次最大可以扩容100GB，扩容后的总容量不超过4096GB。
- 云硬盘EVS存储容量最大支持4096GB，达到4096GB时，不允许再扩容。
- 实例停止后，扩容后的容量仍然有效。

动态扩容 EVS 操作

1. 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。
2. 选择运行中的Notebook实例，单击实例名称，进入Notebook实例详情页面，单击“扩容”。

   图 6-4 Notebook 实例详情页

3. 设置待扩充的存储容量大小，单击“确定”。系统显示“扩容中”，扩容成功后，可以看到扩容后的存储容量。

6.3.8 修改 Notebook SSH 远程连接配置

ModelArts允许用户在Notebook实例中更改SSH配置信息。

在创建Notebook实例时，如果未配置SSH远程连接，当用户需要开启远程连接时，则可以在Notebook的实例详情页打开SSH的配置信息开关；

在创建Notebook实例时，如果设置了允许远程连接Notebook的白名单IP地址，当用户需要更换一个IP地址远程连接Notebook实例时，则可以在Notebook的实例详情页修改白名单IP地址，也可更换密钥对。

约束限制

Notebook实例状态必须在“停止”中。

修改密钥对和远程连接 IP 地址

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 在Notebook列表，进入Notebook实例详情页。打开SSH远程开发开关，更新密钥对和白名单。

   说明

   “远程SSH开发”开关可以手动打开的场景，请打开远程SSH开发开关，参考图6-5操作。SSH配置信息更新后，“远程SSH开发”开关打开后不可关闭。

   “所选镜像必须配置SSH远程开发”的场景，请参考图6-6操作。

图 6-5 更新 SSH 配置信息

图 6-6 修改白名单和密钥对

- 密钥对可单击 ▼ 选择已有的密钥对或“立即创建”创建新的密钥对。
- 白名单IP地址的设置请参考设置远程连接IP地址。修改远程连接的可访问IP地址后， 原来已经建立的链接依然有效， 当链接关闭后失效；新打开建立的链接只允许当前设置的IP进行访问。

设置远程连接 IP 地址

图 6-7 设置远程连接 IP 地址

此处的IP地址，请填写外网IP地址。如果用户使用的访问机器和ModelArts服务的网络有隔离，则访问机器的外网地址需要在主流搜索引擎中搜索“IP地址查询”获取，而不是使用ipconfig或ifconfig/ip命令在本地查询。

图 6-8 查询外网 IP 地址

6.3.9 查看所有子账号的 Notebook 实例

当子用户被授予“listAllNotebooks”和“listUsers”权限时，在Notebook页面上，单击“查看所有”，可以看到IAM项目下所有子用户创建的Notebook实例 。

说明

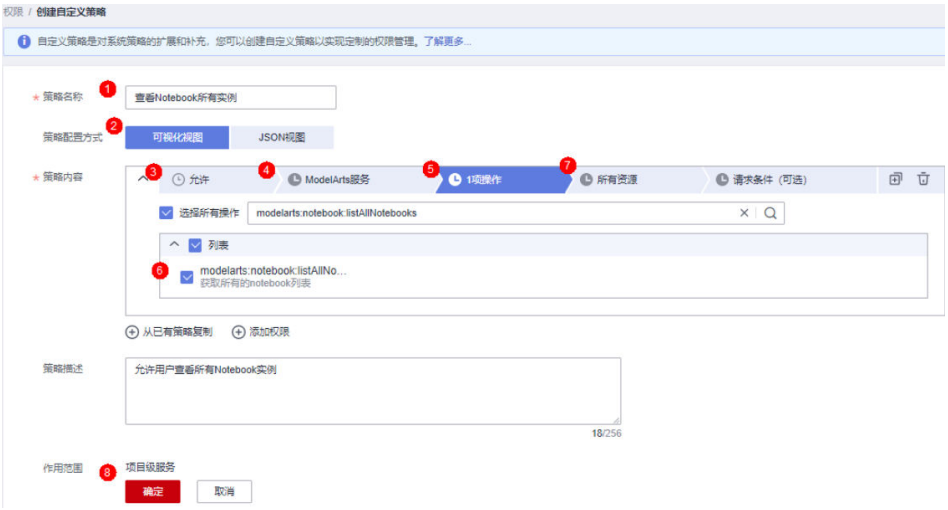
配置该权限后，可以查看子用户的Notebook，也可以在Notebook中访问子用户的OBS、SWR等。

给子账号配置查看所有 Notebook 实例的权限

1. 使用主用户账号登录ModelArts管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。
2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”，需要设置两条策略。

   策略1：设置查看Notebook所有实例，如图6-9所示，单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：查看Notebook所有实例。
   - “策略配置方式”：选择可视化视图。
   - “策略内容”：允许，云服务中搜索ModelArts服务并选中，操作列中搜索关键词modelarts:notebook:listAllNotebooks并选中，所有资源选择默认值。

图 6-9 创建自定义策略



策略2：设置查看Notebook实例创建者信息的策略。

- “策略名称”：设置自定义策略名称，例如：查看所有子用户信息。
- “策略配置方式”：选择可视化视图。
- “策略内容”：允许，云服务中搜索IAM服务并选中，操作列中搜索关键词iam:users:listUsers并选中，所有资源选择默认值。

3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的两条自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限查看该用户组内成员创建的所有Notebook实例。

   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

子用户启动其他用户的 SSH 实例

子用户可以看到所有用户的Notebook实例后，如果要通过SSH方式远程连接其他用户的Notebook实例，需要将SSH密钥对更新成自己的，否则会报错ModelArts.6789，更新密钥对具体操作请参见**修改Notebook SSH远程连接配置**。

具体的错误信息提示：ModelArts.6789：当前用户没有权限使用ssh密钥对xxx，请更新实例密钥对并重试。

6.4 JupyterLab

6.4.1 JupyterLab 操作流程

ModelArts支持通过JupyterLab工具在线打开Notebook，开发基于PyTorch、TensorFlow和MindSpore引擎的AI模型。具体操作流程如下图所示。

图 6-10 使用 JupyterLab 在线开发调试代码

1. 创建Notebook实例。

   在ModelArts控制台创建一个Notebook开发环境实例，选择要使用的AI框架。具体参见**创建Notebook实例**。
2. 使用JupyterLab打开Notebook实例。具体参见**打开JupyterLab**。
3. 准备训练数据和代码文件，上传到JupyterLab中。具体参见**上传本地文件至JupyterLab**。
4. 在JupyterLab中编写代码文件，并运行调试。具体参见**JupyterLab简介及常用操作**。
5. 在JupyterLab中直接调用ModelArts提供的SDK，创建训练作业，上云训练。

   调用SDK创建训练作业的操作请参见**使用ModelArts SDK**。

6.4.2 JupyterLab 简介及常用操作

JupyterLab是一个交互式的开发环境，是Jupyter Notebook的下一代产品，可以使用它编写Notebook、操作终端、编辑MarkDown文本、打开交互模式、查看csv文件及图片等功能。

可以说，JupyterLab是开发者们下一阶段更主流的开发环境。JupyterLab具有和Jupyter Notebook一样的组件，但支持更加灵活和更加强大的项目操作方式。

打开 JupyterLab

下面介绍如何从运行中的Notebook实例打开JupyterLab。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入Notebook页面。
2. 选择状态为“运行中”的Notebook实例，单击操作列的“打开”，访问JupyterLab。

   图 6-11 打开 Notebook 实例

3. 进入JupyterLab页面后，自动打开Launcher页面，如下图所示。您可以使用开源支持的所有功能，详细操作指导可参见**JupyterLab官网文档**。
   - Notebook：选择运行Notebook的一个内核，例如TensorFlow、python
   - Console：可调出终端进行命令控制
   - Other：可编辑其他文件

在 JupyterLab 中新建 ipynb 文件

进入JupyterLab主页后，可在“Notebook”区域下，选择适用的AI引擎，单击后将新建一个对应框架的ipynb文件。

由于每个Notebook实例选择的工作环境不同，其支持的AI框架也不同，下图仅为示例，请根据实际显示界面选择AI框架。

新建的ipynb文件将呈现在左侧菜单栏中。

新建文件并打开 Console

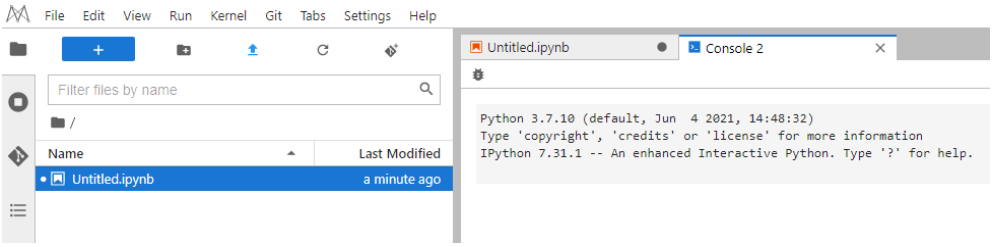
Console的本质为Python终端，输入一条语句就会给出相应的输出，类似于Python原生的IDE。

进入JupyterLab主页后，可在“Console”区域下，选择适用的AI引擎，单击后将新建一个对应框架的Notebook文件。

由于每个Notebook实例选择的工作环境不同，其支持的AI框架也不同，下图仅为示例，请根据实际显示界面选择AI框架。

文件创建成功后，将直接呈现Console页面。

图 6-12 新建文件（Console）

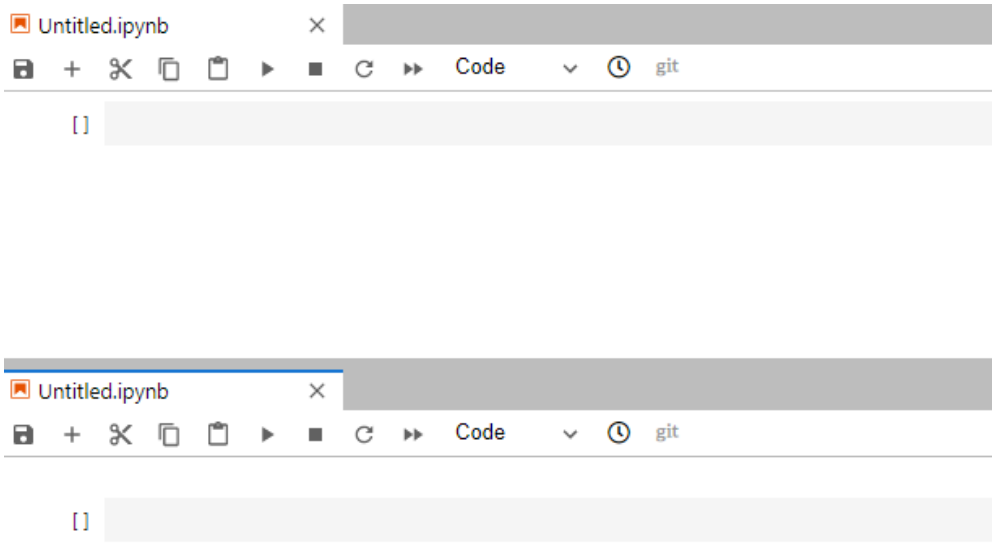


在 JupyterLab 中编辑文件

JupyterLab可以在同一个窗口同时打开几个Notebook或文件（如HTML、TXT、Markdown等），以页签形式展示。

JupyterLab的一大优点是，可以任意排版多个文件。在右侧文件展示区，您可以拖动打开文件，随意调整文件展示位置，可以同时打开多个文件。

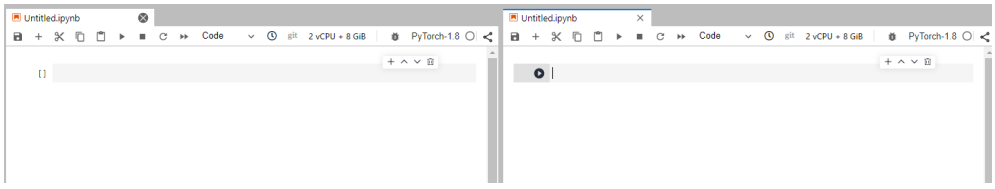
图 6-13 多文件任意编排



当在一个Notebook中写代码时，如果需要实时同步编辑文件并查看执行结果，可以新建该文件的多个视图。

打开ipynb文件，然后单击菜单栏“File > New View for Notebook”，即可打开多个视图。

图 6-14 同一个文件的多个视图



JupyterLab的ipynb文件代码栏中输入代码，需要在代码前加!符号。

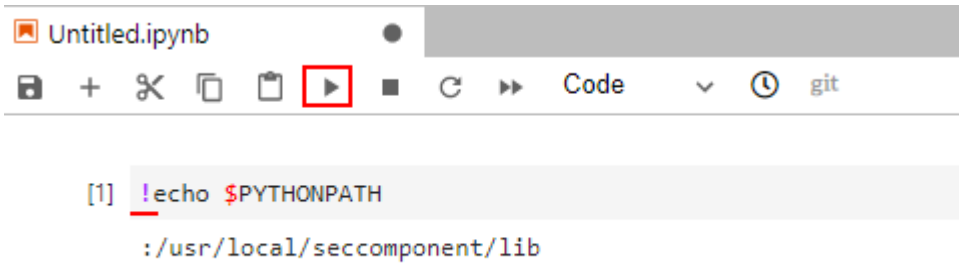
例如：安装外部库Shapely

```
!pip install Shapely
```

例如：查看PythonPath

```
!echo $PYTHONPATH
```

图 6-15 运行代码



自动停止及续期

在创建或启动Notebook时，如果启用了自动停止功能，则在JupyterLab的右上角会显示当前实例停止的剩余时长，在计时结束前可以单击剩余时间进行续期。

图 6-16 自动停止

JupyterLab 常用快捷键和插件栏

图 6-17 JupyterLab 常用快捷键和插件栏

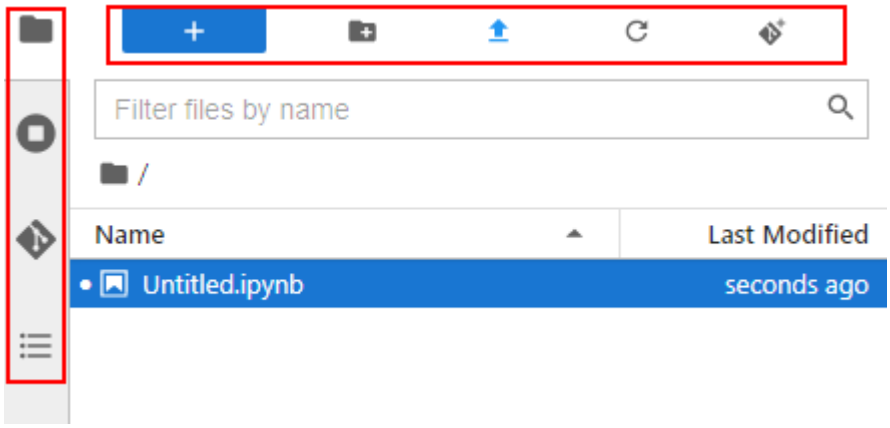


表 6-4 快捷键说明

快捷键	说明
+	快速打开Notebook、Terminal。或打开Launcher页面，可快速创建新的Notebook、Console或其他文件。
	创建文件夹。

快捷键	说明
	上传文件。
	刷新文件目录。
	Git插件，可连接此Notebook实例关联的Github代码库。

表 6-5 插件栏常用插件说明

插件	说明
	文件列表。单击此处，将展示此Notebook实例下的所有文件列表。
	当前实例中正在运行的Terminal和Kernel。
	Git插件，可以方便快捷的使用Github代码库。
	属性检查器。
	文档结构图。

图 6-18 导航栏按钮

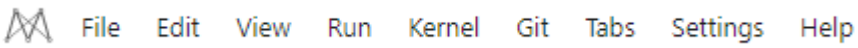


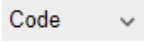
表 6-6 导航栏按钮介绍

按钮	说明
File	新建、关闭、保存、重新加载、重命名、导出、打印Notebook等功能。
Edit	编辑ipynb文件中代码块的相关操作，包括撤销、重做、剪切、复制、粘贴、选择、移动、合并、清除、查找代码块等。
View	查看视图相关操作。
Run	运行代码块相关操作，例如：运行选中代码块、一键运行所有代码块等。
Kernel	中断、重启、关闭、改变Kernel相关操作。

按钮	说明
Git	Git插件相关操作，可以方便快捷的使用Github代码库。
Tabs	同时打开多个ipynb文件时，通过Tabs激活或选择文件。
Settings	JupyterLab工具系统设置。
Help	JupyterLab工具自带的帮助参考。

图 6-19 ipynb 文件菜单栏中的快捷键

表 6-7 ipynb 文件菜单栏中的快捷键

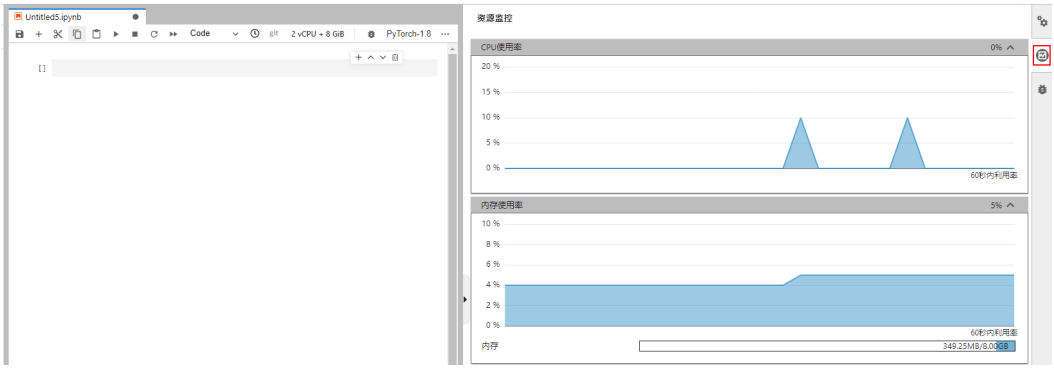
快捷键	说明
	保存文件。
	添加新代码块。
	剪切选中的代码块。
	复制选中的代码块。
	粘贴选中的代码块。
	执行选中的代码块。
	终止kernel。
	重启kernel。
	重启kernel，然后重新运行当前Notebook的所有代码。
	此处下拉框有4个选项，分别是： Code（写python代码），Markdown（写Markdown代码，通常用于注释），Raw（一个转换工具），-（不修改）。
	查看代码历史版本。
	git插件，图标显示灰色表示当前Region不支持。

快捷键	说明
2 vCPU + 4 GiB	当前的资源规格。
PyTorch-1.4	单击可以选择Kernel。
○	表示代码运行状态，变为实心圆●时，表示代码在运行中。

资源监控

在使用过程中，如果想了解资源使用情况，可在右侧区域选择“Resource Monitor”，展示“CPU使用率”和“内存使用率”。

图 6-20 资源监控



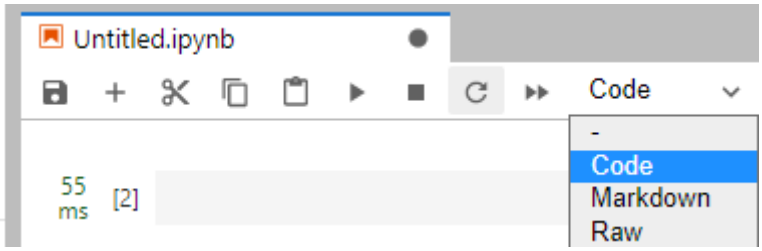
6.4.3 代码参数化插件

代码参数化插件可以降低Notebook案例的复杂度，用户无需感知复杂的源码，按需调整参数快速进行案例复现、模型训练等。该插件可用于定制Notebook案例，适用于比赛、教学等场景。

使用介绍

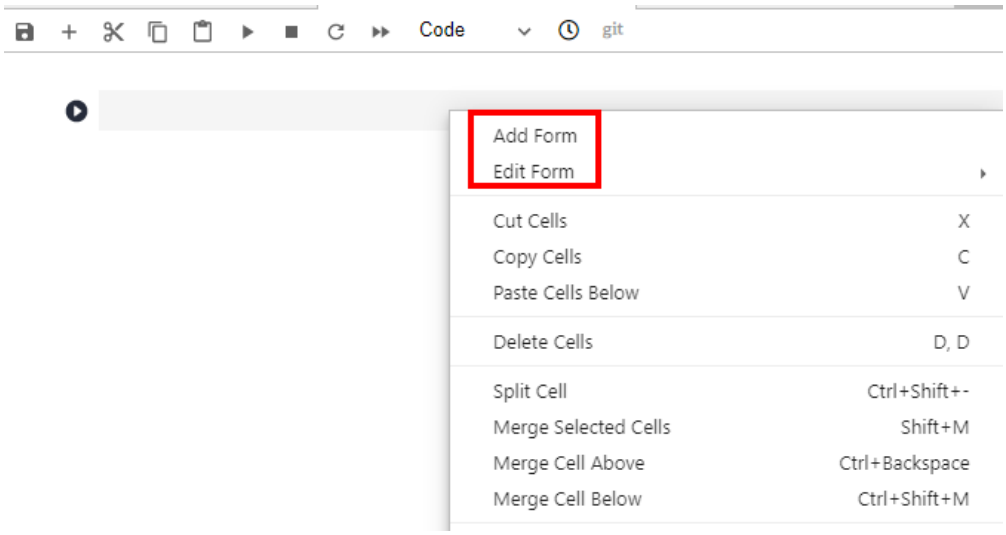
- 仅对Code cell类型新增了Edit Form和Add Form功能，如果cell类型是Markdown或者Raw类型则不支持。如下图所示：

  图 6-21 查看 Code cell

- 打开新的代码后，需先Add Form，再Edit Form。

图 6-22 Code 类型的 cell 右键选项



Add Form 介绍

“Add Form”按钮会将Code cell水平拆分为两种编辑区域，左侧为代码区域，右侧为表单区域。单击表单右侧的“Edit”可修改默认标题。

图 6-23 两种编辑区域

Edit Form 介绍

“Edit Form”按钮有四个子选项，分别是“Add new form field”、“Hide code”、“Hide form”和“show all”四个按钮，下文介绍这四个选项的功能。

- “Add new form field”按钮支持新增“dropdown”、“input”和“slider”类型的表单。如图6-24所示。每新增一个字段，会分别在代码和表单区域中增加对应的变量，修改表单区域的值也会同时修改代码变量值。

  说明

  创建dropdown类型的表单时，“ADD Item”至少创建2项。如图6-25所示。

图 6-24 “dropdown”，“input”，“slider”的表单样式

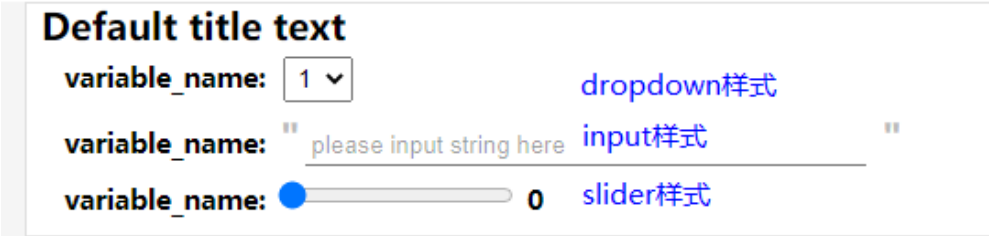


图 6-25 创建“dropdown”类型的表单

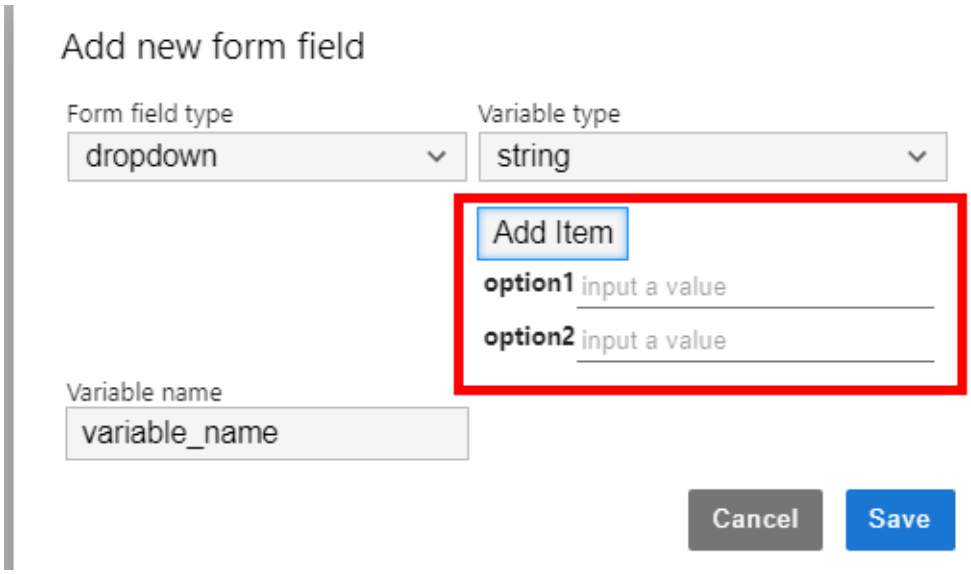


图 6-26 删除表单

  - 表单字段类型为“dropdown”时，支持的变量类型为“raw”和“string”。
  - 表单字段类型为“input”时，支持的变量类型有“boolean”、“date”、“integer”、“number”、“raw”和“string”。
  - 表单字段类型为“slider”时，支持输入滑动条的最小值、最大值和步长。
- “Hide code”按钮会隐藏代码区域。
- “Hide form”按钮会隐藏表单区域。
- “Show all”按钮会同时展示code和form区域。

6.4.4 使用 ModelArts SDK

在Notebook中，通过使用ModelArts SDK，可以完成OBS管理、训练作业管理、模型管理以及在线服务管理。

在Notebook中，已承载了登录用户的鉴权信息（AK/SK）和区域信息，因此SDK session鉴权时，无需输入参数即可完成session鉴权。

示例代码

- 创建训练作业

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
```

```
                modelarts_session=session,
                framework_type='PyTorch',                                   # AI引擎名称
                framework_version='PyTorch-1.0.0-python3.6',                # AI引擎版本
                code_dir='/obs-bucket-name/src/',                           # 训练脚本目录
                boot_file='/obs-bucket-name/src/pytorch_sentiment.py',      # 训练启动脚本目录
                log_url='/obs-bucket-name/log/',                            # 训练日志目录
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/obs-bucket-name/output/',                     # 训练输出目录
                train_instance_type='modelarts.vm.xxx.xxx',                 # 训练环境规格
                train_instance_count=1,                                     # 训练节点个数
                job_description='pytorch-sentiment with ModelArts SDK')     # 训练作业描述
job_instance = estimator.fit(inputs='/obs-bucket-name/data/train/', wait=False,
job_name='my_training_job')
```

- 查询模型列表

```
from modelarts.session import Session
from modelarts.model import Model
session = Session()
model_list_resp = Model.get_model_list(session, model_status="published", model_name="digit",
order="desc")
```

- 查询服务详情

```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_info_resp = predictor_instance.get_service_info()
```

6.4.5 使用 Git 插件

在JupyterLab中使用Git插件可以克隆GitHub开源代码仓库，快速查看及编辑内容，并提交修改后的内容。

前提条件

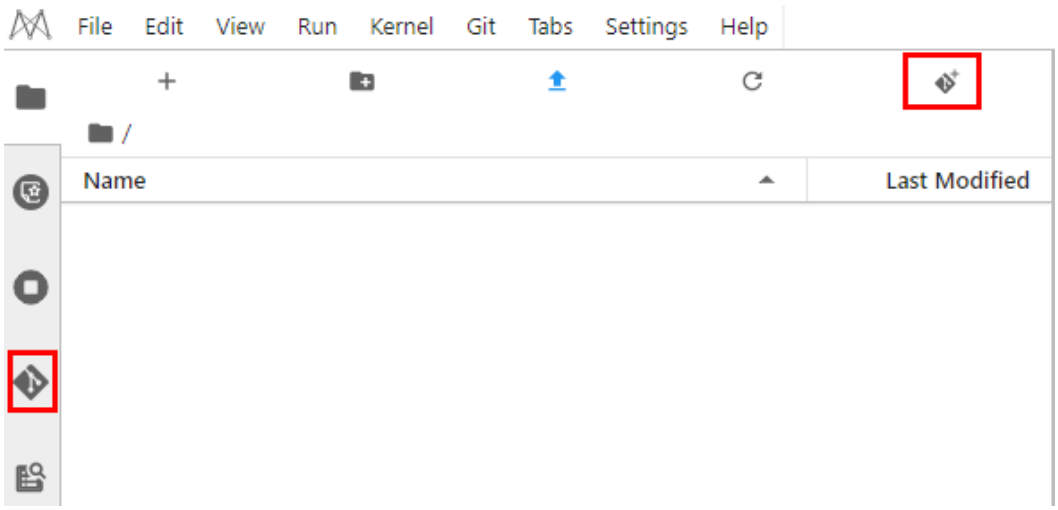
Notebook处于运行中状态。

打开 JupyterLab 的 git 插件

在Notebook列表中，选择一个实例，单击右侧的打开进入“JupyterLab”页面。

图6-27所示图标，为JupyterLab的Git插件。

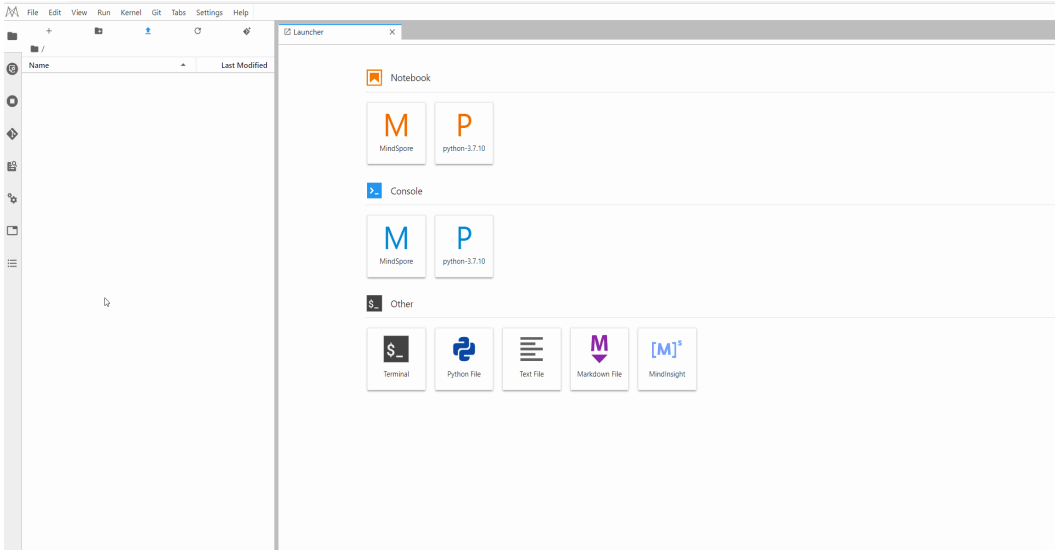
图 6-27 Git 插件



克隆 GitHub 的开源代码仓库

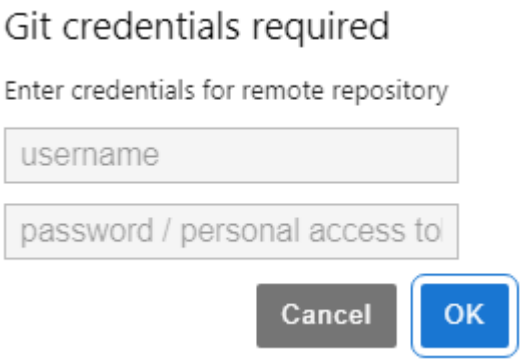
GitHub开源仓库地址：https://github.com/jupyterlab/extension-examplesitHub，单击 ，输入仓库地址，单击确定后即开始克隆，克隆完成后，JupyterLab左侧导航出现代码库文件夹。

图 6-28 使用 git 插件克隆 GitHub 的开源代码仓库



克隆 GitHub 的私有仓库

克隆GitHub私有仓库时，会弹出输入个人凭证的对话框，如下图。此时需要输入GitHub中Personal Access Token信息。



查看Personal Access Token步骤如下：

1. 登录**Github**，打开设置页面。
2. 单击“Developer settings”。
3. 单击“Personal access tokens > Generate new token”。
4. 验证登录账号。
5. 填写Token描述并选择权限，选择私有仓库访问权限，单击“Generate token”生成Token。
6. 复制生成的Token到编译构建服务即可。

须知

- Token生成后，请及时保存，下次刷新页面将无法读取，需要重新生成新Token。
- 注意填写有效的Token描述信息，避免误删除导致构建失败。
- 无需使用时及时删除Token，避免信息泄露。

图 6-29 克隆 GitHub 的私有仓库（目前只支持 Personal Access Token 授权）

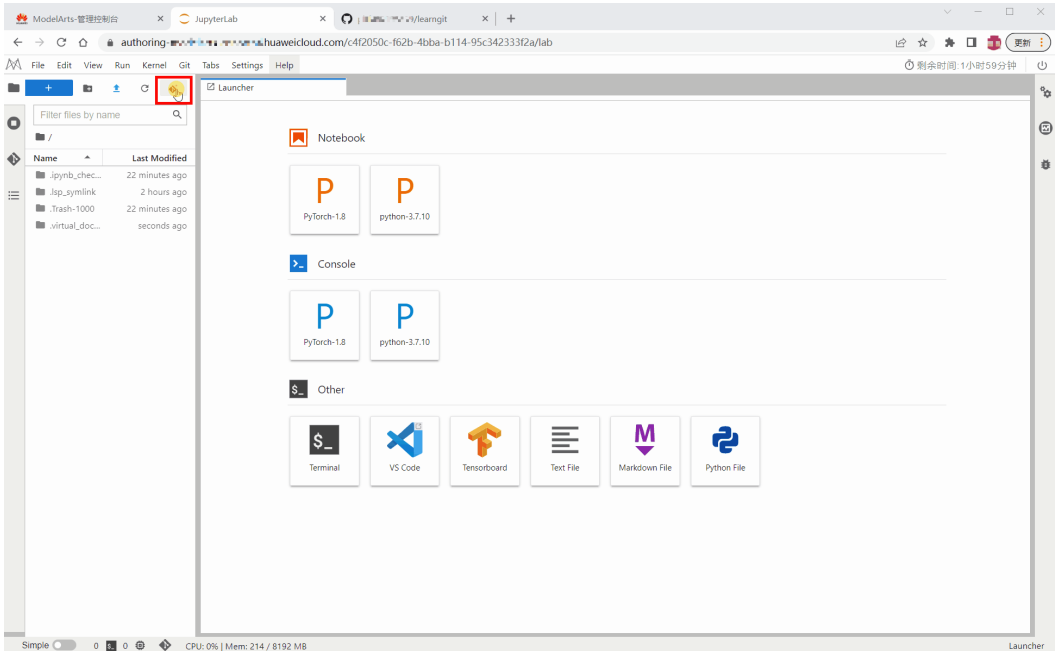
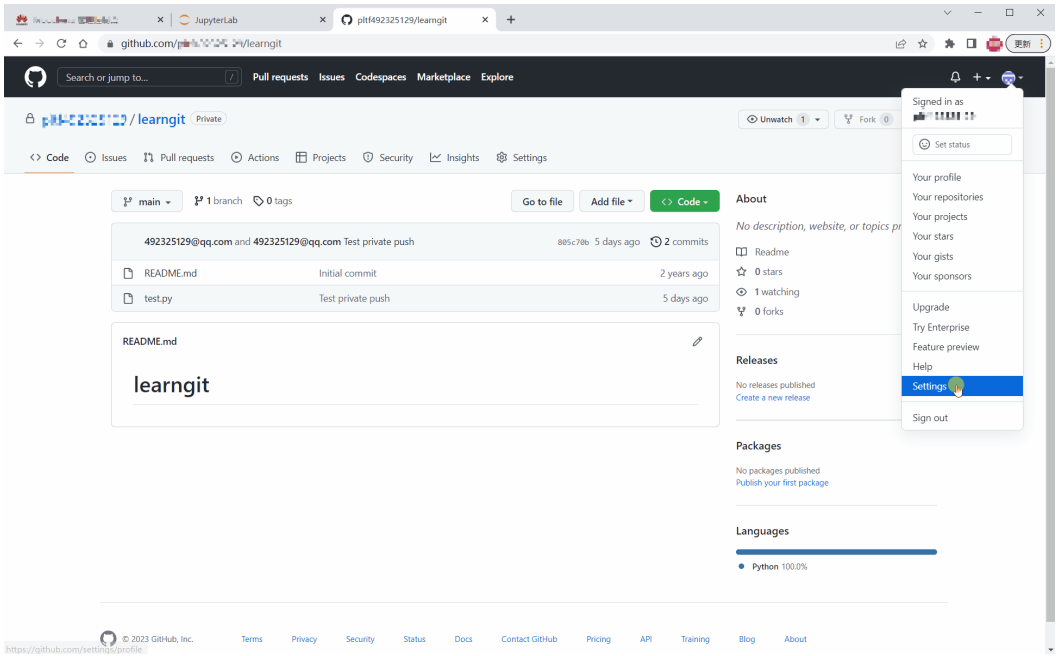


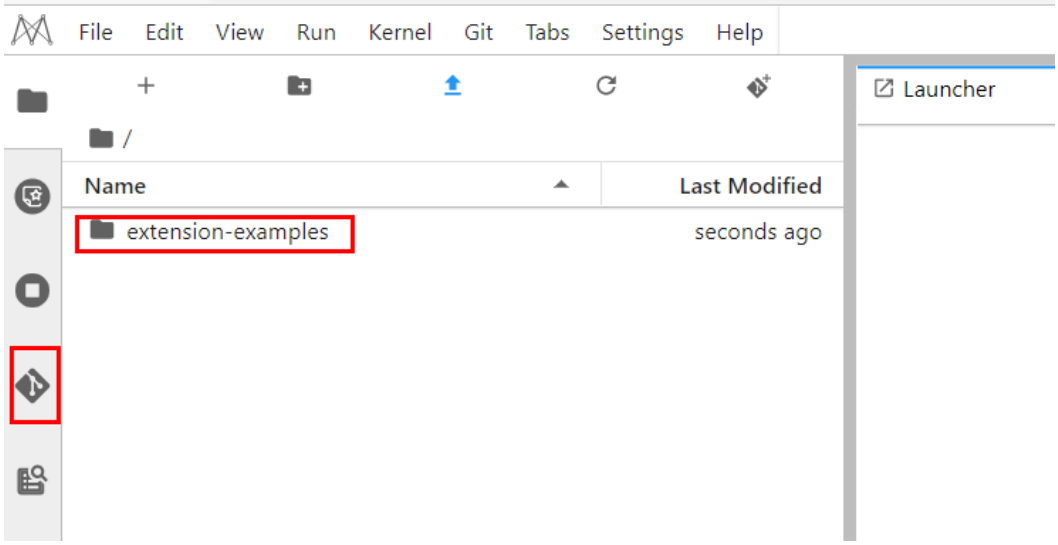
图 6-30 获取 Personal Access Token



查看代码库信息

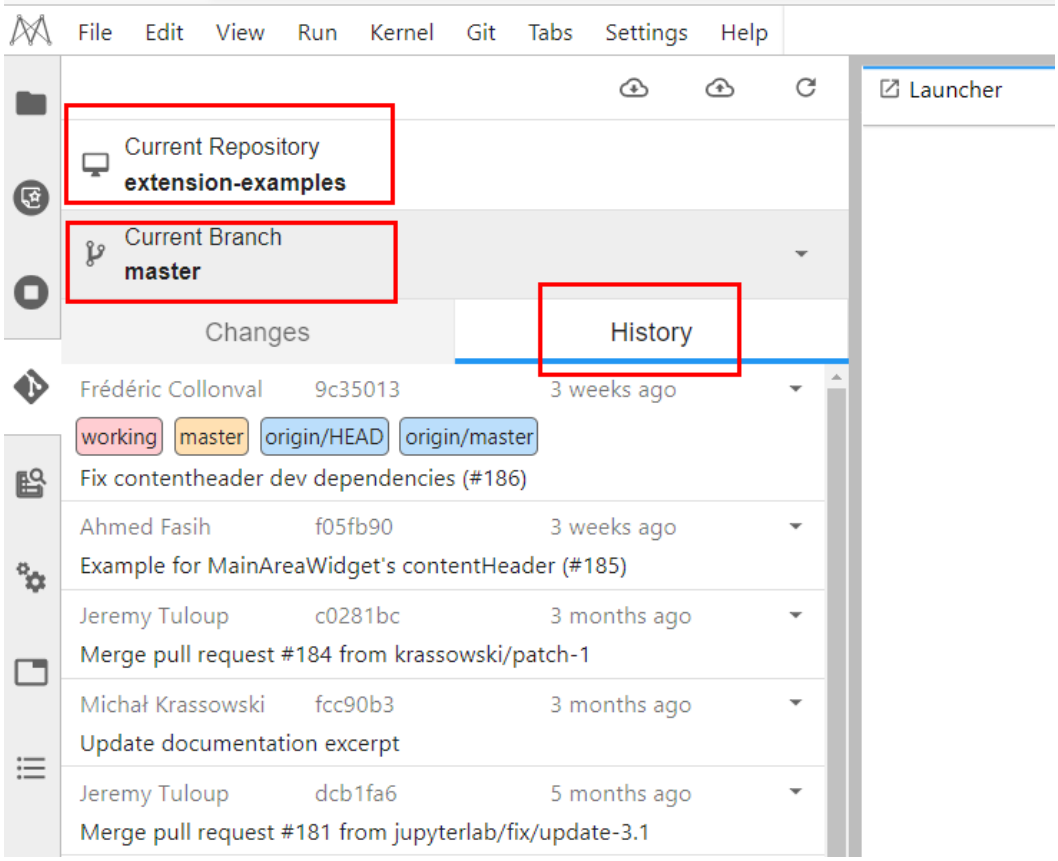
在Name下方列表中，选中您希望使用的文件夹，双击打开，然后单击左侧git插件图标进入此文件夹对应的代码库。

图 6-31 打开文件夹后打开 git 插件



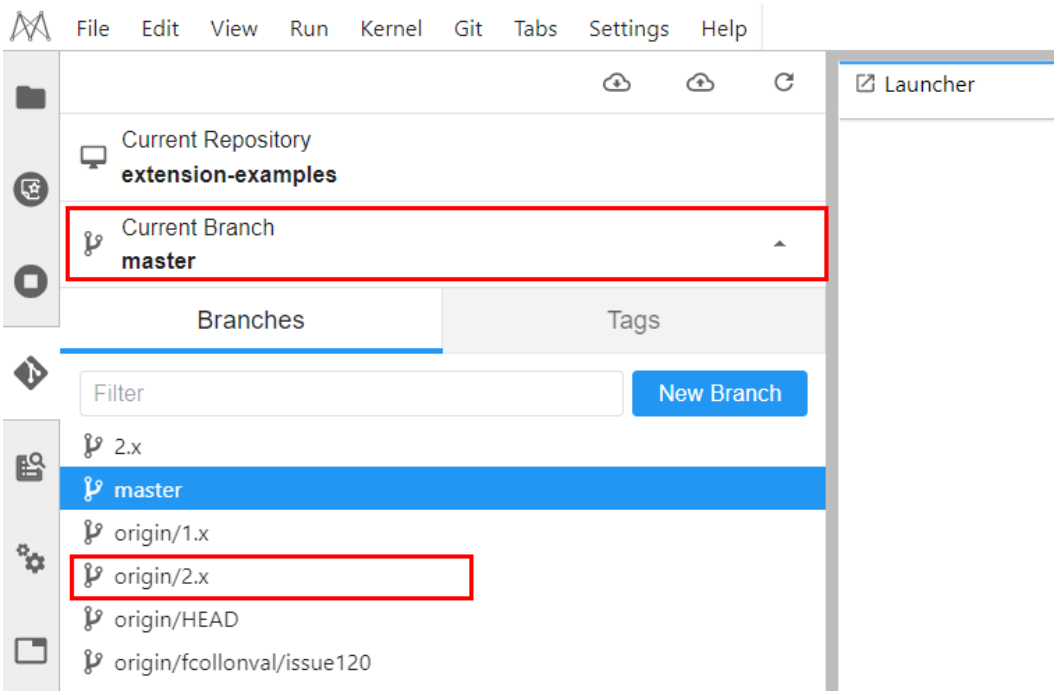
即可看到当前代码库的信息，如仓库名称、分支、历史提交记录等。

图 6-32 查看代码库信息



说明

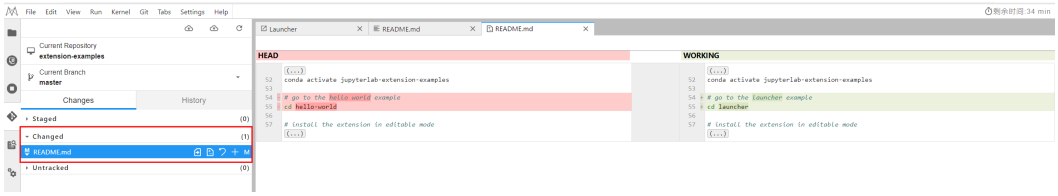
Git插件一般默认克隆master分支，如果要切换分支可单击Current Branch展开所有分支，单击相应分支名称可完成切换。



查看修改的内容

如果修改代码库中的某个文件，在“Changes”页签的“Changed”下可以看到修改的文件，并单击修改文件名称右侧的“Diff this file”，可以看到修改的内容。

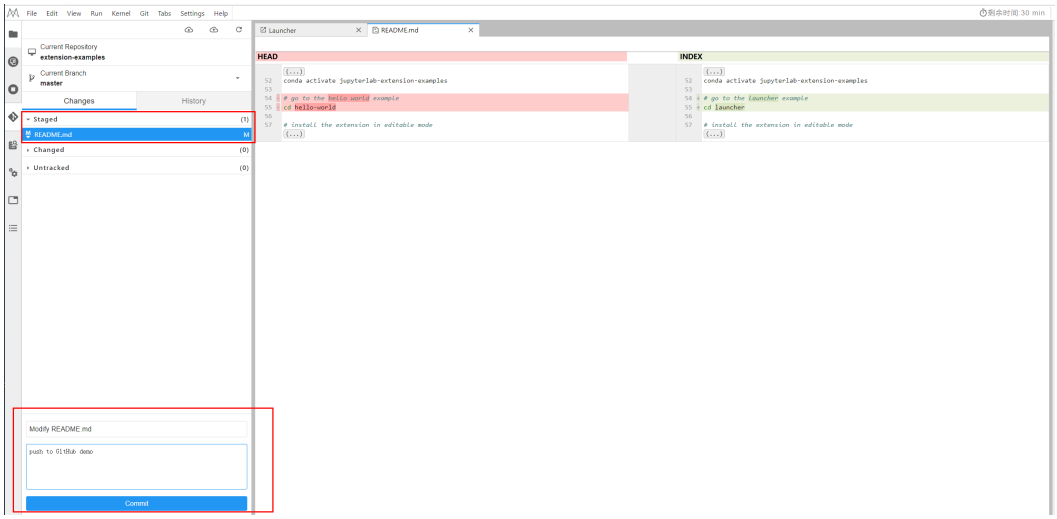
图 6-33 查看修改的内容



提交修改的内容

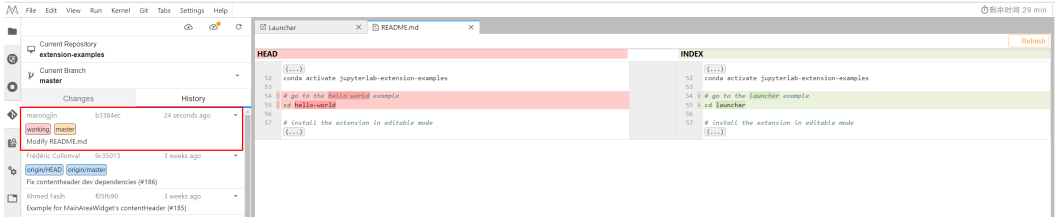
确认修改无误后，单击修改文件名称右侧的“Stage this change”，文件将进入Staged状态，相当于执行了git add命令。在左下方输入本次提交的Message，单击“Commit”，相当于执行了git commit命令。

图 6-34 提交修改内容



此时，可以在“History”页签下看到本地提交已成功。

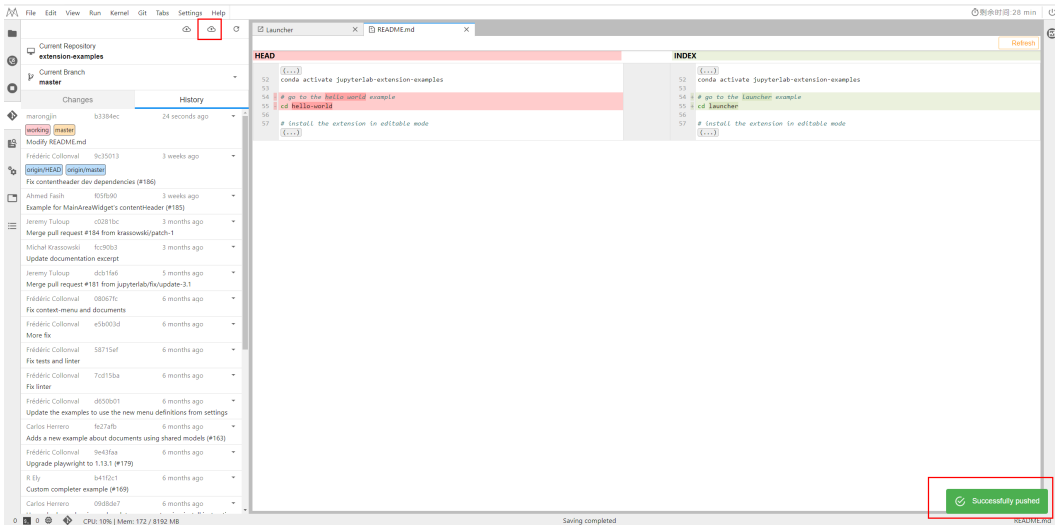
图 6-35 查看是否提交成功



单击“push”按钮，相当于执行git push命令，即可提交代码到GitHub仓库中。提交成功后会提示“Successfully completed ”。若OAuth鉴权的token过期，则此时再push会弹框让输入用户的token或者账户信息，按照提示输入即可。这里推荐使用

Personal Access Token授权方式，如果出现密码失效报错请参考**git插件密码失效如何解决?**

图 6-36 提交代码至 GitHub 仓库



完成上述操作后，可以在JupyterLab的git插件页面的History页签，看到“origin/HEAD”和“orgin/master”已指向最新一次的提交。同时在GitHub对应仓库的commit记录中也可以查找到对应的信息。

6.4.6 可视化训练作业

6.4.6.1 可视化训练作业介绍

ModelArts支持在新版开发环境中开启MindInsight可视化工具。在开发环境中通过小数据集训练调试算法，主要目的是验证算法收敛性、检查是否有训练过程中的问题，方便用户调测。

ModelArts可视化作业支持创建MindInsight类型。

MindInsight能够有效地展示训练作业在运行过程中的变化趋势以及训练中使用到的数据信息。

- MindInsight

  MindInsight能可视化展现出训练过程中的标量、图像、计算图以及模型超参等信息，同时提供训练看板、模型溯源、数据溯源、性能调试等功能，帮助您在更高效地训练调试模型。MindInsight当前支持基于MindSpore引擎的训练作业。MindInsight相关概念请参考**MindSpore官网**。

  MindInsight可视化训练作业，当前支持的镜像如下，请根据实际局点支持的镜像和资源规格选择使用。

  - mindspore1.2.0版本，CPU/GPU规格的资源类型。
  - mindspore1.5.x以上版本，Ascend规格的资源类型。

您可以使用模型训练时产生的Summary文件在开发环境Notebook中创建可视化作业。

- 在开发环境中创建TensorBoard可视化作业，请参见**TensorBoard可视化作业**。

6.4.6.2 MindInsight 可视化作业

ModelArts支持在新版开发环境中开启MindInsight可视化工具。在开发环境中通过小数据集训练调试算法，主要目的是验证算法收敛性、检查是否有训练过程中的问题，方便用户调测。

MindInsight能可视化展现出训练过程中的标量、图像、计算图以及模型超参等信息，同时提供训练看板、模型溯源、数据溯源、性能调试等功能，帮助您在更高效地训练调试模型。MindInsight当前支持基于MindSpore引擎的训练作业。MindInsight相关概念请参考**MindSpore官网**。

MindSpore支持将数据信息保存到Summary日志文件中，并通过可视化界面MindInsight进行展示。

前提条件

使用MindSpore引擎编写训练脚本时，为了保证训练结果中输出Summary文件，您需要在脚本中添加收集Summary相关代码。

将数据记录到Summary日志文件中的具体方式请参考**收集Summary数据**。

注意事项

- 在开发环境跑训练任务，在开发环境使用MindInsight，要求先启动MindInsight，后启动训练进程。
- 仅支持单机单卡训练。

在开发环境中创建 MindInsight 可视化作业流程

Step1 创建开发环境并在线打开

Step2 上传Summary数据

Step3 启动MindInsight

Step4 查看训练看板中的可视化数据

Step1 创建开发环境并在线打开

在ModelArts控制台，进入“开发环境> Notebook”页面，创建MindSpore引擎的开发环境实例。创建成功后，单击开发环境实例操作栏右侧的“打开 ”，在线打开运行中的开发环境。

MindInsight可视化训练作业，当前支持的镜像如下，请根据实际局点支持的镜像和资源规格选择使用。

- mindspore1.2.0版本，CPU/GPU规格的资源类型。
- mindspore1.5.x以上版本，Ascend规格的资源类型。

Step2 上传 Summary 数据

在开发环境中使用MindInsight可视化功能，需要用到Summary数据。

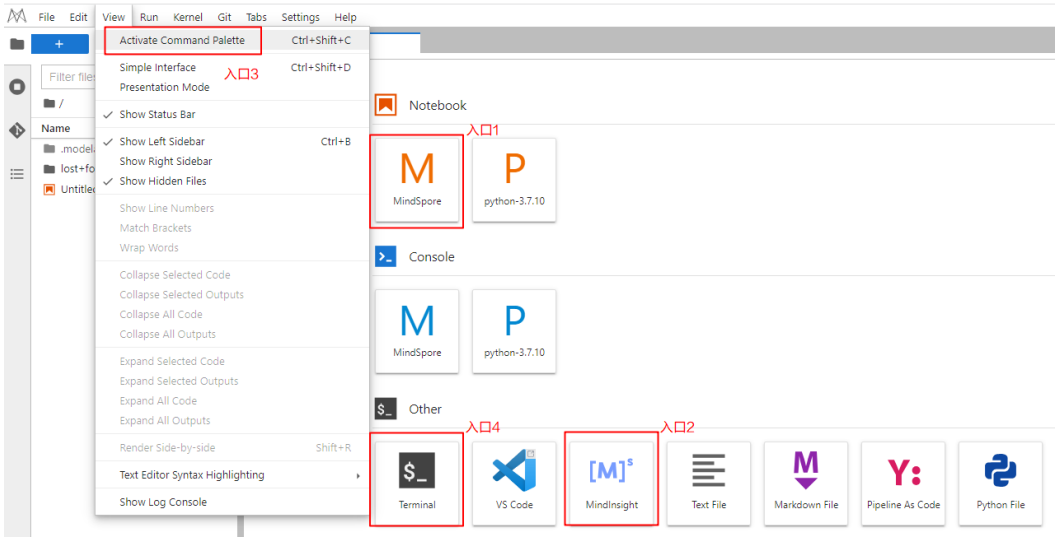
Summary数据可以直接传到开发环境的这个路径下/home/ma-user/work/，也可以放到OBS并行文件系统中。

- Summary数据上传到Notebook路径/home/ma-user/work/下的方式，请参见**上传数据至Notebook**。
- Summary数据如果是通过OBS并行文件系统挂载到Notebook中，请将模型训练时产生的Summary文件先上传到OBS并行文件系统。在Notebook中启动MindInsight时，Notebook会自动从挂载的OBS并行文件系统目录中读取Summary数据。

Step3 启动 MindInsight

在开发环境的JupyterLab中打开MindInsight有多种方法。可根据使用习惯选择。

图 6-37 JupyterLab 中打开 MindInsight 的方法



方式1：

1. 单击入口1 

，进入JupyterLab开发环境，并自动创建“.ipynb”文件。
2. 在对话框中输入MindInsight相应命令，即可展示界面。

```
%reload_ext mindinsight
%mindinsight --port {PORT} --summary-base-dir {SUMMARY_BASE_DIR}
```

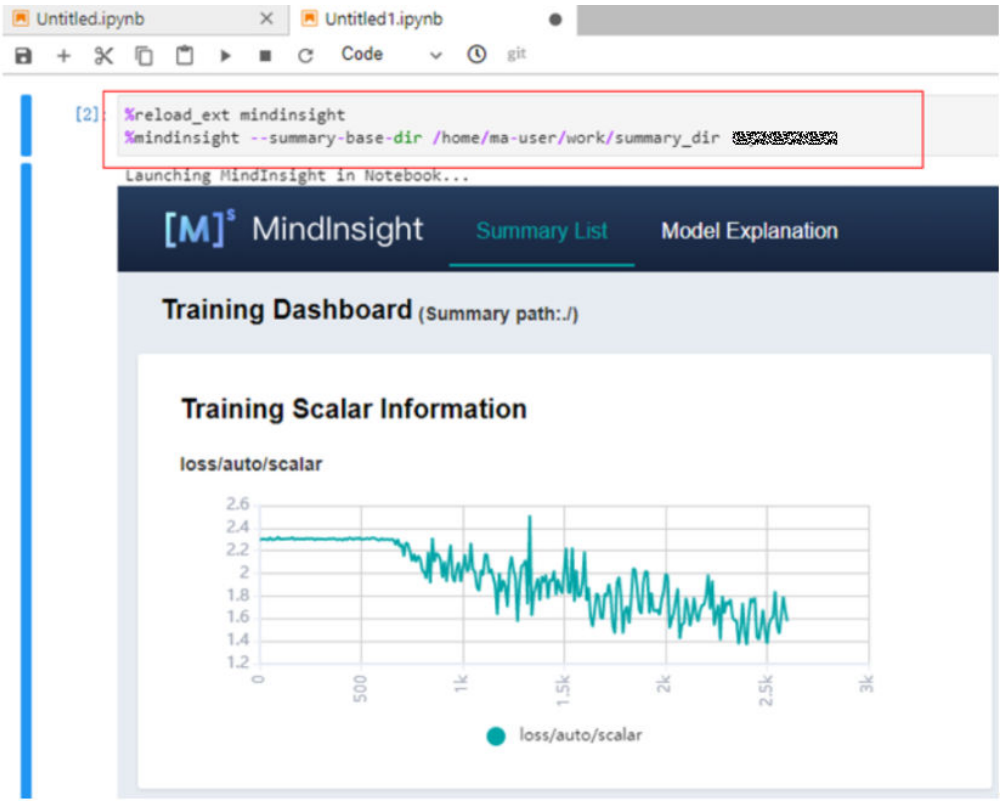
参数解释：

- --port {PORT}：指定Web可视化服务端口。可以不设置，默认使用8080端口。如果8080端口被占用了，需要在1~65535任意指定一个端口。
- --summary-base-dir {SUMMARY_BASE_DIR}：表示数据在开发环境中的存储路径。
  - 开发环境本地路径：“./work/xxx”（相对路径）或/home/ma-user/work/xxx（绝对路径）

- OBS并行文件系统桶的路径：obs://xxx/

```
例如：
#Summary数据如果是在开发环境的这个路径下/home/ma-user/work/，执行下面这条命令
%mindinsight --summary-base-dir /home/ma-user/work/xxx
或者
#Summary数据如果是存在OBS并行文件系统中，执行下面这条命令，开发环境会自动挂载该OBS并行文件系统路径并读取数据
%mindinsight --summary-base-dir obs://xxx/
```

图 6-38 MindInsight 界面（1）



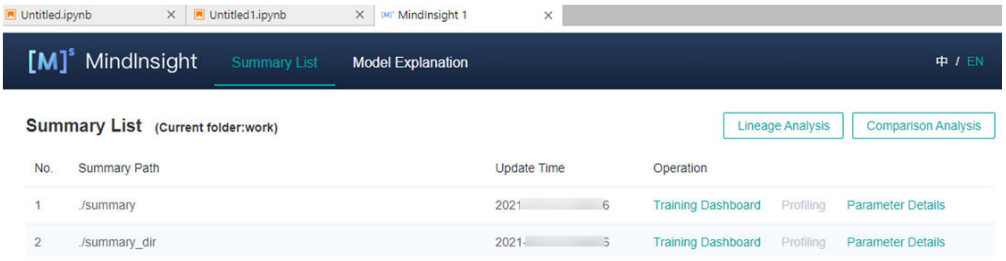
方式2：

单击入口2，直接进入MindInsight可视化界面。

默认读取路径/home/ma-user/work/

当存在两个以及以上工程的log时，界面如下。通过Runs下选择查看相对应的log。

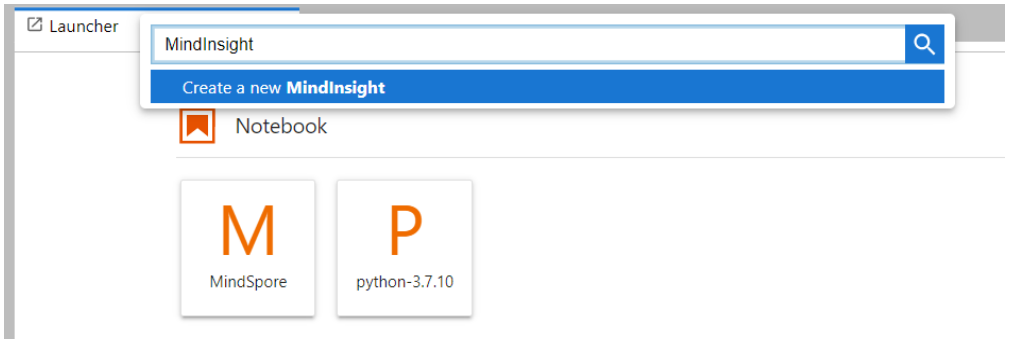
图 6-39 MindInsight 界面（2）



方式3：

1. 单击入口3 View -> Activate Command Palette，在搜索框中输入“MindInsight”，再单击“Create a new MindInsight”。

   图 6-40 Create a new MindInsight

2. 填写需要查看的Summary数据路径，或者OBS并行文件系统桶的路径，单击CREATE。
   - 开发环境本地路径：./summary（相对路径）或/home/ma-user/work/summary（绝对路径）
   - OBS并行文件系统的路径：obs://xxx/

   图 6-41 输入 Summary 数据路径

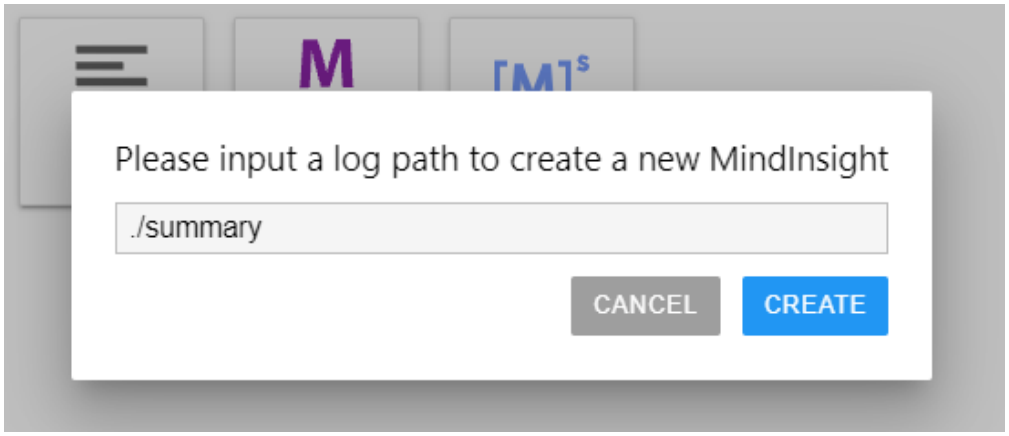
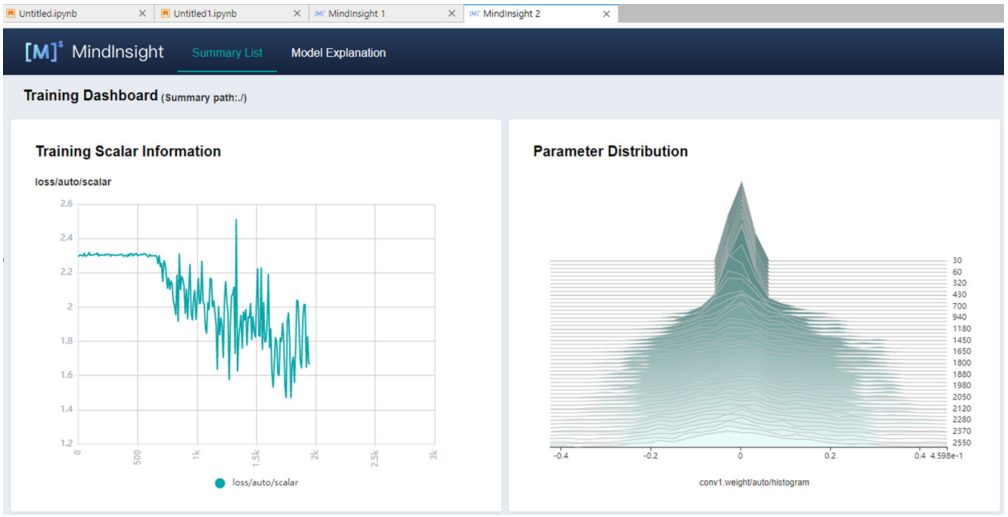
   

图 6-42 MindInsight 界面（3）



说明

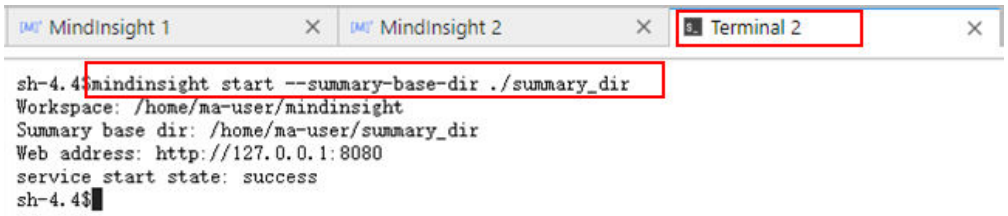
方式2及方式3最多可以启动10个MindInsight实例。

方式4：

单击入口4 Terminal ，输入并执行以下命令，但启动后不能显示UI界面。

```
mindinsight start --summary-base-dir ./summary_dir
```

图 6-43 Terminal 方式打开 MindInsight



Step4 查看训练看板中的可视化数据

训练看板是MindInsight的可视化组件的重要组成部分，而训练看板的标签包含：标量可视化、参数分布图可视化、计算图可视化、数据图可视化、图像可视化和张量可视化等。

更多功能介绍请参见MindSpore官网资料：**查看训练看板中可视的数据**。

相关操作

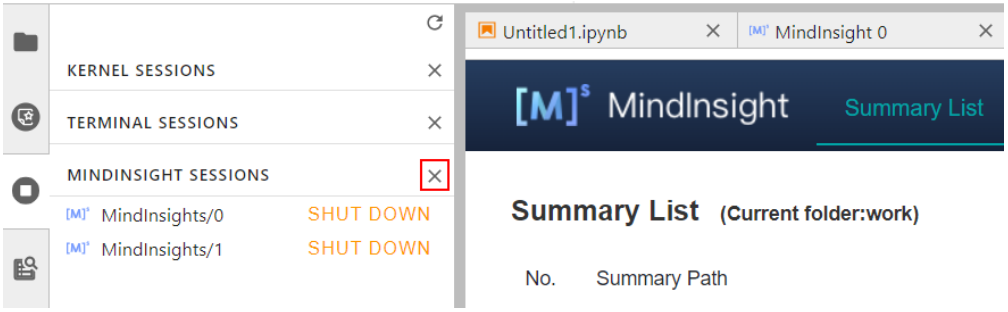
关闭MindInsight方式如下：

- 方式1：在开发环境JupyterLab中的“.ipynb”文件窗口中输入命令，关闭MindInsight。端口号在**启动MindInsight**中设置，默认使用8080，需要替换为实际开启MindInsight时的端口。
```
!mindinsight stop --port 8080
```
- 方式2：单击下方  按钮进入MindInsight实例管理界面，该界面记录了所有启动的MindInsight实例，单击对应实例后面的SHUT DOWN即可停止该实例。

图 6-44 单击 SHUT DOWN 停止实例

- 方式3：单击下方红框中的  按钮可以关闭所有启动的MindInsight实例。

图 6-45 关闭所有启动的 MindInsight 实例



- 方式4（不推荐）：直接在JupyterLab中上关闭MindInsight窗口，此方式仅是关闭MindInsight可视化窗口，并未关闭后台。

6.4.6.3 TensorBoard 可视化作业

ModelArts支持在开发环境中开启TensorBoard可视化工具。TensorBoard是TensorFlow的可视化工具包，提供机器学习实验所需的可视化功能和工具。

TensorBoard能够有效地展示TensorFlow在运行过程中的计算图、各种指标随着时间的变化趋势以及训练中使用到的数据信息。

前提条件

为了保证训练结果中输出Summary文件，在编写训练脚本时，您需要在脚本中添加收集Summary相关代码。

TensorFlow引擎的训练脚本中添加Summary代码，具体方式请参见**TensorFlow官方网站**。

在开发环境中创建 TensorBoard 可视化作业流程

Step1 创建开发环境并在线打开

在ModelArts控制台，进入“开发环境 > Notebook”页面，创建TensorFlow或者PyTorch镜像的开发环境实例。创建成功后，单击开发环境实例操作栏右侧的“打开”，在线打开运行中的开发环境。

TensorBoard可视化训练作业，当前仅支持基于TensorFlow2.1、Pytorch1.4/1.8以上版本镜像，CPU/GPU规格的资源类型。请根据实际局点支持的镜像和资源规格选择使用。

Step2 上传 Summary 数据

在开发环境中使用TensorBoard可视化功能，需要用到Summary数据。

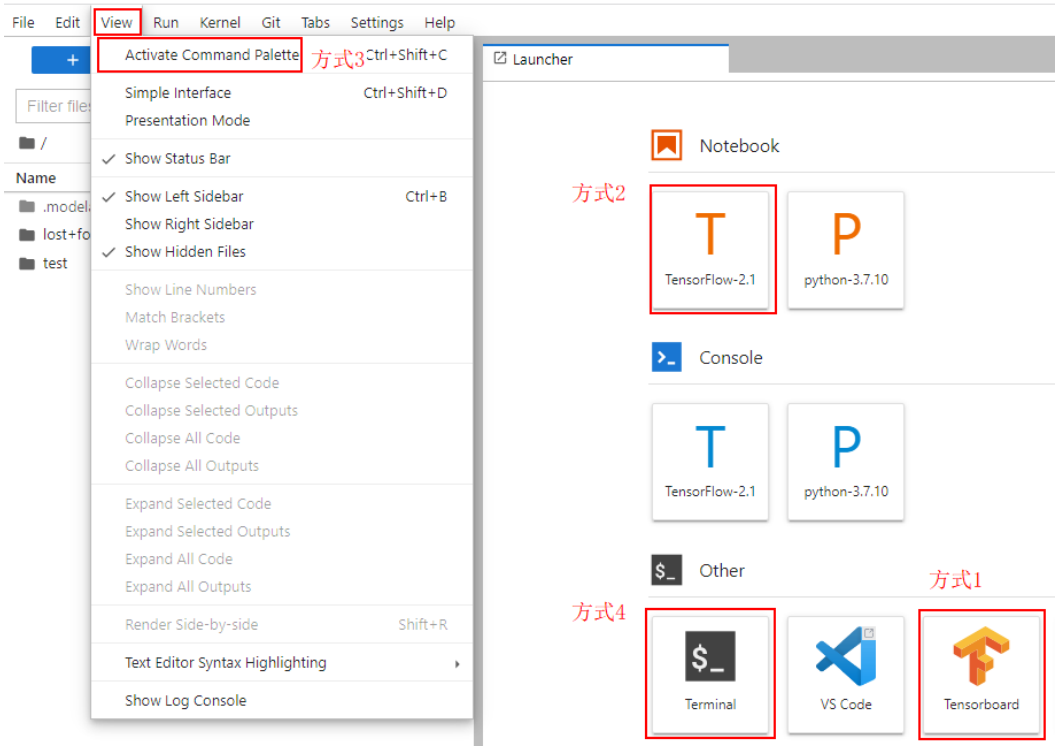
Summary数据可以直接传到开发环境的这个路径下/home/ma-user/work/，也可以放到OBS并行文件系统中。

- Summary数据上传到Notebook路径/home/ma-user/work/下的方式，请参见**上传数据至Notebook**。
- Summary数据如果是通过OBS并行文件系统挂载到Notebook中，请将模型训练时产生的Summary文件先上传到OBS并行文件系统。在Notebook中启动TensorBoard时，Notebook会自动从挂载的OBS并行文件系统目录中读取Summary数据。

Step3 启动 TensorBoard

在开发环境的JupyterLab中打开TensorBoard有多种方法。可根据使用习惯选择。

图 6-46 JupyterLab 中打开 TensorBoard 的方法

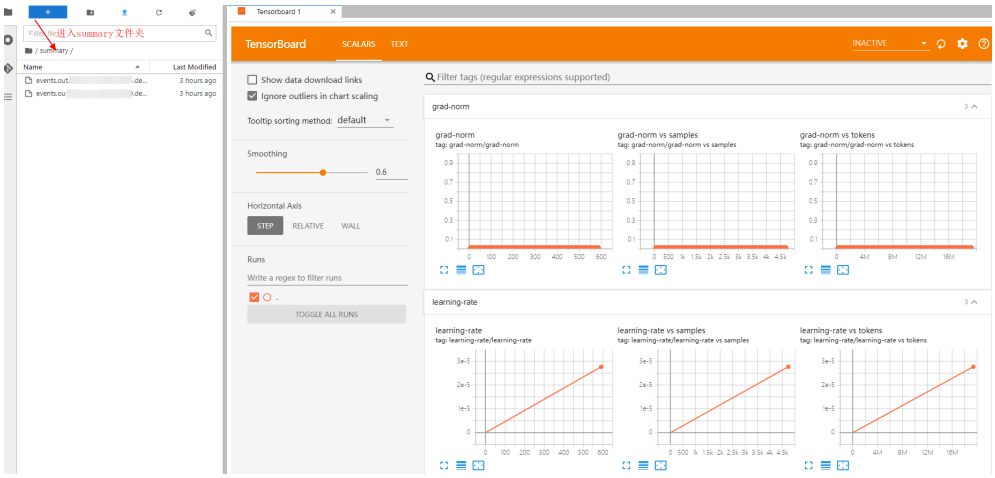


方式1（推荐）：

1. 在JupyterLab左侧导航创建名为“summary”的文件夹，将数据上传到“/home/ma-user/work/summary”路径。注：文件夹命名只能为summary否则无法使用。

2. 进入“summary”文件夹，单击方式1 ，直接进入TensorBoard可视化界面。如图6-47所示。

图 6-47 TensorBoard 界面（1）



方式2：

须知

用户可以自行升级除2.4.0之外的TensorBoard，但需注意升级后只有方式2使用新的TensorBoard，其余方式保持TensorBoard2.1.1不变。

1. 单击方式2，进入JupyterLab开发环境中，并自动创建“.ipynb”文件。
2. 在对话框中输入TensorBoard相应命令，即可展示界面。

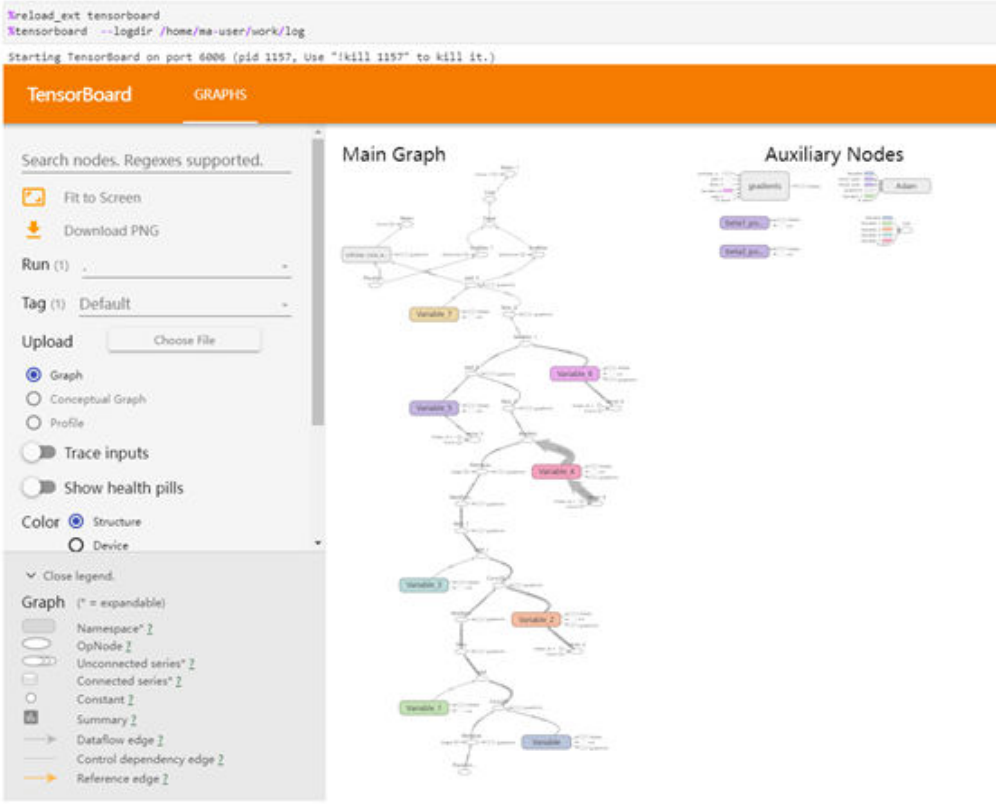
```
%reload_ext ma_tensorboard
%ma_tensorboard  --port {PORT} --logdir {BASE_DIR}
```

参数解释：

- --port {PORT}：指定Web可视化服务端口。可以不设置，默认使用8080端口。如果8080端口被占用了，需要在1~65535任意指定一个端口。
- --logdir {BASE_DIR}：表示数据在开发环境中的存储路径
  - 开发环境本地路径：“./work/xxx”（相对路径）或/home/ma-user/work/xxx（绝对路径）
  - OBS并行文件系统的路径：obs://xxx/

```
例如：
#Summary数据如果是在开发环境的这个路径下/home/ma-user/work/，执行下面这条命令。
%ma_tensorboard  --port {PORT} --logdir /home/ma-user/work/xxx
或者
#Summary数据如果是存在OBS并行文件系统中，执行下面这条命令，开发环境会自动挂载该OBS并行文件系统路径并读取数据。
%ma_tensorboard  --port {PORT} --logdir obs://xxx/
```

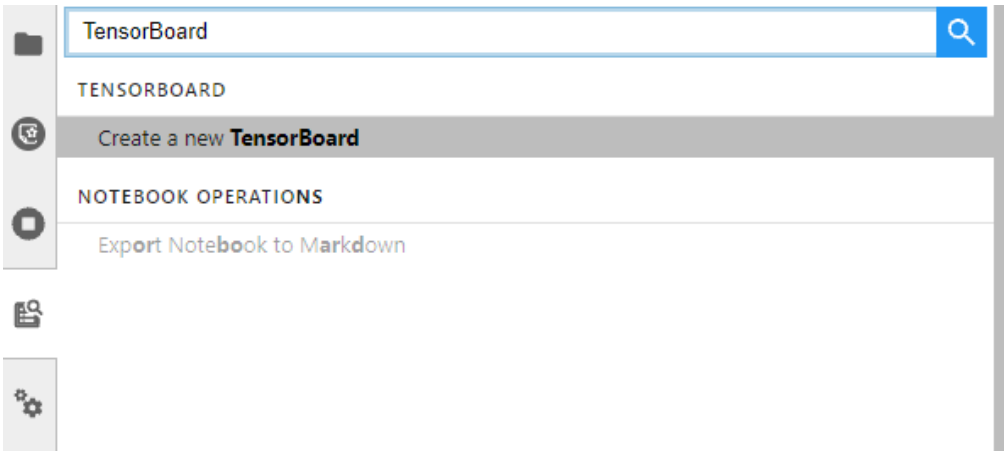
图 6-48 TensorBoard 界面（2）



方式3：

1. 单击方式3 View -> Activate Command Palette，在搜索框中输入“TensorBoard”，再单击“Create a new TensorBoard”。

   图 6-49 Create a new TensorBoard

2. 填写需要查看的Summary数据路径，或者OBS并行文件系统桶的路径。
   - 开发环境本地路径：./summary（相对路径）或/home/ma-user/work/summary（绝对路径）
   - OBS并行文件系统桶的路径：obs://xxx/

图 6-50 输入 Summary 数据路径

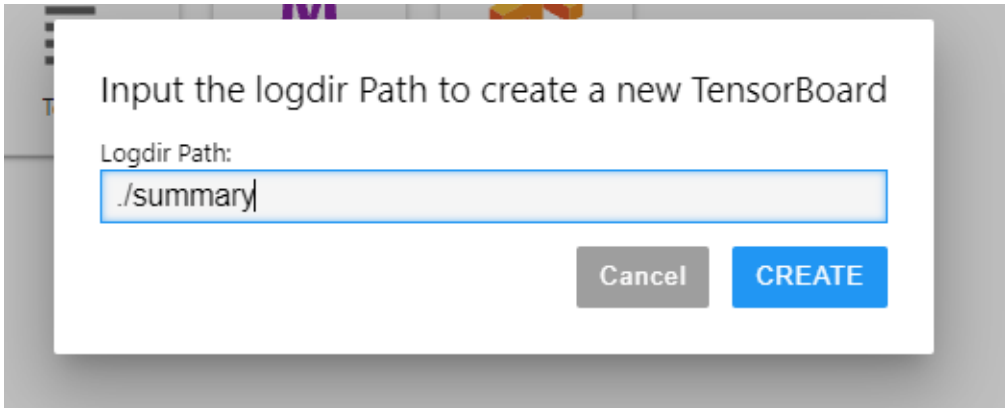
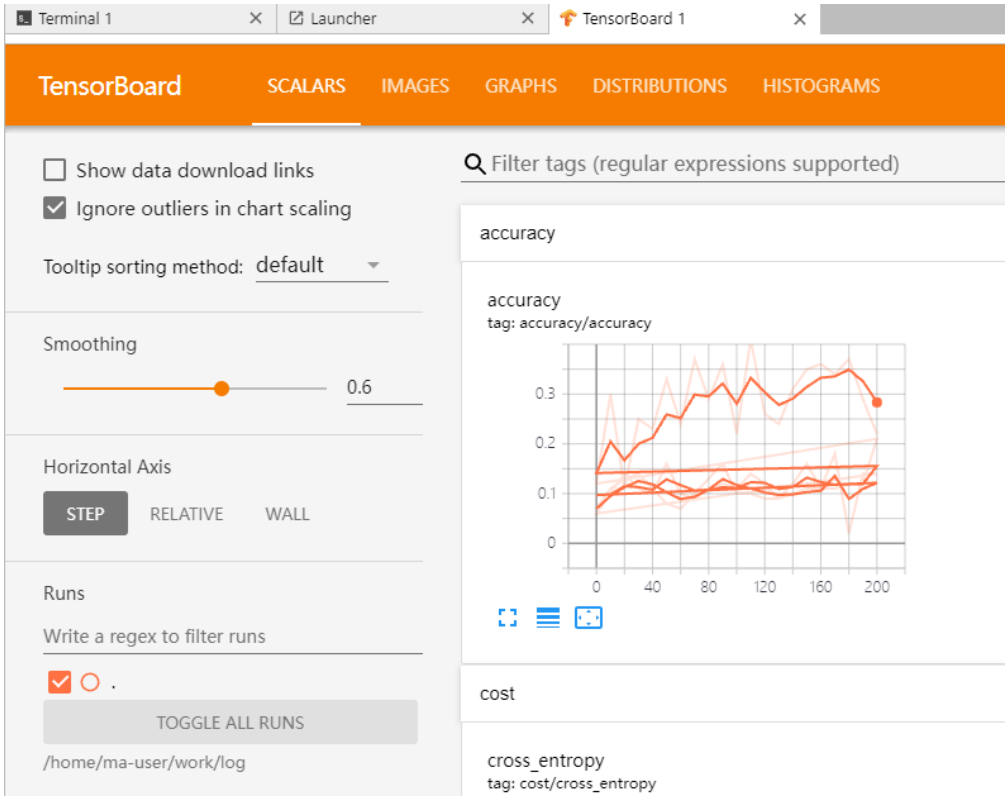


图 6-51 TensorBoard 界面（3）

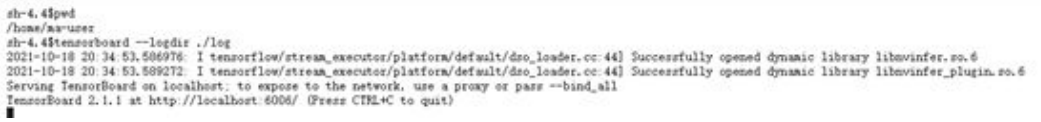


方式4：

单击方式4 ，输入命令执行，但启动后不能显示UI界面。

```
tensorboard --logdir ./log
```

图 6-52 Terminal 方式打开 TensorBoard



Step4 查看训练看板中的可视化数据

训练看板是TensorBoard的可视化组件的重要组成部分，而训练看板的标签包含：标量可视化、图像可视化和计算图可视化等。

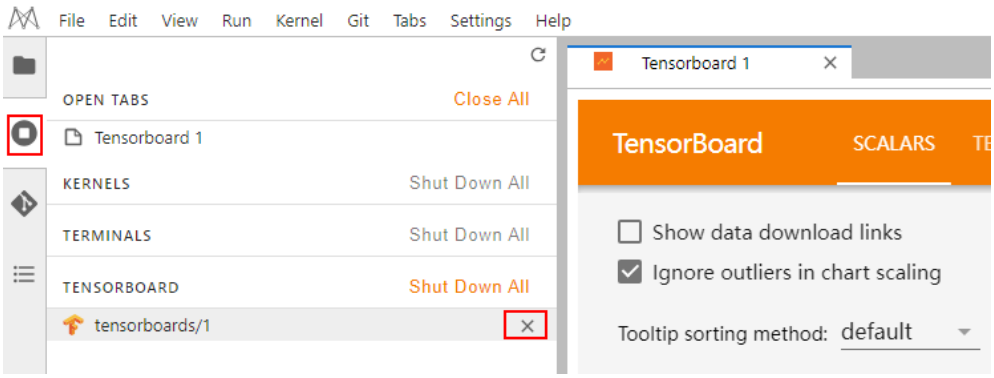
更多功能介绍请参见**TensorBoard官网资料**。

相关操作

关闭TensorBoard方式如下：

- 方式1：单击下图所示的  ，进入TensorBoard实例管理界面，该界面记录了所有启动的TensorBoard实例，单击对应实例后面的SHUT DOWN即可停止该实例。

  图 6-53 单击 SHUT DOWN 停该实例

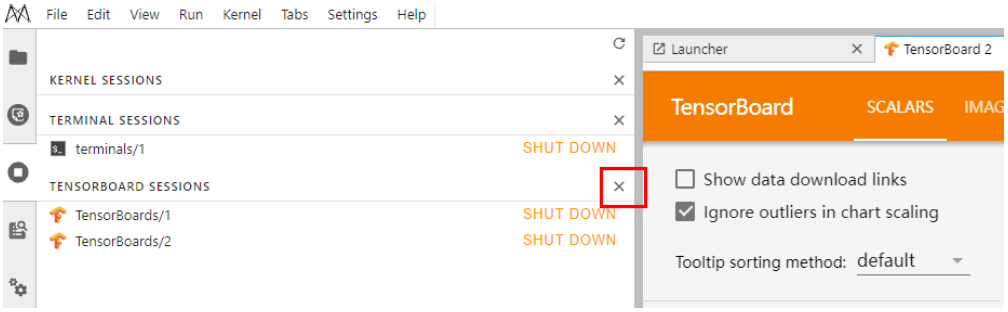
  

- 方式2：在开发环境JupyterLab中的“.ipynb”文件窗口中输入命令，关闭TensorBoard。PID在启动界面有提示或者通过ps -ef | grep tensorboard查看。

  ```
  !kill PID
  ```

- 方式3：单击下方红框中的  按钮可以关闭所有启动的TensorBoard实例。

  图 6-54 关闭所有启动的 TensorBoard 实例

- 方式4（不推荐）：直接在JupyterLab中上关闭TensorBoard窗口，此方式仅关闭可视化窗口，并未关闭后台。

6.4.7 Notebook 中的数据上传下载

6.4.7.1 上传文件至 JupyterLab

6.4.7.1.1 场景介绍

在AI开发过程中，如何将文件方便快速地上传到Notebook几乎是每个开发者都会遇到的问题。

ModelArts之前对文件直接上传到Notebook的大小限制是100MB，超过限制的文件无法直接上传；其次需要上传的文件并不都在本地，可能是GitHub的开源仓库，可能是类似开源数据集（**https://nodejs.org/dist/v12.4.0/node-v12.4.0-linux-x64.tar.xz**）这样的远端文件，也可能是存放在OBS中的文件，ModelArts之前无法将这些文件直接上传到Notebook中；在文件上传过程中，用户无法获得更多的信息，例如上传进度和速度。

ModelArts上传文件特性主要解决了以上三个问题，不仅提供了更多上传文件的功能满足用户需求，而且展示了更多文件上传的细节，提升了用户的体验。

当前的文件上传功能：

- 支持上传本地文件；
- 支持Clone GitHub开源仓库；
- 支持上传OBS文件；
- 支持上传远端文件；
- 将文件上传详情可视化。

6.4.7.1.2 上传本地文件至 JupyterLab

上传场景和入口介绍

Notebook的JupyterLab中提供了多种方式上传文件。

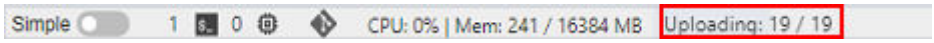
上传文件要求

- 对于大小不超过100MB的文件直接上传，并展示文件大小、上传进度及速度等详细信息。
- 对于大小超过100MB不超过5GB的文件可以使用OBS中转，系统先将文件上传OBS（对象桶或并行文件系统），然后从OBS下载到Notebook，上传完成后，会将文件从OBS中删除。
- 5GB以上的文件上传通过调用ModelArts SDK或者Moxing完成。
- 对于Notebook当前目录下已经有同文件名称的文件，可以覆盖继续上传，也可以取消。
- 支持10个文件同时上传，其余文件显示“等待上传”。不支持上传文件夹，可以将文件夹压缩成压缩包上传至Notebook后， 在Terminal中解压压缩包。

  ```
  unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
  ```

  解压命令的更多使用说明可以在主流搜索引擎中查找Linux解压命令操作。

- 多个文件同时上传时，JupyterLab窗口最下面会显示上传文件总数和已上传文件数。

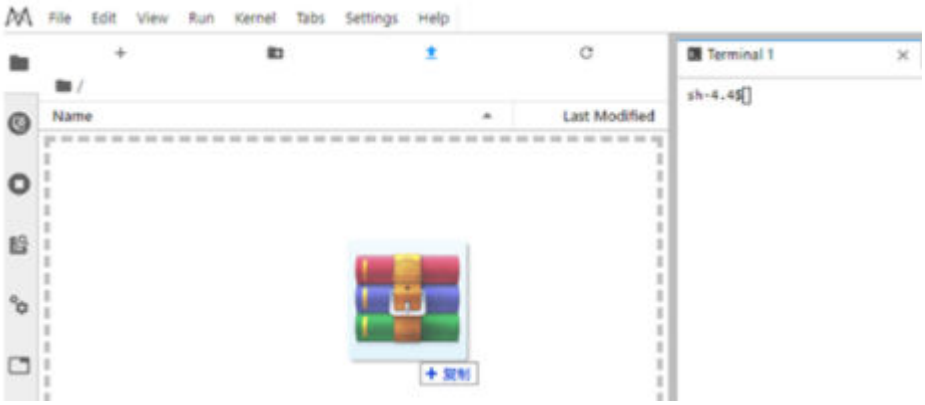


前提条件

使用JupyterLab打开一个运行中的Notebook环境。

上传文件入口 1：将文件直接拖拽至浏览器窗口中

直接将文件拖拽到JupyterLab窗口左边的空白处上传。



上传文件入口 2：通过上传图标传文件

单击窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传界面，拖拽或者选择本地文件上传。

图 6-55 上传文件图标

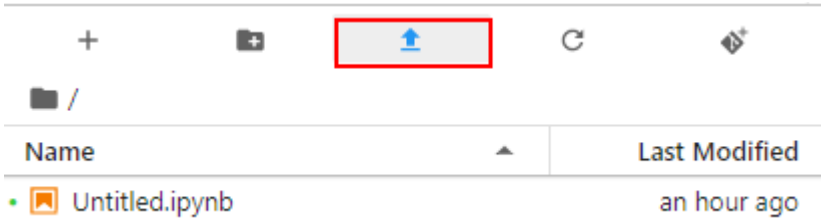
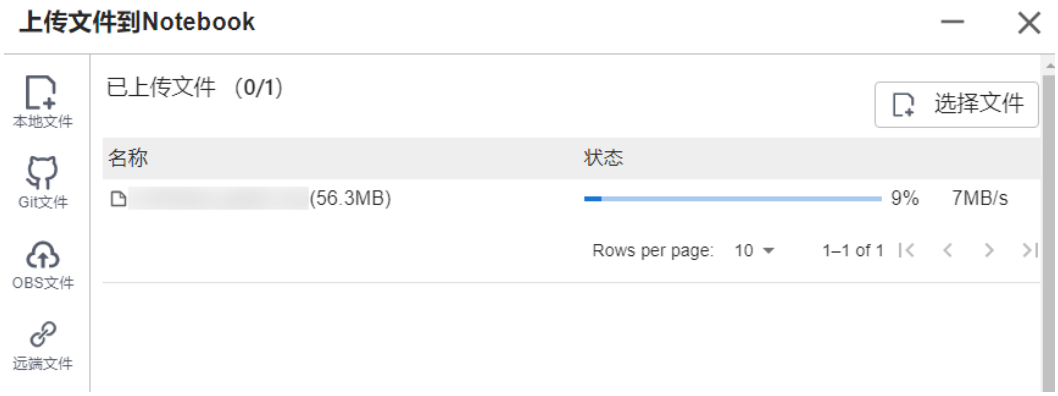


图 6-56 上传文件窗口

上传本地小文件（100MB 以内）至 JupyterLab

对于大小不超过100MB的文件直接上传，并展示文件大小、上传进度及速度等详细信息。

图 6-57 上传 100MB 以下小文件



文件上传完成后给出提示。

图 6-58 上传成功

上传本地大文件（100MB~5GB）至 JupyterLab

对于大小超过100MB不超过5GB的文件可以使用OBS中转，系统先将文件上传至OBS（对象桶或并行文件系统），然后从OBS下载到Notebook。下载完成后，ModelArts会将文件自动从OBS中删除。

例如，对于下面这种情况，可以通过“OBS中转”上传。

图 6-59 通过 OBS 中转上传大文件

若使用OBS中转需要提供一个OBS中转路径，可以通过以下三种方式提供：

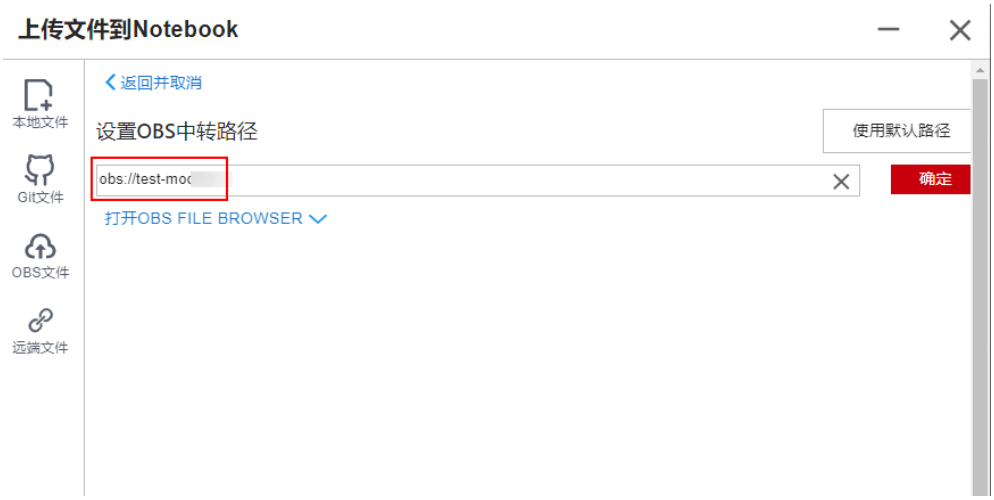
图 6-60 通过 OBS 中转路径上传

说明

仅第一次单击“OBS中转”需要提供OBS中转路径，以后默认使用该路径直接上传，可以通过上传文件窗口左下角的设置按钮 更新OBS中转路径。如图6-64所示。

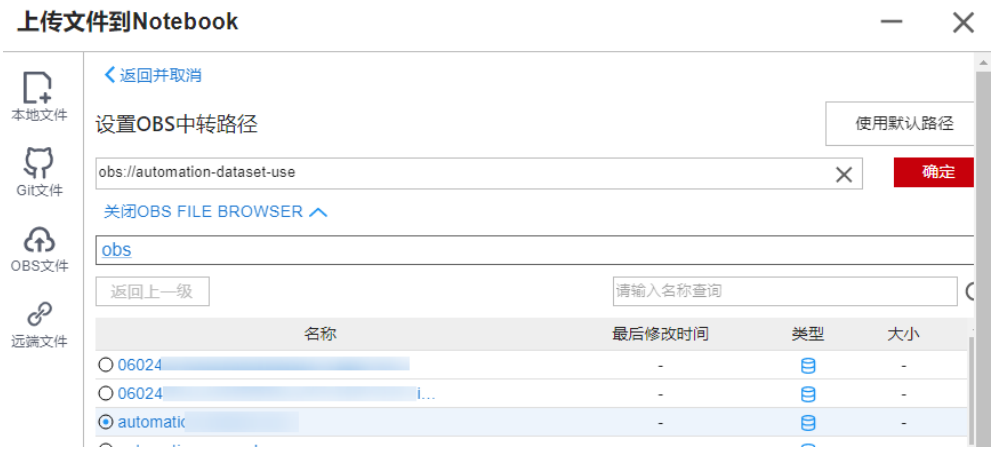
- 方式一：在输入框中直接输入有效的OBS中转路径，然后单击“确定”完成。

图 6-61 输入有效的 OBS 中转路径



- 方式二：打开OBS File Browser选择一个OBS中转路径，然后单击“确定”完成。

图 6-62 打开 OBS File Browser



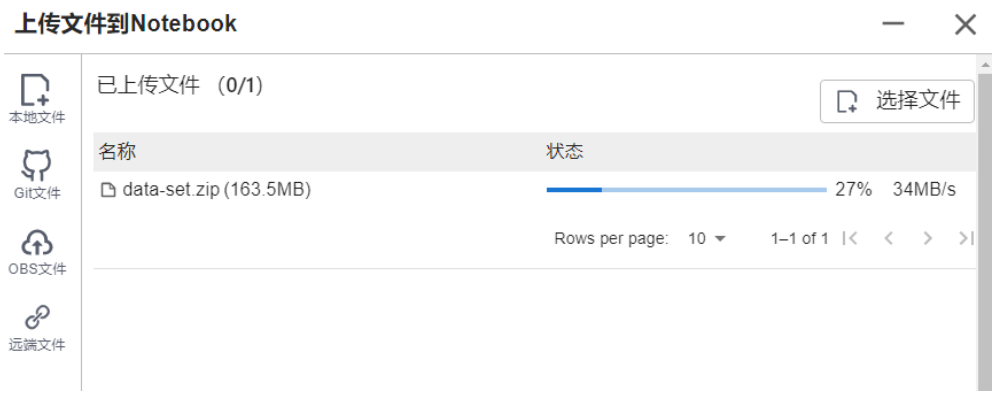
- 方式三：单击“使用默认路径”完成。

图 6-63 使用默认路径上传文件

图 6-64 设置本地文件 OBS 中转路径

完成OBS中转路径设置后，开始上传文件。

图 6-65 上传文件



解压缩文件包

将文件以压缩包形式上传至Notebook JupyterLab后， 可在Terminal中解压缩文件包。

```
unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
```

解压命令的更多使用说明可以在主流搜索引擎中查找Linux解压命令操作。

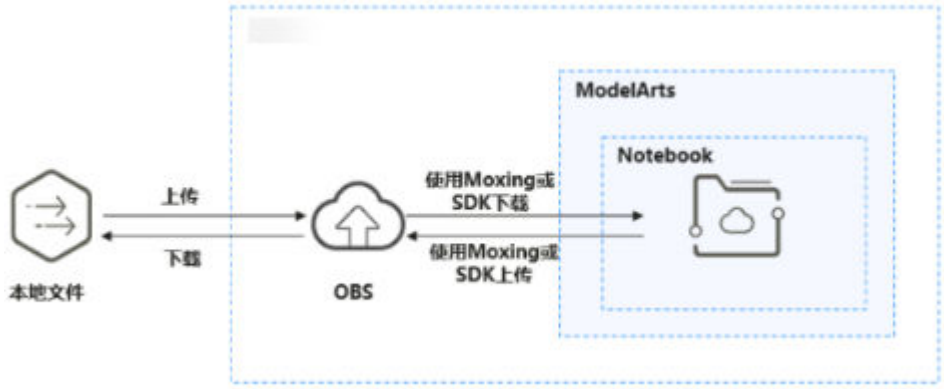
上传本地超大文件（5GB 以上）至 JupyterLab

不支持在Notebook的JupyterLab中直接上传大小超过5GB的文件。

图 6-66 不支持直接上传大小超过 5GB 的文件

5GB以上的文件需要先从本地上传到OBS中，再在Notebook中调用ModelArts的Moxing接口或者SDK接口读写OBS中的文件。

图 6-67 在 Notebook 中上传下载大文件



具体操作如下：

1. 从本地上传文件至OBS。
2. 将OBS中的文件下载到Notebook，可以通过在Notebook中运行代码的方式完成数据下载，具体方式有2种，ModelArts的SDK接口或者调用MoXing接口。
   - 方法一：使用ModelArts SDK接口将OBS中的文件下载到Notebook后进行操作。

     示例代码：

     ```
     from modelarts.session import Session
     session = Session()
     session.obs.copy("obs://bucket-name/obs_file.txt","/home/ma-user/work/")
     ```

- 方法二：使用如下Moxing接口将OBS中的文件同步到Notebook后进行操作。

```
import moxing as mox

#下载一个OBS文件夹sub_dir_0，从OBS下载至Notebook
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/home/ma-user/work/sub_dir_0')
#下载一个OBS文件obs_file.txt，从OBS下载至Notebook
mox.file.copy('obs://bucket_name/obs_file.txt', '/home/ma-user/work/obs_file.txt')
```

如果下载到Notebook中的是zip文件，在Terminal中执行下列命令，解压压缩包。

```
unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
```

代码执行完成后，参考图6-68打开Terminal后执行**ls /home/ma-user/work**命令查看下载到Notebook中的文件。或者在Jupyter左侧导航中显示下载的文件，如果没有显示，请刷新后查看，如图6-69所示。

图 6-68 打开 Terminal

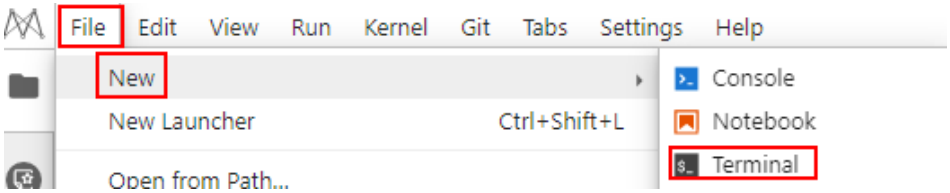
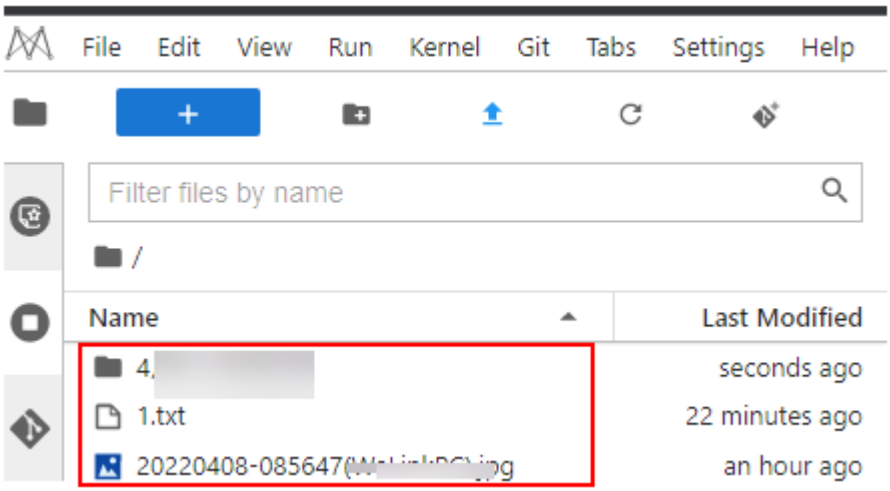


图 6-69 查看下载到 Notebook 中的文件



异常处理

通过OBS下载文件到Notebook中时，提示Permission denied。请依次排查：

- 请确保读取的OBS桶和Notebook处于同一站点区域。不支持跨站点访问OBS桶。
- 请确认操作Notebook的账号有权限读取OBS桶中的数据。

6.4.7.1.3 GitHub 开源仓库 Clone

在Notebook的JupyterLab中，支持从GitHub开源仓库Clone文件。

1. 通过JupyterLab打开一个运行中的Notebook。

2. 单击JupyterLab窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传窗口，选择左侧的 Git文件 进入GitHub开源仓库Clone界面。

图 6-70 上传文件图标

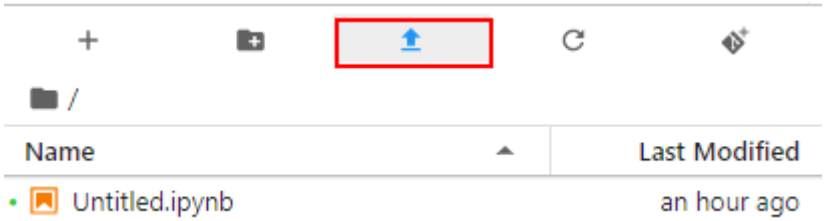
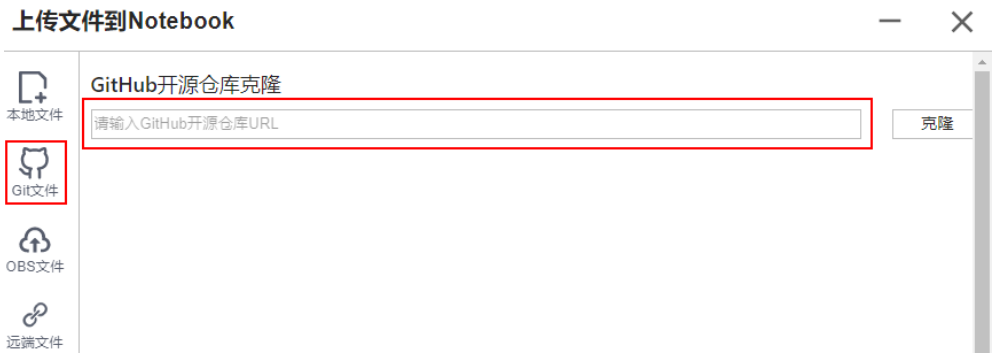


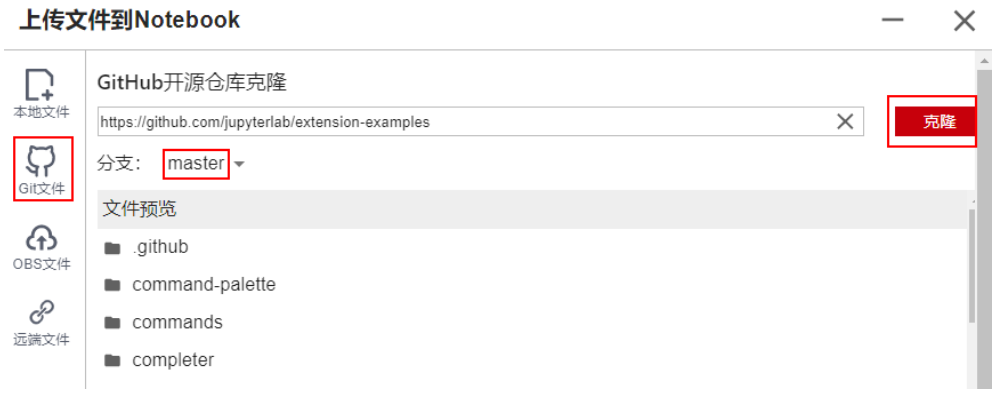
图 6-71 进入 GitHub 开源仓库 Clone 界面



3. 输入有效的GitHub开源仓库地址后会展示该仓库下的文件及文件夹，说明用户输入了有效的仓库地址，同时给出该仓库下所有的分支供选择，选择完成后单击“克隆”开始Clone仓库。

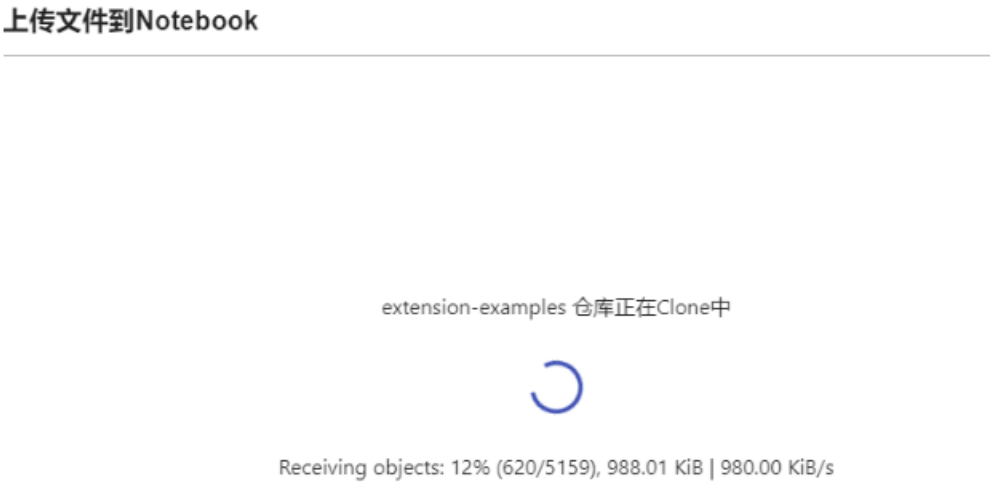
   GitHub开源仓库地址：https://github.com/jupyterlab/extension-examples

图 6-72 输入有效的 GitHub 开源仓库地址



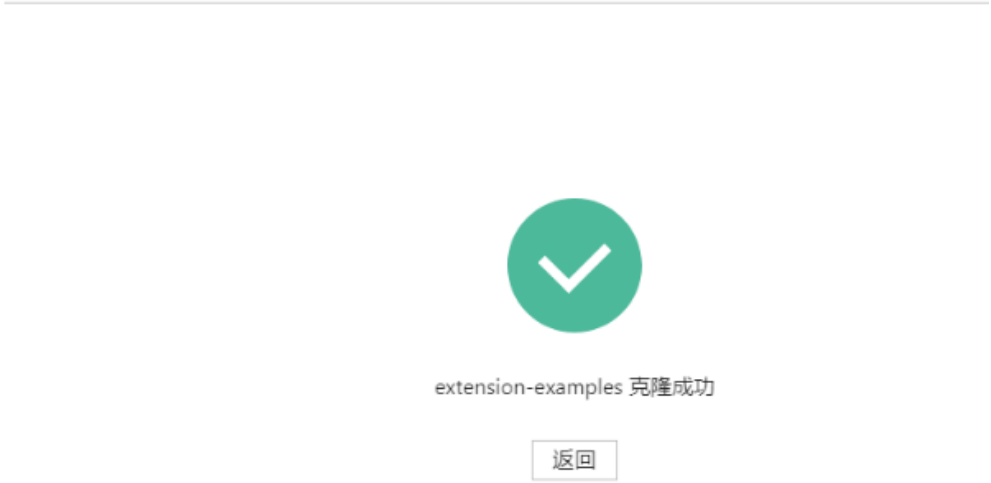
4. Clone仓库的过程中会将进度展示出来。

图 6-73 Clone 仓库的过程



5. Clone仓库成功。

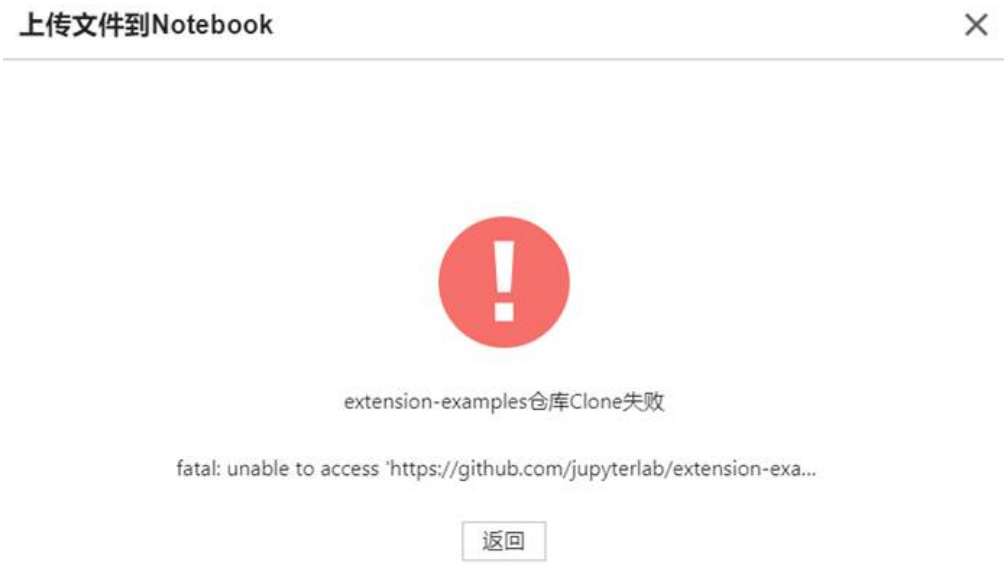
图 6-74 Clone 仓库成功



异常处理

- Clone仓库失败。可能是网络原因问题。可以在JupyterLab的Terminal中通过执行git clone https://github.com/jupyterlab/extension-examples.git测试网络连通情况。

图 6-75 Clone 仓库失败



- 若克隆时遇到Notebook当前目录下已有该仓库，系统给出提示仓库名称重复，此时可以单击“覆盖”继续克隆仓库，也可以单击 取消。

6.4.7.1.4 上传 OBS 文件到 JupyterLab

在Notebook的JupyterLab中，支持将OBS中的文件下载到Notebook。注意：文件大小不能超过10GB，否则会上传失败。

1. 通过JupyterLab打开一个运行中的Notebook。
2. 单击JupyterLab窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传窗口，选择左侧的 进入OBS文件上传界面。

图 6-76 上传文件图标

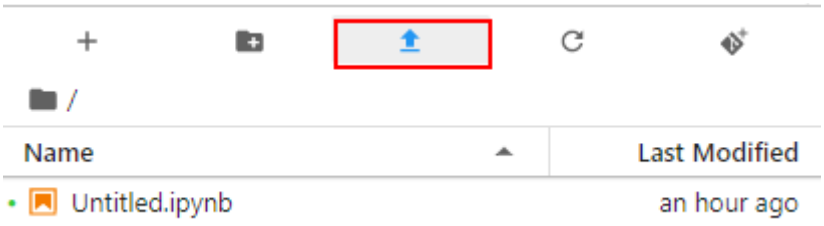
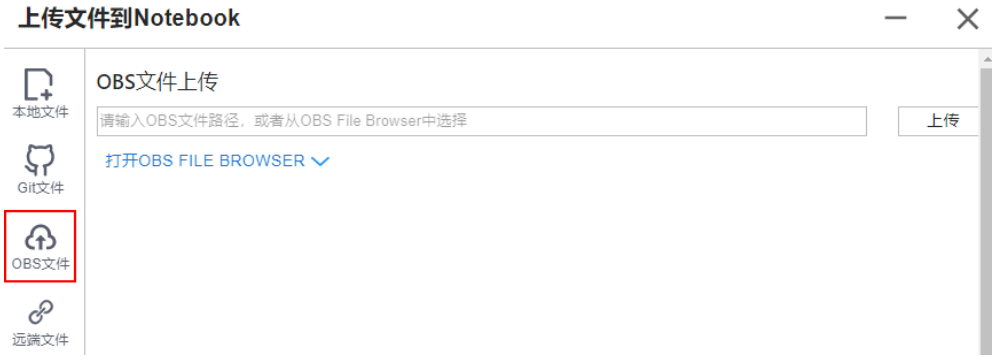
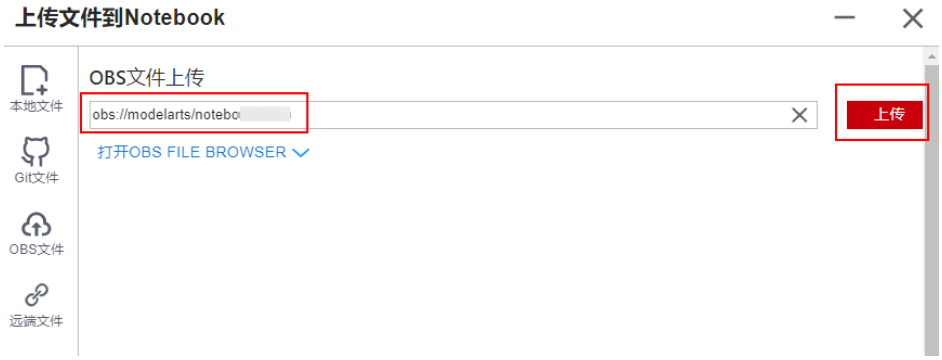


图 6-77 OBS 文件上传界面



3. 需要提供OBS文件路径，可以通过以下两种方式提供：
   - 方式一：在输入框中直接输入有效的OBS文件路径，然后单击“上传”开始传文件。

     图 6-78 输入有效的 OBS 文件路径

     说明

     此处输入的是具体的OBS文件路径，不是文件夹的路径，否则会导致上传失败。

   - 方式二：打开OBS File Browser选择OBS文件路径，然后单击“上传”，开始上传文件。

     图 6-79 上传 OBS 文件

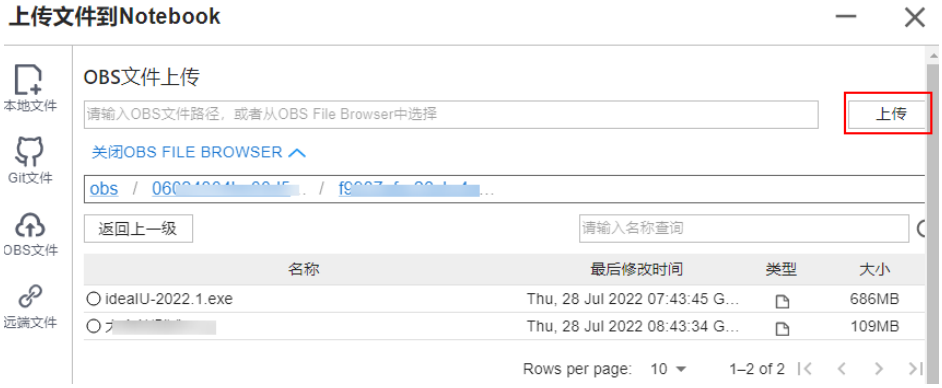
     

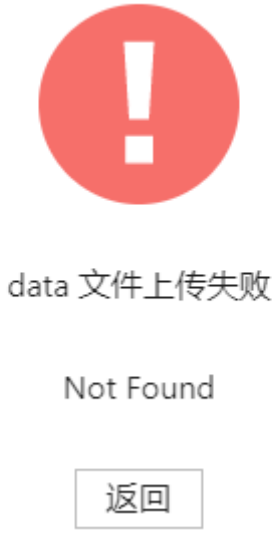
图 6-80 文件上传成功

异常处理

提示文件上传失败，有以下三种常见场景。

- **异常场景1**

  图 6-81 文件上传失败

  可能原因：

  - OBS路径没有设置为具体的文件路径，设置成了文件夹。
  - OBS中的文件设置了加密。请前往OBS控制台查看，确保该文件未加密。
  - OBS桶和Notebook不在同一个区域。请确保读取的OBS桶和Notebook处于同一站点区域，不支持跨站点访问OBS桶。
  - 没有该OBS桶的访问权限。请确认操作Notebook的账号有权限读取OBS桶中的数据。

- OBS文件被删除。请确认待上传的OBS文件是否存在。

- **异常场景2**

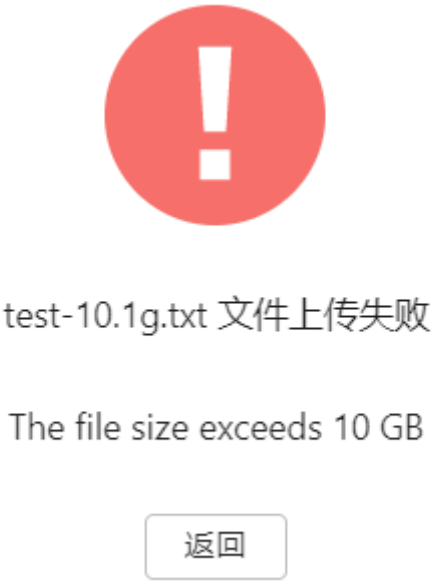
  图 6-82 文件上传失败

  可能原因：
  文件名包含<>'";\`=#$%^&等特殊字符。

- **异常场景3**

  图 6-83 文件上传失败

  可能原因：
  文件大小超过10GB导致上传失败。

6.4.7.1.5 上传远端文件至 JupyterLab

在Notebook的JupyterLab中，支持通过远端文件地址下载文件。

要求：远端文件的URL粘贴在浏览器的输入框中时，可以直接下载该文件。

1. 通过JupyterLab打开一个运行中的Notebook。
2. 单击JupyterLab窗口上方导航栏的 ModelArts Upload Files按钮，打开文件上传窗口，选择左侧的 进入远端文件上传界面。

图 6-84 上传文件图标

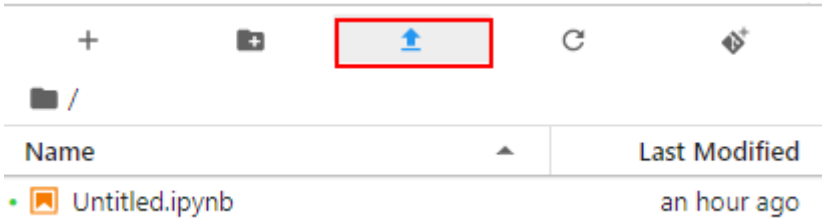


图 6-85 进入远端文件上传界面

3. 输入有效的远端文件URL后，系统会自动识别上传文件名称，单击“上传”，开始上传文件。

图 6-86 输入有效的远端文件 URL

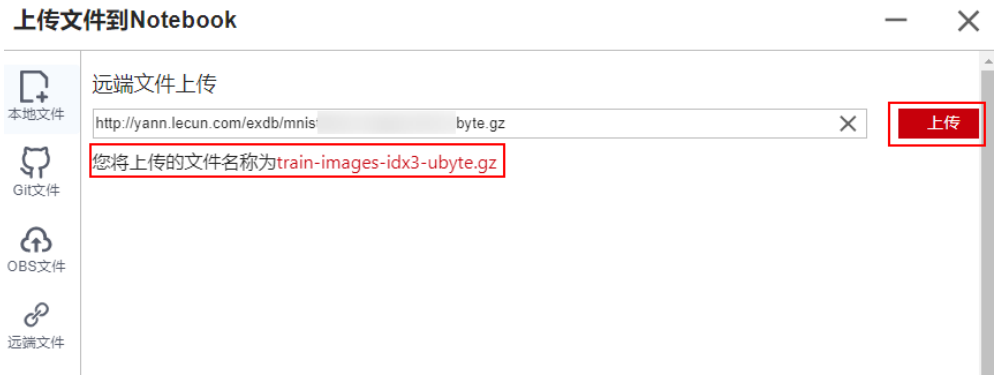
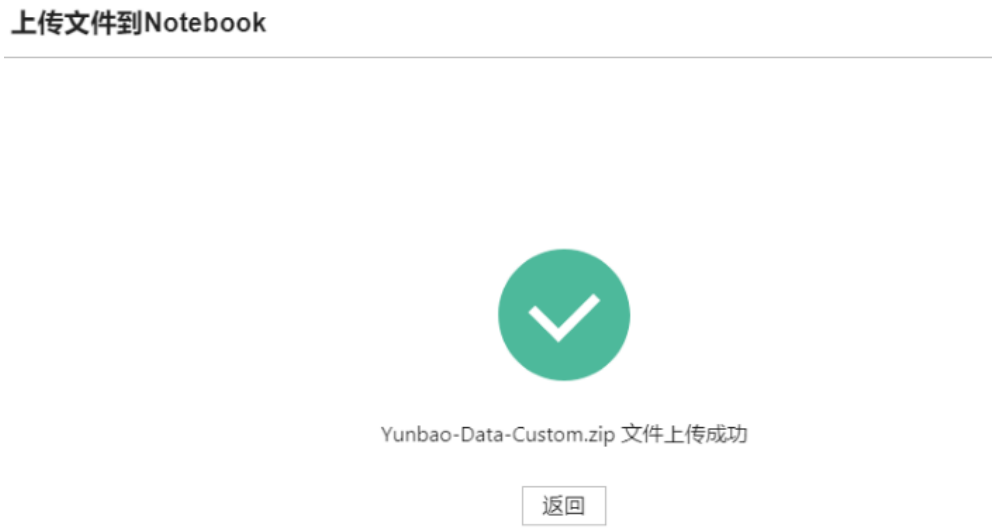


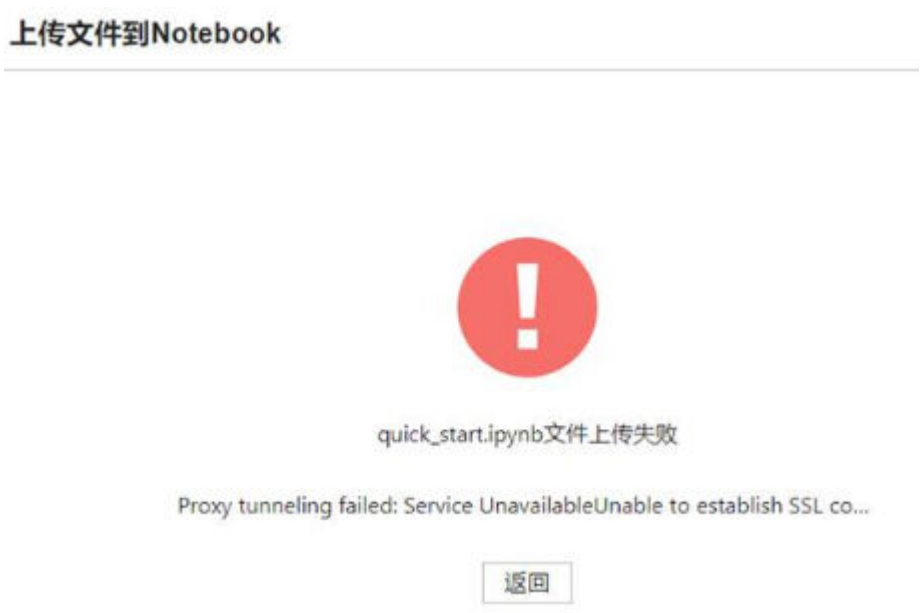
图 6-87 远端文件上传成功



异常处理

远端文件上传失败。可能是网络原因。请先在浏览器中输入该远端文件的URL地址，测试该文件是否能下载。

图 6-88 远端文件上传失败



6.4.7.2 从 JupyterLab 下载文件至本地

在JupyterLab中开发的文件，可以下载至本地。

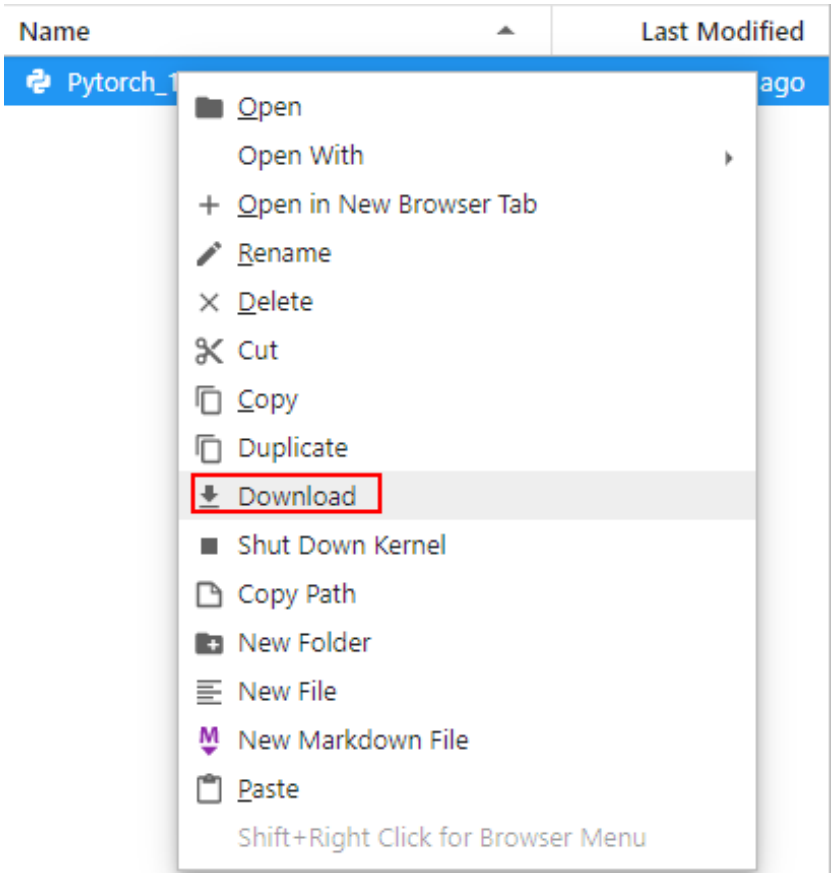
- 不大于100MB的文件，可以直接从JupyterLab中下载到本地，具体操作请参见**从JupyterLab中下载不大于100MB的文件至本地**。

- 大于100MB的文件，需要先从JupyterLab上传到OBS，再通过OBS下载到本地，具体操作请参见**从JupyterLab中下载大于100MB的文件到本地**。

从 JupyterLab 中下载不大于 100MB 的文件至本地

在JupyterLab文件列表中，选择需要下载的文件，单击右键，在操作菜单中选择“Download”下载至本地。下载的目的路径，为您本地浏览器设置的下载目录。

图 6-89 下载文件



从 JupyterLab 中下载大于 100MB 的文件到本地

大于100MB的文件需要先从Notebook中上传到OBS，再从OBS下载到本地，具体操作如下：

1. 在Notebook中，新建一个大于100MB的“ipynb”文件，使用MoXing先将该文件从Notebook上传到OBS中，示例代码如下：

   ```
   import moxing as mox
   mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
   ```

   其中“/home/ma-user/work/obs_file.txt”为文件在Notebook中的存储路径，“obs://bucket_name/obs_file.txt”为该文件上传到OBS的存储路径，其中“bucket_name”为OBS中创建的桶的名称，“obs_file.txt”为上传的文件。

2. 使用OBS或ModelArts SDK将OBS中的文件下载到本地。
   - 方式一：使用OBS进行下载
   - 在OBS中，可以将样例中的“obs_file.txt”下载到本地。如果您的数据较多，推荐OBS Browser+下载数据或文件夹。

- 方式二：使用ModelArts SDK进行下载
  i. 在您的本地环境下载并安装ModelArts SDK。
  ii. 完成ModelArts SDK的Session鉴权。
  iii. 将OBS中的文件下载到本地，详请参见“从OBS下载数据”。示例代码如下：

```
from modelarts.session import Session

# 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
# 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。
__AK = os.environ["HUAWEICLOUD_SDK_AK"]
__SK = os.environ["HUAWEICLOUD_SDK_SK"]
# 如果进行了加密还需要进行解密操作
session = Session(access_key=__AK,secret_key=__SK, project_id='***', region_name='***')

session.download_data(bucket_path="/bucket_name/obs_file.txt",path="/home/user/obs_file.txt")
```

6.5 本地 IDE

6.5.1 本地 IDE 操作流程

ModelArts支持通过本地IDE环境远程连接到Notebook中，开发基于PyTorch、TensorFlow和MindSpore引擎的AI模型。具体操作流程如下图所示。

1. 配置本地IDE
   在用户的PC端配置本地IDE环境。
   支持通过**PyCharm**、**VS Code**、**SSH工具**本地IDE连接云上Notebook。PyCharm和VS Code可以使用插件自动化配置，也可以手工配置。
2. **创建Notebook实例**
   在ModelArts控制台上创建一个Notebook开发环境实例，选择要使用的AI框架，并开启SSH远程开发功能。
3. 使用本地IDE远程连接到ModelArts的开发环境中。
4. **上传数据和代码至开发环境中**，进行代码调试。
   - 代码直接拷贝至本地IDE中即可，本地IDE中会自动同步至云上开发环境。
   - 不大于500MB数据量直接拷贝至本地IDE中即可。
   - **创建训练作业**大于500MB数据量请先上传到OBS中，从OBS上传到云硬盘EVS。
5. 将调试好的训练脚本和用于训练的数据集上传至OBS目录。
6. 提交训练作业。提交训练作业方式如下：
   - 在本地IDE中提交训练作业
   - 在ModelArts的Console控制台页面中提交训练作业。

6.5.2 本地 IDE（PyCharm）

6.5.2.1 PyCharm Toolkit 插件连接 Notebook

6.5.2.1.1 PyCharm ToolKit 介绍

由于AI开发者会使用PyCharm工具开发算法或模型，为方便快速将本地代码提交到ModelArts的训练环境，ModelArts提供了一个PyCharm插件工具PyCharm ToolKit（插件下载请参见**通过Marketplace安装**），协助用户完成代码上传、提交训练作业、将训练日志获取到本地展示等，用户只需要专注于本地的代码开发即可。

使用限制

- 当前仅支持PyCharm 2019.2及以上版本，包括社区版和专业版。
- 使用PyCharm ToolKit远程连接Notebook开发环境，仅限PyCharm专业版。使用云星账号的用户，不支持使用PyCharm ToolKit远程连接到Notebook功能。
- 使用PyCharm ToolKit提交训练作业，社区版和专业版都支持，PyCharm ToolKit latest版本仅限提交新版训练作业。
- PyCharm ToolKit工具仅支持Windows版本的PyCharm。

支持的功能

表 6-8 ToolKit（latest）功能列表

支持的功能	说明	对应操作指导
SSH远程连接	支持SSH远程连接ModelArts的Notebook开发环境。	**配置PyCharm ToolKit远程连接Notebook**
训练模型	支持将本地开发的代码，快速提交至ModelArts并自动创建**新版**训练作业，在训练作业运行期间获取训练日志并展示到本地。	• **提交训练作业（新版训练）** • **停止训练作业** • **查看训练日志**
OBS上传下载	上传本地文件或文件夹至OBS，从OBS下载文件或文件夹到本地。	**在PyCharm中上传数据至Notebook**

6.5.2.1.2 下载并安装 ToolKit 工具

在使用PyCharm ToolKit之前，您需要根据如下操作指导完成在PyCharm中的安装配置。

前提条件

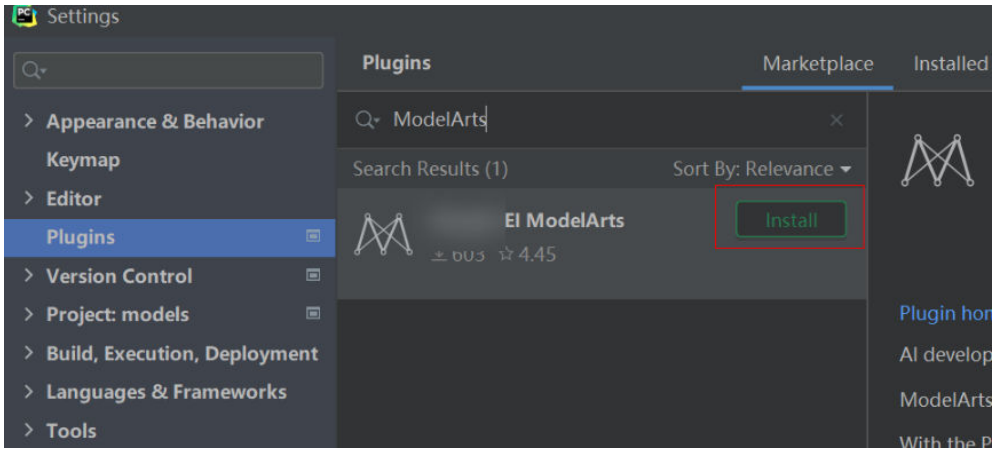
本地已安装2019.2及以上版本的PyCharm社区版或专业版。

- 使用PyCharm ToolKit远程连接Notebook开发环境，仅限PyCharm专业版。
- 使用PyCharm ToolKit提交训练作业，社区版和专业版都支持，但2.x版本仅限于提交旧版训练作业，latest版本仅限于提交新版训练作业。

通过 Marketplace 安装

在PyCharm中选择“File > Settings > Plugins”，在Marketplace里搜索“ModelArts”，单击“Install”即可完成安装。

图 6-90 通过 Marketplace 安装



说明

- 通过该方式安装的PyCharm ToolKit为latest版本。
- 如果Marketplace里搜索不到ModelArts，可能是用户网络限制原因，请确保可以正常访问外网。

6.5.2.1.3 PyCharm ToolKit 连接 Notebook

ModelArts提供了一个PyCharm插件工具PyCharm ToolKit，协助用户完成SSH远程连接Notebook、代码上传、提交训练作业、将训练日志获取到本地展示等，用户只需要专注于本地的代码开发即可。

前提条件

本地已安装2019.2及以上版本的PyCharm专业版。SSH远程开发功能只限PyCharm专业版。单击**PyCharm工具下载地址**下载并完成安装。

使用云星账号的用户，不支持使用PyCharm ToolKit远程连接到Notebook功能。

Step1 创建 Notebook 实例

创建一个Notebook实例，并开启远程SSH开发，配置远程访问IP白名单。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。

Step2 下载并安装 PyCharm ToolKit

在PyCharm中选择“File > Settings > Plugins”，在Marketplace里搜索“ModelArts”，单击“Install”即可完成安装。具体下载和安装过程请参见**下载并安装ToolKit工具**。

Step3 配置更多局点

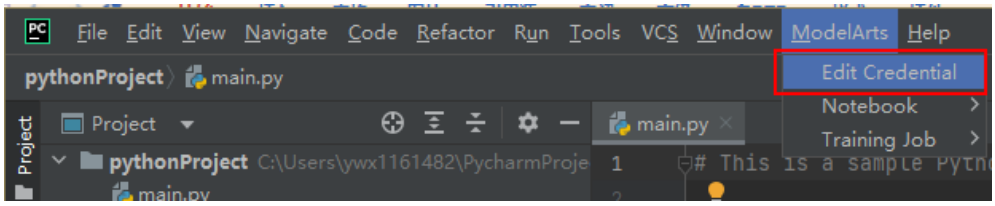
1. 在PyCharm工具中，选择菜单栏的“ModelArts > Edit Credential”，弹出“Edit Credential”对话框。
2. 联系局点运营公司获取YAML配置文件和host信息。在“Edit Credential”对话框中，单击“Config”，导入已经下载好的YAML文件，导入后会显示“Import successful”的提示，此时可以看到局点信息已配置成功。

Step4 登录插件

使用访问密钥完成登录认证操作如下：

1. 打开已安装ToolKit工具的PyCharm，在菜单栏中选择“ModelArts > Edit Credential”。

   图 6-91 Edit Credential

2. 在弹出的对话框中，选择您使用的ModelArts所在区域、填写AK、SK（获取方式**参考链接**），然后单击“OK”完成登录。
   - “Region”：从下拉框中选择区域。必须与ModelArts管理控制台在同一区域。
   - “Project”：Region选择后，Project自动填充为Region对应的项目。
   - “Access Key ID”：填写访问密钥的AK。
   - “Secret Access Key”：填写访问密钥的SK。
3. 查看认证结果。

   在Event Log区域中，当提示如下类似信息时，表示访问密钥添加成功。

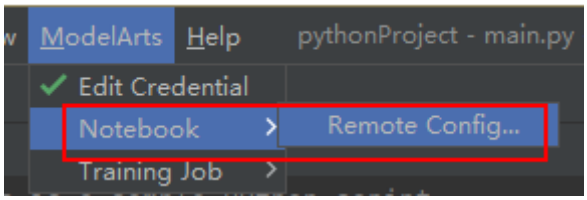
   ```
   16:01Validate Credential Success: The credential is valid.
   ```

   如果认证失败，请参考案例**PyCharm ToolKit工具中Edit Credential时，出现错误**解决。

Step5 插件自动化配置

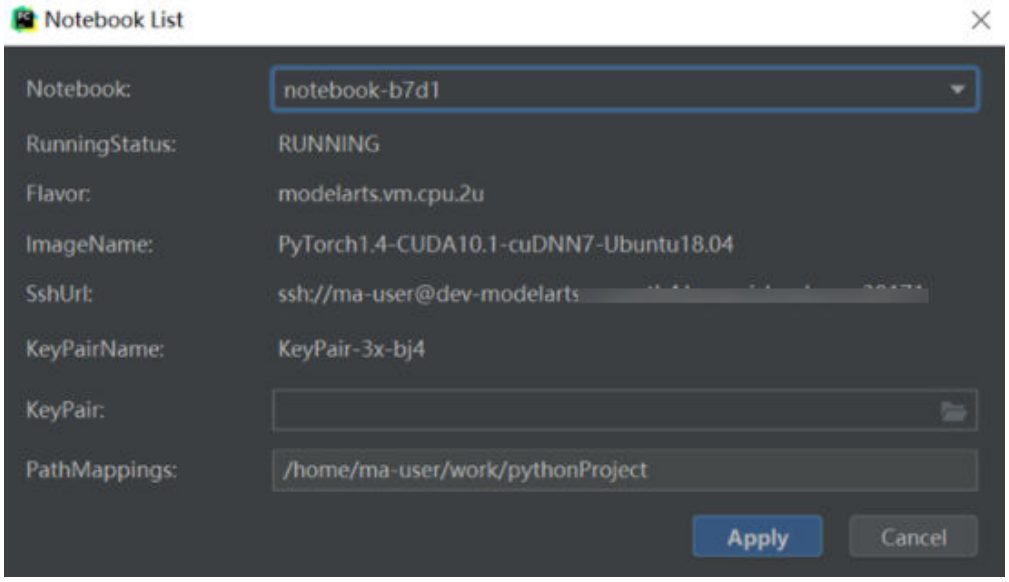
1. 在本地的PyCharm开发环境中，单击“ModelArts > Notebook > Remote Config...”，配置插件。

   图 6-92 配置插件

2. 此时，会出现该账号已创建的所有包含SSH功能的Notebook列表，下拉进行选择对应Notebook。

图 6-93 Notebook 列表



- KeyPair: 需要选择保存在本地的Notebook对应的keypair认证。即创建Notebook时创建的密钥对文件，创建时会直接保存到浏览器默认的下载文件夹中。
- PathMappings: 该参数为本地IDE项目和Notebook对应的同步目录，默认为/home/ma-user/work/project名称，可根据自己实际情况更改。

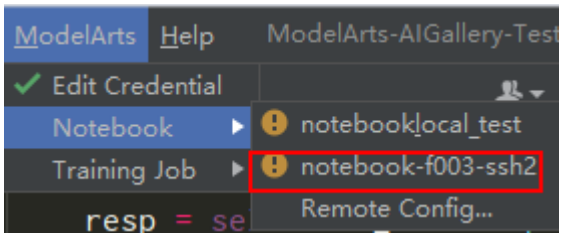
3. 单击“Apply”，配置完成后，重启IDE生效。

   重启后初次进行update python interpreter需要耗费20分钟左右。

Step6 使用插件连接云上 Notebook

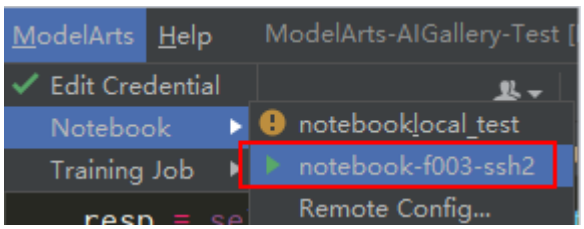
与Notebook断开连接的状态下，单击Notebook名称，根据提示启动本地IDE与Notebook的连接(默认启动时间4小时)。

图 6-94 启动连接 Notebook



连接状态下，单击Notebook名称，根据提示断开本地IDE与云上Notebook的连接。

图 6-95 停止连接 Notebook



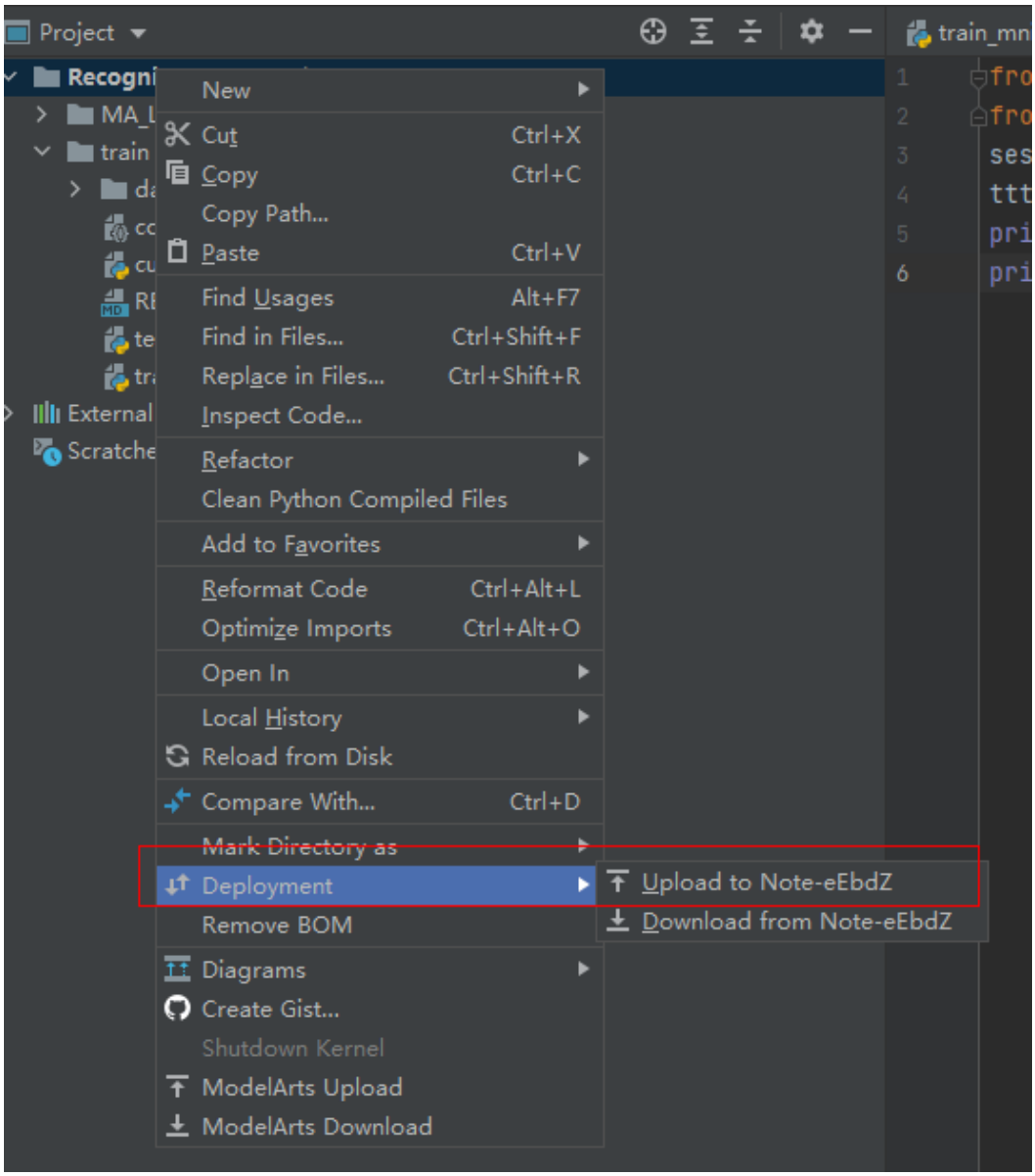
Step7 同步上传本地文件至 Notebook

本地文件中的代码直接复制至本地IDE中即可，本地IDE中会自动同步至云上开发环境。

初始化同步:

在本地IDE的Project目录下，单击右键，选择“Deployment”，单击“Upload to xxx”（Notebook名称），将本地工程文件上传至指定的Notebook。

图 6-96 同步本地文件至 Notebook

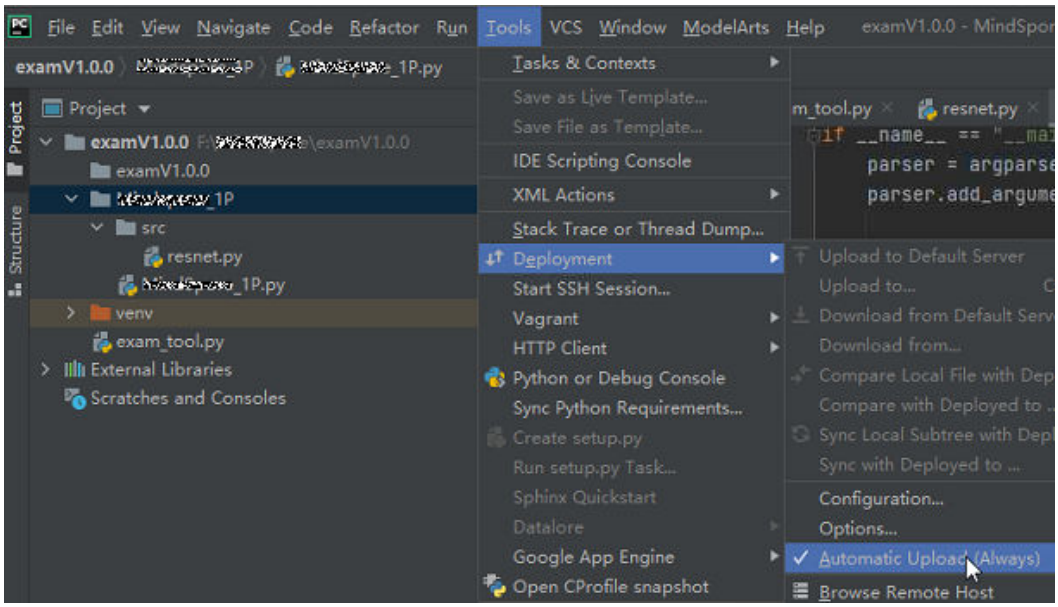


后续同步:

只需修改代码后保存（ctrl+s），即可进行自动同步。

插件安装完成后在本地IDE中开启了“Automatic Upload”，本地目录中的文件会自动上传至云端开发环境Notebook。如果未开启，请参考下图开启自动上传。

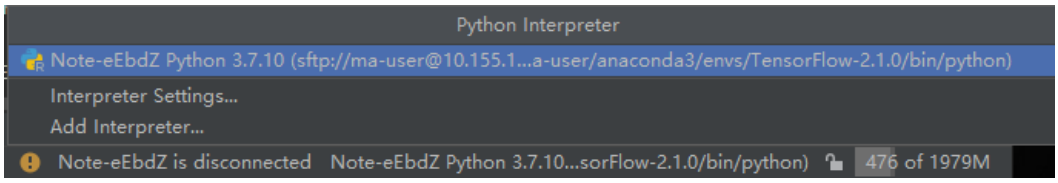
图 6-97 开启自动上传



Step8 远程调试

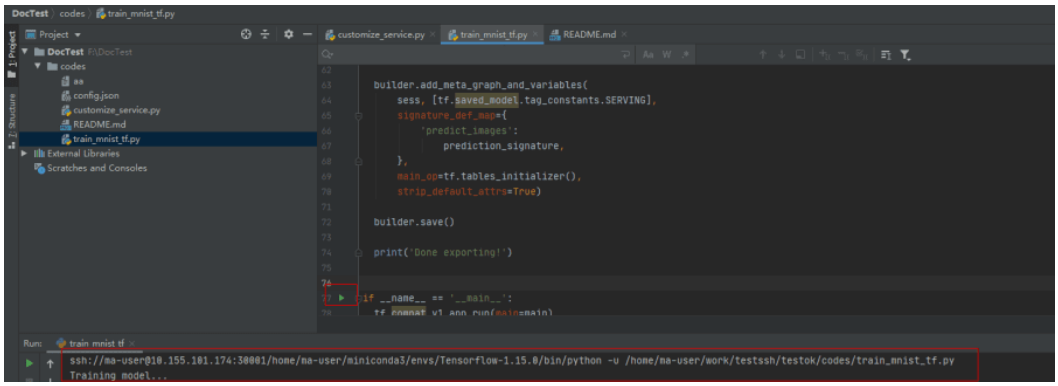
单击本地IDE右下角interpreter，选择Notebook的python解释器。

图 6-98 选择 Python 解释器



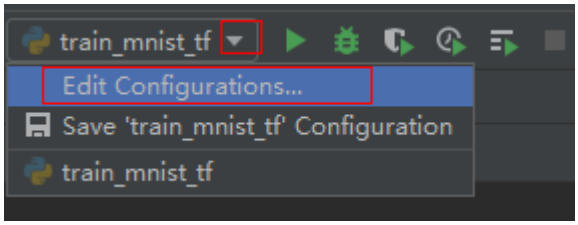
像本地运行代码一样，直接单击运行按钮运行代码即可，此时虽然是在本地IDE点的运行按钮，实际上运行的是云端Notebook里的代码，日志可以回显在本地的日志窗口。

图 6-99 查看运行日志



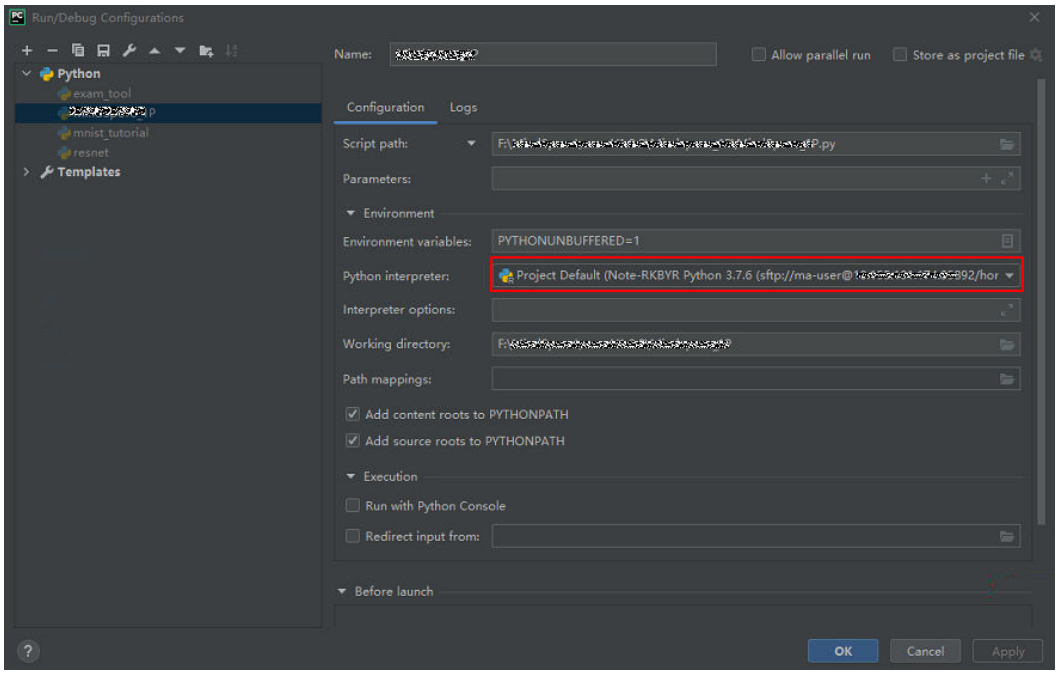
也可以单击本地IDE右上角的Run/Debug Configuration按钮来设置运行参数。

图 6-100 设置运行参数（1）



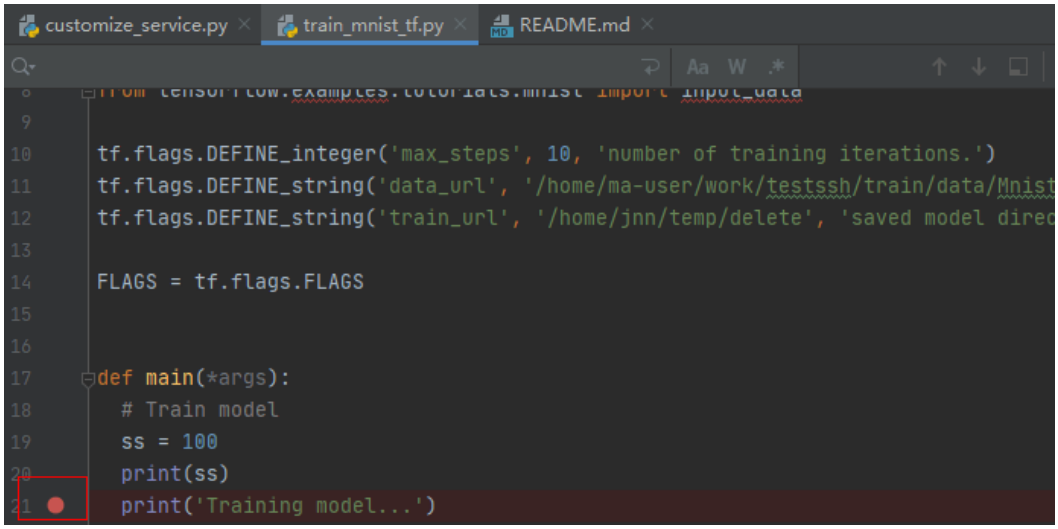
选择远程连接到云上开发环境实例对应的Python解释器。

图 6-101 设置运行参数（2）



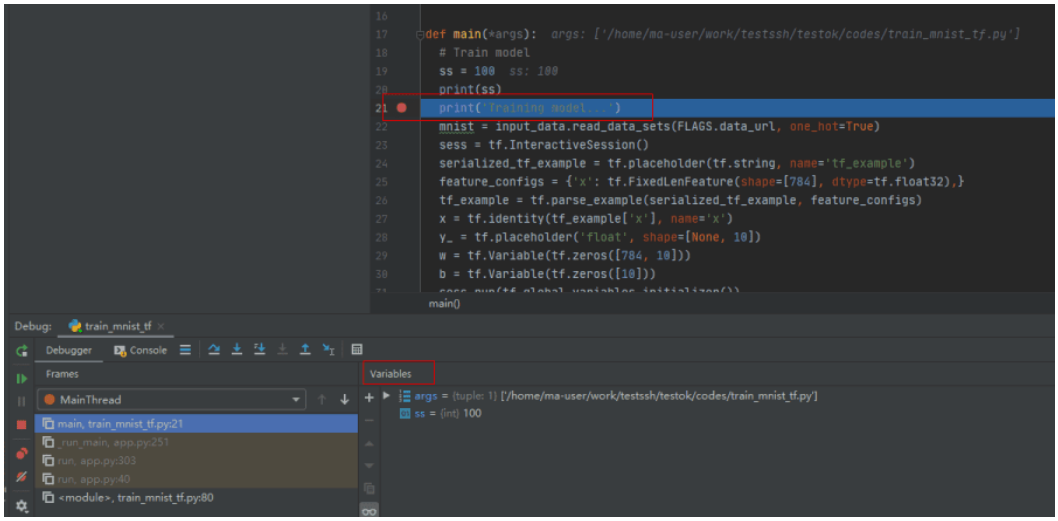
当需要调试代码时，可以直接打断点，然后使用debug方式运行程序。

图 6-102 使用 debug 方式运行程序



此时可以进入debug模式，代码运行暂停在该行，且可以查看变量的值。

图 6-103 Debug 模式下查看变量值



6.5.2.2 PyCharm 手动连接 Notebook

本地IDE环境支持Pycharm和VS Code。通过简单配置，即可用本地IDE远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

本章节介绍基于PyCharm环境访问Notebook的方式。

前提条件

- 本地已安装2019.2及以上版本的PyCharm专业版。SSH远程调试功能只限PyCharm专业版。
- 创建一个Notebook实例，并开启远程SSH开发。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。
- 在Notebook实例详情页面获取开发环境IP地址和端口号。

  图 6-104 Notebook 实例详情页面

- 准备好密钥对。

  密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。

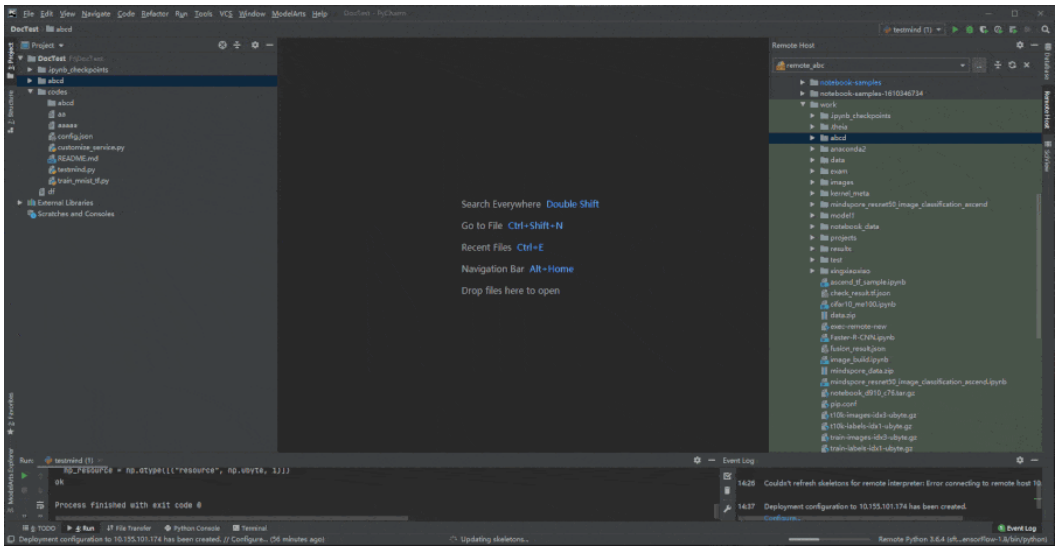
Step1 配置 SSH

1. 在本地的PyCharm开发环境中，单击File -> Settings -> Tools -> SSH Configurations，单击+号，增加一个SSH连接配置。
   - Host: 云上开发环境的IP地址，即在开发环境实例页面远程访问模块获取的IP地址。

- Port: 云上开发环境的端口，即在开发环境实例页面远程访问模块获取的端口号。
- User name: 固定为ma-user
- Authentication type: Key pair方式
- Private key file: 存放在本地的云上开发环境私钥文件，即在创建开发环境实例时创建并保存的密钥对文件。

2. 单击  将连接重命名，可以自定义一个便于识别的名字，单击OK。
3. 配置完成后，单击Test Connection测试连通性。
4. 选择Yes，显示Successfully connected表示网络可以连通，单击OK。
5. 在最下方再单击OK保存配置。

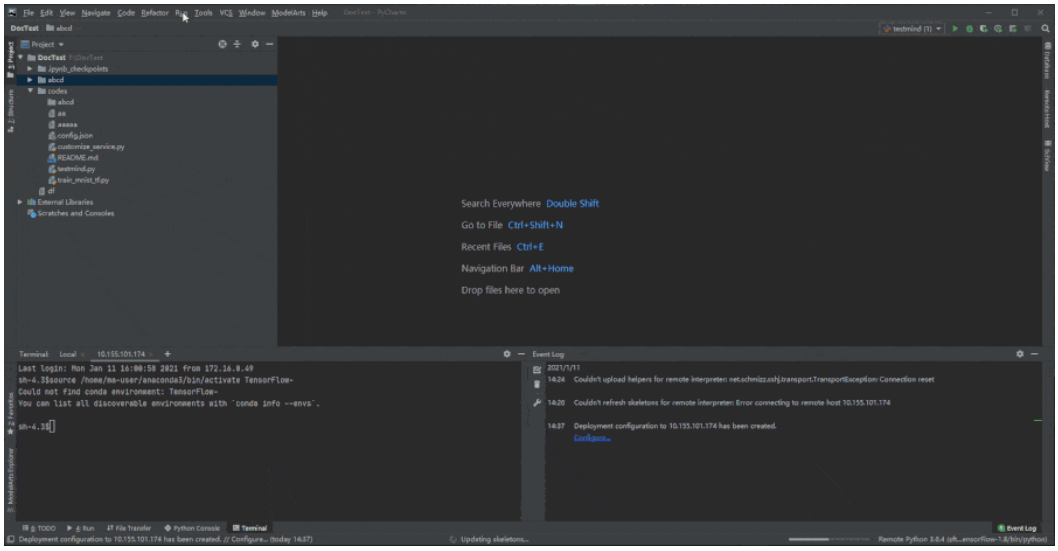
图 6-105 配置 SSH



Step2 获取开发环境预置虚拟环境路径

1. 单击“Tools > Start SSH Session”，则可连接到云端开发环境内。
2. 执行如下命令可在/home/ma-user/下面的README文件查看当前环境内置的Python虚拟环境。
   ```
   cat /home/ma-user/README
   ```
3. 执行source命令可以切换到具体的Python环境中。
4. 执行which python查看python路径并复制出来，以备后续配置云上Python Interpreter使用。

图 6-106 获取开发环境预置虚拟环境路径



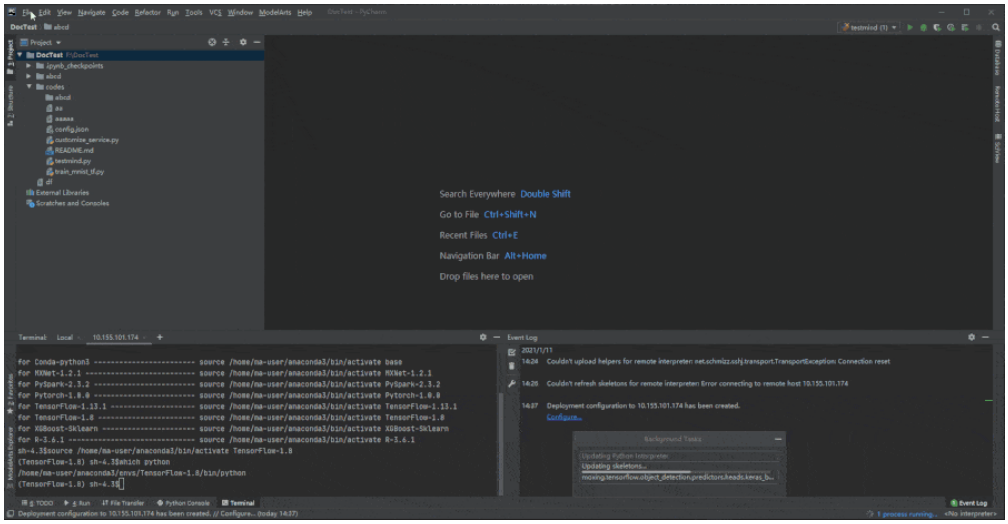
Step3 配置云上 Python Interpreter

1. 单击“File > Settings > Project：PythonProject > Python Interpreter”，单击设置图标  ，再单击“Add”，添加一个新的interpreter。
2. 选择“Existing server configuration”，在下拉菜单中选择上一步配置好的SSH configuration，单击“Next”。
3. 配置Python Interpreter
   - Interpreter：填写第一步复制的python路径，例如：/home/ma-user/anaconda3/envs/Pytorch-1.0.0/bin/python
     如果路径为~/anaconda3/envs/Pytorch-1.0.0/bin/python 把~替换为/home/ma-user即可。
   - Sync folders: 需要配置本地的工程目录文件同步到云上开发环境中的某个目录，推荐配置为/home/ma-user下的某个目录中（其他目录可能没有访问权限），例如/home/ma-user/work/projects。
4. 单击右侧文件夹图标，勾选上“Automatically upload”选项，以便于本地修改的文件自动上传到容器环境中。
5. 单击“Finish”，结束配置。

   可以看到本地的工程文件已经自动往云上环境上传了。后续本地的文件每修改一次，都会自动的同步到云上的环境中。

   右下角可以看到当前的Interpreter为Remote Interpreter。

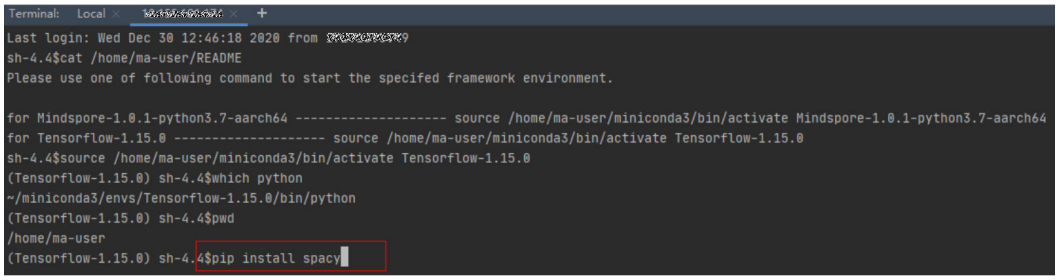
图 6-107 配置云上 Python Interpreter



Step4 云上环境依赖库安装

在进入开发环境后，可以使用不同的虚拟环境，例如TensorFlow、PyTorch等，但是实际开发中，通常还需要安装其他依赖包，此时可以通过Terminal连接到环境里操作。

单击工具栏“Tools >Start SSH session”，选择SSH Configuration中配置的开发环境。可以执行pip install安装所需要的包。

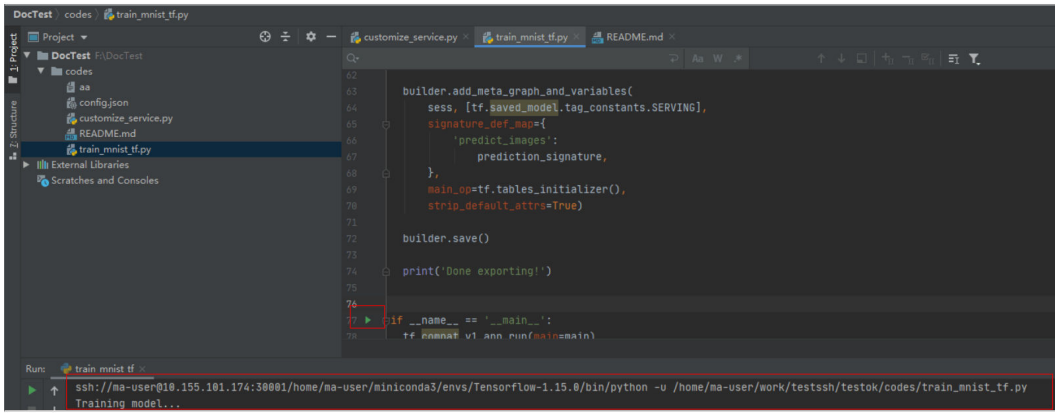


Step5 在开发环境中调试代码

由于已经连接至云端开发环境，此时可以方便的在本地PyCharm中编码、调测并运行。运行实际环境为云上开发环境，资源为云上昇腾AI处理器资源。可以做到本地编写修改代码，直接在云上环境运行。

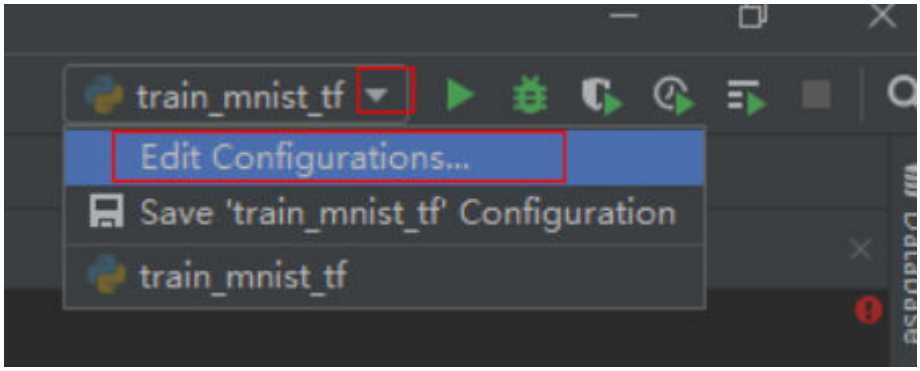
像本地运行代码一样，直接单击运行按钮运行代码即可，此时虽然是在本地IDE单击的运行按钮，实际上运行的是云端开发环境里的代码，日志可以回显在本地的日志窗口。

图 6-108 调试代码



也可以单击右上角的Run/Debug Configuration来设置运行的参数。

图 6-109 设置运行参数



当需要调试代码时，可以直接打断点，然后使用debug方式运行程序。

图 6-110 代码打断点

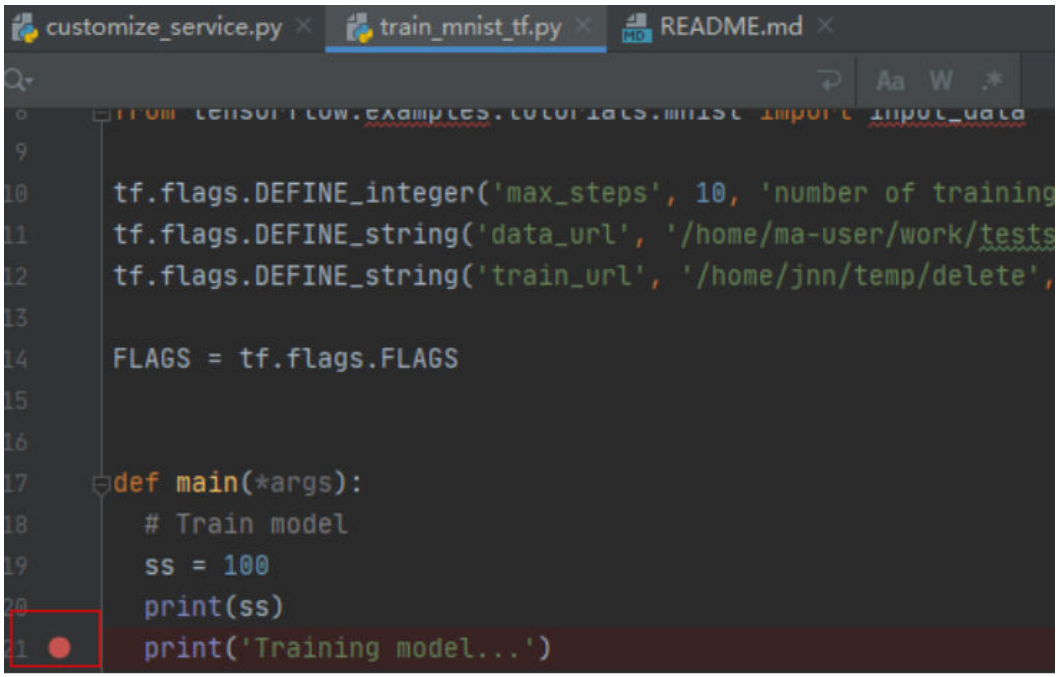
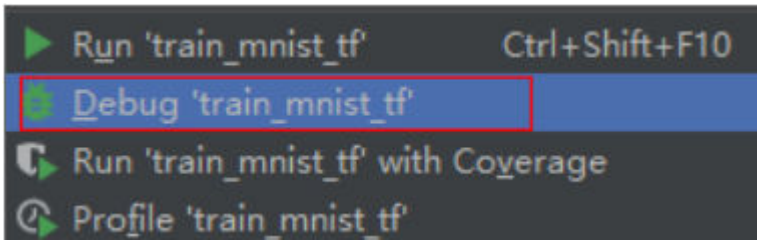
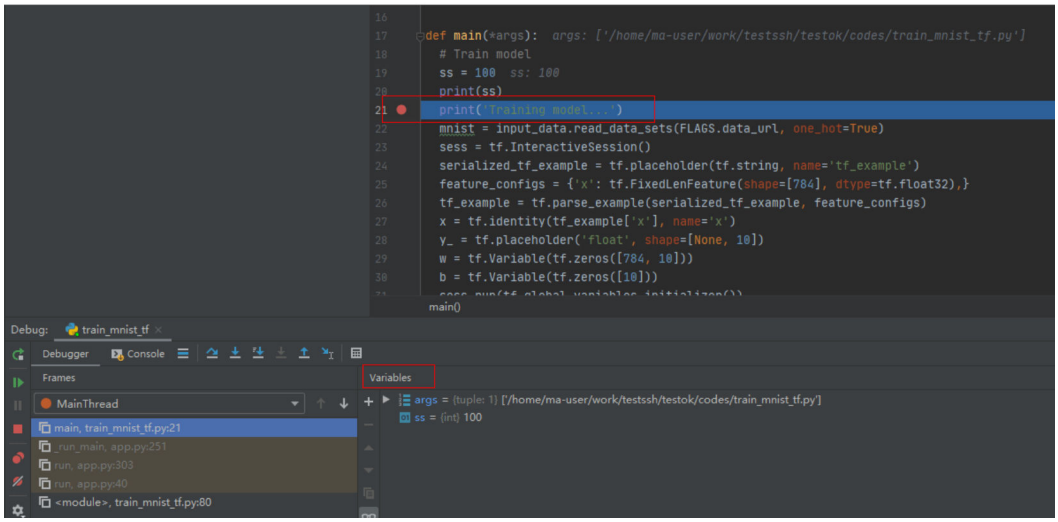


图 6-111 Debug 方式调试



此时可以进入debug模式，代码运行暂停在该行，且可以查看变量的值。

图 6-112 Debug 模式



使用debug方式调试代码的前提是本地的代码和云端的代码是完全一致的，如果不一致可能会导致在本地打断点的行和实际运行时该行的代码并不一样，会出现意想不到的错误。

因此在配置云上Python Interpreter时，推荐选择Automatically upload选项，以保证本地的文件修改能自动上传到云端。如果没有选择自动上传，则本地代码修改完后，也可以参考**Step7 同步上传本地文件至Notebook**手动上传目录或代码。

6.5.2.3 PyCharm Toolkit 提交训练作业

6.5.2.3.1 提交训练作业（新版训练）

使用PyCharm ToolKit（latest版本）工具，可以快速将本地开发的训练代码，提交至ModelArts侧进行训练。

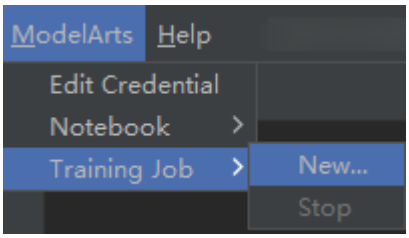
前提条件

- 在本地PyCharm中已有训练代码工程。
- 已在OBS中创建桶和文件夹，用于存放数据集和训练输出模型。 训练作业使用的数据已上传至OBS。
- 已配置credential，详细请参考**使用访问密钥登录**。
- 使用PyCharm ToolKit（latest版本）提交训练作业，仅限于新版训练作业，不支持旧版训练作业。

配置训练作业参数

1. 在PyCharm中，打开训练代码工程和训练启动文件，然后在菜单栏中选择“ModelArts > Training Job > New...”。

   图 6-113 选择作业配置

2. 在弹出的对话框中，设置训练作业相关参数，详细参数说明请参见**表6-9**。

   表 6-9 训练作业配置参数说明

参数	说明
Job Name	训练作业的名称。 系统会自动生成一个名称，您可以根据业务需求重新命名，命名规则如下： • 支持1~64位字符。 • 并包含大小写字母、数字、中划线（-）或下划线（_）。

参数	说明
Job Description	训练作业的简要描述。
Algorithm Source	训练算法来源，分为“常用框架”和“自定义镜像”两种，二者选一项即可。 常用框架指使用ModelArts训练管理中支持的常用AI引擎。 如果您使用的AI引擎为支持列表之外的，建议使用自定义镜像的方式创建训练作业。
AI Engine	选择代码使用的AI引擎及其版本。支持的AI引擎与ModelArts管理控制台里一致。
Boot File Path	训练启动文件，所选启动文件必须是当前PyCharm训练工程中的文件。当“Algorithm Souce”选“Frequently-used”时，显示此参数。
Code Directory	训练代码目录，系统会自动填写为训练启动文件所在的目录，用户可根据需要修改，所选目录必须是当前工程中的目录且包含启动文件。 当算法来源为自定义镜像，训练代码已预置在镜像中时，该参数可以为空。
Image Path(optional)	SWR镜像的URL地址。
Boot Command	启动本次训练作业的运行命令。例如“bash /home/work/run_train.sh python {python启动文件及参数}”。当“Algorithm Souce”选“Custom”时，显示此参数。 当用户输入的命令中不包含“--data_url”和“--train_url”参数时，工具在提交训练作业时会在命令后面自动添加这两个参数，分别对应存储训练数据的OBS路径和存放训练输出的OBS路径。
Data OBS Path	设置为存储训练数据的OBS路径，例如“/test-modelarts2/mnist/dataset-mnist/”，其中“test-modelarts2”为桶名称。
Training OBS Path	设置OBS路径，该路径下会自动创建用于存放训练输出模型和训练日志的目录。
Running Parameters	运行参数。如果您的代码需要添加一些运行参数，可以在此处添加，多个运行参数使用英文分号隔开， 例如"key1=value1;key2=value2"。此参数也可以不设置，即保持为空。
Specifications	训练使用资源类型。目前支持公共资源池和专属资源池两种类型。 专属资源池规格以“Dedicated Resource Pool”标识。
Compute Nodes	计算资源节点个数。数量设置为1时，表示单机运行；数量设置大于1时，表示后台的计算模式为分布式。

参数	说明
Available/Total Nodes	当“Specifications”选择专属资源池规格时，显示专属资源池的可用节点数和总节点数，用户选择“Compute Nodes”的个数不要超过可用节点数。

图 6-114 配置训练作业参数（公共资源池）

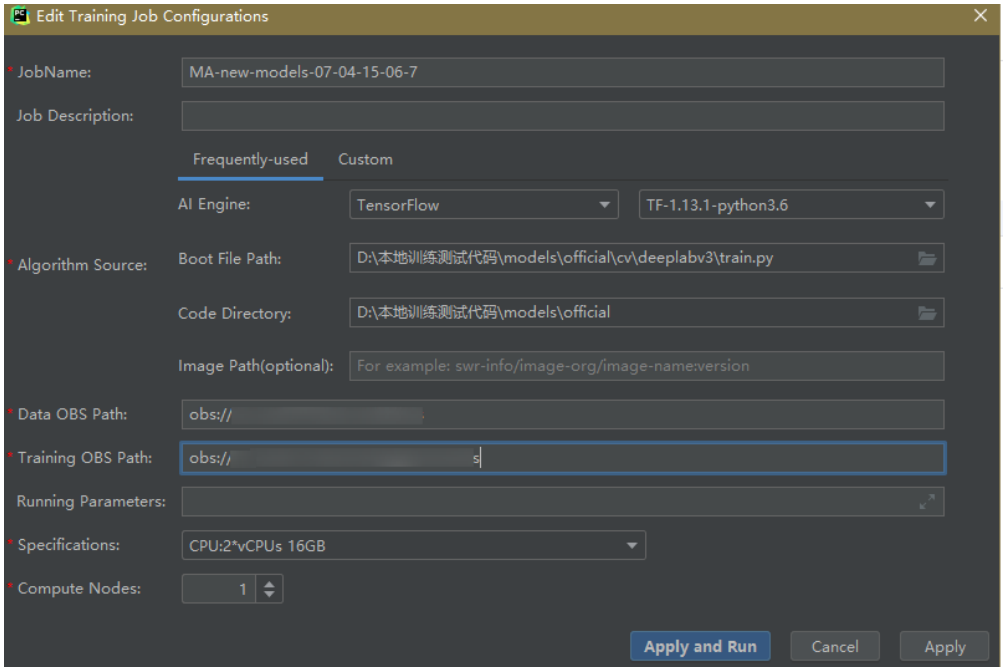


图 6-115 配置训练作业参数（专属资源池）

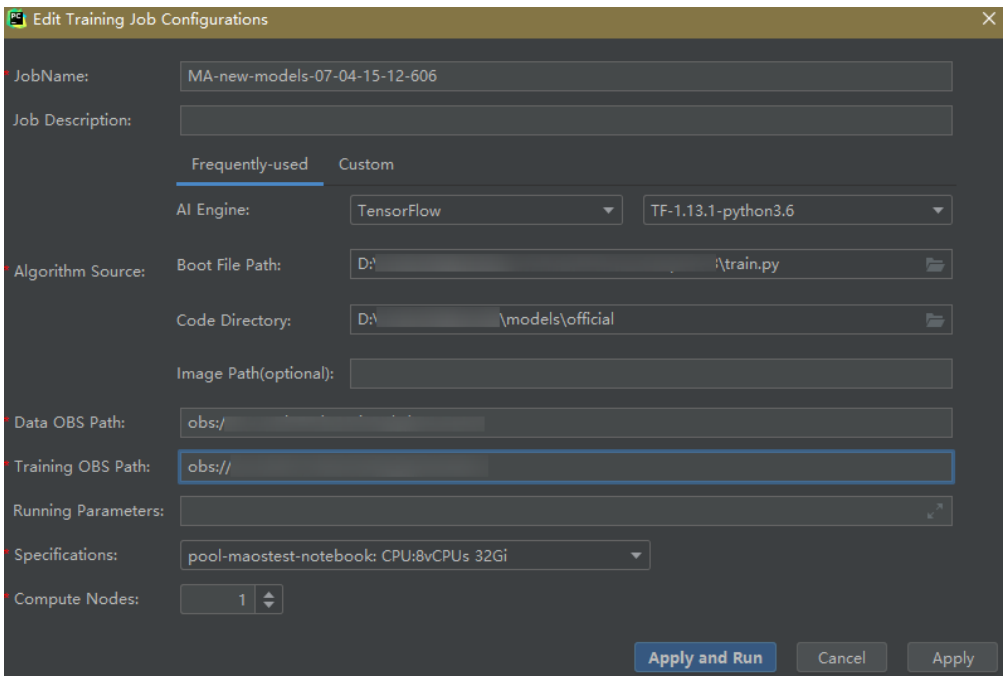
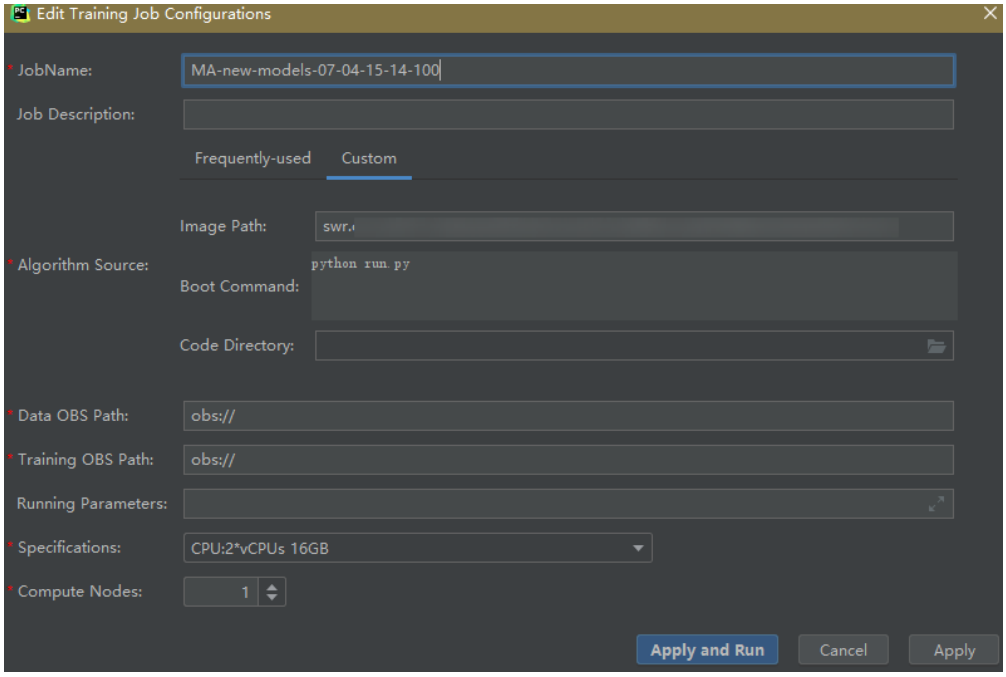


图 6-116 配置训练作业参数（自定义镜像）

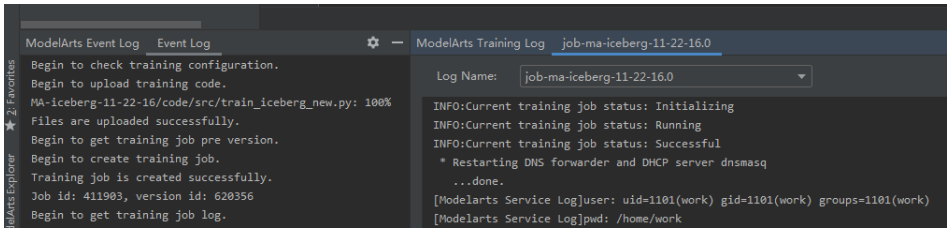


3. 参数填写完成后，单击“Apply and Run”，即自动上传本地代码至云端并启动训练，在工具下方的Training Log区域，会实时展示训练作业运行情况。当训练日志中出现“Current training job status: Successful”类似信息时，表示训练作业运行成功。

   说明

   - 在单击“Apply and Run”按钮后，系统将自动开始执行训练作业。如果您想停止此作业，可以选择菜单栏中的“ModelArts > Training Job >Stop”停止此作业。
   - 如果单击“Apply”，不会直接启动运行，只是保存训练作业的设置，如果需要启动作业，可以单击“Apply and Run”。

图 6-117 训练日志展示样例



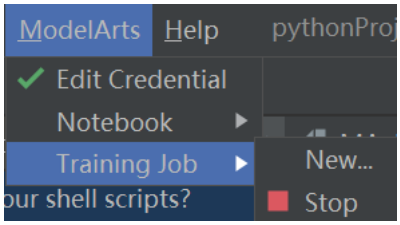
6.5.2.3.2 停止训练作业

当训练作业在运行过程中时，您可以执行停止作业的操作。

停止作业

当训练作业在运行过程中时，您可以在PyCharm菜单栏中，选择“ModelArts >Training Job > Stop ”停止此作业。

图 6-118 停止作业



6.5.2.3.3 查看训练日志

本章节介绍如何查看训练作业产生的日志。

在 OBS 中查看

提交训练作业时，系统将自动在您配置的OBS Path中，使用作业名称创建一个新的文件夹，用于存储训练输出的模型、日志和代码。

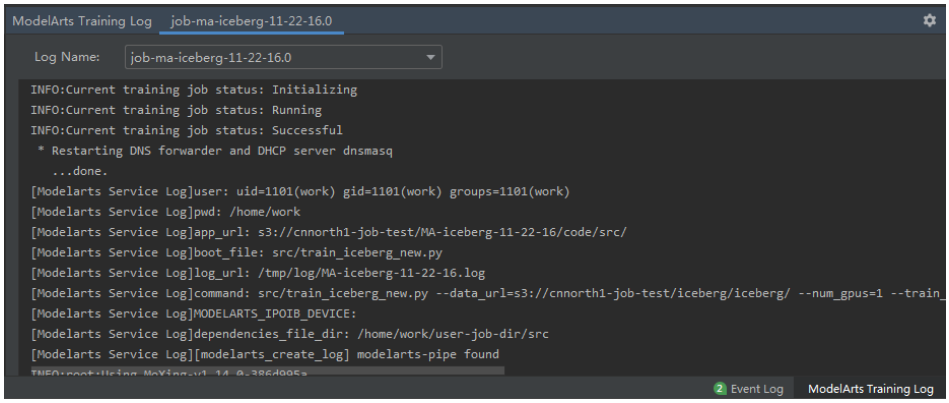
例如“train-job-01”作业，提交作业时会在“test-modelarts2”桶下创建一个命名为“train-job-01”的文件夹，且此文件夹下分别新建了三个文件夹“output”、“log”、“code”，分别用于存储输出模型、日志和训练代码。“output”文件夹还会根据您的训练作业版本再创建子文件夹，结构示例如下。

```
test-modelarts2
  |---train-job-01
      |---output
      |---log
      |---code
```

在 ToolKit 工具中查看

在PyCharm工具中，单击页面右下角的ModelArts Training Log，展示训练日志。

图 6-119 查看训练日志



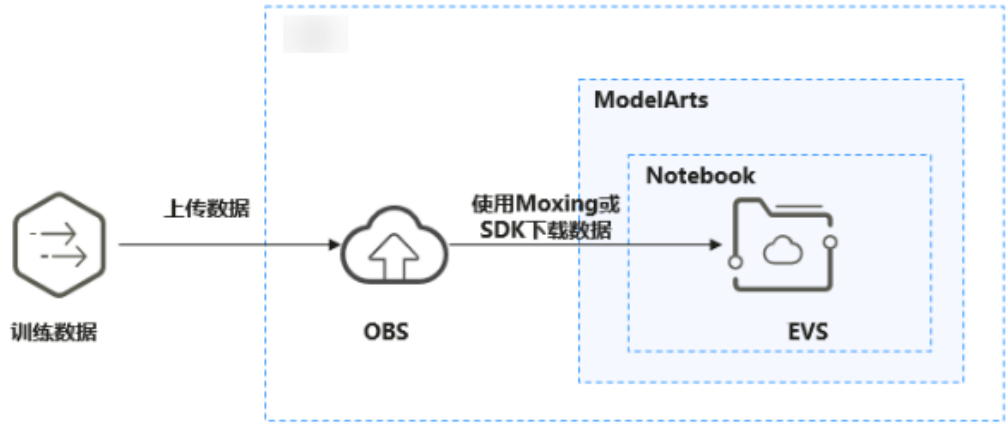
6.5.2.4 在 PyCharm 中上传数据至 Notebook

不大于500MB数据量，直接拷贝至本地IDE中即可。

大于500MB数据量，请先上传到OBS中，再从OBS上传到云上开发环境。

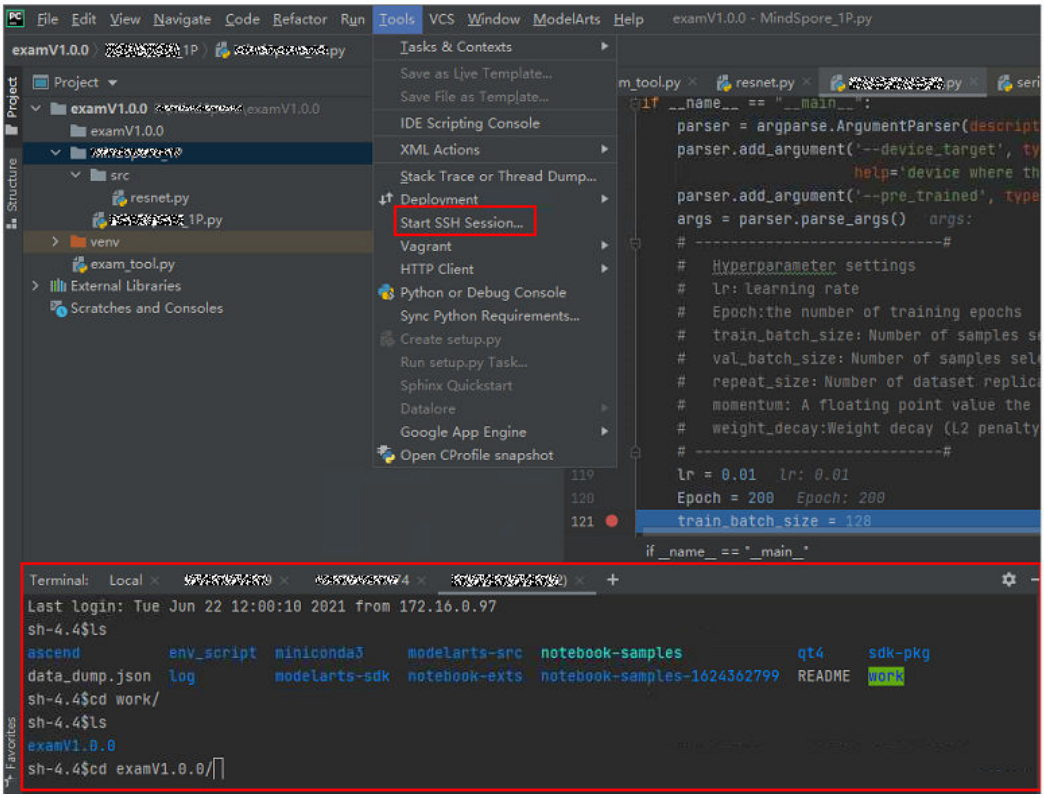
1. 上传数据至OBS。
2. 将OBS中的数据传至Notebook中，通过在本地IDE的Terminal中使用ModelArts提供的Moxing库的文件操作API（mox.file.copy_parallel）完成。

图 6-120 数据通过 OBS 中转上传到 Notebook



下图以PyCharm环境中开启Terminal为例。

图 6-121 PyCharm 环境开启 Terminal



在本地IDE的Terminal中使用Moxing下载OBS文件到开发环境的操作示例如下：

```
#手动source进入开发环境
cat /home/ma-user/README
#然后选择要source的环境
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
#输入python并回车，进入python环境
python
#使用moxing
import moxing as mox
#下载一个OBS文件夹，从OBS下载至EVS（OBS -> EVS）
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/tmp/sub_dir_0')
```

6.5.3 本地 IDE（VS Code）

6.5.3.1 VS Code 连接 Notebook 方式介绍

当用户创建完成支持SSH的Notebook实例后，使用VS Code的开发者可以通过以下三种方式连接到开发环境中：

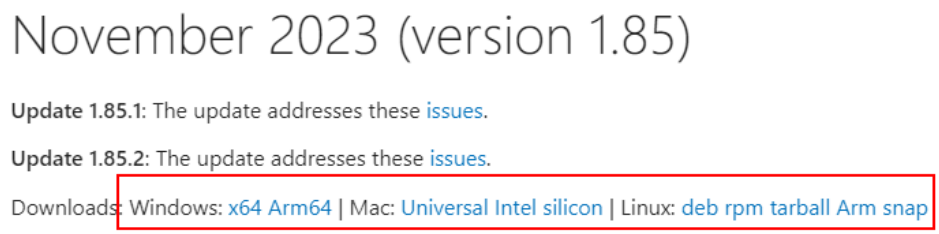
- **VS Code一键连接Notebook**（推荐）
  该方式是指在开发环境Console控制台上提供VS Code按钮，通过该入口自动打开VS Code并连接实例。
- **VS Code ToolKit连接Notebook**（推荐）
  该方式是指用户在VS Code上使用ModelArts VS Code Toolkit插件提供的登录和连接按钮，连接云上实例。
- **VS Code手动连接Notebook**
  该方式是指用户使用VS Code Remote SSH插件手工配置连接信息，连接云上实例。

6.5.3.2 安装 VS Code 软件

VS Code下载方式：

- 下载地址: **https://code.visualstudio.com/updates/v1_85**

图 6-122 VS Code 的下载位置



VS Code版本要求：

建议用户使用VS Code 1.85.2版本或者最新版本进行远程连接。

VS Code安装指导如下：

Linux系统下，执行命令**sudo dpkg -i code_1.85.2-1705561292_amd64.deb**安装。

说明

Linux系统用户，需要在非root用户进行VS Code安装。

6.5.3.3 VS Code 一键连接 Notebook

前提条件

- 已经创建Notebook实例 ，实例已经开启SSH连接，实例状态为运行中。 请参考**创建Notebook实例**。

- 实例的密钥文件已经下载至本地的如下目录或其子目录中：
  Windows：C:\Users\{{user}}
  Mac/Linux: Users/{{user}}

操作步骤

步骤1 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”页面。

步骤2 该界面显示已创建实例的状态为“运行中”。当前有两种方式，可以打开VS Code连接。单击“操作”列的“更多 > VS Code接入”；或者单击“操作”列的“打开”，自动进入Launcher页面，然后单击“VS Code”。弹出“是否打开Visual Studio Code？”对话框。

图 6-123 打开 VS Code 接入

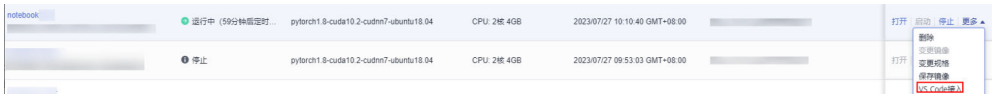
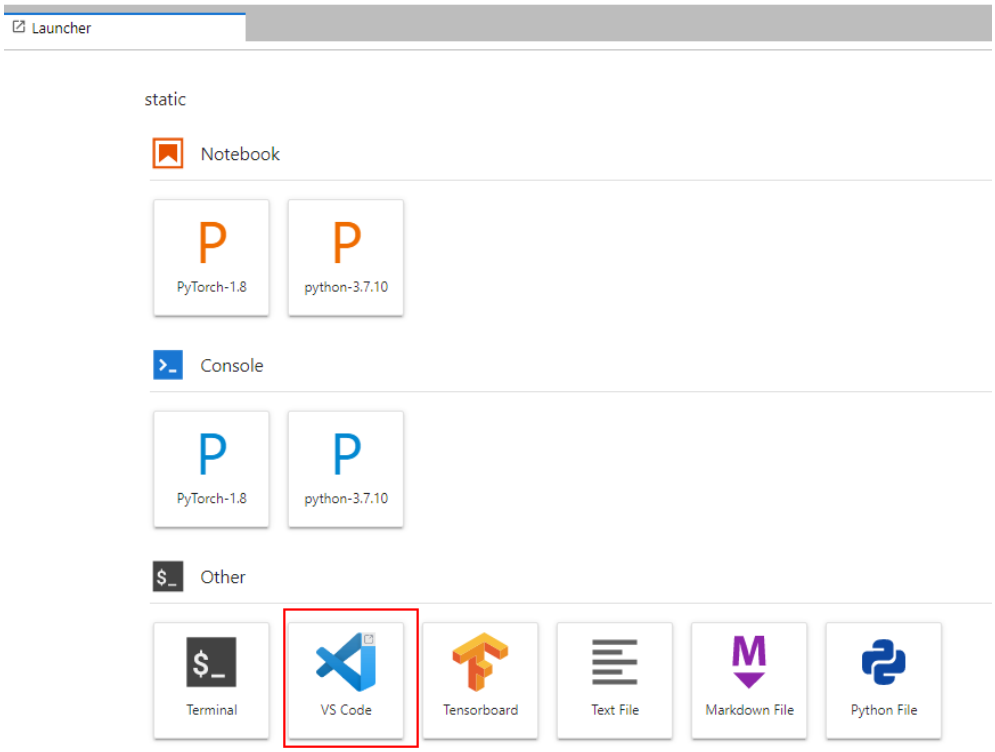
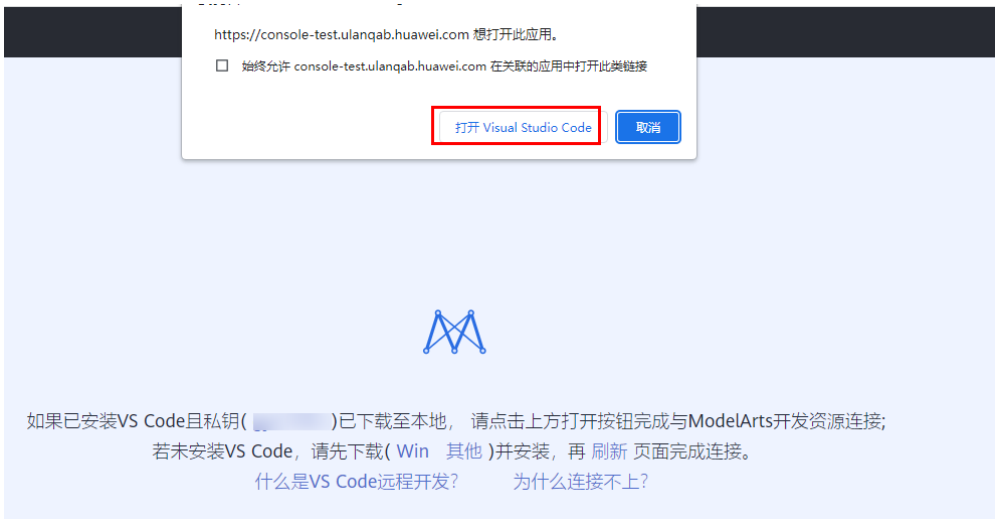


图 6-124 从 Launcher 页面打开 VS Code 接入



步骤3 如果本地已安装VS Code，请单击“打开 Visual Studio Code”，进入“Visual Studio Code”页面。

图 6-125 打开 Visual Studio Code

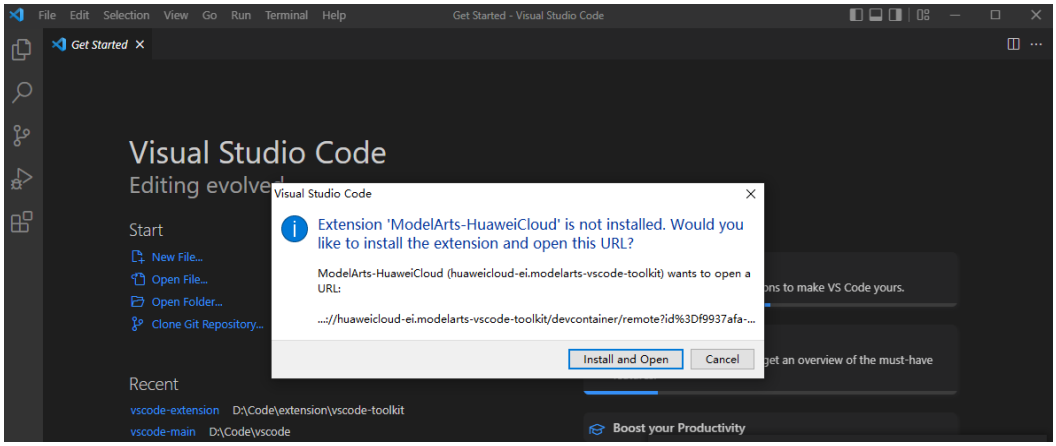


如果本地未安装VS Code，请根据实际选择“win”或“其他”下载并安装VS Code。VS Code安装请参考**安装VS Code软件**。

图 6-126 下载并安装 VS Code

步骤4 如果用户之前未安装过ModelArts VS Code插件，此时会弹出安装提示，请单击“Install and Open”进行安装；如果之前已经安装过插件，则不会有该提示，请跳过此步骤，直接执行**5**。

图 6-127 安装 VS Code 插件

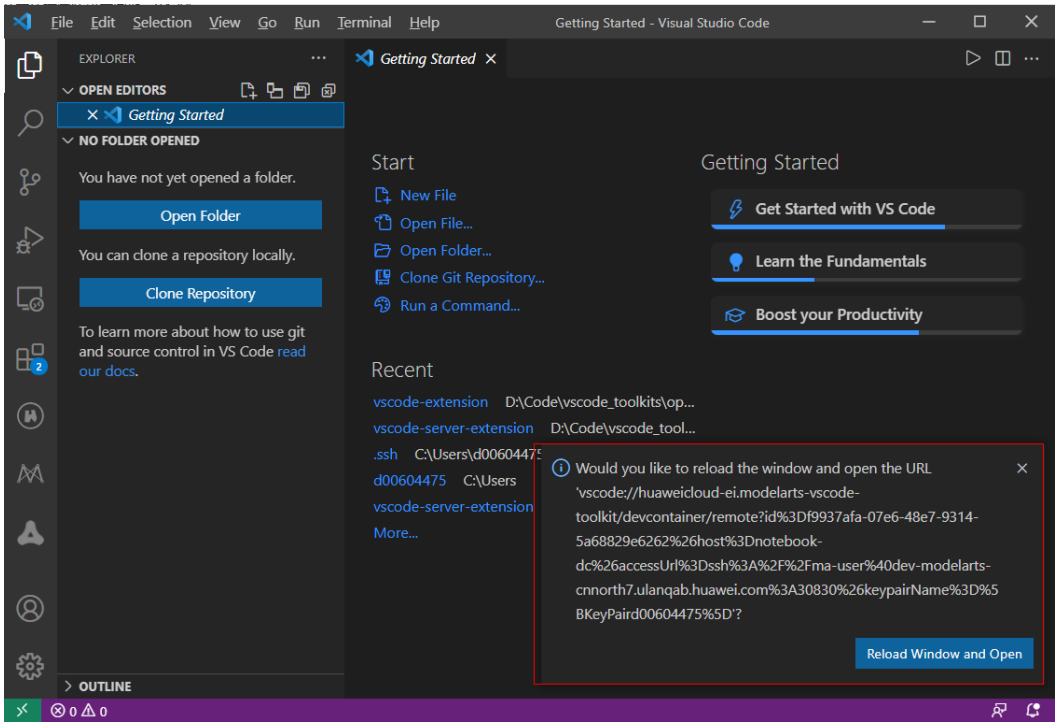


安装过程预计1~2分钟，安装完成后右下角会弹出对话框，请单击“Reload Window and Open”。

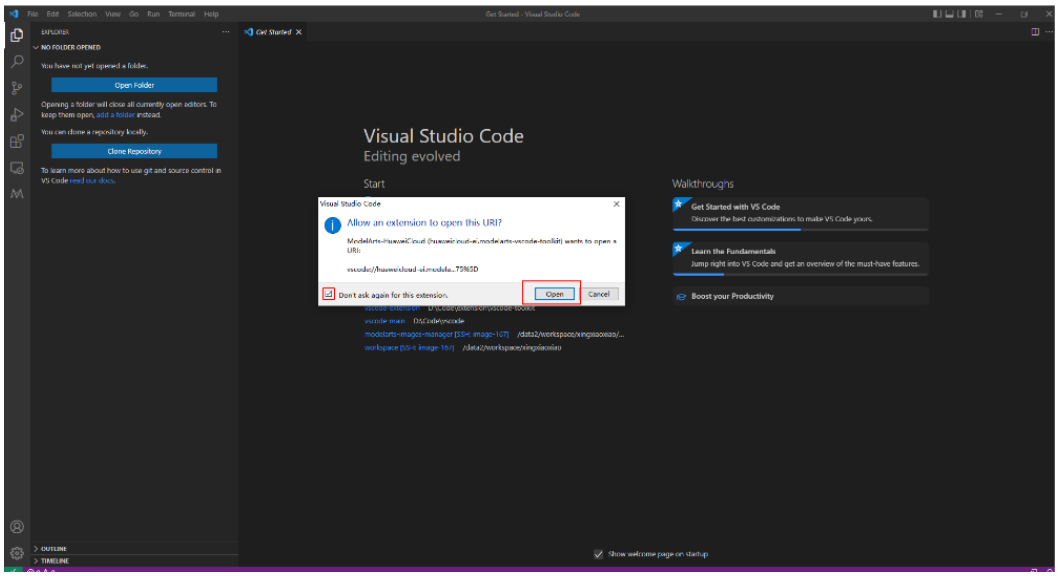
说明

本文以VS Code 1.78.2版本的操作为例，其它版本的VS Code可能不会弹出“Reload Window and Open”，请直接执行5。

图 6-128 Reload Window and Open



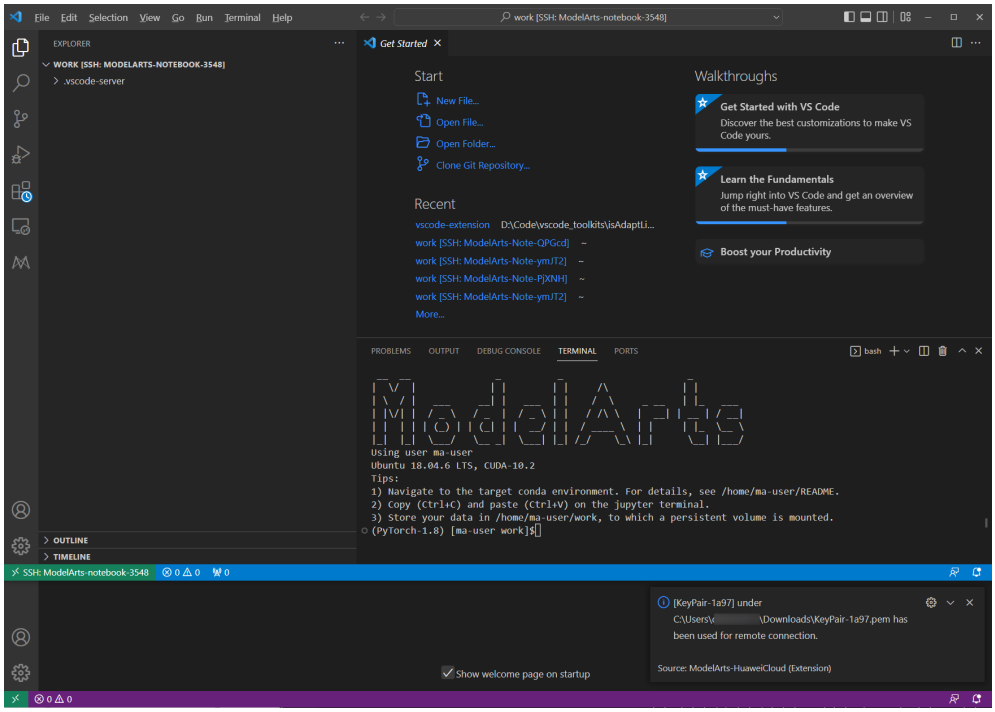
在弹出的提示中，勾选“Don't ask again for this extension”，然后单击"Open"。



步骤5 远程连接Notebook实例。

- 远程连接执行前，会自动在（Windows：C:\Users\{{user}}\.ssh或者downloads，Mac/Linux: Users/{{user}}/.ssh或者downloads）目录下根据密钥名称查找密钥文件，如果找到则直接使用该密钥打开新窗口并尝试连接远程实例，此时无需选择密钥。

图 6-129 远程连接 Notebook 实例

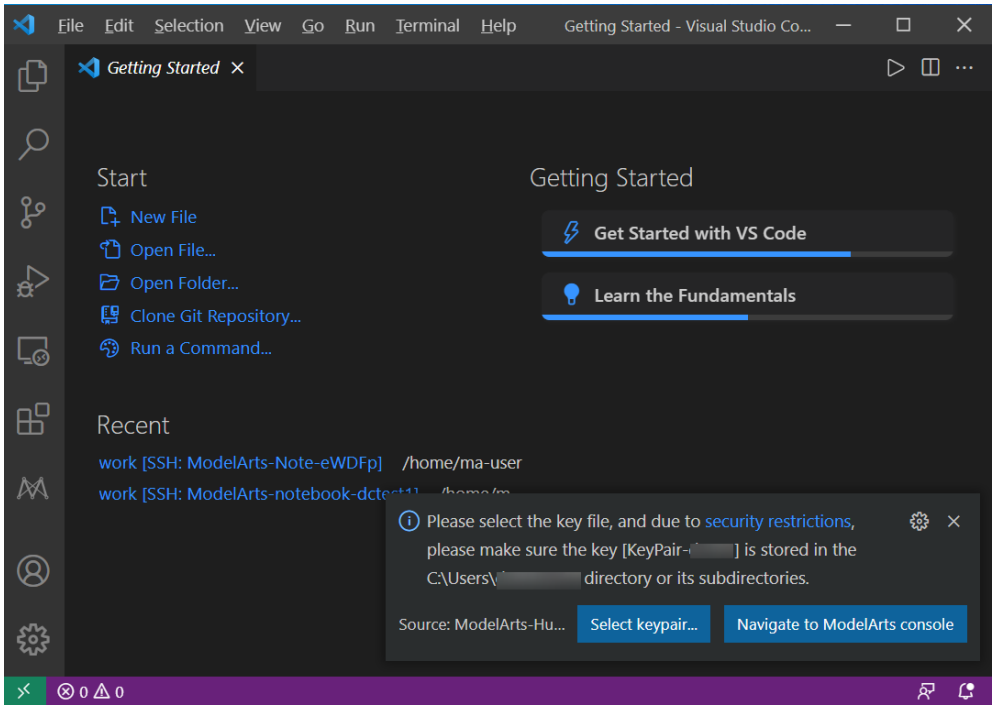


- 如果未找到会弹出选择框，请根据提示选择正确的密钥。

说明

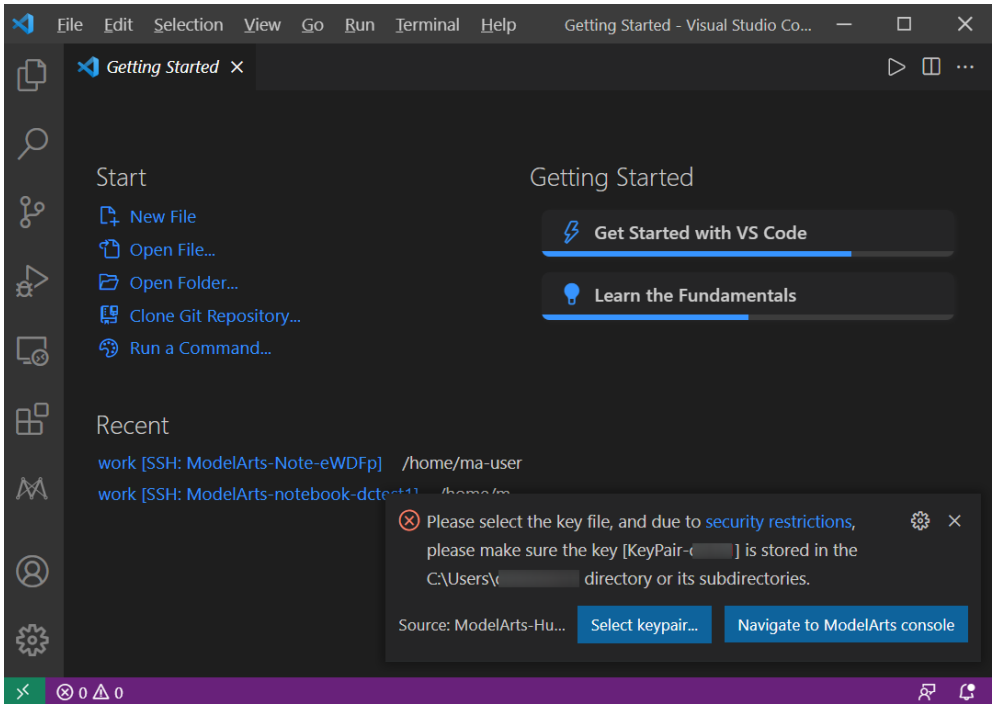
密钥文件名不能包含中文字符。

图 6-130 选择密钥文件



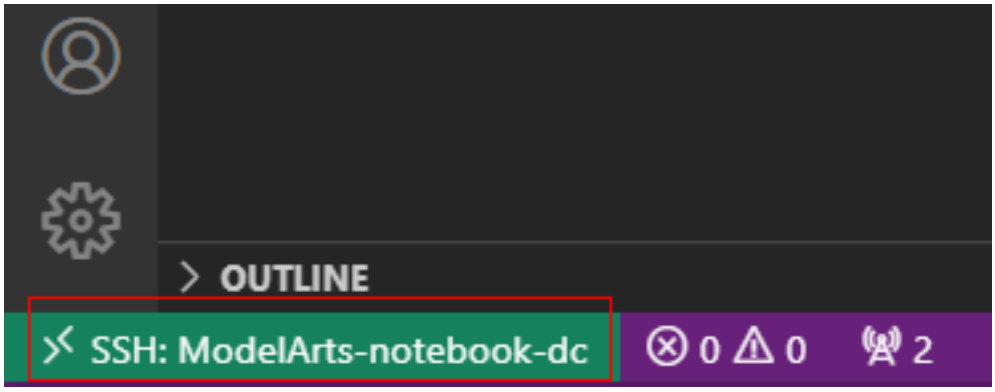
- 如果密钥选择错误，则弹出提示信息，请根据提示信息选择正确密钥。

图 6-131 选择正确的密钥文件



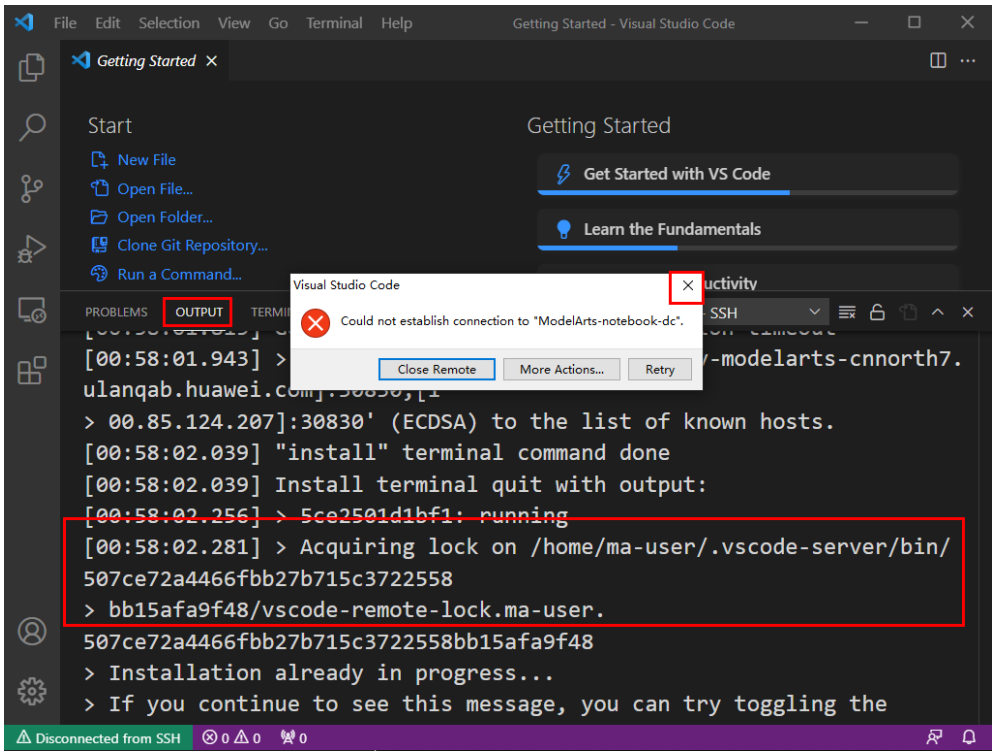
当左下角显示如下状态时，代表实例连接成功：

图 6-132 实例连接成功



当弹出如下错误时，代表实例连接失败，请关闭弹窗，并查看OUTPUT窗口的输出日志，请查看**FAQ**并排查失败原因。

图 6-133 实例连接失败



----结束

6.5.3.4 VS Code ToolKit 连接 Notebook

本节介绍如何在本地使用ModelArts提供的VS Code插件工具VS Code ToolKit，协助用户完成SSH远程连接Notebook。

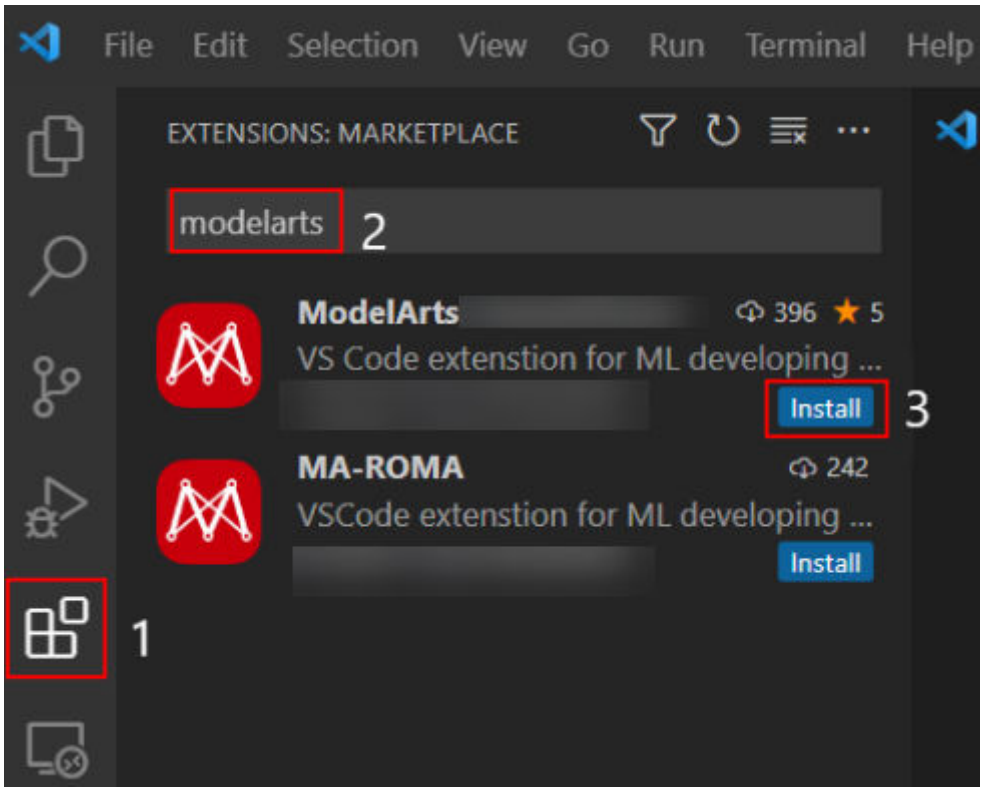
前提条件

已下载并安装VS Code。详细操作请参考**安装VS Code软件**。

Step1 安装 VS Code 插件

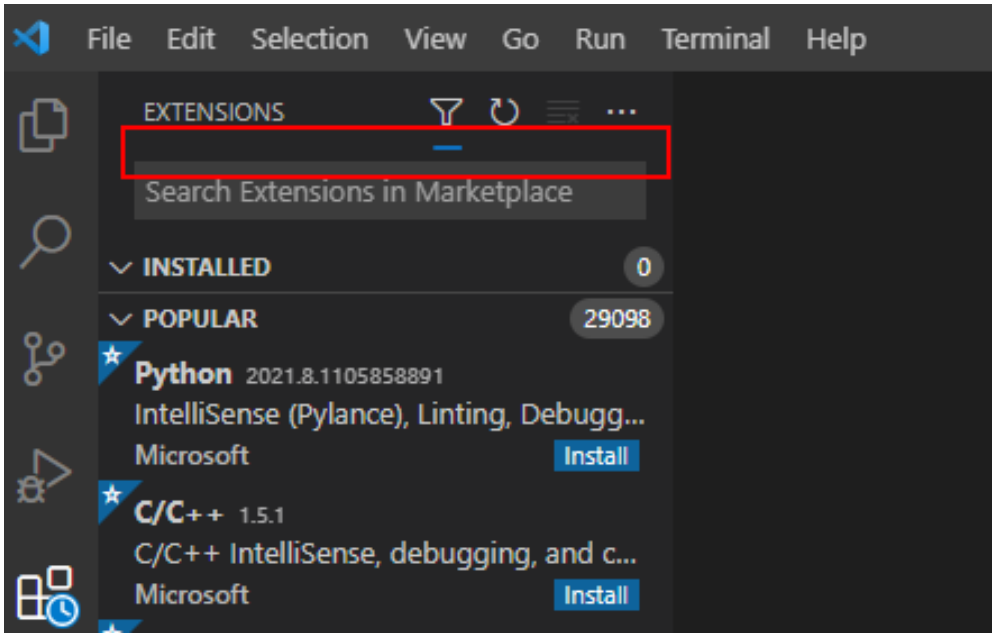
1. 在本地的VS Code开发环境中，如图6-134所示，在VS Code扩展中搜索“ModelArts”并单击“安装”。

图 6-134 安装 VS Code 插件



2. 安装过程预计1~2分钟，如图6-135所示，请耐心等待。

图 6-135 安装过程



3. 安装完成后，系统右下角提示安装完成，导航左侧出现ModelArts图标和SSH远程连接图标，表示VS Code插件安装完成。

图 6-136 安装完成提示

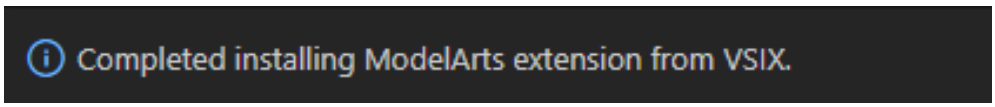
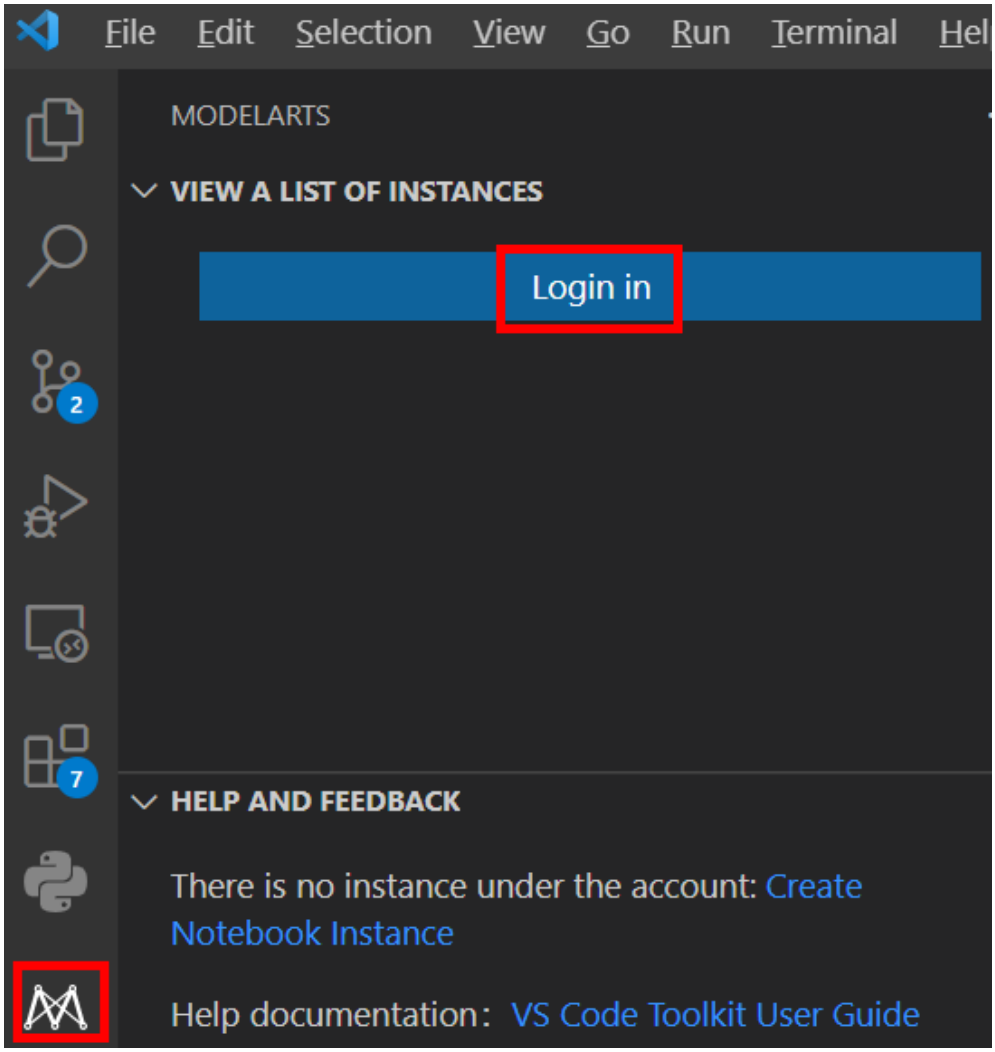
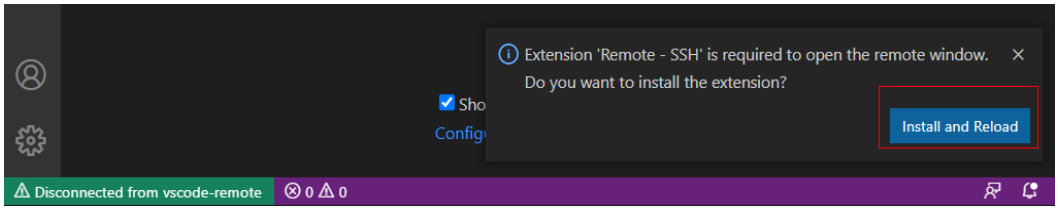


图 6-137 安装完成



当前网络不佳时SSH远程连接插件可能未安装成功，此时无需操作，在**Step5 连接Notebook实例**的**1**之后，会弹出如下图对话框，单击Install and Reload即可。

图 6-138 重新连接远程 SSH



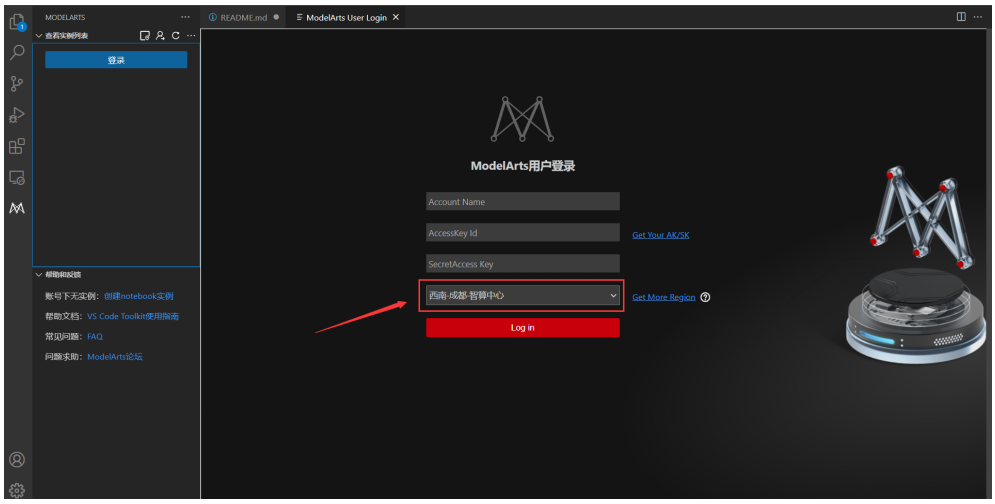
Step2 配置更多局点

1. 在VS Code插件中使用配置文件。

   联系局点运营公司获取YAML配置文件和host信息。单击左侧图标打开VS Code插件， 单击  ，选择“import Region Profile”，然后单击右下角的“选择本地文件（From local file）”，输入本地YAML文件路径后按回车。

2. 登录VS Code插件获得更多功能。

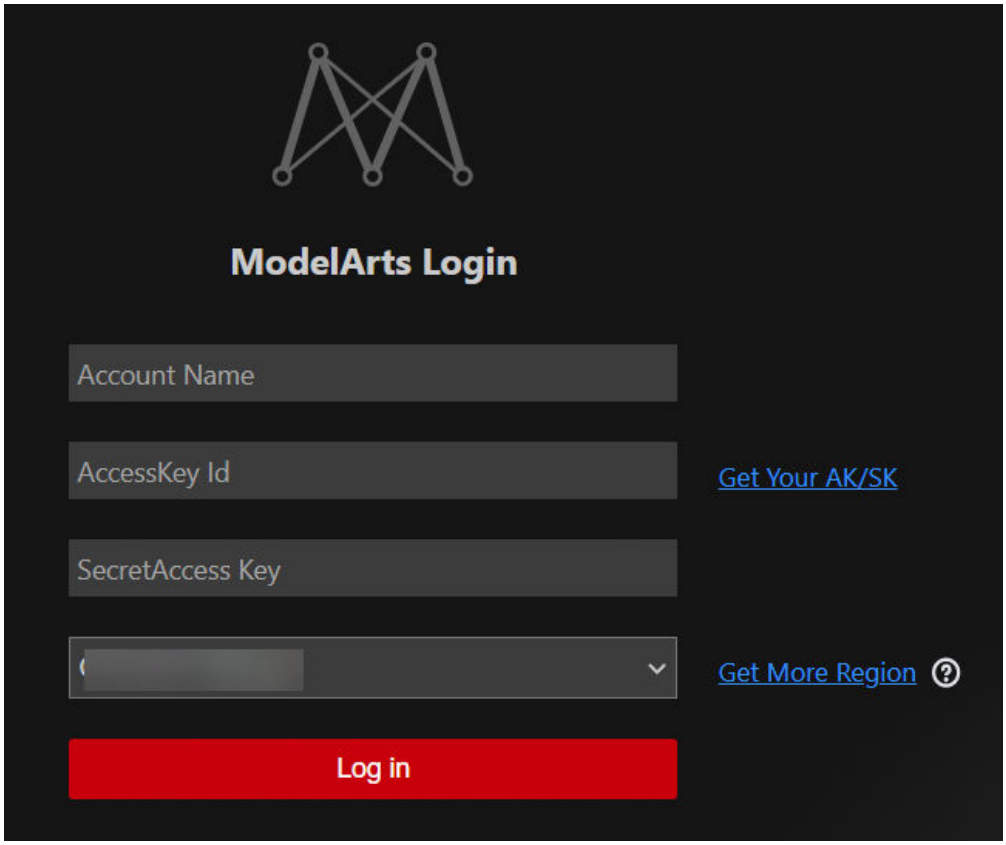
   配置文件导入成功后，region变为自己的局点，请输入Account Name、AK/SK完成登录操作。



Step3 登录 VS Code 插件

1. 在本地的VS Code开发环境中，单击ModelArts图标  ，单击“User Settings”，配置用户登录信息。

图 6-139 登录插件



输入如下用户登录信息，单击“登录”。

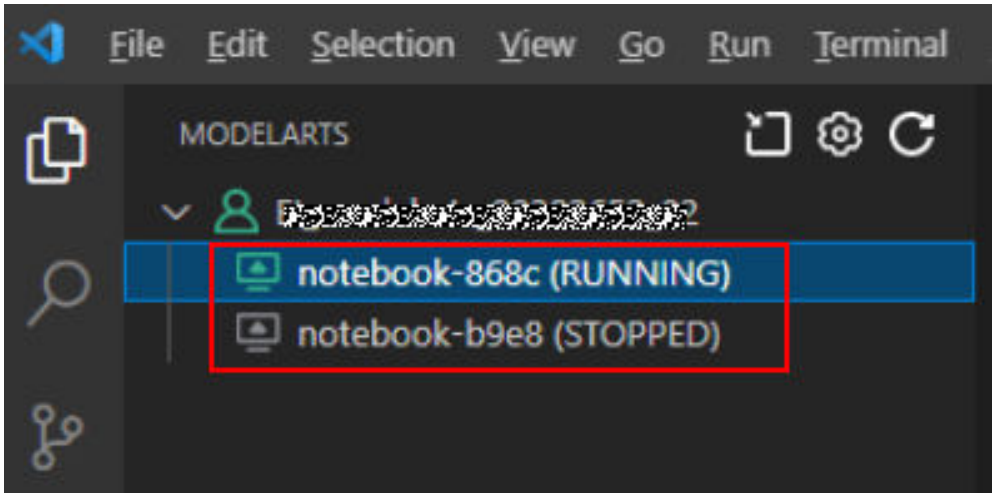
- Name：自定义用户名，仅用于VS Code页面展示，不与任何用户关联。
- AK、SK：在“账号中心 > 我的凭证 > 访问密钥”中创建访问密钥，获取AK、SK（**参考链接**）。
- 选择站点：此处的站点必须和远程连接的Notebook在同一个站点，否则会导致连接失败。

2. 登录成功后显示Notebook实例列表。

说明

此处仅显示ModelArts控制台default工作空间下的Notebook实例。

图 6-140 登录成功



Step4 创建 Notebook 实例

注意

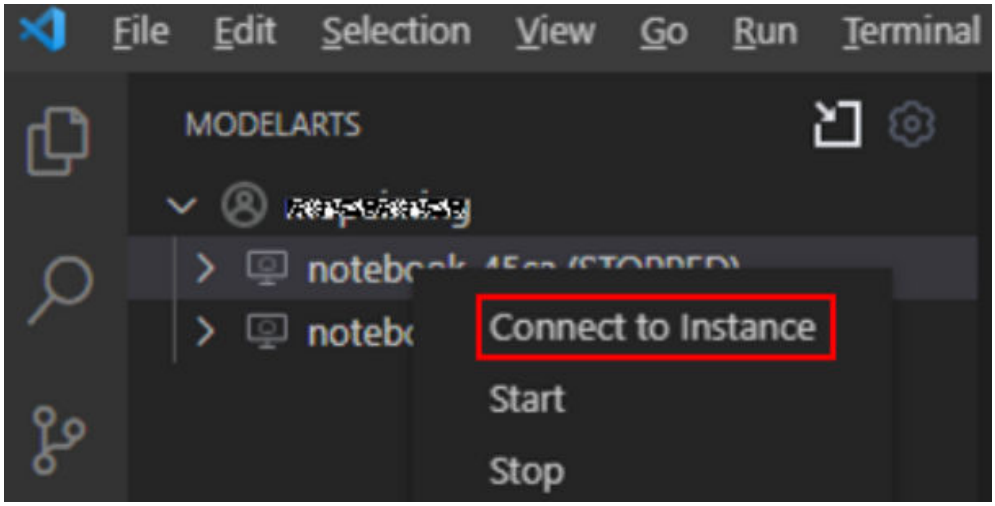
- 创建实例时，需开启“SSH远程开发”，并下载保存密钥对至本地如下目录。
  Windows：C:\Users\{{user}}
  macOS/Linux: Users/{{user}}
- 密钥对在用户第一次创建时自动下载，之后使用相同的密钥时不会再有下载界面（请妥善保管），或者每次都使用新的密钥对。

创建一个Notebook实例，并开启远程SSH开发，具体参见**创建Notebook实例**。

Step5 连接 Notebook 实例

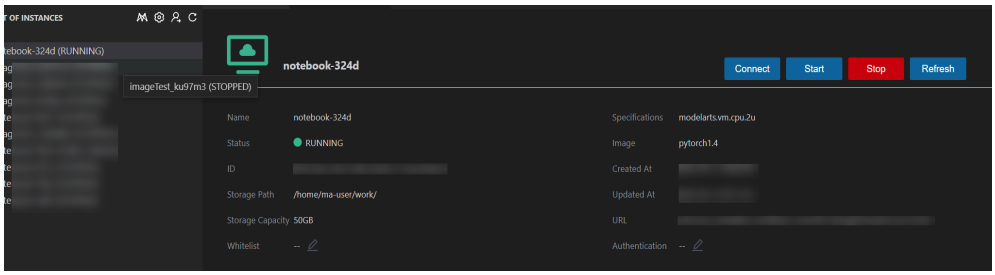
1. 在本地的VS Code开发环境中，右键单击实例名称，单击“Connect to Instance”，启动并连接Notebook实例。
   Notebook实例状态处于“运行中”或“停止”状态都可以，如果Notebook实例是停止状态，连接Notebook时，VS Code插件会先启动实例再去连接。

图 6-141 连接 Notebook 实例



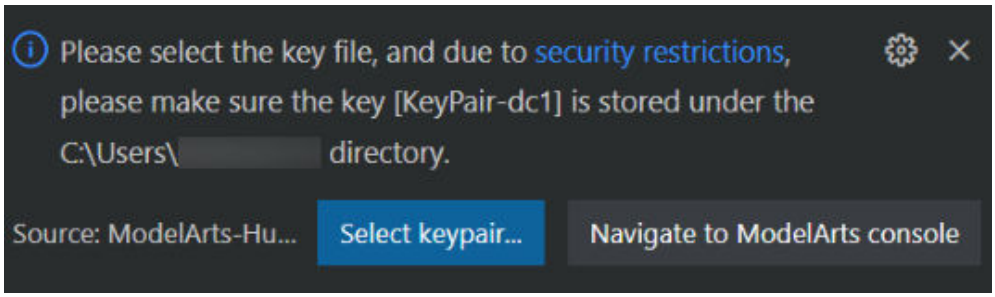
或者单击实例名称，在VS Code开发环境中显示Notebook实例详情页，单击“连接”，系统自动启动该Notebook实例并进行远程连接。

图 6-142 查看 Notebook 实例详情页



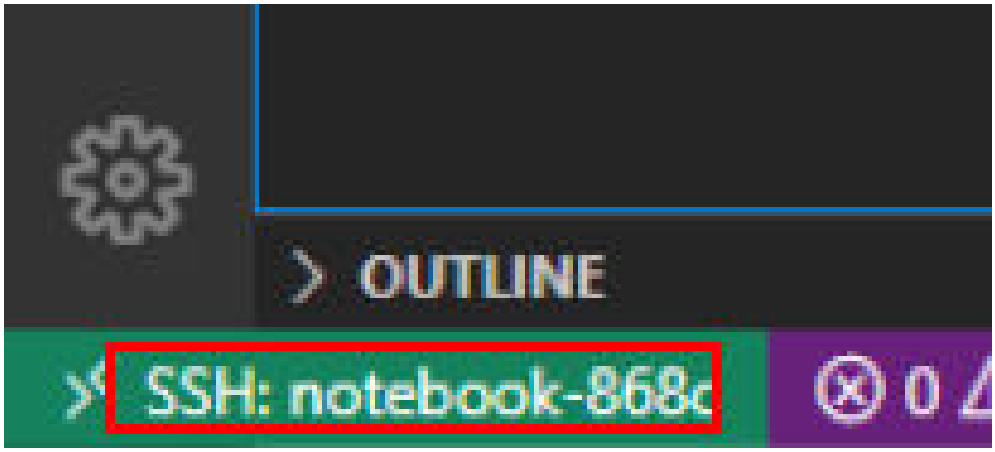
2. 第一次连接Notebook时，系统右下角会提示需要先配置密钥文件。选择本地密钥pem文件，根据系统提示单击“OK”。

图 6-143 配置密钥文件



3. 单击“确定”后，插件自动连接远端Notebook实例。首次连接大约耗时1~2分钟，取决于本地的网络情况。VS Code环境左下角显示类似下图即为连接成功。

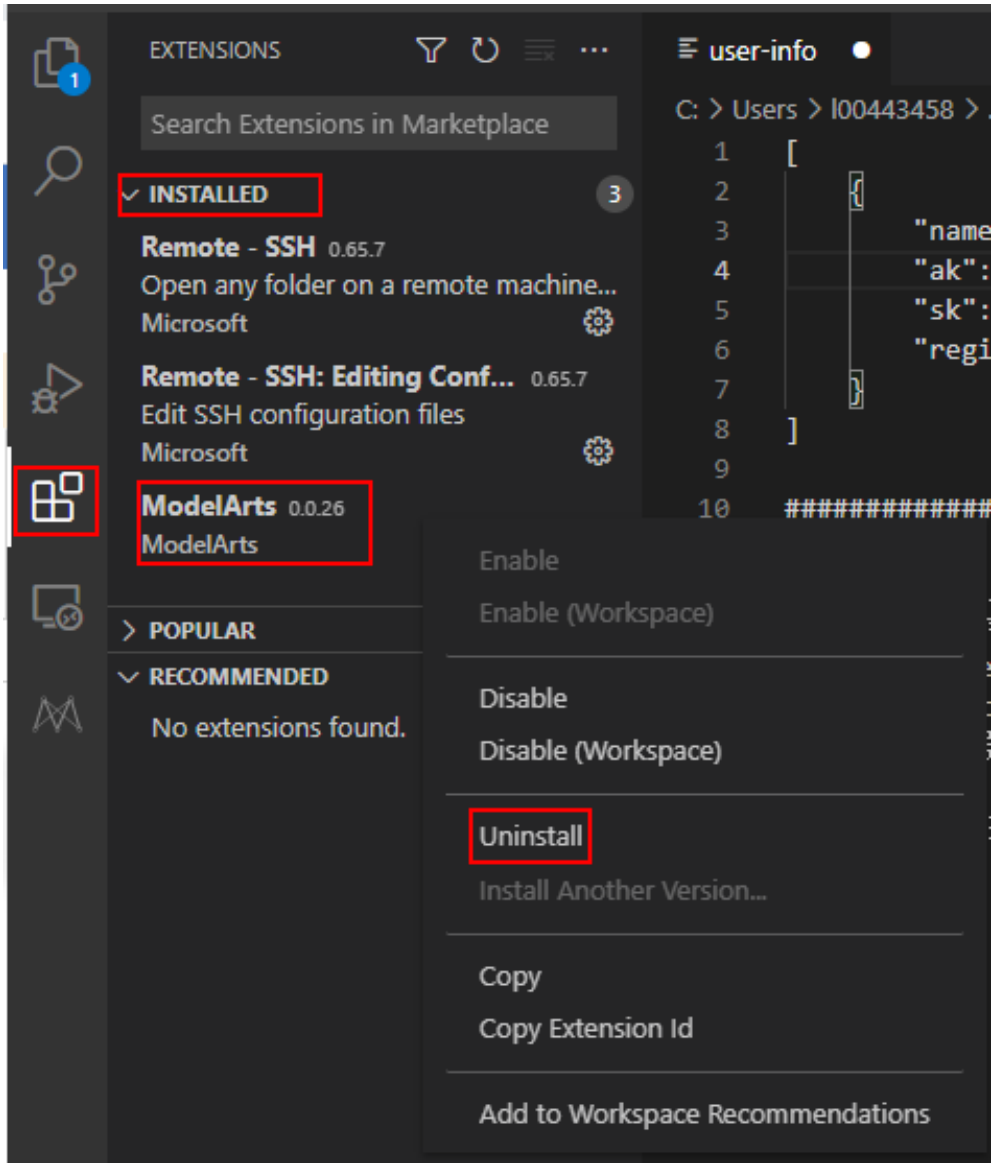
图 6-144 连接成功



相关操作

卸载VS Code插件操作如图6-145所示。

图 6-145 卸载 VS Code 插件



6.5.3.5 VS Code 手动连接 Notebook

本地IDE环境支持PyCharm和VS Code。通过简单配置，即可用本地IDE远程连接到ModelArts的Notebook开发环境中，调试和运行代码。

本章节介绍基于VS Code环境访问Notebook的方式。

前提条件

- 已下载并安装VS Code。详细操作请参考**安装VS Code软件**。
- 用户本地PC或服务器的操作系统中建议先安装Python环境，详见**VSCode官方指导**。
- 创建一个Notebook实例，并开启远程SSH开发。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。
- 在Notebook实例详情页面获取开发环境访问地址和端口号。

图 6-146 Notebook 实例详情页面

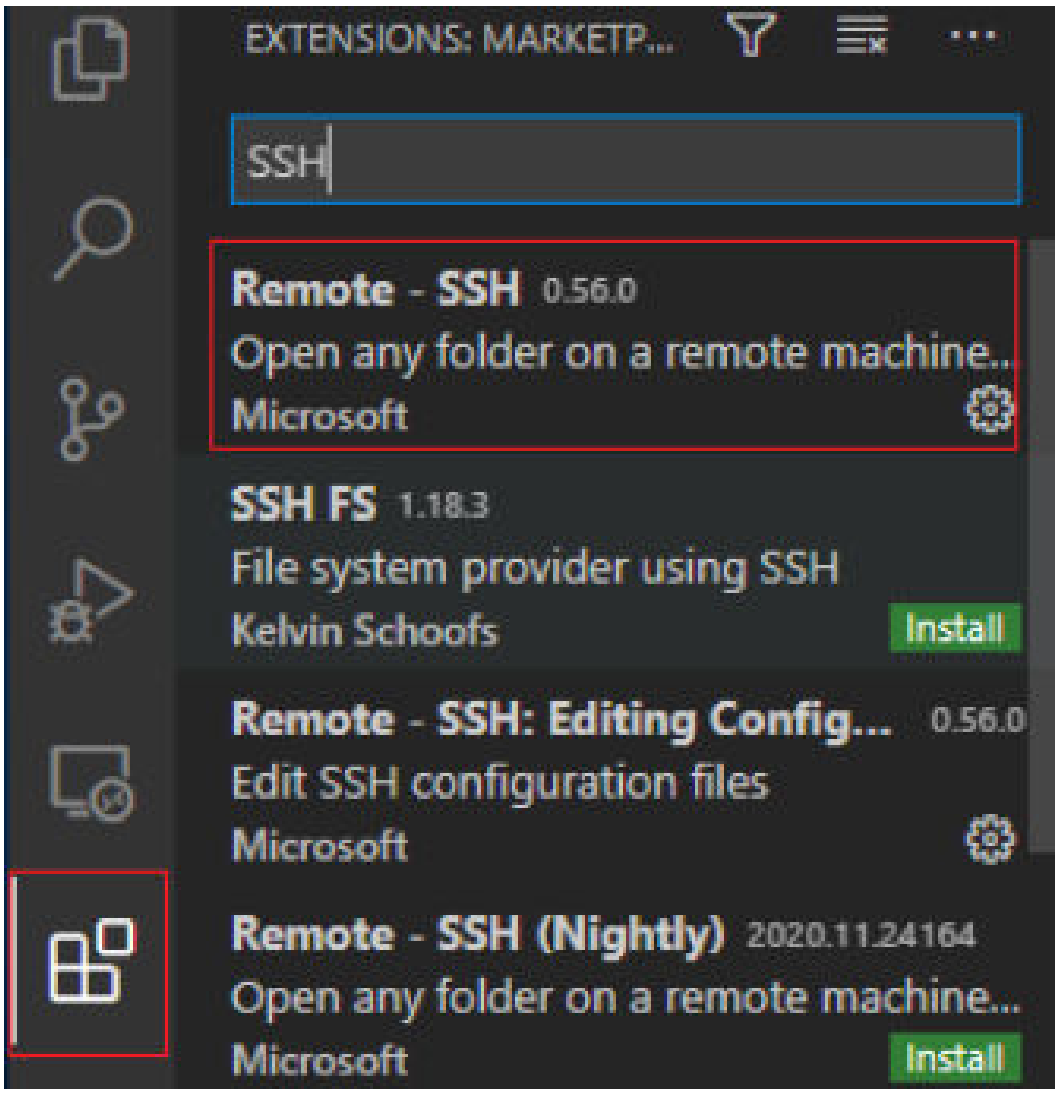
- 准备好密钥对。

  密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。

Step1 添加 Remote-SSH 插件

在本地的VS Code开发环境中，单击左侧列表的Extensions图标选项  ，在搜索框中输入SSH，单击Remote-SSH插件的install按钮，完成插件安装。

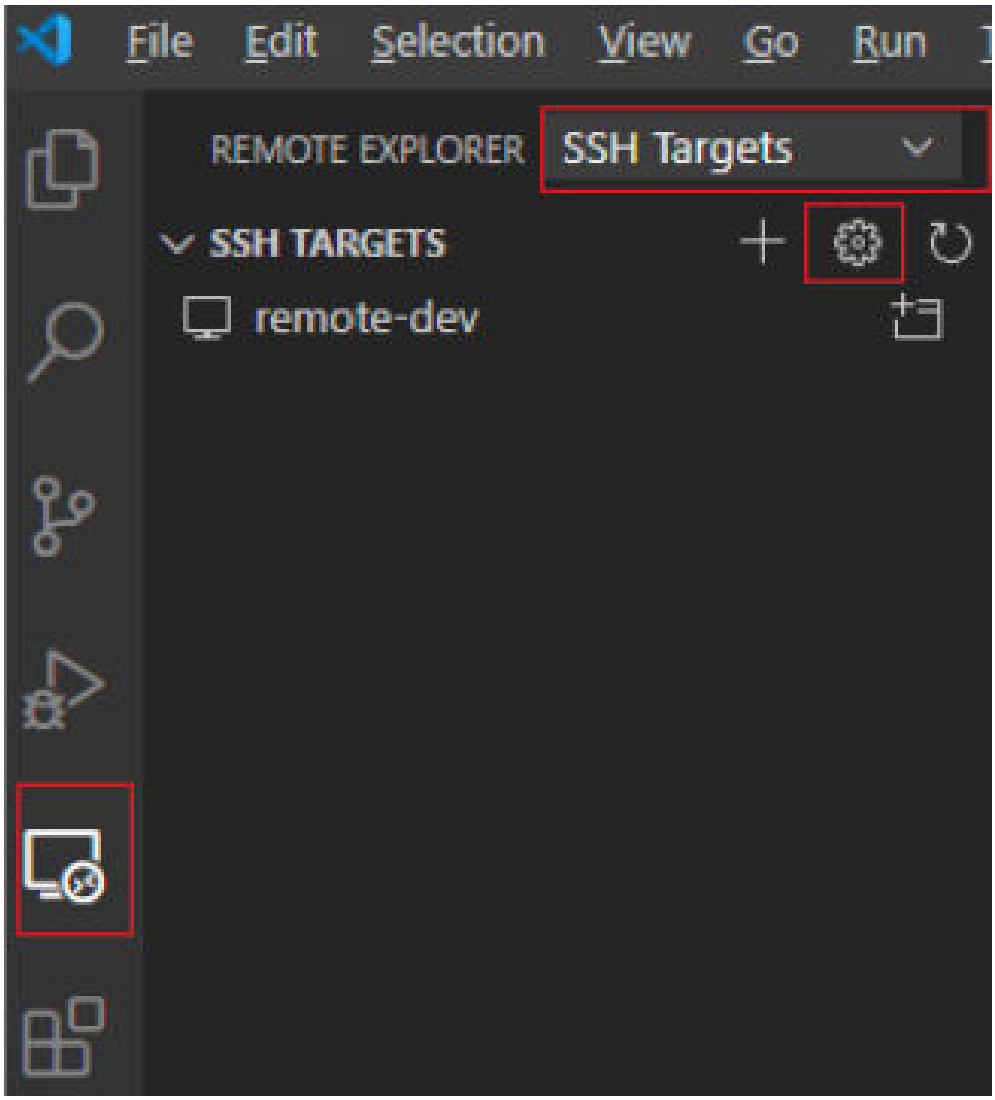
图 6-147 添加 Remote-SSH 插件



Step2 配置 SSH

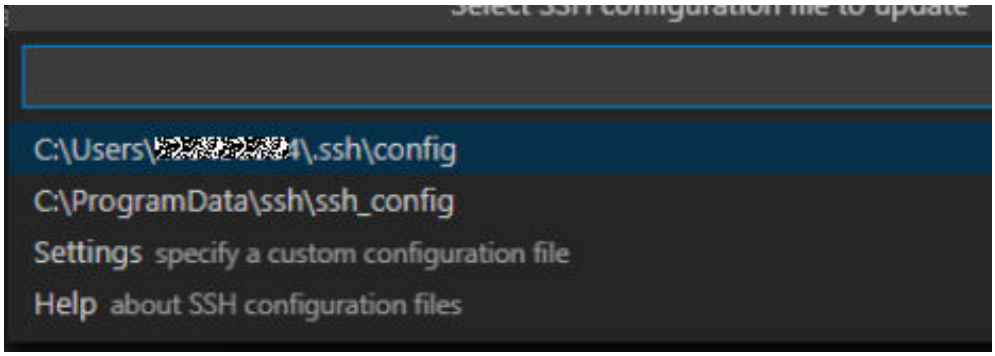
1. 在本地的VS Code开发环境中，单击左侧Remote Explorer按钮，在上方的下拉列表中选择“SSH Target”，再单击页面上的设置按钮，此时会出现SSH配置文件路径。

   图 6-148 配置 SSH Targets 页面

2. 单击列表中出现的SSH路径按钮，打开config文件，进行配置。

图 6-149 配置 SSH Config 文件

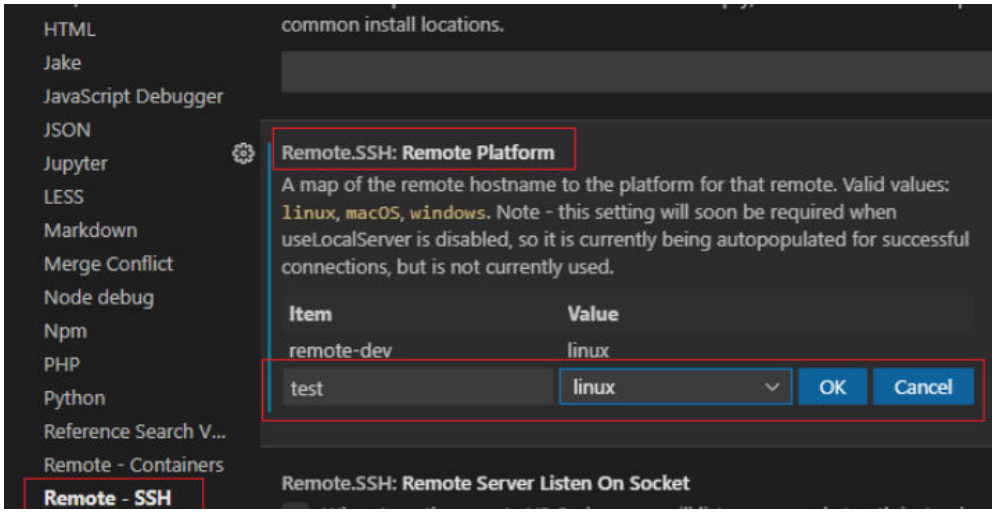



```
HOST remote-dev
    hostname <instance connection host>
    port <instance connection port>
    user ma-user
    IdentityFile ~/.ssh/test.pem
    UserKnownHostsFile=/dev/null
    StrictHostKeyChecking no
```

- Host: 自定义设置的云上开发环境名称。
- HostName: 云上开发环境的访问地址，即在开发环境实例页面远程访问模块获取的访问地址。
- Port: 云上开发环境的端口，即在开发环境实例页面远程访问模块获取的端口号。
- User：登录用户只支持ma-user进行登录。
- IdentityFile: 存放在本地的云上开发环境私钥文件，即前提条件**准备好密钥对**中准备的密钥对。

3. 配置云上开发环境系统平台，单击“File > Preference > Settings > Extensions > Remote-SSH”，在“Remote Platform”中，单击“Add Item”选项，设置“Item”和“Value”，配置完成后，单击“OK”。

图 6-150 配置云上开发环境系统平台

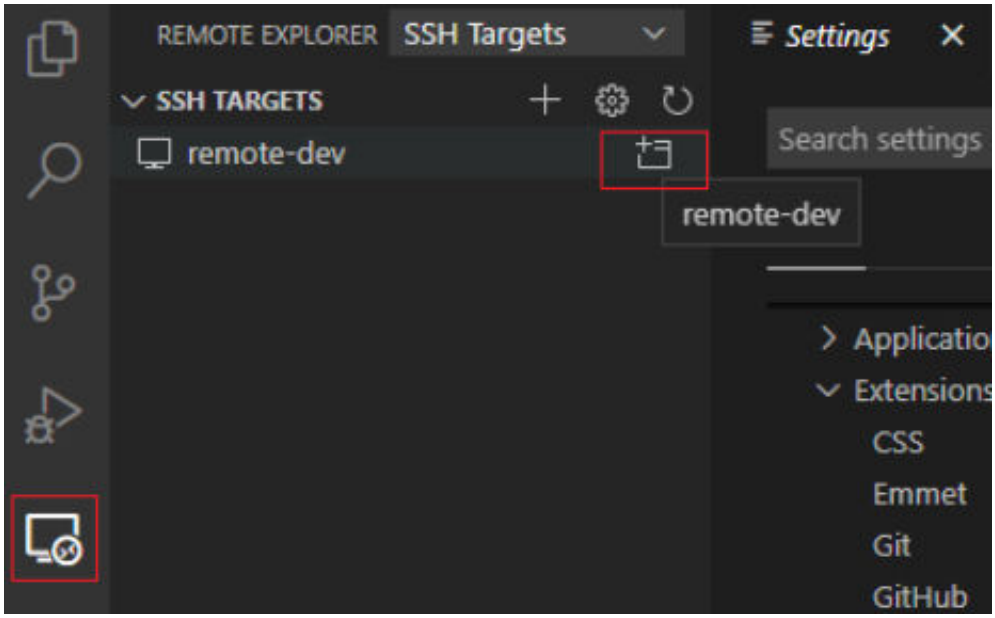


“Item”： 在SSH Config中配置的Host的名称。

“Value”： 在下拉选择框中选择远端开发环境平台。

4. 再回到SSH Targets页面，单击右侧的Connect to Host in New Window按钮 ，该按钮会显示远程开发环境名称，选中并打开。

图 6-151 打开开发环境



在新打开的页面中，看到下图所示界面，即表示连接成功。

图 6-152 开发环境远程连接成功

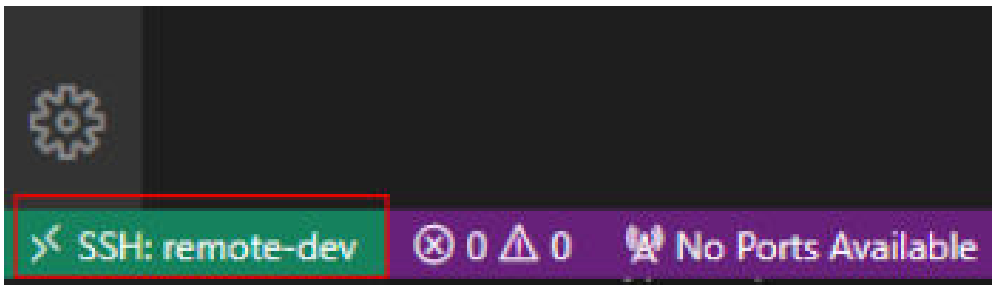
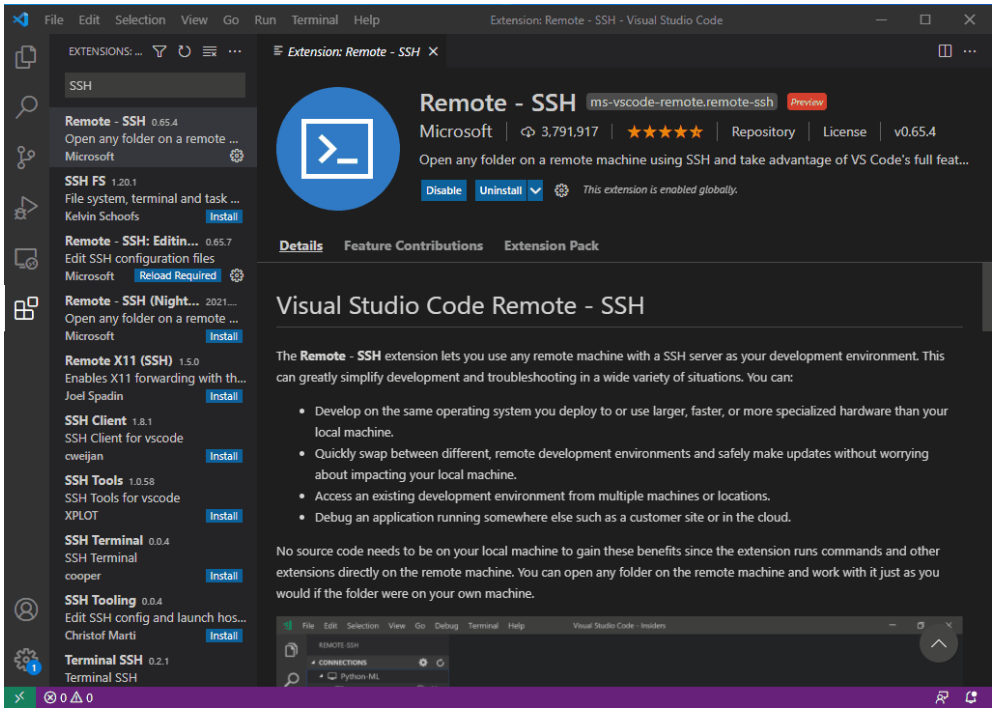


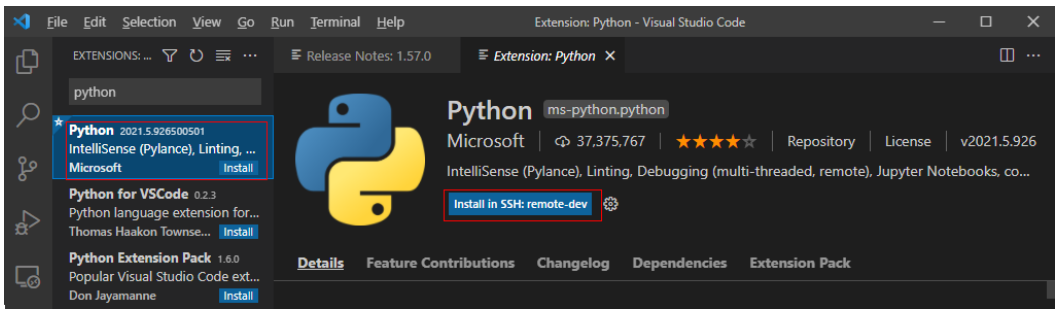
图 6-153 完整配置示例



Step3 安装云端 Python 插件

在新打开的VS Code界面，单击左侧列表的Extensions选项  ，在搜索框中输入Python，在下拉列表中单击“Install”进行安装。

图 6-154 安装云端 Python 插件



如果安装云端的Python插件不成功时，建议通过离线包的方式安装。

Step4 云上环境依赖库安装

在进入容器环境后，可以使用不同的虚拟环境，例如TensorFlow、PyTorch等，但是实际开发中，通常还需要安装其他依赖包，此时可以通过Terminal连接到环境里操作。

1. 在VS Code环境中，执行Ctrl+Shift+P。
2. 搜Python: Select Interpreter，选择对应的Python环境。
3. 单击页面上方的“Terminal > New Terminal”，此时打开的命令行界面即为远端容器环境命令行。

4. 进入引擎后，通过执行如下命令安装依赖包。
```
pip install spacy
```

6.5.3.6 在 VS Code 中远程调试代码

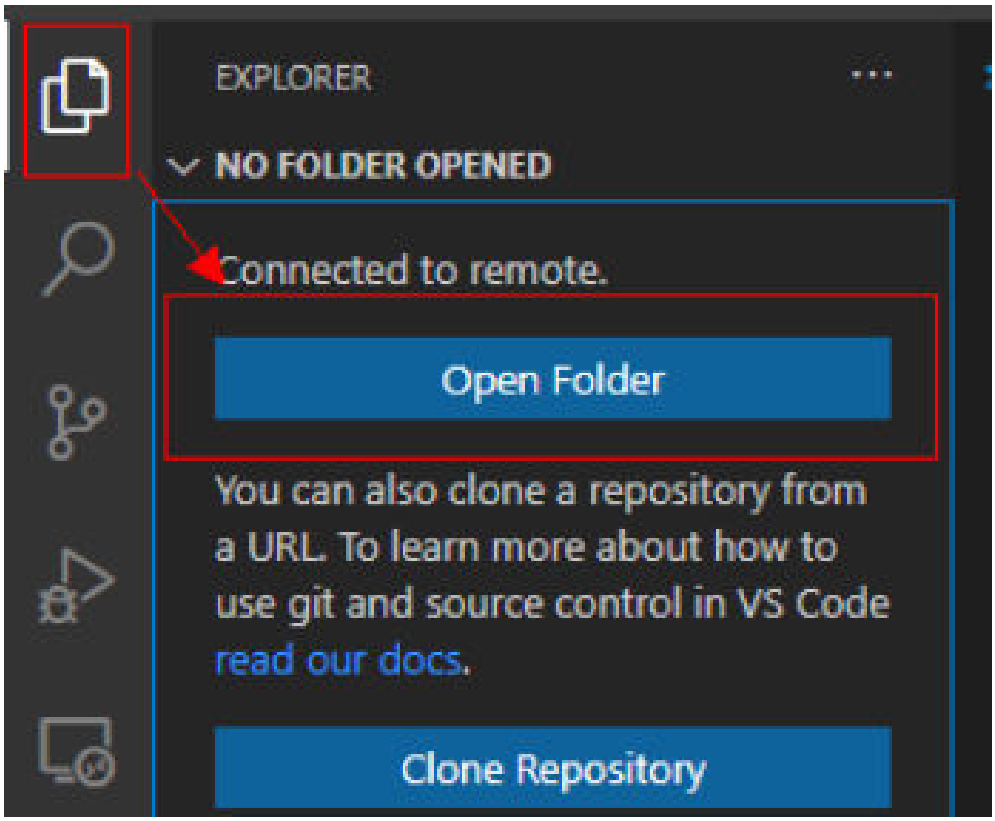
前提条件

VS Code已连接到Notebook。

Step1 上传本地代码到云端开发环境

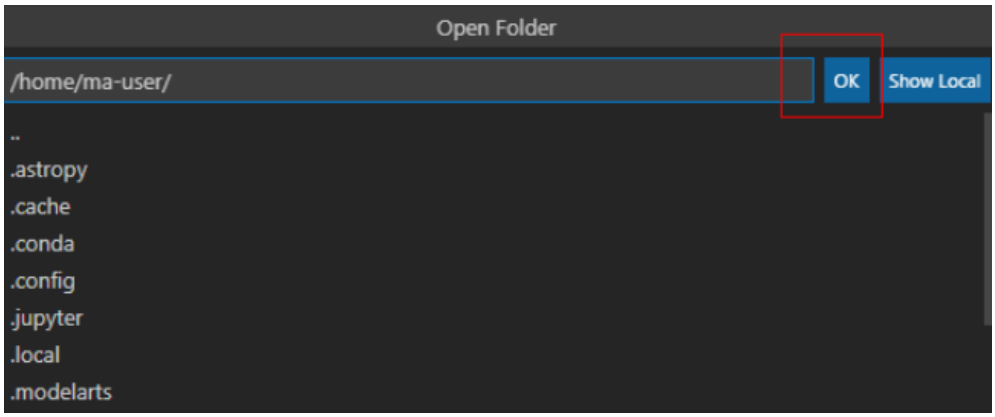
1. 在VS Code界面，单击“File > OpenFolder”打开云端路径。

图 6-155 Open Folder



2. 选择要打开的路径，单击“OK”。

图 6-156 选择文件路径

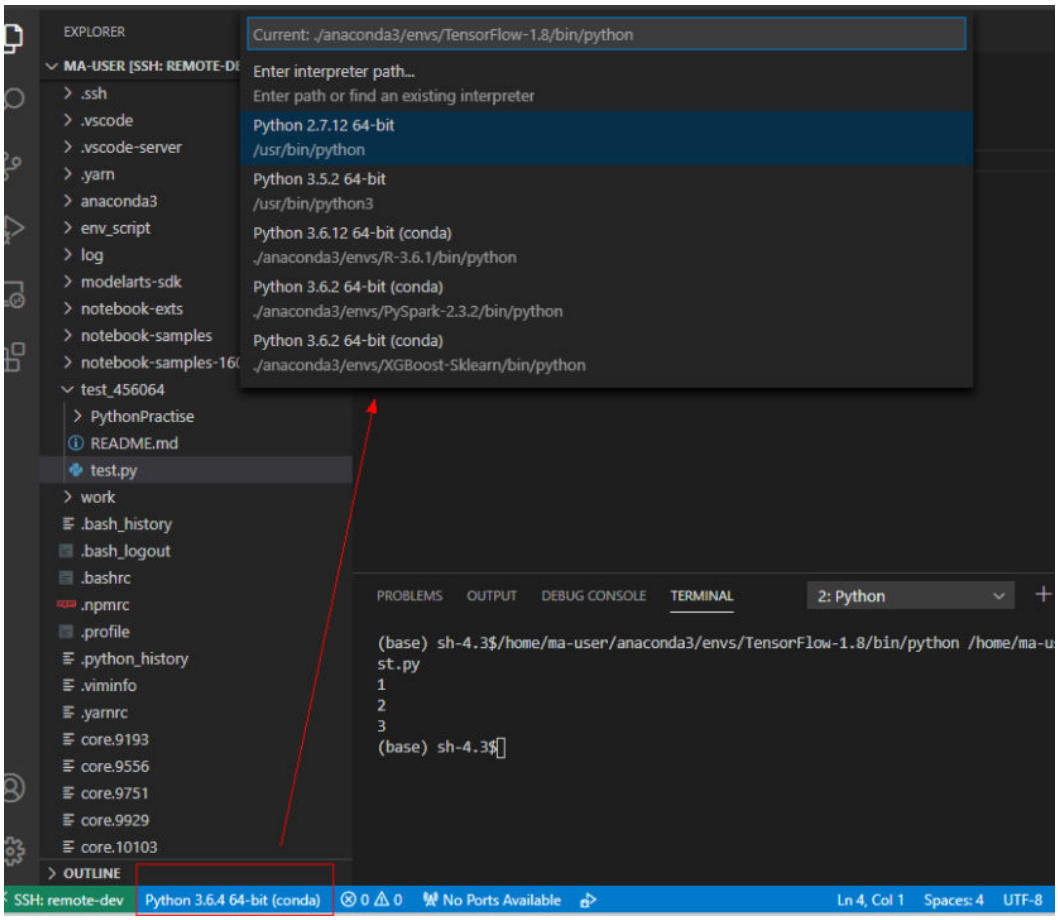


3. 此时，会在IDE左侧出现该开发环境下的目录结构，把想要上传的代码及其他文件直接拖拽至对应的文件夹内即完成本地代码上传至云端。

Step2 远程调试代码

在VS Code中打开要执行的代码文件，在执行代码之前需要选择合适的Python版本路径，单击下方默认的Python版本路径，此时在上方会出现该远程环境上所有的python版本，选择自己需要的版本即可。

图 6-157 选择 Python 版本



- 对于打开的代码文件，单击run按钮，即可执行，可以在下方的Terminal中看到代码输出信息。
- 如果执行较长时间的训练任务，建议使用nohup命令后台运行，否则SSH窗口关闭或者网络断连会影响正在运行的训练任务，命令参考：
  ```
  nohup your_train_job.sh > output.log 2>&1  &  tail -f  output.log
  ```
- 如果要对代码进行debug调试，步骤如下：
  a. 单击左侧“Run > Run and Debug”。
  b. 选择当前打开的默认的python代码文件进行调试。
  c. 对当前代码进行打断点，即在代码左侧进行单击，就会出现小红点。
  d. 此时，即可按照正常的代码调试步骤对代码调试，在界面左边会显示debug信息，代码上方有相应的调试步骤。

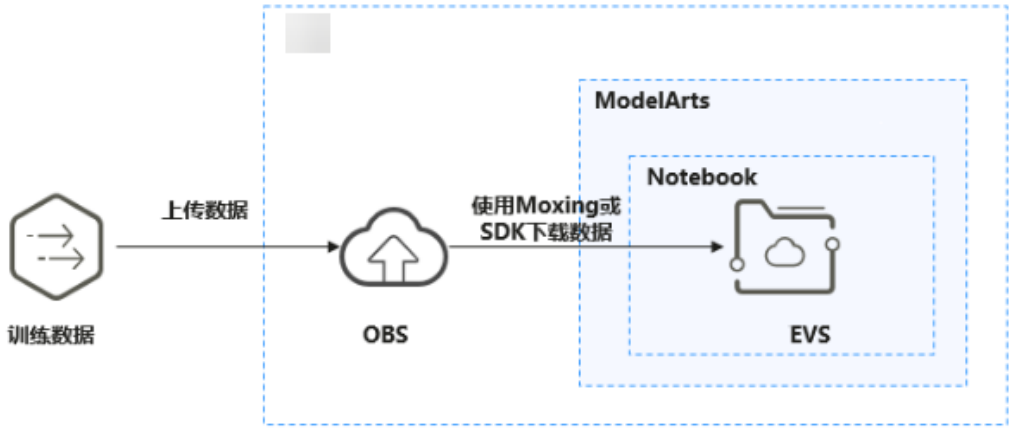
6.5.3.7 在 VS Code 中上传下载文件

在本地 IDE 中上传数据至 Notebook

不大于500MB数据量，直接拷贝至本地IDE中即可。

大于500MB数据量，请先上传到OBS中，再从OBS上传到云上开发环境。

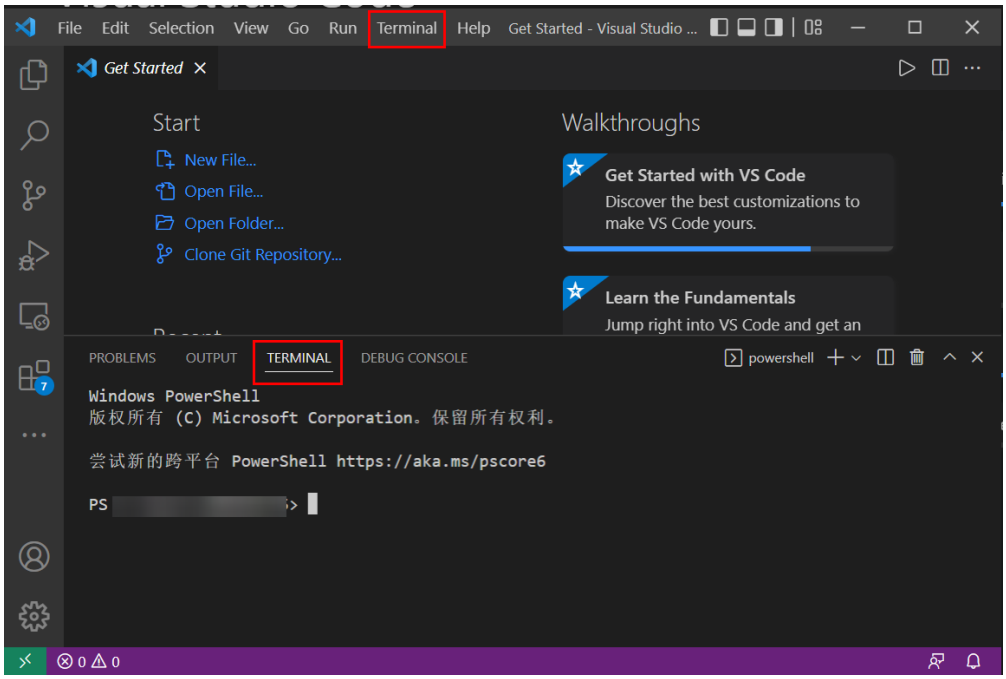
图 6-158 数据通过 OBS 中转上传到 Notebook



操作步骤

1. 上传数据至OBS。或者在本地VS Code的Terminal中使用ModelArts SDK完成数据上传至OBS。
   首先在本地VS Code环境中开启Terminal。

图 6-159 本地 VS Code 环境开启 Terminal



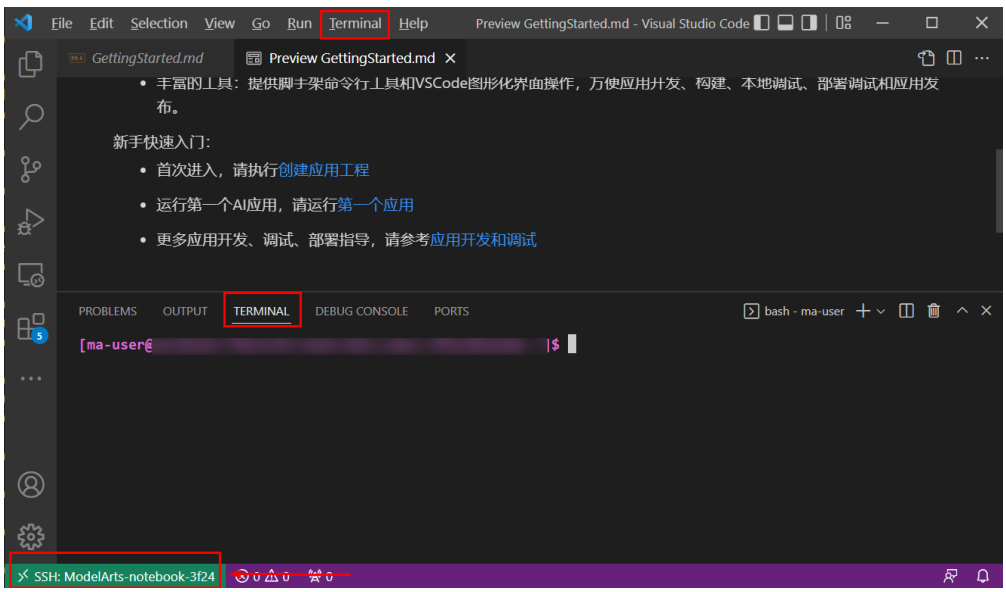
输入python并回车，进入python环境。

```
python
```

在本地VS Code的Terminal中使用ModelArts SDK上传本地文件至OBS，详情请参考**SDK参考**的“OBS管理> 文件传输（推荐）”章节进行OBS传输操作。

2. 在远程连接VS Code的Terminal中使用ModelArts SDK下载OBS文件到开发环境的操作示例如下：

图 6-160 远程连接 VS Code 环境开启 Terminal




```
#手动source进入开发环境
cat /home/ma-user/README
#然后选择要source的环境
```

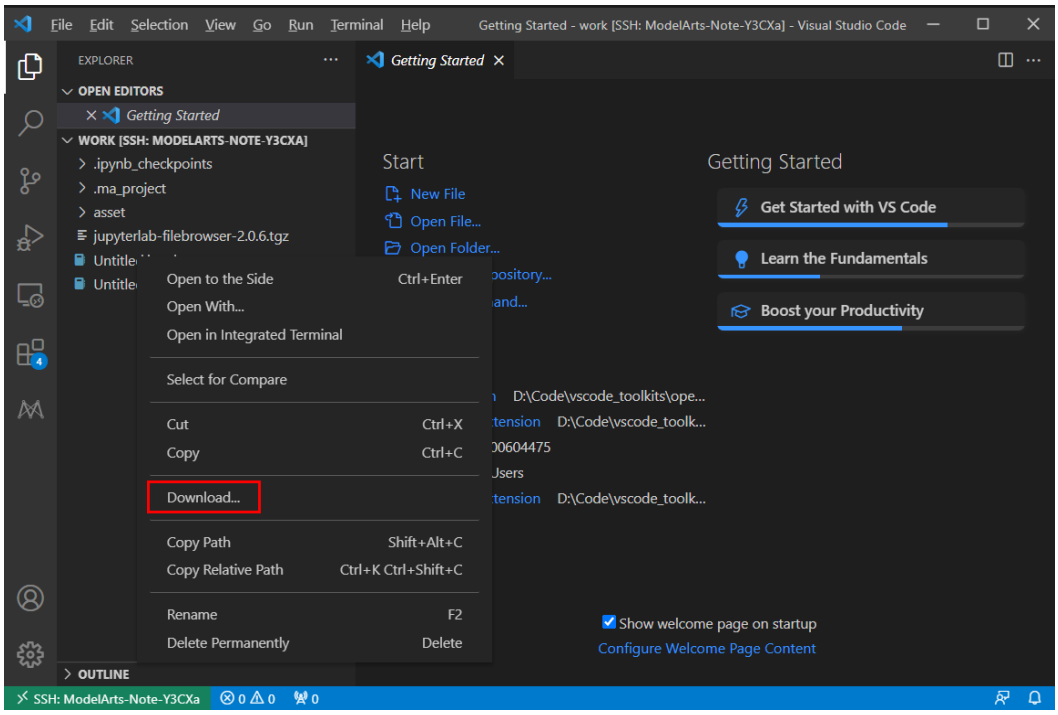
```
source /home/ma-user/miniconda3/bin/activate MindSpore-python3.7-aarch64
#输入python并回车，进入python环境
python
```

然后参考**SDK参考**的“OBS管理> 文件传输（推荐）”章节进行OBS传输操作。

下载 Notebook 中的文件至本地

在Notebook中开发的文件，可以下载至本地。在本地IDE的Project目录下的Notebook2.0工程单击右键，单击“Download...”将文件下载到本地。

图 6-161 VS Code 环境下载 Notebook 中的文件至本地



6.5.4 本地 IDE（SSH 工具连接）

本节操作介绍在Windows环境中使用PuTTY SSH远程登录云上Notebook实例的操作步骤。

前提条件

- 创建一个Notebook实例，并开启远程SSH开发，配置远程访问IP白名单。该实例状态必须处于“运行中”，具体参见**创建Notebook实例**章节。
- 在Notebook实例详情页面获取开发环境访问地址和端口号。

  图 6-162 Notebook 实例详情页面

- 准备好密钥对文件。

密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。

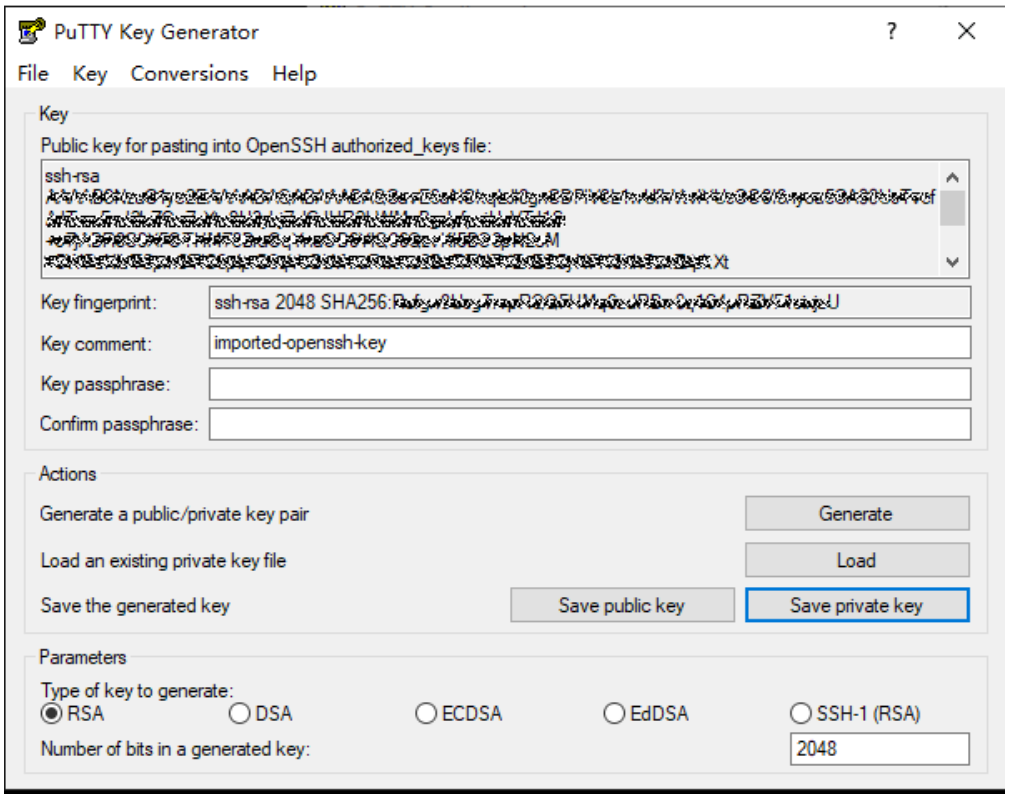
Step1 安装 SSH 工具

下载并安装SSH远程连接工具，以PuTTY为例，**下载链接**。

Step2 使用 puttygen 将密钥对.pem 文件转成.ppk 文件

1. **下载puttygen**，并双击运行puttygen。
2. 单击“Load”，上传.pem密钥（即在创建Notebook实例时创建并保存的密钥对文件）。
3. 单击“Save private key”，保存生成的.ppk文件。.ppk文件的名字可以自定义，例如key.ppk。

图 6-163 将密钥对.pem 文件转成.ppk 文件

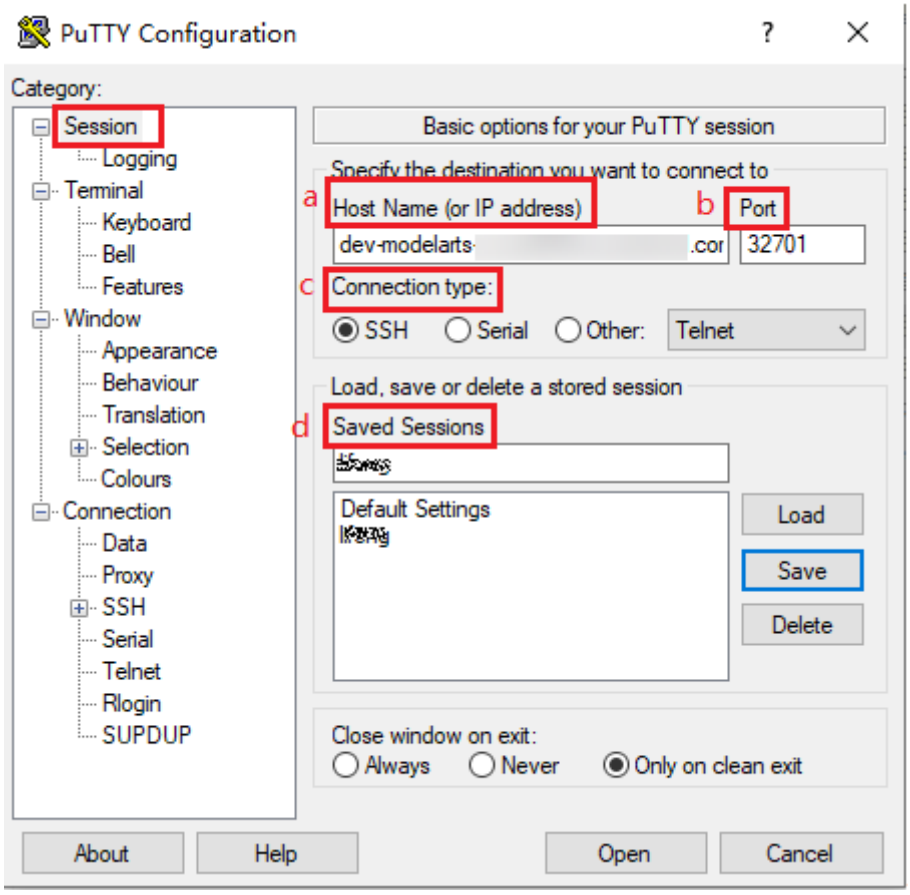


Step3 使用 SSH 工具连接云上 Notebook 实例

1. 运行PuTTY。
2. 单击“Session”，填写以下参数。
   a. Host Name (or IP address)：云上开发环境Notebook实例的访问地址，即在Notebook实例详情页获取的地址。
   b. Port：云上Notebook实例的端口，即在Notebook实例详情页获取的端口号。例如：32701。
   c. Connection Type：选择SSH。

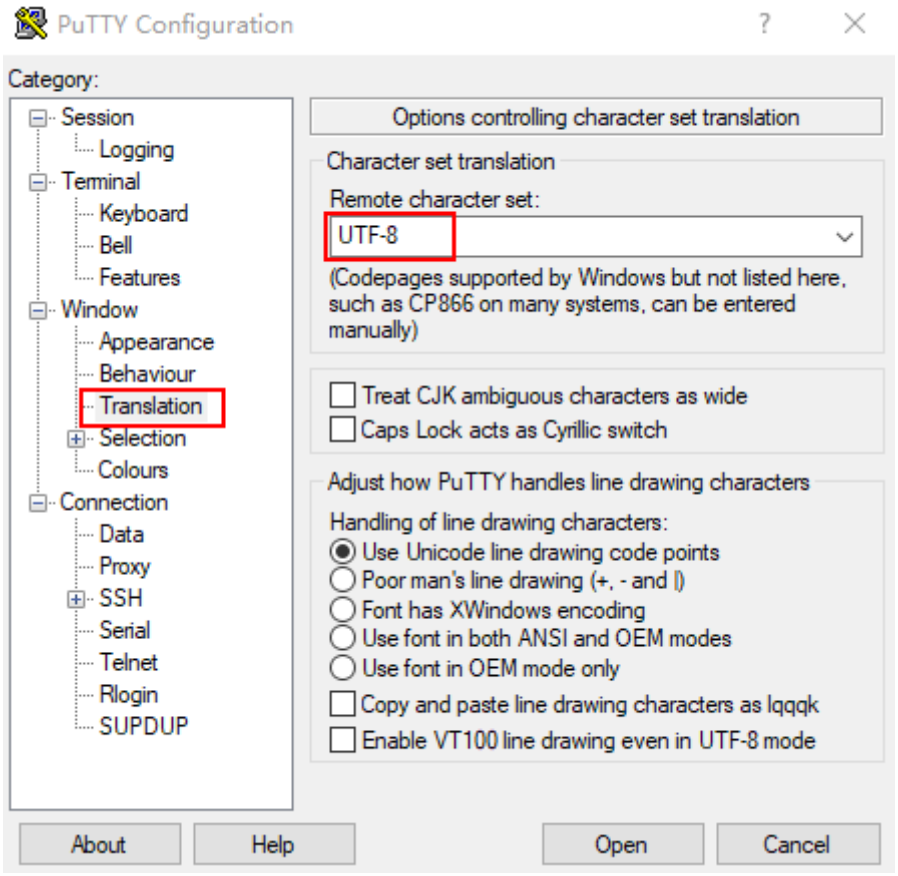
d. Saved Sessions：任务名称，在下一次使用PuTTY时就可以单击保存的任务名称，即可打开远程连接。

图 6-164 设置 Session



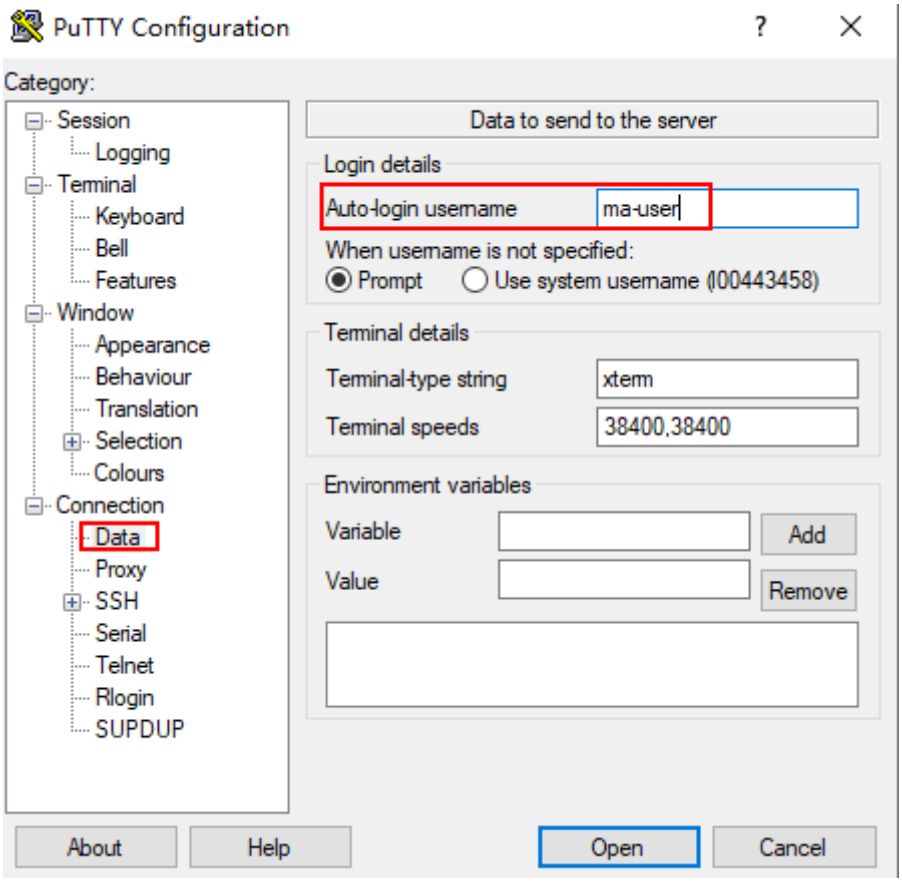
3. 选择“Window > Translation”，在“Remote character set:”中选择“UTF-8”。

图 6-165 设置字符格式

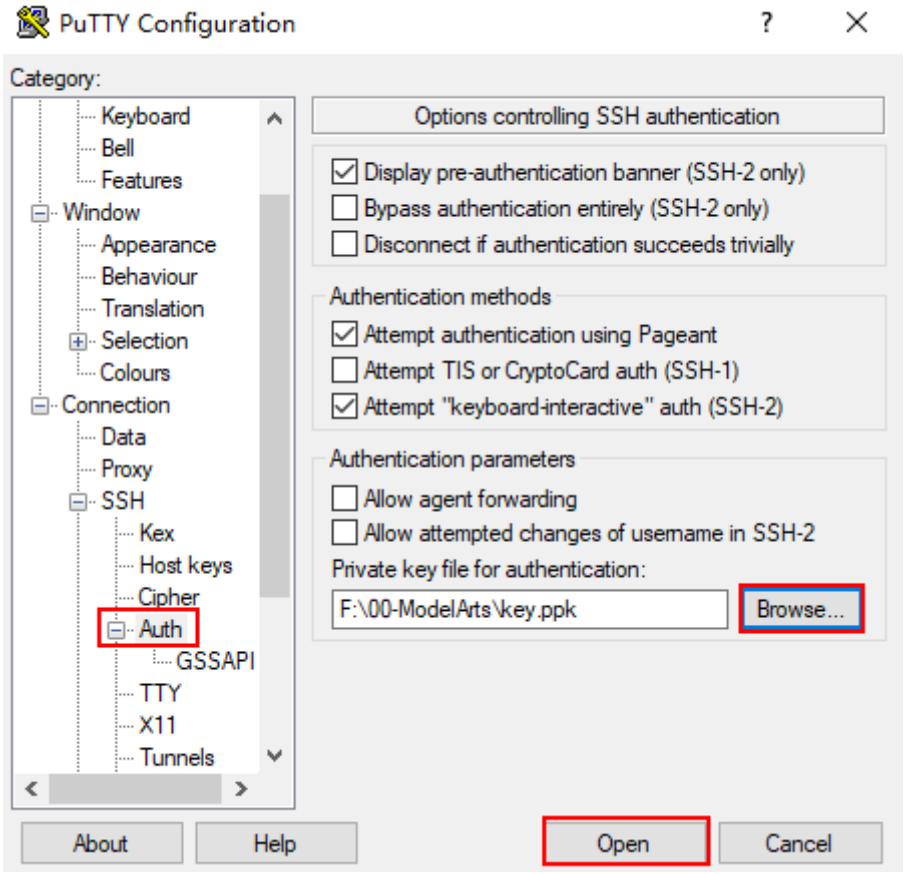


4. 选择“Connection > Data”， 在“Auto-login username” 中填写用户名“ma-user”。

图 6-166 填写用户名

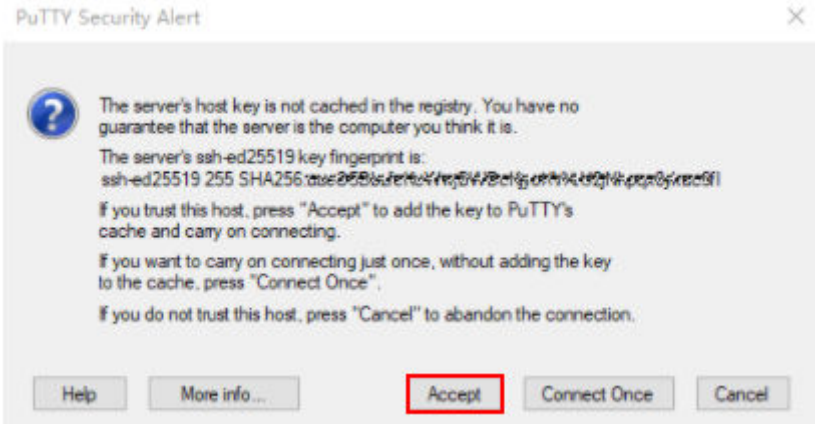


5. 选择“Connection > SSH > Auth”，单击“Browse”，选择“.ppk文件”（由**Step2**密钥对.pem文件生成）。



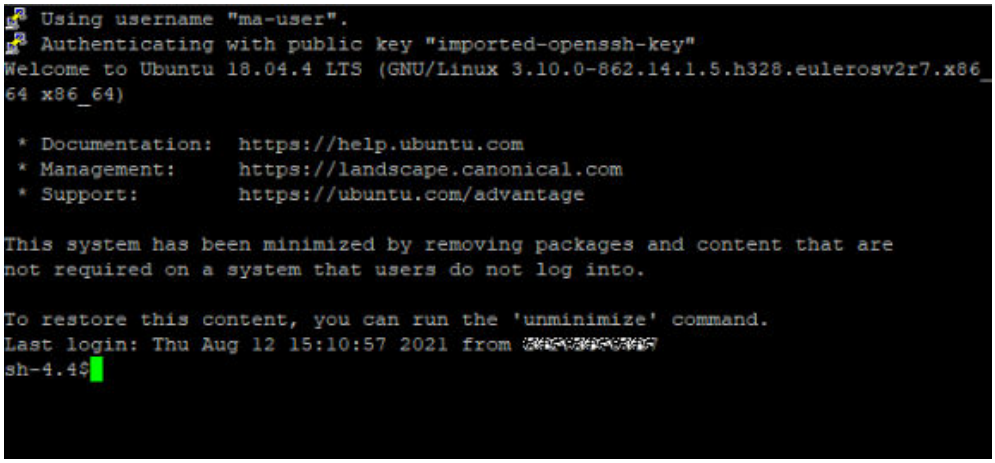
6. 单击“Open”。如果首次登录，PuTTY会显示安全警告对话框，询问是否接受服务器的安全证书。单击“Accept”将证书保存到本地注册表中。

图 6-167 询问是否接受服务器的安全证书



7. 成功连接到云上Notebook实例。

图 6-168 连接到云上 Notebook 实例



6.6 使用 Notebook 开发 Ascend 算子

概述

训练、推理场景下，使用第三方框架时遇到不支持的算子，需要自己开发；网络调优时，发现一些算子组合性能较低，需重新开发高性能算子替换低性能的算子，此时可以通过VS Code一键连接云上Notebook，使用云上资源，在VS Code端进行算子开发、调试等。Notebook中已经配置好环境，您无需进行CANN软件安装和环境变量配置就可以进行工程化开发。

说明

本文档提供了一套算子工程样例代码，您可以直接使用。如果需要了解Ascend算子的编程模型等，请参见**昇腾文档**。Notebook中已经配置好环境，无需进行CANN软件安装和环境变量配置，直接在VS Code远端环境中直接进行算子分析及后续操作。

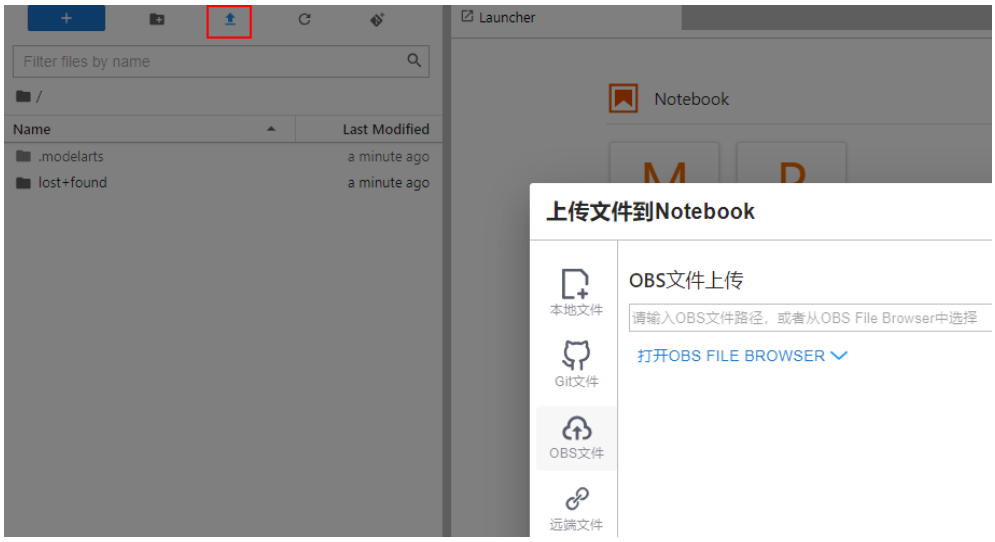
准备工作

- 单击**链接**下载算子样例并上传到OBS桶。
- 创建基于mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b引擎的Notebook实例，并打开SSH远程开发开关。且该Notebook实例状态必须为“运行中”。具体操作参考**创建Notebook实例**。

  说明

  本文档只针对选用“mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b”的引擎进行算子调测，如果使用其它AI引擎可能会报错。

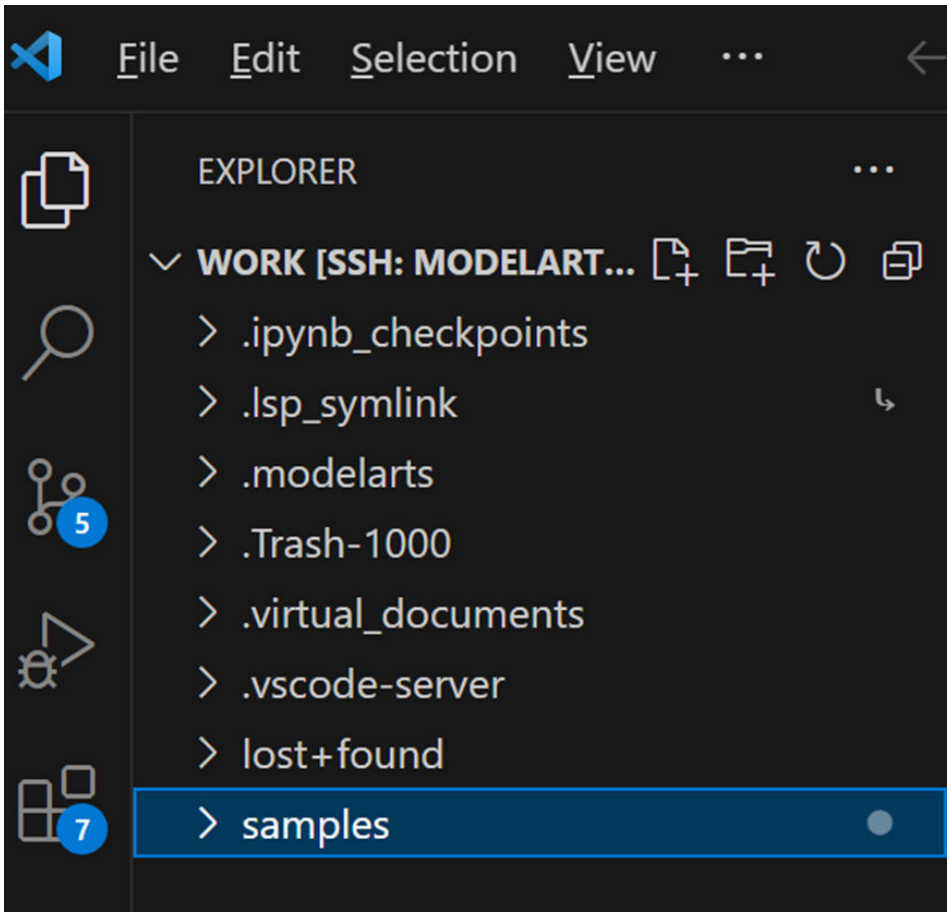
- 打开JupyterLab，单击文件上传按钮  ，将OBS桶的样例文件传至Notebook。详细操作请参考**上传OBS文件到JupyterLab**。



通过 VS Code 连接云端 Notebook

1. Notebook创建完成后处于运行状态，单击操作列的“更多 > VS Code接入”，参考**VS Code一键连接Notebook**连接云上开发环境。
2. 成功连接云上开发环境后，VS Code界面上会显示云上已下载的工程如**图6-169**所示。

图 6-169 工程目录

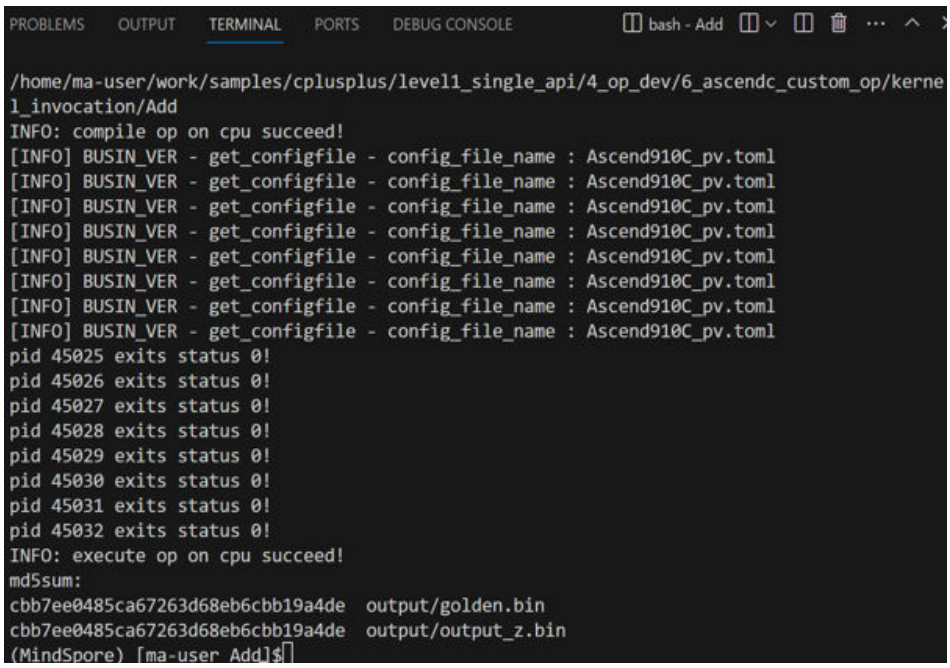


在 VS Code 中调试 Add 算子

1. 在Terminal中执行如下命令进入Add算子所在目录。
   ```
   cd samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Add
   ```
2. 执行如下命令编译和运行脚本。
   a. CPU模式下执行如下命令。
      ```
      bash run.sh add_custom ascend910B1 VectorCore cpu
      ```
      其中，add_custom表示需要运行的算子，ascend910B1表示算子运行的AI处理器型号，VectorCore表示在VectorCore上运行，cpu表示算子以cpu模式运行。

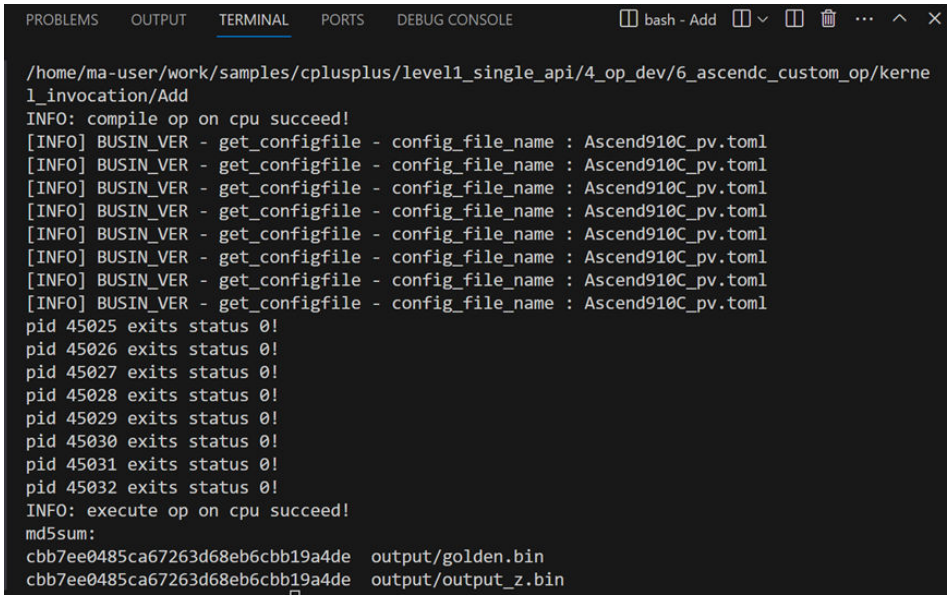
      运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

      图 6-170 CPU 模式运行结果

   b. NPU模式下执行如下命令
      ```
      bash run.sh add_custom ascend910B1 VectorCore npu
      ```
      运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

图 6-171 NPU 模式运行结果



在 VS Code 中调试 matmul 算子

1. matmul算子所在目录为“samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Matmul”。
2. 在work目录下执行如下命令。
   ```
   cd samples/cplusplus/level1_single_api/4_op_dev/6_ascendc_custom_op/kernel_invocation/Matmul
   ```
3. 执行如下命令修改main.cpp文件，此文件为调用算子的应用程序文件。
   ```
   vim main.cpp
   ```
   将param4FileSize的值改为192

   图 6-172 修改 param4FileSize 为 192

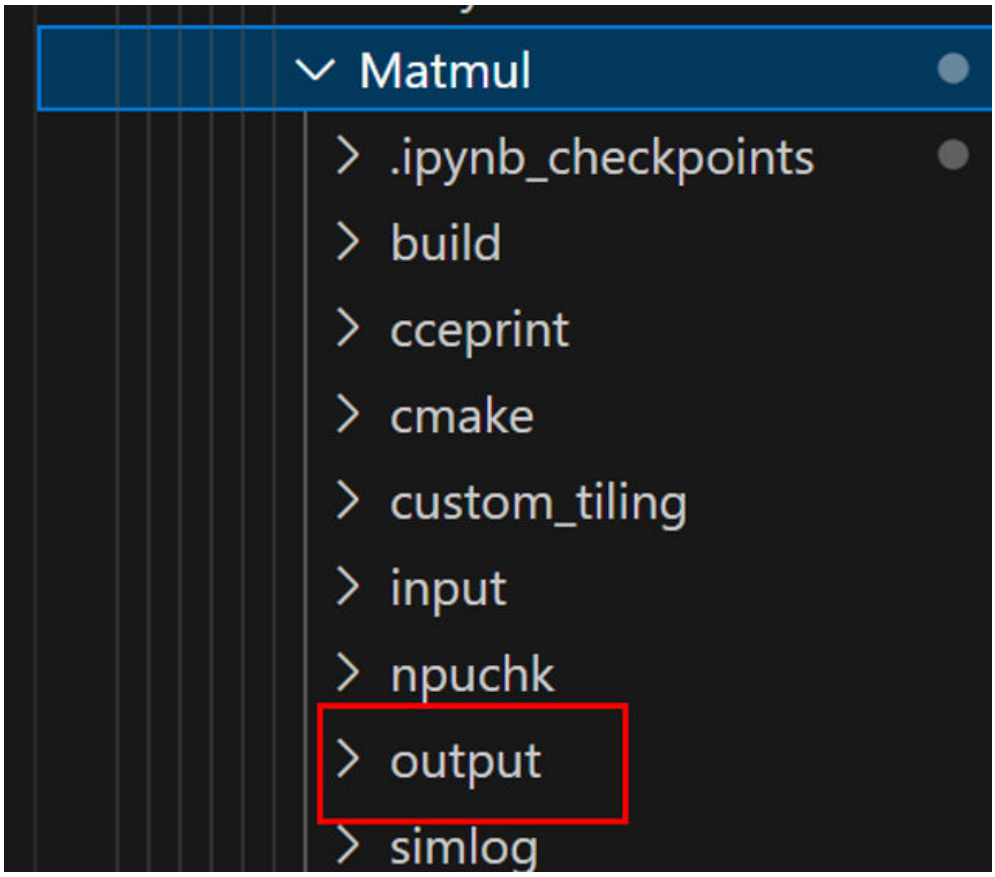
   ```
   int32_t main(int32_t argc, char* argv[])
   {
       size_t param1FileSize = 512 * 512 * sizeof(uint16_t);  // uint16_t represent half
       size_t param2FileSize = 512 * 1024 * sizeof(uint16_t);  // uint16_t represent half
       size_t param3FileSize = 512 * 1024 * sizeof(float);
       size_t param4FileSize = 192 * sizeof(uint32_t);
   "main.cpp" 100L, 4472B                                                1,1           Top
   ```
4. 执行如下命令修改vim matmul_custom.cpp文件。
   ```
   vim matmul_custom.cpp
   ```
   将 matmul_custom.cpp中的tiling.K更改成tiling.Ka

   图 6-173 tiling.K 更改成 tiling.Ka

   ```
       if (GetBlockIdx() >= tiling.usedCoreNum) {
           return;
       }

       GlobalTensor<A_T> aGlobal;
       GlobalTensor<B_T> bGlobal;
       GlobalTensor<C_T> cGlobal;

       aGlobal.SetGlobalBuffer(reinterpret_cast<__gm__ A_T*>(a), tiling.M * tiling.Ka);
       bGlobal.SetGlobalBuffer(reinterpret_cast<__gm__ B_T*>(b), tiling.Ka * tiling.N);
       cGlobal.SetGlobalBuffer(reinterpret_cast<__gm__ C_T*>(c), tiling.M * tiling.N);
   ```

5. 手动在Matmul目录下创建名为“output”的文件夹。

图 6-174 创建 output 文件夹



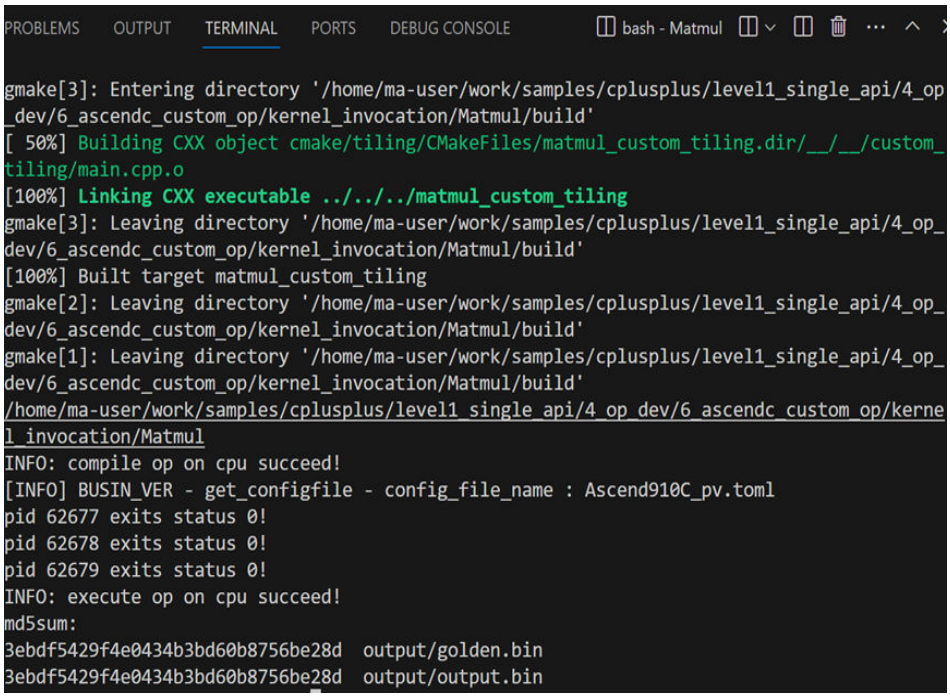
6. 执行如下命令编译和运行脚本。

   a. CPU模式下执行如下命令

   ```
   bash run.sh matmul_custom ascend910B1 AiCore cpu ONBOARD CUSTOM_TILING
   ```

   运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

图 6-175 CPU 模式运行结果

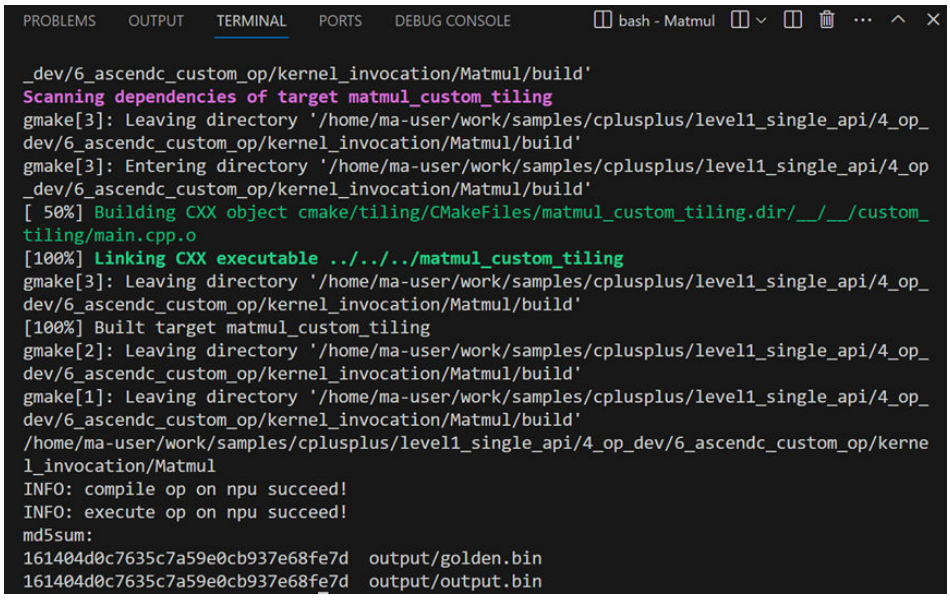


b. NPU模式下执行如下命令

```
bash run.sh matmul_custom ascend910B1 AiCore npu ONBOARD CUSTOM_TILING
```

运行结果如下，当前使用md5sum对比了所有输出bin文件，md5值一致表示实际的输出数据和真值数据相符合。

图 6-176 NPU 模式运行结果

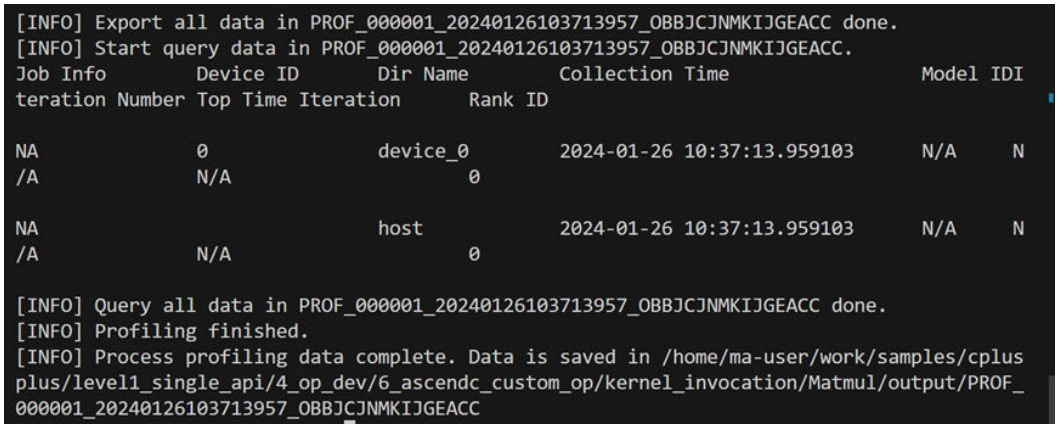


生成 profile

NPU调试后，会在工程目录下生成matmul_custom_npu可执行文件，执行如下命令生成profiling。

```
msprof --application="matmul_custom_npu" --output="./output"
```

图 6-177 生成 profile



停止 Notebook 实例前备份文件

Notebook实例停止时，后端对应的容器环境会被删除，只有“/home/ma-user/work”目录下的内容会持久化保存，其他目录下的修改都会丢失。

备份方法

可以在停止Notebook实例前手工复制文件到/home/ma-user/work目录下。

需要复制的目录内容包括：

1. /home/ma-user/ AscendProjects目录下的自建工程
2. /home/ma-user/modelzoo/目录下的模型转换后的om文件、配置文件、评估报告
3. /home/ma-user/.mindstudio目录下的ssh配置
4. 其他用户自己修改的内容

当Notebook实例再次启动时，用户将手工备份的目录内容复制回原始目录后即可正常继续使用。

6.7 ModelArts CLI 命令参考

6.7.1 ModelArts CLI 简介

功能介绍

ModelArts CLI，即ModelArts命令行工具，是一个跨平台命令行工具，用于连接ModelArts服务并在ModelArts资源上执行管理命令。用户可以使用交互式命令行提示符或脚本通过终端执行命令。为了方便理解，下面将ModelArts CLI统称为ma-cli。ma-cli支持用户在ModelArts Notebook及线下虚拟机中与云端服务交互，使用ma-cli命令可以实现命令自动补全、鉴权、镜像构建、提交ModelArts训练作业、提交DLI Spark作业、OBS数据拷贝等。

使用场景

- ma-cli已经集成在ModelArts开发环境Notebook中，可以直接使用。

  登录ModelArts控制台，在“开发环境 > Notebook”中创建Notebook实例，打开Terminal，使用ma-cli命令。

- ma-cli在本地Windows/Linux环境中需要安装后在本地Terminal中使用。安装步骤具体可参考**（可选）本地安装ma-cli**。

  说明

  - ma-cli不支持在git-bash上使用。
  - 推荐使用Linux Bash、ZSH、Fish，WSL或PowerShell等Terminal。在使用过程中，注意您的敏感信息数据保护，避免敏感信息泄露。

命令预览

```
$ ma-cli -h
Usage: ma-cli [OPTIONS] COMMAND [ARGS]...

Options:
  -V, -v, --version       1.2.1
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help          Show this message and exit.

Commands:
  configure        Configures authentication and endpoints info for the CLI.
  image            Support get registered image list、register or unregister image、debug image, build image
in Notebook.
  obs-copy         Copy file or directory between OBS and local path.
  ma-job           ModelArts job submission and query job details.
  dli-job          DLI spark job submission and query job details.
  auto-completion  Auto complete ma-cli command in terminal, support "bash(default)/zsh/fish".
```

其中，-C、-D、-P，-h参数属于全局可选参数。

- -C表示在执行此命令时可以手动指定鉴权配置文件，默认使用~/.modelarts/ma-cli-profile.yaml配置文件；
- -P表示鉴权文件中的某一组鉴权信息，默认是DEFAULT；
- -D表示是否开启debug模式（默认关闭），当开启debug模式后，命令的报错堆栈信息将会打印出来，否则只会打印报错信息；
- -h表示显示命令的帮助提示信息。

命令说明

表 6-10 ma-cli 支持的命令

命令	命令详情
configure	ma-cli鉴权命令，支持用户名密码、AK/SK
image	ModelArts镜像构建、镜像注册、查询已注册镜像信息等
obs-copy	本地和OBS文件/文件夹间的相互拷贝
ma-job	ModelArts训练作业管理，包含作业提交、资源查询等

命令	命令详情
dli-job	DLI Spark任务提交及资源管理
auto-completion	命令自动补全

6.7.2 （可选）本地安装 ma-cli

使用场景

本文以Windows系统为例，介绍如何在Windows环境中安装ma-cli。

Step1：安装 ModelArts SDK

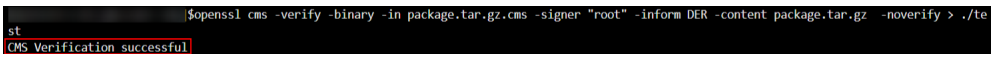
参考《 ModelArts SDK参考 》 > 准备工作 > 本地安装ModelArts SDK完成SDK的安装。

Step2：下载 ma-cli

1. 。
2. 完成软件包签名校验。
   a. 。
   b. 安装openssl并执行如下命令进行签名校验。

```
openssl cms -verify -binary -in D:\ma_cli-latest-py3-none-any.whl.cms -inform DER -content D:\ma_cli-latest-py3-none-any.whl  -noverify > ./test
```

> **说明**
>
> 本示例以软件包在D:\举例，请根据软件包实际路径修改。



Step3：安装 ma-cli

1. 在本地环境cmd中执行命令**python --version**，确认环境已经安装完成Python。（ Python版本需大于3.7.x且小于3.10.x版本，推荐使用3.7.x版本 ）

```
C:\Users\xxx>python --version
Python *.*.*
```

2. 执行命令**pip --version**，确认Python通用包管理工具pip已经存在。

```
C:\Users\xxx>pip --version
pip **.*.* from c:\users\xxx\appdata\local\programs\python\python**\lib\site-packages\pip (python *.*)
```

3. 执行如下命令，安装ma-cli。

   pip install *{ma-cli软件包路径}*\ma_cli-latest-py3-none-any.whl

```
C:\Users\xxx>pip install C:\Users\xxx\Downloads\ma_cli-latest-py3-none-any.whl
.....
Successfully installed ma_cli.*.*.*
```

   在安装ma-cli时会默认同时安装所需的依赖包。当显示“Successfully installed”时，表示ma-cli安装完成。

说明

如果在安装过程中报错提示缺少相应的依赖包，请根据报错提示执行如下命令进行依赖包安装。

pip install xxxx

其中，xxxx为依赖包的名称。

6.7.3 ma-cli auto-completion 自动补全命令

命令行自动补全是指用户可以在Terminal中输入命令前缀通过Tab键自动提示支持的ma-cli命令。ma-cli自动补全功能需要手动在Terminal中激活。执行**ma-cli auto-completion**命令，用户根据提示的补全命令，拷贝并在当前Terminal中执行，就可以自动补全ma-cli的命令。目前支持Bash、Fish及Zsh三种Shell，默认是Bash。

以Bash命令为例：在Terminal中执行**eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"**激活自动补全功能。

```
eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

此外，可以通过“ma-cli auto-completion Fish”或“ma-cli auto-completion Fish”命令查看“Zsh”、“Fish”中的自动补全命令。

命令概览

```
$ ma-cli auto-completion -h
Usage: ma-cli auto-completion [OPTIONS] [[Bash|Zsh|Fish]]

  Auto complete ma-cli command in terminal.

  Example:

  # print bash auto complete command to terminal
  ma-cli auto-completion Bash

Options:
  -H, -h, --help  Show this message and exit.
```

```
# 默认显示Bash Shell自动补全命令

$ ma-cli auto-completion

Tips: please paste following shell command to your terminal to activate auto complation.

[ OK ] eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"

# 执行上述命令，此时Terminal已经支持自动补全

$ eval "$(_MA_CLI_COMPLETE=bash_source ma-cli)"
```

```
# 显示Fish Shell自动补全命令
$ ma-cli auto-completion Fish
Tips: please paste following shell command to your terminal to activate auto complation.

[ OK ] eval (env _MA_CLI_COMPLETE=fish_source ma-cli)
```

6.7.4 ma-cli configure 鉴权命令

鉴权信息说明

- 在虚拟机及个人PC场景，需要配置鉴权信息，目前支持用户名密码鉴权（默认）和AK/SK鉴权；

- 在使用账号认证时，需要指定username和password；在使用IAM用户认证时，需要指定account、username和password；在使用云星账号登录时，需要指定account和password；
- 在ModelArts Notebook中可以不用执行鉴权命令，默认使用委托信息，不需要手动进行鉴权操作；
- 如果用户在ModelArts Notebook中也配置了鉴权信息，那么将会优先使用用户指定的鉴权信息。

说明

在鉴权时，注意您的敏感信息数据保护，避免敏感信息泄露。

命令参数总览

```
$ ma-cli configure -h
Usage: ma-cli configure [OPTIONS]

Options:
  -auth, --auth [PWD|AKSK|ROMA]  Authentication type.
  -rp, --region-profile PATH      ModelArts region file path.
  -a, --account TEXT              Account of an IAM user.
  -u, --username TEXT             Username of an IAM user.
  -p, --password TEXT             Password of an IAM user
  -ak, --access-key TEXT          User access key.
  -sk, --secret-key TEXT          User secret key.
  -r, --region TEXT               The region you want to visit.
  -pi, --project-id TEXT          User project id.
  -C, --config-file TEXT          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```

表 6-11 鉴权命令参数说明

参数名	参数类型	是否必选	参数说明
-auth / --auth	String	否	鉴权方式，支持PWD（用户名密码）、AKSK（access key和secret key），默认是PWD。
-rp / --region-profile	String	否	指定ModelArts region配置文件信息。
-a / --account	String	否	IAM租户账号，在使用IAM用户认证或者云星账号场景时需要指定，属于PWD鉴权的一部分。
-u / --username	String	否	用户名，在使用账号认证时表示账号名，IAM认证时表示IAM用户名，在云星账号场景不需要指定，属于PWD鉴权的一部分。
-p / --password	String	否	密码，属于PWD鉴权的一部分。
-ak / --access-key	String	否	access key，属于AKSK鉴权的一部分。
-sk / --secret-key	String	否	secret key，属于AKSK鉴权的一部分。

参数名	参数类型	是否必选	参数说明
-r / --region	String	否	region名称，如果不填会默认使用REGION_NAME环境变量的值。
-pi / --project-id	String	否	项目ID，如果不填会默认使用对应region的值，或者使用PROJECT_ID环境变量。
-P / --profile	String	否	鉴权配置项，默认是DEFAULT。
-C / --config-file	String	否	配置文件本地路径，默认路径为~/.modelarts/ma-cli-profile.yaml。

配置用户名密码鉴权

以在虚拟机上使用**ma-cli configure**为例，介绍如何配置用户名密码进行鉴权。本地虚拟机场景需要指定yaml文件和局点的endpoint信息，请联系局点运营公司获取。使用方式如下：

说明

以下样例中所有以${}装饰的字符串都代表一个变量，用户可以根据实际情况指定对应的值。

比如${your_password}表示输入用户自己的密码信息。

```
# 默认使用DEFAULT鉴权配置项，默认提示账号、用户名及密码（其中账号和用户名如果不需要填写可以使用Enter跳过）
$ ma-cli configure --auth PWD --region ${your_region} --region-profile ${your_region-profile path}
account: ${your_account}
username: ${your_username}
password: ${your_password}  # 输入在控制台不会回显
```

其中，${your_region-profile path}为yaml文件本地相对路径，举例：“./ModelArts-region-profile.yaml”。

AKSK 鉴权

如下命令表示使用AKSK进行鉴权，需要交互式输入AK及SK信息。默认提示AK和SK，且输入在控制台不会回显。

注意

以下样例中所有以${}装饰的字符串都代表一个变量，用户可以根据实际情况指定对应的值。

比如${access key}表示输入用户自己的access key。

```
ma-cli configure --auth AKSK
access key [***]: ${access key}
secret key [***]: ${secret key}
```

执行完鉴权命令后，将会在~/.modelarts/ma-cli-profile.yaml配置文件中保存相应的鉴权信息。

6.7.5 ma-cli image 镜像构建命令

6.7.5.1 ma-cli image 镜像构建命令概述

ma-cli image命令支持：查询用户已注册的镜像、查询/加载镜像构建模板、Dockerfile镜像构建、查询/清理镜像构建缓存、注册/取消注册镜像、调试镜像是否可以在Notebook中使用等。具体命令及功能可执行**ma-cli image -h**命令查看。

镜像构建命令总览

```
$ ma-cli image -h
Usage: ma-cli image [OPTIONS] COMMAND [ARGS]...
  Support get registered image list, register or unregister image, debug image, build image in Notebook.

Options:
  -H, -h, --help  Show this message and exit.

Commands:
  add-template, at  List build-in dockerfile templates.
  build             Build docker image in Notebook.
  debug             Debug SWR image as a Notebook in ECS.
  df                Query disk usage.
  get-image, gi     Query registered image in ModelArts.
  get-template, gt  List build-in dockerfile templates.
  prune             Prune image build cache.
  register          Register image to ModelArts.
  unregister        Unregister image from ModelArts.
```

表 6-12 镜像构建支持的命令

命令	命令详情
get-template	查询镜像构建模板。
add-template	加载镜像构建模板。
get-image	查询ModelArts已注册镜像。
register	注册SWR镜像到ModelArts镜像管理。
unregister	取消注册ModelArts镜像管理中的已注册镜像。
build	基于指定的Dockerfile构建镜像 （只支持ModelArts Notebook里使用）。
df	查询镜像构建缓存（只支持ModelArts Notebook里使用）。
prune	清理镜像构建缓存 （只支持ModelArts Notebook里使用）。
debug	在ECS上调试SWR镜像是否能在ModelArts Notebook中使用 （只支持已安装docker环境的ECS）。

6.7.5.2 查询镜像构建模板

ma-cli提供了一些常用的镜像构建模板，模板中包含了在ModelArts Notebook上进行Dockerfile开发的牵引指导。

```
$ ma-cli image get-template -h
Usage: ma-cli image get-template [OPTIONS]

  List build-in dockerfile templates.

  Example:

  # List build-in dockerfile templates
  ma-cli image get-template [--filer <filter_info>]  [--page-num <yourPageNum>] [--page-size
<yourPageSize>]

Options:
  --filter TEXT                  filter by keyword.
  -pn, --page-num INTEGER RANGE   Specify which page to query.  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query.  [x>=1]
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
(PyTorch-1.4) [ma-user work]$
```

表 6-13 参数说明

参数名	参数类型	是否必选	参数说明
--filter	String	否	根据模板名称关键字过滤模板列表。
-pn / --page-num	Int	否	镜像页索引，默认是第1页。
-ps / --page-size	Int	否	每页显示的镜像数量，默认是20。

示例

查看镜像构建模板。

```
ma-cli image get-template
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image get-template
Template Name                                          Description
-----------------------------------------------------  ------------------------------------------------------------------------------------------------------------
customize_from_ubuntu_18.04_to_modelarts               Add ma-user, apt install packages and create a new conda environment with pip based on scratch ubuntu 18.04
upgrade_current_notebook_apt_packages                  Install apt packages like ffmpeg, gcc-8, g++-8 based on current Notebook image
migrate_3rd_party_image_to_modelarts                   General template for migrating your own or open source image to ModelArts
migrate_official_torch_110_cu113_image_to_modelarts    Reconstructing and migrating the official torch 1.10.0 with cuda11.3 image to ModelArts
build_handwritten_number_inference_application         Create a new AI application, used to generate an image to deploy and infer in ModelArts
update_dli_image_pip_package                           Install pip packages based on DLI image
forward_compat_cuda_11_image_to_modelarts              Migrate and forward compat cuda-11.x to ModelArts by upgrading only user-mode CUDA components
```

6.7.5.3 加载镜像构建模板

ma-cli可以使用**add-template**命令将镜像模板加载到指定文件夹下，默认路径为当前命令所在的路径。

比如${current_dir}/.ma/${template_name}/。也可以通过--dest命令指定保存的路径。当保存的路径已经有同名的模板文件夹时，可以使用--force | -f参数进行强制覆盖。

```
$ ma-cli image add-template -h
Usage: ma-cli image add-template [OPTIONS] TEMPLATE_NAME

  Add buildin dockerfile templates into disk.

  Example:

  # List build-in dockerfile templates
  ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts --force

Options:
  --dst TEXT          target save path.
  -f, --force         Override templates that has been installed.
  -D, --debug         Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT  CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help      Show this message and exit.
```

表 6-14 参数说明

参数名	参数类型	是否必选	参数说明
--dst	String	否	加载模板到指定路径，默认是当前路径。
-f / --force	Bool	否	是否强制覆盖已存在的同名模板，默认不覆盖。

示例

加载**customize_from_ubuntu_18.04_to_modelarts**镜像构建模板。

```
ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image add-template customize_from_ubuntu_18.04_to_modelarts
[ OK ] Successfully add configuration template [ customize_from_ubuntu_18.04_to_modelarts ] under folder [ /home/ma-user/work/.ma/customize_from_ubuntu_18.04_to_modelarts ]
```

6.7.5.4 查询 ModelArts 已注册镜像

Dockerfile一般需要提供一个基础镜像的地址，目前支持从docker hub等开源镜像仓拉取公开镜像，以及SWR的公开或私有镜像。其中ma-cli提供了查询ModelArts预置镜像和用户已注册镜像列表及SWR地址。

```
$ma-cli image get-image -h
Usage: ma-cli image get-image [OPTIONS]

  Get registered image list.

  Example:

  # Query images by image type and only image id, show name and swr_path
  ma-cli image get-image --type=DEDICATED

  # Query images by image id
  ma-cli image get-image --image-id ${image_id}

  # Query images by image type and show more information
  ma-cli image get-image --type=DEDICATED -v

  # Query images by image name
  ma-cli image get-image --filter=torch
```

```
Options:
  -t, --type [BUILD_IN|DEDICATED|ALL]
                                  Image type(default ALL)
  -f, --filter TEXT               Image name to filter
  -v, --verbose                   Show detailed information on image.
  -i, --image-id TEXT             Get image details by image id
  -n, --image-name TEXT           Get image details by image name
  -wi, --workspace-id TEXT        The workspace where you want to query image(default "0")
  -pn, --page-num INTEGER RANGE   Specify which page to query  [x>=1]
  -ps, --page-size INTEGER RANGE  The maximum number of results for this query  [x>=1]
  -C, --config-file PATH          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
```

表 6-15 参数说明

参数名	参数类型	是否必选	参数说明
-t / --type	String	否	查询的镜像类型，支持BUILD_IN、DEDICATED和ALL三种查询类型。 ● BUILD_IN：预置镜像 ● DEDICATED：用户已注册的自定义镜像 ● ALL：所有镜像
-f / --filter	String，	否	镜像名关键字。根据镜像名关键字过滤镜像列表。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。
-i / --image-id	String	否	查询指定镜像ID的镜像详情。
-n / --image-name	String	否	查询指定镜像名称的镜像详情。
-wi / --workspace-id	String	否	查询指定工作空间下的镜像信息。
-pn / --page-num	Int	否	镜像页索引，默认是第1页。
-ps / --page-size	Int	否	每页显示的镜像数量，默认是20。

示例

查询ModelArts已注册的自定义镜像。

```
ma-cli image get-image --type=DEDICATED
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image get-image --type=DEDICATED
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
| INDEX |              IMAGE ID                |      NAME       |                            SWR PATH                             |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   1   | c857e5a8                 fc5e3d002f  |    0314test     |          huaweicloud.com/notebook_test/0314test:1.0.0           |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   2   | 193b2557                 d39093a811  |      0328       |          7.myhuaweicloud.com/notebook_test/0328:1                |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   3   | 171fe036                 3b37e9aa7c  |      0926       |          aweicloud.com/ei_modelarts_y00218826_05/0926:1          |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   4   | 1b48bb0a                 689b0a7267  |      0926       |          weicloud.com/ei_modelarts_y00218826_05/0926:111         |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   5   | c8667cf0                 d2e3563107  |        1        |          huaweicloud.com/ei_modelarts_y00218826_05/1:6           |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   6   | 3e6cda6a                 a360eea80e  |        1        |          huaweicloud.com/ei_modelarts_y00218826_05/1:1           |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   7   | 42e86ca5                 ec198be968  |       111       |          .myhuaweicloud.com/notebook_test/111:1227               |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   8   | 0f349cef                 c411011ef2  |   11111110801   |          aweicloud.com/notebook_test/11111110801:111111          |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   9   | 3a082e32                 4f485aadb6  |     112121      |          eicloud.com/ei_modelarts_y00218826_05/112121:123         |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   10  | db0d02f6                 74eb00e1ce  |      1203       |          myhuaweicloud.com/notebook_test/1203:1.2.3              |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   11  | 031dc02e                 d92cd457d8  |      1227       |          .myhuaweicloud.com/notebook_test/1227:111               |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   12  | f7d95648                 7aaec8b1cc  |      1227       |          .myhuaweicloud.com/notebook_test/1227:888               |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   13  | 2f720610                 a1d1db9d7d  |      1227       |          .myhuaweicloud.com/notebook_test/1227:6666              |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   14  | 42221bf2                 22d726d270  |      1229       |          .myhuaweicloud.com/notebook_test/1229:123               |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   15  | 70deea1e                 70b2414ae7  |       123       |          myhuaweicloud.com/mindspore-dis-train/123:2            |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   16  | e6cc5414                 ce318069f4  |       123       |          .myhuaweicloud.com/notebook_test/123:45678             |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   17  | 6e7a86c9                 319fb3bb28  |      1234       |          .myhuaweicloud.com/notebook_test/1234:666              |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   18  | ec036306                 8c9dc6b391  |      1234       |          7.myhuaweicloud.com/notebook_test/1234:1               |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   19  | b37f8f3b                 7a9941c978  |     441211      |          .myhuaweicloud.com/notebook_test/441211:11             |
+-------+--------------------------------------+-----------------+-----------------------------------------------------------------+
|   20  | d5acd51b                 ef16534d68  |       aaa       |          .myhuaweicloud.com/notebook_test/aaa:1.1.1             |
```

6.7.5.5 在 ModelArts Notebook 中进行镜像构建

使用**ma-cli image build**命令基于指定的Dockerfile进行镜像构建，仅支持在ModelArts Notebook里使用该命令。

```
$ ma-cli image build -h
Usage: ma-cli image build [OPTIONS] FILE_PATH

  Build docker image in Notebook.

  Example:

  # Build a image and push to SWR
  ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr my_organization/
my_image:0.0.1

  # Build a image and push to SWR, dockerfile context path is current dir
  ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr my_organization/
my_image:0.0.1 -context .

  # Build a local image and save to local path and OBS
  ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile --target ./build.tar --
obs_path obs://bucket/object --swr-path my_organization/my_image:0.0.1

Options:
  -t, --target TEXT       Name and optionally a tag in the 'name:tag' format.
  -swr, --swr-path TEXT   SWR path without swr endpoint, eg:organization/image:tag.  [required]
  --context DIRECTORY     build context path.
  -arg, --build-arg TEXT  build arg for Dockerfile.
  -obs, --obs-path TEXT   OBS path to save local built image.
  -f, --force             Force to overwrite the existing swr image with the same name and tag.
  -C, --config-file PATH  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

表 6-16 参数说明

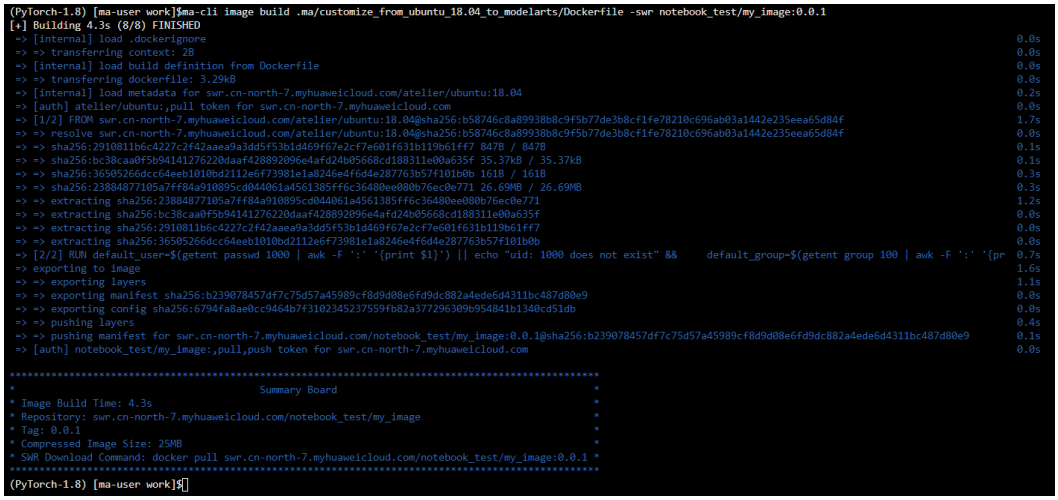
参数名	参数类型	是否必选	参数说明
FILE_PATH	String	是	Dockerfile文件所在的路径。
-t / --target	String	否	表示构建生成的tar包保存在本地的路径，默认是当前文件夹目录。
-swr / --swr-path	String	是	SWR镜像名称，遵循organization/image_name:tag格式，针对于构建保存tar包场景可以省略。
--context	String	否	Dockerfile构建时的上下文信息路径，主要用于数据拷贝。
-arg / --build-arg	String	否	指定构建参数，多个构建参数可以使用--build-arg VERSION=18.04 --build-arg ARCH=X86_64
-obs / --obs-path	String	否	将生成的tar包自动上传到OBS中。
-f / --force	Bool	否	是否强制覆盖已存在的SWR镜像，默认不覆盖。

示例

在ModelArts Notebook里进行镜像构建。

```
ma-cli image build .ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile -swr notebook_test/my_image:0.0.1
```

其中“.ma/customize_from_ubuntu_18.04_to_modelarts/Dockerfile”为Dockerfile文件所在路径，“notebook_test/my_image:0.0.1”为构建的新镜像的SWR路径。



6.7.5.6 在 ModelArts Notebook 中查询镜像构建缓存

使用**ma-cli image df**命令查询镜像构建缓存，仅支持在ModelArts Notebook里使用该命令。

```
$ ma-cli image df -h
Usage: ma-cli image df [OPTIONS]

  Query disk usage used by image-building in Notebook.

  Example:

  # Query image disk usage
  ma-cli image df

Options:
  -v, --verbose       Show detailed information on disk usage.
  -D, --debug          Debug Mode. Shows full stack trace when error occurs.

  -h, -H, --help      Show this message and exit.
```

表 6-17 参数说明

参数名	参数类型	是否必选	参数说明
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

- 在ModelArts Notebook里查看所有镜像缓存。
  ma-cli image df

```
(PyTorch-1.8) [ma-user work]$ma-cli image df
ID                                                              RECLAIMABLE     SIZE            LAST ACCESSED
iwrrwsi9pdcjafe1ij6d0r918                                       true            98.50MB
cp52c4q81ud2abu2vp7sj5vyt                                       true            1.04MB
4jbo6v06r2wl575ddq3w8g12e                                       true            139.68kB
ojdjw5mok71s1nh2cauant051                                       true            86.86kB
k2jm6g061n5twmz7gmonmqjsh                                       true            16.55kB
efu5kwgig1ve44fe7smbrncnh*                                      true            8.19kB
uzikwqk5taxnslvajm14jrbje*                                      true            4.10kB
2g8p0qcb014g3qva7ucawkv87*                                      true            4.10kB
Reclaimable:    99.80MB
Total:          99.80MB
```

- 显示镜像的详细信息。
  ma-cli image df --verbose

```
(PyTorch-1.8) [ma-user work]$ma-cli image df --verbose
ID:             iwrrwsi9pdcjafe1ij6d0r918
Created at:     2023-03-28 12:23:28.353759532 +0000 UTC
Mutable:        false
Reclaimable:    true
Shared:         false
Size:           98.50MB
Description:    pulled from swr                .myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b5874                                                                ea65d84f
Usage count:    1
Last used:      2023-03-28 12:23:28.37337776 +0000 UTC
Type:           regular

ID:             cp52c4q81ud2abu2vp7sj5vyt
Parents:        iwrrwsi9pdcjafe1ij6d0r918
Created at:     2023-03-28 12:23:28.366910223 +0000 UTC
Mutable:        false
Reclaimable:    true
Shared:         false
Size:           1.04MB
Description:    pulled from swr.               .myhuaweicloud.com/atelier/ubuntu:18.04@sha256:b5874                                                               2e235eea65d84f
Usage count:    1
Last used:      2023-03-28 12:23:28.38560437 +0000 UTC
Type:           regular

ID:             4jbo6v06r2wl575ddq3w8g12e
Parents:        k2jm6g061n5twmz7gmonmqjsh
Created at:     2023-03-28 12:23:30.681643727 +0000 UTC
Mutable:        false
Reclaimable:    true
Shared:         false
Size:           139.68kB
Description:    mount / from exec /bin/sh -c default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" &&     default_group=$(getent
up 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" &&     if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then         userdel -r ${defau
user};     fi &&     if [ ! -z ${default_group} ] && [ ${default_group} != "ma-group" ]; then         groupdel -f ${default_group};     fi &&     groupadd -g 100 ma-group
useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user &&     chmod -R 750 /home/ma-user
Usage count:    2
Last used:      2023-03-28 12:25:39.149080471 +0000 UTC
Type:           regular
```

6.7.5.7 在 ModelArts Notebook 中清理镜像构建缓存

使用**ma-cli image prune**命令清理镜像构建缓存，仅支持在ModelArts Notebook里使用该命令。

```
$ ma-cli image prune -h
Usage: ma-cli image prune [OPTIONS]

  Prune image build cache by image-building in Notebook.

  Example:

  # Prune image build cache
  ma-cli image prune

Options:
  -ks, --keep-storage INTEGER  Amount of disk space to keep for cache below this limit (in MB) (default: 0).
  -kd, --keep-duration TEXT    Keep cache newer than this limit, support second(s), minute(m) and hour(h)
(default:   0s).
  -v, --verbose                Show more verbose output.
  -D, --debug                  Debug Mode. Shows full stack trace when error occurs.
  -h, -H, --help               Show this message and exit.
```

表 6-18 参数说明

参数名	参数类型	是否必选	参数说明
-ks / --keep-storage	Int	否	清理缓存时保留的缓存大小，单位是MB，默认是0，表示全部清理。
-kd / --keep-duration	String	否	清理缓存时保留较新的缓存，只清除历史缓存，单位为s（秒）、m（分钟）、h（小时），默认是0s，表示全部清理。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

清理保留1MB镜像缓存。

```
ma-cli image prune -ks 1
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image prune -ks 1
ID                                                                RECLAIMABLE     SIZE    LAST ACCESSED
uzikwqk5taxnslvajm14jrbje*                                        true            4.10kB
4jbo6v06r2w1575ddq3w8g12e                                         true            139.68kB
k2jm6g061n5twmz7gmonmqjsh                                         true            16.55kB
ojdjw5mok71s1nh2cauant051                                         true            86.86kB
cp52c4q81ud2abu2vp7sj5vyt                                         true            1.04MB
iwrrwsi9pdcjafe1ij6d0r918                                         true            98.50MB
Total:  99.79MB
```

6.7.5.8 注册 SWR 镜像到 ModelArts 镜像管理

调试完成后，使用**ma-cli image register**命令将新镜像注册到ModelArts镜像管理服务中，进而在能够在ModelArts中使用该镜像。

```
$ma-cli image register -h
Usage: ma-cli image register [OPTIONS]
```

```
Register image to ModelArts.

Example:

# Register image into ModelArts service
ma-cli image register --swr-path=xx

# Share SWR image to DLI service
ma-cli image register -swr xx -td

# Register image into ModelArts service and specify architecture to be 'AARCH64'
ma-cli image register --swr-path=xx --arch AARCH64

Options:
  -swr, --swr-path TEXT           SWR path without swr endpoint, eg:organization/image:tag.  [required]
  -a, --arch [X86_64|AARCH64]     Image architecture (default: X86_64).
  -s, --service [NOTEBOOK|MODELBOX]
                                  Services supported by this image(default NOTEBOOK).
  -rs, --resource-category [CPU|GPU|ASCEND]
                                  The resource category supported by this image (default: CPU and GPU).
  -wi, --workspace-id TEXT        The workspace to register this image (default: "0").
  -v, --visibility [PUBLIC|PRIVATE]
                                  PUBLIC: every user can use this image. PRIVATE: only image owner can use this
image (Default: PRIVATE).
  -td, --to-dli                   Register swr image to DLI, which will share SWR image to DLI service.
  -d, --description TEXT          Image description (default: "").
  -C, --config-file PATH          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```

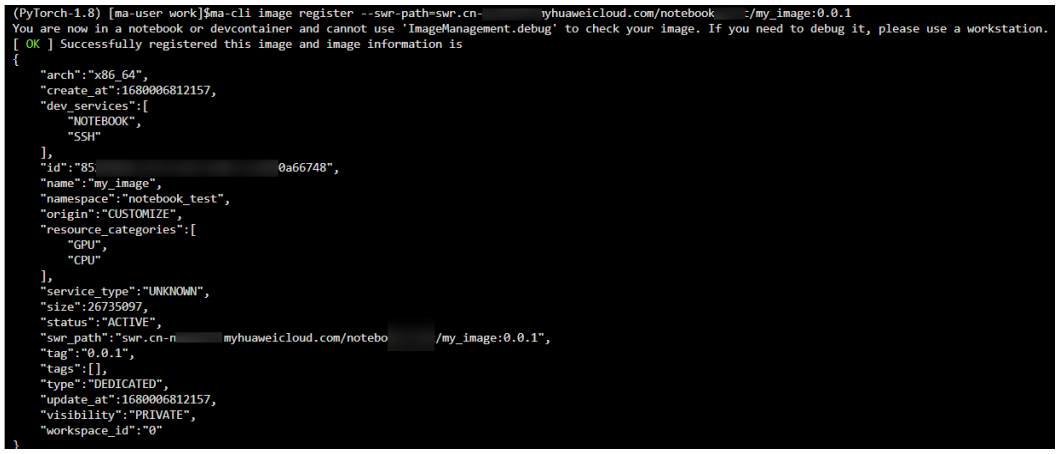
表 6-19 参数说明

参数名	参数类型	是否必选	参数说明
-swr / --swr-path	String	是	需要注册的镜像的SWR路径。
-a / --arch	Sring	否	注册镜像的架构，X86_64或者AARCH64，默认是X86_64。
-s / --service	String	否	注册镜像的服务类型，NOTEBOOK或者MODELBOX，默认是NOTEBOOK。 可以输入多个值，如-s NOTEBOOK -s MODELBOX。
-rs / --resource-category	String	否	注册镜像能够使用的资源类型，默认是CPU和GPU。
-wi / --workspace-id	String	否	注册镜像到指定的工作空间，workspace ID默认是0。
-v / --visibility	Bool	否	注册的镜像可见性，PRIVATE（仅自己可见）或者PUBLIC（所有用户可见），默认是PRIVATE。
-td / --to-dli	Bool	否	注册镜像到DLI服务。
-d/ --description	String	否	填写镜像描述，默认为空。

示例

注册SWR镜像到ModelArts。

```
ma-cli image register --swr-path=xx
```



6.7.5.9 取消注册 ModelArts 镜像管理中的已注册镜像

使用**ma-cli image unregister**命令将注册的镜像从ModelArts中删除。

```
$ ma-cli image unregister -h
Usage: ma-cli image unregister [OPTIONS]

  Unregister image from ModelArts.

  Example:

  # Unregister image
  ma-cli image unregister --image-id=xx

  # Unregister image and delete it from swr
  ma-cli image unregister --image-id=xx -d

Options:
  -i, --image-id TEXT     Unregister image details by image id.  [required]
  -d, --delete-swr-image  Delete the image from swr.
  -C, --config-file PATH  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help          Show this message and exit.
```

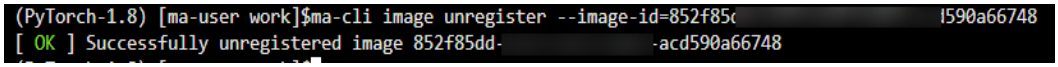
表 6-20 参数说明

参数名	参数类型	是否必选	参数说明
-i / -image-id	String	是	需要取消注册的镜像ID。
-d / --delete-swr-image	Bool	否	取消注册后同步删除SWR镜像开关，默认关闭。

示例

取消ModelArts镜像管理中已注册镜像。

```
ma-cli image unregister --image-id=xx
```



6.7.5.10 在 ECS 上调试 SWR 镜像是否能在 ModelArts Notebook 中使用

ma-cli支持在ECS上调试SWR镜像是否可以在ModelArts开发环境中运行，发现镜像中可能存在的问题。

```
ma-cli image debug -h
Usage: ma-cli image debug [OPTIONS]

  Debug SWR image as a Notebook in ECS.

  Example:

  # Debug cpu notebook image
  ma-cli image debug --swr-path=xx --service=NOTEBOOK --region=

  # Debug gpu notebook image
  ma-cli image debug --swr-path=xx --service=NOTEBOOK --region= --gpu

Options:
  -swr, --swr-path TEXT           SWR path without SWR endpoint, eg:organization/image:tag.  [required]
  -r, --region TEXT               Region name.  [required]
  -s, --service [NOTEBOOK|MODELBOX]
                                  Services supported by this image(default NOTEBOOK).
  -a, --arch [X86_64|AARCH64]     Image architecture(default X86_64).
  -g, --gpu                       Use all gpus to debug.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help                  Show this message and exit.
```

表 6-21 参数说明

参数名	参数类型	是否必选	参数说明
-swr / --swr-path	String	是	需要调试的镜像的SWR路径。
-r / --region	String	是	需要调试的镜像所在的区域。
-s / --service	String	否	调试镜像的服务类型，NOTEBOOK或者MODELBOX，默认是NOTEBOOK。
-a / --arch	Sring	否	调试镜像的架构，X86_64或者AARCH64，默认是X86_64。
-g / --gpu	Bool	否	使用GPU进行调试开关，默认关闭。

6.7.6 使用 ma-cli ma-job 命令提交 ModelArts 训练作业

6.7.6.1 ma-cli ma-job 命令概述

使用**ma-cli ma-job**命令可以提交训练作业，查询训练作业日志、事件、使用的AI引擎、资源规格及停止训练作业等。

```
$ ma-cli ma-job -h
Usage: ma-cli ma-job [OPTIONS] COMMAND [ARGS]...

  ModelArts job submission and query jod details.

Options:
  -h, -H, --help  Show this message and exit.

Commands:
  delete      Delete training job by job id.
  get-engine  Get job engines.
  get-event   Get job running event.
  get-flavor  Get job flavors.
  get-job     Get job details.
  get-log     Get job log details.
  get-pool    Get job engines.
  stop        Stop training job by job id.
  submit      Submit training job.
```

表 6-22 训练作业支持的命令

命令	命令详情
get-job	查询ModelArts训练作业列表及详情。
get-log	查询ModelArts训练作业运行日志。
get-engine	查询ModelArts训练AI引擎。
get-event	查询ModelArts训练作业事件。
get-flavor	查询ModelArts训练资源规格。
get-pool	查询ModelArts训练专属池。
stop	停止ModelArts训练作业。
submit	提交ModelArts训练作业。
delete	删除指定作业id的训练作业。

6.7.6.2 查询 ModelArts 训练作业

使用**ma-cli ma-job get-job**命令可以查看训练作业列表或某个作业详情。

```
$ ma-cli ma-job get-job -h
Usage: ma-cli ma-job get-job [OPTIONS]

  Get job details.

  Example:

  # Get train job details by job name
```

```
ma-cli ma-job get-job -n ${job_name}

# Get train job details by job id
ma-cli ma-job get-job -i ${job_id}

# Get train job list
ma-cli ma-job get-job --page-size 5 --page-num 1
```

Options:

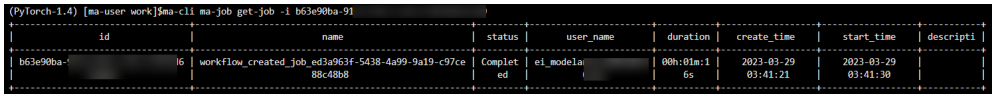
```
-i, --job-id TEXT               Get training job details by job id.
-n, --job-name TEXT             Get training job details by job name.
-pn, --page-num INTEGER         Specify which page to query.  [x>=1]
-ps, --page-size INTEGER RANGE  The maximum number of results for this query.  [1<=x<=50]
-v, --verbose                   Show detailed information about training job details.
-C, --config-file TEXT          Configure file path for authorization.
-D, --debug                     Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
-h, -H, --help                  Show this message and exit.
```

表 6-23 参数说明

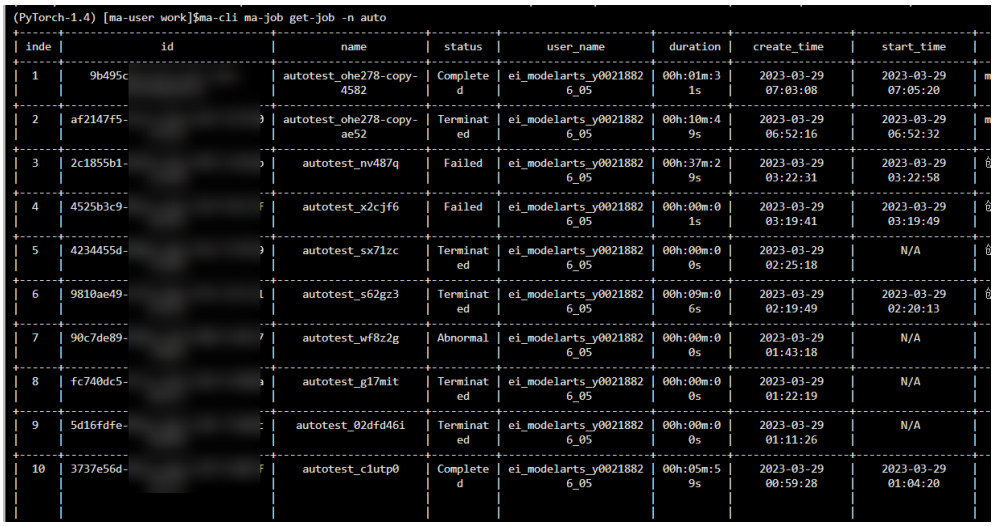
参数名	参数类型	是否必选	参数说明
-i / --job-id	String	否	查询指定训练任务ID的任务详情。
-n / --job-name	String	否	查询指定任务名称的训练任务或根据任务名称关键字过滤训练任务。
-pn / --page-num	Int	否	页面索引，默认是第1页。
-ps / --page-size	Int	否	每页显示的训练作业数量，默认是10。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

- 查询指定任务ID的训练任务。
  ma-cli ma-job get-job -i *b63e90xxx*



- 根据任务名称关键字“auto”过滤训练任务。
  ma-cli ma-job get-job -n *auto*



```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-job -n auto
```

inde	id	name	status	user_name	duration	create_time	start_time
1	9b495c	autotest_ohe278-copy-4582	Completed	ei_modelarts_y0021882_6_05	00h:01m:31s	2023-03-29 07:03:08	2023-03-29 07:05:20
2	af2147f5-	autotest_ohe278-copy-ae52	Terminated	ei_modelarts_y0021882_6_05	00h:10m:49s	2023-03-29 06:52:16	2023-03-29 06:52:32
3	2c1855b1-	autotest_nv487q	Failed	ei_modelarts_y0021882_6_05	00h:37m:29s	2023-03-29 03:22:31	2023-03-29 03:22:58
4	4525b3c9-	autotest_x2cjf6	Failed	ei_modelarts_y0021882_6_05	00h:00m:01s	2023-03-29 03:19:41	2023-03-29 03:19:49
5	4234455d-	autotest_sx71zc	Terminated	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 02:25:18	N/A
6	9810ae49-	autotest_s62gz3	Terminated	ei_modelarts_y0021882_6_05	00h:09m:06s	2023-03-29 02:19:49	2023-03-29 02:20:13
7	90c7de89-	autotest_wf8z2g	Abnormal	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 01:43:18	N/A
8	fc740dc5-	autotest_g17mit	Terminated	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 01:22:19	N/A
9	5d16fdfe-	autotest_02dfd46i	Terminated	ei_modelarts_y0021882_6_05	00h:00m:00s	2023-03-29 01:11:26	N/A
10	3737e56d-	autotest_c1utp0	Completed	ei_modelarts_y0021882_6_05	00h:05m:59s	2023-03-29 00:59:28	2023-03-29 01:04:20

6.7.6.3 提交 ModelArts 训练作业

执行**ma-cli ma-job submit**命令提交ModelArts训练作业。

ma-cli ma-job submit命令需要指定一个位置参数YAML_FILE表示作业的配置文件路径，如果不指定该参数，则表示配置文件为空。配置文件是一个YAML格式的文件，里面的参数就是命令的option参数。此外，如果用户在命令行中同时指定YAML_FILE配置文件和option参数，命令行中指定的option参数的值将会覆盖配置文件相同的值。

```
$ma-cli ma-job submit -h
Usage: ma-cli ma-job submit [OPTIONS] [YAML_FILE]...

  Submit training job.

  Example:

  ma-cli ma-job submit --code-dir obs://your_bucket/code/
                       --boot-file main.py
                       --framework-type PyTorch
                       --working-dir /home/ma-user/modelarts/user-job-dir/code
                       --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
                       --data-url obs://your_bucket/dataset/
                       --log-url obs://your_bucket/logs/
                       --train-instance-type modelarts.vm.cpu.8u
                       --train-instance-count 1

Options:
  --name TEXT                   Job name.
  --description TEXT            Job description.
  --image-url TEXT              Full swr custom image path.
  --uid TEXT                    Uid for custom image (default: 1000).
  --working-dir TEXT            ModelArts training job working directory.
  --local-code-dir TEXT         ModelArts training job local code directory.
  --user-command TEXT           Execution command for custom image.
  --pool-id TEXT                Dedicated pool id.
  --train-instance-type TEXT    Train worker specification.
  --train-instance-count INTEGER  Number of workers.
  --data-url TEXT               OBS path for training data.
  --log-url TEXT                OBS path for training log.
  --code-dir TEXT               OBS path for source code.
  --output TEXT                 Training output parameter with OBS path.
  --input TEXT                  Training input parameter with OBS path.
  --env-variables TEXT          Env variables for training job.
  --parameters TEXT             Training job parameters (only keyword parameters are supported).
```

```
--boot-file TEXT                Training job boot file path behinds `code_dir`.
--framework-type TEXT           Training job framework type.
--framework-version TEXT        Training job framework version.
--workspace-id TEXT             The workspace where you submit training job(default "0")
--policy [regular|economic|turbo|auto]
                                Training job policy, default is regular.
--volumes TEXT                  Information about the volumes attached to the training job.
-q, --quiet                     Exit without waiting after submit successfully.
-C, --config-file PATH          Configure file path for authorization.
-D, --debug                     Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help                  Show this message and exit.
```

表 6-24 参数说明

参数名	参数类型	是否必选	参数说明
YAML_FILE	String	否	表示训练作业的配置文件，若不传则表示配置文件为空。
--code-dir	String	是	训练源代码的OBS路径。
--data-url	String	是	训练数据的OBS路径。
--log-url	String	是	存放训练生成日志的OBS路径。
--train-instance-count	String	是	训练作业计算节点个数，默认是1，表示单节点。
--boot-file	String	否	当使用自定义镜像或自定义命令时可以省略，当使用预置命令提交训练作业时需要指定该参数。
--name	String	否	训练任务名称。
--description	String	否	训练任务描述信息。
--image-url	String	否	自定义镜像SWR地址，遵循organization/image_name:tag
--uid	String	否	自定义镜像运行的UID，默认值1000。
--working-dir	String	否	运行算法时所在的工作目录。
--local-code-dir	String	否	算法的代码目录下载到训练容器内的本地路径。
--user-command	String	否	自定义镜像执行命令。需为/home下的目录。 当code-dir以file://为前缀时，当前字段不生效。

参数名	参数类型	是否必选	参数说明
--pool-id	String	否	训练作业选择的资源池ID。可在ModelArts管理控制台，单击左侧“专属资源池”，在专属资源池列表中查看资源池ID。
--train-instance-type	String	否	训练作业选择的资源规格。
--output	String	否	训练的输出信息，指定后，训练任务将会把训练脚本中指定输出参数对应训练容器的输出目录上传到指定的OBS路径。如果需要指定多个参数，可以使用--output output1=obs://bucket/output1 --output output2=obs://bucket/output2
--input	String	否	训练的输入信息，指定后，训练任务将会把对应OBS上的数据下载到训练容器，并将数据存储路径通过指定的参数传递给训练脚本。如果需要指定多个参数，可以使用--input data_path1=obs://bucket/data1 --input data_path2=obs://bucket/data2
--env-variables	String	否	训练时传入的环境变量，如果需要指定多个参数，可以使用--env-variables ENV1=env1 --env-variables ENV2=env2
--parameters	String	否	训练入参，可以通过--parameters "--epoch 0 --pretrained"指定多个参数。
--framework-type	String	否	训练作业选择的引擎规格。
--framework-version	String	否	训练作业选择的引擎版本。
-q / --quiet	Bool	否	提交训练任务成功后直接退出，不再同步打印作业状态。
--workspace-id	String	否	作业所处的工作空间，默认值为“0”。
--policy	String	否	训练资源规格模式，可选值regular、economic、turbo、auto。

参数名	参数类型	是否必选	参数说明
--volumes	String	否	挂载EFS，如果需要指定多个参数，可以使用--volumes。 "local_path=/xx/yy/zz;read_only=false;nfs_server_path=xxx.xxx.xxx.xxx:/" -volumes "local_path=/xxx/yyy/zzz;read_only=false;nfs_server_path=xxx.xxx.xxx.xxx:/"

基于 ModelArts 预置镜像提交训练作业

指定命令行options参数提交训练作业

```
ma-cli ma-job submit --code-dir obs://your-bucket/mnist/code/ \
                --boot-file main.py \
                --framework-type PyTorch \
                --working-dir /home/ma-user/modelarts/user-job-dir/code \
                --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 \
                --data-url obs://your-bucket/mnist/dataset/MNIST/ \
                --log-url obs://your-bucket/mnist/logs/ \
                --train-instance-type modelarts.vm.cpu.8u \
                --train-instance-count 1  \
                -q
```

使用预置镜像的train.yaml样例：

```
# .ma/train.yaml样例（预置镜像）
# pool_id: pool_xxxx
train-instance-type: modelarts.vm.cpu.8u
train-instance-count: 1
data-url: obs://your-bucket/mnist/dataset/MNIST/
code-dir: obs://your-bucket/mnist/code/
working-dir: /home/ma-user/modelarts/user-job-dir/code
framework-type: PyTorch
framework-version: pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
boot-file: main.py
log-url: obs://your-bucket/mnist/logs/

##[Optional] Uncomment to set uid when use custom image mode
uid: 1000

##[Optional] Uncomment to upload output file/dir to OBS from training platform
output:
   - name: output_dir
     obs_path: obs://your-bucket/mnist/output1/

##[Optional] Uncomment to download input file/dir from OBS to training platform
input:
   - name: data_url
     obs_path: obs://your-bucket/mnist/dataset/MNIST/

##[Optional] Uncomment pass hyperparameters
parameters:
   - epoch: 10
   - learning_rate: 0.01
   - pretrained:
```

```
##[Optional] Uncomment to use dedicated pool
pool_id: pool_xxxx

##[Optional] Uncomment to use volumes attached to the training job
volumes:
  - efs:
      local_path: /xx/yy/zz
      read_only: false
      nfs_server_path: xxx.xxx.xxx.xxx:/
```

基于自定义镜像创建训练作业

指定命令行options参数提交训练作业

```
ma-cli ma-job submit --image-url atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e \
                --code-dir obs://your-bucket/mnist/code/ \
                --user-command "export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH &&
cd /home/ma-user/modelarts/user-job-dir/code && /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/
python main.py" \
                --data-url obs://your-bucket/mnist/dataset/MNIST/ \
                --log-url obs://your-bucket/mnist/logs/ \
                --train-instance-type modelarts.vm.cpu.8u \
                --train-instance-count 1  \
                -q
```

使用自定义镜像的train.yaml样例：

```
# .ma/train.yaml样例（自定义镜像）
image-url: atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-
x86_64-20220926104358-041ba2e
user-command: export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH && cd /home/ma-
user/modelarts/user-job-dir/code && /home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python main.py
train-instance-type: modelarts.vm.cpu.8u
train-instance-count: 1
data-url: obs://your-bucket/mnist/dataset/MNIST/
code-dir: obs://your-bucket/mnist/code/
log-url: obs://your-bucket/mnist/logs/

##[Optional] Uncomment to set uid when use custom image mode
uid: 1000

##[Optional] Uncomment to upload output file/dir to OBS from training platform
output:
   - name: output_dir
     obs_path: obs://your-bucket/mnist/output1/

##[Optional] Uncomment to download input file/dir from OBS to training platform
input:
   - name: data_url
     obs_path: obs://your-bucket/mnist/dataset/MNIST/

##[Optional] Uncomment pass hyperparameters
parameters:
   - epoch: 10
   - learning_rate: 0.01
   - pretrained:

##[Optional] Uncomment to use dedicated pool
pool_id: pool_xxxx

##[Optional] Uncomment to use volumes attached to the training job
volumes:
  - efs:
      local_path: /xx/yy/zz
      read_only: false
      nfs_server_path: xxx.xxx.xxx.xxx:/
```

示例

- 基于yaml文件提交训练作业
  ma-cli ma-job submit ./train-job.yaml

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job submit ./train_job.yaml
[ OK ] Current training job id is: 4d7c8584-b213-4f88-9833-d3e6a82f9e42
[ OK ] Creating
[ OK ] Running
```

- 基于命令行和预置镜像pytorch1.8-cuda10.2-cudnn7-ubuntu18.04提交训练作业。

```
ma-cli ma-job submit --code-dir obs://automation-use-only/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/code/ \
                     --boot-file test-pytorch.py \
                     --framework-type PyTorch \
                     --working-dir /home/ma-user/modelarts/user-job-dir/code \
                     --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 \
                     --data-url obs://automation-use-only/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/ \
                     --log-url obs://automation-use-only/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/logs/ \
                     --train-instance-type modelarts.vm.cpu.8u \
                     --train-instance-count 1 \
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job submit --code-dir obs://au                    /Original/TrainJob/T                 ?/pytorch1.8.0_cuda10.2/code/ \
>                       --boot-file  test-pytorch.py \
>                       --framework-type PyTorch \
>                       --working-dir /home/ma-user/modelarts/user-job-dir/code \
>                       --framework-version pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 \
>                       --data-url obs://autom                    y/Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/ \
>                       --log-url obs://automa                    /Original/TrainJob/TrainJob-v2/pytorch1.8.0_cuda10.2/data/logs/ \
>                       --train-instance-type modelarts.vm.cpu.8u \
>                       --train-instance-count 1 \
>
[ OK ] Current training job id is: 7db3e6f9-181d-4142-ba13-235213499430
[ OK ] Creating
[ OK ] Running
```

6.7.6.4 查询 ModelArts 训练作业日志

执行**ma-cli ma-job get-log**命令查询ModelArts训练作业日志。

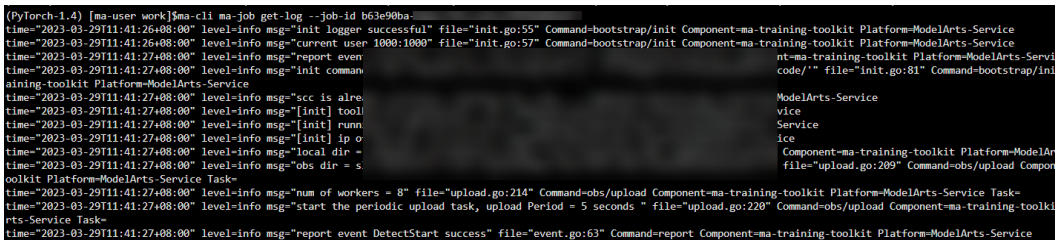
```
$ ma-cli ma-job get-log -h
Usage: ma-cli ma-job get-log [OPTIONS]

  Get job log details.

  Example:

  # Get job log by job id
  ma-cli ma-job get-log --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get training job details by job id.  [required]
  -t, --task-id TEXT      Get training job details by task id (default "worker-0").
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -h, -H, --help          Show this message and exit.
```

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	查询指定训练任务ID的任务日志。
-t / --task-id	String	否	查询指定task的日志，默认是work-0。

示例

查询指定训练任务ID的作业日志。

```
ma-cli ma-job get-log --job-id b63e90baxxx
```



6.7.6.5 查询 ModelArts 训练作业事件

执行**ma-cli ma-job get-event**命令查看ModelArts训练作业事件。

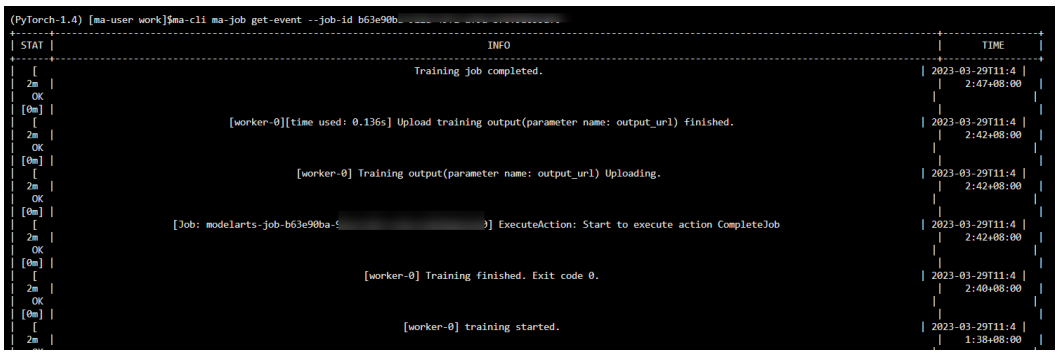
```
$ ma-cli ma-job get-event -h
Usage: ma-cli ma-job get-event [OPTIONS]

  Get job running event.

  Example:

  # Get training job running event
  ma-cli ma-job get-event --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get training job event by job id.  [required]
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	查询指定训练任务ID的事件。

示例

查看指定ID的训练作业的事件详情等。

```
ma-cli ma-job get-event --job-id b63e90baxxx
```



6.7.6.6 查询 ModelArts 训练 AI 引擎

执行**ma-cli ma-job get-engine**命令查询ModelArts训练使用的AI引擎。

```
$ ma-cli ma-job get-engine -h
Usage: ma-cli ma-job get-engine [OPTIONS]

  Get job engine info.

  Example:

  # Get training job engines
  ma-cli ma-job get-engine

Options:
  -v, --verbose           Show detailed information about training engines.
  -C, --config-file TEXT  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

表 6-25 参数说明

参数名	参数类型	是否必选	参数说明
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

查看训练作业的AI引擎。

```
ma-cli ma-job get-engine
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-engine
+-------+-----------------------------------------------------------+-----------------------+----------+
| index |                         engine id                         |      engine name      | run user |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   1   |                    caffe-1.0.0-python2.7                  |         Caffe         |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   2   |                    horovod-cp36-tf-1.16.2                 |        Horovod        |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   3   | horovod_0.20.0-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64 |        Horovod        |   1102   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   4   | horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64 |        Horovod        |   1102   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   5   |                 kungfu-0.2.2-tf-1.13.1-python3.6          |         KungFu        |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   6   |        mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64        |          MPI          |   1102   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   7   |       mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64       | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   8   |       mindspore_1.8.0-cann_5.1.2-py_3.7-euler_2.8.3-aarch64       | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   9   |       mindspore_1.9.0-cann_6.0.0-py_3.7-euler_2.8.3-aarch64       | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   10  |                    mxnet-1.2.1-python3.6                  |         MXNet         |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   11  |    optverse_0.2.0-pygrassland_1.1.0-py_3.7-ubuntu_18.04-x86_64    |           OR          |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   12  |                      pytorch-cp36-1.0.0                   |        PyTorch        |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   13  |                      pytorch-cp36-1.3.0                   |        PyTorch        |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   14  |                      pytorch-cp36-1.4.0                   |        PyTorch        |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   15  |       pytorch_1.8.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64         | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   16  |        pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64         |        PyTorch        |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   17  |       pytorch_1.8.1-cann_5.1.2-py_3.7-euler_2.8.3-aarch64         | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   18  |       pytorch_1.8.1-cann_6.0.0-py_3.7-euler_2.8.3-aarch64         | Ascend-Powered-Engine |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   19  |        pytorch_1.8.1-cuda_11.1-py_3.7-ubuntu_18.04-x86_64         |        PyTorch        |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   20  |        pytorch_1.8.2-cuda_10.2-py_3.7-ubuntu_18.04-x86_64         |        PyTorch        |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   21  |        pytorch_1.9.1-cuda_11.1-py_3.7-ubuntu_18.04-x86_64         |        PyTorch        |   1000   |
+-------+-----------------------------------------------------------+-----------------------+----------+
|   22  |                      ray-cp36-0.7.4                       |          Ray          |          |
+-------+-----------------------------------------------------------+-----------------------+----------+
```

6.7.6.7 查询 ModelArts 训练资源规格

执行**ma-cli ma-job get-flavor**命令查询ModelArts训练的资源规格。

```
$ ma-cli ma-job get-flavor -h
Usage: ma-cli ma-job get-flavor [OPTIONS]

  Get job flavor info.

  Example:

  # Get training job flavors
  ma-cli ma-job get-flavor

Options:
  -t, --flavor-type [CPU|GPU|Ascend]
                                  Type of training job flavor.
  -v, --verbose                   Show detailed information about training flavors.
  -C, --config-file TEXT          Configure file path for authorization.
  -D, --debug                     Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT              CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help                  Show this message and exit.
```

表 6-26 参数说明

参数名	参数类型	是否必选	参数说明
-t / --flavor-type	String	否	资源规格类型，若不指定默认返回所有的资源规格。
-v / --verbose	Bool	否	显示详细的信息开关，默认关闭。

示例

查看训练作业的资源规格及类型。

```
ma-cli ma-job get-flavor
```

```
(PyTorch-1.4) [ma-user work]$ma-cli ma-job get-flavor
+-------+-----------------------------+-----------------------------------+-------------+
| index |          flavor id          |            flavor name            | flavor type |
+-------+-----------------------------+-----------------------------------+-------------+
|   1   |    modelarts.kat1.8xlarge   | Computing NPU(8*Ascend) instance  |    Ascend   |
+-------+-----------------------------+-----------------------------------+-------------+
|   2   |    modelarts.kat1.xlarge    |  Computing NPU(Ascend) instance   |    Ascend   |
+-------+-----------------------------+-----------------------------------+-------------+
|   3   |     modelarts.vm.cpu.2u     |    Computing CPU(2U) instance     |     CPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   4   |     modelarts.vm.cpu.8u     |    Computing CPU(8U) instance     |     CPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   5   | modelarts.vm.cpu.8u16g.119  |    Computing CPU(8U) instance     |     CPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   6   |   modelarts.vm.v100.large   |   Computing GPU(V100) instance    |     GPU     |
+-------+-----------------------------+-----------------------------------+-------------+
|   7   | modelarts.vm.v100.large.free |   Computing GPU(V100) instance    |     GPU     |
+-------+-----------------------------+-----------------------------------+-------------+
```

6.7.6.8 停止 ModelArts 训练作业

执行**ma-cli ma-job stop**命令，可停止指定作业id的训练作业。

```
$ ma-cli ma-job stop -h
Usage: ma-cli ma-job stop [OPTIONS]

  Stop training job by job id.

  Example:

  Stop training job by job id
  ma-cli ma-job stop --job-id ${job_id}

Options:
  -i, --job-id TEXT       Get training job event by job id.  [required]
```

```
-y, --yes               Confirm stop operation.
-C, --config-file TEXT  Configure file path for authorization.
-D, --debug             Debug Mode. Shows full stack trace when error occurs.
-P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
-H, -h, --help          Show this message and exit.
```

表 6-27 参数说明

参数名	参数类型	是否必选	参数说明
-i / --job-id	String	是	ModelArts训练作业ID。
-y / --yes	Bool	否	强制关闭指定训练作业。

示例

停止运行中的训练作业。

```
ma-cli ma-job stop --job-id efd3e2f8xxx
```

6.7.7 使用 ma-cli 拷贝 OBS 数据

使用**ma-cli obs-copy [SRC] [DST]**可以实现本地和OBS文件或文件夹的相互拷贝。

```
$ma-cli obs-copy -h
Usage: ma-cli obs-copy [OPTIONS     ] SRC DST

  Copy file or directory to between OBS and local path. Example:

  # Upload local file to OBS path
  ma-cli obs-copy ./test.zip obs://your-bucket/copy-data/

  # Upload local directory to OBS path
  ma-cli obs-copy ./test/ obs://your-bucket/copy-data/

  # Download OBS file to local path
  ma-cli obs-copy obs://your-bucket/copy-data/test.zip ./test.zip

  # Download OBS directory to local path
  ma-cli obs-copy obs://your-bucket/copy-data/ ./test/

Options:
  -d, --drop-last-dir     Whether to drop last directory when copy folder. if True, the last directory of the
source folder will not copy to the destination folder.  [default: False]
  -C, --config-file PATH  Configure file path for authorization.
  -D, --debug             Debug Mode. Shows full stack trace when error occurs.
  -P, --profile TEXT      CLI connection profile to use. The default profile is "DEFAULT".
  -H, -h, --help          Show this message and exit.
```

表 6-28 参数说明

参数名	参数类型	是否必选	参数说明
-d / --drop-last-dir	Bool	否	若指定，在拷贝文件夹时不会将源文件夹最后一级目录拷贝至目的文件夹下，仅对文件夹拷贝有效。

命令示例

上传文件到OBS中

```
$ ma-cli obs-copy ./test.csv obs://${your_bucket}/test-copy/
[ OK ] local src path: [ /home/ma-user/work/test.csv ]
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

上传文件夹到OBS中，对应上传到OBS的目录为obs://${your_bucket}/test-copy/data/

```
$ ma-cli obs-copy /home/ma-user/work/data/ obs://${your_bucket}/test-copy/
[ OK ] local src path: [ /home/ma-user/work/data/ ]
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

上传文件夹到OBS中，并指定--drop-last-dir，对应上传到OBS的目录为obs://${your_bucket}/test-copy/

```
$ ma-cli obs-copy /home/ma-user/work/data/ obs://${your_bucket}/test-copy/ --drop-last-dir
[ OK ] local src path: [ /home/ma-user/work/data ]
[ OK ] obs dst path: [ obs://${your_bucket}/test-copy/ ]
```

从OBS下载文件夹到本地磁盘中

```
$ ma-cli obs-copy obs://${your_bucket}/test-copy/ ~/work/test-data/
[ OK ] obs src path: [ obs://${your_bucket}/test-copy/ ]
[ OK ] local dst path: [ /home/ma-user/work/test-data/ ]
```

7 训练管理

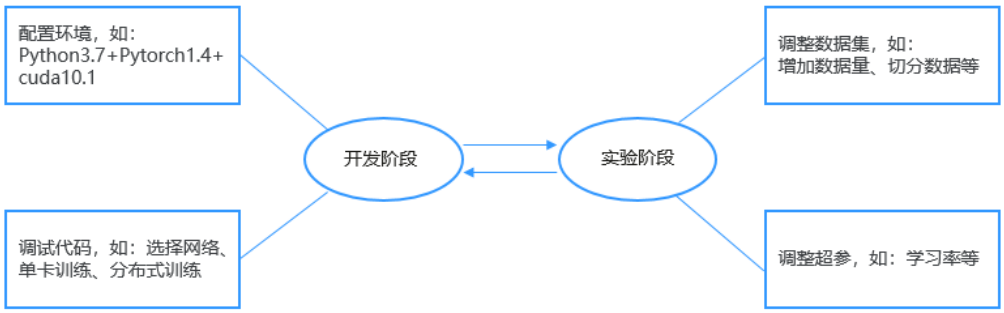
7.1 模型开发简介

AI模型开发的过程，称之为Modeling，一般包含两个阶段：

- 开发阶段：准备并配置环境，调试代码，使代码能够开始进行深度学习训练，推荐在ModelArts开发环境中调试。
- 实验阶段：调整数据集、调整超参等，通过多轮实验，训练出理想的模型，推荐在ModelArts训练中进行实验。

两个过程可以相互转换。如开发阶段代码稳定后，则会进入实验阶段，通过不断尝试调整超参来迭代模型；或在实验阶段，有一个可以优化训练的性能的想法，则会回到开发阶段，重新优化代码。

图 7-1 模型开发过程



ModelArts提供了模型训练的功能，方便您查看训练情况并不断调整您的模型参数。您还可以基于不同的数据，选择不同规格的资源池用于模型训练。

请参考以下指导在ModelArts上训练模型：

- 将已标注的数据上传至OBS服务使用，请参考**准备数据**。
- 训练模型的算法实现与指导请参考**准备算法**章节。
- 使用控制台创建训练作业请参考**创建训练作业**章节。

- 关于训练作业日志、训练资源占用等详情请参考**查看训练作业日志**。
- 停止或删除模型训练作业，请参考**停止、重建或查找作业**。
- 如果您在训练过程中遇到问题，文档中提供了部分故障案例供参考，请参考**训练故障排查**。

7.2 准备数据

ModelArts使用对象存储服务（Object Storage Service，简称OBS）进行数据存储以及模型的备份和快照，实现安全、高可靠和低成本的存储需求。

- **OBS简介**
- **使用训练数据的两种方式**

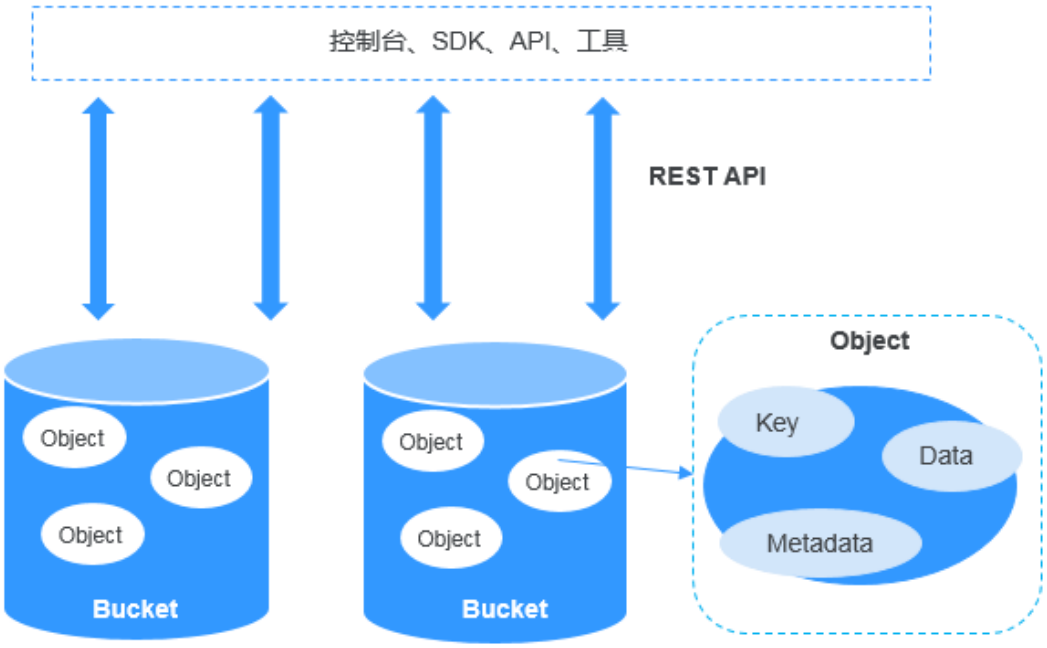
OBS 简介

对象存储服务OBS是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力。对象存储服务OBS的基本组成是桶和对象。桶是OBS中存储对象的容器，每个桶都有自己的存储类别、访问权限、所属区域等属性，用户在互联网上通过桶的访问域名来定位桶。对象是OBS中数据存储的基本单位。

对ModelArts来说，obs服务是一个数据存储中心。AI开发过程中的输入数据、输出数据、中间缓存数据都可以在obs桶中进行存储、读取。

因此，在使用ModelArts之前您需要**创建一个OBS桶**，然后在OBS桶中创建文件夹用于存放数据。

图 7-2 对象存储服务 OBS



使用训练数据的两种方式

ModelArts模型训练支持2种读取训练数据的方式：

- 使用OBS桶中存储的数据集

  如果您的数据集已完成数据标注和数据预处理，可以将数据上传至OBS桶。当创建训练作业时，在训练输入参数位置填写训练数据所在的OBS桶路径即可完成训练配置。

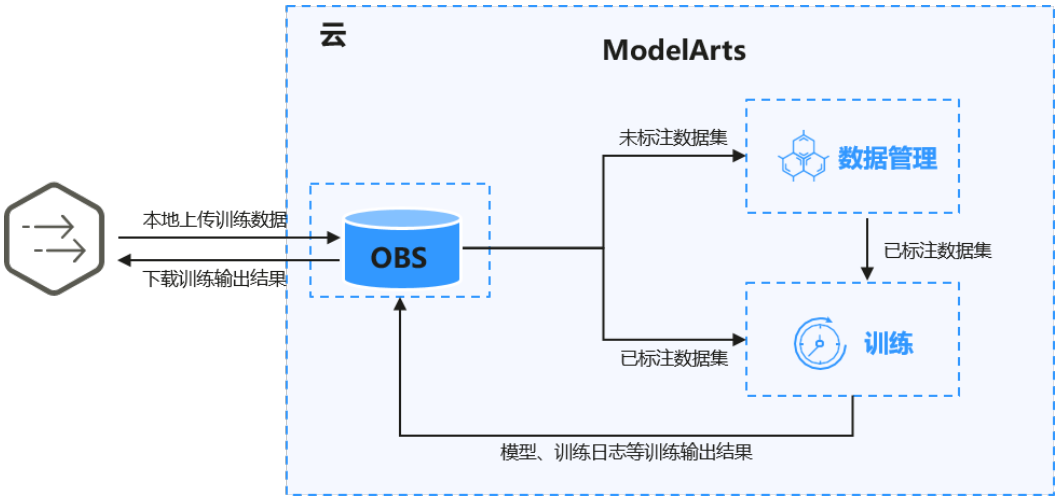
- 使用数据管理中的数据集

  如果您的数据集未标注或者需要进一步的数据预处理，可以将数据导入ModelArts数据管理模块进行数据预处理。

说明

ModelArts数据管理模块在重构升级中，对未使用过数据管理的用户不可见。建议新用户将训练数据存放至OBS桶中使用。

图 7-3 准备数据



7.3 准备算法

7.3.1 准备算法简介

机器学习从有限的观测数据中学习一般性的规律，并利用这些规律对未知的数据进行预测。为了获取更准确的预测结果，用户需要选择一个合适的算法来训练模型。针对不同的场景，ModelArts提供大量的算法样例。以下章节提供了关于业务场景、算法学习方式、算法实现方式的指导。

选择算法的实现方式

ModelArts提供如下方式实现模型训练。

- 使用预置框架

  如果您需要使用自己开发的算法，可以选择使用ModelArts预置框架。ModelArts支持了大多数主流的AI引擎，详细请参见**预置训练引擎**。这些预置引擎预加载了一些额外的python包，例如numpy等；也支持您通过在代码目录中使用“requirements.txt”文件安装依赖包。使用预置框架创建训练作业请参考**使用预置框架（自定义脚本）**指导。

- 使用自定义镜像（新版训练请参考**使用自定义镜像训练模型**）

订阅算法和预置框架涵盖了大部分的训练场景。针对特殊场景，ModelArts支持用户构建自定义镜像用于模型训练。自定义镜像需上传至容器镜像服务（SWR），才能用于ModelArts上训练。由于自定义镜像的制作要求用户对容器相关知识有比较深刻的了解，除非订阅算法和预置引擎无法满足需求，否则不推荐使用。

选择算法的学习方式

ModelArts支持用户根据实际需求进行不同方式的模型训练。

- 离线学习

  离线学习是训练中最基本的方式。离线学习需要一次性提供训练所需的所有数据，在训练完成后，目标函数的优化就停止了。使用离线学习的优势是模型稳定性高，便于做模型的验证与评估。

- 增量学习

  增量学习是一个连续不断的学习过程。相较于离线学习，增量学习不需要一次性存储所有的训练数据，缓解了存储资源有限的问题；另一方面，增量学习节约了重新训练中需要消耗大量算力、时间以及经济成本。

7.3.2 使用预置框架（自定义脚本）

7.3.2.1 使用预置框架简介

如果订阅算法不能满足需求或者用户希望迁移本地算法至ModelArts上训练，可以考虑使用ModelArts支持的预置框架实现算法构建。这种方式在创建算法时被称为“使用预置框架”模式。

以下章节介绍了如何使用预置框架创建算法。

- 如果需要了解ModelArts模型训练支持的预置引擎和模型，请参考**预置的训练引擎**。
- 本地开发的算法迁移至ModelArts需要做代码适配，如何适配请参考**开发自定义脚本**章节。
- 通过ModelArts控制台界面使用预置框架创建算法可以参考**创建算法**章节。

预置的训练引擎

当前ModelArts支持的训练引擎及对应版本如下所示。

说明

不同区域支持的AI引擎有差异，请以实际环境为准。

表 7-1 训练作业支持的 AI 引擎

工作环境	系统架构	系统版本	AI引擎与版本	支持的cuda或Ascend版本
Ascend-Powered-Engine	aarch64	Euler2.8	mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0

工作环境	系统架构	系统版本	AI引擎与版本	支持的cuda或Ascend版本
PyTorch	aarch64	Euler2.8	pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0
TensorFlow	aarch64	Euler2.8	tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0

7.3.2.2 开发自定义脚本

当您使用预置框架创建算法时，您需要提前完成算法的代码开发。本章详细介绍如何改造本地代码以适配ModelArts上的训练。

创建算法时，您需要在创建页面提供代码目录路径、代码目录路径中的启动文件、训练输入路径参数和训练输出路径参数。这四种输入搭建了用户代码和ModelArts后台交互的桥梁。

- 代码目录路径

  您需要在OBS桶中指定代码目录，并将训练代码、依赖安装包或者预生成模型等训练所需文件上传至该代码目录下。训练作业创建完成后，ModelArts会将代码目录及其子目录下载至后台容器中。

  例如：OBS路径“obs://obs-bucket/training-test/demo-code”作为代码目录，OBS路径下的内容会被自动下载至训练容器的“${MA_JOB_DIR}/demo-code”目录中，demo-code为OBS存放代码路径的最后一级目录，用户可以根据实际修改。

  请注意不要将训练数据放在代码目录路径下。训练数据比较大，训练代码目录在训练作业启动后会下载至后台，可能会有下载失败的风险。建议训练代码目录大小小于或等于50MB。

- 代码目录路径中的启动文件

  代码目录路径中的启动文件作为训练启动的入口，当前只支持python格式。

- 训练输入路径参数

  训练数据需上传至OBS桶或者存储至数据集中。在训练代码中，用户需解析**输入路径参数**。系统后台会自动下载输入参数路径中的训练数据至训练容器的本地目录。请保证您设置的桶路径有读取权限。在训练作业启动后，ModelArts会挂载硬盘至“/cache”目录，用户可以使用此目录来存储临时文件。“/cache”目录大小请参考**训练环境中不同规格资源“/cache”目录的大小**。

- 训练输出路径参数

  建议设置一个空目录为训练输出路径。在训练代码中，您需要解析**输出路径参数**。系统后台会自动上传训练输出至指定的训练输出路径，请保证您设置的桶路径有写入权限和读取权限。

在ModelArts中，训练代码需包含以下步骤：

（可选）引入依赖

1. 当您使用自定义脚本创建算法的时候，如果您的模型引用了其他依赖，您需要在“算法管理 > 创建算法”的“代码目录”下放置相应的文件或安装包。
   - 安装python依赖包请参考**模型中引用依赖包时，如何创建训练作业?**
   - 安装C++的依赖库请参考**如何安装C++的依赖库?**
   - 在预训练模型中加载参数请参考常见问题中的**如何在训练中加载部分训练好的参数?**

解析输入路径参数、输出路径参数

运行在ModelArts的模型读取存储在OBS服务的数据，或者输出至OBS服务指定路径，输入和输出数据需要配置2个地方：

1. 训练代码中需解析输入路径参数和输出路径参数。ModelArts推荐以下方式实现参数解析。

```
import argparse
# 创建解析
parser = argparse.ArgumentParser(description='train mnist')

# 添加参数
parser.add_argument('--data_url', type=str, default="./Data/mnist.npz", help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='path where the model is saved')

# 解析参数
args = parser.parse_args()
```

   完成参数解析后，用户使用“data_url”、“train_url”代替算法中数据来源和数据输出所需的路径。

2. 在创建训练作业时，填写输入路径和输出路径。

   训练输入选择对应的OBS路径或者数据集路径；训练输出选择对应的OBS路径。

训练代码正文和保存模型

训练代码正文和保存模型涉及的代码与您使用的AI引擎密切相关，以下案例以Tensorflow框架为例。案例中使用到的“mnist.npz”文件需要提前**下载**并上传至OBS桶中，训练输入为“mnist.npz”所在OBS路径。

```
import os
import argparse
import tensorflow as tf

parser = argparse.ArgumentParser(description='train mnist')
parser.add_argument('--data_url', type=str, default="./Data/mnist.npz", help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='path where the model is saved')
args = parser.parse_args()

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data(args.data_url)
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])
```

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)

model.save(os.path.join(args.train_url, 'model'))
```

7.3.2.3 创建算法

针对您在本地或使用其他工具开发的算法，支持上传至ModelArts中统一管理。在创建自定义算法过程中，您需要关注以下内容：

1. **前提条件**
2. **进入创建算法页面**
3. **设置算法基本信息**
4. **设置算法启动方式**
5. **输入输出管道设置**
6. **定义超参**
7. **支持的策略**
8. **添加训练约束**
9. **运行环境预览**
10. **后续操作**

前提条件

- 数据已完成准备：已在ModelArts中创建可用的数据集，或者您已将用于训练的数据集上传至OBS目录。
- 请准备好训练脚本，并上传至OBS目录。训练脚本开发指导参见**开发自定义脚本**。
- 已在OBS创建至少1个空的文件夹，用于存储训练输出的内容。

进入创建算法页面

1. 登录ModelArts管理控制台，单击左侧菜单栏的“算法管理”。
2. 在“我的算法”管理页面，单击“创建”，进入“创建算法”页面。

设置算法基本信息

填写算法的基本信息，包含“名称”和“描述”。

设置算法启动方式

选择“预置框架”创建算法。

用户需根据实际算法代码情况设置“镜像”、“代码目录”和“启动文件”。选择的AI镜像和编写算法代码时选择的框架必须一致。例如编写算法代码使用的是TensorFlow，则在创建算法时也要选择TensorFlow镜像。

表 7-2 启动方式参数说明

参数	说明
“启动方式-预置框架”	选择算法使用的预置框架引擎和引擎版本。
“代码目录”	算法代码存储的OBS路径。训练代码、依赖安装包或者预生成模型等训练所需文件上传至该代码目录下。 请注意不要将训练数据放在代码目录路径下。训练数据比较大，训练代码目录在训练作业启动后会下载至后台，可能会有下载失败的风险。 训练作业创建完成后，ModelArts会将代码目录及其子目录下载至训练后台容器中。 例如：OBS路径“obs://obs-bucket/training-test/demo-code”作为代码目录，OBS路径下的内容会被自动下载至训练容器的“${MA_JOB_DIR}/demo-code”目录中，demo-code为OBS存放代码路径的最后一级目录，用户可以根据实际修改。 **说明** • 编程语言不限。 • 文件数（含文件、文件夹数量）小于或等于1000个。 • 文件总大小小于或等于5GB。
“启动文件”	必须为“代码目录”下的文件，且以“.py”结尾，即ModelArts目前只支持使用Python语言编写的启动文件。 代码目录路径中的启动文件为训练启动的入口。

输入输出管道设置

训练过程中，基于预置框架的算法需要从OBS桶或者数据集中获取数据进行模型训练，训练产生的输出结果也需要存储至OBS桶中。用户的算法代码中需解析输入输出参数实现ModelArts后台与OBS的数据交互，用户可以参考**开发自定义脚本**完成适配ModelArts训练的代码开发。

创建基于预置框架的算法时，用户需要配置算法代码中定义的输入输出参数。

- 输入配置

  表 7-3 输入配置

参数	参数说明
参数名称	根据实际代码中的输入数据参数定义此处的名称。此处设置的代码路径参数必须与算法代码中解析的训练输入数据参数保持一致，否则您的算法代码无法获取正确的输入数据。 例如，算法代码中使用argparse解析的data_url作为输入数据的参数，那么创建算法时就需要配置输入数据的参数名称为“data_url”。
描述	输入参数的说明，用户可以自定义描述。

参数	参数说明
获取方式	输入参数的获取方式，默认使用“超参”，也可以选择“环境变量”。
输入约束	开启后，用户可以根据实际情况限制数据输入来源。输入来源可以选择“数据存储位置”或者“ModelArts数据集”。 如果用户选择数据来源为ModelArts数据集，还可以约束以下三种： • 标注类型。数据类型请参考**标注数据**。 • 数据格式。可选“Default”和“CarbonData”，支持多选。其中“Default”代表Manifest格式。 • 数据切分。仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。可选“仅支持切分的数据集”、“仅支持未切分数据集”和“无限制”。数据切分详细内容可参考**发布数据版本**。
添加	用户可以根据实际算法添加多个输入数据来源。

- 输出配置

表 7-4 输出配置

参数	参数说明
参数名称	根据实际代码中的训练输出参数定义此处的名称。此处设置的代码路径参数必须与算法代码中解析的训练输出参数保持一致，否则您的算法代码无法获取正确的输出路径。 例如，算法代码中使用argparse解析的train_url作为训练输出数据的参数，那么创建算法时就需要配置输出数据的参数名称为“train_url”。
描述	输出参数的说明，用户可以自定义描述。
获取方式	输出参数的获取方式，默认使用“超参”，也可以选择“环境变量”。
添加	用户可以根据实际算法添加多个输出数据路径。

定义超参

使用预置框架创建算法时，ModelArts支持用户自定义超参，方便用户查阅或修改。定义超参后会体现在启动命令中，以命令行参数的形式传入您的启动文件中。

1. 导入超参

   您可以单击“增加超参”手动添加超参。

2. 编辑超参

   超参的参数说明参见**表7-5**。

表 7-5 超参编辑参数

参数	说明
名称	填入超参名称。 超参名称支持64个以内字符，仅支持大小写字母、数字、下划线和中划线。
类型	填入超参的数据类型。支持String、Integer、Float和Boolean。
默认值	填入超参的默认值。创建训练作业时，默认使用该值进行训练。
约束	单击“约束”。在弹出对话框中，支持用户设置默认值的取值范围或者枚举值范围。
必需	选择是或否。 ● 选择否，则在使用该算法创建训练作业时，支持在创建训练作业页面删除该超参。 ● 选择是，则在使用该算法创建训练作业时，不支持在创建训练作业页面删除该超参。
描述	填入超参的描述说明。 超参描述支持大小写字母、数字、空格、中划线、下划线、逗号和句号。

支持的策略

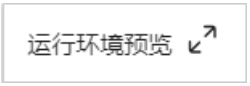
自动搜索目前仅支持pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64，tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64镜像。

添加训练约束

用户可以根据实际情况定义此算法的训练约束。

- 资源类型：选择适用的资源类型，支持多选。
- 多卡训练：选择是否支持多卡训练。
- 分布式训练：选择是否支持分布式训练。

运行环境预览

创建算法时，可以打开创建页面右下方的运行环境预览窗口 

，辅助您了解代码目录、启动文件、输入输出等数据配置在训练容器中的路径。

后续操作

创建算法完成后，可以使用算法创建训练作业，详细操作请参见**创建训练作业**。

7.3.3 使用自定义镜像

订阅算法和预置框架涵盖了大部分的训练场景。针对特殊场景，ModelArts支持用户构建自定义镜像用于模型训练。

自定义镜像的制作要求用户对容器相关知识有比较深刻的了解，除非订阅算法和预置框架无法满足需求，否则不推荐使用。自定义镜像需上传至容器镜像服务（SWR），才能用于ModelArts上训练。

ModelArts上使用自定义镜像训练支持2种方式：

- 使用预置框架 + 自定义镜像：

  如果先前基于预置框架且通过指定代码目录和启动文件的方式来创建的训练作业；但是随着业务逻辑的逐渐复杂，您期望可以基于预置框架修改或增加一些软件依赖的时候，此时您可以使用预置框架 + 自定义镜像的功能，即选择预置框架名称后，在预置框架版本下拉列表中选择“自定义”。
- 完全自定义镜像：

  用户遵循ModelArts镜像的规范要求制作镜像，选择自己的镜像，并且通过指定代码目录（可选）和启动命令的方式来创建的训练作业。

  说明

  当使用完全自定义镜像创建训练作业时，“启动命令”必须在“/home/ma-user”目录下执行，否则训练作业可能会运行异常。

使用预置框架 + 自定义镜像

此功能与直接基于预置框架创建训练作业的区别仅在于，镜像是由用户自行选择的。用户可以基于预置框架制作自定义镜像。基于预置框架制作自定义镜像可参考**使用基础镜像构建新的训练镜像**章节。

图 7-4 使用预置框架+自定义镜像创建算法

该功能的行为与直接基于预置框架创建的训练作业相同，例如：

- 系统将会自动注入一系列环境变量
  - PATH=${MA_HOME}/anaconda/bin:${PATH}
  - LD_LIBRARY_PATH=${MA_HOME}/anaconda/lib:${LD_LIBRARY_PATH}
  - PYTHONPATH=${MA_JOB_DIR}:${PYTHONPATH}
- 您选择的启动文件将会被系统自动以python命令直接启动，因此请确保镜像中的Python命令为您预期的Python环境。注意到系统自动注入的PATH环境变量，您可以参考下述命令确认训练作业最终使用的Python版本：
  - export MA_HOME=/home/ma-user; docker run --rm {image} ${MA_HOME}/anaconda/bin/python -V

  - docker run --rm {image} $(which python) -V
- 系统将会自动添加预置框架关联的超参

完全使用自定义镜像

图 7-5 完全使用自定义镜像创建算法

新版训练支持的自定义镜像使用说明，请参考**使用自定义镜像创建训练作业**。

完全使用自定义镜像场景下，指定的“conda env”启动训练方法如下：

由于训练作业运行时不是shell环境，因此无法直接使用“conda activate”命令激活指定的 “conda env”，需要使用其他方式以达成使用指定“conda env”来启动训练的效果。

假设您的自定义镜像中的“conda”安装于“/home/ma-user/anaconda3”目录“conda env”为“python-3.7.10”，训练脚本位于“/home/ma-user/modelarts/user-job-dir/code/train.py”。可通过以下方式使用指定的“conda env”启动训练：

- 方式一：为镜像设置正确的“DEFAULT_CONDA_ENV_NAME”环境变量与“ANACONDA_DIR”环境变量。

  您可以使用Python命令启动训练脚本。启动命令示例如下：
  ```
  python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```
- 方式二：使用“conda env python”的绝对路径。

  您可以使用“/home/ma-user/anaconda3/envs/python-3.7.10/bin/python”命令启动训练脚本。启动命令示例如下：
  ```
  /home/ma-user/anaconda3/envs/python-3.7.10/bin/python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```
- 方式三：设置PATH环境变量。

  您可以将指定的“conda env bin”目录配置到PATH环境变量中。您可以使用Python命令启动训练脚本。启动命令示例如下：
  ```
  export PATH=/home/ma-user/anaconda3/envs/python-3.7.10/bin:$PATH; python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```
- 方式四：使用“conda run -n”命令。

  您可以使用“/home/ma-user/anaconda3/bin/conda run -n python-3.7.10”命令来执行训练命令，启动命令示例如下：

```
/home/ma-user/anaconda3/bin/conda run -n python-3.7.10 python /home/ma-user/modelarts/user-job-dir/code/train.py
```

说明

如果在训练时发生找不到“$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib ”目录下“.so”文件的相关报错，可以尝试将该目录加入到“LD_LIBRARY_PATH”，将以下命令放在上述启动方式命令前：

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib:$LD_LIBRARY_PATH;
```

例如，方式一的启动命令示例此时变为：

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib:$LD_LIBRARY_PATH; python /home/ma-user/modelarts/user-job-dir/code/train.py
```

7.3.4 查看算法详情

1. 登录ModelArts管理控制台。
2. 在左侧导航栏中，选择“算法管理”，进入“我的算法”页面。
3. 在“我的算法”列表中，单击算法名称进入详情页。
   - 选择“基本信息”页签可以查看算法信息。

     表 7-6 算法基本信息

参数	说明
名称	算法的名称。
ID	算法的唯一标识。
描述	算法的简介。 单击编辑图标，可以更新描述。
预置框架	算法使用的预置框架引擎及版本。只有使用预置框架创建的算法才有该参数。
自定义镜像	算法使用的容器镜像。只有使用自定义引擎版本或自定义镜像创建的算法才有该参数。
代码目录	算法代码存放的OBS目录。
启动文件	启动文件的OBS存放目录。
输入	算法的输入参数。
输出	算法的输出参数。
超参	算法的超参信息。
支持的策略	算法的自动搜索策略。如果空白即表示不支持自动搜索，否则显示自动搜索的参数信息。
训练约束	算法的训练约束信息，如果显示“否”则表示不存在约束，否则会显示支持的资源类型和训练场景的支持情况。

   - 选择“训练列表”页签可以查看使用该算法的训练作业信息，例如训练作业名称、状态。

4. “基本信息”页签，单击“编辑”，支持修改除名称和ID之外的算法信息。修改完成，单击“保存”即可完成修改。

7.3.5 查找算法

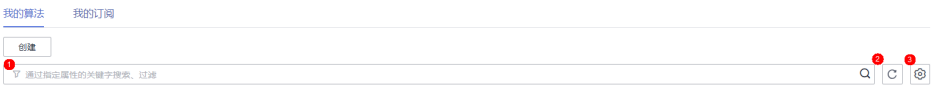
ModelArts提供查找算法功能帮助用户快速查找算法。

操作一：按照名称、镜像、代码目录、描述、创建时间筛选的高级搜索。

操作二：单击右上角“刷新”图标，刷新算法列表。

操作三：自定义列功能设置。

图 7-6 查找算法



如果需要对算法排序，可单击表头中的 箭头进行排序。

7.3.6 删除算法

删除我的算法

在“算法管理 > 我的算法”页面，“删除”运行结束的训练作业。您可以单击“操作”列的“删除”，在弹出的提示框中单击“确认”，删除对应的算法。

7.4 完成一次训练

7.4.1 创建训练作业

模型训练是一个不断迭代和优化模型权重的过程。ModelArts的训练模块支持创建训练作业、查看训练情况以及管理训练版本。通过模型训练试验模型结构、数据和超参的各种组合，便于找到最佳的模型结构和权重。

前提条件

- 已经将用于训练作业的数据上传至OBS目录。
- 已经在OBS目录下创建了至少1个空的文件夹，用于存储训练输出的内容。

  说明

  ModelArts不支持加密的OBS桶，创建OBS桶时，请勿开启桶加密。
- 检查是否配置了访问授权。如果未配置，请参见**使用委托授权**完成操作。
- （可选）如果使用已有算法创建训练作业，需要确认“算法管理”中已准备好算法，具体操作请参见**准备算法简介**。
- （可选）如果使用自定义镜像创建训练作业，需要上传镜像到SWR服务中，具体操作请参见**如何登录并上传镜像到SWR**。

操作流程介绍

创建训练作业的操作步骤如下所示。

步骤1 **进入创建训练作业页面**。

步骤2 **配置训练作业基本信息**。

步骤3 根据不同的算法来源，选择不同的训练作业创建方式。

- 使用预置镜像创建训练作业：**选择创建方式（使用预置镜像）**
- 使用自定义镜像创建训练作：**选择创建方式（使用自定义镜像）**
- 使用已有算法创建训练作业：**选择创建方式（使用我的算法）**

步骤4 **配置训练参数**：配置训练作业的输入、输出、超参、环境变量等参数。

步骤5 根据需要选择不同的资源池用于训练作业，推荐使用专属资源池。

- **配置资源池（公共资源池）**
- **配置资源池（专属资源池）**

步骤6 **（可选）选择训练模式**：当训练作业的算法框架选用的是MindSpore类引擎、资源池类型选用的是Ascend资源时，支持选择训练模式。

步骤7 **后续操作**。

----结束

进入创建训练作业页面

1. 登录ModelArts管理控制台。
2. 在左侧导航栏中，选择“训练管理 > 训练作业”进入训练作业列表。
3. 单击“创建训练作业”，进入创建训练作业页面。

配置训练作业基本信息

在创建训练作业页面填写训练作业基本信息。

表 7-7 创建训练作业的基本信息

参数名称	说明
名称	必填，训练作业的名称。 系统会自动生成一个名称，可以根据业务需求重新命名，命名规则如下： ● 支持1~64位字符。 ● 可以包含大小写字母、数字、中划线（-）或下划线（_）。
描述	训练作业的简介，便于在训练作业列表了解作业信息。

参数名称	说明
设置实验	将训练作业分类有序地放入实验中进行管理。 ● 如果选择“纳入新实验”，需要配置“新建实验名称”和“新建实验描述”。 ● 如果选择“纳入已有实验”，需要选择“实验名称”。 ● 如果选择“不纳入实验”，则不在实验中进行统一管理。

选择创建方式（使用预置镜像）

如果选择使用预置镜像创建训练作业，则参考表7-8选择训练作业的创建方式。

表 7-8 创建训练作业的创建方式（使用预置镜像）

参数名称	说明
创建方式	必选，选择“自定义算法”。
启动方式	必选，选择“预置框架”，并选择训练作业要使用的预置框架引擎和引擎版本。 如果引擎版本选择“自定义”，则需要配置“镜像”参数，选择自定义镜像用于训练作业。
镜像	仅当预置框架的引擎版本选择“自定义”时才显示该参数，且是必填参数。 容器镜像地址的填写支持如下方式。 ● 选择自有镜像或他人共享的镜像：单击右边的“选择”，从容器镜像中选择用于训练的容器镜像。所需镜像需要提前上传到SWR服务中。 ● 选择公开镜像：直接输入SWR服务中公开镜像的地址。地址直接填写“组织名称/镜像名称:版本名称”，不需要带域名信息（swr.<region>.xxx.com），系统会自动拼接域名地址。例如：某公开镜像的SWR地址为“swr.<region>.xxx.com/test-image/tensorflow2_1_1:1.1.1”，则此处填写“test-images/tensorflow2_1_1:1.1.1”。
代码目录	必填，选择训练代码文件所在的OBS目录。 ● 需要提前将代码上传至OBS桶中，目录内文件总大小要小于或等于5GB，文件数要小于或等于1000个，文件深度要小于或等于32。 ● 训练代码文件会在训练作业启动的时候被系统自动下载到训练容器的“${MA_JOB_DIR}/demo-code”目录中，“demo-code”为存放代码目录的最后一级OBS目录。例如，“代码目录”选择的是“/test/code”，则训练代码文件会被下载到训练容器的“${MA_JOB_DIR}/code”目录中。

参数名称	说明
启动文件	必填，选择代码目录中训练作业的Python启动脚本。 ModelArts只支持使用Python语言编写的启动文件，因此启动文件必须以“.py”结尾。
本地代码目录	指定训练容器的本地目录，启动训练时系统会将代码目录下载至此目录。 此参数可选，默认本地代码目录为“/home/ma-user/modelarts/user-job-dir”。
工作目录	训练时，系统会自动cd到此目录下执行启动文件。

选择创建方式（使用自定义镜像）

如果选择使用自定义镜像创建训练作业，则参考表7-9选择训练作业的创建方式。

表 7-9 创建训练作业的创建方式（使用自定义镜像）

参数名称	说明
创建方式	必选，选择“自定义算法”。
启动方式	必选，选择“自定义”。
镜像	必填，填写容器镜像的地址。 容器镜像地址的填写支持如下方式。 • 选择自有镜像或他人共享的镜像：单击右边的“选择”，从容器镜像中选择用于训练的容器镜像。所需镜像需要提前上传到SWR服务中。 • 选择公开镜像：直接输入SWR服务中公开镜像的地址。地址直接填写“组织名称/镜像名称:版本名称”，不需要带域名信息（swr.<region>.xxx.com），系统会自动拼接域名地址。例如：某公开镜像的SWR地址为“swr.<region>.xxx.com/test-image/tensorflow2_1_1:1.1.1”，则此处填写“test-images/tensorflow2_1_1:1.1.1”。
代码目录	选择训练代码文件所在的OBS目录。如果自定义镜像中不含训练代码则需要配置该参数，如果自定义镜像中已包含训练代码则不需要配置。 • 需要提前将代码上传至OBS桶中，目录内文件总大小要小于或等于5GB，文件数要小于或等于1000个，文件深度要小于或等于32。 • 训练代码文件会在训练作业启动的时候被系统自动下载到训练容器的“${MA_JOB_DIR}/demo-code”目录中，“demo-code”为存放代码目录的最后一级OBS目录。例如，“代码目录”选择的是“/test/code”，则训练代码文件会被下载到训练容器的“${MA_JOB_DIR}/code”目录中。

参数名称	说明
运行用户ID	容器运行时的用户ID，该参数为选填参数，建议使用默认值1000。 如果需要指定uid，则uid数值需要在规定范围内，不同资源池的uid范围如下： ● 公共资源池：1000-65535 ● 专属资源池：0-65535
启动命令	必填，镜像的启动命令。 运行训练作业时，当“代码目录”下载完成后，“启动命令”会被自动执行。 ● 如果训练启动脚本用的是py文件，例如“train.py”，则启动命令如下所示。 python ${MA_JOB_DIR}/demo-code/train.py ● 如果训练启动脚本用的是sh文件，例如“main.sh”，则启动命令如下所示。 bash ${MA_JOB_DIR}/demo-code/main.sh 启动命令支持使用“;”和“&&”拼接多条命令，命令中的“demo-code”为存放代码目录的最后一级OBS目录，以实际情况为准。
本地代码目录	指定训练容器的本地目录，启动训练时系统会将代码目录下载至此目录。 此参数可选，默认本地代码目录为“/home/ma-user/modelarts/user-job-dir”。
工作目录	训练时，系统会自动cd到此目录下执行启动文件。

选择创建方式（使用我的算法）

如果选择使用已有算法创建训练作业，则“创建方式”选择“我的算法”，在算法列表中选择算法。如果没有满足条件的算法，也可以新建算法，具体操作请参见**创建算法**。

配置训练参数

训练过程中需要从OBS桶或者数据集中获取输入数据进行模型训练，训练输出的结果也要存储至OBS桶中。创建训练作业时可以参考**表7-10**配置输入、输出、超参、环境变量等参数。

说明

创建训练作业时选择的创建方式不同，训练作业的输入、输出和超参显示不同。如果参数值置灰，即表示该参数已经在算法代码中配置了且不支持修改。

表 7-10 创建训练作业的训练参数

参数名称	子参数	说明
输入	参数名称	算法代码需要通过“输入”的“参数名称”去读取训练的输入数据。 建议设置为“data_url”。训练输入参数要与所选算法的“输入”参数匹配，请参见创建算法时的表7-3。
	数据集	单击“数据集”，在ModelArts数据集列表中勾选目标数据集并选择对应的版本。 训练启动时，系统将自动下载输入路径中的数据到训练运行容器。 **说明** ModelArts数据管理模块在重构升级中，对未使用过数据管理的用户不可见。建议新用户将训练数据存放至OBS桶中使用。
	数据存储位置	单击“数据存储位置”，从OBS桶中选择训练输入数据的存储位置。 训练启动时，系统将自动下载输入路径中的数据到训练运行容器。

	获取方式	以参数名称为“data_path”的训练输入为例，说明获取方式的作用。
		• 当参数的“获取方式”为“超参”时，可以参考如下代码来读取数据。
		`import argparse`
		`parser = argparse.ArgumentParser()`
		`parser.add_argument('--data_path')`
		`args, unknown = parser.parse_known_args()`
		`data_path = args.data_path`
		• 当参数的“获取方式”为“环境变量”时，可以参考如下代码来读取数据。
		`import os`
		`data_path = os.getenv("data_path", "")`

输出	参数名称	算法代码需要通过“输出”的“参数名称”去读取训练的输出目录。 建议设置为“train_url”。训练输出参数要与所选算法的“输出”参数匹配，请参见创建算法时的表7-4。
	数据存储位置	单击“数据存储位置”，从OBS桶中选择训练输出数据的存储位置。训练过程中，系统将自动从训练容器的本地代码目录下同步文件到数据存储位置。 **说明** 数据存储位置仅支持OBS路径。为避免数据存储冲突，建议选择一个空目录用作“数据存储位置”。

参数名称	子参数	说明

	获取方式	以参数名称为“train_url”的训练输出为例，说明获取方式的作用。
		● 当参数的“获取方式”为“超参”时，可以参考如下代码来读取数据。
		`import argparse`
		`parser = argparse.ArgumentParser()`
		`parser.add_argument('--train_url')`
		`args, unknown = parser.parse_known_args()`
		`train_url = args.train_url`
		● 当参数的“获取方式”为“环境变量”时，可以参考如下代码来读取数据。
		`import os`
		`train_url = os.getenv("train_url", "")`

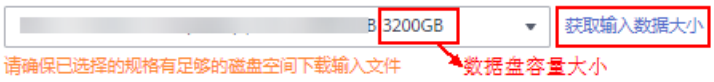
	预下载至本地目录	选择是否将输出目录下的文件预下载至本地目录。 ● 不下载：表示启动训练作业时不会将输出数据的存储位置中的文件下载到训练容器的本地代码目录中。 ● 下载：表示系统会在启动训练作业时自动将输出数据的存储位置中的所有文件下载到训练容器的本地代码目录中。下载时间会随着文件变大而变长，为了防止训练时间过长，请及时清理训练容器的本地代码目录中的无用文件。如果要使用**断点续训练和增量训练**，则必须选择“下载”。
超参	-	超参用于训练调优。此参数由选择的算法决定，如果在算法中已经定义了超参，则此处会显示算法中所有的超参。 超参支持修改和删除，状态取决于算法中的超参“约束”设置，详情请参见**定义超参**。
环境变量	-	根据业务需求增加环境变量。训练容器中预置的环境变量请参见**查看训练容器环境变量**。
故障自动重启	-	打开开关后，可以设置故障的重启次数，训练作业失败时会自动重新下发并运行训练作业。在训练作业详情页中，用户可以查看训练作业发生故障后的重启次数。 ● 此参数默认关闭。 ● 打开故障自动重启后，需要设置重启次数，重启次数取值范围1~3次，重启次数设置后无法修改。

配置资源池（公共资源池）

如果使用公共资源池创建训练作业，则参考**表7-11**配置公共资源池。

表 7-11 创建训练作业的公共资源池

参数名称	说明
资源池	必选，选择“公共资源池”。

参数名称	说明
资源类型	必选，选择训练需要的资源类型。当训练代码中已定义资源类型时，则根据算法的约束条件选择合适的资源类型。例如，训练代码中定义的资源类型为CPU，这里选择其他类型时会导致训练失败。如果部分资源类型不可见或不可选，表示不支持。
规格	必选，根据不同的资源类型，选择所需的资源规格。 当“输入”参数选择“数据存储位置”时，在选择资源池规格时可以单击右侧的“获取输入数据大小”，检查输入数据的大小是否超出数据盘的容量限制，避免训练过程中出现内存不足的情况。  **须知** 资源规格为“GPU:n*nvidia-t4”（n表示具体数字）的资源不支持多进程的训练任务。
计算节点个数	必填，根据需要选择计算节点的个数。默认值为“1”。 • 当“计算节点个数 = 1”时，创建的是单机训练作业，ModelArts只会在一个节点上启动一个训练容器，该训练容器独享所选规格的计算资源。 • 当“计算节点个数 > 1”时，创建的是分布式训练作业，更多分布式训练配置请参见**分布式训练功能介绍**。
永久保存日志	选择CPU或者GPU资源时，支持选择是否打开“永久保存日志”开关。 • 开关关闭（默认关闭）：表示不永久保存日志，则训练日志会在30天后会被清理。可以在作业详情页下载全部日志至本地。 • 开关打开：表示永久保存日志，此时必须配置“作业日志路径”，系统会将训练日志永久保存至指定的OBS路径。
作业日志路径	选择Ascend资源时或者打开“永久保存日志”开关时，必须配置“作业日志路径”，用于存放训练作业产生的日志文件。 建议选择一个空的OBS文件目录存放运行中产生的日志文件，同时需要OBS文件目录的读写权限。

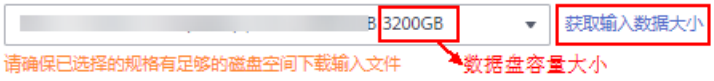
参数名称	说明
事件通知	选择是否打开“事件通知”开关。 ● 开关关闭（默认关闭）：表示不启用消息通知服务。 ● 开关打开：表示订阅消息通知服务，当训练作业发生特定事件（如作业状态变化或疑似卡死）时会发送通知。此时必须配置“主题名”和“事件”。 – “主题名”：事件通知的主题名称。单击“创建主题”，前往消息通知服务中创建主题。 – “事件”：选择要订阅的事件类型。例如“作业开始”、“作业结束”、“作业失败”、“作业终止”、“作业疑似卡死”等。 **说明** ● 需要为消息通知服务中创建的主题添加订阅，当订阅状态为“已确认”后，方可收到事件通知。 ● 只有资源类型为“GPU”的训练作业才支持通知“作业疑似卡死”的事件。
自动停止	● 开关关闭（默认关闭）：表示训练作业将一直运行直至训练完成。 ● 开关打开：表示启用自动停止功能，此时必须配置自动停止时间，支持设置为“1小时”、“2小时”、“4小时”、6小时或“自定义”，自定义时间取值范围为1~720小时。启用该参数并设置时间后，运行时长到期后将会自动终止训练，准备排队等状态不扣除运行时长。

配置资源池（专属资源池）

如果使用专属资源池创建训练作业，则参考表7-12配置公共资源池。

表 7-12 创建训练作业的专属资源池

参数名称	说明
资源池	必选，选择“专属资源池”并选择要使用的资源池。 选择专属资源池时，支持查看当前资源池的状态、节点规格、空闲/碎片节点数、可用节点/总节点数以及卡数信息。当资源池有可用卡数时，单击“空闲/碎片节点数”列的“查看”，可用查看碎片详情，确认资源池是否满足训练需求。

参数名称	说明
规格	必选，根据不同的资源类型，选择所需的资源规格。 当“输入”参数选择“数据存储位置”时，在选择资源池规格时可以单击右侧的“获取输入数据大小”，检查输入数据的大小是否超出数据盘的容量限制，避免训练过程中出现内存不足的情况。  **须知** 资源规格为“GPU:n*nvidia-t4”（n表示具体数字）的资源不支持多进程的训练任务。
自定义规格	选择是否打开“自定义规格”开关。训练作业支持基于专属资源池的规格自定义资源规格，进而提升资源池的利用率。 • 开关关闭（默认关闭）：表示直接使用专属资源池的规格。 • 开关打开：表示自定义专属资源池的规格，自定义规格只能小于或等于专属资源池的“节点规格”。CPU规格只支持自定义核数和内存，GPU和Ascend规格支持自定义核数、内存和卡数。 **说明** 如果启用“自定义规格”，则“规格”参数将配置无效。
计算节点个数	必填，根据需要选择计算节点的个数。默认值为“1”。 • 当“计算节点个数 = 1”时，创建的是单机训练作业，ModelArts只会在一个节点上启动一个训练容器，该训练容器独享所选规格的计算资源。 • 当“计算节点个数 > 1”时，创建的是分布式训练作业，更多分布式训练配置请参见**分布式训练功能介绍**。
作业优先级	使用专属资源池创建训练作业时，支持设置训练作业的优先级。取值为1~3，默认优先级为1，最高优先级为3。 • 默认用户权限可选择优先级1和2，配置了“**设置作业为高优先级权限**”的用户可选择优先级1~3。 • 如果训练作业长时间处于“等待中”的状态，则可以通过修改作业优先级来减少排队时长，请参见**修改训练作业优先级**。

参数名称	说明
SFS Turbo	当ModelArts和SFS Turbo间网络直通时，训练作业支持挂载多个SFS Turbo存放训练数据。单击“增加挂载配置”，填写如下参数。 ● “文件系统”：选择一个SFS Turbo。 ● “云上挂载路径”：输入SFS Turbo对应在训练容器内的云上挂载路径。 ● “存储位置”：选择SFS Turbo的存储位置。如果用户配置了文件夹控制权限，请选择存储位置；如果用户未配置文件夹控制权限，可以保持默认值“/”或者自定义位置。 ● “挂载方式”：显示挂载SFS Turbo的权限。根据SFS Turbo存储位置的权限显示“读写”或“只读”，如果用户未配置文件夹控制权限，则该参数不可见。 **说明** ● 相同的文件系统只能挂载一次 ，且只能对应一个挂载路径，挂载路径均不可重复。最多可以挂载8个盘。 ● 如果要使用训练作业挂载SFS Turbo功能，需要配置ModelArts和SFS Turbo间网络直通。 ● 云上挂载路径有如下限制：不能为 / 目录，不能为 /cache、 /home/ma-user/modelarts等系统已经默认挂载的路径。
永久保存日志	选择CPU或者GPU资源时，支持选择是否打开“永久保存日志”开关。 ● 开关关闭（默认关闭）：表示不永久保存日志，则训练日志会在30天后会被清理。可以在作业详情页下载全部日志至本地。 ● 开关打开：表示永久保存日志，此时必须配置“作业日志路径”，系统会将训练日志永久保存至指定的OBS路径。
作业日志路径	选择Ascend资源时或者打开“永久保存日志”开关时，必须配置“作业日志路径”，用于存放训练作业产生的日志文件。 建议选择一个空的OBS文件目录存放运行中产生的日志文件，同时需要OBS文件目录的读写权限。

参数名称	说明
事件通知	选择是否打开“事件通知”开关。 ● 开关关闭（默认关闭）：表示不启用消息通知服务。 ● 开关打开：表示订阅消息通知服务，当训练作业发生特定事件（如作业状态变化或疑似卡死）时会发送通知。此时必须配置“主题名”和“事件”。 – “主题名”：事件通知的主题名称。单击“创建主题”，前往消息通知服务中创建主题。 – “事件”：选择要订阅的事件类型。例如“作业开始”、“作业结束”、“作业失败”、“作业终止”、“作业疑似卡死”等。 **说明** ● 需要为消息通知服务中创建的主题添加订阅，当订阅状态为“已确认”后，方可收到事件通知。 ● 只有资源类型为“GPU”的训练作业才支持通知“作业疑似卡死”的事件。
自动停止	● 开关关闭（默认关闭）：表示训练作业将一直运行直至训练完成。 ● 开关打开：表示启用自动停止功能，此时必须配置自动停止时间，支持设置为“1小时”、“2小时”、“4小时”、6小时或“自定义”，自定义时间取值范围为1~720小时。启用该参数并设置时间后，运行时长到期后将会自动终止训练，准备排队等状态不扣除运行时长。

（可选）选择训练模式

当训练作业的算法框架选用的是MindSpore类引擎、资源池类型选用的是Ascend资源时，支持选择训练模式。ModelArts提供了3种训练模式供用户选择，支持根据实际场景获取不同的诊断信息，详细使用请参见**训练模式选择**。

- 普通模式：默认训练场景。
- 高性能模式：最小化调测信息，可以提升运行速度，适合于网络稳定并追求高性能的场景。
- 故障诊断模式：收集更多的信息用于定位，适合于执行出现问题需要收集故障信息进行定位的场景。此模式提供故障诊断，用户可以根据实际需求选择诊断类别。

后续操作

当创建训练作业的参数配置完成后，单击“提交”，在信息确认页面单击“确定”，提交创建训练作业任务。

训练作业一般需要运行一段时间，前往训练作业列表，可以查看训练作业的基本情况。

- 在训练作业列表中，刚创建的训练作业状态为“等待中”。

- 当训练作业的状态变为“已完成”时，表示训练作业运行结束，其生成的模型将存储至对应的“输出”目录中。
- 当训练作业的状态变为“运行失败”或“异常”时，可以单击训练作业的名称进入详情页面，通过查看日志等手段处理问题。

7.4.2 查看训练作业详情

1. 登录ModelArts管理控制台。
2. 在左侧导航栏中，选择“训练管理 > 训练作业”，进入“训练作业”列表。
3. 在“训练作业”列表中，单击作业名称，进入训练作业详情页。
4. 在训练作业详情页的左侧，可以查看此次训练作业的基本信息和算法配置的相关信息。
   - **训练作业基本信息**

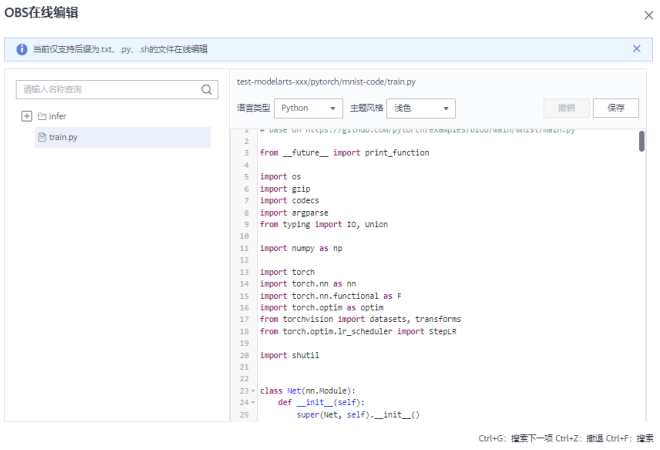
     表 7-13 训练作业基本信息

参数	说明
“作业ID”	训练作业唯一标识。
“作业状态”	训练作业状态。
“创建时间”	记录训练作业创建时间。
“运行时长”	记录训练作业运行时长。
“重启次数”	创建训练作业时如果出现故障，作业自动重启的次数。仅当创建训练作业时开启“故障自动重启”功能时可见。
“描述”	训练作业的描述。 您可以单击编辑图标，更新训练作业的描述。

   - **训练作业参数**

     表 7-14 训练作业参数

参数	说明
“算法名称”	本次训练作业使用的算法。单击算法名称，可以跳转至算法详情页面。
“预置镜像”	本次训练作业使用的预置镜像框架。

参数	说明
“代码目录”	训练作业代码目录所在的OBS路径。 您可以单击代码目录后的“编辑代码”，在“OBS在线编辑”对话框中实时编辑训练脚本代码。当训练作业状态为“等待中”、“创建中”和“运行中”时，不支持“OBS在线编辑”功能。  **说明** 当您使用AI Hub中订阅的算法创建训练作业时，不支持该参数。
“启动文件”	训练作业启动文件位置。 **说明** 当您使用AI Hub中订阅的算法创建训练作业时，不支持该参数。
“运行用户ID”	容器运行时的用户ID。
“本地代码目录”	训练代码在训练容器中的存放路径。
“工作目录”	训练启动文件在训练容器中的路径。
“计算节点个数”	本次训练作业设置的计算节点个数。
“专属资源池”	专属资源池信息，仅当训练作业使用专属资源池时可见。
“规格”	本次训练作业使用的训练规格。
“输入-输入路径”	本次训练中，输入数据的OBS路径。
“输入-参数名称”	算法代码中，输入路径指代的参数。
“输入-获取方式”	本次训练作业的输入采用的获取方式。

参数	说明
“输入-本地路径（训练参数值）”	训练启动后，ModelArts将OBS路径中的数据下载至后台容器，本地路径指ModelArts后台容器中存储输入数据的路径。
“输出-输出路径”	本次训练中，输出数据的OBS路径。
“输出-参数名称”	算法代码中，输出路径指代的参数。
“输出-获取方式”	本次训练作业的输出采用的获取方式。
“输出-本地路径（训练参数值）”	ModelArts后台容器中存储训练输出的路径。
“超参”	本次训练作业使用的超参。
“环境变量”	本次训练作业设置的环境变量。

7.4.3 查看训练作业事件

训练作业的（从用户可看见训练任务开始）整个生命周期中，每一个关键事件点在系统后台均有记录，用户可随时在对应训练作业的详情页面进行查看。

方便用户更清楚的了解训练作业运行过程，遇到任务异常时，更加准确的排查定位问题。当前支持的作业事件如下所示：

- 训练作业创建成功
- 训练作业创建失败报错：
- 准备阶段超时。可能原因是跨区域算法同步或者创建共享存储超时
- 训练作业已排队，正在等待资源分配
- 训练作业排队失败
- 训练作业开始运行
- 训练作业运行成功
- 训练作业运行失败
- 训练作业被抢占
- 系统检测到您的作业疑似卡死，请及时前往作业详情界面查看并处理
- 训练作业已重启
- 训练作业已被手动终止
- 训练作业已被终止（最大运行时长：1h）
- 训练作业已被终止（最大运行时长：3h）
- 训练作业已被手动删除
- 计费信息同步结束
- [worker-0] 训练环境预检中

- [worker-0] [耗时： 秒] 预检完成
- [worker-0] [耗时： 秒] 检查失败。发现异常：
- [worker-0] [耗时： 秒] 检查失败。发现错误：
- [worker-0] 训练代码下载中
- [worker-0] [耗时： 秒] 训练代码下载完成
- [worker-0] [耗时： 秒] 训练代码下载失败，失败原因：
- [worker-0] 训练输入下载中
- [worker-0] [耗时： 秒] 训练输入（ 参数名称： ）下载完成
- [worker-0] [耗时： 秒] 训练输入（ 参数名称： ）下载失败，失败原因：
- [worker-0] 正在安装Python依赖包，导入文件：
- [worker-0] [耗时： 秒] Python依赖包安装完成，导入文件：
- [worker-0] 训练任务开始运行
- [worker-0] 训练任务运行结束，退出码
- [worker-0] 训练输入上传中
- [worker-0] [耗时： 秒] 训练输出（ 参数名称： ）上传完成

训练运行到结束的过程中，关键事件支持手动/自动刷新。

查看操作

1. 在ModelArts管理控制台的左侧导航栏中选择“训练管理 > 训练作业”。
2. 在训练作业列表中，单击作业名称进入训练作业详情页面。
3. 在训练作业详情页面，单击“事件”页签查看事件信息。

7.4.4 查看训练作业日志

7.4.4.1 什么是训练作业日志

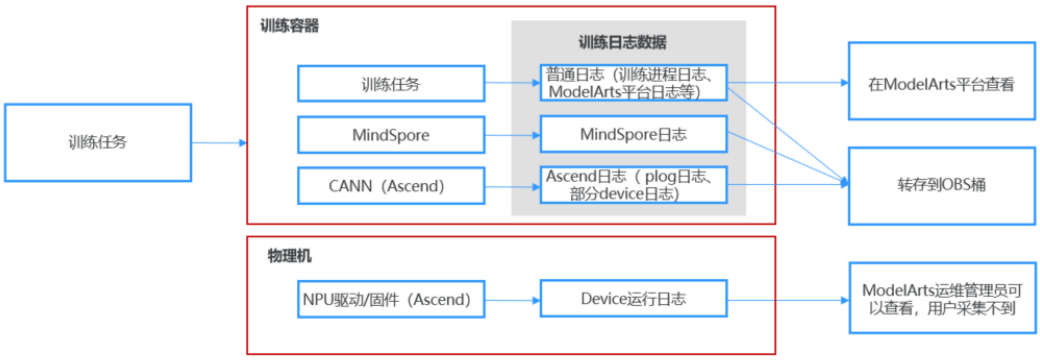
训练日志定义

训练日志用于记录训练作业运行过程和异常信息，为快速定位作业运行中出现的问题提供详细信息。用户代码中的标准输出、标准错误信息会在训练日志中呈现。在ModelArts中训练作业遇到问题时，可首先查看日志，多数场景下的问题可以通过日志报错信息直接定位。

训练日志包括普通训练日志和Ascend相关日志。

- **普通日志说明**：当使用Ascend之外的资源训练时仅产生普通训练日志，普通日志中包含训练进程日志、pip-requirement.txt安装日志和ModelArts平台日志。
- **Ascend场景日志说明**：使用Ascend资源训练时会产生device日志、plog日志、proc log单卡训练日志、MindSpore日志、普通日志。

图 7-7 ModelArts 训练日志



说明

只有MindSpore+Ascend训练场景下会产生单独的MindSpore日志。其他AI引擎的日志都包含在普通日志中，无法区分。

训练日志的时效性

从日志产生的时效性上可以分为以下3种情况：

- 实时日志：训练作业实时运行时产生，在ModelArts训练作业详情页面上可以查看。
- 历史日志：训练作业结束后，可以在ModelArts训练作业详情页面上查看历史日志，ModelArts系统自动保存30天。
- 永久日志：转存到OBS桶中的训练日志，在创建训练作业时，打开永久保存日志开关设置作业日志路径即可将日志转存至OBS路径。Ascend训练场景下，默认要求填写作业日志在OBS的存放路径，其他资源的训练场景下，永久保存日志开关需要用户手动开启。

图 7-8 开启永久保存日志开关

实时日志和历史日志都是标准日志输出，内容上没有区别。Ascend训练场景下，永久日志中会包含Ascend日志，这部分日志内容在ModelArts界面上看不到。

相关章节

- 在ModelArts训练作业详情页，训练日志窗口提供了日志预览、日志下载、日志中搜索关键字能力，具体请参见**如何查看训练作业日志**。
- ModelArts还提供了训练作业失败定位与分析功能，方便用户通过日志快速定位并解决训练作业问题，具体请参见**如何通过训练日志定位问题**。

7.4.4.2 普通日志说明

普通日志中包含训练进程日志、pip-requirement.txt安装日志和ModelArts平台日志。

普通日志类型

表 7-15 普通日志类型

日志类型	说明
训练进程日志	用户训练代码的标准输出。
pip-requirement.txt安装日志	如果用户有定义pip-requirement.txt文件，会产生pip包安装日志。
ModelArts平台日志	ModelArts平台产生的系统日志，主要用于运维人员定位平台问题。

普通日志的文件格式

普通日志的文件格式如下，其中task id为训练作业中的节点id。

```
统一日志格式：modelarts-job-[job id]-[task id].log
样例：log/modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-worker-0.log
```

- 单机训练作业只会生成一个日志文件，单机作业的task id默认为worker-0。
- 分布式场景下有多个节点日志文件并存，通过task id区分不同节点，例如：worker-0，worker-1等。

训练进程日志、pip-requirement.txt安装日志和ModelArts平台日志都包含在普通日志文件modelarts-job-[job id]-[task id].log中

ModelArts 平台日志

ModelArts平台日志可以通过关键字在训练的普通日志文件modelarts-job-[job id]-[task id].log中筛查，筛查关键字有：[ModelArts Service Log]或Platform=ModelArts-Service。

- 类型一：[ModelArts Service Log] xxx
  ```
  [ModelArts Service Log][init] download code_url: s3://dgg-test-user/snt9-test-cases/mindspore/lenet/
  ```
- 类型二：time=“xxx” level=“xxx” msg=“xxx” file=“xxx” Command=xxx Component=xxx Platform=xxx
  ```
  time="2021-07-26T19:24:11+08:00" level=info msg="start the periodic upload task, upload period = 5 seconds " file="upload.go:46" Command=obs/upload Component=ma-training-toolkit Platform=ModelArts-Service
  ```

7.4.4.3 Ascend 场景日志说明

Ascend 场景日志说明

使用Ascend资源运行训练作业时，会产生Ascend相关日志。Ascend训练场景下会生成device日志、plog日志、proc log单卡训练日志、MindSpore日志、普通日志。

其中，Ascend训练场景下的普通日志包括训练进程日志、pip-requirement.txt安装日志、ModelArts平台日志、ma-pre-start日志和davincirun日志。

Ascend日志结构举例说明如下：

```
obs://dgg-test-user/snt9-test-cases/log-out/                                  # 作业日志路径
├──modelarts-job-9ccf15f2-6610-42f9-ab99-059ba049a41e
   ├── ascend
      ├── process_log
            ├── rank_0
         ├── plog                                  # plog日志
                     ...
         ├── device-0                               # device日志
                     ...
   ├── mindspore                                                  # MindSpore日志
├──modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-worker-0.log              # 普通日志
├──modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-proc-rank-0-device-0.txt    # proc log单卡训练日志
```

表 7-16 Ascend 场景下日志说明

日志类型	日志说明	日志文件名
device日志	HOST侧用户进程，在DEVICE侧产生的AICPU、HCCP的日志，回传到HOST侧（训练容器）。 在训练进程结束后，该日志会上传至ModelArts训练容器的~/ascend/log/目录中。 如果出现如下情况，则device日志会获取不到。 ● 节点异常重启 ● 被主动停止的节点	“~/ascend/log/device-{device-id}/device-{pid}_{timestamp}.log” 其中，pid是HOST侧用户进程号。 样例： device-166_20220718191853764.log
plog日志	HOST侧用户进程，在HOST侧产生的日志（例如:ACL / GE）。 默认设置plog日志打印，“ascend/log”文件夹及相关文件默认不生成日志文件。	“~/ascend/log/plog/plog-{pid}_{timestamp}.log” 其中，pid是HOST侧用户进程号。 样例： plog-166_20220718191843620.log

日志类型	日志说明	日志文件名
proc log	proc log是单卡训练日志重定向文件，方便用户快速定位对应计算节点的日志。	“[modelarts-job-uuid]-proc-rank-[rank id]-device-[device logic id].txt” • device id为本次训练作业的NPU卡编号，取值单卡为0，8卡为0~7。 例如：Ascend规格为 8*Snt9时，device id取值为0~7；Ascend规格为 1*Snt9时，device id取值为0。 • rank id为本次训练作业的全局NPU卡编号，取值为0~计算节点数*卡数-1，单个计算节点下，rank id与device id取值相同。 样例： modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-proc-rank-0-device-0.txt
MindSpore日志	使用MindSpore+Ascend训练时会产生单独的MindSpore日志。	MindSpore的日志介绍请参见MindSpore官网。
普通训练日志	• ma-pre-start日志（Ascend场景特有）：如果用户有定义ma-pre-start脚本，会产生该脚本执行日志。 • davincirun日志（Ascend场景特有）：Ascend训练进程通过davincirun.py文件启动，该启动文件产生的日志。 • 训练进程日志：用户训练代码的标准输出。 • pip-requirement.txt安装日志：如果用户有定义pip-requirement.txt文件，会产生pip包安装日志。 • ModelArts平台日志：ModelArts平台产生的系统日志，主要用于运维人员定位平台问题。	合并输出在日志文件modelarts-job-[job id]-[task id].log中。 task id表示计算节点id，单节点时取值为worker-0，多节点时取值为worker-0、worker-1、...worker-{n-1}，n为计算节点个数。 样例： modelarts-job-95f661bd-1527-41b8-971c-eca55e513254-worker-0.log

Ascend 场景日志存放路径

Ascend训练场景下，当训练进程退出后，ModelArts会上传训练容器中的日志文件至“作业日志路径” 参数设置的OBS目录中。

Ascend 日志相关环境变量设置

您可以通过ma-pre-start脚本修改默认环境变量配置。

```
ASCEND_GLOBAL_LOG_LEVEL=3     # 设置日志级别  debug级别为0;info级别为1;warning级别为2;error级别为3。
ASCEND_SLOG_PRINT_TO_STDOUT=1  # 设置plog日志是否在屏幕上显示，1表示默认设置在屏幕上显示日志。
ASCEND_GLOBAL_EVENT_ENABLE=1   # 设置事件级别  不开启Event日志级别为0；开启Event日志级别为1。
```

ma-pre-start脚本在与训练启动文件同级的目录下放置，命名为ma-pre-start.sh or ma-pre-start.py脚本。

在训练启动文件被执行前，系统会在 /home/work/user-job-dir/ 目录下执行上述ma-pre-start脚本，使用该机制可以更新容器镜像内安装的Ascend RUN包，或者设置一些训练运行时额外需要的全局环境变量。

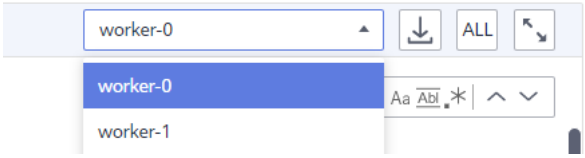
7.4.4.4 如何查看训练作业日志

在训练作业详情页，训练日志窗口提供日志预览、日志下载、日志中搜索关键字、系统日志过滤能力。

- 预览

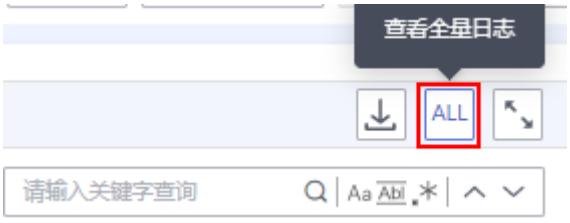
  系统日志窗口提供训练日志预览功能，如果训练作业有多个节点，则支持查看不同计算节点的日志，通过右侧下拉框可以选择目标节点预览。

  图 7-9 查看不同计算节日志

  当日志文件过大时，系统日志窗口仅加载最新的部分日志，并在日志窗口上方提供全量日志访问链接。打开该链接可在新页面查看全部日志。

  图 7-10 查看全量日志

  说明

  - 如果全部日志超过500M，可能会引起浏览页面卡顿，建议您直接下载日志查看。
  - 预览链接在生成后的一小时内，支持任何人打开并查看。您可以分享链接至他人。
  - **请注意日志中不能包含隐私内容，否则会造成信息泄露。**

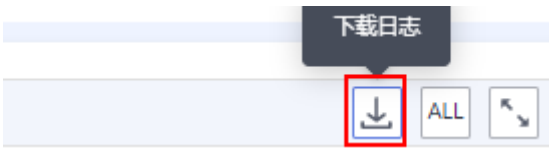
- 下载

  训练日志仅保留30天，超过30天会被清理。如果用户需要永久保存日志，请单击系统日志窗口右上角下载按钮下载日志至本地保存，支持批量下载多节点日志。

用户也可以在创建训练作业时打开永久保存日志按钮，保存训练日志至指定OBS路径。

针对使用Ascend规格创建的训练作业，部分系统日志暂不支持直接在训练日志窗口下载，请在创建训练作业时指定OBS路径用于保存训练日志。

图 7-11 下载日志



- 搜索关键字

  用户可以在系统日志右上角的搜索框搜索关键字。

  系统支持高亮关键字并实现搜索结果间的跳转。搜索功能仅支持搜索当前页面加载的日志，如果日志加载不全（请关注页面提示）则需要下载或者通过打开全量日志访问链接进行搜索。全量日志访问链接打开的新页面可以通过Ctrl+F进行搜索。

- 系统日志过滤

图 7-12 系统日志复选框

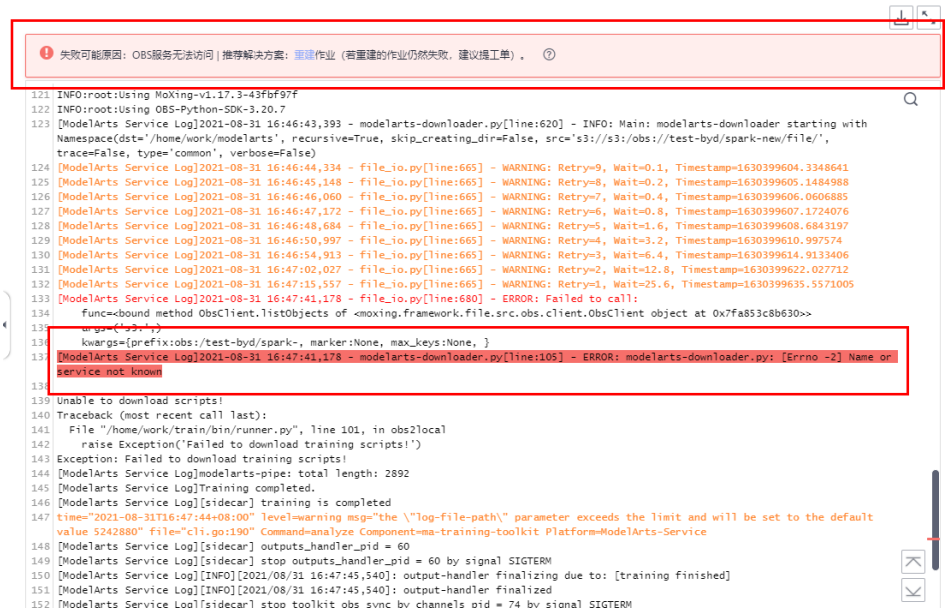
如果勾选了系统日志复选框，则日志中呈现系统日志和用户日志。如果去勾选，则只显示用户日志。

7.4.4.5 如何通过训练日志定位问题

在ModelArts中训练作业遇到问题时，可首先查看日志，多数场景下的问题可以通过日志报错信息直接定位。

ModelArts提供了训练作业失败定位与分析功能，如果训练作业运行失败，ModelArts会自动识别导致作业失败的原因，在训练日志界面上给出提示。提示包括三部分：失败的可能原因、推荐的解决方案以及对应的日志（底色标红部分）。

图 7-13 训练故障识别



ModelArts会对部分常见训练错误给出分析建议，目前还不能识别所有错误，提供的失败可能原因仅供参考。针对分布式作业，只会显示当前节点的一个分析结果，作业的失败需要综合各个节点的失败原因做一个综合判断。

常见训练问题定位思路如下：

1. 根据日志界面提示中提供的分析建议解决。
   - 参考案例解决：会提供当前故障对应的指导文档链接，请参照文档中的解决方案修复问题。
   - 重建作业：建议重建作业进行重试，大概率能修复问题。
2. 上一步不能解决问题时，可以尝试分析日志中提示的错误信息，定位并解决问题。
3. 最后，如果以上均不能解决问题，可以提工单进行人工咨询。

7.4.5 Cloud Shell

7.4.5.1 使用 Cloud Shell 登录训练容器

使用场景

允许用户使用ModelArts控制台提供的Cloud Shell登录运行中的训练容器。

约束限制

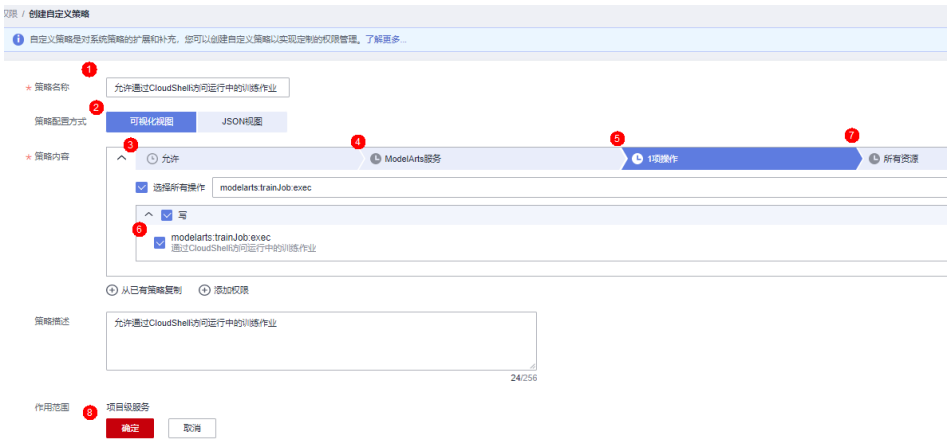
仅专属资源池均支持使用Cloud Shell，且训练作业必须处于“运行中”状态。

前提条件：给子账号配置允许使用 Cloud Shell 的权限

1. 使用主用户账号登录管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。

2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”按如下要求设置完成后单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：允许通过Cloud Shell访问运行中的训练作业。
   - “策略配置方式”：选择可视化视图。
   - “策略内容”：允许，云服务中搜索ModelArts服务并选中，操作列中搜索关键词modelarts:trainJob:exec并选中，所有资源选择默认值。

   图 7-14 创建自定义策略

3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限通过Cloud Shell登录运行中的训练作业容器。

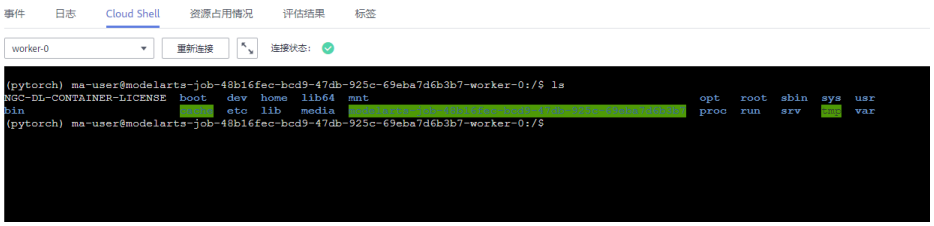
   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

如何使用 Cloud Shell

1. 参考**前提条件：给子账号配置允许使用Cloud Shell的权限**，完成配置。
2. 在ModelArts管理控制台的左侧导航栏中选择“训练管理 > 训练作业”。
3. 在训练作业列表中，单击作业名称进入训练作业详情页面。
4. 在训练作业详情页面，单击“Cloud Shell”页签，登录训练容器。

   连接成功后，Cloud Shell界面提示如下。

   图 7-15 Cloud Shell 界面

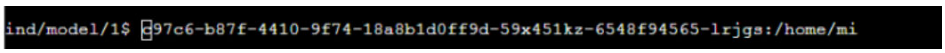
   

当作业处于非运行状态或权限不足时会导致无法使用Cloud Shell，请根据提示定位原因即可。

说明

部分用户登录Cloud Shell界面时，可能会出现路径显示异常情况，此时在Cloud Shell中单击回车键即可恢复正常。

图 7-16 路径异常



7.4.6 查看训练作业资源利用率

如何查看训练作业资源使用详情

1. 在ModelArts管理控制台的左侧导航栏中选择“训练管理 > 训练作业”。
2. 在训练作业列表中，单击作业名称进入训练作业详情页面。
3. 在训练作业详情页面，单击“资源占用情况”页签查看计算节点的资源使用情况，最多可显示最近三天的数据。在“资源占用情况”窗口打开时，会定期向后台获取最新的资源使用率数据并刷新。

   操作一：如果训练作业使用多个计算节点，可以通过实例名称的下拉框切换节点。

   操作二：单击图例“cpuUsage”、“gpuMemUsage”、“gpuUtil”、“memUsage”“npuMemUsage”、“npuUtil”、可以添加或取消对应参数的使用情况图。

   操作三：鼠标悬浮在图片上的时间节点，可查看对应时间节点的占用率情况。

表 7-17 参数说明

参数	说明
cpuUsage	cpu使用率。
gpuMemUsage	gpu内存使用率。
gpuUtil	gpu使用情况。
memUsage	内存使用率。
npuMemUsage	npu内存使用率。
npuUtil	npu使用情况。

如何判断训练作业资源利用率高低

在训练管理的训练作业列表页可以查看作业资源利用率情况。当作业worker-0实例的GPU/NPU的平均利用率低于50%时，在训练作业列表中会进行告警提示。

图 7-17 作业列表显示作业资源利用率情况

此处的作业资源利用率只涉及GPU和NPU资源。作业worker-0实例的GPU/NPU平均利用率计算方法：将作业worker-0实例的各个GPU/NPU加速卡每个时间点的利用率汇总取平均值。

如何提高训练作业资源利用率

- 适当增大batch_size：较大的batch_size可以让GPU/NPU计算单元获得更高的利用率，但是也要根据实际情况来选择batch_size，防止batch_YLLsize过大导致内存溢出。
- 提升数据读取的效率：如果读取一个batch数据的时间要长于GPU/NPU计算一个batch的时间，就有可能出现GPU/NPU利用率上下浮动的情况。建议优化数据读取和数据增强的性能，例如将数据读取并行化，或者使用NVIDIA Data Loading Library（DALI）等工具提高数据增强的速度。
- 模型保存不要太频繁：模型保存操作一般会阻塞训练，如果模型较大，并且较频繁地进行保存，就会影响GPU/NPU利用率。同理，其他非GPU/NPU操作尽量不要阻塞训练主进程太多的时间，如日志打印，保存训练指标信息等。

7.4.7 评估训练结果

训练作业运行结束后，ModelArts可为您的模型进行评估，并且给出调优诊断和建议。

- 针对使用预置算法创建训练作业，无需任何配置，即可查看此评估结果（由于每个模型情况不同，系统将自动根据您的模型指标情况，给出一些调优建议，请仔细阅读界面中的建议和指导，对您的模型进行进一步的调优）。
- 针对用户自己编写训练脚本或自定义镜像方式创建的训练作业，则需要在您的训练代码中添加评估代码，才可以在训练作业结束后查看相应的评估诊断建议。

  说明

  - 只支持验证集的数据格式为图片
  - 目前，仅如下常用框架的训练脚本支持添加评估代码。
    - TF-1.13.1-python3.6
    - TF-2.1.0-python3.6
    - PyTorch-1.4.0-python3.6

下文将介绍如何在训练中使用评估代码。对训练代码做一定的适配和修正，分为三个方面：**添加输出目录**、**复制数据集到本地**、**映射数据集路径到OBS**。

添加输出目录

添加输出目录的代码比较简单，即在代码中添加一个输出评估结果文件的目录，被称为train_url，也就是页面上的训练输出位置。并把train_url添加到使用的函数analysis中，使用save_path来获取train_url。示例代码如下所示：

```
FLAGS = tf.app.flags.FLAGS
tf.app.flags.DEFINE_string('model_url', '', 'path to saved model')
tf.app.flags.DEFINE_string('data_url', '', 'path to output files')
tf.app.flags.DEFINE_string('train_url', '', 'path to output files')
tf.app.flags.DEFINE_string('adv_param_json',
                           '{"attack_method":"FGSM","eps":40}',
                           'params for adversarial attacks')
FLAGS(sys.argv, known_only=True)

...

# analyse
res = analyse(
    task_type=task_type,
    pred_list=pred_list,
    label_list=label_list,
    name_list=file_name_list,
    label_map_dict=label_dict,
    save_path=FLAGS.train_url)
```

复制数据集到本地

复制数据集到本地主要是为了防止长时间访问OBS容易导致OBS连接中断使得作业卡住，所以一般先将数据复制到本地再进行操作。

数据集复制有两种方式，推荐使用OBS路径复制。

- OBS路径（推荐）

  直接使用moxing的copy_parallel接口，复制对应的OBS路径。

- ModelArts数据管理中的数据集（即manifest文件格式）

  使用moxing的copy_manifest接口将文件复制到本地并获取新的manifest文件路径，然后使用SDK解析新的manifest文件。

说明

ModelArts数据管理模块在重构升级中，对未使用过数据管理的用户不可见。建议新用户将训练数据存放至OBS桶中使用。

```
if data_path.startswith('obs://'):
    if '.manifest' in data_path:
        new_manifest_path, _ = mox.file.copy_manifest(data_path, '/cache/data/')
        data_path = new_manifest_path
    else:
        mox.file.copy_parallel(data_path, '/cache/data/')
        data_path = '/cache/data/'
    print('-------------- download dataset success ------------')
```

映射数据集路径到 OBS

由于最终JSON体中需要填写的是图片文件的真实路径，也就是OBS对应的路径，所以在复制到本地做完分析和评估操作后，需要将原来的本地数据集路径映射到OBS路径，然后将新的list送入analysis接口。

如果使用的是OBS路径作为输入的data_url，则只需要替换本地路径的字符串即可。

```
if FLAGS.data_url.startswith('obs://'):
    for idx, item in enumerate(file_name_list):
        file_name_list[idx] = item.replace(data_path, FLAGS.data_url)
```

如果使用manifest文件，需要再解析一遍原版的manifest文件获取list，然后再送入analysis接口。

```
if or FLAGS.data_url.startswith('obs://'):
    if 'manifest' in FLAGS.data_url:
```

```
        file_name_list = []
        manifest, _ = get_sample_list(
            manifest_path=FLAGS.data_url, task_type='image_classification')
        for item in manifest:
            if len(item[1]) != 0:
                file_name_list.append(item[0])
```

完整的适配了训练作业创建的图像分类样例代码如下：

```
import json
import logging
import os
import sys
import tempfile

import h5py
import numpy as np
from PIL import Image

import moxing as mox
import tensorflow as tf
from deep_moxing.framework.manifest_api.manifest_api import get_sample_list
from deep_moxing.model_analysis.api import analyse, tmp_save
from deep_moxing.model_analysis.common.constant import TMP_FILE_NAME

logging.basicConfig(level=logging.DEBUG)

FLAGS = tf.app.flags.FLAGS
tf.app.flags.DEFINE_string('model_url', '', 'path to saved model')
tf.app.flags.DEFINE_string('data_url', '', 'path to output files')
tf.app.flags.DEFINE_string('train_url', '', 'path to output files')
tf.app.flags.DEFINE_string('adv_param_json',
                           '{"attack_method":"FGSM","eps":40}',
                           'params for adversarial attacks')
FLAGS(sys.argv, known_only=True)


def _preprocess(data_path):
    img = Image.open(data_path)
    img = img.convert('RGB')
    img = np.asarray(img, dtype=np.float32)
    img = img[np.newaxis, :, :, :]
    return img


def softmax(x):
    x = np.array(x)
    orig_shape = x.shape
    if len(x.shape) > 1:
        # Matrix
        x = np.apply_along_axis(lambda x: np.exp(x - np.max(x)), 1, x)
        denominator = np.apply_along_axis(lambda x: 1.0 / np.sum(x), 1, x)
        if len(denominator.shape) == 1:
            denominator = denominator.reshape((denominator.shape[0], 1))
        x = x * denominator
    else:
        # Vector
        x_max = np.max(x)
        x = x - x_max
        numerator = np.exp(x)
        denominator = 1.0 / np.sum(numerator)
        x = numerator.dot(denominator)
    assert x.shape == orig_shape
    return x


def get_dataset(data_path, label_map_dict):
    label_list = []
    img_name_list = []
    if 'manifest' in data_path:
```

```
        manifest, _ = get_sample_list(
            manifest_path=data_path, task_type='image_classification')
        for item in manifest:
            if len(item[1]) != 0:
                label_list.append(label_map_dict.get(item[1][0]))
                img_name_list.append(item[0])
            else:
                continue
    else:
        label_name_list = os.listdir(data_path)
        label_dict = {}
        for idx, item in enumerate(label_name_list):
            label_dict[str(idx)] = item
            sub_img_list = os.listdir(os.path.join(data_path, item))
            img_name_list += [
                os.path.join(data_path, item, img_name) for img_name in sub_img_list
            ]
            label_list += [label_map_dict.get(item)] * len(sub_img_list)
    return img_name_list, label_list


def deal_ckpt_and_data_with_obs():
    pb_dir = FLAGS.model_url
    data_path = FLAGS.data_url

    if pb_dir.startswith('obs://'):
        mox.file.copy_parallel(pb_dir, '/cache/ckpt/')
        pb_dir = '/cache/ckpt'
        print('------------- download success ------------')
    if data_path.startswith('obs://'):
        if '.manifest' in data_path:
            new_manifest_path, _ = mox.file.copy_manifest(data_path, '/cache/data/')
            data_path = new_manifest_path
        else:
            mox.file.copy_parallel(data_path, '/cache/data/')
            data_path = '/cache/data/'
        print('------------- download dataset success ------------')
    assert os.path.isdir(pb_dir), 'Error, pb_dir must be a directory'
    return pb_dir, data_path


def evalution():
    pb_dir, data_path = deal_ckpt_and_data_with_obs()
    index_file = os.path.join(pb_dir, 'index')
    try:
        label_file = h5py.File(index_file, 'r')
        label_array = label_file['labels_list'][:].tolist()
        label_array = [item.decode('utf-8') for item in label_array]
    except Exception as e:
        logging.warning(e)
        logging.warning('index file is not a h5 file, try json.')
        with open(index_file, 'r') as load_f:
            label_file = json.load(load_f)
        label_array = label_file['labels_list'][:]
    label_map_dict = {}
    label_dict = {}
    for idx, item in enumerate(label_array):
        label_map_dict[item] = idx
        label_dict[idx] = item
    print(label_map_dict)
    print(label_dict)

    data_file_list, label_list = get_dataset(data_path, label_map_dict)

    assert len(label_list) > 0, 'missing valid data'
    assert None not in label_list, 'dataset and model not match'

    pred_list = []
    file_name_list = []
```

```
    img_list = []

    for img_path in data_file_list:
        img = _preprocess(img_path)
        img_list.append(img)
        file_name_list.append(img_path)

    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    config.gpu_options.visible_device_list = '0'
    with tf.Session(graph=tf.Graph(), config=config) as sess:
        meta_graph_def = tf.saved_model.loader.load(
            sess, [tf.saved_model.tag_constants.SERVING], pb_dir)
        signature = meta_graph_def.signature_def
        signature_key = 'predict_object'
        input_key = 'images'
        output_key = 'logits'
        x_tensor_name = signature[signature_key].inputs[input_key].name
        y_tensor_name = signature[signature_key].outputs[output_key].name
        x = sess.graph.get_tensor_by_name(x_tensor_name)
        y = sess.graph.get_tensor_by_name(y_tensor_name)
        for img in img_list:
            pred_output = sess.run([y], {x: img})
            pred_output = softmax(pred_output[0])
            pred_list.append(pred_output[0].tolist())

    label_dict = json.dumps(label_dict)
    task_type = 'image_classification'

    if FLAGS.data_url.startswith('obs://'):
        if 'manifest' in FLAGS.data_url:
            file_name_list = []
            manifest, _ = get_sample_list(
                manifest_path=FLAGS.data_url, task_type='image_classification')
            for item in manifest:
                if len(item[1]) != 0:
                    file_name_list.append(item[0])
        for idx, item in enumerate(file_name_list):
            file_name_list[idx] = item.replace(data_path, FLAGS.data_url)
    # analyse
    res = analyse(
        task_type=task_type,
        pred_list=pred_list,
        label_list=label_list,
        name_list=file_name_list,
        label_map_dict=label_dict,
        save_path=FLAGS.train_url)

if __name__ == "__main__":
    evalution()
```

7.4.8 查看故障恢复详情

当训练作业发生故障恢复时（例如进程级恢复、POD级重调度、JOB级重调度等），作业详情页面中会出现“故障恢复详情”页签，里面记录了训练作业的启停情况。

1. 在ModelArts管理控制台的左侧导航栏中选择“训练管理 > 训练作业”。
2. 在训练作业列表中，单击作业名称进入训练作业详情页面。
3. 在训练作业详情页面，单击“故障恢复详情”页签查看故障恢复信息。

7.4.9 查看训练容器环境变量

什么是环境变量

本章节展示了训练容器环境中预置的环境变量，方便用户查看，主要包括以下类型。

- 路径相关环境变量
- 分布式训练任务环境变量
- NCCL（Nvidia Collective multi-GPU Communication Library）环境变量
- OBS环境变量
- PIP源环境变量
- API网关地址环境变量
- 作业元信息环境变量

如何修改环境变量

用户可以在创建训练作业页面增加新的环境变量，也可以设置新的取值覆盖当前训练容器中预置的环境变量值。

训练容器中预置的环境变量

训练容器中预置的环境变量如下面表格所示，包括表7-18、表7-19、表7-20、表7-21、表7-22、表7-23、表7-24。

此处的环境变量取值仅为示例，涉及不同规格、引擎、Region可能取值不一样，此处仅供参考。

表 7-18 路径相关环境变量

变量名	说明	示例
PATH	可执行文件路径，已包含常用的可执行文件路径。	“PATH=/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin”
LD_LIBRARY_PATH	动态链接库路径，已包含常用的动态库路径。	“LD_LIBRARY_PATH=/usr/local/seccomponent/lib:/usr/local/cuda/lib64:/usr/local/cuda/compat:/root/miniconda3/lib:/usr/local/nvidia/lib:/usr/local/nvidia/lib64”
LIBRARY_PATH	静态库路径，已包含常用的静态库路径。	“LIBRARY_PATH=/usr/local/cuda/lib64/stubs”
MA_HOME	训练作业的主目录。	“MA_HOME=/home/ma-user”
MA_JOB_DIR	训练算法文件夹所在的父目录。	“MA_JOB_DIR=/home/ma-user/modelarts/user-job-dir”

变量名	说明	示例
MA_MOUNT_PATH	ModelArts挂载至训练容器内的路径，用于临时存放训练算法、算法输入、算法输出、日志等文件。	“MA_MOUNT_PATH=/home/ma-user/modelarts”
MA_LOG_DIR	训练日志目录。	“MA_LOG_DIR=/home/ma-user/modelarts/log”
MA_SCRIPT_INTERPRETER	训练脚本解释器。	“MA_SCRIPT_INTERPRETER=”
WORKSPACE	训练算法目录。	“WORKSPACE=/home/ma-user/modelarts/user-job-dir/code”

表 7-19 分布式训练任务环境变量

变量名	说明	示例
MA_CURRENT_IP	作业容器IP。	“MA_CURRENT_IP=192.168.23.38”
MA_NUM_GPUS	作业容器的加速卡数量。	“MA_NUM_GPUS=8”
MA_TASK_NAME	作业容器的角色名，例如： ● MindSpore、PyTorch为worker ● 强化学习引擎为learner，worker ● TensorFlow为ps，worker	“MA_TASK_NAME=worker”
MA_NUM_HOSTS	训练作业使用的节点数量。	“MA_NUM_HOSTS=4”
VC_TASK_INDEX	多节点训练时，当前作业容器的序列号，第一个容器的值为0。	“VC_TASK_INDEX=0”
VC_WORKER_NUM	训练作业使用的节点数量。	“VC_WORKER_NUM=4”

变量名	说明	示例
VC_WORKER_HOSTS	多节点训练时，每个节点的域名地址，按顺序以英文逗号分隔，可以通过域名解析获取IP地址。	“VC_WORKER_HOSTS=modelarts-job-a0978141-1712-4f9b-8a83-00000000000000-worker-0.modelarts-job-a0978141-1712-4f9b-8a83-00000000000000,modelarts-job-a0978141-1712-4f9b-8a83-00000000000000-worker-1.ob-a0978141-1712-4f9b-8a83-00000000000000,modelarts-job-a0978141-1712-4f9b-8a83-00000000000000-worker-2.modelarts-job-a0978141-1712-4f9b-8a83-00000000000000,ob-a0978141-1712-4f9b-8a83-00000000000000-worker-3.modelarts-job-a0978141-1712-4f9b-8a83-00000000000000”

表 7-20 NCCL 环境变量

变量名	说明	示例
NCCL_VERSION	NCCL版本。	“NCCL_VERSION=2.7.8”
NCCL_DEBUG	NCCL日志等级。	“NCCL_DEBUG=INFO”
NCCL_IB_HCA	指定NCCL使用的IB网卡。	“NCCL_IB_HCA=^mlx5_bond_0”
NCCL_SOCKET_IFNAME	指定NCCL使用的SOCKET网卡。	“NCCL_SOCKET_IFNAME=bond0,eth0”

表 7-21 OBS 环境变量

变量名	说明	示例
S3_ENDPOINT	OBS地址。	“S3_ENDPOINT=https://obs.region.xxx.com”
S3_VERIFY_SSL	访问OBS是否使用SSL。	“S3_VERIFY_SSL=0”
S3_USE_HTTPS	访问OBS是否使用HTTPS。	“S3_USE_HTTPS=1”

表 7-22 PIP 源和 API 网关地址环境变量

变量名	说明	示例
MA_PIP_HOST	PIP源域名。	“MA_PIP_HOST=repo.xxx.com”
MA_PIP_URL	PIP源地址。	“MA_PIP_URL=http://repo.xxx.com/repository/pypi/simple/”
MA_APIGW_ENDPOINT	ModelArts API网关地址。	“MA_APIGW_ENDPOINT=https://modelarts.region.xxx.xxx.com”

表 7-23 作业元信息环境变量

变量名	说明	示例
MA_CURRENT_INSTANCE_NAME	多节点训练时，当前节点的名称。	“MA_CURRENT_INSTANCE_NAME=modelarts-job-a0978141-1712-4f9b-8a83-000000000000-worker-1”

表 7-24 预检相关环境变量

变量名	说明	示例
MA_SKIP_IMAGE_DETECT	ModelArts预检是否开启。默认为1，1表示开启预检，0表示关闭预检。 推荐开启预检，预检可提前发现节点故障、驱动故障。	“1”

7.4.10 停止、重建或查找作业

另存为算法

当您需要修改训练作业的算法时，可以在训练作业详情页面右上角，单击“另存为算法”。

在“创建算法”页面中，会自动填充上一次训练作业的算法参数配置，您可以根据业务需求在原来算法配置基础上进行修改。

说明

在AI Hub中订阅的算法不支持另存为算法。

停止训练作业

在训练作业列表中，针对“创建中”、“等待中”、“运行中”的训练作业，您可以单击“操作”列的“终止”，停止正在运行中的训练作业。

运行结束的训练作业，如“已完成”、“运行失败”、“已终止”、“异常”的作业，不涉及“终止”操作。

重建训练作业

当对创建的训练作业不满意时，您可以单击操作列的重建，重新创建训练作业。在重创训练作业页面，会自动填入上一次训练作业设置的参数，您仅需在原来的基础上进行修改即可重新创建训练作业。

查找训练作业

当用户使用IAM账号登录时，训练作业列表会显示IAM账号下所有训练作业。ModelArts提供查找训练作业功能帮助用户快速查找训练作业。

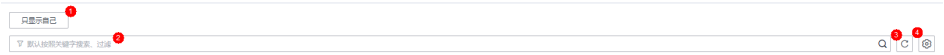
操作一：单击“只显示自己”按钮，训练作业列表仅显示当前子账号下创建的训练作业。

操作二：按照名称、ID、作业类型、状态、创建时间、算法、资源池等条件筛选的高级搜索。

操作三：单击作业列表右上角“刷新”图标，刷新作业列表。

操作四：自定义列功能设置。

图 7-18 查找训练作业



7.4.11 清除训练作业资源

如果不再需要使用此训练任务，建议清除相关资源。

- 在“训练作业”页面，“删除”运行结束的训练作业。您可以单击“操作”列的“删除”，在弹出的提示框中单击“确认”，删除对应的训练作业。
- 进入OBS，删除本示例使用的OBS桶及文件。

完成资源清除后，您可以在总览页面的使用情况确认资源删除情况。

图 7-19 查看使用情况

7.5 训练实验

7.5.1 什么是实验

实验为ModelArts提供的一种作业管理能力， 您可以将训练作业分类有序地放入实验中进行管理。

请参考以下指导在实验中管理训练作业：

- 将训练作业纳入实验，请参考**如何将训练作业纳入实验**。
- 查看实验信息，请参考**查看实验**。
- 删除实验，请参考**删除实验**。

7.5.2 如何将训练作业纳入实验

训练作业纳入实验的方式是在“创建训练作业”页面指定作业所属实验。有以下几个选项：

- 纳入新实验：实验不支持单独创建，只支持随着训练作业创建。此处填写新建的实验名称，在提交作业成功后，会同时创建该实验，并将该作业纳入新建的实验中。此处会校验实验名称，如果重名则无法提交作业。
- 纳入已有实验：您可以在下拉框中选择已有实验，将作业纳入已存在的实验。
- 不纳入实验：如果您不想通过实验管理您的作业，可以选择此选项。不纳入实验的作业无法在训练作业首页的“实验”中查看到。

新建纳入实验的作业

进入ModelArts控制台，选择“训练管理 > 训练作业”，单击右上角“创建训练作业”，进入“训练作业”页面。

填写实验设置，“设置实验”默认选中“纳入新实验”，同时填写“新建实验名称”，即可在新建训练作业时新建实验。

图 7-20 新建训练作业

已创建作业纳入实验

进入ModelArts控制台，选择“训练管理 > 训练作业”，选择“操作>重建”，进入“训练作业”页面。也可在作业列表中，单击作业“名称/ID”，进入作业详情页，单击右上角“重建”，进入“训练作业”页面。

- 未纳入实验的作业：“设置实验”默认选中“纳入新实验”，填写“新建实验名称”，即可在重建训练作业时新建实验。
- 已纳入实验的作业：“设置实验”默认选中“纳入已有实验”，同时选中源作业所在的实验。

图 7-21 重建训练作业

7.5.3 查看实验

查看实验列表

1. 登录ModelArts管理控制台，在左侧导航栏中选择“训练管理 >训练作业”，进入“训练作业”列表。
2. 单击“实验”切换到**实验**页签，进入实验列表。实验列表展示了实验的一些基本信息。

表 7-25 实验基本信息

参数	说明
实验名称	实验名，该名称支持在实验详情页面修改。
训练作业数	该实验下所包含的训练作业数量。
创建时间	实验创建时间。
修改时间	实验的修改时间，指实验产生以下变化的时间： ● 修改实验名称 ● 修改实验描述 ● 实验下新增/删除了训练作业
描述	实验描述，可修改。
操作	当前仅支持删除实验。

- 支持按照实验名称、训练作业数、创建时间、修改时间、描述等条件筛选的高级搜索。
- 支持作业列表刷新。单击作业列表右上角“刷新”图标，刷新作业列表。
- 支持自定义列表选项。单击实验列表右上角“设置”图标，自定义选择需要在实验列表中显示的选项。
- 支持对实验排序。单击表头中的箭头可根据就特定列的信息进行排序。

查看实验详情

在实验列表单击某个实验名称进入实验详情页面。实验详情上方会显示实验的基本信息，下方会显示该实验下的作业列表。

图 7-22 查看实验详情

- 单击支持编辑实验名称/描述。
- 打开“只看自己”按钮，可查看自己创建的纳入实验的作业。

  说明

  账号下有多个IAM用户时，默认只显示当前登录的IAM用户的作业列表。
- 按照名称、ID、算法、状态、创建时间、作业类型、资源池等条件筛选的高级搜索。
- 支持作业列表刷新。单击作业列表右上角“刷新”图标，刷新作业列表。
- 支持自定义列表选项。单击作业列表右上角“设置”图标，自定义选择需要在作业列表中显示的选项。

7.5.4 删除实验

实验支持删除操作，您可以在实验列表页单击“删除”，或者在实验详情页右上角单击“删除实验”，进入“删除实验”页面。“删除实验”页面会展示实验下所有的作业。如确需执行该操作，请输入“DELETE”，再单击“确认”进行删除。

注意

删除实验为高危操作，会同时删除实验下的所有作业，且不可恢复。

7.6 训练进阶

7.6.1 训练模式选择

如果训练作业选用的是MindSpore类引擎和Ascend资源，则ModelArts提供3种训练模式选择（普通模式、高性能模式和故障诊断模式），支持用户根据实际场景获取不同的诊断信息。

模式说明

训练作业默认设置为普通模式，普通模式的调测信息可参考**查看训练作业日志**。

- 高性能模式的使用场景：最小化调测信息，可以提升运行速度，适合于网络稳定并追求高性能的场景。
- 故障诊断模式的使用场景：收集更多的信息用于定位，适合于执行出现问题需要收集故障信息进行定位的场景。此模式提供故障诊断，用户可以根据实际需求选择诊断类别。

各模式获取的调测信息如**表7-26**所示。

表 7-26 MindSpore 引擎各模式的调测信息

调测信息	普通模式	高性能模式	故障诊断模式	说明
MindSpore框架日志级别	Info级别	error级别	Info级别	MindSpore框架运行时日志。
RDR(Running Data Recorder)	关闭	关闭	开启	出现运行异常会自动地导出MindSpore中预先记录的数据以辅助定位运行异常的原因。不同的运行异常将会导出不同的数据。 RDR详细的介绍请参考**MindSpore官网说明**。
analyze_fail.dat	默认提供，上传至训练作业日志路径中			图编译失败自动导出故障信息，用于infer过程分析。
dump数据	默认提供，上传至训练作业日志路径中			后端执行期异常触发dump数据。

在故障诊断模式下，开启故障诊断功能后，支持用户查看以下故障诊断数据。以下数据存储至训练日志路径的OBS目录下。

故障诊断模式的训练输出日志文件说明：

```
{obs-log-path}/
    modelarts-job-{job-id}-worker-{index}.log # 在屏幕上显示日志（汇总）
    modelarts-job-{job-id}-proc-rank-{rank-id}-device-{device-id}.txt # 每个device的日志显示在屏幕上
    modelarts-job-{job-id}/
        ascend/
            npu_collect/rank_{id}/   # TFAdapter DUMP GRAPH 与 GE DUMP GRAPH 的输出路径，仅在使用TensorFlow框架时生成
            process_log/rank_{id}/   # Plog 日志路径
            msnpureport/{task-index}/  #msnpureport工具执行日志，用户无需关注
        mindspore/
            log/  # MindSpore 框架日志与 MindSpore 故障诊断数据
```

表 7-27 故障诊断数据一览表（MindSpore）

故障诊断分类	故障诊断内容
CANN框架日志和故障诊断数据	HOST侧的INFO及INFO以上级别日志，包括HOST侧CANN软件栈日志、HOST侧驱动日志文件等。
MindSpore框架日志和故障诊断数据	MindSpore框架生成的日志，INFO及INFO以上级别日志。
	RDR(Running Data Recorder)文件。 出现运行异常会自动地导出MindSpore中预先记录的数据以辅助定位运行异常的原因。不同的运行异常将会导出不同的数据。
	analyze_fail.dat，图编译失败自动导出故障信息，用于infer过程分析。

故障诊断分类	故障诊断内容
	dump数据，后端执行期异常触发dump数据。

操作步骤

在创建训练作业页面，选择算法为MindSpore引擎，资源类型为Ascend，可以选择训练模式。

图 7-23 选择算法

图 7-24 选择资源类型

图 7-25 开启故障诊断

7.6.2 训练故障自动恢复

7.6.2.1 训练容错检查

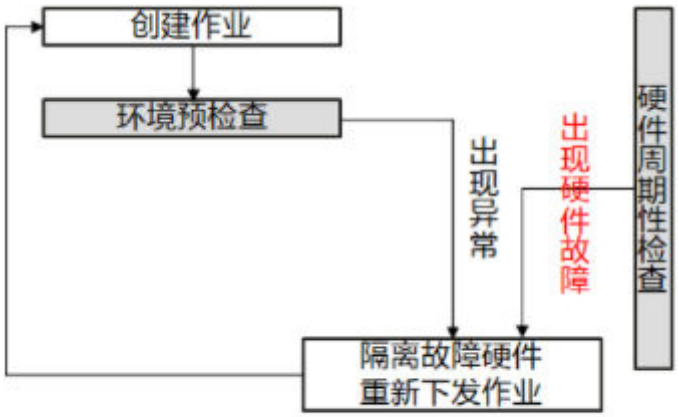
用户在训练模型过程中，存在因硬件故障而产生的训练失败场景。针对硬件故障场景，ModelArts提供容错检查功能，帮助用户隔离故障节点，优化用户训练体验。

容错检查包括两个检查项：环境预检测与硬件周期性检查。当环境预检查或者硬件周期性检查任一检查项出现故障时，隔离故障硬件并重新下发训练作业。针对于分布式场景，容错检查会检查本次训练作业的全部计算节点。

下图中有四个场景，其中场景四为正常训练作业失败场景，其他三个场景下可开启容错功能进行训练作业自动恢复。

- 场景一：环境预检测失败、硬件检测出现故障，隔离所有故障节点并重新下发训练作业。

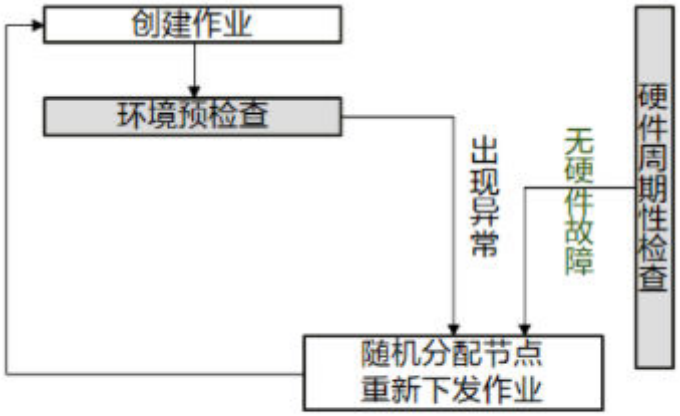
图 7-26 预检失败&硬件故障



1. 预检失败&硬件故障

- 场景二：环境预检测失败、硬件无故障，隔离所有故障节点并重新下发训练作业。

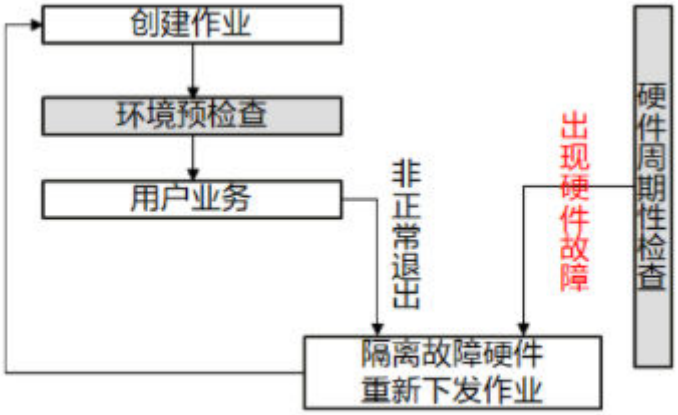
图 7-27 预检失败&硬件正常



2. 预检失败&硬件正常

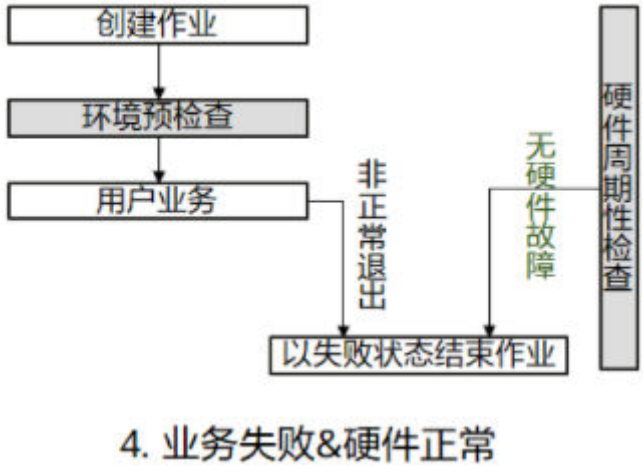
- 场景三：环境预检测成功并进入用户业务阶段，硬件检测出现故障并且用户业务非正常退出，隔离所有故障节点并重新下发训练作业。

图 7-28 业务失败&硬件故障



- 场景四：环境预检测成功并进入用户业务阶段，硬件无故障，当用户业务异常时以失败状态结束作业。

图 7-29 业务失败&硬件正常



隔离故障节点后，系统会在新的计算节点上重新创建训练作业。如果资源池规格紧张，重新下发的训练作业会以第一优先级进行排队。如果排队时间超过30分钟，训练任务会自动退出。该现象表明资源池规格任务紧张，训练作业无法正常启动，推荐您购买专属资源池补充计算节点。

如果您使用专属资源池创建训练作业，容错检查识别的故障节点会被剔除。系统自动补充健康的计算节点至专属资源池。（该功能即将上线）

容错检查详细介绍请参考：

1. **开启容错检查**
2. **检测项目与执行条件**
3. **触发容错环境检测达到的效果**

4. 环境预检查通过后，如果发生硬件故障会导致用户业务中断。您可以在训练中补充reload ckpt的代码逻辑，使能读取训练中断前保存的预训练模型。指导请参考**断点续训练和增量训练**。

开启容错检查

用户可以在创建训练作业时通过设置故障自动重启的方式开启容错检查。

- 使用ModelArts控制台的创建训练作业页面设置故障自动重启：

  用户可以在控制台页面通过开关的方式开启故障自动重启。“故障自动重启”开关默认不开启，表示不做重新下发作业，也不会启用环境检测。打开开关后，允许设置重启次数为1~3次。

- 使用API接口设置容错检查：

  用户可以通过API接口的方式开启故障自动重启。创建训练作业时，在“metadata”字段的“annotations”中传入“fault-tolerance/job-retry-num”字段。

  添加“fault-tolerance/job-retry-num”字段，视为开启故障自动重启，value的范围可以设置为1~3的整数。value值表示最大允许重新下发作业的次数。如果不传入则默认为0，表示不做重新下发作业，也不会启用环境检测。

  图 7-30 设置 API

```
{
    "kind": "job",
    "metadata": {
        "annotations": {
            "fault-tolerance/job-retry-num": "3"
        },
```

检测项目与执行条件

检测项目	item（日志关键字）	执行条件	检测成功要求
域名检测	dns	无	volcano容器的域名都解析成功（/etc/volcano下的“.host”文件中的域名解析成功）
磁盘空间-容器根目录	disk-size root	无	大于32GB
磁盘空间-/dev/shm目录	disk-size shm	无	大于1GB
磁盘空间-/cache目录	disk-size cache	无	大于32GB

检测项目	item（日志关键字）	执行条件	检测成功要求
ulimit检查	ulimit	使用IB网络时	• max locked memory > 16000 • open files > 1000000 • stack size > 8000 • max user processes > 1000000
gpu检查	gpu-check	使用gpu，且使用v2训练引擎时	检测到gpu

触发容错环境检测达到的效果

- 容错检查正常通过时，会打印检测项目的日志，表示具体涉及的检查项目成功。您可以通过在日志中搜索“item”关键字查看。当容错检查正常通过时，可以减少运行故障上报问题。

```
[Modelarts Service Log][task] Detect
[Modelarts Service Log][INFO][detect] code: 0, message: ok, item: dns
[Modelarts Service Log][INFO][detect] code: 0, message: ok, item: disk-size root
[Modelarts Service Log][INFO][detect] code: 0, message: ok, item: disk-size shm
[Modelarts Service Log][INFO][detect] code: 0, message: ok, item: disk-size cache
[Modelarts Service Log][init] download code_url: s3://test-qianjiajun/tolerance_test/
```

- 容错检查失败时，会打印检查失败的日志。您可以通过在日志中搜索“item”关键字查看失败信息。

```
[Modelarts Service Log][init] running
[Modelarts Service Log][init] ip of the pod: 172.16.0.160
[Modelarts Service Log][INFO]detect item: dns; json:{"code": 0, "message": "ok"}
[Modelarts Service Log][INFO][task][detect] code: 0, message: ok, item: dns
[Modelarts Service Log][INFO]detect item: disk-size root; json:{"code": 13, "message": "the disk space of the path
\"/\" is 4892852224, which is less than 34359738368"}
[Modelarts Service Log][ERROR][task][detect] code: 13, message: the disk space of the path "/" is 4892852224, which is
less than 34359738368, item: disk-size root
[Modelarts Service Log][init] exiting...
[Modelarts Service Log][init] wait python processes exit...
```

如果作业重启次数没有达到设定的次数，则会自动做重新下发作业。您可以通过搜索“error,exiting”关键字查找作业重启失败结束的日志。

使用 reload ckpt 恢复中断的训练

在容错机制下，如果因为硬件问题导致训练作业重启，用户可以在代码中读取预训练模型，恢复至重启前的训练状态。用户需要在代码里加上reload ckpt的代码，使能读取训练中断前保存的预训练模型。具体请参见**断点续训练和增量训练**。

7.6.2.2 故障临终遗言

使用场景

随着模型规模和数据集的急剧增长，需要利用大规模的训练集训练大规模的神经网络。在大规模集群分布式训练时，会遇到集群中某个芯片、某台服务器故障，导致分

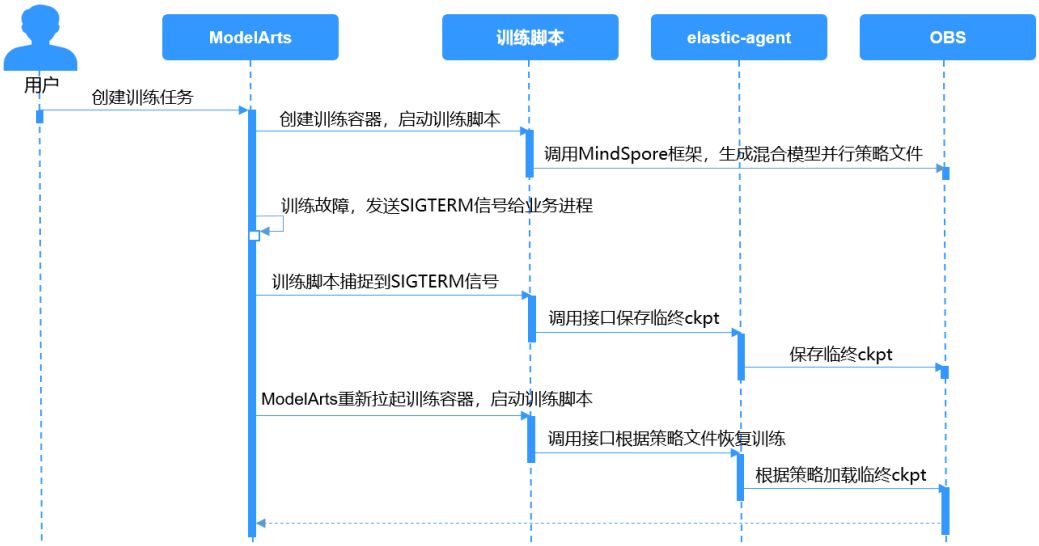
布式训练任务失败。临终遗言是指中断的训练任务支持自动恢复，并可以在上一次训练中断的基础上继续训练，而不用从头开始。

约束限制

表 7-28 约束限制

资源规格	Ascend
训练框架	MindSpore

特性原理



临终遗言处理流程如下：

1. 用户在ModelArts控制台创建训练任务。
2. 训练引擎创建训练容器，启动训练脚本。
3. 训练脚本启动后，调用MindSpore框架，生成混合并行策略文件strategy.proto，该文件记录了混合并行场景下，算子在NPU卡上的分布情况。
4. 训练出现故障后，ModelArts训练组件对当前业务进程发送SIGTERM信号。
5. 训练脚本捕获到SIGTERM信号，调用elastic-agent模块，该模块会调用mindspore框架，生成临终ckpt。
6. ModelArts训练组件重新拉起训练容器，启动训练脚本。
7. 训练脚本调用elastic-agent模块，该模块根据configmap中故障NPU信息和strategy.proto文件生成策略恢复文件。
8. 训练脚本根据策略恢复文件，加载临终ckpt进行续训练。

在数据并行场景下，也是类似的流程，只是更为简单，无需生成并行策略文件和策略恢复文件，只要保存和加载临终ckpt文件即可。

特性使用操作

1. **安装临终遗言二进制包**

   通过ma_pre_start.sh安装whl包。

```
echo "[ma-pre-start] Enter the input directory"
cd /home/ma-user/modelarts/inputs/data_url_0/
echo "[ma-pre-start] Start to install mindx-elastic 0.0.1版本"
export PATH=/home/ma-user/anaconda/bin:$PATH
pip install ./mindx_elastic-0.0.1-py3-none-any.whl
echo "[ma-pre-start] Clean run package"
sudo rm -rf ./script ./*.run ./run_package *.whl
echo "[ma-pre-start] Set ENV"
export GLOG_v=2    # 当前使用诊断模式需要用户手动设置成INFO日志级别 echo "[ma-pre-start] End"
```

2. **创建训练任务**

   - 约束：MindSpore版本要求1.6.0及以上。
   - 修改样例代码，增加如下内容：

```
# 载入依赖接口
from mindx_elastic.terminating_message import ExceptionCheckpoint
...

if args_opt.do_train:
dataset = create_dataset()
loss_cb = LossMonitor()
cb = [loss_cb]
if int(os.getenv('RANK_ID')) == 0:
batch_num = dataset.get_dataset_size()
# 开启临终遗言保存
config_ck = CheckpointConfig(save_checkpoint_steps=batch_num,
keep_checkpoint_max=35,
async_save=True,
append_info=[{"epoch_num": cur_epoch_num}],
exception_save=True)

ckpoint_cb = ModelCheckpoint(prefix="train_resnet_cifar10",
directory=args_opt.train_url,
config=config_ck)
# 定义临终遗言ckpt保存callback
ckpoint_exp = ExceptionCheckpoint(
prefix="train_resnet_cifar10",
directory=args_opt.train_url,
config=config_ck)
# 添加临终遗言ckpt保存callback
cb += [ckpoint_cb, ckpoint_exp]
```

7.6.3 断点续训练和增量训练

什么是断点续训练和增量训练

断点续训练是指因为某些原因（例如容错重启、资源抢占、作业卡死等）导致训练作业还未完成就被中断，下一次训练可以在上一次的训练基础上继续进行。这种方式对于需要长时间训练的模型而言比较友好。

增量训练是指增加新的训练数据到当前训练流程中，扩展当前模型的知识和能力。

断点续训练和增量训练均是通过checkpoint机制实现。

checkpoint的机制是：在模型训练的过程中，不断地保存训练结果（包括但不限于EPOCH、模型权重、优化器状态、调度器状态）。即便模型训练中断，也可以基于checkpoint接续训练。

当需要从训练中断的位置接续训练，只需要加载checkpoint，并用checkpoint信息初始化训练状态即可。用户需要在代码里加上reload ckpt的代码，使能读取前一次训练保存的预训练模型。

ModelArts 中如何实现断点续训练和增量训练

在ModelArts训练中实现断点续训练或增量训练，建议使用“训练输出”功能。

在创建训练作业时，设置训练“输出”参数为“train_url”，在指定的训练输出的数据存储位置中保存**checkpoint**，且**“预下载至本地目录”**选择“下载”。选择预下载至本地目录时，系统在训练作业启动前，自动将数据存储位置中的**checkpoint**文件下载到训练容器的本地目录。

断点续训练建议和训练容错检查（即故障自动重启）功能同时使用。在创建训练作业页面，开启“故障自动重启”开关。训练环境预检测失败、或者训练容器硬件检测故障、或者训练作业失败时会自动重新下发并运行训练作业。

MindSpore 版 reload ckpt

```
import os
import argparse
parser.add_argument("--train_url", type=str)
args = parser.parse_known_args()
# train_url 将被赋值为"/home/ma-user/modelarts/outputs/train_url_0"
train_url = args.train_url

# 初始定义的网络、损失函数及优化器
net = resnet50(args_opt.batch_size, args_opt.num_classes)
ls = SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
opt = Momentum(filter(lambda x: x.requires_grad, net.get_parameters()), 0.01, 0.9)
# 首次训练的epoch初始值，mindspore1.3及以后版本会支持定义epoch_size初始值
# cur_epoch_num = 0
# 判断输出obs路径中是否有模型文件。如果无文件则默认从头训练，如果有模型文件，则加载epoch值最大的ckpt文件当做预训练模型。
if os.listdir(train_url):
    last_ckpt = sorted([file for file in os.listdir(train_url) if file.endswith(".ckpt")])[-1]
    print('last_ckpt:', last_ckpt)
    last_ckpt_file = os.path.join(train_url, last_ckpt)
     # 加载断点
    param_dict = load_checkpoint(last_ckpt_file)
    print('> load last ckpt and continue training!!')
    # 加载模型参数到net
    load_param_into_net(net, param_dict)
    # 加载模型参数到opt
    load_param_into_net(opt, param_dict)

    # 获取保存的epoch值，模型会在此epoch的基础上继续训练,此参数在mindspore1.3及以后版本会支持
    # if param_dict.get("epoch_num"):
    #     cur_epoch_num = int(param_dict["epoch_num"].data.asnumpy())
model = Model(net, loss_fn=ls, optimizer=opt, metrics={'acc'})
# as for train, users could use model.train
if args_opt.do_train:
    dataset = create_dataset()
    batch_num = dataset.get_dataset_size()
    config_ck = CheckpointConfig(save_checkpoint_steps=batch_num,
                                 keep_checkpoint_max=35)
    # append_info=[{"epoch_num": cur_epoch_num}],mindspore1.3及以后版本会支持append_info参数，保存当前时刻的epoch值
    ckpoint_cb = ModelCheckpoint(prefix="train_resnet_cifar10",
                                 directory=args_opt.train_url,
                                 config=config_ck)
    loss_cb = LossMonitor()
    model.train(epoch_size, dataset, callbacks=[ckpoint_cb, loss_cb])
    # model.train(epoch_size-cur_epoch_num, dataset, callbacks=[ckpoint_cb, loss_cb])，mindspore1.3及以后版本支持从断点恢复训练
```

7.6.4 训练作业卡死检测

什么是训练作业卡死检测

训练作业在运行中可能会因为某些未知原因导致作业卡死，如果不能及时发现，就会导致无法及时释放资源，从而造成极大的资源浪费。为了节省训练资源成本，提高使用体验，ModelArts提供了卡死检测功能，能自动识别作业是否卡死，并在日志详情界面上展示，同时能配置通知及时提醒用户作业卡死。

检测规则

卡死检测主要是通过监控作业进程的状态和资源利用率来判定作业是否卡死。会启动一个进程来周期性地监控上述两个指标的变化情况。

- 进程状态：只要训练作业中存在进程IO有变化，进入下一个检测周期。如果在多个检测周期内，作业所有进程IO都没有变化，则进入资源利用率检测阶段。
- 资源利用率：在作业进程IO没有变化的情况下，采集一定时间段内的GPU利用率，并根据这段时间内的GPU利用率的方差和中位数来判断资源使用率是否有变化。如果没有变化，则判定作业卡死。

约束限制

目前卡死检测仅支持资源类型为GPU的训练作业。

操作步骤

卡死检测无需额外配置，作业运行中会自动执行检测。检测到作业卡死后会在训练作业详情页提示作业疑似卡死。如需检测到卡死后发送通知（短信、邮件等）请在作业创建页面配置事件通知。

常见案例

训练作业卡死常见现象的案例和解决方案如下：

复制数据卡死

训练前卡死

训练中途卡死

训练最后一个epoch卡死

7.6.5 修改训练作业优先级

使用专属资源池（New）进行训练作业时，支持在创建训练作业时设置任务优先级，也支持作业在长时间处于“等待中”的状态时调整优先级。如通过调整作业优先级可以减少作业的排队时长。

什么是训练作业优先级

在用户运行训练作业过程中，有需要对训练任务（也叫训练作业）做优先级划分。比如有一些任务是低优先级，可能是跑一些测试、也可能是跑一些简单的不重要的实验。在这类场景下，当有高优先级任务的时候，需要能比低优先级任务更快进入排队队列。

在资源使用高峰期，用户可以通过提供或降低训练作业的优先级，来动态调节作业的执行顺序，保障关键业务的及时运行。

约束限制

- 仅使用新版专属资源池训练时才支持设置训练作业优先级。公共资源池和旧版专属资源池均不支持设置训练作业优先级。
- 作业优先级取值为1~3，默认优先级为1，最高优先级为3。默认用户权限可选择优先级1和2，配置了“**设置作业为高优先级权限**”的用户可选择优先级1~3。

如何设置训练作业优先级

在创建训练作业页面可以设置训练的“作业优先级”。取值为1~3，默认优先级为1，最高优先级为3。

如何修改训练作业优先级

在训练作业列表页面，选择“状态”为“等待中”的训练作业，单击“作业优先级”列的  ，在弹窗中修改优先级后单击“确定”。

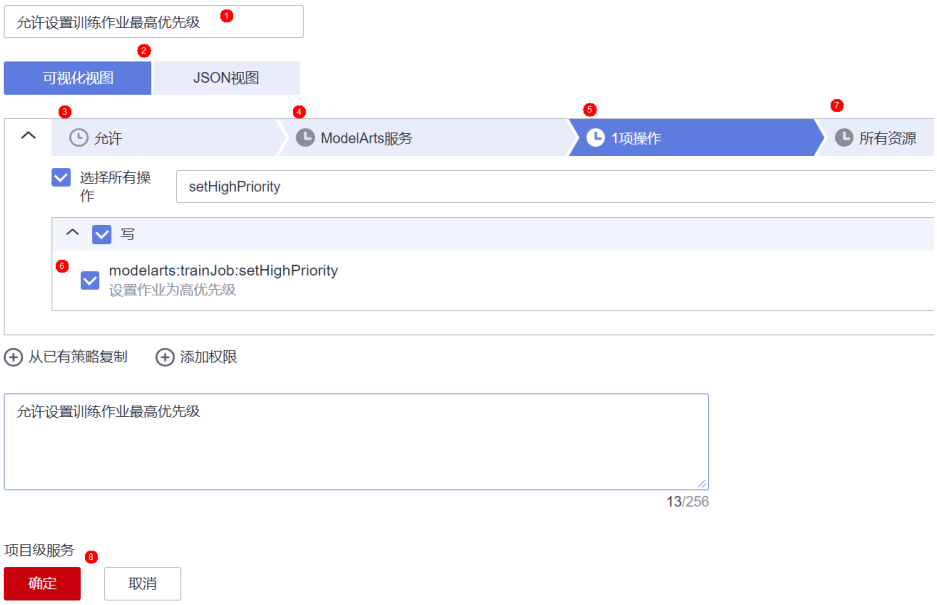
7.6.6 设置作业为高优先级权限

使用专属资源池（New）进行训练的作业，支持在创建训练作业时设置任务优先级，支持在作业排队过程中调整优先级。优先级取值为1~3，默认优先级为1，最高优先级为3。默认用户权限可选择优先级1和2，配置了"设置作业为高优先级"权限的用户可选择优先级1~3。

给子账号配置"设置作业为高优先级"权限

1. 使用主用户账号登录管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。
2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”按如下要求设置完成后单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：允许用户设置训练作业最高优先级。
   - “策略配置方式”：选择可视化视图。
   - “策略内容”：允许，云服务中搜索ModelArts服务并选中，操作列中搜索关键词“modelarts:trainJob:setHighPriority”并选中，所有资源选择默认值。

图 7-31 创建自定义策略



3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限通过Cloud Shell登录运行中的训练作业容器。

   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

7.7 分布式训练

7.7.1 分布式训练功能介绍

ModelArts提供了如下能力：

- 丰富的官方预置镜像，满足用户的需求。
- 支持基于预置镜像自定义制作专属开发环境，并保存使用。
- 丰富的教程，帮助用户快速适配分布式训练，使用分布式训练极大减少训练时间。
- 分布式训练调测的能力，可在PyCharm/VSCode/JupyterLab等开发工具中调试分布式训练。

约束限制

- 如果切换了Notebook的规格，那么只能在Notebook进行单机调测，不能进行分布式调测，也不能提交远程训练任务。
- 当前仅支持Pytorch和MindSpore AI框架，如果MindSpore要进行多机分布式训练调试，则每台机器上都必须有8张卡。

- 本文档提供的调测代码中涉及到的OBS路径，请用户替换为自己的实际OBS路径。
- 本文档提供的调测代码是以Pytorch为例编写的，不同的AI框架之间，整体流程是完全相同的，只需要修改个别的参数即可。

相关章节

- **单机多卡数据并行-DataParallel(DP)**：介绍单机多卡数据并行分布式训练原理和代码改造点。
- **多机多卡数据并行-DistributedDataParallel(DDP)**：介绍多机多卡数据并行分布式训练原理和代码改造点。
- **分布式调测适配及代码示例**：提供了分布式训练调测具体的代码适配操作过程和代码示例。
- **分布式训练完整代码示例**：针对Resnet18在cifar10数据集上的分类任务，给出了分布式训练改造(DDP)的完整代码示例，供用户学习参考。

7.7.2 单机多卡数据并行-DataParallel(DP)

本章节介绍基于Pytorch引擎的单机多卡数据并行训练。

MindSpore引擎的分布式训练参见**MindSpore官网**。

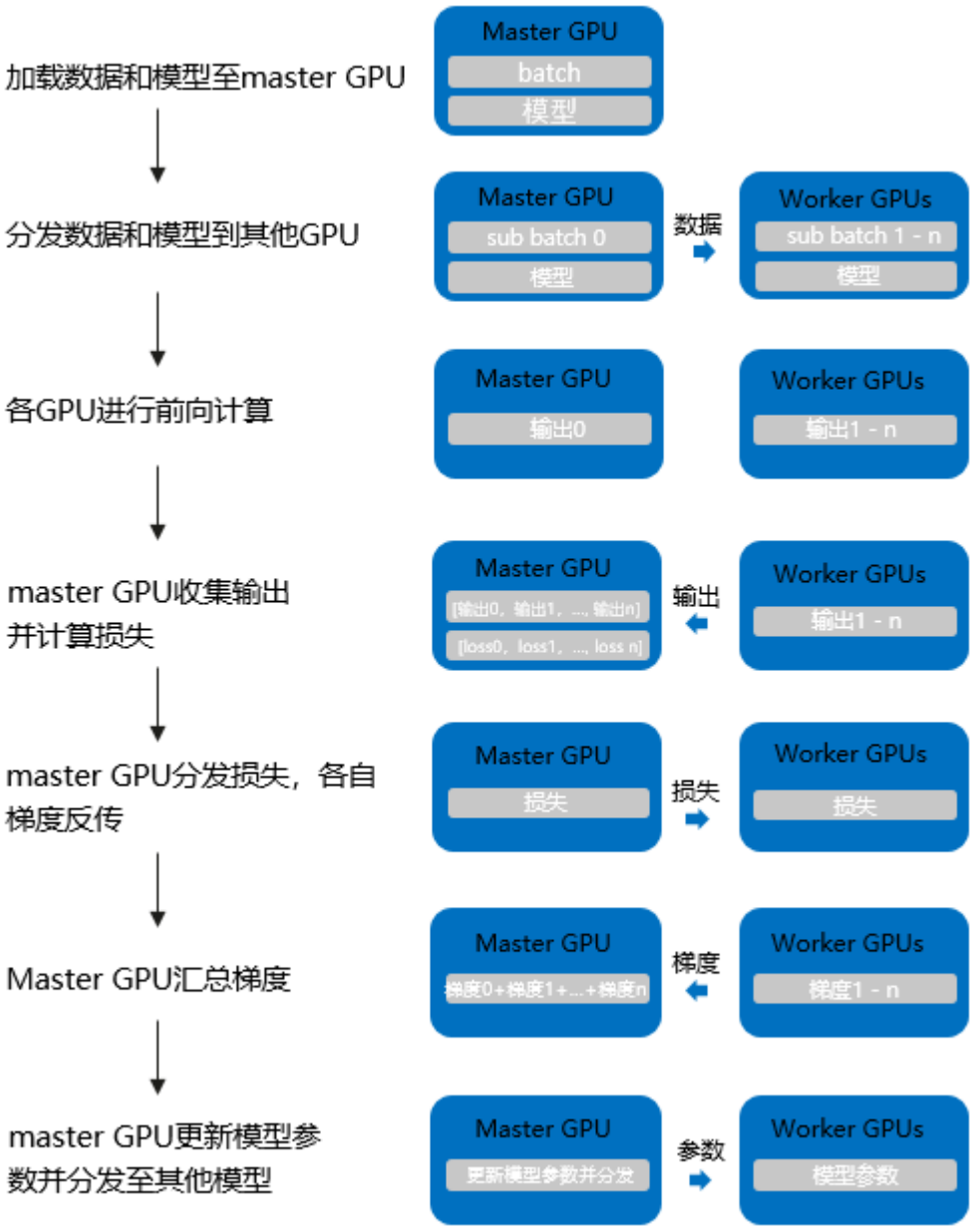
训练流程简述

单机多卡数据并行训练流程介绍如下：

1. 将模型复制到多个GPU上
2. 将一个Batch的数据均分到每一个GPU上
3. 各GPU上的模型进行前向传播，得到输出
4. 主GPU（逻辑序号为0）收集各GPU的输出，汇总后计算损失
5. 分发损失，各GPU各自反向传播梯度
6. 主GPU收集梯度并更新参数，将更新后的模型参数分发到各GPU

具体流程图如下：

图 7-32 单机多卡数据并行训练



DataParallel 进行单机多卡训练的优缺点

- 代码简单：仅需修改一行代码。
- 通信瓶颈 ： 负责reducer的GPU更新模型参数后分发到不同的GPU，因此有较大的通信开销。
- GPU负载不均衡：负责reducer的GPU需要负责汇总输出、计算损失和更新权重，因此显存和使用率相比其他GPU都会更高。

代码改造点

模型分发：DataParallel(model)

完整代码由于代码变动较少，此处进行简略介绍。

```
import torch
class Net(torch.nn.Module):
    pass

model = Net().cuda()

### DataParallel Begin ###
model = torch.nn.DataParallel(Net().cuda())
### DataParallel End ###
```

7.7.3 多机多卡数据并行-DistributedDataParallel(DDP)

本章节介绍基于Pytorch引擎的多机多卡数据并行训练。

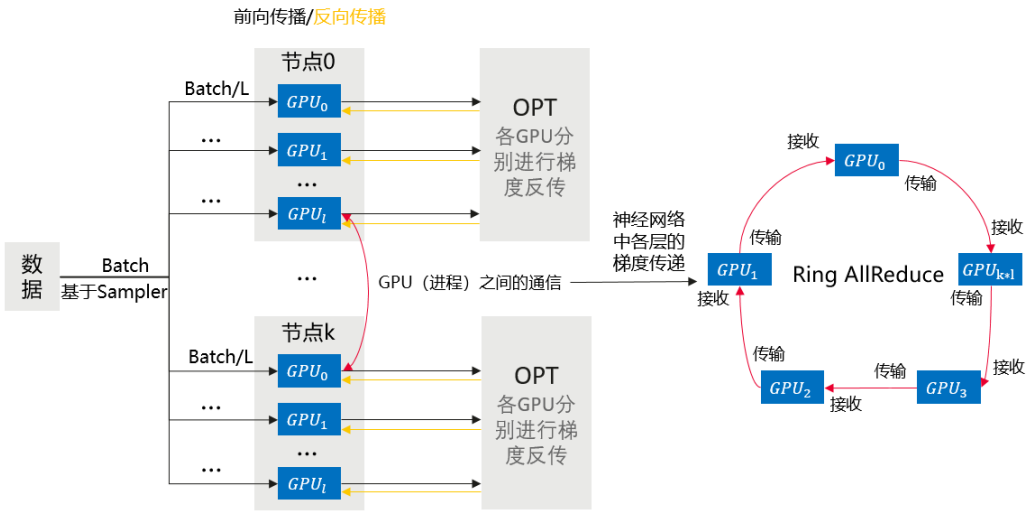
训练流程简述

相比于DP，DDP能够启动多进程进行运算，从而大幅度提升计算资源的利用率。可以基于torch.distributed实现真正的分布式计算，具体的原理此处不再赘述。大致的流程如下：

1. 初始化进程组。
2. 创建分布式并行模型，每个进程都会有相同的模型和参数。
3. 创建数据分发Sampler，使每个进程加载一个mini batch中不同部分的数据。
4. 网络中相邻参数分桶，一般为神经网络模型中需要进行参数更新的每一层网络。
5. 每个进程前向传播并各自计算梯度。
6. 模型某一层的参数得到梯度后会马上进行通讯并进行梯度平均。
7. 各GPU更新模型参数。

具体流程图如下：

图 7-33 多机多卡数据并行训练



DistributedDataParallel 进行多机多卡训练的优缺点

- **通信更快**：相比于DP，通信速度更快
- **负载相对均衡**：相比于DP，GPU负载相对更均衡
- **运行速度快**：因为通信时间更短，效率更高，能更快速的完成训练任务

代码改造点

- 引入多进程启动机制：初始化进程
- 引入几个变量：tcp协议，rank进程序号，worldsize开启的进程数量
- 分发数据：DataLoader中多了一个Sampler参数，避免不同进程数据重复
- 模型分发：DistributedDataParallel(model)
- 模型保存：在序号为0的进程下保存模型

```
import torch
class Net(torch.nn.Module):
    pass

model = Net().cuda()

### DistributedDataParallel Begin ###
model = torch.nn.parallel.DistributedDataParallel(Net().cuda())
### DistributedDataParallel End ###
```

相关操作

- 分布式训练调测具体的代码适配操作过程和代码示例请参见**分布式调测适配及代码示例**章节。
- 文档还针对Resnet18在cifar10数据集上的分类任务，给出了分布式训练改造(DDP)的完整代码示例，供用户学习参考，具体请参见**分布式训练完整代码示例**。

7.7.4 分布式调测适配及代码示例

在DistributedDataParallel中，不同进程分别从原始数据中加载batch的数据，最终将各个进程的梯度进行平均作为最终梯度，由于样本量更大，因此计算出的梯度更加可靠，可以适当增大学习率。

以下对resnet18在cifar10数据集上的分类任务，给出了单机训练和分布式训练改造(DDP)的代码。直接执行代码为多节点分布式训练且支持CPU分布式和GPU分布式，将代码中的分布式改造点注释掉后即可进行单节点单卡训练。

训练代码中包涵三部分入参，分别为训练基础参数、分布式参数和数据相关参数。其中分布式参数由平台自动入参，无需自行定义。数据相关参数中的custom_data表示是否使用自定义数据进行训练，该参数为“true”时使用基于torch自定义的随机数据进行训练和验证。

数据集

cifar10数据集

在Notebook中，无法直接使用默认版本的torchvision获取数据集，因此示例代码中提供了三种训练数据加载方式。

cifar-10数据集**下载链接**，单击“CIFAR-10 python version”。

- 尝试基于torchvision获取cifar10数据集。
- 基于数据链接下载数据并解压，放置在指定目录下，训练集和测试集的大小分别为(50000，3，32，32)和(10000，3，32，32)。
- 考虑到下载cifar10数据集较慢，基于torch生成类似cifar10的随机数据集，训练集和测试集的大小分别为(5000，3，32，32)和(1000，3，32，32)，标签仍为10类，指定custom_data = 'true'后可直接进行训练任务，无需加载数据。

训练代码

以下代码中以“### 分布式改造，... ###”注释的代码即为多节点分布式训练需要适配的代码改造点。

不对示例代码进行任何修改，适配数据路径后即可在ModelArts上完成多节点分布式训练。

注释掉分布式代码改造点，即可完成单节点单卡训练。完整代码见**分布式训练完整代码示例**。

- **导入依赖包**

```
import datetime
import inspect
import os
import pickle
import random

import argparse
import numpy as np
import torch
import torch.distributed as dist
from torch import nn, optim
from torch.utils.data import TensorDataset, DataLoader
from torch.utils.data.distributed import DistributedSampler
from sklearn.metrics import accuracy_score
```

- **定义加载数据的方法和随机数**，由于加载数据部分代码较多，此处省略

```
def setup_seed(seed):
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    np.random.seed(seed)
    random.seed(seed)
    torch.backends.cudnn.deterministic = True

def get_data(path):
    pass
```

- **定义网络结构**

```
class Block(nn.Module):

    def __init__(self, in_channels, out_channels, stride=1):
        super().__init__()
        self.residual_function = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(out_channels)
        )

        self.shortcut = nn.Sequential()
        if stride != 1 or in_channels != out_channels:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(out_channels)
            )

    def forward(self, x):
        out = self.residual_function(x) + self.shortcut(x)
        return nn.ReLU(inplace=True)(out)


class ResNet(nn.Module):

    def __init__(self, block, num_classes=10):
        super().__init__()
        self.conv1 = nn.Sequential(
```

```
            nn.Conv2d(3, 64, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True))
        self.conv2 = self.make_layer(block, 64, 64, 2, 1)
        self.conv3 = self.make_layer(block, 64, 128, 2, 2)
        self.conv4 = self.make_layer(block, 128, 256, 2, 2)
        self.conv5 = self.make_layer(block, 256, 512, 2, 2)
        self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
        self.dense_layer = nn.Linear(512, num_classes)

    def make_layer(self, block, in_channels, out_channels, num_blocks, stride):
        strides = [stride] + [1] * (num_blocks - 1)
        layers = []
        for stride in strides:
            layers.append(block(in_channels, out_channels, stride))
            in_channels = out_channels
        return nn.Sequential(*layers)

    def forward(self, x):
        out = self.conv1(x)
        out = self.conv2(out)
        out = self.conv3(out)
        out = self.conv4(out)
        out = self.conv5(out)
        out = self.avg_pool(out)
        out = out.view(out.size(0), -1)
        out = self.dense_layer(out)
        return out
```

- **进行训练和验证**

```
def main():
    file_dir = os.path.dirname(inspect.getframeinfo(inspect.currentframe()).filename)

    seed = datetime.datetime.now().year
    setup_seed(seed)

    parser = argparse.ArgumentParser(description='Pytorch distribute training',
                                     formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('--enable_gpu', default='true')
    parser.add_argument('--lr', default='0.01', help='learning rate')
    parser.add_argument('--epochs', default='100', help='training iteration')

    parser.add_argument('--init_method', default=None, help='tcp_port')
    parser.add_argument('--rank', type=int, default=0, help='index of current task')
    parser.add_argument('--world_size', type=int, default=1, help='total number of tasks')

    parser.add_argument('--custom_data', default='false')
    parser.add_argument('--data_url', type=str, default=os.path.join(file_dir, 'input_dir'))
    parser.add_argument('--output_dir', type=str, default=os.path.join(file_dir, 'output_dir'))
    args, unknown = parser.parse_known_args()

    args.enable_gpu = args.enable_gpu == 'true'
    args.custom_data = args.custom_data == 'true'
    args.lr = float(args.lr)
    args.epochs = int(args.epochs)

    if args.custom_data:
        print('[warning] you are training on custom random dataset, '
              'validation accuracy may range from 0.4 to 0.6.')

    ### 分布式改造，DDP初始化进程，其中init_method, rank和world_size参数均由平台自动入参 ###
    dist.init_process_group(init_method=args.init_method, backend="nccl", world_size=args.world_size,
rank=args.rank)
    ### 分布式改造，DDP初始化进程，其中init_method, rank和world_size参数均由平台自动入参 ###

    tr_set, val_set = get_data(args.data_url, custom_data=args.custom_data)

    batch_per_gpu = 128
    gpus_per_node = torch.cuda.device_count() if args.enable_gpu else 1
    batch = batch_per_gpu * gpus_per_node
```

```
    tr_loader = DataLoader(tr_set, batch_size=batch, shuffle=False)

    ### 分布式改造，构建DDP分布式数据sampler，确保不同进程加载到不同的数据 ###
    tr_sampler = DistributedSampler(tr_set, num_replicas=args.world_size, rank=args.rank)
    tr_loader = DataLoader(tr_set, batch_size=batch, sampler=tr_sampler, shuffle=False, drop_last=True)
    ### 分布式改造，构建DDP分布式数据sampler，确保不同进程加载到不同的数据 ###

    val_loader = DataLoader(val_set, batch_size=batch, shuffle=False)

    lr = args.lr * gpus_per_node
    max_epoch = args.epochs
    model = ResNet(Block).cuda() if args.enable_gpu else ResNet(Block)

    ### 分布式改造，构建DDP分布式模型 ###
    model = nn.parallel.DistributedDataParallel(model)
    ### 分布式改造，构建DDP分布式模型 ###

    optimizer = optim.Adam(model.parameters(), lr=lr)
    loss_func = torch.nn.CrossEntropyLoss()

    os.makedirs(args.output_dir, exist_ok=True)

    for epoch in range(1, max_epoch + 1):
        model.train()
        train_loss = 0

        ### 分布式改造，DDP sampler, 基于当前的epoch为其设置随机数，避免加载到重复数据 ###
        tr_sampler.set_epoch(epoch)
        ### 分布式改造，DDP sampler, 基于当前的epoch为其设置随机数，避免加载到重复数据 ###

        for step, (tr_x, tr_y) in enumerate(tr_loader):
            if args.enable_gpu:
                tr_x, tr_y = tr_x.cuda(), tr_y.cuda()
            out = model(tr_x)
            loss = loss_func(out, tr_y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            train_loss += loss.item()
        print('train | epoch: %d | loss: %.4f' % (epoch, train_loss / len(tr_loader)))

        val_loss = 0
        pred_record = []
        real_record = []
        model.eval()
        with torch.no_grad():
            for step, (val_x, val_y) in enumerate(val_loader):
                if args.enable_gpu:
                    val_x, val_y = val_x.cuda(), val_y.cuda()
                out = model(val_x)
                pred_record += list(np.argmax(out.cpu().numpy(), axis=1))
                real_record += list(val_y.cpu().numpy())
                val_loss += loss_func(out, val_y).item()
        val_accu = accuracy_score(real_record, pred_record)
        print('val | epoch: %d | loss: %.4f | accuracy: %.4f' % (epoch, val_loss / len(val_loader), val_accu),
'\n')

        if args.rank == 0:
            # save ckpt every epoch
            torch.save(model.state_dict(), os.path.join(args.output_dir, f'epoch_{epoch}.pth'))


if __name__ == '__main__':
    main()
```

- **结果对比**

  分别以单机单卡和两节点16卡两种资源类型完成100epoch的cifar-10数据集训练，训练时长和测试集准确率如下。

表 7-29 训练结果对比

资源类型	单机单卡	两节点16卡
耗时	60分钟	20分钟
准确率	80+	80+

7.7.5 分布式训练完整代码示例

以下对resnet18在cifar10数据集上的分类任务，给出了分布式训练改造(DDP)的完整代码示例。

训练启动文件main.py内容如下（如果需要执行单机单卡训练任务，则将分布式改造的代码删除）：

```
import datetime
import inspect
import os
import pickle
import random
import logging

import argparse
import numpy as np
from sklearn.metrics import accuracy_score
import torch
from torch import nn, optim
import torch.distributed as dist
from torch.utils.data import TensorDataset, DataLoader
from torch.utils.data.distributed import DistributedSampler

file_dir = os.path.dirname(inspect.getframeinfo(inspect.currentframe()).filename)


def load_pickle_data(path):
    with open(path, 'rb') as file:
        data = pickle.load(file, encoding='bytes')
    return data


def _load_data(file_path):
    raw_data = load_pickle_data(file_path)
    labels = raw_data[b'labels']
    data = raw_data[b'data']
    filenames = raw_data[b'filenames']

    data = data.reshape(10000, 3, 32, 32) / 255
    return data, labels, filenames


def load_cifar_data(root_path):
    train_root_path = os.path.join(root_path, 'cifar-10-batches-py/data_batch_')
    train_data_record = []
    train_labels = []
    train_filenames = []
    for i in range(1, 6):
        train_file_path = train_root_path + str(i)
        data, labels, filenames = _load_data(train_file_path)
        train_data_record.append(data)
```

```
        train_labels += labels
        train_filenames += filenames
    train_data = np.concatenate(train_data_record, axis=0)
    train_labels = np.array(train_labels)

    val_file_path = os.path.join(root_path, 'cifar-10-batches-py/test_batch')
    val_data, val_labels, val_filenames = _load_data(val_file_path)
    val_labels = np.array(val_labels)

    tr_data = torch.from_numpy(train_data).float()
    tr_labels = torch.from_numpy(train_labels).long()
    val_data = torch.from_numpy(val_data).float()
    val_labels = torch.from_numpy(val_labels).long()
    return tr_data, tr_labels, val_data, val_labels


def get_data(root_path, custom_data=False):
    if custom_data:
        train_samples, test_samples, img_size = 5000, 1000, 32
        tr_label = [1] * int(train_samples / 2) + [0] * int(train_samples / 2)
        val_label = [1] * int(test_samples / 2) + [0] * int(test_samples / 2)
        random.seed(2021)
        random.shuffle(tr_label)
        random.shuffle(val_label)
        tr_data, tr_labels = torch.randn((train_samples, 3, img_size, img_size)).float(),
torch.tensor(tr_label).long()
        val_data, val_labels = torch.randn((test_samples, 3, img_size, img_size)).float(),
torch.tensor(
            val_label).long()
        tr_set = TensorDataset(tr_data, tr_labels)
        val_set = TensorDataset(val_data, val_labels)
        return tr_set, val_set
    elif os.path.exists(os.path.join(root_path, 'cifar-10-batches-py')):
        tr_data, tr_labels, val_data, val_labels = load_cifar_data(root_path)
        tr_set = TensorDataset(tr_data, tr_labels)
        val_set = TensorDataset(val_data, val_labels)
        return tr_set, val_set
    else:
        try:
            import torchvision
            from torchvision import transforms
            tr_set = torchvision.datasets.CIFAR10(root='./data', train=True,
                                                  download=True, transform=transforms)
            val_set = torchvision.datasets.CIFAR10(root='./data', train=False,
                                                   download=True, transform=transforms)
            return tr_set, val_set
        except Exception as e:
            raise Exception(
                f"{e}, you can download and unzip cifar-10 dataset manually, "
                "the data url is http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz")


class Block(nn.Module):

    def __init__(self, in_channels, out_channels, stride=1):
        super().__init__()
        self.residual_function = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1,
bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False),
```

```
            nn.BatchNorm2d(out_channels)
        )

        self.shortcut = nn.Sequential()
        if stride != 1 or in_channels != out_channels:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(out_channels)
            )

    def forward(self, x):
        out = self.residual_function(x) + self.shortcut(x)
        return nn.ReLU(inplace=True)(out)


class ResNet(nn.Module):

    def __init__(self, block, num_classes=10):
        super().__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True))
        self.conv2 = self.make_layer(block, 64, 64, 2, 1)
        self.conv3 = self.make_layer(block, 64, 128, 2, 2)
        self.conv4 = self.make_layer(block, 128, 256, 2, 2)
        self.conv5 = self.make_layer(block, 256, 512, 2, 2)
        self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
        self.dense_layer = nn.Linear(512, num_classes)

    def make_layer(self, block, in_channels, out_channels, num_blocks, stride):
        strides = [stride] + [1] * (num_blocks - 1)
        layers = []
        for stride in strides:
            layers.append(block(in_channels, out_channels, stride))
            in_channels = out_channels
        return nn.Sequential(*layers)

    def forward(self, x):
        out = self.conv1(x)
        out = self.conv2(out)
        out = self.conv3(out)
        out = self.conv4(out)
        out = self.conv5(out)
        out = self.avg_pool(out)
        out = out.view(out.size(0), -1)
        out = self.dense_layer(out)
        return out


def setup_seed(seed):
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    np.random.seed(seed)
    random.seed(seed)
    torch.backends.cudnn.deterministic = True


def obs_transfer(src_path, dst_path):
    import moxing as mox
    mox.file.copy_parallel(src_path, dst_path)
    logging.info(f"end copy data from {src_path} to {dst_path}")
```

```
def main():
    seed = datetime.datetime.now().year
    setup_seed(seed)

    parser = argparse.ArgumentParser(description='Pytorch distribute training',
                                     formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('--enable_gpu', default='true')
    parser.add_argument('--lr', default='0.01', help='learning rate')
    parser.add_argument('--epochs', default='100', help='training iteration')

    parser.add_argument('--init_method', default=None, help='tcp_port')
    parser.add_argument('--rank', type=int, default=0, help='index of current task')
    parser.add_argument('--world_size', type=int, default=1, help='total number of tasks')

    parser.add_argument('--custom_data', default='false')
    parser.add_argument('--data_url', type=str, default=os.path.join(file_dir, 'input_dir'))
    parser.add_argument('--output_dir', type=str, default=os.path.join(file_dir, 'output_dir'))
    args, unknown = parser.parse_known_args()

    args.enable_gpu = args.enable_gpu == 'true'
    args.custom_data = args.custom_data == 'true'
    args.lr = float(args.lr)
    args.epochs = int(args.epochs)

    if args.custom_data:
        logging.warning('you are training on custom random dataset, '
            'validation accuracy may range from 0.4 to 0.6.')

    ### 分布式改造，DDP初始化进程，其中init_method, rank和world_size参数均由平台自动入参
###
    dist.init_process_group(init_method=args.init_method, backend="nccl",
world_size=args.world_size, rank=args.rank)
    ### 分布式改造，DDP初始化进程，其中init_method, rank和world_size参数均由平台自动入参
###

    tr_set, val_set = get_data(args.data_url, custom_data=args.custom_data)

    batch_per_gpu = 128
    gpus_per_node = torch.cuda.device_count() if args.enable_gpu else 1
    batch = batch_per_gpu * gpus_per_node

    tr_loader = DataLoader(tr_set, batch_size=batch, shuffle=False)

    ### 分布式改造，构建DDP分布式数据sampler，确保不同进程加载到不同的数据 ###
    tr_sampler = DistributedSampler(tr_set, num_replicas=args.world_size, rank=args.rank)
    tr_loader = DataLoader(tr_set, batch_size=batch, sampler=tr_sampler, shuffle=False,
drop_last=True)
    ### 分布式改造，构建DDP分布式数据sampler，确保不同进程加载到不同的数据 ###

    val_loader = DataLoader(val_set, batch_size=batch, shuffle=False)

    lr = args.lr * gpus_per_node * args.world_size
    max_epoch = args.epochs
    model = ResNet(Block).cuda() if args.enable_gpu else ResNet(Block)

    ### 分布式改造，构建DDP分布式模型 ###
    model = nn.parallel.DistributedDataParallel(model)
    ### 分布式改造，构建DDP分布式模型 ###

    optimizer = optim.Adam(model.parameters(), lr=lr)
```

```
    loss_func = torch.nn.CrossEntropyLoss()

    os.makedirs(args.output_dir, exist_ok=True)

    for epoch in range(1, max_epoch + 1):
        model.train()
        train_loss = 0

        ### 分布式改造，DDP sampler, 基于当前的epoch为其设置随机数，避免加载到重复数据
###
        tr_sampler.set_epoch(epoch)
        ### 分布式改造，DDP sampler, 基于当前的epoch为其设置随机数，避免加载到重复数据
###

        for step, (tr_x, tr_y) in enumerate(tr_loader):
            if args.enable_gpu:
                tr_x, tr_y = tr_x.cuda(), tr_y.cuda()
            out = model(tr_x)
            loss = loss_func(out, tr_y)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            train_loss += loss.item()
        print('train | epoch: %d | loss: %.4f' % (epoch, train_loss / len(tr_loader)))

        val_loss = 0
        pred_record = []
        real_record = []
        model.eval()
        with torch.no_grad():
            for step, (val_x, val_y) in enumerate(val_loader):
                if args.enable_gpu:
                    val_x, val_y = val_x.cuda(), val_y.cuda()
                out = model(val_x)
                pred_record += list(np.argmax(out.cpu().numpy(), axis=1))
                real_record += list(val_y.cpu().numpy())
                val_loss += loss_func(out, val_y).item()
        val_accu = accuracy_score(real_record, pred_record)
        print('val | epoch: %d | loss: %.4f | accuracy: %.4f' % (epoch, val_loss / len(val_loader),
val_accu), '\n')

        if args.rank == 0:
            # save ckpt every epoch
            torch.save(model.state_dict(), os.path.join(args.output_dir, f'epoch_{epoch}.pth'))


if __name__ == '__main__':
    main()
```

常见问题

1、示例代码中如何使用不同的数据集?

- 上述代码如果使用cifar10数据集，则将数据集**下载**并解压后，上传至OBS桶中，文件目录结构如下：

```
DDP
|--- main.py
|--- input_dir
|------ cifar-10-batches-py
|-------- data_batch_1
|-------- data_batch_2
|-------- ...
```

其中“DDP”为创建训练作业时的“代码目录”，“main.py”为上文代码示例（即创建训练作业时的“启动文件”），“cifar-10-batches-py”为解压后的数据集文件夹（放在input_dir文件夹下）。

- 如果使用自定义的随机数据，则将代码示例中的参数“custom_data”改为“true”，修改后内容如下：
  ```
  parser.add_argument('--custom_data', default='true')
  ```
  然后直接运行代码示例“main.py”即可，创建训练作业的参数与上图相同。

2、为什么DDP可以不输入主节点ip?

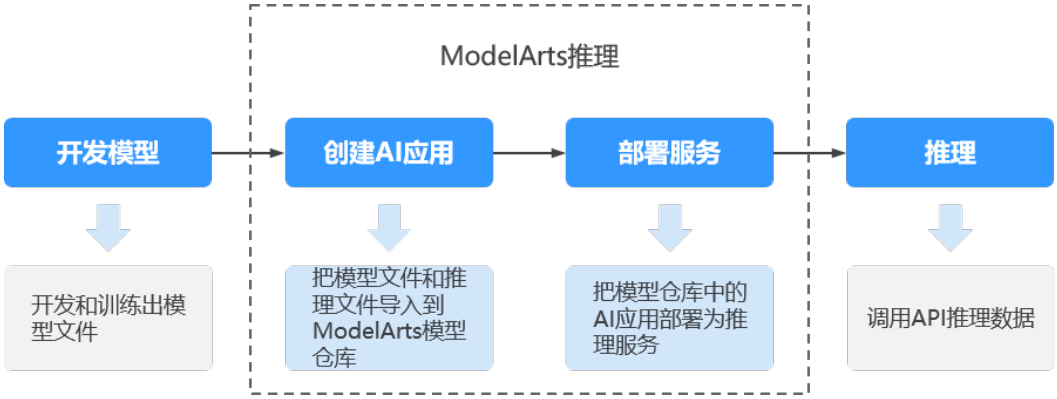
“parser.add_argument('--init_method', default=None, help='tcp_port')”中的init method参数值会包含主节点的ip和端口，由平台自动入参，不需要用户输入主节点的ip和端口。

8 推理部署

8.1 推理简介

AI模型开发完成后，在ModelArts服务中可以将AI模型创建为AI应用，将AI应用快速部署为推理服务，您可以通过调用API的方式把AI推理能力集成到自己的IT平台。

图 8-1 推理简介



- 开发模型：模型开发可以在ModelArts服务中进行，也可以在您的本地开发环境进行，本地开发的模型需要上传到OBS服务。
- 创建AI应用：把模型文件和推理文件导入到ModelArts的模型仓库中，进行版本化管理，并构建为可运行的AI应用。
- 部署服务：把AI应用在资源池中部署为容器实例，注册外部可访问的推理API。
- 推理：在您的应用中增加对推理API的调用，在业务流程中集成AI推理能力。

部署服务

在完成AI应用的创建后，可在“部署上线”页面对AI应用进行部署。ModelArts当前支持如下几种部署类型：

- **在线服务**

  将AI应用部署为一个Web Service，并且提供在线的测试UI与监控功能。

- **批量服务**

  批量服务可对批量数据进行推理，完成数据处理后自动停止。

8.2 管理 AI 应用

8.2.1 管理 AI 应用简介

AI开发和调优往往需要大量的迭代和调试，数据集、训练代码或参数的变化都可能会影响模型的质量，如不能统一管理开发流程元数据，可能会出现无法重现最优模型的现象。

ModelArts的AI应用可导入所有训练生成的元模型、上传至对象存储服务（OBS）中的元模型和容器镜像中的元模型，可对所有迭代和调试的AI应用进行统一管理。

约束与限制

- 自动学习项目中，在完成模型部署后，其生成的模型也将自动上传至AI应用列表中。但是自动学习生成的AI应用无法下载，只能用于部署上线。

创建 AI 应用的几种场景

- **从训练中选择**：在ModelArts中创建训练作业，并完成模型训练，在得到满意的模型后，可以将训练后得到的模型创建为AI应用，用于部署服务。
- **从对象存储服务（OBS）中选择**：如果您使用常用框架在本地完成模型开发和训练，可以将本地的模型按照模型包规范上传至OBS桶中，从OBS将模型导入至ModelArts中，创建为AI应用，直接用于部署服务。
- **从容器镜像中选择**：针对ModelArts目前不支持的AI引擎，可以通过自定义镜像的方式将编写的模型镜像导入ModelArts，创建为AI应用，用于部署服务。

AI 应用的功能描述

表 8-1 AI 应用相关功能

支持的功能	说明
创建AI应用	将训练后的模型导入至ModelArts创建为AI应用，便于进行统一管理，支持如下几种场景的导入方式，不同场景对应的操作指导请参见： • **从训练中选择元模型** • **从容器镜像中选择元模型**
查看AI应用详情	当AI应用创建成功后，您可以进入AI应用详情页查看AI应用的信息。
管理AI应用版本	为方便溯源和模型反复调优，在ModelArts中提供了AI应用版本管理的功能，您可以基于版本对AI应用进行管理。

推理支持的 AI 引擎

在ModelArts创建AI应用时，若使用预置镜像“从模板中选择”或“从OBS中选择”导入模型，则支持如下常用引擎及版本的模型包。

说明

- 标注“推荐”的Runtime来源于统一镜像，后续统一镜像将作为主流的推理基础镜像。
- 推荐将旧版镜像切换为统一镜像，旧版镜像后续将会逐渐下线。
- 待下线的基本镜像不再维护。
- 统一镜像Runtime的命名规范：<AI引擎名字及版本> - <硬件及版本：cpu或cuda或cann> - <python版本> - <操作系统版本> - <CPU架构>

表 8-2 支持的常用引擎及其 Runtime

模型使用的引擎类型	支持的运行环境（Runtime）
TensorFlow	tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64
MindSpore	mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64
Pytorch	pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64

8.2.2 创建 AI 应用

8.2.2.1 从训练中选择元模型

在ModelArts中创建训练作业，并完成模型训练，在得到满意的模型后，可以将训练后得到的模型导入至模型管理，方便统一管理，同时支持将模型快速部署上线为服务。

约束与限制

- 针对使用订阅算法的训练作业，无需推理代码和配置文件，其生成的模型可直接导入ModelArts。
- ARM架构的ModelArts不支持从训练中选择元模型导入。
- 使用容器化部署，导入的元模型有大小限制，详情请参见**导入AI应用对于镜像大小限制**。

前提条件

- 请确保训练作业已运行成功，且模型已存储至训练输出的OBS目录下（输入参数为train_url）。
- 针对使用常用框架或自定义镜像创建的训练作业，需根据**模型包规范介绍**，将推理代码和配置文件上传至模型的存储目录中。
- 确保您使用的OBS目录与ModelArts在同一区域。

创建 AI 应用操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“AI应用管理 > AI应用”，进入AI应用列表页面。

2. 单击左上角的“创建”，进入“创建AI应用”页面。
3. 在“创建AI应用”页面，填写相关参数。
   a. 填写AI应用基本信息，详细参数说明请参见表8-3。

表 8-3 AI 应用基本信息参数说明

参数名称	说明
名称	AI应用名称。支持1～64位可见字符，名称可以包含字母、数字、中划线、下划线。
版本	设置所创建AI应用的版本。第一次导入时，默认为0.0.1。 **说明** AI应用创建完成后，可以通过**创建新版本**，导入不同的元模型进行调优。
描述	AI应用的简要描述。

   b. 填写元模型来源及其相关参数。当“元模型来源”选择“从训练中选择”时，其相关的参数配置请参见表8-4。

表 8-4 元模型来源参数说明

参数	说明
“元模型来源”	选择“从训练中选择>训练作业”或者“从训练中选择>训练作业（New）”。 ● 在“选择训练作业”右侧下拉框中选择当前账号下已完成运行的训练作业及其“版本”。
“AI引擎”	元模型使用的推理引擎，选择训练作业后会自动匹配。
“推理代码”	推理代码自定义AI应用的推理处理逻辑。显示推理代码URL，您可以直接复制此URL使用。
“运行时依赖”	罗列选中模型对环境的依赖。
“AI应用说明”	为了帮助其他AI应用开发者更好的理解及使用您的AI应用，建议您提供AI应用的说明文档。单击“添加AI应用说明”，设置“文档名称”及其“URL”。AI应用说明最多支持3条。
“部署类型”	选择此AI应用支持部署服务的类型，部署上线时只支持部署为此处选择的部署类型，例如此处只选择在线服务，那您导入后只能部署为在线服务。

   c. 确认信息填写无误，单击“立即创建”，完成AI应用的创建。

   在AI应用列表中，您可以查看刚创建的AI应用及其对应的版本。当AI应用状态变更为“正常”时，表示AI应用导入成功。在此页面，您还可以创建新版本、快速部署服务等操作。

后续操作

部署服务：在“AI应用列表”中，单击AI应用名称左侧的单选按钮，在列表页底部展开此AI应用下的“版本列表”。在对应版本所在行，单击“操作”列的部署按钮，可以将AI应用部署上线为创建AI应用时所选择的部署类型。

8.2.2.2 从对象存储服务（OBS）中选择元模型

针对使用常用框架完成模型开发和训练的场景，可以将您的模型导入至ModelArts中，创建为AI应用，并进行统一管理。

约束与限制

- 针对创建AI应用的模型，需符合ModelArts的模型包规范，推理代码和配置文件也需遵循ModelArts的要求，详细说明请参见**模型包规范介绍**、**模型配置文件编写说明**、**模型推理代码编写说明**。
- 使用容器化部署，导入的元模型有大小限制，详情请参见**导入AI应用对于镜像大小限制**。

前提条件

- 已完成模型开发和训练，使用的AI引擎为ModelArts支持的类型和版本，详细请参见**推理支持的AI引擎**。
- 已完成训练的模型包，及其对应的推理代码和配置文件，且已上传至OBS目录中。
- 确保您使用的OBS与ModelArts在同一区域。

创建 AI 应用操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“AI应用管理 > AI应用”，进入AI应用列表页面。
2. 单击左上角的“创建”，进入“创建AI应用”页面。
3. 在“创建AI应用”页面，填写相关参数。
   a. 填写AI应用基本信息，详细参数说明请参见**表8-5**。

   表 8-5 AI 应用基本信息参数说明

参数名称	说明
名称	AI应用名称。支持1～64位可见字符，名称可以包含字母、数字、中划线、下划线。
版本	设置所创建AI应用的版本。第一次导入时，默认为0.0.1。 **说明** AI应用创建完成后，可以通过**创建新版本**，导入不同的元模型进行调优。
描述	AI应用的简要描述。

   b. 填写元模型来源及其相关参数。当“元模型来源”选择“从对象存储服务（OBS）中选择”时，其相关的参数配置请参见**表8-6**。

针对从OBS导入的元模型，ModelArts要求根据**模型包规范**，编写推理代码和配置文件，并将推理代码和配置文件放置元模型存储的“model”文件夹下。如果您选择的目录下不符合模型包规范，将无法创建AI应用。

表 8-6 元模型来源参数说明

参数	说明
“选择元模型”	选择元模型存储的OBS路径。 OBS路径不能含有空格，否则创建AI应用会失败。
“AI引擎”	根据您选择的元模型存储路径，将自动关联出元模型使用的“AI引擎”。 如果“AI引擎”是Custom引擎时，需要配置容器调用接口，用于指定模型启动的协议和端口号。固定请求协议是HTTPS，端口号为8080。
“健康检查”	用于指定模型的健康检查。选择了“AI引擎”和“运行环境”后，部分支持健康检查的引擎会显示该参数。使用Custom引擎时，需要在镜像中配置健康检查接口，否则会导致服务部署失败。 ● **检查方式：**可以选择“HTTP请求检查”或者“执行命令检查”。 使用Custom引擎时，支持选择“HTTP请求检查”或者“执行命令检查”。 使用非Custom引擎时，仅支持选择“HTTP请求检查”。 ● **健康检查URL**：“检查方式”选择“HTTP请求检查”时显示，填写健康检查的URL，默认值为“/health”。 ● **健康检查命令：**“检查方式”选择“执行命令检查”时显示，填写健康检查的命令。 ● **健康检查周期**：填写1-2147483647之前的整数，单位为秒。 ● **延迟时间（秒）：**实例启动后，延迟执行健康检查的时间。填写0-2147483647之间的整数，单位为秒，不能为空。 ● **健康检查最大失败次数**：填写1-2147483647之间的整数。在服务启动阶段，当健康检查请求连续失败达到所填次数后，服务会进入异常状态；在服务运行阶段，当健康检查请求连续失败达到所填次数后，服务会进入告警状态。 **说明** 使用Custom引擎时需要符合自定义引擎规范，请参见**使用自定义引擎创建AI应用**。 当AI应用配置了健康检查，部署的服务在收到停止指令后，会延后3分钟才停止。
“运行时依赖”	罗列选中模型对环境的依赖。

参数	说明
“AI应用说明”	为了帮助其他AI应用开发者更好的理解及使用您的AI应用，建议您提供AI应用的说明文档。单击“添加AI应用说明”，设置“文档名称”及其“URL”。AI应用说明支持增加3条。
“配置文件”	系统默认关联您存储在OBS中的配置文件。打开开关，您可以直接在当前界面查看或编辑模型配置文件。 **说明** 该功能即将下线，后续请根据“AI引擎”、“运行时依赖”和“apis定义”修改模型的配置信息。
“部署类型”	选择此AI应用支持部署服务的类型，部署上线时只支持部署为此处选择的部署类型，例如此处只选择在线服务，那您导入后只能部署为在线服务。
“apis定义”	提供AI应用对外Restfull api数据定义，用于定义AI应用的输入、输出格式。apis定义填写规范请参见**模型配置文件编写说明**中的apis参数说明，示例代码请参见**apis参数代码示例。**

c. 确认信息填写无误，单击“立即创建”，完成AI应用创建。

在AI应用列表中，您可以查看刚创建的AI应用及其对应的版本。当AI应用状态变更为“正常”时，表示AI应用创建成功。在此页面，您还可以创建新版本、快速部署服务等操作。

后续操作

部署服务：在“AI应用列表”中，单击AI应用名称左侧的单选按钮，在列表页底部展开此AI应用下的“版本列表”。在对应版本所在行，单击“操作”列的部署按钮，可以将AI应用部署上线为创建AI应用时所选择的部署类型。

8.2.2.3 从容器镜像中选择元模型

针对ModelArts目前不支持的AI引擎，您可以通过自定义镜像的方式将编写的模型导入ModelArts。

约束与限制

- 关于自定义镜像规范和说明，请参见**模型镜像规范**。
- 针对您开发并训练完成的模型，需要提供对应的模型配置文件，此文件需遵守ModelArts的填写规范，详情请参见**模型配置文件编写说明**。编写完成后，需将此文件上传至OBS指定目录下。
- 使用容器化部署，导入的元模型有大小限制，详情请参见**导入AI应用对于镜像大小限制**。

前提条件

确保您使用的OBS目录与ModelArts在同一区域。

创建 AI 应用操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“AI应用管理 > AI应用”，进入AI应用列表页面。
2. 单击左上角的“创建”，进入“创建AI应用”页面。
3. 在“创建AI应用”页面，填写相关参数。
   a. 填写AI应用基本信息，详细参数说明请参见**表8-7**。

表 8-7 AI 应用基本信息参数说明

参数名称	说明
名称	AI应用名称。支持1～64位可见字符，名称可以包含字母、数字、中划线、下划线。
版本	设置所创建AI应用的版本。第一次导入时，默认为0.0.1。 **说明** AI应用创建完成后，可以通过**创建新版本**，导入不同的元模型进行调优。
描述	AI应用的简要描述。

   b. 填写元模型来源及其相关参数。当“元模型来源”选择“从容器镜像中选择”时，其相关的参数配置请参见**表8-8**。

表 8-8 元模型来源参数说明

参数	说明
“容器镜像所在的路径”	单击  从容器镜像中导入模型的镜像，其中，模型均为Image类型，且不再需要用配置文件中的“swr_location”来指定您的镜像位置。 制作自定义镜像的操作指导及规范要求，请参见**模型镜像规范**。 **说明** 您选择的模型镜像将共享给系统管理员，请确保具备共享该镜像的权限（不支持导入其他账户共享给您的镜像），部署上线时，ModelArts将使用该镜像部署成推理服务，请确保您的镜像能正常启动并提供推理接口。
“容器调用接口”	用于指定AI应用启动的协议和端口号。 **说明** ModelArts提供的请求协议和端口号的缺省值是HTTP和8080。用户需根据实际的自定义镜像进行配置。

参数	说明
“镜像复制”	镜像复制开关，选择是否将容器镜像中的模型镜像复制到ModelArts中。 ● 关闭时，表示不复制模型镜像，可极速创建AI应用，更改或删除SWR源目录中的镜像会影响服务部署。 ● 开启时，表示复制模型镜像，无法极速创建AI应用，SWR源目录中的镜像更改或删除不影响服务部署。 **说明** 若使用他人共享的镜像，需要开启镜像复制功能，否则会导致创建AI应用失败。
“健康检查”	用于指定AI应用的健康检查。仅当自定义镜像中配置了健康检查接口，才能配置“健康检查”，否则会导致AI应用创建失败。 ● **检查方式：**可以选择“HTTP请求检查”或者“执行命令检查”。 ● **健康检查URL**：“检查方式”选择“HTTP请求检查”时显示，填写健康检查的URL，默认值为“/health”。 ● **健康检查命令：**“检查方式”选择“执行命令检查”时显示，填写健康检查的命令。 ● **健康检查周期**：填写1-2147483647之前的整数，单位为秒。 ● **延迟时间（秒）：**实例启动后，延迟执行健康检查的时间。填写0-2147483647之间的整数，单位为秒，不能为空。 ● **健康检查最大失败次数**：填写1-2147483647之间的整数。在服务启动阶段，当健康检查请求连续失败达到所填次数后，服务会进入异常状态；在服务运行阶段，当健康检查请求连续失败达到所填次数后，服务会进入告警状态。 **说明** 当AI应用配置了健康检查，部署的服务在收到停止指令后，会延后3分钟才停止。
“AI应用说明”	为了帮助其他AI应用开发者更好的理解及使用您的AI应用，建议您提供AI应用的说明文档。单击“添加AI应用说明”，设置“文档名称”及其“URL”。AI应用说明支持增加3条。
“部署类型”	选择此AI应用支持部署服务的类型，部署上线时只支持部署为此处选择的部署类型，例如此处只选择在线服务，那您导入后只能部署为在线服务。
“启动命令”	指定模型的启动命令，您可以自定义该命令。

参数	说明
“apis定义”	提供AI应用对外Restfull api数据定义，用于定义AI应用的输入、输出格式。apis定义填写规范请参见**模型配置文件编写说明**中的apis参数说明，示例代码请参见**apis参数代码示例。**

c. 确认信息填写无误，单击“立即创建”，完成AI应用创建。

在AI应用列表中，您可以查看刚创建的AI应用及其对应的版本。当AI应用状态变更为“正常”时，表示AI应用创建成功。在此页面，您还可以进行创建新版本、快速部署服务等操作。

后续操作

部署服务：在“AI应用列表”中，单击AI应用名称左侧的单选按钮，在列表页底部展开此AI应用下的“版本列表”。在对应版本所在行，单击“操作”列的部署按钮，可以将AI应用部署上线为创建AI应用时所选择的部署类型。

8.2.3 查看 AI 应用列表

当AI应用创建成功后，您可在AI应用列表页查看所有创建的AI应用。AI应用列表页包含以下信息。

表 8-9 AI 应用列表

参数	说明
AI应用名称	AI应用的名称。
最新版本	AI应用的当前最新版本。
状态	AI应用当前状态。
部署类型	AI应用支持部署的服务类型。
版本数量	AI应用的版本数量。
请求模式	在线服务的请求模式。 • 同步请求：单次推理，可同步返回结果（约<60s）。例如: 图片、较小视频文件。 • 异步请求：单次推理，需要异步处理返回结果（约>60s）。例如: 实时视频推理、大视频文件。
创建时间	AI应用的创建时间。
描述	AI应用的描述。

参数	说明
操作	• 创建新版本：创建新的AI应用版本。参数配置除版本外，将默认选择上一个版本的配置信息，您可以对参数配置进行修改。 • 删除：删除对应的AI应用。 **说明** 如果AI应用的版本已经部署服务，需先删除关联的服务后再执行删除操作。AI应用删除后不可恢复，请谨慎操作。

单击AI应用名称左侧的单选按钮，展开列表页面底部的延展视图，可查看版本列表信息（未展开延展视图时，可单击页面底部浮层区域或右下角 ^ 展开）。

图 8-2 版本列表

版本列表中包含以下信息。

表 8-10 版本列表

参数	说明
版本	AI应用当前版本。
状态	AI应用当前状态。
部署类型	AI应用支持部署的服务类型。
AI应用大小	AI应用的大小。
模型来源	显示AI应用模型的来源。
创建时间	AI应用的创建时间。
描述	AI应用的描述。
操作	• 部署：将AI应用发布为在线服务、批量服务或边缘服务。 • 发布：将AI应用发布至AI Gallery。 • 删除：针对AI应用的某一版本进行删除。

8.2.4 查看 AI 应用详情

当AI应用创建成功后，您可以进入AI应用详情页查看AI应用的信息。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“AI应用”，进入“我的AI应用”列表页面。
2. 单击目标AI应用名称，进入AI应用详情页面。
   您可以查看AI应用的基本信息、模型精度，以及切换页签查看更多信息。

表 8-11 AI 应用基本信息

参数	说明
名称	AI应用的名称。
状态	AI应用当前状态。
版本	AI应用当前版本。
ID	AI应用的ID。
描述	单击编辑按钮，可以添加AI应用的描述。
部署类型	AI应用支持部署的服务类型。
元模型来源	显示元模型的来源，主要有从训练中选择、从对象存储服务（OBS）中选择、从容器镜像中选择。不同来源的元模型，AI应用显示的参数会不同。
训练作业名称	若元模型来源于训练作业，则显示关联的训练作业，单击训练作业名称可以直接跳转到训练作业详情页面。
训练作业版本	若元模型来源于训练作业且为旧版训练作业，显示训练作业版本。
元模型存储路径	若元模型来源于对象存储服务，显示元模型的存放路径。
容器镜像存储路径	若元模型来源于容器镜像，显示容器镜像存储路径。
AI引擎	若元模型来源于训练作业/对象存储服务，显示AI应用使用的AI引擎。
引擎包地址	若元模型来源于对象存储服务（AI引擎为Custom），显示引擎包地址。
运行环境	若元模型来源于训练作业/对象存储服务（AI引擎为预置引擎），显示元模型依赖的运行环境。
容器调用接口	若元模型来源于对象存储服务（AI引擎为Custom）/容器镜像，显示AI应用启动的协议和端口号。
推理代码	若元模型来源于训练作业且为旧版训练作业，则显示推理代码的存放路径。
镜像复制	若元模型来源于对象存储服务/容器镜像，显示镜像复制功能状态。
大小	AI应用的大小。

参数	说明
健康检查	若元模型来源于对象存储服务/容器镜像，显示健康检查状态。当健康检查为开启时，显示以下字段：检查方式、健康检查URL、健康检查周期、延迟时间、健康检查最大失败次数。
AI应用说明	显示创建AI应用时添加的AI应用说明文档信息。
系统运行架构	显示系统运行架构。
推理加速卡类型	显示推理加速卡类型。

表 8-12 AI 应用页签详情

参数	说明
模型精度	显示该AI应用的模型召回率、精准率、准确率和F1值。
参数配置	可以查看AI应用的apis定义详情，以及AI应用的入参和出参。
运行时依赖	查看模型对环境的依赖。当构建任务失败后可以编辑运行时依赖，保存修改后将触发镜像重新构建。
事件	展示AI应用创建过程中的关键操作进展。 事件保存周期为3个月，3个月后自动清理数据。 查看AI应用的事件类型和事件信息，请参见**查看AI应用的事件**
使用约束	根据创建AI应用时的设置，显示部署服务的使用约束，如请求模式、启动命令、模型加密等。对于异步请求模式的AI应用，可显示输入模式、输出模式、服务启动参数和作业配置参数等参数。
关联服务	展示使用该AI应用部署的服务列表，单击服务名称可以直接跳转到服务详情页面。

8.2.5 管理 AI 应用版本

为方便溯源和AI应用反复调优，在ModelArts中提供了AI应用版本管理的功能，您可以基于版本对AI应用进行管理。

前提条件

已在ModelArts中创建AI应用。

创建新版本

在“AI应用”页面，单击操作列的“创建新版本”进入“创建新版本”页面，参数配置除版本外，将默认选择上一个版本的配置信息，您可以对参数配置进行修改，参数说明请参见**创建AI应用**。单击“立即创建”，完成新版本的创建操作。

删除版本

在“AI应用”页面，单击AI应用名称左侧的单选按钮，在列表页面底部展开“版本列表”，在版本列表中，单击“操作”列的“删除”，即可删除对应的版本。

说明

如果AI应用的版本已经部署服务，需先删除关联的服务后再执行删除操作。版本删除后不可恢复，请谨慎操作。

删除 AI 应用

在“AI应用”页面，单击AI应用“操作”列的“删除”，即可删除对应的AI应用。

说明

如果AI应用的版本已经部署服务，需先删除关联的服务后再执行删除操作。AI应用删除后不可恢复，请谨慎操作。

8.2.6 查看 AI 应用的事件

创建AI应用的（从用户可看见创建AI应用任务开始）过程中，每一个关键事件点在系统后台均有记录，用户可随时在对应AI应用的详情页面进行查看。

方便用户更清楚的了解创建AI应用过程，遇到任务异常时，更加准确的排查定位问题。可查看的事件点包括：

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
正常	开始导入模型。 Start model import.	-
异常	构建镜像失败。 Failed to build the image.	构建镜像失败原因较多，需根据具体的报错定位和处理问题。FAQ
异常	自定义镜像不支持指定依赖。 Customize model does not support dependencies.	自定义镜像导入不支持配置运行时依赖，在构建镜像的dockerfile文件中安装pip依赖包。FAQ
异常	非自定义镜像不支持指定swr_location字段。 Non-custom type models should not contain swr_location.	请删除模型配置文件config.json中的swr_location字段后重试。
异常	自定义镜像健康检查接口必须是xxx。 The health check url of custom image model must be %s.	请修改自定义镜像健康检查接口后重试。

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
正常	当前镜像构建任务状态为xxx。 The status of the image building task is %s.	-
异常	镜像xxx不存在名为xxx的标签。 Image %s does not have the %s tag.	请联系技术支持。
异常	模型配置文件包含非法参数值：xxx。 Invalid %s in config.json.	请删除模型配置文件中的非法参数后重试。
异常	获取镜像xxx的标签列表失败。 Failed to obtain the tag list of image %s.	请联系技术支持。
异常	xxx大于xxxG，无法导入。 %s [%s] is larger than %dG and cannot be imported.	模型或镜像大小超过限制，请精简模型或镜像后，重新导入。**FAQ**
异常	用户xxx没有OBS的obs:object:PutObjectAcl权限。 User %s does not have obs:object:PutObjectAcl permission	子用户没有OBS的obs:object:PutObjectAcl权限，为子用户添加委托权限。**FAQ**
异常	镜像构建任务超时。限制超时时间为xxx分钟。 Image building task timeout. The %s-minute limit is over.	imagePacker构建镜像有超时时间限制，请精简代码，提高编译效率。**FAQ**
正常	模型描述已更新。 Model description updated.	-
正常	模型运行时依赖未更新。 Model running dependencies not updated.	-
正常	模型运行时依赖已更新。正在重新构建镜像 Model running dependencies updated. Rebuild the image.	-
异常	触发SWR限流，请稍后重试。 The throttling threshold of swr has been reached.	触发SWR限流，请稍后重试。
正常	系统升级中，请稍后重试。 System is upgrading, please try again later.	-

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
异常	获取源镜像失败。认证错误，token已失效。 Failed to access source image. Authenticate Error, token expired.	请联系技术支持。
异常	获取源镜像失败。检查该镜像是否存在。 Failed to access source image. Check whether the image exists.	请联系技术支持。
正常	源镜像大小计算完成。 Source image size calculated successfully.	-
正常	源镜像共享成功。 Source image shared successfully.	-
异常	构建镜像失败，因为触发了限流。请稍后重试。 Failed to build the image due to the threshold has been reached. Please try again later.	触发了限流，请稍后重试。
异常	发送构建镜像请求失败。 Failed to send image building request.	请联系技术支持。
异常	共享源镜像失败。请检查镜像是否存在或者是否有权限共享该镜像。 Failed to share source image. Check whether the image exists or whether you have the share permission on the image.	请检查镜像是否存在或者是否有权限共享该镜像。
正常	模型导入成功。 Model imported successfully.	-
正常	模型文件导入成功。 Model file imported successfully.	-
正常	模型大小计算完成。 Model size calculated successfully.	-
异常	模型导入失败。 Failed to import the model.	模型导入失败情况较多，请参考**FAQ**定位和处理。

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
异常	复制模型文件失败，请检查OBS权限是否正常。 Failed to copy model file due to obs exception. Please Check your obs access right.	请检查OBS权限是否正常。FAQ
异常	镜像构建任务调度失败。 Image building task scheduling failed.	请联系技术支持。
异常	启动镜像构建任务失败。 Failed to start the image building task.	请联系技术支持。
异常	罗马镜像构建完成，无法分享给资源租户。 The ROMA image is successfully built but cannot be shared to resource tenants.	请联系技术支持。
正常	镜像构建完成。 Image built successfully.	-
正常	启动镜像构建任务。 Start the image building task.	-
正常	启动环境镜像构建任务。 Start the env image building task.	-
正常	收到构建模型环境镜像请求。 Received another env image building request of the model.	-
正常	收到构建模型镜像请求。 Received another image building request of the model.	-
正常	使用现有环境镜像。 Use cached env image.	-
异常	构建镜像失败。详细信息请查看构建日志。 Failed to build the image. For details, view the building log.	查看构建日志定位和处理问题。FAQ
异常	因系统内部原因构建镜像失败。请联系技术支持。 Failed to build the image due to system errors. Contact the administrator.	请联系技术支持。

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
异常	模型文件xxx大于5G，无法导入。 Model file %s is larger than 5G and cannot be imported.	模型文件xxx大于5G，请精简模型文件后重试，或者使用动态加载功能进行导入。FAQ
异常	因系统内部原因创建OBS桶失败，请联系技术支持。 Failed to create bucket due to system errors. Contact the administrator.	请联系技术支持。
异常	模型大小计算失败。子路径xxx在路径xxx下不存在。 Model size calculated failed.Can not find %s child directory in current model directory %s.	修改子路径为正确的路径后重试，或者联系技术支持。
异常	模型大小计算失败。xxx类型模型不存在路径xxx下。 Model size calculated failed.Can not find %s file in current model directory %s.	检查xxx类型模型的存储位置，修改为正确的路径后重试，或者联系技术支持。
提示	模型大小计算失败。多于一个xxx模型文件在路径xxx下。 Model size calculated failed.Find more than one %s file in current model directory %s.	-

创建AI应用的过程中，关键事件支持手动/自动刷新。

查看操作

1. 在ModelArts管理控制台的左侧导航栏中选择“AI应用”，在AI应用列表中，您可以单击AI应用名称，进入AI应用详情页面。
2. 在AI应用详情页面，切换到“事件”页签，查看事件信息。

8.3 部署 AI 应用（部署上线）

8.3.1 部署 AI 应用（在线服务）

8.3.1.1 部署为在线服务

AI应用准备完成后，您可以将AI应用部署为在线服务，对在线服务进行预测和调用。

约束与限制

单个用户最多可创建20个在线服务。

前提条件

- 数据已完成准备：已在ModelArts中创建状态“正常”可用的AI应用。

注意事项

针对使用公共资源池部署的在线服务，服务处于“异常”或“停止”等状态时，也占用配额资源。如果发现配额不足，无法部署更多服务时，可先删除部分异常服务释放资源。

配额计算：

- 使用专属池部署在线服务不会再扣减配额，仅在创建/变更/删除专属池时增加或减少配额。
- 使用共享池部署在线服务，在做新建/变更实例数/删除操作时会增加或减少配额。

计量计算：

- 使用专属池部署在线服务不会被计量，只计量它所属的专属池的数据。
- 使用共享池部署在线服务会被计量其使用的规格的数据。

操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“部署上线 > 在线服务”，默认进入“在线服务”列表。
2. 在“在线服务”列表中，单击左上角“部署”，进入“部署”页面。
3. 在“部署”页面，填写在线服务相关参数。
   a. 填写基本信息，详细参数说明请参见表8-13。

   表 8-13 基本信息参数说明

参数名称	说明
“名称”	在线服务的名称，请按照界面提示规则填写。
“是否自动停止”	启用该参数并设置时间后，服务将在指定时间后自动停止。默认开启自动停止功能，且默认值为“1小时后”。 目前支持设置为“1小时后”、“2小时后”、“4小时后”、“6小时后”、“自定义”。如果选择“自定义”的模式，可在右侧输入框中输入1~24范围内的任意整数。
“描述”	在线服务的简要说明。

   b. 填写资源池和AI应用配置等关键信息，详情请参见表8-14。

表 8-14 参数说明

参数名称	子参数	说明
“资源池”	“公共资源池”	公共资源池有CPU或GPU两种规格。
	“专属资源池”	在专属资源池规格中选择对应的规格进行使用。暂不支持选择创建了逻辑子池的物理池。 **说明** • 旧版“专属资源池”将逐渐迁移至新版“专属资源池”。 • 新用户和旧版“专属资源池”迁移完成的老用户在ModelArts管理控制台只能看到新版的“专属资源池”。 • 旧版“专属资源池”未迁移的老用户，可以看到两个专属资源池，其中“专属资源池 New”为新版的专属资源池。 了解新版“专属资源池”请参见**ModelArts资源池管理功能全面升级**
“选择AI应用及配置”	“AI应用来源”	根据您的实际情况选择“我的AI应用”。
	“选择AI应用及版本”	选择状态“正常”的AI应用及版本。
	“分流”	设置当前实例节点的流量占比，服务调用请求根据该比例分配到当前版本上。 如您仅部署一个版本的AI应用，请设置为100%。如您添加多个版本进行灰度发布，多个版本分流之和设置为100%。
	“计算节点规格”	请根据界面显示的列表，选择可用的规格，置灰的规格表示当前环境无法使用。 如果公共资源池下规格为空数据，表示当前环境无公共资源。建议使用专属资源池，或者联系系统管理员创建公共资源池。 **说明** 使用所选规格部署服务时，会产生必要的系统消耗，因此服务实际占用的资源会略大于该规格。
	“计算节点个数”	设置当前版本AI应用的实例个数。如果节点个数设置为1，表示后台的计算模式是单机模式；如果节点个数设置大于1，表示后台的计算模式为分布式的。请根据实际编码情况选择计算模式。
	“环境变量”	设置环境变量，注入环境变量到容器实例。为确保您的数据安全，在环境变量中，请勿输入敏感信息。

参数名称	子参数	说明
	“部署超时时间”	用于设置单个模型实例的超时时间，包括部署和启动时间。默认值为20分钟，输入值必须在3到120之间。
	“添加AI应用版本进行灰度发布”	当选择的AI应用有多个版本时，您可以添加多个AI应用版本，并配置其分流占比，完成多版本和灵活流量策略的灰度发布，实现AI应用版本的平滑过渡升级。 **说明** 当前免费计算规格不支持多版本灰度发布。
	“存储挂载”	资源池为专属资源池时显示该参数。在服务运行时将存储卷以本地目录的方式挂载到计算节点（计算实例），模型或输入数据较大时建议使用。存储卷类型目前仅支持OBS并行文件系统。 ● 源地址：选择并行文件的存储路径。不支持选择跨区域（Region）的OBS并行文件系统。 ● 挂载路径：指定容器内部的挂载路径，如“/obs-mount/”。 – 请选择全新目录，选择存量目录会覆盖存量文件，OBS挂载仅开放对挂载目录文件新增、查看、修改功能不支持删除挂载目录文件对象，若需要删除文件请到OBS并行文件系统中手动删除。 – 建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。 – 挂载路径必须以/开头，仅允许输入英文、数字和特殊字符_-且不超过1024个字符。 **说明** 使用专属资源池部署服务才允许使用存储挂载的能力。
“服务流量限制”	-	服务流量限制是指每秒内一个服务能够被访问的次数上限。您可以根据实际需求设置每秒流量限制。
“升级为WebSocket”	-	设置在线服务是否部署为WebSocket服务。了解在线服务支持WebSocket，请参考**WebSocket在线服务全流程开发**。 **说明** ● 要求AI应用的元模型来源为从容器镜像中选择，并且镜像支持WebSocket。 ● 设置“升级为WebSocket”后，不支持设置“服务流量限制”。 ● “升级为WebSocket”参数配置，不支持修改。

4. 确认填写信息无误后，根据界面提示完成在线服务的部署。部署服务一般需要运行一段时间，根据您选择的数据量和资源不同，部署时间将耗时几分钟到几十分钟不等。

   说明

   在线服务部署完成后，将立即启动。

   您可以前往在线服务列表，查看在线服务的基本情况。在线服务列表中，刚部署的服务“状态”为“部署中”，当在线服务的“状态”变为“运行中”时，表示服务部署完成。

8.3.1.2 查看服务详情

当AI应用部署为在线服务成功后，您可以进入“在线服务”页面，来查看服务详情。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线>在线服务”，进入“在线服务”管理页面。
2. 单击目标服务名称，进入服务详情页面。

   您可以查看服务的“名称”、“状态”等信息，详情说明请参见**表8-15**。

   表 8-15 在线服务配置

参数	说明
名称	在线服务名称。
状态	在线服务当前状态。
来源	在线服务的来源。
服务ID	在线服务的ID。
描述	您可以单击编辑按钮，添加服务描述。
资源池	当前服务使用的资源池规格。若使用公共资源池部署，则不显示该参数。
个性化配置	您可以为在线服务的不同版本设定不同配置条件，并支持携带自定义运行参数，丰富版本分流策略或同一版本内的不同运行配置。您可以打开个性化配置按钮，单击“查看配置”**修改服务个性化配置**。
服务流量限制	服务流量限制是指每秒内一个服务能够被访问的次数上限。
升级为WebSocket	是否升级为WebSocket服务。

3. 您可以进入在线服务的详情页面，通过切换页签查看更多详细信息，详情说明请参见**表8-16**。

表 8-16 在线服务详情

参数	说明
调用指南	展示API接口公网地址、AI应用信息、输入参数、输出参数。您可以通过复制API接口公网地址，调用服务。
预测	对在线服务进行预测。具体操作请参见**测试服务**。
配置更新记录	展示“当前配置”详情和“历史更新记录”。 • “当前配置”：展示AI应用名称、版本、状态、计算节点规格、分流、计算节点个数、部署超时时间、环境变量、存储挂载等信息。专属资源池部署的服务，同时展示资源池信息。 • “历史更新记录”：展示历史AI应用相关信息。
监控信息	展示当前服务的“资源统计信息”和“AI应用调用次数统计”。 • “资源统计信息”：包括CPU、内存、GPU、NPU的可用和已用信息。 • “AI应用调用次数统计”：当前AI应用的调用次数，从AI应用状态为“已就绪”后开始统计。（websocket服务不显示）
事件	展示当前服务使用过程中的关键操作，比如服务部署进度、部署异常的详细原因、服务被启动、停止、更新的时间点等。 事件保存周期为1个月，1个月后自动清理数据。 查看服务的事件类型和事件信息，请参见**查看服务的事件**
日志	展示当前服务下每个AI应用的日志信息。包含最近5分钟、最近30分钟、最近1小时和自定义时间段。 自定义时间段您可以选择开始时间和结束时间。 日志搜索规则说明： • 不支持带有分词符的字符串搜索（当前默认分词符有 ,'";=()[]{}@&<>/:\n\t\r）。 • 支持关键词精确搜索。关键词指相邻两个分词符之间的单词。 • 支持关键词模糊匹配搜索，例如输入“error”或“er?or”或“rro*”或“er*r”。 • 支持短语精确搜索。例如输入“Start to refresh”。 • 启用运行日志输出前，支持关键词的“与”、“或”组合搜索。格式为“query logs&&erro*”或“query logs||erro*”。启用运行日志输出后，支持关键词的“与”、“或”组合搜索。格式为“query logs AND erro*”或“query logs OR erro*”。

修改服务个性化配置

服务个性化配置规则由配置条件、访问版本、自定义运行参数（包括配置项名称和配置项值）组成。

您可以为在线服务的不同版本设定不同配置条件，并支持携带自定义运行参数。

个性化配置规则的优先级与顺序相对应，从高到低设置。您可以通过拖动个性化配置规则的顺序更换优先级。

当匹配了某一规则后就不再继续下一规则的判断，最多允许配置10个条件。

表 8-17 个性化配置参数

参数	是否必选	说明
配置条件	必选	SPEL（Spring Expression Language）规则的表达式，当前仅支持字符型的“相等”、“matches”和hashCode函数计算。
访问版本	必选	服务个性化配置规则对应的访问版本。当匹配到规则时，请求该版本的在线服务。
配置项名称	可选	自定义运行参数的Key值，不超过128个字符。 当需要通过Header（http消息头）携带自定义运行参数至在线服务时，可以配置。
配置项值	可选	自定义运行参数的Value值，不超过256个字符。 当需要通过Header（http消息头）携带自定义运行参数至在线服务时，可以配置。

可以设置以下三种场景：

- 如果在线服务部署多个版本用于灰度发布，可以使用个性化配置实现按用户分流。

  表 8-18 按内置变量配置条件

内置变量	说明
DOMAIN_NAME	调用预测请求的账号名。
DOMAIN_ID	调用预测请求的账号ID。
PROJECT_NAME	调用预测请求的项目名。
PROJECT_ID	调用预测请求的项目ID。
USER_NAME	调用预测请求的用户名。
USER_ID	调用预测请求的用户ID。

“#”表示引用变量，匹配的字符串需要用单引号。

```
#{内置变量} == '字符串'
#{内置变量} matches '正则表达式'
```

- 示例一：

  当调用预测请求的账号名为“zhangsan”时，匹配至指定版本。

  ```
  #DOMAIN_NAME == 'zhangsan'
  ```

- 示例二：

  当调用预测请求的账号名以“op”开头时，匹配至指定版本。

  ```
  #DOMAIN_NAME matches 'op.*'
  ```

表 8-19 常用的正则匹配表达式

字符	描述
“.”	匹配除“\n”之外的任何单个字符串。需匹配包括“\n”在内的任何字符，请使用“(.|\n)”的模式。
“*”	匹配前面的子表达式零次或多次。例如，“zo*”能匹配“z”以及“zoo”。
“+”	匹配前面的子表达式一次或多次。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。
“?”	匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“does”或“does”中的“do”。
“^”	匹配输入字符串的开始位置。
“$”	匹配输入字符串的结束位置。
“{n}”	*n*是一个非负整数。匹配确定的*n*次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个“o”。
“x|y”	匹配x或y。例如，“z|food”能匹配“z”或“food”。“(z|f)ood”则匹配“zood”或“food”。
“[xyz]”	字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”。

图 8-3 按用户分流

- 如果在线服务部署多个版本用于灰度发布，可以使用个性化配置实现通过Header来访问不同版本。

  您需要通过"#HEADER_"开头说明引用header作为条件

```
#HEADER_{key} == '{value}'
#HEADER_{key} matches '{value}'
```

- 示例一：

  当预测的http请求的header中存在version，且值为0.0.1则符合条件。不存在此header或者值不为0.0.1都不符合条件。

  ```
  #HEADER_version == '0.0.1'
  ```

- 示例二：

  当预测的http请求的header中存在testheader且值符合正以mock开头时，可匹配到这条规则。

  ```
  #HEADER_testheader matches 'mock.*'
  ```

- 示例三：

  当预测的http请求的header中存在uid且其哈希值符合指定的分桶算法时，可匹配到这条规则。

  ```
  #HEADER_uid.hashCode() % 100 < 10
  ```

图 8-4 通过 Header 访问不同版本

- 如果在线服务部署的版本支持使用不同的运行配置，您可以通过“配置项名称”和“配置项值”携带自定义运行参数至在线服务，实现不同用户使用不同运行配置。

  示例：

  用户zhangsan访问时，AI应用使用配置A；用户lisi访问时，AI应用使用配置B。当匹配到运行配置条件时，ModelArts会在请求里增加一个Header，传入自定义运行参数，其中Key是“配置项名称”，Value是“配置项值”。

图 8-5 个性化配置规则支持传入自定义运行参数。

8.3.1.3 测试服务

AI应用部署为在线服务成功后，您可以在“预测”页签进行代码调试或添加文件测试。根据AI应用定义的输入请求不同（JSON文本或文件），测试服务包括如下两种方式：

- **JSON文本预测**：如当前部署服务的AI应用，其输入类型指定的为JSON文本类，即不含有文件类型的输入，可以在“预测”页签输入JSON代码进行服务预测。
- **文件预测**：如当前部署服务的AI应用，其输入类型指定为文件类，可包含图片、音频或视频等场景，可以在“预测”页签添加图片进行服务预测。

说明

- 如果您的输入类型为图片，请注意测试服务单张图片输入应小于8MB。
- JSON文本预测，请求体的大小不超过8MB。
- 因APIG（API网关）的限制，单次预测的时间不能超过40S。
- 图片支持以下类型：“png”、“psd”、“jpg”、“jpeg”、“bmp”、“gif”、“webp”、“psd”、“svg”、“tiff”。
- 该功能为调测使用，实际生产建议使用API调用。根据鉴权方式的不同，可以根据实际情况选择**访问在线服务（Token认证）**。

了解服务的输入参数

针对您部署上线的服务，您可以在服务详情页面的“调用指南”中，了解本服务的输入参数，即上文提到的输入请求类型。

调用指南中的输入参数取决于您选择的AI应用来源：

- 如果您的元模型来源于自动学习或预置算法，其输入输出参数由ModelArts官方定义，请直接参考“调用指南”中的说明，并在预测页签中输入对应的JSON文本或文件进行服务测试。
- 如果您的元模型是自定义的，即推理代码和配置文件是自行编写的（**配置文件编写说明**），“调用指南”只是将您编写的配置文件进行了可视化展示。调用指南的输入参数与配置文件对应关系如下所示。

图 8-6 配置文件与调用指南的对应关系

JSON 文本预测

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线>在线服务”，进入“在线服务”管理页面。
2. 单击目标服务名称，进入服务详情页面。在“预测”页签的预测代码下，输入预测代码，然后单击“预测”即可进行服务的预测。

文件预测

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线>在线服务”，进入“在线服务”管理页面。

2. 单击目标服务名称，进入服务详情页面。在“预测”页签，单击“上传”，然后选择测试文件。文件上传成功后，单击“预测”即可进行服务的预测。

8.3.1.4 访问在线服务

8.3.1.4.1 访问在线服务简介

在线服务的状态处于“运行中”，则表示在线服务已部署成功，部署成功的在线服务，将为用户提供一个可调用的API，此API为标准Restful API。在集成至生产环境之前，需要对此API进行调测。

在线服务的API默认为HTTPS访问，同时还支持WebSocket访问。在线服务部署时如果选择了“升级为WebSocket”，服务部署完成后，API接口公网地址将是一个WebSocket协议地址。请参见**Websocket访问在线服务**。

当前ModelArts支持访问在线服务的认证方式有以下方式（均以HTTPS请求为例）：

- **Token认证**

ModelArts支持通过以下几种方式调用API访问在线服务：

- **访问在线服务（公网访问通道）**
- **访问在线服务（VPC高速访问通道）**

调用API访问在线服务时，对预测请求体大小和预测时间有限制：

- 请求体的大小不超过12MB，超过后请求会被拦截。
- 因APIG（API网关）限制，平台每次请求预测的时间不超过40秒。

8.3.1.4.2 认证方式

访问在线服务（Token 认证）

若在线服务的状态处于“运行中”，则表示在线服务已部署成功，部署成功的在线服务，将为用户提供一个可调用的API，此API为标准Restful API。在集成至生产环境之前，需要对此API进行调测，您可以使用以下方式向在线服务发起预测请求：

- **方式一：使用图形界面的软件进行预测（以Postman为例）**。Windows系统建议使用Postman。
- **方式二：使用curl命令发送预测请求**。Linux系统建议使用curl命令。
- **方式三：使用Python语言发送预测请求**。

前提条件

已经获取**用户Token**、**预测文件的本地路径**、**在线服务的调用地址**和**在线服务的输入参数**信息。

- 预测文件的本地路径既可使用绝对路径（如Windows格式"D:/test.png"，Linux格式"/opt/data/test.png"），也可以使用相对路径（如"./test.png"）。
- 在线服务的调用地址和输入参数信息，可以在控制台的“在线服务详情 > 调用指南”页面获取。

“API接口公网地址”即在线服务的调用地址。当模型配置文件中apis定义了路径，调用地址后需拼接自定义路径。如：“{在线服务的调用地址}/predictions/poetry”。

方式一：使用图形界面的软件进行预测（以 Postman 为例）

1. 下载Postman软件并安装，您也可以直接在Chrome浏览器添加Postman扩展程序（也可使用其他支持发送post请求的软件）。Postman推荐使用7.24.0版本。
2. 打开Postman。
3. 在Postman界面填写参数，以图像分类举例说明。
   - 选择POST任务，将在线服务的调用地址复制到POST后面的方框。Headers页签的Key值填写为“X-Auth-Token”，Value值为用户Token。
   - 在Body页签，根据AI应用的输入参数不同，可分为2种类型：文件输入、文本输入。
     - **文件输入**

       选择“form-data”。在“KEY”值填写AI应用的入参，和在线服务的输入参数对应，比如本例中预测图片的参数为“images”。然后在“VALUE”值，选择文件，上传一张待预测图片（当前仅支持单张图片预测）。
     - **文本输入**

       选择“raw”，选择JSON(application/json)类型，在下方文本框中填写请求体，请求体样例如下：

       ```
       {
         "meta": {
           "uuid": "10eb0091-887f-4839-9929-cbc884f1e20e"
         },
         "data": {
           "req_data": [
             {
               "sepal_length": 3,
               "sepal_width": 1,
               "petal_length": 2.2,
               "petal_width": 4
             }
           ]
         }
       }
       ```

       其中，“meta”中可携带“uuid”，调用时传入一个“uuid”，返回预测结果时回传此“uuid”用于跟踪请求，如无此需要可不填写meta。“data”包含了一个“req_data”的数组，可传入单条或多条请求数据，其中每个数据的参数由AI应用决定，比如本例中的“sepal_length”、“sepal_width”等。
4. 参数填写完成，单击“send”发送请求，结果会在“Response”下的对话框里显示。
   - 文件输入形式的预测返回结果的字段值根据不同AI应用可能有所不同。
   - 文本输入形式的预测，请求体包含“meta”及“data”。如输入请求中包含“uuid”，则输出结果中回传此“uuid”。如未输入，则为空。“data”包含了一个“resp_data”的数组，返回单条或多条输入数据的预测结果，其中每个结果的参数由AI应用决定，比如本例中的“sepal_length”、“predictresult”等。

方式二：使用 curl 命令发送预测请求

使用curl命令发送预测请求的命令格式也分为文件输入、文本输入两类。

- 文件输入
  curl -kv -F '**images**=@**图片路径**' -H '**X-Auth-Token**:**Token值**' -X POST **在线服务地址**
  - “-k”是指允许不使用证书到SSL站点。
  - “-F”是指上传数据的是文件，本例中参数名为“images”，这个名字可以根据具体情况变化，@后面是图片的存储路径。
  - “-H”是post命令的headers，Headers的Key值为“X-Auth-Token”，这个名字为固定的， Token值是获取的用户Token。
  - “POST”后面跟随的是在线服务的调用地址。

  curl命令文件输入样例：

  ```
  curl -kv -F 'images=@/home/data/test.png' -H 'X-Auth-Token:MIISkAY***80T9wHQ==' -X POST https://modelarts-infers-1.xxx/v1/infers/eb3e0c54-3dfa-4750-af0c-95c45e5d3e83
  ```

- 文本输入
  ```
  curl -kv -d '{"data":{"req_data":[{"sepal_length":3,"sepal_width":1,"petal_length":2.2,"petal_width":4}]}}' -H 'X-Auth-Token:MIISkAY***80T9wHQ==' -H 'Content-type: application/json' -X POST https://modelarts-infers-1.xxx/v1/infers/eb3e0c54-3dfa-4750-af0c-95c45e5d3e83
  ```

  “-d”是Body体的文本内容。

方式三：使用 Python 语言发送预测请求

1. 下载Python SDK并在开发工具中完成SDK配置。
2. 创建请求体，进行预测请求。
   - **输入为文件格式**

     ```
     # coding=utf-8

     import requests

     if __name__ == '__main__':
         # Config url, token and file path.
         url = "在线服务的调用地址"
         token = "用户Token"
         file_path = "预测文件的本地路径"

         # Send request.
         headers = {
             'X-Auth-Token': token
         }
         files = {
             'images': open(file_path, 'rb')
         }
         resp = requests.post(url, headers=headers, files=files)

         # Print result.
         print(resp.status_code)
         print(resp.text)
     ```

     “files”中的参数名由在线服务的输入参数决定，需要和“类型”为“file”的输入参数“名称”保持一致。
   - **输入为文本格式（json类型）**

     读取本地预测文件并进行base64编码的请求体示例如下：

     ```
     # coding=utf-8

     import base64
     import requests
     ```

```
if __name__ == '__main__':
    # Config url, token and file path
    url = "在线服务的调用地址"
    token = "用户Token"
    file_path = "预测文件的本地路径"
    with open(file_path, "rb") as file:
        base64_data = base64.b64encode(file.read()).decode("utf-8")

    # Set body,then send request
    headers = {
        'Content-Type': 'application/json',
        'X-Auth-Token': token
    }
    body = {
        'image': base64_data
    }
    resp = requests.post(url, headers=headers, json=body)

    # Print result
    print(resp.status_code)
    print(resp.text)
```

“body”中的参数名由在线服务的输入参数决定，需要和“类型”为“string”的输入参数“名称”保持一致。“body”中的base64_data值为string类型。

8.3.1.4.3 访问方式

访问在线服务（公网访问通道）

背景描述

ModelArts推理默认使用公网访问在线服务。在线服务部署成功后，将为用户提供一个可调用的API，此API为标准Restful API。您可以在服务详情页面，调用指南页签中查看API接口公网地址。

图 8-7 API 接口公网地址

访问在线服务

公网访问在线服务有以下认证方式，API调用请参见认证详情：

- **访问在线服务（Token认证）**

访问在线服务（VPC 访问通道）

背景说明

如果您希望在自己账号的VPC内部节点访问ModelArts推理的在线服务，可以使用VPC访问通道的功能，用户通过在自己账号的指定VPC下创建终端节点，连接到ModelArts的终端节点服务，即可在自己的VPC节点中访问在线服务。

操作步骤

VPC访问通道访问在线服务操作步骤如下：

1. **获取ModelArts终端节点服务地址**
2. **购买连接ModelArts终端节点**
3. **在线服务设置VPC访问通道**
4. **创建DNS内网域名**
5. **VPC访问在线服务**

步骤1 获取ModelArts终端节点服务地址

1. 登录ModelArts管理控制台，选择“部署上线 > 在线服务”。
2. 单击“VPC访问通道”，在弹出框中查看终端节点服务地址。

图 8-8 查看终端节点服务地址

步骤2 购买连接ModelArts终端节点

1. 登录虚拟私有云（VPC）管理控制台，单击左侧导航栏中的“VPC 终端节点>终端节点”，进入“终端节点”页面。
2. 单击右上角的“购买终端节点”，进入购买页面。
   - 区域：终端节点所在区域。
     不同区域的资源之间内网不互通，请确保与ModelArts所在区域保持一致。
   - 服务类别：请选择“按名称查找服务”。
   - 服务名称：填入1中获取的“终端节点服务地址”。单击右侧验证按钮，系统将为您自动填入虚拟私有云、子网和节点IP。
   - 创建内网域名：保持默认值。
3. 确认规格无误后，单击“立即购买”后提交任务，界面自动跳转至终端节点列表页面。

步骤3 在线服务设置VPC访问通道

1. 登录ModelArts管理控制台，左侧导航栏选择“部署上线 > 在线服务”。
2. 单击“VPC访问通道”，在弹出框中选择2中使用的VPC后，终端节点之后的“终端节点ID”和“终端节点IP”自动关联显示。

图 8-9 选择 VPC，自动关联“终端节点 ID”和“终端节点 IP”

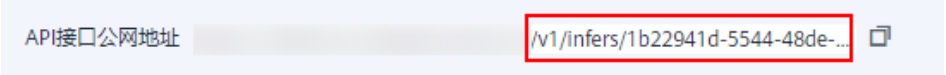
步骤4 创建DNS内网域名

1. 登录云解析服务DNS管理控制台，左侧导航栏选择“内网域名”。
2. 单击“创建内网域名”，打开创建内网域名弹出框。填写以下参数配置：
   - 域名：遵循命名规范“infer-modelarts-<regionId>.xxx.com”，当前区域ID去掉中划线为regionId的值。
   - VPC：选择内网域名关联的VPC。
3. 单击“确定”，完成DNS内网域名的创建。

步骤5 VPC访问在线服务

1. 通过VPC访问通道访问在线服务，API如下：
   ```
   https://{DNS内网域名}/{URL}
   ```
   - DNS内网域名：设置的内网域名。您还可以通过在线服务列表页，单击“VPC访问通道”，打开弹出框，查看“访问域名”。
   - URL：在线服务的URL为服务详情页，调用指南页签中获取的“API接口公网地址”截取域名之后的地址部分。

   图 8-10 获取 URL

2. 使用图形界面的软件、curl命令、Python语言等多种方式访问在线服务。可参考**访问在线服务（Token认证）**。

----结束

访问在线服务（VPC 高速访问通道）

背景说明

访问在线服务的实际业务中，用户可能会存在如下需求：

- 高吞吐量、低时延
- TCP或者RPC请求

因此，ModelArts提供了VPC直连的高速访问通道功能以满足用户的需求。

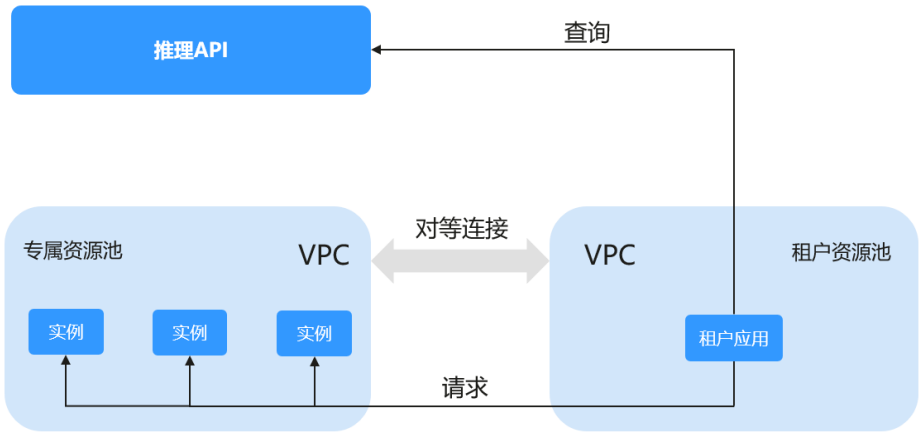
使用VPC直连的高速访问通道，用户的业务请求不需要经过推理平台，而是直接经VPC对等连接发送到实例处理，访问速度更快。

说明

由于请求不经过推理平台，所以会丢失以下功能：

- 认证鉴权
- 流量按配置分发
- 负载均衡
- 告警、监控和统计

图 8-11 VPC 直连的高速访问通道示意图



准备工作

使用专属资源池部署在线服务，服务状态为“运行中”。

须知

- 需使用新版专属资源池部署服务，详情请参见**ModelArts资源池管理功能全面升级**。
- 只有专属资源池部署的服务才支持VPC直连的高速访问通道。
- VPC直连的高速访问通道，目前只支持访问在线服务。
- 因流量限控，获取在线服务的IP和端口号次数有限制，每个主账号租户调用次数不超过2000次/分钟，每个子账号租户不超过20次/分钟。
- 目前仅支持自定义镜像导入模型，部署的服务支持高速访问通道。

操作步骤

使用VPC直连的高速访问通道访问在线服务，基本操作步骤如下：

1. **将专属资源池的网络打通VPC**
2. **VPC下创建弹性云服务器**
3. **获取在线服务的IP和端口号**
4. **通过IP和端口号直连应用**

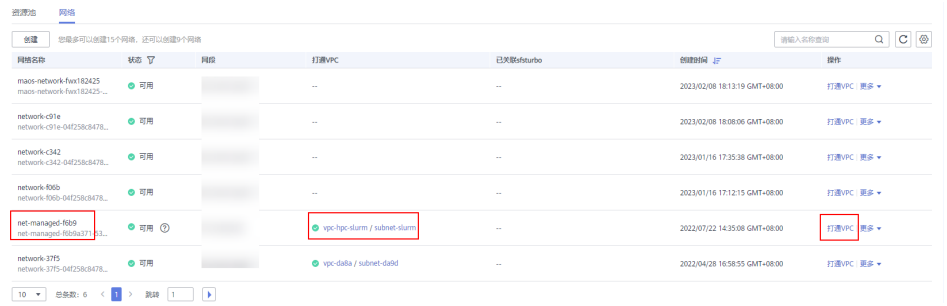
步骤1 将专属资源池的网络打通VPC

登录ModelArts控制台，进入“专属资源池 > 弹性集群”找到服务部署使用的专属资源池，单击“名称/ID”，进入资源池详情页面，查看网络配置信息。返回专属资源池列表，选择“网络”页签，找到专属资源池关联的网络，打通VPC。打通VPC网络后，网络列表和资源池详情页面将显示VPC名称，单击后可以跳转至VPC详情页面。

图 8-12 查找专属资源池

图 8-13 查看网络配置

图 8-14 打通 VPC



步骤2 VPC下创建弹性云服务器

登录弹性云服务器ECS控制台，单击右上角“购买弹性云服务器”，进入购买弹性云服务器页面，完成基本配置后单击“下一步：网络配置”，进入网络配置页面，选择1中打通的VPC，完成其他参数配置，完成高级配置并确认配置，下发购买弹性云服务器的任务。等待服务器的状态变为“运行中”时，弹性云服务器创建成功。单击“名称/ID”，进入服务器详情页面，查看虚拟私有云配置信息。

图 8-15 购买弹性云服务器时选择 VPC

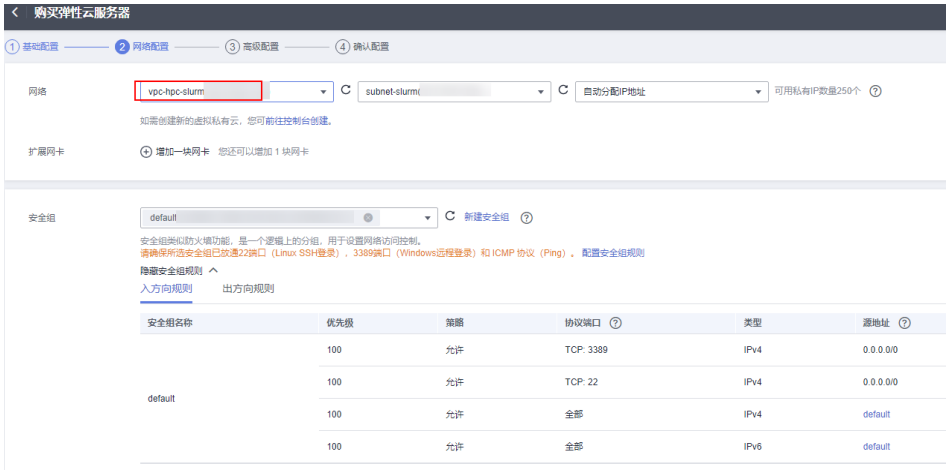


图 8-16 虚拟私有云

步骤3 获取在线服务的IP和端口号

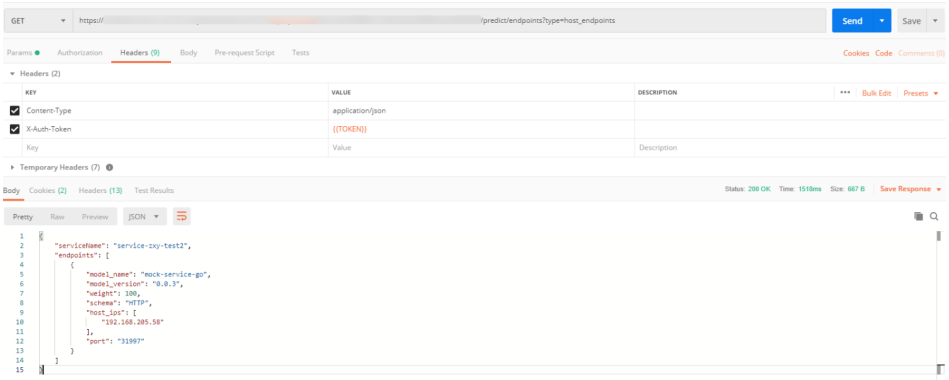
可以通过使用图形界面的软件（以Postman为例）获取服务的IP和端口号，也可以登录弹性云服务器（ECS），创建Python环境运行代码，获取服务IP和端口号。

API接口：

```
GET /v1/{project_id}/services/{service_id}/predict/endpoints?type=host_endpoints
```

- 方式一：图形界面的软件获取服务的IP和端口号

图 8-17 接口返回示例



- 方式二：Python语言获取IP和端口号

  Python代码如下，下述代码中以下参数需要手动修改：

  - project_id：用户项目ID，获取方法请参见《 ModelArts API参考 》中的“公共参数 > 获取项目ID和名称”章节。
  - service_id：服务ID，在服务详情页可查看。
  - REGION_ENDPOINT：服务的终端节点，查询请参见《 ModelArts API参考 》中的“使用前必读 > 终端节点”章节。

```
def get_app_info(project_id, service_id):
    list_host_endpoints_url = "{}/v1/{}/services/{}/predict/endpoints?type=host_endpoints"
    url = list_host_endpoints_url.format(REGION_ENDPOINT, project_id, service_id)
    headers = {'X-Auth-Token': X_Auth_Token}
    response = requests.get(url, headers=headers)
    print(response.content)
```

步骤4 通过IP和端口号直连应用

登录弹性云服务器（ ECS ），可以通过Linux命令行访问在线服务，也可以创建Python环境运行Python代码访问在线服务。schema、 ip、port参数值从3获取。

- 执行命令示例如下，直接访问在线服务。

```
curl --location --request POST 'http://192.168.205.58:31997' \
--header 'Content-Type:  application/json' \
--data-raw '{"a":"a"}'
```

图 8-18 访问在线服务

- 创建Python环境，运行Python代码访问在线服务。

```
def vpc_infer(schema, ip, port, body):
    infer_url = "{}://{}:{}"
    url = infer_url.format(schema, ip, port)
    response = requests.post(url, data=body)
    print(response.content)
```

说明

由于高速通道特性会缺失负载均衡的能力，因此在多实例时需要自主制定负载均衡策略。

----结束

8.3.1.4.4 WebSocket 访问在线服务

背景说明

WebSocket是一种网络传输协议，可在单个TCP连接上进行全双工通信，位于OSI模型的应用层。WebSocket协议在2011年由IETF标准化为RFC 6455，后由RFC 7936补充规范。Web IDL中的WebSocket API由W3C标准化。

WebSocket使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在WebSocket API中，浏览器和服务器只需要完成一次握手，两者之间就可以建立持久性的连接，并进行双向数据传输。

前提条件

- 在线服务部署时需选择“升级为WebSocket”。
- 在线服务中的AI应用导入选择的镜像需支持WebSocket协议。

约束与限制

- WebSocket协议只支持部署在线服务。
- 只支持自定义镜像导入AI应用部署的在线服务。

WebSocket 在线服务调用

WebSocket协议本身不提供额外的认证方式。不管自定义镜像里面是ws还是wss，经过ModelArts平台出去的WebSocket协议都是wss的。同时wss只支持客户端对服务端的单向认证，不支持服务端对客户端的双向认证。

可以使用ModelArts提供的以下认证方式：

- **token认证**

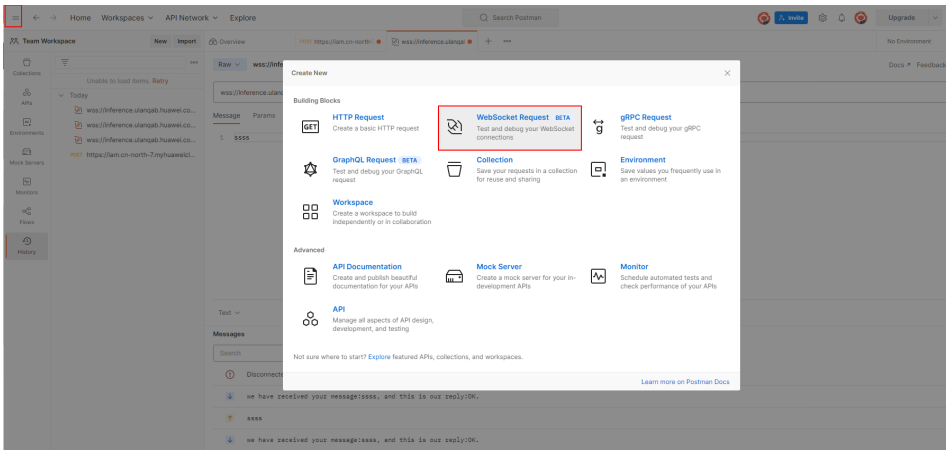
WebSocket服务调用步骤如下（以图形界面的软件Postman进行预测，token认证为例）：

1. **WebSocket连接的建立**
2. **WebSocket客户端和服务端双向传输数据**

步骤1 WebSocket连接的建立

1. 打开Postman（需选择8.5 以上版本，以10.12.0为例）工具，单击左上角 ，选择“File>New”，弹出新建对话框，选择“WebSocket Request”（当前为beta版本）功能：

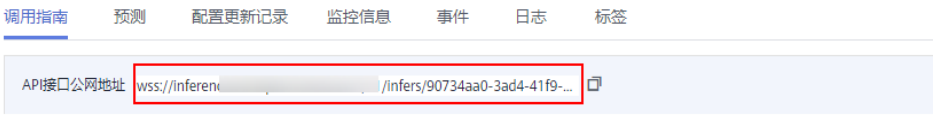
图 8-19 选择 WebSocket Request 功能



2. 在新建的窗口中填入WebSocket连接信息：

   左上角选择Raw，不要选择Socket.IO（一种WebSocket实现，要求客户端跟服务端都要基于Socket.IO），地址栏中填入从服务详情页“调用指南”页签中获取“API接口调用公网地址”后面的地址。如果自定义镜像中有更细粒度的地址，则在地址后面追加该URL。如果有queryString，那么在params栏中添加参数。在header中添加认证信息（不同认证方式有不同header，跟https的推理服务相同）。选择单击右上的connect按钮，建立WebSocket连接。

图 8-20 获取 API 接口调用公网地址

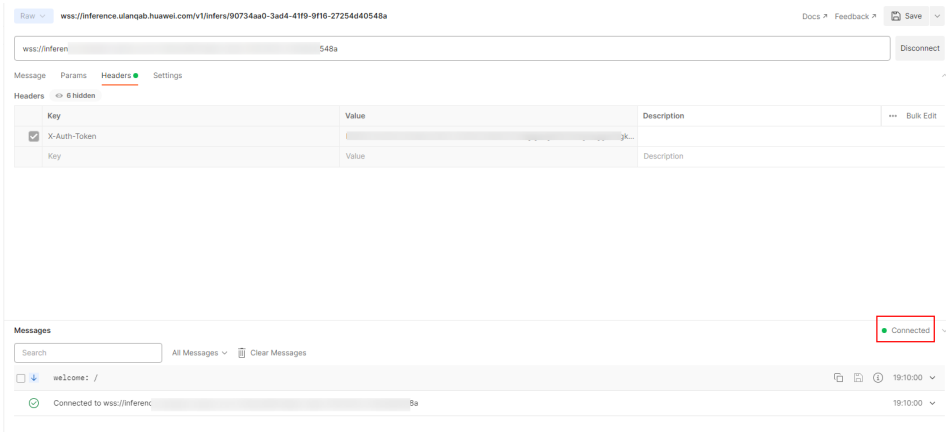


说明

- 如果信息正确，右下角连接状态处会显示：CONNECTED；
- 如果无法建立连接，如果是401状态码，检查认证信息；
- 如果显示WRONG_VERSION_NUMBER等关键字，检查自定义镜像的端口和ws跟wss的配置是否正确。

连接成功后结果如下：

图 8-21 连接成功



须知

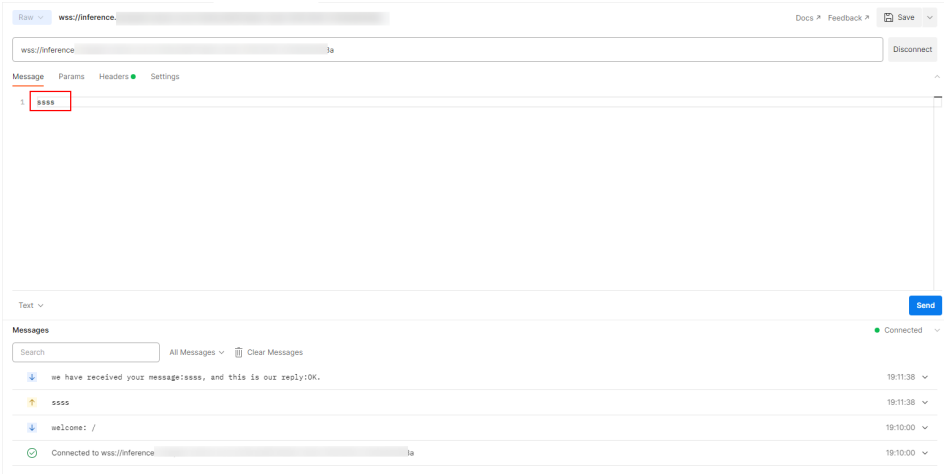
优先验证自定义镜像提供的websocket服务的情况，不同的工具实现的websocket服务会有不同，可能出现连接建立后维持不住，可能出现请求一次后连接就中断需要重新连接的情况，ModelArts平台只保证，未上ModelArts前自定义镜像的websocket的形态跟上了ModelArts平台后的websocket形态相同（除了地址跟认证方式不同）。

步骤2 WebSocket客户端和服务端双向传输数据

连接建立后，WebSocket使用TCP完成全双工通信。WebSocket的客户端可以往服务端发送数据，客户端有不同的实现，同一种语言也存在不同的lib包的实现，这里不考虑实现的不同种类。

客户端发送的内容在协议的角度不限定格式，Postman支持Text/Json/XML/HTML/Binary，以text为例，在输入框中输入要发送的文本，单击右侧中部的Send按钮即可将请求发往服务端，当文本内容过长，可能会导致postman工具卡住。

图 8-22 发送数据



----结束

8.3.1.4.5 Server-Sent Events 访问在线服务

背景说明

Server-Sent Events（SSE）是一种服务器向客户端推送数据的技术，它是一种基于HTTP的推送技术，服务器可以向客户端推送事件。这种技术通常用于实现服务器向客户端推送实时数据，例如聊天应用、实时新闻更新等。

SSE主要解决了客户端与服务器之间的单向实时通信需求（例如ChatGPT回答的流式输出），相较于WebSocket（双向实时），它更加轻量级且易于实现。

前提条件

在线服务中的AI应用导入选择的镜像需支持SSE协议。

约束与限制

- SSE协议只支持部署在线服务。
- 只支持自定义镜像导入AI应用部署的在线服务。

SSE 在线服务调用

SSE协议本身不提供额外的认证方式，和http请求方式一致。

可以使用ModelArts提供的以下认证方式：

- **token认证**

SSE服务调用如下（以图形界面的软件Postman进行预测，token认证为例）：

图 8-23 SSE 服务调用

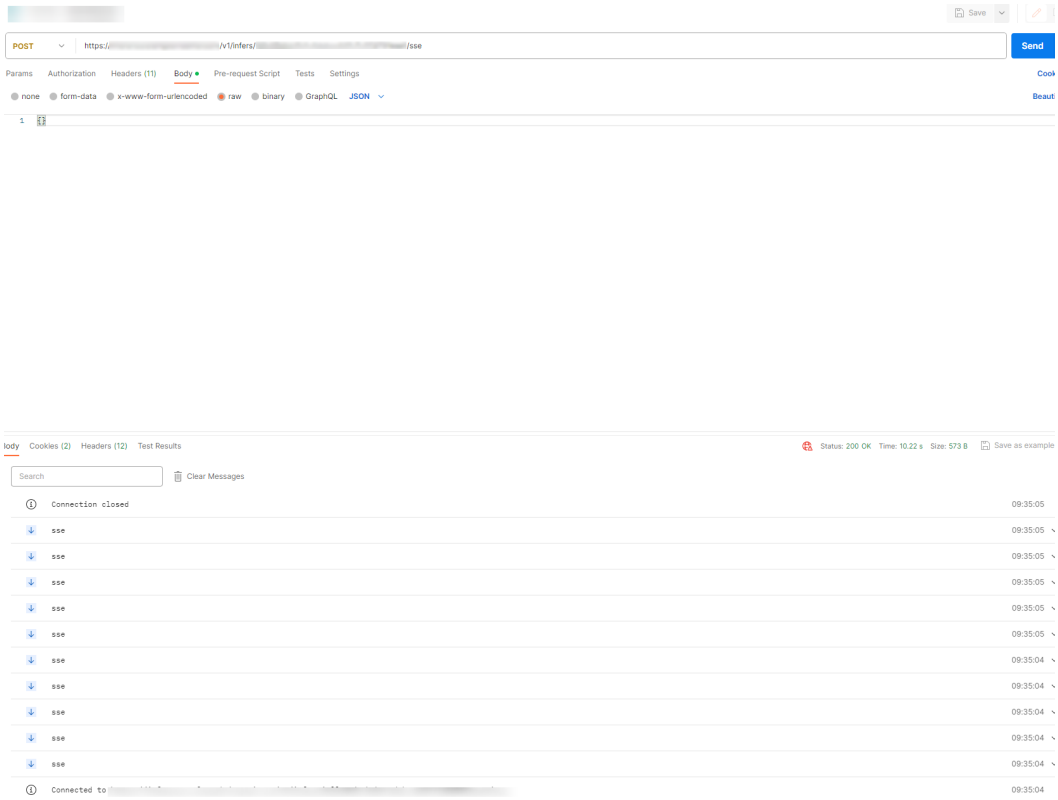


图 8-24 响应头 Content-Type

说明

正常情况下，可以观察到响应头Content-Type为text/event-stream;charset=UTF-8。

8.3.1.5 CloudShell

使用场景

允许用户使用ModelArts控制台提供的CloudShell登录运行中在线服务实例容器。

约束限制

- 只支持专属资源池部署的在线服务使用CloudShell访问容器。
- 在线服务必须处于“运行中”状态，才支持CloudShell访问容器。

如何使用 CloudShell

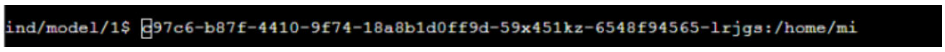
1. 登录ModelArts控制台，左侧菜单选择“部署上线 > 在线服务”。
2. 在线服务列表页面单击“名称/ID”，进入在线服务详情页面。
3. 单击CloudShell页签，选择AI应用版本和计算节点，当连接状态变为 时，即登录实例容器成功。

   若遇到异常情况服务器主动断开或超过10分钟未操作自动断开，此时可单击“重新连接”重新登录实例容器。

   说明

   部分用户登录Cloud Shell界面时，可能会出现路径显示异常情况，此时在Cloud Shell中单击回车键即可恢复正常。

   图 8-25 路径异常

8.3.2 部署 AI 应用（批量服务）

8.3.2.1 部署为批量服务

AI应用准备完成后，您可以将AI应用部署为批量服务。在“部署上线>批量服务”界面，列举了用户所创建的批量服务。

前提条件

- 数据已完成准备：已在ModelArts中创建状态“正常”可用的AI应用。
- 准备好需要批量处理的数据，并上传至OBS目录。
- 已在OBS创建至少1个空的文件夹，用于存储输出的内容。

背景信息

- 用户最多可创建1000个批量服务。
- 根据AI应用定义的输入请求不同（JSON文本或文件），不同的AI应用输入，需要填写的参数不同。当AI应用输入为JSON文件时，则需要根据配置文件生成映射文件；如果AI应用输入为文件时，则不需要。

- 批量服务只支持使用公共资源池，暂不支持使用专属资源池。

操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“部署上线 > 批量服务”，默认进入“批量服务”列表。
2. 在批量服务列表中，单击左上角“部署”，进入“部署”页面。
3. 在部署页面，填写批量服务相关参数。
   a. 填写基本信息。基本信息包含“名称”、“描述”。其中“名称”默认生成。例如：service-bc0d，您也可以根据实际情况填写“名称”和“描述”信息等。
   b. 填写服务参数。包含AI应用配置等关键信息，详情请参见**表8-20**。

表 8-20 参数说明

参数名称	说明
“AI应用来源”	根据您的实际情况选择“我的AI应用”。
“选择AI应用及版本”	选择状态“正常”的AI应用及版本。
“输入数据目录位置”	选择输入数据的OBS路径，即您上传数据的OBS目录。只能选择文件夹或“.manifest”文件。“.manifest”文件规范请参见**Manifest文件规范**。 **说明** • 输入数据为图片时，建议单张图片小于12MB。 • 输入数据格式为csv时，建议不要包含中文。如需使用中文，请将文件编码格式设置为UTF-8编码。您可以使用代码方式转换文件编码格式，也可以将csv文件用记事本方式打开，在另存为弹出的窗口页面设置编码格式。 • 输入数据格式为csv时，建议文件大小不超过12MB。
“请求路径”	批量服务中调用AI应用的接口URL，表示服务的请求路径，此值来自AI应用配置文件中apis的url字段。
“映射关系”	如果AI应用输入是json格式时，系统将根据此AI应用对应的配置文件自动生成映射关系。如果AI应用的输入是文件，则不需要映射关系。 自动生成的映射关系文件，填写每个参数对应到csv单行数据的字段索引，索引index从0开始计数。 映射关系生成规则：映射规则来源于模型配置文件“config.json”中输入参数（request）。当“type”定义为“string/number/integer/boolean”基本类型时，需要配置映射规则参数，即index参数。请参见**映射关系示例**了解其规则。 index必须是从0开始的正整数，当index设置不规则不符时，最终的请求将忽略此参数。配置映射规则后，其对应的csv数据必须以英文半角逗号分隔。
“输出数据目录位置”	选择批量预测结果的保存位置，可以选择您创建的空文件夹。

参数名称	说明
“计算节点规格”	请根据界面显示的列表，选择可用的规格，置灰的规格表示当前局点无法使用。
“计算节点个数”	设置当前版本AI应用的实例个数。如果节点个数设置为1，表示后台的计算模式是单机模式；如果节点个数设置大于1，表示后台的计算模式为分布式的。请根据实际编码情况选择计算模式。
“环境变量”	设置环境变量，注入环境变量到容器实例。为确保您的数据安全，在环境变量中，请勿输入敏感信息。
“部署超时时间”	用于设置单个模型实例的超时时间，包括部署和启动时间。默认值为20分钟，输入值必须在3到120之间。

4. 完成参数填写后，根据界面提示完成批量服务的部署。部署服务一般需要运行一段时间，根据您选择的数据量和资源不同，部署时间将耗时几分钟到几十分钟不等。

   您可以前往批量服务列表，查看批量服务的基本情况。在批量服务列表中，刚部署的服务“状态”为“部署中”，当批量服务的“状态”变为“运行完成”时，表示服务部署完成。

Manifest 文件规范

推理平台批量服务支持使用manifest文件，manifest文件可用于描述数据的输入输出。

输入manifest文件样例

- 文件名：“test.manifest”
- 文件内容：

```
{"source": "obs://test/data/1.jpg"}
{"source": "s3://test/data/2.jpg"}
{"source": "https://infers-data.obs.xxx.com:443/xgboosterdata/data.csv?AccessKeyId=2Q0V0TQ461N26DDL18RB&Expires=1550611914&Signature=wZBttZj5QZrReDhz1uDzwve8GpY%3D&x-obs-security-token=gQpzb3V0aGNoaW5hixvY8V9a1SnsxmGoHYmB1SArYMyqnQT-ZaMSxHvl68kKLAy5feYvLDM..."}
```

- 文件要求：
  a. 文件名后缀需为“.manifest”；
  b. 文件内容是多行JSON，每行JSON描述一个输入数据，需精确到文件，不能是文件夹；
  c. source的字段值是OBS的文件地址，格式为“<obs path>/{{桶名}}/{{对象名}}”。

输出manifest文件样例

批量服务的输出结果目录会有一个manifest文件。

- 假设用户输出结果路径为/test-bucket/test/，则结果存放位置如下：

```
OBS桶/目录名
├── test-bucket
│   ├── test
│   │   ├── infer-result-{{task_id}}.manifest
│   │   ├── infer-result
│   │   │   ├── 1.jpg_result.txt
│   │   │   ├── 2.jpg_result.txt
```

- infer-result-0.manifest文件内容：
  {"source": "obs://obs-data-bucket/test/data/1.jpg","result":"SUCCESSFUL","inference-loc": "obs://test-bucket/test/infer-result/1.jpg_result.txt"}
  {"source": "s3://obs-data-bucket/test/data/2.jpg","result":"FAILED","error_message": "Download file failed."}
  {"source ": "https://infers-data.obs.xxx.com:443/xgboosterdata/2.jpg?AccessKeyId=2Q0V0TQ461N26DDL18RB&Expires=1550611914&Signature=wZBttZj5QZrReDhz1uDzwve8GpY%3D&x-obs-security-token=gQpzb3V0aGNoaW5hixvY8V9a1SnsxmGoHYmB1SArYMyqnQT-ZaMSxHvl68kKLAy5feYvLDMNZWxzhBZ6Q-3HcoZMh9gISwQOVBwm4ZytB_m8sg1fL6isU7T3CnoL9jmvDGgT9VBC7dC1EyfSJrUcqfB_N0ykCsfrA1Tt_IQYZFDu_HyqVk-GunUcTVdDfWlCV3TrYcpmznZjliAnYUO89kAwCYGeRZsCsC0ePu4PHMsBvYV9gWmN9AUZIDn1sfRL4voBpwQnp6tnAgHW49y5a6hP2hCAoQ-95SpUriJ434QlymoeKfTHVMKOeZxZea-JxOvevOCGI5CcGehEJaz48sgH81UiHzl21zocNB_hpPfus2jY6KPglEJxMv6Kwmro-ZBXWuSJUDOnSYXI-3ciYjg9-h10b8W3sW1mOTFCWNGoWsd74it7l_5-7UUhoIeyPByO_REwkur2FOJsuMpGlRaPyglZxXm_jfdLFXobYtzZhbul4yWXga6oxTOkfcwykTOYH0NPoPRt5MYGYweOXXxFs3d5w2rd0y7p0QYhyTzIkk5CIz7FlWNapFISL7zdhsl8RfchTqESq94KgkeqatSF_iIvnYMW2r8P8x2k_eb6NJ7U_q5ztMbO9oWEcfr0D2f7n7Bl_nb2HIB_H9tjzKvqwngaimYhBbMRPfibvttW86GiwVP8vrC27FOn39Be9z2hSfJ_8pHej0yMlyNqZ481FQ5vWT_vFV3JHM-7I1ZB0_hIdaHfItm-J69cTfHSEOzt7DGaMIES1o7U3w%3D%3D","result":"SUCCESSFUL","inference-loc": "obs://test-bucket/test/infer-result/2.jpg_result.txt"}
- 文件格式：
  a. 文件名为“infer-result-{{task_id}}.manifest”，task_id为批量任务id，批量服务对应唯一的批量任务id。
  b. 当处理文件数目较多时，可能会有多个manifest文件，后缀相同，均为“.manifest”，文件名以后缀区分，例如“infer-result-{{task_id}}_1.manifest”等。
  c. manifest同一目录下会创建infer-result-{{task_id}}目录存放文件处理结果。
  d. 文件内容是多行JSON，每行JSON描述一个输入数据的对应输出结果。
  e. JSON内容包含多个字段。
     i. source：输入数据描述，与输入的manifest一致。
     ii. result：文件处理结果，值为SUCCESSFUL或FAILED，分别代表成功与失败。
     iii. inference-loc：输出结果路径，result为SUCCESSFUL时有此字段，格式为“obs://{{桶名}}/{{对象名}}”。
     iv. error_message：错误信息，result为FAILED时有此字段。

映射关系示例

如下示例展示了配置文件、映射规则、csv数据以及最终推理请求的关系。

假设，您的模型所用配置文件，其apis参数如下所示：

```
[
  {
    "method": "post",
    "url": "/",
    "request": {
      "Content-type": "multipart/form-data",
      "data": {
        "type": "object",
        "properties": {
          "data": {
            "type": "object",
            "properties": {
              "req_data": {
                "type": "array",
                "items": [
                  {
                    "type": "object",
```

```
                            "properties": {
                                "input_1": {
                                    "type": "number"
                                },
                                "input_2": {
                                    "type": "number"
                                },
                                "input_3": {
                                    "type": "number"
                                },
                                "input_4": {
                                    "type": "number"
                                }
                            }
                        }
                    ]
                }
            }
        }
    }
]
```

此时，其对应的映射关系如下所示。ModelArts管理控制台将从配置文件中自动解析映射关系，如果您调用ModelArts API时，需要自行根据规则编写映射关系。

```
{
    "type": "object",
    "properties": {
        "data": {
            "type": "object",
            "properties": {
                "req_data": {
                    "type": "array",
                    "items": [
                        {
                            "type": "object",
                            "properties": {
                                "input_1": {
                                    "type": "number",
                                    "index": 0
                                },
                                "input_2": {
                                    "type": "number",
                                    "index": 1
                                },
                                "input_3": {
                                    "type": "number",
                                    "index": 2
                                },
                                "input_4": {
                                    "type": "number",
                                    "index": 3
                                }
                            }
                        }
                    ]
                }
            }
        }
    }
}
```

用户需要进行推理的数据，即CSV数据，格式如下所示。数据必须以英文逗号隔开。

```
5.1,3.5,1.4,0.2
4.9,3.0,1.4,0.2
4.7,3.2,1.3,0.2
```

根据定义好的映射关系，最终推理请求样例如下所示，与在线服务使用的格式类似：

```
{
   "data": {
      "req_data": [{
         "input_1": 5.1,
         "input_2": 3.5,
         "input_3": 1.4,
         "input_4": 0.2
      }]
   }
}
```

8.3.2.2 查看批量服务预测结果

当您在部署批量服务时，会选择输出数据目录位置，您可以查看“运行完成”状态的批量服务运行结果。

操作步骤

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线>批量服务”，进入“批量服务”管理页面。
2. 单击状态为“运行完成”的目标服务名称，进入服务详情页面。
   - 您可以查看服务的“名称”、“状态”、“服务ID”、“输入数据目录位置”、“输出数据目录位置”和“描述”。
   - 您也可以通过单击描述右侧的  ，对描述信息进行编辑。
3. 从“输出数据目录位置”参数右侧获取详细OBS地址，前往此OBS目录，可以获取批量服务预测结果，包括预测结果文件和AI应用预测结果。

   若预测成功，目录下有预测结果文件和AI应用预测结果；若预测失败，目录下只有预测结果文件。
   - 预测结果文件：文件格式为“xxx.manifest”，里面包含文件路径、预测结果等信息。
   - AI应用预测结果输出：
     - 当输入为图片时，每张图片输出一个结果，输出结果格式为“图片名_result.txt”。例如：IMG_20180919_115016.jpg_result.txt。
     - 当输入为音频时，每个音频输出一个结果，输出结果格式为“音频名_result.txt”。例如：1-36929-A-47.wav_result.txt。
     - 当输入为表格数据时，输出结果格式为“表格名_result.txt”。例如：train.csv_result.txt。

8.3.3 部署 AI 应用（边缘服务）

8.3.3.1 部署为边缘服务

AI应用准备完成后，您可以将AI应用部署为边缘服务。在“部署上线>边缘服务”界面，列举了用户所创建的边缘服务。

前提条件

- 数据已完成准备：已在ModelArts中创建状态“正常”可用的AI应用。
- 如果选择“节点 > ModelArts边缘节点”部署边缘服务，请先创建ModelArts边缘节点。创建ModelArts边缘节点请参见**创建边缘节点**。
- 如果选择“边缘资源池”部署边缘服务，请先创建边缘资源池，创建边缘资源池请参见**创建边缘资源池**。

背景信息

用户最多可创建1000个边缘服务。

部署边缘服务（同步请求）

1. 登录ModelArts管理控制台，在左侧导航栏中选择“部署上线>边缘服务”，默认进入“边缘服务”列表。
2. 在边缘服务列表中，单击左上角“部署”，进入“部署”页面。
3. 在部署页面，填写边缘服务相关参数。
   a. 填写基本信息。基本信息包含“名称”、“描述”。其中“名称”默认生成。例如：service-bc0d，您也可以根据实际情况填写“名称”和“描述”信息。
   b. 填写服务参数。包含资源池、AI应用配置等关键信息，详情请参见**表8-21**。

表 8-21 参数说明

参数名称	说明
“部署方式”	可选择“节点”、“节点组”或“资源池”。 • **节点：**可选ModelArts边缘节点和IEF边缘节点。请指定“边缘节点类型”和“选择边缘节点”。 • **节点组：**在IEF创建的铂金版实例的边缘节点组。请指定对应的铂金版“资源实例”和“部署实例个数”。 • **资源池：**在ModelArts创建的边缘资源池。请指定“部署实例个数”和“选择边缘资源池”
“部署实例个数”	设置部署的实例个数。
“选择边缘资源池”	选择边缘资源池。
“选择AI应用及配置”	设置AI应用及对应配置。参见**表8-22**。

表 8-22 选择 AI 应用及配置

参数名称	说明
“AI应用来源”	根据您的实际情况选择“我的AI应用”。
“选择AI应用及版本”	选择状态“正常”的AI应用及版本。
“计算节点规格”	请根据界面显示的列表，选择可用的规格，置灰的规格表示当前无法使用。
“环境变量”	设置环境变量，注入环境变量到容器实例。为确保您的数据安全，在环境变量中，请勿输入敏感信息。
“网络配置”	当部署方式为边缘资源池且选择ModelArts边缘节点时显示该参数，当前仅支持“端口映射”作为容器镜像应用的访问方式。通过该方式访问时，需要配置容器端口、主机网卡地址、主机端口信息。主机端口可以指定或者自动获取，自动获取时，需要设置主机端口的上限和下限。
“数据存储”	当部署方式为边缘资源池且选择ModelArts边缘节点时显示该参数，设置数据的存储，需要配置存储卷类型、挂载卷名称、磁盘源、挂载路径、存储介质、权限参数。存储卷类型支持主机路径、临时路径和NFS。 **说明** 使用NFS数据存储，应提前在相关节点上安装好NFS服务，具体可参考**NFS服务安装与配置**。

4. 完成参数填写后，根据界面提示完成边缘服务的部署。部署服务一般需要运行一段时间，根据您选择的数据量和资源不同，部署时间将耗时几分钟到几十分钟不等。

   您可以前往边缘服务列表，查看边缘服务的基本情况。在边缘服务列表中，刚部署的服务“状态”为“部署中”，当边缘服务的“状态”变为“运行中”时，表示服务部署完成。在边缘服务列表中，可以查看边缘服务的请求模式和部署方式。

部署边缘服务（异步请求）

1. 登录ModelArts管理控制台，在左侧导航栏中选择“部署上线>边缘服务”，默认进入“边缘服务”列表。
2. 在边缘服务列表中，单击左上角“部署”，进入“部署”页面。
3. 在部署页面，填写边缘服务相关参数。

   a. 填写基本信息。基本信息包含“名称”、“描述”。其中“名称”默认生成。例如：service-bc0d，您也可以根据实际情况填写“名称”和“描述”信息。

   b. 填写服务参数。包含资源池、AI应用配置等关键信息，详情请参见**表8-23**。

表 8-23 参数说明

参数名称	说明
“选择边缘节点、选择边缘节点组、选择边缘资源池、”	可选择“节点”、“节点组”或“资源池”。 • **节点：**可选ModelArts边缘节点和IEF边缘节点。请指定“边缘节点类型”和“选择边缘节点”。 • **节点组：**在IEF创建的铂金版实例的边缘节点组。请指定对应的铂金版“资源实例”和“部署实例个数”。 • **资源池：**在ModelArts创建的边缘资源池。请指定“部署实例个数”和“选择边缘资源池”
“AI应用来源”	根据您的实际情况选择“我的AI应用”。
“选择AI应用及版本”	选择状态“正常”的AI应用及版本。
“流数量”	模型运行后支持同时处理视频流的数量。
“计算节点规格”	请根据界面显示的列表，选择可用的规格，置灰的规格表示当前无法使用。
“部署实例个数”	设置当前版本AI应用的实例个数。如果节点个数设置为1，表示后台的计算模式是单机模式。请根据实际编码情况选择部署实例个数。 **须知** 单次部署和修改边缘服务时，部署实例个数建议不大于10，否则可能触发限流导致部署失败。
“服务启动参数”	选中的AI应用在创建时设置了服务启动参数时显示。根据用户配置模型时候设置的服务启动参数进行配置。

4. 完成参数填写后，根据界面提示完成边缘服务的部署。部署服务一般需要运行一段时间，根据您选择的数据量和资源不同，部署时间将耗时几分钟到几十分钟不等。

   您可以前往边缘服务列表，查看边缘服务的基本情况。在边缘服务列表中，刚部署的服务“状态”为“部署中”，当边缘服务的“状态”变为“运行中”时，表示服务部署完成。

8.3.3.2 访问边缘服务（IEF 边缘节点）

当边缘服务和边缘节点的状态都处于“运行中”状态，表示边缘服务已在边缘节点成功部署。

您可以通过以下两种方式，对部署在边缘池上的边缘服务发起预测请求。

- **方式一：使用图形界面的软件进行预测（以Postman为例）**
- **方式二：使用curl命令发送预测请求**

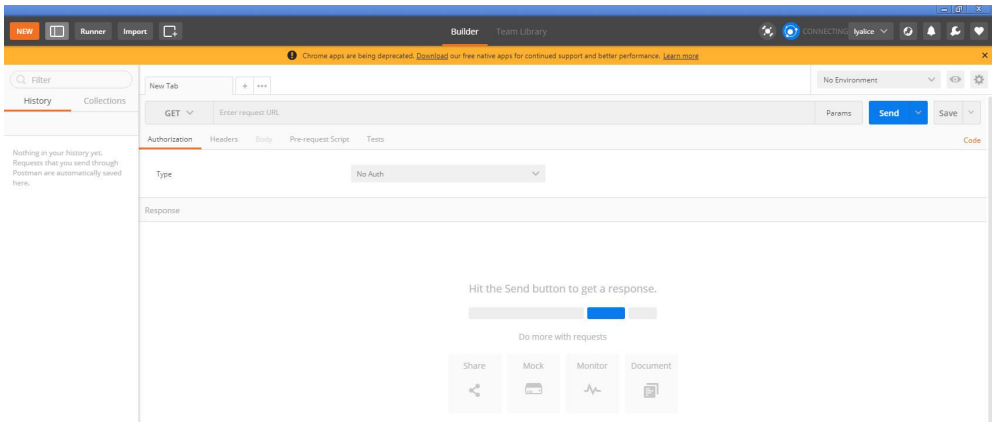
说明

本章节访问边缘服务方法，仅适用于IEF边缘节点类型的边缘服务。

方式一：使用图形界面的软件进行预测（以 Postman 为例）

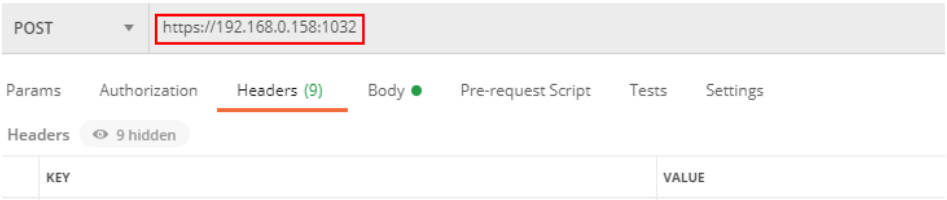
1. 下载Postman软件并安装，您可以直接在Chrome浏览器添加Postman扩展程序（也可使用其它支持发送post请求的软件）。
2. 打开Postman，如图8-26所示。

图 8-26 Postman 软件界面



3. 在Postman界面填写参数，以图像分类举例说明。
   - 选择POST任务，将某个边缘实例的调用地址（通过边缘服务详情中的“实例列表”页签查看URL，单击所属节点，跳转至节点详情界面-概览页签查看IP，当前支持IPv4和IPv6两种协议地址）复制到POST后面的方框。

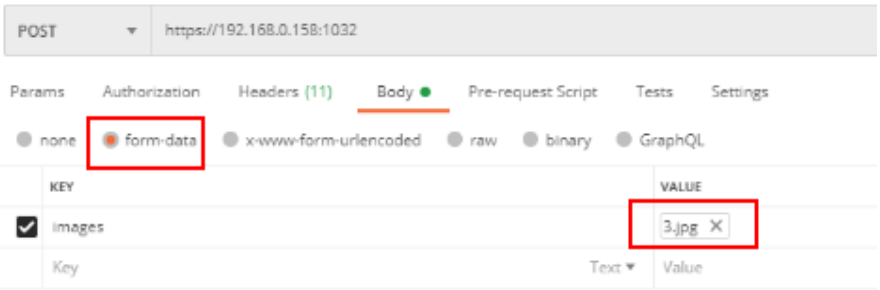
     图 8-27 POST 参数填写

   - 在Body页签，根据AI应用的输入参数不同，可分为“文件输入”或“文本输入”。
     - **文件输入**

       选择“form-data”。在“KEY”值填写AI应用的入参，比如本例中预测图片的参数为“images”。然后在“VALUE”值，选择文件，上传一张待预测图片（当前仅支持单张图片预测）。

       图 8-28 填写 Body 配置

- **文本输入**

  选择“raw”，选择JSON(application/json)类型，在下方文本框中填写请求体，请求体样例如下。

  ```
  {
  "meta": {
  "uuid": "10eb0091-887f-4839-9929-cbc884f1e20e"
  },
  "data": {
  "req_data": [
  {
  "sepal_length": 3,
  "sepal_width": 1,
  "petal_length": 2.2,
  "petal_width": 4
  }
  ]
  }
  }
  ```

  其中，“meta”中可携带“uuid”，返回预测结果时回传此“uuid”用于定位请求，**如无此需要可不填写meta。**“data”包含了一个“req_data”的数组，可传入单条或多条请求数据，其中每个数据的参数由AI应用决定，比如本例中的“sepal_length”、“sepal_width”等。

4. 参数填写完成，单击“Send”发送请求，结果会在Response下的对话框里显示。
   - 文件输入形式的预测结果，返回结果的字段值根据不同AI应用可能有所不同。
   - 文本输入形式的预测结果，请求体包含“meta”及“data”。如输入请求中包含“uuid”，则输出结果中回传此“uuid”。如未输入，则为空。“data”包含了一个“req_data”的数组，可传入单条或多条请求数据，其中每个数据的参数由模型决定，比如本例中的“sepal_length”、“sepal_width”等。

方式二：使用 curl 命令发送预测请求

使用curl命令发送预测请求的命令格式也分为文件输入、文本输入两类。

1. 文件输入

   curl -F '**images**=@**图片路径**'-X POST **边缘实例的调用地址** -k

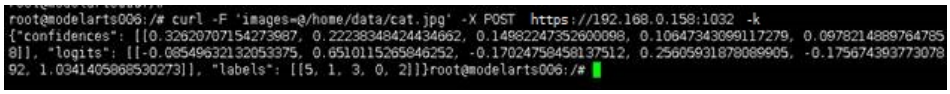
   - “-F”是指上传数据的是文件，本例中参数名为**images**，这个名字可以根据具体情况变化，@后面是图片的存储路径。
   - “POST”后面跟随的是边缘实例的调用地址。

   curl命令文件输入预测样例：

   ```
   curl -F 'images=@/home/data/cat.jpg' -X POST https://192.168.0.158:1032 -k
   ```

   预测结果如图8-29所示。

   图 8-29 curl 命令文件输入预测结果

2. 文本输入

   ```
   curl -d '{
   "meta": {
   "uuid": "10eb0091-887f-4839-9929-cbc884f1e20e"
   },
   ```

```
"data": {
"req_data": [
{
"sepal_length": 3,
"sepal_width": 1,
"petal_length": 2.2,
"petal_width": 4
}
]
}
}' -X POST <边缘实例的调用地址> -k
```

– “-d”是Body体的文本内容，如AI应用为文本输入，则需要用此参数。

curl命令文本输入预测样例：

```
curl -d '{
"meta": {
"uuid": "10eb0091-887f-4839-9929-cbc884f1e20e"
},
"data": {
"req_data": [
{
"sepal_length": 3,
"sepal_width": 1,
"petal_length": 2.2,
"petal_width": 4
}
]
}
}' -X POST https://192.168.0.158:1033 -k
```

8.3.3.3 访问边缘服务（ModelArts 边缘节点/资源池）

当边缘服务和边缘节点的状态都处于“运行中”状态，表示边缘服务已在边缘节点成功部署。

您可以通过以下两种方式，对部署在边缘资源池上的边缘服务发起预测请求。

- **方式一：使用图形界面的软件进行预测（以Postman为例）**
- **方式二：使用curl命令发送预测请求**

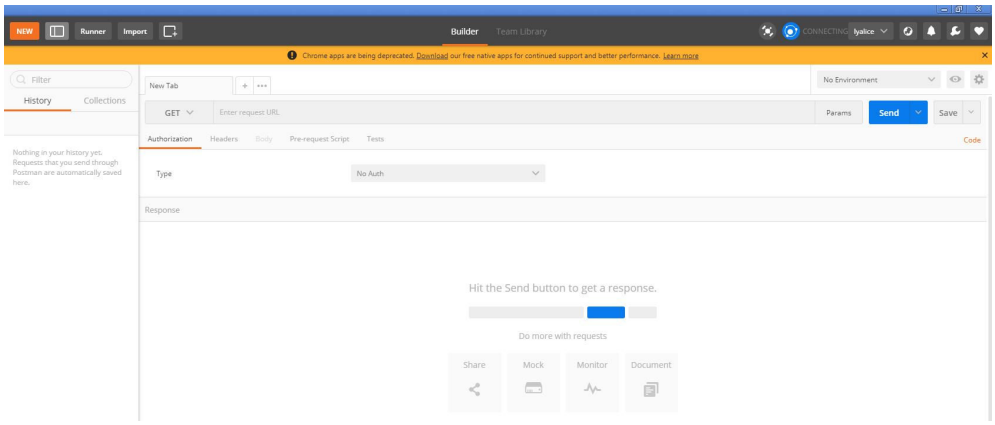
说明

本章节访问边缘服务方法，仅适用于ModelArts边缘节点和边缘资源池类型的边缘服务。

方式一：使用图形界面的软件进行预测（以 Postman 为例）

1. 下载Postman软件并安装，您可以直接在Chrome浏览器添加Postman扩展程序（也可使用其它支持发送post请求的软件）。
2. 打开Postman，如**图8-30**所示。

图 8-30 Postman 软件界面

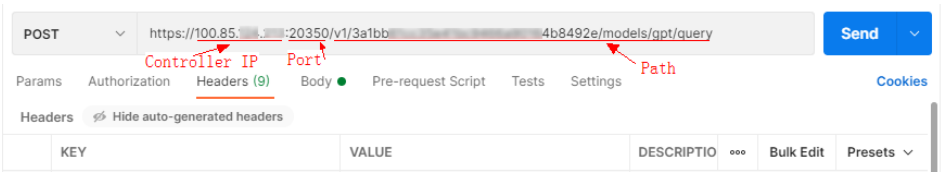


3. 在Postman界面填写参数，以智能问答举例说明。

   a. 选择POST任务，将某个边缘实例的调用地址复制到POST后面的方框。

   通过边缘服务中的“负载均衡”页签查看URL。查看资源池的转发配置，根据侦听端口、路径匹配规则填写URL。IP地址为主控节点的IP，登录主控节点Linux界面使用ifconfig命令可查看。

   图 8-31 POST 参数填写



   b. 在Body页签，根据AI应用的用途，输入要预测推理的内容。

   选择“raw”，选择JSON(application/json)类型，在下方文本框中填写请求体，请求体样例如下。

```
{
   "data":["Who are you?"]
}
```

   其中，“data”包含了一个数组，可传入单条或多条请求数据。

4. 参数填写完成，单击“Send”发送请求，结果会在Response下的对话框里显示。预测结果根据不同AI应用可能有所不同。

方式二：使用 curl 命令发送预测请求

使用curl命令发送预测请求的命令格式如下：

```
curl --location --request POST <边缘实例的调用地址>
--header 'Content-Type: application/json'
--data-raw '{
   "data":["Who are you?"]
}' -k
```

其中，“--data-raw”是Body体的文本内容，是AI应用的文本输入。

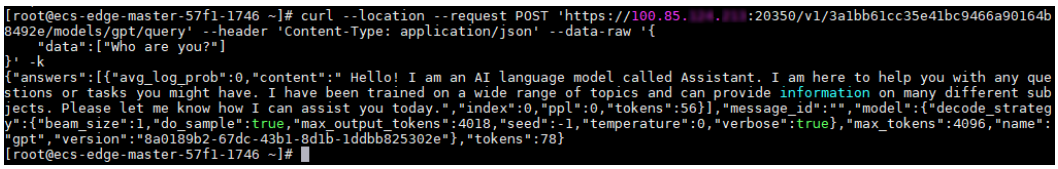
curl命令输入预测样例如下：

```
curl --location --request POST 'https://100.85.xxx.xxx:20350/v1/3a1bb61cc35e41bc9466a90164b8492e/
models/gpt/query'
--header 'Content-Type: application/json'
--data-raw '{
```

```
    "data":["Who are you?"]
}' -k
```

预测结果如图8-32所示。

图 8-32 curl 命令文本输入预测结果



8.3.3.4 负载均衡

ModelArts支持为ModelArts单个边缘节点或指定边缘资源池使用负载均衡，侦听节点和资源池物理IP的端口号，通过轮询算法，根据添加的转发配置，将请求转发到目标边缘服务和目标访问端口进行处理，提升应用的可靠性和稳定性。

创建负载均衡必须有可用的ModelArts边缘节点或边缘资源池，创建ModelArts边缘节点请参见**创建边缘节点**，创建边缘资源池请参见**创建边缘资源池**。

创建负载均衡

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“边缘服务”。
2. 选择“负载均衡”页签，在负载均衡列表页单击“创建”，进入创建负载均衡页面。
3. 在创建负载均衡页面，参见下表填写参数。

表 8-24 创建负载均衡参数说明

参数名称	说明
“名称”	负载均衡的名称，创建负载均衡后不可更改。
“描述”	负载均衡的简要描述。
“配置对象”	选择负载均衡的对象，可以对单个节点或指定资源池使用负载均衡。 • **节点：**选择ModelArts边缘节点。 • **资源池：**选择边缘资源池。
“侦听端口”	配置负载均衡侦听节点或资源池物理IP的端口号，取值范围为1-65535。
“分配策略”	资源池进行负载均衡调度时使用的算法。轮询算法（Round-Robin）指将请求轮流分配，从第1个节点到第N个节点，以此循环。
“会话保持”	负载均衡监听是基于IP地址的会话保持，开启会话保持后，来自同一个地址的访问请求转发到统一节点的边缘服务上。
“协议”	选择负载均衡的协议。可选HTTP和HTTPS，当前仅支持HTTP。

参数名称	说明
“转发配置”	设置转发配置，当请求的访问地址与转发配置匹配时，请求将会转发到目标进行处理。支持**“添加”**和**“删除”**转发配置。 ● “配置对象”为“节点”时，添加转测配置需要进行如下参数设置： – **路径匹配规则：**选择转发匹配的规则。当前支持前缀匹配、精确匹配、正则匹配。 – **路径：**设置转发匹配路径。匹配路径不能为空，必须以/开头，允许输入英文字母，数字，星号（*），斜杠（/）和中划线（-），长度为1-255位。 – **目标边缘服务名称**：选择转发的目标边缘服务。 – **容器端口**：设置容器的端口。 ● “配置对象”为“资源池”时，添加转测配置需要进行如下参数设置： – **路径匹配规则：**选择转发匹配的规则。当前支持前缀匹配、精确匹配、正则匹配。 – **路径：**设置转发匹配路径。匹配路径不能为空，必须以/开头，允许输入英文字母，数字，星号（*），斜杠（/）和中划线（-），长度为1-255位。 – **目标访问端口名称**：选择转发的目标访问端口。 – **访问端口**：选择访问服务的端口。

4. 确认配置无误后，单击“确认”，进入负载均衡列表页，等待负载均衡状态变为“运行中”，负载均衡创建完成。创建完成的负载均衡支持修改和删除。

创建访问端口

创建负载均衡之前，需要有创建完成的ModelArts资源池和ModelArts资源池部署的边缘服务。

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“边缘服务”。
2. 选择“负载均衡 > 访问端口”页签，在访问端口列表页单击“创建”，进入创建访问端口页面。
3. 在创建访问端口页面，参见下表填写参数。

表 8-25 创建访问端口参数说明

参数名称	说明
“名称”	访问端口的名称。
“访问类型”	访问端口的访问类型。当前仅支持ClusterIP类型。
“关联边缘服务”	请选择访问端口关联的边缘服务。

参数名称	说明
“端口配置”	设置端口配置，包括协议、访问端口和容器端口参数。支持**“添加”**和**“删除”**端口配置，至少添加1条数据。 ● **协议**：当前仅支持TCP协议。 ● **访问端口**：访问服务的端口。 ● **容器端口**：服务访问目标容器的端口，此端口与容器中运行的应用相关。

4. 确认配置无误后，单击“确认”，进入访问端口列表页，等待访问端口状态变为“运行中”，访问端口创建完成。创建完成的访问端口支持修改和删除。

访问负载均衡

根据上面创建好的访问端口和负载均衡，可以做如下调用：

```
curl -X POST \
https://100.85.220.207:13458/v1/models/gpt/query \
-d`{
    "data": ["你好"]
  }`
```

其中，100.85.220.207是主控节点IP，13458是创建负载均衡时指定的侦听端口，URI /v1/models/gpt/query要满足路由匹配规则。

8.3.3.5 NFS 服务安装与配置

NFS服务是ModelArts边缘资源池提供的数据存储卷服务，创建部署时可通过NFS挂载的方式访问共享数据，比如obs的模型文件。

涉及以下场景时，必须为资源池配置NFS服务：

- 创建AI应用时，元模型来源选择“从对象存储服务（OBS）中选择”，且AI引擎选择“Custom”。
- 创建部署时，数据存储使用NFS类型的存储卷。

安装 NFS 服务

1. 登录存储节点。

   在边缘资源池中，选定一个节点作为存储节点。该节点提供NFS网盘服务，用于存放集群共享的文件。建议使用存储空间足够大（能存放下大模型文件）的节点作为存储节点。使用Putty工具登录存储节点。

   ```
   ssh <用户名>@<节点IP>
   ```

   - 用户名：登录服务器的用户名。
   - 节点IP：登录服务器的IP地址。若节点是云服务器，可在云服务器控制台中查询。

2. 安装NFS。

   该步骤需要设备联网下载软件依赖包。

   - **Ubuntu系统**

     在线安装：

     ```
     sudo apt install nfs-kernel-server
     ```

- **Euler OS系统**

  在线安装：

  ```
  sudo yum install nfs-utils
  ```

3. 创建模型目录。

   该路径的存储空间能够存储大模型文件。

   ```
   mkdir -p /var/docker/hilens
   ```

4. 添加访问权限。

   配置nfs-server访问白名单和文件存储路径。

   ```
   vim /etc/exports
   ```

   添加如下配置：

   ```
   /var/docker/hilens 192.168.0.0/24(rw,no_all_squash,anonuid=1000,anongid=100,fsid=0)
   ```

   192.168.0.0/24为集群内网IP网段（登录主控节点，使用ifconfig命令查看IP地址）。

5. 加载配置。

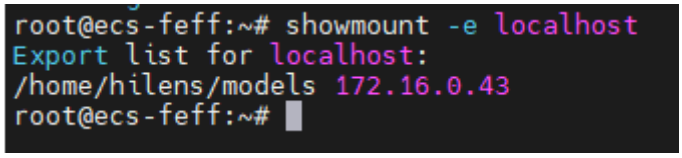
   ```
   exportfs -rv
   ```

6. 启动NFS和rpcbind。

   ```
   systemctl enable nfs-server && systemctl enable rpcbind && systemctl start rpcbind nfs-server
   ```

7. 执行如下命令，验证以上配置内容是否正确。如下图，表示配置正确，即NFS服务安装成功。

   ```
   showmount -e localhost
   ```

ModelArts 节点信息配置

1. 登录主控节点的Linux机器。

   ```
   ssh <用户名>@<节点IP>
   ```

   - 用户名：登录服务器的用户名。
   - 节点IP：登录服务器的IP地址。若节点是云服务器，可在云服务器控制台中查询。

2. 配置固件启动参数。

   ```
   vim /etc/hilens/hda.conf
   ```

   增加如下配置，“192.168.xxx.xxx”需要替换为您实际的NFS存储节点的内网IP：

   ```
   hilens.nfs.server.ip=192.168.xxx.xxx
   hilens.nfs.mount.dir=/home/mind/model
   hilens.nfs.source.dir=/var/docker/hilens
   ```

   参数说明：

   - hilens.nfs.server.ip ： NFS存储节点的内网IP。
   - hilens.nfs.mount.dir：大模型默认挂载路径，即容器内访问路径，由镜像决定。
   - hilens.nfs.source.dir：大模型下载路径，即存储节点的共享目录。该目录必须先在/etc/exports中配置共享权限；否则会导致无权限挂载。

3. 重启固件。

   ```
   systemctl restart hdad
   ```

8.3.4 升级服务

对于已部署的服务，您可以修改服务的基本信息以匹配业务变化，更换AI应用的版本号，实现服务升级。

您可以通过如下两种方式修改服务的基本信息：

方式一：通过服务管理页面修改服务信息

方式二：通过服务详情页面修改服务信息

前提条件

服务已部署成功，“部署中”的服务不支持修改服务信息进行升级。

约束限制

- 服务升级关系着业务实现，不当的升级操作会导致升级期间业务中断的情况，请谨慎操作。
- ModelArts支持部分场景下在线服务进行无损滚动升级。按要求进行升级前准备，做好验证，即可实现业务不中断的无损升级。

表 8-26 支持无损滚动升级的场景

创建AI应用的元模型来源	服务使用的是公共资源池	服务使用的是专属资源池
从训练中选择元模型	不支持	不支持
从模板中选择元模型	不支持	不支持
从容器镜像中选择元模型	不支持	支持，创建AI应用的自定义镜像需要满足**创建AI应用的自定义镜像规范**。
从OBS中选择元模型	不支持	不支持

方式一：通过服务管理页面修改服务信息

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。
2. 在服务列表中，单击目标服务操作列的“修改”，修改服务基本信息，然后根据提示提交修改任务。

   当修改了服务的某些参数配置时，系统会自动重启服务使修改生效。在提交修改服务任务时，若涉及重启，会有弹窗提醒。

   - 在线服务参数说明请参见**部署为在线服务**。修改在线服务还需要配置“最大无效实例数”设置并行升级的最大节点数，升级阶段节点无效。
   - 批量服务参数说明请参见**部署为批量服务**。

方式二：通过服务详情页面修改服务信息

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。

2. 单击目标服务名称，进入服务详情页面。
3. 您可以通过单击页面右上角“修改”，修改服务基本信息，然后根据提示提交修改任务。

   当修改了服务的某些参数配置时，系统会自动重启服务使修改生效。在提交修改服务任务时，若涉及重启，会有弹窗提醒。

   - 在线服务参数说明请参见**部署为在线服务**。修改在线服务还需要配置“最大无效实例数”设置并行升级的最大节点数，升级阶段节点无效。
   - 批量服务参数说明请参见**部署为批量服务**。

8.3.5 启动、停止、删除、重启服务

启动服务

您可以对处于“运行完成”、“异常”和“停止”状态的服务进行启动操作，“部署中”状态的服务无法启动。您可以通过如下方式启动服务：

- 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。您可以单击“操作”列的“启动”，启动服务。
- 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。单击目标服务名称，进入服务详情页面。您可以单击页面右上角“启动”，启动服务。

说明

部署方式为ModelArts边缘节点和ModelArts边缘资源池的服务不支持启动。

停止服务

您可以通过如下方式停止服务：

- 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。您可以单击“操作”列的“停止”（在线服务在操作列选择“更多 > 停止”），停止服务。
- 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。单击目标服务名称，进入服务详情页面。您可以单击页面右上角“停止”，停止正在运行中服务。

说明

部署方式为ModelArts边缘节点和ModelArts边缘资源池的服务不支持停止。

删除服务

如果服务不再使用，您可以删除服务释放资源。

登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入目标服务类型管理页面。

- 在线服务
  - 单击在线服务列表“操作”列的“更多>删除”删除服务。
  - 勾选在线服务列表中的服务，然后单击列表左上角“删除”按钮，批量删除服务。

  - 单击目标服务名称，进入服务详情页面，单击右上角“删除”删除服务。
- 批量服务
  - 单击批量服务列表“操作”列的“删除”，删除服务。
  - 勾选批量服务列表中的服务，然后单击列表左上角“删除”按钮，批量删除服务。
  - 单击目标服务名称，进入服务详情页面，单击右上角“删除”按钮进行删除。

说明

- 删除操作无法恢复，请谨慎操作。
- 没有委托授权时，无法删除服务。

重启服务

只有当在线服务处于“运行中”或“告警”状态时，才可进行重启操作。批量服务、边缘服务不支持重启。您可以通过如下方式重启在线服务：

- 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入在线服务列表页面。您可以单击“操作”列的“更多>重启”，重启服务。
- 登录ModelArts管理控制台，在左侧菜单栏中选择“部署上线”，进入在线服务列表页面。单击目标服务名称，进入服务详情页面。您可以单击页面右上角“重启”，重启在线服务

说明

部署方式为ModelArts边缘节点和ModelArts边缘资源池的服务不支持重启。

8.3.6 查看服务的事件

服务的（从用户可看见部署服务任务开始）整个生命周期中，每一个关键事件点在系统后台均有记录，用户可随时在对应服务的详情页面进行查看。

方便用户更清楚的了解服务部署和运行过程，遇到任务异常时，更加准确的排查定位问题。可查看的事件点包括：

表 8-27 事件

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
正常	开始部署服务。 Start to deploy service.	-
异常	资源不足，等待资源释放。 Lack of resources, transform state to waiting.	等待资源释放后重试。
异常	xxx资源不足，服务调度失败。补充信息：xxx %s %s Schedule failed due to insufficient resources. Retry later. %s nodes are available: %s Insufficient memory.	根据补充信息，了解资源不足详情，参考FAQ处理。

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
正常	开始构建镜像。 Start to build image.	-
异常	构建模型(xxx) 镜像失败，构建日志:\nxxx。 Failed to build image for model (%s %s), docker build log:\n%s.	根据构建日志定位和处理问题。
异常	构建镜像失败。 Failed to build image.	请联系技术支持。
正常	构建镜像完成。 Image built successfully.	-
异常	xxx服务失败。错误信息：xxx Failed to %s service, retry later. Error message: %s	请根据错误信息定位和处理问题。
异常	更新服务失败，执行回滚操作。 Failed to update service, rollback it.	请联系技术支持。
正常	服务更新中。 Updating service.	-
正常	服务启动中。 Starting service.	-
正常	服务停止中。 Stopping service.	-
正常	服务已停止。 Service stopped.	-
正常	自动停止开关已关闭。 Auto-stop switched off.	-
正常	自动关闭功能开启，服务将在xs后停止。 Auto-stop switched on, service will be stopped in %d %s.	-
正常	到达自动停止时间，服务停止。 Service stopped automatically because due time is reached.	-
异常	配额超限，服务停止。 Service stopped automatically because over quota.	请联系技术支持。

事件类型	事件信息（“XXX”表示占位符，以实际返回信息为准）	解决方案
异常	自动停止服务失败，错误信息: xxx Failed to stop service automatically, error message: %s	请根据错误信息定位和处理问题。
正常	删除资源池(xxx)上服务实例。 Model in node(%s) deleted.	-
正常	停止资源池(xxx)上服务实例。 Model in node(%s) stopped.	-
异常	批量服务失败，请稍后重试。错误信息：xxx Failed to %s batch service, retry later. Error message: %s.	请根据错误信息定位和处理问题。
正常	服务运行完成。 Service stopped automatically after running.	-
异常	停止服务失败，错误信息：xxx Failed to stopped service, error message: %s	请根据错误信息定位和处理问题。
正常	订阅许可即将超期：xxx Impending expiration notice: %s	-
正常	服务xxx启动成功。 Service %s started successfully.	-
异常	启动服务xxx失败。 Service %s started failed.	启动服务失败情况较多，请参考FAQ定位和处理。
异常	部署服务超时，错误信息：xxx Deploying timeout, details: %s	请根据错误信息定位和处理问题。
正常	更新服务失败，执行回滚操作成功。 Failed to update service, rollback succeeded.	-
异常	更新服务失败，执行回滚操作失败。 Failed to update service, rollback failed.	请联系技术支持。

服务部署和运行过程中，关键事件支持手动/自动刷新。

查看操作

1. 在ModelArts管理控制台的左侧导航栏中选择“部署上线 > 在线服务|批量服务|边缘服务”，在服务列表中，您可以单击名称/ID，进入服务详情页面。
2. 在服务详情页面，切换到“事件”页签，查看事件信息。

8.4 边缘资源池

8.4.1 边缘资源池简介

边缘资源池是边缘服务部署专用的资源池，是租户边缘侧的运行节点集合。推理服务在边缘池上运行，用户创建对应异步服务或边缘同步服务后，边缘服务会调度选择合适的节点运行异步算法容器，进行异步服务或边缘同步服务的处理。

- **节点**

  ModelArts边缘节点是ModelArts平台提供的用于部署边缘服务的终端设备。创建边缘资源池之前需要先创建ModelArts边缘节点并激活节点。

- **资源池**

  边缘资源池是边缘服务部署专用的资源池。创建边缘资源池时，可以添加ModelArts边缘节点设备，也可以添加IEF纳管的边缘节点设备。

图 8-33 创建 ModelArts 边缘资源池流程图

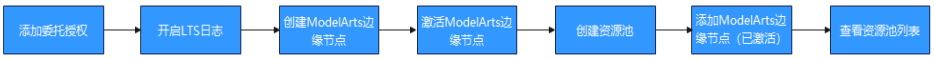


图 8-34 创建 IEF 边缘资源池流程图

8.4.2 节点

ModelArts边缘节点是ModelArts平台提供的用于部署边缘服务的终端设备。创建边缘资源池之前需先创建ModelArts边缘节点。节点创建完成后，同步下载证书和边缘Agent固件，及时将固件复制到节点上，并执行注册命令完成设备的注册。

您可以对边缘节点进行激活、修改、删除和创建边缘服务操作，同时支持查看边缘节点的详情信息。

边缘节点规格要求

边缘节点既可以是物理机，也可以是虚拟机。边缘节点需要满足下表的规格要求。

项目	规格
OS	• x86_64架构 Ubuntu LTS (Xenial Xerus)、Ubuntu LTS (Bionic Beaver) 、CentOS、EulerOS、openEuler • aarch64（arm64）架构 Ubuntu LTS (Bionic Beaver)、CentOS、EulerOS、 openEuler
内存	边缘节点内存基础开销约64M，根据业务不同，内存开销不同，建议配置大于256M。
CPU	基础开销>=1核，docker+k3s/k8s场景建议配置>=4核
硬盘	>=512M
GPU（可选）	同一个边缘节点/边缘池的GPU型号必须相同。注册边缘节点时，可以选择是否使用GPU。如果边缘节点要使用GPU，需要在注册边缘节点前安装GPU驱动。 **说明** 当前支持Nvidia Tesla系列P3/P4、T4等型号GPU。
NPU（可选）	昇腾AI加速处理器。注册边缘节点时，可以选择是否使用NPU加速。 • 如果边缘节点要使用NPU，需求确保边缘节点已安装驱动，具体安装方式请联系设备厂商获取支持。 • 如果要在边缘资源池（集群）场景下使用NPU加速卡，还需要安装集群相关的插件，具体安装方式请联系设备厂商获取支持。 **说明** 当前支持Snt9/Snt9B/Snt3P/Snt3系列的NPU加速卡。

容器引擎	Docker版本大于等于19.0.0。可通过以下命令安装：
	`curl -fsSL get.docker.com -o get-docker.sh`
	`sh get-docker.sh`
	`sudo systemctl daemon-reload`
	`sudo systemctl restart docker`

K3s引擎	v1.21.12+k3s1
端口使用	边缘节点需要使用5066/5067端口。对于边缘池（集群）场景，还需要预留k8s/k3s相关端口，包括2379、2380、6443等。 **说明** 5066端口用于提供http/https rest服务。5067作为工作节点k8s client代理。
时间同步	边缘节点时间需要与UTC标准时间保持一致，否则会导致边缘节点的监控数据出现偏差。您可以选择合适的NTP服务器进行时间同步，从而保持时间一致。也可以保持和云端常连接，同步云端时间。

委托授权

边缘节点在使用时，需要用到日志服务，需为用户进行委托授权，并**开启LTS日志**功能。为用户添加委托授权步骤如下：

步骤1 创建“hilens_admin_trust”委托。

1. 登录ModelArts控制台，在左侧菜单栏中选择“全局配置”。

2. 单击“添加授权”新增委托，设置委托名称为“hilens_admin_trust”，“权限配置”选择“自定义”，并选中OBS Administrator和SWR Administrator权限。
3. 单击“创建”，完成委托添加。

步骤2 为“hilens_admin_trust”委托添加“LTS Administrator”权限。

1. 在控制台上方菜单栏单击“系统”。
2. 选择“权限管理 > 委托管理”，在列表中单击“hilens_admin_trust”委托操作列的“修改”。
3. 切换到“委托权限”页签，单击“添加权限”，选择“资源空间”，并在权限列表右上方搜索框中搜索并选中“LTS Administrator”和“AOM FullAccess”权限。
4. 单击“确定”，在确认弹框“确认”后，即完成权限添加。

----结束

创建边缘节点

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“边缘资源池”。
2. 在“节点”列表页，单击“创建”，进入创建边缘节点页面。
3. 在创建边缘节点页面参见下表填写参数。

表 8-28 创建边缘节点参数说明

参数名称	说明
节点名称	节点的名称。由中文字符，英文字母，数字，下划线和中划线组成，输入长度为1-64位。
描述	节点的简要描述。不允许输入字符#~^$%&*<>[]|/及ASCII(0-31)，输入长度为0-255位。
AI加速卡	设置节点是否使用加速卡。 ● 不使用：不使用节点加速卡。 ● **GPU**：使用GPU加速卡。 ● **Ascend**：使用Ascend加速卡。Ascend加速卡类型包括：snt3、snt3p和snt9。
批量注册	多个节点可使用同一份证书进行节点注册。默认关闭。开启后，设置“批量注册数量”配置节点数个数。
日志存储时间	日志存储时长，到期后系统会自动清理过期的日志数据。单位为天，输入值范围必须在1-30之间。

参数名称	说明
系统日志设置	设置系统日志的等级、大小限制、滚动数量以及日志是否上传。 ● 日志等级：选择日志的等级，默认为Debug。可选Error、Warning、Info、Debug级别的日志，分别代表：错误、警告、信息、调试。 ● 日志大小限制：对该注册节点的日志大小限制，默认为50MB，不可自定义。 ● 日志滚动数量：每隔N天，根据日志数量对本地日志进行一次滚动删除，删除最旧的日志文件。每个日志文件固定10M。默认为5天。 ● 日志上传：默认关闭。系统日志与应用日志默认在本地存储，开启“日志上传”后，并选择日志等级，将对应等级的日志上传至云日志服务（LTS）。
应用日志设置	设置应用日志的等级、大小限制、滚动数量以及日志是否上传。 ● 日志等级：选择日志的等级，默认为Debug。可选Error、Warning、Info、Debug级别的日志，分别代表：错误、警告、信息、调试。 ● 日志大小限制：对该注册节点的日志大小限制，默认为50MB，不可自定义。 ● 日志滚动数量：每隔N天，根据日志数量对本地日志进行一次滚动删除，删除最旧的日志文件。每个日志文件固定10M。默认为5天。 ● 日志上传：默认关闭。系统日志与应用日志默认在本地存储，开启“日志上传”后，并选择日志等级，将对应等级的日志上传至云日志服务（LTS）。

4. 确认配置无误后，单击“确认”，创建节点任务下发，进入“完成，下载证书和固件”页面。
5. 在证书名称后单击“立即下载”，下载证书文件。证书在下载后24小时内有效，请在证书有效期内完成注册。证书文件仅在填写注册基础信息后可下载， 关闭页面后无法再次下载 。
6. 下载边缘Agent固件，配置参数如下：

表 8-29 下载边缘 Agent 配置参数说明

参数名称	说明
CPU架构	选择CPU架构类型。可选X86、ARM32、ARM64。
操作系统	选择操作系统。可选Linux、Windows。
Agent名称	选择需下载的Agent名称

参数名称	说明
版本号	请选择一个Agent对应的版本，当前可选的版本号为2.0.26。 **说明** 在安装部署完ModelArts后，需先手动上传Agent固件，固件上传后，控制台中才会显示固件版本号。上传固件可参见“ModelArts 6.5.0.1 维护指南(for HCS Online 24.3.0) 01 > ModelArts 配置指南 01” 的 ModelArts边缘节点录入新固件（ModelArts Edge Agent）章节。

说明

根据节点类型下载固件，根据**注册证书**，将固件复制到节点上，执行注册命令完成设备注册。

7. 完成证书和固件下载后，勾选“我已完成证书和固件下载”复选框，单击“完成”。

注册证书

创建边缘节点成功后，需要注册节点证书。固件的操作系统不同，节点注册证书的方式不同。

- Windows操作系统

  前提条件：PC系统版本要求为Windows 10及以上。

  a. 解压**6**下载的固件压缩包，并运行程序。
  b. 将**5**下载的证书，复制到解压的文件目录下。
  c. 打开CMD命令行程序，切换到解压的文件目录下。
  d. 执行注册命令。
  ```
  hdad.exe hdactl bind -p {证书名称}
  ```

- Linux操作系统

  a. 登录Linux机器后，将**6**下载的固件压缩包，复制到Linux机器的任意位置。
  b. 解压Agent固件压缩包并安装Agent。命令如下：
  ```
  tar -xvf {固件包名}
  ```
  c. 安装Agent固件。命令如下：
  ```
  sh {运行文件}
  ```
  d. 将**5**下载的证书，复制到固件解压包的文件目录下。
  e. 执行注册命令。
  ```
  hdactl bind -p {证书名称}
  ```

激活边缘节点

节点注册完成后，需要进行激活操作。未注册（UNCONNECTED）和已激活（ACTIVATED）状态的节点不支持激活操作。

1. 在“边缘资源池 > 节点”页面，单击目标节点操作列的“激活”，或者批量选择节点，单击列表上方的“激活”按钮，进入激活边缘节点页面，参见下表填写参数。

表 8-30 激活边缘节点参数说明

参数名称	说明
节点名称	激活节点的名称。
节点数量	激活节点的数量。当前仅支持激活50个节点。
生效时间	设置节点激活的生效时间。当前仅支持“立即生效”。
试用时长	选择边缘节点的试用时长，当前支持选择“1个月”、“3个月”。

2. 确认配置无误后，单击“确认”，激活节点。

修改边缘节点

在“边缘资源池 > 节点”页面，单击目标节点操作列的“修改”，进入修改边缘节点页面，对节点进行修改操作。参数说明请参见表8-28。

说明

- 所属子用户、工作空间、批量注册参数不支持重新配置。
- 节点名称、日志大小限制、日志滚动数量不支持修改。

删除边缘节点

在“边缘资源池 > 节点”页面，选择目标操作列的“更多 > 删除”，删除边缘节点。删除边缘节点前，需先删除节点绑定的资源池。边缘节点删除后，无法恢复，请谨慎操作。也可单击“强制删除”，删除工作节点。

说明

- 边缘资源池的“主控节点”不能删除。
- 边缘资源池的状态是“运行中”时，可通过“强制删除”功能，删除故障工作节点。
- 已创建服务和负载均衡的节点，可通过“强制删除”功能，进行删除操作。

查看边缘节点详情

在“边缘资源池 > 节点”页面，单击目标节点名称，进入节点详情页面，查看边缘节点详情信息。

表 8-31 基本信息参数说明

参数名称	说明
名称	节点的名称。
ID	节点的ID。
节点规格	节点的规格。
描述	节点的简要描述。

参数名称	说明
操作系统	节点的操作系统。
架构	节点的架构。

表 8-32 管理信息参数说明

参数名称	说明
状态	节点的状态。节点状态枚举值：未注册（UNCONNECTED）、运行中（RUNNING）、故障（FAULTY）、升级中（UPGRADING）、冻结（FREEZE）。
Agent版本	节点绑定固件的版本。
Agent名称	节点绑定固件的名称。
证书/配置文件	节点的证书和配置文件。
推理加速卡	节点上挂载的推理加速卡。
IP	节点的IP。
激活状态	节点的激活状态。激活状态枚举值：未激活（INACTIVE）、已激活（ACTIVATED）。
激活到期时间	节点激活的过期时间。
创建时间	节点的创建时间。
更新时间	节点的最后更新时间。
所属IAM用户	节点所属的子用户。
所属资源池	节点所属的资源池。

您可以在节点详情页面，通过切换页签查看更多详细信息。

表 8-33 节点详情信息

参数名称	说明
边缘服务	展示该边缘节点下的边缘服务部署情况，支持以下操作： ● 查询绑定的边缘服务列表，如名称、状态、请求模式、创建时间、描述。 ● 创建新的边缘服务。 ● 修改、删除边缘服务。
日志设置	展示边缘节点的日志设置。日志存储时间、系统日志设置和应用日志设置。同时支持重新编辑日志设置。

参数名称	说明
监控信息	展示边缘节点的监控信息。例如CPU、内存、磁盘。如果节点有NPU/GPU，可在NPU/GPU图标右上角下拉框中，选择对应的NPU或GPU。信息采集周期为5分钟。

8.4.3 资源池

创建资源池之前，需要先激活ModelArts边缘节点或者纳管IEF边缘节点。边缘资源池创建完成后，您可以对资源池进行修改和删除操作，同时支持查看资源池详情信息。

创建边缘资源池

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“边缘资源池”。
2. 单击“ 资源池”页签，进入资源池列表页。
3. 单击“创建”，进入创建边缘资源池页面，参见下表填写参数。

表 8-34 资源池信息参数

参数名称	说明
名称	边缘资源池名称。
描述	边缘资源池简要描述。
边缘节点类型	选择边缘节点的类型。目前支持ModelArts边缘节点和IEF边缘节点。 ● ModelArts边缘节点：即创建边缘节点创建的边缘节点。 – 主控节点：资源池中的控制节点，负责整个资源池的管理和控制。最多可添加3个主控节点。 – 最大工作节点数限制：资源池最多容纳的工作节点数。取值范围为1-64。 – 工作节点：资源池中的机器节点，运行由主控节点分配的工作。 **说明** 主控节点和工作节点不能重合。 ● IEF边缘节点：即被IEF纳管的边缘节点。 – 资源实例：选择铂金版服务实例。 – 边缘节点：选择边缘节点设备，用于运行边缘应用，处理您的数据，并安全、便捷地和云端应用进行协同。

4. 确认配置无误后，单击“确认”，开始创建边缘资源池。

修改边缘资源池

1. 在“边缘资源池 > 资源池”页面，单击目标资源池操作列的“修改”，进入修改边缘资源池界面，对资源池进行修改操作。参数说明如下：

表 8-35 修改边缘资源池参数说明

参数名称	说明
名称	资源池的名称，不支持修改。
描述	资源池的简要描述。
边缘节点类型	边缘节点类型不支持修改。 当资源池的节点是ModelArts边缘节点，配置参数如下： ● 主控节点：资源池中的控制节点，负责整个资源池的管理和控制。不支持修改。 ● 最大工作节点数：边缘资源池最大工作节点数。取值范围为1-64。不支持修改。 ● 工作节点：可以为资源池添加或删除边缘节点。 **说明** 主控节点和工作节点不能重合。 当资源池的节点是IEF边缘节点，可修改配置参数如下： ● 资源池实例类型：可选择专业版服务实例和一体机铂金实例类型。 ● 边缘节点：边缘节点是您自己的边缘计算设备，用于运行边缘应用，处理您的数据，并安全、便捷地和云端应用进行协同。

2. 单击“立即修改”，完成边缘资源池的修改。

删除边缘资源池

在“边缘资源池 > 资源池”页面，单击目标资源池操作列的“删除”，删除边缘资源池。删除边缘资源池前，需先删除资源池关联的边缘服务、负载均衡和访问端口。边缘资源池删除后，无法恢复，请谨慎操作。

查看边缘资源池详情

在“边缘资源池 > 资源池”页面，单击目标资源池的名称，进入边缘资源池详情页面，查看资源池详情信息。

表 8-36 基本信息参数说明

参数名称	说明
名称	资源池的名称。
创建时间	资源池的创建时间。
更新时间	资源池最后更新时间。
描述	资源池的描述。
状态	资源池的状态。

参数名称	说明
ID	资源池的ID。使用ModelArts边缘节点创建的边缘资源池显示。
资源实例ID	资源实例的ID。使用IEF边缘节点创建的边缘资源池显示。
边缘节点类型	选择资源池的边缘节点类型，为ModelArts边缘节点或IEF边缘节点。

您可以在资源池详情页面，通过切换页签查看更多详细信息。如下：

表 8-37 资源池详情

参数名称	说明
边缘服务	展示资源池关联的边缘服务。支持创建、修改、删除边缘服务操作。
节点	展示资源池绑定的节点信息。支持添加和删除节点操作。

8.4.4 开启 LTS 日志

边缘节点支持开启LTS日志，开启日志需要通过配置，本文主要介绍了如何开启LTS日志。

步骤1 创建节点。

1. 左侧菜单栏选择“边缘资源池”，进入“节点”页签。
2. 单击“创建”，创建边缘节点。填写“基本信息”，在“日志设置”中开启“日志上传”。单击“确定”，下载证书和固件。
3. 完成后，单击“确定”，完成节点创建。

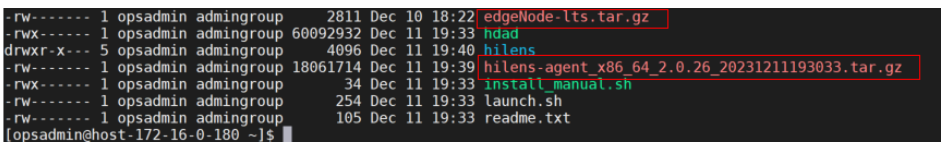
步骤2 登录边缘节点，上传设备证书和固件。

1. 使用Putty工具登录虚拟机。

   ssh *<用户名>*@*<虚拟机IP>*

   - ***用户名***：登录服务器的用户名。
   - ***虚拟机IP***：可在云服务器控制台中，在云服务列表中选择任一云服务，并复制其弹性IP。
2. 通过工具，上传设备证书和固件。

   图 8-35 上传证书和固件



步骤3 安装边缘agent，并注册绑定节点。

1. 解压agent固件包。
   ```
   tar –xvf hilens-agent_x86_64_2.0.26_20231211193033.tar.gz
   ```
   其中，*hilens-agent_x86_64_2.0.26_20231211193033.tar.gz*为固件包，实际使用时，需替换为自己的固件包。
2. 安装agent。
   ```
   sh install_manual.sh
   ```
3. 修改agent配置。
   ```
   vi /etc/hilens/hda.conf
   ```
   添加配置如下：
   ```
   hilens.lts.upload.url=https://8.28.30.68:8102
   hilens.request.ctx.lts=v2
   hilens.lts.url=https://lts.ei-a3-1.externala3.com
   ```
   - hilens.lts.upload.url的IP获取方式为：登录云日志服务控制台，在左侧菜单栏选择“主机管理”，单击页面右上角“安装ICAgent”，在弹出页面的“复制ICAgent安装命令”中获取“https://”后的IP，即为hilens.lts.upload.url的IP。
   - hilens.lts.url的值拼接规则为：https://lts.{*region*}.{*external_global_domain_name*}。其中*external_global_domain_name*可从LLD“基本参数”页签搜索“*external_global_domain_name*”，获取其对应的参数值。
4. 重启agent。
   ```
   systemctl restart hdad
   ```
5. 注册并绑定边缘节点。
   ```
   hdactl bind –p edgeNode-lts.tar.gz
   ```

步骤4 激活边缘节点。

1. 登录ModelArts控制台，左侧菜单栏选择“边缘资源池”。
2. 在“节点”页签找到已绑定的节点，选择“操作 > 激活”，激活节点。

步骤5 登录LTS控制台，即可看到上传的日志。

在LTS控制台左侧菜单栏单击“日志管理”，在“日志组列表”中选择带有节点名称的日志组，日志流选择节点ID对应的日志流。

----结束

8.5 推理规范说明

8.5.1 模型包规范

8.5.1.1 模型包规范介绍

创建AI应用时，如果是从OBS中导入元模型，则需要符合一定的模型包规范。

说明

- 模型包规范适用于单模型场景，若是多模型场景（例如含有多个模型文件）推荐使用自定义镜像方式。
- ModelArts推理平台不支持的AI引擎，推荐使用自定义镜像方式。
- 请参考**创建AI应用的自定义镜像规范**和**从0-1制作自定义镜像并创建AI应用**，制作自定义镜像。
- 更多的自定义脚本代码示例，请参考**自定义脚本代码示例**。

模型包里面必需包含“model”文件夹，“model”文件夹下面放置模型文件，模型配置文件，模型推理代码文件。

- **模型文件**：在不同模型包结构中模型文件的要求不同，具体请参见**模型包结构示例**。
- **模型配置文件**：模型配置文件必需存在，文件名固定为“config.json”，有且只有一个，模型配置文件编写请参见**模型配置文件编写说明**。
- **模型推理代码文件**：模型推理代码文件是必选的。文件名固定为“customize_service.py”，此文件有且只能有一个，模型推理代码编写请参见**模型推理代码编写说明**。
  - customize_service.py依赖的py文件可以直接放model目录下，推荐采用相对导入方式导入自定义包。
  - customize_service.py依赖的其他文件可以直接放model目录下，需要采用绝对路径方式访问。绝对路径获取请参考**绝对路径如何获取**。

ModelArts也提供了常用AI引擎对应的自定义脚本示例，请参见**自定义脚本代码示例**。

若您在导入元模型过程中遇到问题，可协助解决故障。

模型包结构示例

- TensorFlow模型包结构

  发布该模型时只需要指定到“ocr”目录。

```
OBS桶/目录名
├── ocr
│   ├── model 必选： 固定子目录名称，用于放置模型相关文件
│   │   ├── <<自定义python包>> 可选：用户自有的Python包，在模型推理代码中可以直接引用
│   │   ├── saved_model.pb 必选: protocol buffer格式文件，包含该模型的图描述
│   │   ├── variables 对*.pb模型主文件而言必选；固定子目录名称，包含模型的权重偏差等信息
│   │   │   ├── variables.index 必选
│   │   │   ├── variables.data-00000-of-00001 必选
│   │   ├──config.json 必选：模型配置文件，文件名称固定为config.json, 只允许放置一个
│   │   ├──customize_service.py 必选：模型推理代码，文件名称固定为customize_service.py, 只允许放置一个
customize_service.py依赖的文件可以直接放model目录下
```

- MindSpore模型包结构

```
OBS桶/目录名
├── resnet
│   ├── model 必选： 固定子目录名称，用于放置模型相关文件
│   │   ├── <<自定义python包>> 可选：用户自有的Python包，在模型推理代码中可以直接引用
│   │   ├── checkpoint_lenet_1-1_1875.ckpt 必选；使用mindspore训练出来的ckpt格式/om格式的模型文件
│   │   ├── config.json 必选：模型配置文件，文件名称固定为config.json, 只允许放置一个
│   │   ├── customize_service.py 必选：模型推理代码，文件名称固定为customize_service.py, 只允许放置一个，customize_service.py依赖的文件可以直接放model目录下
│   │   ├── tmp.om 必选；空的.om文件，保证模型包可以正常导入
```

- PyTorch模型包结构

  发布该模型时只需要指定到“resnet”目录。

```
OBS桶/目录名
├── resnet
|   ├── model  必选： 固定子目录名称，用于放置模型相关文件
|   |   ├── <<自定义Python包>>  可选：用户自有的Python包，在模型推理代码中可以直接引用
|   |   ├── resnet50.pth 必选，pytorch模型保存文件，保存为“state_dict”，存有权重变量等信息。
|   |   ├──config.json 必选：模型配置文件，文件名称固定为config.json, 只允许放置一个
|   |   ├──customize_service.py  必选：模型推理代码，文件名称固定为customize_service.py, 只允许放置一个，customize_service.py依赖的文件可以直接放model目录下
```

- Custom模型包结构，与您自定义镜像中AI引擎有关。例如自定义镜像中的AI引擎为TensorFlow，则模型包采用TensorFlow模型包结构。

8.5.1.2 模型配置文件编写说明

模型开发者发布模型时需要编写配置文件config.json。模型配置文件描述模型用途、模型计算框架、模型精度、推理代码依赖包以及模型对外API接口。

配置文件格式说明

配置文件为JSON格式，参数说明如**表8-38**所示。

表 8-38 参数说明

参数	是否必选	参数类型	描述
model_algorithm	是	String	模型算法，表示该模型的用途，由模型开发者填写，以便使用者理解该模型的用途。只能以英文字母开头，不能包含中文以及&!'\"<>=，不超过36个字符。常见的模型算法有image_classification（图像分类）、object_detection（物体检测）、predict_analysis（预测分析）等。
model_type	是	String	模型AI引擎，表明模型使用的计算框架，支持常用AI框架和“Image”。 ● 可选的常用AI框架请参见**推理支持的AI引擎**。 ● 当model_type设置为Image，表示以自定义镜像方式创建AI应用，此时swr_location为必填参数。Image镜像制作规范可参见**创建AI应用的自定义镜像规范**。
runtime	否	String	模型运行时环境，系统默认使用python3.6。runtime可选值与model_type相关，当model_type设置为Image时，不需要设置runtime，当model_type设置为其他常用框架时，请选择您使用的引擎所对应的运行时环境。
metrics	否	object数据结构	模型的精度信息，包括平均数、召回率、精确率、准确率，metrics object数据结构说明如**表8-39**所示。 结果会显示在AI应用详情页面的“模型精度”模块。

参数	是否必选	参数类型	描述
apis	否	api数据结构数组	表示模型接收和返回的请求样式，为结构体数据。 即模型可对外提供的Restful API数组，API数据结构如表8-40所示。示例代码请参见apis参数代码示例。 • “model_type”为“Image”时，即自定义镜像的模型场景，“apis”可根据镜像实际对外暴露的请求路径在“apis”中声明不同路径的API。 • “model_type”不为“Image”时，“apis”只能声明一个请求路径为“/”的API，因为系统预置的AI引擎仅暴露一个请求路径为“/”的推理接口。
dependencies	否	dependency结构数组	表示模型推理代码需要依赖的包，为结构体数据。 模型开发者需要提供包名、安装方式、版本约束。目前只支持pip安装方式。dependency结构数组说明如表8-43所示。 如果模型包内没有推理代码customize_service.py文件，则该字段可不填。自定义镜像模型不支持安装依赖包。
health	否	health数据结构	镜像健康接口配置信息，只有“model_type”为“Image”时才需填写。 如果在滚动升级时要求不中断业务，那么必需提供健康检查的接口供ModelArts调用。health数据结构如表8-45所示。

表 8-39 metrics object 数据结构说明

参数	是否必选	参数类型	描述
f1	否	Number	平均数。精确到小数点后17位，超过17位时，取前17位数值。
recall	否	Number	召回率。精确到小数点后17位，超过17位时，取前17位数值。
precision	否	Number	精确率。精确到小数点后17位，超过17位时，取前17位数值。
accuracy	否	Number	准确率。精确到小数点后17位，超过17位时，取前17位数值。

表 8-40 api 数据结构说明

参数	是否必选	参数类型	描述
url	否	String	请求路径。默认值为“/”。自定义镜像的模型（即model_type为Image时）需要根据镜像内实际暴露的请求路径填写“url”。非自定义镜像模型（即model_type不为Image时）时，“url”只能为“/”。
method	否	String	请求方法。默认值为“POST”。
request	否	Object	请求体，request结构说明如**表8-41**所示。
response	否	Object	响应体，response结构说明如**表8-42**所示。

表 8-41 request 结构说明

参数	是否必选	参数类型	描述
Content-type	在线服务-非必选 批量服务-必选	String	data以指定内容类型发送。默认值为“application/json”。 一般情况包括如下两种内容类型： ● “application/json”，发送json数据。 ● “multipart/form-data”，上传文件。 **说明** 针对机器学习类模型，仅支持“application/json”
data	在线服务-非必选 批量服务-必选	String	请求体以json schema描述。参数说明请参考**官方指导**。

表 8-42 response 结构说明

参数	是否必选	参数类型	描述
Content-type	在线服务-非必选 批量服务-必选	String	data以指定内容类型发送。默认值为“application/json”。 **说明** 针对机器学习类模型，仅支持“application/json”

参数	是否必选	参数类型	描述
data	在线服务-非必选 批量服务-必选	String	响应体以json schema描述。参数说明请参考官方指导。

表 8-43 dependency 结构数组说明

参数	是否必选	参数类型	描述
installer	是	String	安装方式，当前只支持“pip”。
packages	是	package结构数组	依赖包集合，package结构数组说明如表8-44所示。

表 8-44 package 结构数组说明

参数	是否必选	参数类型	描述
package_name	是	String	依赖包名称。不能含有中文及特殊字符&!'"<>=。
package_version	否	String	依赖包版本，如果不强依赖于版本号，则该项不填。不能含有中文及特殊字符&!'"<>=。
restraint	否	String	版本限制条件，当且仅当“package_version”存在时必填，可选“EXACT/ATLEAST/ATMOST”。 • “EXACT”表示安装给定版本。 • “ATLEAST”表示安装版本不小于给定版本。 • “ATMOST”表示安装包版本不大于给定版本。 **说明** • 如果对版本有明确要求，优先使用“EXACT”；如果使用“EXACT”与系统安装包有冲突，可以选择“ATLEAST” • 如果对版本没有明确要求，推荐不填写“restraint”、“package_version”，只保留“package_name”参数

表 8-45 health 数据结构说明

参数	是否必选	参数类型	描述
check_method	是	String	健康检查方式。可选“HTTP/EXEC”。 ● HTTP：HTTP请求检查 ● EXEC：执行命令检查。
command	否	String	健康检查命令。健康检查方式为EXEC时必选。
url	否	String	健康检查接口请求路径。健康检查方式为HTTP时必选。
protocol	否	String	健康检查接口请求协议，默认为http。健康检查方式为HTTP时必选。
initial_delay_seconds	否	String	健康检查初始化延迟时间。
timeout_seconds	否	String	健康检查超时时间。
period_seconds	是	String	健康检查周期。填写大于0且小于等于2147483647的整数，单位为秒。
failure_threshold	是	String	健康检查最大失败次数。填写大于0且小于等于2147483647的整数。

apis 参数代码示例

```
[{
    "url": "/",
    "method": "post",
    "request": {
        "Content-type": "multipart/form-data",
        "data": {
            "type": "object",
            "properties": {
                "images": {
                    "type": "file"
                }
            }
        }
    },
    "response": {
        "Content-type": "applicaton/json",
        "data": {
            "type": "object",
            "properties": {
                "mnist_result": {
                    "type": "array",
                    "item": [
                        {
                            "type": "string"
                        }
```

```
            ]
          }
        }
      }
    }
  }
}]
```

目标检测模型配置文件示例

如下代码以TensorFlow引擎为例，您可以根据实际使用的引擎类型修改model_type参数后使用。

- 模型输入

  key：images

  value：图片文件

- 模型输出

```
{
  "detection_classes": [
    "face",
    "arm"
  ],
  "detection_boxes": [
    [
      33.6,
      42.6,
      104.5,
      203.4
    ],
    [
      103.1,
      92.8,
      765.6,
      945.7
    ]
  ],
  "detection_scores": [0.99, 0.73]
}
```

- 配置文件

```
{
  "model_type": "TensorFlow",
  "model_algorithm": "object_detection",
  "metrics": {
    "f1": 0.345294,
    "accuracy": 0.462963,
    "precision": 0.338977,
    "recall": 0.351852
  },
  "apis": [{
    "url": "/",
    "method": "post",
    "request": {
      "Content-type": "multipart/form-data",
      "data": {
        "type": "object",
        "properties": {
          "images": {
            "type": "file"
          }
        }
      }
    },
    "response": {
      "Content-type": "application/json",
      "data": {
        "type": "object",
```

```
            "properties": {
                "detection_classes": {
                    "type": "array",
                    "items": [{
                        "type": "string"
                    }]
                },
                "detection_boxes": {
                    "type": "array",
                    "items": [{
                        "type": "array",
                        "minItems": 4,
                        "maxItems": 4,
                        "items": [{
                            "type": "number"
                        }]
                    }]
                },
                "detection_scores": {
                    "type": "array",
                    "items": [{
                        "type": "number"
                    }]
                }
            }
        }
    }
  }],
  "dependencies": [{
      "installer": "pip",
      "packages": [{
            "restraint": "EXACT",
            "package_version": "1.15.0",
            "package_name": "numpy"
        },
        {
            "restraint": "EXACT",
            "package_version": "5.2.0",
            "package_name": "Pillow"
        }
    ]
  }]
}
```

图像分类模型配置文件示例

如下代码以TensorFlow引擎为例，您可以根据实际使用的引擎类型修改model_type参数后使用。

- 模型输入

  key：images

  value：图片文件

- 模型输出

```
{
  "predicted_label": "flower",
  "scores": [
    ["rose", 0.99],
    ["begonia", 0.01]
  ]
}
```

- 配置文件

```
{
  "model_type": "TensorFlow",
  "model_algorithm": "image_classification",
  "metrics": {
```

```
        "f1": 0.345294,
        "accuracy": 0.462963,
        "precision": 0.338977,
        "recall": 0.351852
    },
    "apis": [{
        "url": "/",
        "method": "post",
        "request": {
            "Content-type": "multipart/form-data",
            "data": {
                "type": "object",
                "properties": {
                    "images": {
                        "type": "file"
                    }
                }
            }
        },
        "response": {
            "Content-type": "application/json",
            "data": {
                "type": "object",
                "properties": {
                    "predicted_label": {
                        "type": "string"
                    },
                    "scores": {
                        "type": "array",
                        "items": [{
                            "type": "array",
                            "minItems": 2,
                            "maxItems": 2,
                            "items": [
                                {
                                    "type": "string"
                                },
                                {
                                    "type": "number"
                                }
                            ]
                        }]
                    }
                }
            }
        }
    }],
    "dependencies": [{
        "installer": "pip",
        "packages": [{
                "restraint": "ATLEAST",
                "package_version": "1.15.0",
                "package_name": "numpy"
            },
            {
                "restraint": "",
                "package_version": "",
                "package_name": "Pillow"
            }
        ]
    }]
}
```

如下代码以MindSpore引擎为例，您可以根据实际使用的引擎类型修改model_type参数后使用。

- 模型输入

  key：images

value：图片文件

- 模型输出

```
"[[-2.404526   -3.0476532  -1.9888215   0.45013925 -1.7018927   0.40332815\n  -7.1861157
11.290332   -1.5861531   5.7887416 ]]"
```

- 配置文件

```
{
    "model_algorithm": "image_classification",
    "model_type": "MindSpore",
    "metrics": {
        "f1": 0.124555,
        "recall": 0.171875,
        "precision": 0.0023493892851938493,
        "accuracy": 0.00746268656716417
    },
    "apis": [{
            "url": "/",
            "method": "post",
            "request": {
                "Content-type": "multipart/form-data",
                "data": {
                    "type": "object",
                    "properties": {
                        "images": {
                            "type": "file"
                        }
                    }
                }
            },
            "response": {
                "Content-type": "applicaton/json",
                "data": {
                    "type": "object",
                    "properties": {
                        "mnist_result": {
                            "type": "array",
                            "item": [{
                                "type": "string"
                            }]
                        }
                    }
                }
            }
        }
    ],
    "dependencies": []
}
```

预测分析模型配置文件示例

如下代码以TensorFlow引擎为例，您可以根据实际使用的引擎类型修改model_type参数后使用。

- 模型输入

```
{
    "data": {
        "req_data": [
            {
                "buying_price": "high",
                "maint_price": "high",
                "doors": "2",
                "persons": "2",
                "lug_boot": "small",
                "safety": "low",
                "acceptability": "acc"
            },
            {
```

```
            "buying_price": "high",
            "maint_price": "high",
            "doors": "2",
            "persons": "2",
            "lug_boot": "small",
            "safety": "low",
            "acceptability": "acc"
        }
    ]
  }
}
```

- 模型输出

```
{
  "data": {
    "resp_data": [
      {
        "predict_result": "unacc"
      },
      {
        "predict_result": "unacc"
      }
    ]
  }
}
```

- 配置文件

  说明

  代码中request结构和response结构中的data参数是json schema数据结构。data/properties里面的内容对应“模型输入”和“模型输出”。

```
{
  "model_type": "TensorFlow",
  "model_algorithm": "predict_analysis",
  "metrics": {
    "f1": 0.345294,
    "accuracy": 0.462963,
    "precision": 0.338977,
    "recall": 0.351852
  },
  "apis": [
    {
      "url": "/",
      "method": "post",
      "request": {
        "Content-type": "application/json",
        "data": {
          "type": "object",
          "properties": {
            "data": {
              "type": "object",
              "properties": {
                "req_data": {
                  "items": [
                    {
                      "type": "object",
                      "properties": {}
                    }
                  ],
                  "type": "array"
                }
              }
            }
          }
        }
      },
      "response": {
        "Content-type": "application/json",
        "data": {
```

```
                "type": "object",
                "properties": {
                    "data": {
                        "type": "object",
                        "properties": {
                            "resp_data": {
                                "type": "array",
                                "items": [
                                    {
                                        "type": "object",
                                        "properties": {}
                                    }
                                ]
                            }
                        }
                    }
                }
            }
        }
    ],
    "dependencies": [
        {
            "installer": "pip",
            "packages": [
                {
                    "restraint": "EXACT",
                    "package_version": "1.15.0",
                    "package_name": "numpy"
                },
                {
                    "restraint": "EXACT",
                    "package_version": "5.2.0",
                    "package_name": "Pillow"
                }
            ]
        }
    ]
}
```

自定义镜像类型的模型配置文件示例

模型输入和输出与**目标检测模型配置文件示例**类似。

- 模型预测输入为**图片类型**时，request请求示例如下：

  该实例表示模型预测接收一个参数名为images、参数类型为file的预测请求，在推理界面会显示文件上传按钮，以文件形式进行预测。

  ```
  {
      "Content-type": "multipart/form-data",
      "data": {
          "type": "object",
          "properties": {
              "images": {
                  "type": "file"
              }
          }
      }
  }
  ```

- 模型预测输入为**json数据类型**时，request请求示例如下：

  该实例表示模型预测接收json请求体，只有一个参数名为input、参数类型为string的预测请求，在推理界面会显示文本输入框，用于填写预测请求。

  ```
  {
      "Content-type": "application/json",
      "data": {
          "type": "object",
  ```

```
        "properties": {
            "input": {
                "type": "string"
            }
        }
    }
}
```

完整请求示例如下：

```
{
    "model_algorithm": "image_classification",
    "model_type": "Image",
    "metrics": {
        "f1": 0.345294,
        "accuracy": 0.462963,
        "precision": 0.338977,
        "recall": 0.351852
    },
    "apis": [{
        "url": "/",
        "method": "post",
        "request": {
            "Content-type": "multipart/form-data",
            "data": {
                "type": "object",
                "properties": {
                    "images": {
                        "type": "file"
                    }
                }
            }
        },
        "response": {
            "Content-type": "application/json",
            "data": {
                "type": "object",
                "required": [
                    "predicted_label",
                    "scores"
                ],
                "properties": {
                    "predicted_label": {
                        "type": "string"
                    },
                    "scores": {
                        "type": "array",
                        "items": [{
                            "type": "array",
                            "minItems": 2,
                            "maxItems": 2,
                            "items": [{
                                    "type": "string"
                                },
                                {
                                    "type": "number"
                                }
                            ]
                        }]
                    }
                }
            }
        }
    }]
}
```

机器学习类型的模型配置文件示例

以下代码以XGBoost为例。

- 模型输入：

```
{
    "req_data": [
        {
            "sepal_length": 5,
            "sepal_width": 3.3,
            "petal_length": 1.4,
            "petal_width": 0.2
        },
        {
            "sepal_length": 5,
            "sepal_width": 2,
            "petal_length": 3.5,
            "petal_width": 1
        },
        {
            "sepal_length": 6,
            "sepal_width": 2.2,
            "petal_length": 5,
            "petal_width": 1.5
        }
    ]
}
```

- 模型输出：

```
{
    "resp_data": [
        {
            "predict_result": "Iris-setosa"
        },
        {
            "predict_result": "Iris-versicolor"
        }
    ]
}
```

- 配置文件：

```
{
    "model_type": "XGBoost",
    "model_algorithm": "xgboost_iris_test",
    "runtime": "python2.7",
    "metrics": {
        "f1": 0.345294,
        "accuracy": 0.462963,
        "precision": 0.338977,
        "recall": 0.351852
    },
    "apis": [
        {
            "url": "/",
            "method": "post",
            "request": {
                "Content-type": "application/json",
                "data": {
                    "type": "object",
                    "properties": {
                        "req_data": {
                            "items": [
                                {
                                    "type": "object",
                                    "properties": {}
                                }
                            ],
                            "type": "array"
                        }
                    }
                }
            },
            "response": {
```

```
            "Content-type": "applicaton/json",
            "data": {
                "type": "object",
                "properties": {
                    "resp_data": {
                        "type": "array",
                        "items": [
                            {
                                "type": "object",
                                "properties": {
                                    "predict_result": {}
                                }
                            }
                        ]
                    }
                }
            }
        }
    }
  ]
}
```

使用自定义依赖包的模型配置文件示例

如下示例中，定义了1.16.4版本的numpy的依赖环境。

```
{
  "model_algorithm": "image_classification",
  "model_type": "TensorFlow",
  "runtime": "python3.6",
  "apis": [
    {
        "url": "/",
        "method": "post",
        "request": {
            "Content-type": "multipart/form-data",
            "data": {
                "type": "object",
                "properties": {
                    "images": {
                        "type": "file"
                    }
                }
            }
        },
        "response": {
            "Content-type": "applicaton/json",
            "data": {
                "type": "object",
                "properties": {
                    "mnist_result": {
                        "type": "array",
                        "item": [
                            {
                                "type": "string"
                            }
                        ]
                    }
                }
            }
        }
    }
  ],
  "metrics": {
    "f1": 0.124555,
    "recall": 0.171875,
    "precision": 0.00234938928519385,
    "accuracy": 0.00746268656716417
```

```
  },
  "dependencies": [
    {
      "installer": "pip",
      "packages": [
        {
          "restraint": "EXACT",
          "package_version": "1.16.4",
          "package_name": "numpy"
        }
      ]
    }
  ]
}
```

8.5.1.3 模型推理代码编写说明

本章节介绍了在ModelArts中模型推理代码编写的通用方法及说明，本文在编写说明下方提供了一个TensorFlow引擎的推理代码示例以及一个在推理脚本中自定义推理逻辑的示例。

ModelArts推理因API网关（APIG）的限制，模型单次预测的时间不能超过40S，模型推理代码编写需逻辑清晰，代码简洁，以此达到更好的推理效果。

推理代码编写指导

1. 在模型代码推理文件“customize_service.py”中，需要添加一个子类，该子类继承对应模型类型的父类，各模型类型的父类名称和导入语句如表8-46所示。导入语句所涉及的Python包在ModelArts环境中已配置，用户无需自行安装。

表 8-46 各模型类型的父类名称和导入语句

模型类型	父类	导入语句
TensorFlow	TfServingBaseService	from model_service.tfserving_model_service import TfServingBaseService
PyTorch	PTServingBaseService	from model_service.pytorch_model_service import PTServingBaseService
MindSpore	SingleNodeService	from model_service.model_service import SingleNodeService

2. 可以重写的方法有以下几种。

表 8-47 重写方法

方法名	说明
__init__(self, model_name, model_path)	初始化方法，适用于深度学习框架模型。该方法内加载模型及标签等（pytorch和caffe类型模型必须重写，实现模型加载逻辑）。
__init__(self, model_path)	初始化方法，适用于机器学习框架模型。该方法内初始化模型的路径（self.model_path）。在Spark_MLlib中，该方法还会初始化SparkSession（self.spark）。

方法名	说明
_preprocess(self, data)	预处理方法，在推理请求前调用，用于将API接口输入的用户原始请求数据转换为模型期望输入数据。
_inference(self, data)	实际推理请求方法（不建议重写，重写后会覆盖ModelArts内置的推理过程，运行自定义的推理逻辑）。
_postprocess(self, data)	后处理方法，在推理请求完成后调用，用于将模型输出转换为API接口输出。

说明

- 用户可以选择重写preprocess和postprocess方法，以实现API输入数据的预处理和推理输出结果的后处理。
- 重写模型父类的初始化方法init可能导致AI应用“运行异常”。

3. 可以使用的属性为模型所在的本地路径，属性名为“self.model_path”。另外pyspark模型在“customize_service.py”中可以使用“self.spark”获取SparkSession对象。

   说明

   推理代码中，需要通过绝对路径读取文件。模型所在的本地路径可以通过self.model_path属性获得。

   - 当使用TensorFlow、Caffe、MXNet时，self.model_path为模型文件目录路径，读取文件示例如下：

     ```
     # model目录下放置label.json文件，此处读取
     with open(os.path.join(self.model_path, 'label.json')) as f:
         self.label = json.load(f)
     ```

   - 当使用PyTorch、Scikit_Learn、pyspark时，self.model_path为模型文件路径，读取文件示例如下：

     ```
     # model目录下放置label.json文件，此处读取
     dir_path = os.path.dirname(os.path.realpath(self.model_path))
     with open(os.path.join(dir_path, 'label.json')) as f:
         self.label = json.load(f)
     ```

4. 预处理方法、实际推理请求方法和后处理方法中的接口传入“data”当前支持两种content-type，即“multipart/form-data”和“application/json”。

   - “multipart/form-data”请求

     ```
     curl -X POST \
       <modelarts-inference-endpoint> \
       -F image1=@cat.jpg \
       -F images2=@horse.jpg
     ```

     对应的传入data为

     ```
     [
       {
         "image1":{
           "cat.jpg":"<cat.jpg file io>"
         }
       },
       {
         "image2":{
           "horse.jpg":"<horse.jpg file io>"
         }
       }
     ]
     ```

- “application/json”请求

```
curl -X POST \
  <modelarts-inference-endpoint> \
  -d '{
   "images":"base64 encode image"
   }'
```

对应的传入data为python dict

```
{
   "images":"base64 encode image"
}
```

TensorFlow 的推理脚本示例

TensorFlow MnistService示例如下。

- 推理代码

```
from PIL import Image
import numpy as np
from model_service.tfserving_model_service import TfServingBaseService

class MnistService(TfServingBaseService):

    def _preprocess(self, data):
        preprocessed_data = {}

        for k, v in data.items():
            for file_name, file_content in v.items():
                image1 = Image.open(file_content)
                image1 = np.array(image1, dtype=np.float32)
                image1.resize((1, 784))
                preprocessed_data[k] = image1

        return preprocessed_data

    def _postprocess(self, data):

        infer_output = {}

        for output_name, result in data.items():

            infer_output["mnist_result"] = result[0].index(max(result[0]))

        return infer_output
```

- 请求

```
curl -X POST \ 在线服务地址 \ -F images=@test.jpg
```

- 返回

```
{"mnist_result": 7}
```

在上面的代码示例中，完成了将用户表单输入的图片的大小调整，转换为可以适配模型输入的shape。首先通过Pillow库读取“32×32”的图片，调整图片大小为“1×784”以匹配模型输入。在后续处理中，转换模型输出为列表，用于Restful接口输出展示。

自定义推理逻辑的推理脚本示例

首先，需要在配置文件中，定义自己的依赖包，详细示例请参见**使用自定义依赖包的模型配置文件示例**。然后通过如下示例代码，实现了“saved_model”格式模型的加载推理。

说明

当前推理基础镜像使用的python的logging模块，采用的是默认的日志级别Warnning，即当前只有warning级别的日志可以默认查询出来。如果想要指定INFO等级的日志能够查询出来，需要在代码中指定logging的输出日志等级为INFO级别。

```
# -*- coding: utf-8 -*-
import json
import os
import threading
import numpy as np
import tensorflow as tf
from PIL import Image
from model_service.tfserving_model_service import TfServingBaseService
import logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

class MnistService(TfServingBaseService):
    def __init__(self, model_name, model_path):
        self.model_name = model_name
        self.model_path = model_path
        self.model_inputs = {}
        self.model_outputs = {}

        # label文件可以在这里加载,在后处理函数里使用
        # label.txt放在OBS和模型包的目录

        # with open(os.path.join(self.model_path, 'label.txt')) as f:
        #     self.label = json.load(f)

        # 非阻塞方式加载saved_model模型，防止阻塞超时
        thread = threading.Thread(target=self.get_tf_sess)
        thread.start()

    def get_tf_sess(self):
        # 加载saved_model格式的模型
        # session要重用，建议不要用with语句
        sess = tf.Session(graph=tf.Graph())
        meta_graph_def = tf.saved_model.loader.load(sess, [tf.saved_model.tag_constants.SERVING],
self.model_path)
        signature_defs = meta_graph_def.signature_def
        self.sess = sess
        signature = []

        # only one signature allowed
        for signature_def in signature_defs:
            signature.append(signature_def)
        if len(signature) == 1:
            model_signature = signature[0]
        else:
            logger.warning("signatures more than one, use serving_default signature")
            model_signature = tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY

        logger.info("model signature: %s", model_signature)

        for signature_name in meta_graph_def.signature_def[model_signature].inputs:
            tensorinfo = meta_graph_def.signature_def[model_signature].inputs[signature_name]
            name = tensorinfo.name
            op = self.sess.graph.get_tensor_by_name(name)
            self.model_inputs[signature_name] = op

        logger.info("model inputs: %s", self.model_inputs)

        for signature_name in meta_graph_def.signature_def[model_signature].outputs:
            tensorinfo = meta_graph_def.signature_def[model_signature].outputs[signature_name]
            name = tensorinfo.name
            op = self.sess.graph.get_tensor_by_name(name)
            self.model_outputs[signature_name] = op
```

```
    logger.info("model outputs: %s", self.model_outputs)

def _preprocess(self, data):
    # https两种请求形式
    # 1. form-data文件格式的请求对应：data = {"请求key值":{"文件名":<文件io>}}
    # 2. json格式对应：data = json.loads("接口传入的json体")
    preprocessed_data = {}

    for k, v in data.items():
        for file_name, file_content in v.items():
            image1 = Image.open(file_content)
            image1 = np.array(image1, dtype=np.float32)
            image1.resize((1, 28, 28))
            preprocessed_data[k] = image1

    return preprocessed_data

def _inference(self, data):
    feed_dict = {}
    for k, v in data.items():
        if k not in self.model_inputs.keys():
            logger.error("input key %s is not in model inputs %s", k, list(self.model_inputs.keys()))
            raise Exception("input key %s is not in model inputs %s" % (k, list(self.model_inputs.keys())))
        feed_dict[self.model_inputs[k]] = v

    result = self.sess.run(self.model_outputs, feed_dict=feed_dict)
    logger.info('predict result : ' + str(result))
    return result

def _postprocess(self, data):
    infer_output = {"mnist_result": []}
    for output_name, results in data.items():

        for result in results:
            infer_output["mnist_result"].append(np.argmax(result))

    return infer_output

def __del__(self):
    self.sess.close()
```

说明

对于ModelArts不支持的结构模型或者多模型加载，需要__init__方法中自己指定模型加载的路径。示例代码如下：

```
# -*- coding: utf-8 -*-
import os
from model_service.tfserving_model_service import TfServingBaseService

class MnistService(TfServingBaseService):
    def __init__(self, model_name, model_path):
        # 获取程序当前运行路径，即model文件夹所在的路径
        root = os.path.dirname(os.path.abspath(__file__))
        # test.onnx为待加载模型文件的名称，需要放在model文件夹下
        self.model_path = os.path.join(root, test.onnx)

        # 多模型加载，例如：test2.onnx
        # self.model_path2 = os.path.join(root, test2.onnx)
```

MindSpore 的推理脚本示例

- Snt9芯片推理脚本如下：

```
import threading

import mindspore
import mindspore.nn as nn
import numpy as np
```

```
import logging
from mindspore import Tensor, context
from mindspore.common.initializer import Normal
from mindspore.train.serialization import load_checkpoint, load_param_into_net
from model_service.model_service import SingleNodeService
from PIL import Image

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)


context.set_context(mode=context.GRAPH_MODE, device_target="Ascend")


class LeNet5(nn.Cell):
    """Lenet network structure."""

    # define the operator required
    def __init__(self, num_class=10, num_channel=1):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Conv2d(num_channel, 6, 5, pad_mode='valid')
        self.conv2 = nn.Conv2d(6, 16, 5, pad_mode='valid')
        self.fc1 = nn.Dense(16 * 5 * 5, 120, weight_init=Normal(0.02))
        self.fc2 = nn.Dense(120, 84, weight_init=Normal(0.02))
        self.fc3 = nn.Dense(84, num_class, weight_init=Normal(0.02))
        self.relu = nn.ReLU()
        self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()

    # use the preceding operators to construct networks
    def construct(self, x):
        x = self.max_pool2d(self.relu(self.conv1(x)))
        x = self.max_pool2d(self.relu(self.conv2(x)))
        x = self.flatten(x)
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
        return x


class MnistService(SingleNodeService):
    def __init__(self, model_name, model_path):
        self.model_name = model_name
        self.model_path = model_path
        logger.info("self.model_name:%s self.model_path: %s", self.model_name,
                    self.model_path)
        self.network = None
        # 非阻塞方式加载模型，防止阻塞超时
        thread = threading.Thread(target=self.load_model)
        thread.start()

    def load_model(self):
        logger.info("load network ... \n")
        self.network = LeNet5()
        ckpt_file = self.model_path + "/checkpoint_lenet_1-1_1875.ckpt"
        logger.info("ckpt_file: %s", ckpt_file)
        param_dict = load_checkpoint(ckpt_file)
        load_param_into_net(self.network, param_dict)
        # 模型预热，否则首次推理的时间会很长
        self.network_warmup()
        logger.info("load network successfully ! \n")

    def network_warmup(self):
        # 模型预热，否则首次推理的时间会很长
        logger.info("warmup network ... \n")
        images = np.array(np.random.randn(1, 1, 32, 32), dtype=np.float32)
        inputs = Tensor(images, mindspore.float32)
        inference_result = self.network(inputs)
        logger.info("warmup network successfully ! \n")
```

```
def _preprocess(self, input_data):
    preprocessed_result = {}
    images = []
    for k, v in input_data.items():
        for file_name, file_content in v.items():
            image1 = Image.open(file_content)
            image1 = image1.resize((1, 32 * 32))
            image1 = np.array(image1, dtype=np.float32)
            images.append(image1)

    images = np.array(images, dtype=np.float32)
    logger.info(images.shape)
    images.resize([len(input_data), 1, 32, 32])
    logger.info("images shape: %s", images.shape)
    inputs = Tensor(images, mindspore.float32)
    preprocessed_result['images'] = inputs

    return preprocessed_result

def _inference(self, preprocessed_result):
    inference_result = self.network(preprocessed_result['images'])
    return inference_result

def _postprocess(self, inference_result):
    return str(inference_result)
```

8.5.2 自定义脚本代码示例

8.5.2.1 TensorFlow

TensorFlow存在两种接口类型，keras接口和tf接口，其训练和保存模型的代码存在差异，但是推理代码编写方式一致。

训练模型（keras 接口）

```
from keras.models import Sequential
model = Sequential()
from keras.layers import Dense
import tensorflow as tf

# 导入训练数据集
mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

print(x_train.shape)

from keras.layers import Dense
from keras.models import Sequential
import keras
from keras.layers import Dense, Activation, Flatten, Dropout

# 定义模型网络
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(units=5120,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(units=10, activation='softmax'))

# 定义优化器，损失函数等
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.summary()
# 训练
model.fit(x_train, y_train, epochs=2)
# 评估
model.evaluate(x_test, y_test)
```

保存模型（keras 接口）

```
from keras import backend as K

# K.get_session().run(tf.global_variables_initializer())

# 定义预测接口的inputs和outputs
# inputs和outputs字典的key值会作为模型输入输出tensor的索引键
# 模型输入输出定义需要和推理自定义脚本相匹配
predict_signature = tf.saved_model.signature_def_utils.predict_signature_def(
    inputs={"images" : model.input},
    outputs={"scores" : model.output}
)

# 定义保存路径
builder = tf.saved_model.builder.SavedModelBuilder('./mnist_keras/')

builder.add_meta_graph_and_variables(

    sess = K.get_session(),
    # 推理部署需要定义tf.saved_model.tag_constants.SERVING标签
    tags=[tf.saved_model.tag_constants.SERVING],
    """
    signature_def_map：items只能有一个，或者需要定义相应的key为
    tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY
    """
    signature_def_map={
        tf.saved_model.signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY:
            predict_signature
    }

)
builder.save()
```

训练模型（tf 接口）

```
from __future__ import print_function

import gzip
import os
import urllib

import numpy
import tensorflow as tf
from six.moves import urllib

# 训练数据来源于yann lecun官方网站http://yann.lecun.com/exdb/mnist/
SOURCE_URL = 'http://yann.lecun.com/exdb/mnist/'
TRAIN_IMAGES = 'train-images-idx3-ubyte.gz'
TRAIN_LABELS = 'train-labels-idx1-ubyte.gz'
TEST_IMAGES = 't10k-images-idx3-ubyte.gz'
TEST_LABELS = 't10k-labels-idx1-ubyte.gz'
VALIDATION_SIZE = 5000


def maybe_download(filename, work_directory):
    """Download the data from Yann's website, unless it's already here."""
    if not os.path.exists(work_directory):
        os.mkdir(work_directory)
    filepath = os.path.join(work_directory, filename)
    if not os.path.exists(filepath):
        filepath, _ = urllib.request.urlretrieve(SOURCE_URL + filename, filepath)
        statinfo = os.stat(filepath)
```

```
        print('Successfully downloaded %s %d bytes.' % (filename, statinfo.st_size))
    return filepath


def _read32(bytestream):
    dt = numpy.dtype(numpy.uint32).newbyteorder('>')
    return numpy.frombuffer(bytestream.read(4), dtype=dt)[0]


def extract_images(filename):
    """Extract the images into a 4D uint8 numpy array [index, y, x, depth]."""
    print('Extracting %s' % filename)
    with gzip.open(filename) as bytestream:
        magic = _read32(bytestream)
        if magic != 2051:
            raise ValueError(
                'Invalid magic number %d in MNIST image file: %s' %
                (magic, filename))
        num_images = _read32(bytestream)
        rows = _read32(bytestream)
        cols = _read32(bytestream)
        buf = bytestream.read(rows * cols * num_images)
        data = numpy.frombuffer(buf, dtype=numpy.uint8)
        data = data.reshape(num_images, rows, cols, 1)
        return data


def dense_to_one_hot(labels_dense, num_classes=10):
    """Convert class labels from scalars to one-hot vectors."""
    num_labels = labels_dense.shape[0]
    index_offset = numpy.arange(num_labels) * num_classes
    labels_one_hot = numpy.zeros((num_labels, num_classes))
    labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
    return labels_one_hot


def extract_labels(filename, one_hot=False):
    """Extract the labels into a 1D uint8 numpy array [index]."""
    print('Extracting %s' % filename)
    with gzip.open(filename) as bytestream:
        magic = _read32(bytestream)
        if magic != 2049:
            raise ValueError(
                'Invalid magic number %d in MNIST label file: %s' %
                (magic, filename))
        num_items = _read32(bytestream)
        buf = bytestream.read(num_items)
        labels = numpy.frombuffer(buf, dtype=numpy.uint8)
        if one_hot:
            return dense_to_one_hot(labels)
        return labels


class DataSet(object):
    """Class encompassing test, validation and training MNIST data set."""

    def __init__(self, images, labels, fake_data=False, one_hot=False):
        """Construct a DataSet. one_hot arg is used only if fake_data is true."""

        if fake_data:
            self._num_examples = 10000
            self.one_hot = one_hot
        else:
            assert images.shape[0] == labels.shape[0], (
                    'images.shape: %s labels.shape: %s' % (images.shape,
                                                           labels.shape))
            self._num_examples = images.shape[0]

            # Convert shape from [num examples, rows, columns, depth]
```

```
            # to [num examples, rows*columns] (assuming depth == 1)
            assert images.shape[3] == 1
            images = images.reshape(images.shape[0],
                                    images.shape[1] * images.shape[2])
            # Convert from [0, 255] -> [0.0, 1.0].
            images = images.astype(numpy.float32)
            images = numpy.multiply(images, 1.0 / 255.0)
        self._images = images
        self._labels = labels
        self._epochs_completed = 0
        self._index_in_epoch = 0

    @property
    def images(self):
        return self._images

    @property
    def labels(self):
        return self._labels

    @property
    def num_examples(self):
        return self._num_examples

    @property
    def epochs_completed(self):
        return self._epochs_completed

    def next_batch(self, batch_size, fake_data=False):
        """Return the next `batch_size` examples from this data set."""
        if fake_data:
            fake_image = [1] * 784
            if self.one_hot:
                fake_label = [1] + [0] * 9
            else:
                fake_label = 0
            return [fake_image for _ in range(batch_size)], [
                fake_label for _ in range(batch_size)
            ]
        start = self._index_in_epoch
        self._index_in_epoch += batch_size
        if self._index_in_epoch > self._num_examples:
            # Finished epoch
            self._epochs_completed += 1
            # Shuffle the data
            perm = numpy.arange(self._num_examples)
            numpy.random.shuffle(perm)
            self._images = self._images[perm]
            self._labels = self._labels[perm]
            # Start next epoch
            start = 0
            self._index_in_epoch = batch_size
            assert batch_size <= self._num_examples
        end = self._index_in_epoch
        return self._images[start:end], self._labels[start:end]


def read_data_sets(train_dir, fake_data=False, one_hot=False):
    """Return training, validation and testing data sets."""

    class DataSets(object):
        pass

    data_sets = DataSets()

    if fake_data:
        data_sets.train = DataSet([], [], fake_data=True, one_hot=one_hot)
        data_sets.validation = DataSet([], [], fake_data=True, one_hot=one_hot)
        data_sets.test = DataSet([], [], fake_data=True, one_hot=one_hot)
```

```
        return data_sets

    local_file = maybe_download(TRAIN_IMAGES, train_dir)
    train_images = extract_images(local_file)

    local_file = maybe_download(TRAIN_LABELS, train_dir)
    train_labels = extract_labels(local_file, one_hot=one_hot)

    local_file = maybe_download(TEST_IMAGES, train_dir)
    test_images = extract_images(local_file)

    local_file = maybe_download(TEST_LABELS, train_dir)
    test_labels = extract_labels(local_file, one_hot=one_hot)

    validation_images = train_images[:VALIDATION_SIZE]
    validation_labels = train_labels[:VALIDATION_SIZE]
    train_images = train_images[VALIDATION_SIZE:]
    train_labels = train_labels[VALIDATION_SIZE:]

    data_sets.train = DataSet(train_images, train_labels)
    data_sets.validation = DataSet(validation_images, validation_labels)
    data_sets.test = DataSet(test_images, test_labels)
    return data_sets

training_iteration = 1000

modelarts_example_path =  './modelarts-mnist-train-save-deploy-example'

export_path = modelarts_example_path + '/model/'
data_path = './'

print('Training model...')
mnist = read_data_sets(data_path, one_hot=True)
sess = tf.InteractiveSession()
serialized_tf_example = tf.placeholder(tf.string, name='tf_example')
feature_configs = {'x': tf.FixedLenFeature(shape=[784], dtype=tf.float32), }
tf_example = tf.parse_example(serialized_tf_example, feature_configs)
x = tf.identity(tf_example['x'], name='x')  # use tf.identity() to assign name
y_ = tf.placeholder('float', shape=[None, 10])
w = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
sess.run(tf.global_variables_initializer())
y = tf.nn.softmax(tf.matmul(x, w) + b, name='y')
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
values, indices = tf.nn.top_k(y, 10)
table = tf.contrib.lookup.index_to_string_table_from_tensor(
    tf.constant([str(i) for i in range(10)]))
prediction_classes = table.lookup(tf.to_int64(indices))
for _ in range(training_iteration):
    batch = mnist.train.next_batch(50)
    train_step.run(feed_dict={x: batch[0], y_: batch[1]})
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
print('training accuracy %g' % sess.run(
    accuracy, feed_dict={
        x: mnist.test.images,
        y_: mnist.test.labels
    }))
print('Done training!')
```

保存模型（tf 接口）

```
# 导出模型
# 模型需要采用saved_model接口保存
print('Exporting trained model to', export_path)
builder = tf.saved_model.builder.SavedModelBuilder(export_path)

tensor_info_x = tf.saved_model.utils.build_tensor_info(x)
```

```
tensor_info_y = tf.saved_model.utils.build_tensor_info(y)

# 定义预测接口的inputs和outputs
# inputs和outputs字典的key值会作为模型输入输出tensor的索引键
# 模型输入输出定义需要和推理自定义脚本相匹配
prediction_signature = (
    tf.saved_model.signature_def_utils.build_signature_def(
        inputs={'images': tensor_info_x},
        outputs={'scores': tensor_info_y},
        method_name=tf.saved_model.signature_constants.PREDICT_METHOD_NAME))

legacy_init_op = tf.group(tf.tables_initializer(), name='legacy_init_op')
builder.add_meta_graph_and_variables(
    # tag设为serve/tf.saved_model.tag_constants.SERVING
    sess, [tf.saved_model.tag_constants.SERVING],
    signature_def_map={
        'predict_images':
            prediction_signature,
    },
    legacy_init_op=legacy_init_op)

builder.save()

print('Done exporting!')
```

推理代码（keras 接口和 tf 接口）

在模型代码推理文件customize_service.py中，需要添加一个子类，该子类继承对应模型类型的父类，各模型类型的父类名称和导入语句如请参考**表8-46**。本案例中调用父类“_inference(self, data)”推理请求方法，因此下文代码中不需要重写方法。

```
from PIL import Image
import numpy as np
from model_service.tfserving_model_service import TfServingBaseService


class MnistService(TfServingBaseService):

    # 预处理中处理用户HTTPS接口输入匹配模型输入
    # 对应上述训练部分的模型输入为{"images":<array>}
    def _preprocess(self, data):

        preprocessed_data = {}
        images = []
        # 对输入数据进行迭代
        for k, v in data.items():
            for file_name, file_content in v.items():
                image1 = Image.open(file_content)
                image1 = np.array(image1, dtype=np.float32)
                image1.resize((1,784))
                images.append(image1)
        # 返回numpy array
        images = np.array(images,dtype=np.float32)
        # 对传入的多个样本做batch处理，shape保持和训练时输入一致
        images.resize((len(data), 784))
        preprocessed_data['images'] = images
        return preprocessed_data

    # 对应的上述训练部分保存模型的输出为{"scores":<array>}
    # 后处理中处理模型输出为HTTPS的接口输出
    def _postprocess(self, data):
        infer_output = {"mnist_result": []}
        # 迭代处理模型输出
        for output_name, results in data.items():
            for result in results:
                infer_output["mnist_result"].append(result.index(max(result)))
        return infer_output
```

8.6 云监控平台 ModelArts 监控

8.6.1 ModelArts 支持的监控指标

功能说明

为使用户更好地掌握自己的ModelArts在线服务和对应模型负载的运行状态，云服务平台提供了云监控。您可以使用该服务监控您的ModelArts在线服务和对应模型负载，执行自动实时监控、告警和通知操作，帮助您更好地了解服务和模型的各项性能指标。

命名空间

SYS.ModelArts

监控指标

表 8-48 ModelArts 支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期
cpu_usage	CPU使用率	该指标用于统计ModelArts用户服务的CPU使用率。 单位：百分比。	≥ 0%	ModelArts模型负载	1分钟
mem_usage	内存使用率	该指标用于统计ModelArts用户服务的内存使用率。 单位：百分比。	≥ 0%	ModelArts模型负载	1分钟
gpu_util	GPU使用率	该指标用于统计ModelArts用户服务的GPU使用情况。 单位：百分比。	≥ 0%	ModelArts模型负载	1分钟
gpu_mem_usage	GPU显存使用率	该指标用于统计ModelArts用户服务的GPU显存使用情况。 单位：百分比。	≥ 0%	ModelArts模型负载	1分钟
npu_util	NPU使用率	该指标用于统计ModelArts用户服务的NPU使用情况。 单位：百分比。	≥ 0%	ModelArts模型负载	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期
npu_mem_usage	NPU显存使用率	该指标用于统计ModelArts用户服务的NPU显存使用情况。 单位：百分比。	≥ 0%	ModelArts模型负载	1分钟
successfully_called_times	调用成功次数	统计ModelArts用户调用服务的成功次数。 单位：次/分钟。	≥Count/min	ModelArts模型负载 ModelArts在线服务	1分钟
failed_called_times	调用失败次数	统计ModelArts用户调用服务的失败次数。 单位：次/分钟。	≥Count/min	ModelArts模型负载 ModelArts在线服务	1分钟
total_called_times	调用总次数	统计ModelArts用户调用服务的次数。 单位：次/分钟。	≥Count/min	ModelArts模型负载 ModelArts在线服务	1分钟

对于有多个测量维度的测量对象，使用接口查询监控指标时，所有测量维度均为必选。

- 查询单个监控指标时，多维度dim使用样例：
  dim.0=service_id,530cd6b0-86d7-4818-837f-935f6a27414d&dim.1="model_id,3773b058-5b4f-4366-9035-9bbd9964714a。
- 批量查询监控指标时，多维度dim使用样例：
  "dimensions": [
  {
  "name": "service_id",
  "value": "530cd6b0-86d7-4818-837f-935f6a27414d"
  }
  {
  "name": "model_id",
  "value": "3773b058-5b4f-4366-9035-9bbd9964714a"
  }
  ]

维度

表 8-49 维度说明

Key	Value
service_id	在线服务ID。
model_id	模型负载ID。

8.6.2 设置告警规则

操作场景

通过设置ModelArts在线服务和模型负载告警规则，用户可自定义监控目标与通知策略，及时了解ModelArts在线服务和模型负载状况，从而起到预警作用。

设置ModelArts服务和模型的告警规则包括设置告警规则名称、监控对象、监控指标、告警阈值、监控周期和是否发送通知等参数。本节介绍了设置ModelArts服务和模型告警规则的具体方法。

说明

只有“运行中”的在线服务，支持对接CES监控。

前提条件

- 已创建ModelArts在线服务。
- 已在云监控服务创建ModelArts监控服务。登录“云监控服务”控制台，在“自定义监控”页面，根据界面提示创建ModelArts监控服务。

操作步骤

设置告警规则有多种方式。您可以根据实际应用场景，选择设置告警规则的方式。

- 对ModelArts服务设置告警规则
- 对单个服务设置告警规则
- 对模型版本设置告警规则
- 对服务或模型版本的单个指标设置告警规则

方式一：对整个 ModelArts 服务设置告警规则

1. 登录管理控制台。
2. 在“服务列表”中选择“管理与监管 > 云监控服务”，进入“云监控服务”管理控制台。
3. 在左侧导航栏，选择“告警 > 告警规则”页面，单击“创建告警规则”。
4. 在“创建告警规则”页面，“资源类型”选择“ModelArts”，“维度”选择“服务”，“触发规则”选择“自定义创建”，设置告警策略，完成其他信息填写后，单击“立即创建”。

方式二：对单个服务设置告警规则

1. 登录管理控制台。
2. 在“服务列表”中选择“管理与监管 > 云监控服务”，进入“云监控服务”管理控制台。
3. 在左侧导航栏，选择“云服务监控 > ModelArts”。
4. 选择需要添加告警规则的在线服务名称，单击操作列的“创建告警规则”。
5. 在“创建告警规则”界面，根据界面提示设置ModelArts在线服务和模型负载的告警规则。

方式三：对单个版本设置告警规则

1. 登录管理控制台。
2. 在“服务列表”中选择“管理与监管 > 云监控服务”，进入“云监控服务”管理控制台。
3. 在左侧导航栏，选择“云服务监控 > ModelArts”。
4. 单击在线服务名称前面的小三角，展示模型版本列表，选择需要设置告警规则的模型版本，单击操作列的“创建告警规则”。
5. 在“创建告警规则”界面，根据界面提示设置模型负载的告警规则。

方式四：对服务或模型版本的单个指标设置告警规则

1. 登录管理控制台。
2. 在“服务列表”中选择“管理与监管 > 云监控服务”，进入“云监控服务”管理控制台。
3. 在左侧导航栏，选择“云服务监控 > ModelArts”。
4. 单击在线服务名称或单击在线服务名称前面的小三角，展示模型版本列表，单击模型版本名称，查看告警规则详情。
5. 在告警规则详情页，单击单个指标右上角的加号按钮，对服务或模型版本的单个指标设置告警规则。

8.6.3 查看监控指标

操作场景

云服务平台提供的云监控，可以对ModelArts在线服务和模型负载运行状态进行日常监控。您可以通过管理控制台，直观地查看ModelArts在线服务和模型负载的各项监控指标。由于监控数据的获取与传输会花费一定时间，因此，云监控显示的是当前时间5～10分钟前的状态。如果您的在线服务刚创建完成，请等待5～10分钟后查看监控数据。

前提条件

- ModelArts在线服务正常运行。
- 已在云监控页面设置告警规则，具体操作请参见**设置告警规则**。
- 在线服务已正常运行一段时间（约10分钟）。
- 对于新创建的在线服务，需要等待一段时间，才能查看上报的监控数据和监控视图。
- 故障、删除状态的在线服务，无法在云监控中查看其监控指标。当在线服务再次启动或恢复后，即可正常查看。

对接云监控之前，用户无法查看到未对接资源的监控数据。具体操作，请参见**设置告警规则**。

操作步骤

1. 登录管理控制台。
2. 在“服务列表”中选择“管理与监管 > 云监控服务”，进入“云监控服务”管理控制台。
3. 在左侧导航栏，选择“云服务监控 > ModelArts”。
4. 查看监控图表。
   - 查看在线服务监控图表：单击目标在线服务“操作”列的“查看监控指标”。
   - 查看模型负载监控图标：单击目标在线服务左侧的，在下拉列表中选择模型负载“操作”列的“查看监控指标”。
5. 在监控区域，您可以通过选择时长，查看对应时间的监控数据。

   当前支持查看近1小时、近3小时和近12小时的监控数据，查看更长时间范围监控曲线，请在监控视图中单击进入大图模式查看。

9 资源管理

9.1 资源池介绍

ModelArts 资源池说明

在使用ModelArts进行AI开发时，您可以选择使用如下两种资源池：

- **专属资源池**：专属资源池不与其他用户共享，资源更可控。在使用专属资源池之前，您需要先创建一个专属资源池，然后在AI开发过程中选择此专属资源池。其中专属资源池分为弹性集群和弹性裸金属。
  - 弹性集群：分为Standard弹性集群与Lite弹性集群。其中：
    - Standard弹性集群提供独享的计算资源，使用Modelarts开发平台的训练作业、部署模型以及开发环境时，通过Standard弹性集群的计算资源进行实例下发。
    - Lite弹性集群面向k8s资源型用户，提供托管式k8s集群，并预装主流AI开发插件以及加速插件，以云原生方式直接向用户提供AI Native的资源、任务等能力，用户可以直接操作资源池中的节点和k8s集群。
  - 弹性裸金属：弹性裸金属提供不同型号的xPU裸金属服务器，您可以通过弹性公网IP进行访问，在给定的操作系统镜像上可以自行安装GPU&NPU相关的驱动和其他软件，使用SFS或OBS进行数据存储和读取相关的操作，满足算法工程师进行日常训练的需要。
- **公共资源池**：公共资源池提供公共的大规模计算集群，根据用户作业参数分配使用，资源按作业隔离。 用户下发训练作业、部署模型、使用开发环境实例等，均可以使用ModelArts提供的公共资源池完成，按照使用量计费，方便快捷。

专属资源池和公共资源池的能力差异

- 专属资源池为用户提供独立的计算集群、网络，不同用户间的专属资源池物理隔离，公共资源池仅提供逻辑隔离，专属资源池的隔离性、安全性要高于公共资源池。
- 专属资源池用户资源独享，在资源充足的情况下，作业是不会排队的；而公共资源池使用共享资源，在任何时候都有可能排队。

- 专属资源池支持打通用户的网络，在该专属资源池中运行的作业可以访问打通网络中的存储和资源。例如，在创建训练作业时选择打通了网络的专属资源池，训练作业创建成功后，支持在训练时访问SFS中的数据。
- 专属资源池支持自定义物理节点运行环境相关的能力，例如GPU/Ascend驱动的自助升级，而公共资源池暂不支持。

9.2 弹性集群

9.2.1 ModelArts 资源池管理功能全面升级

ModelArts服务专属资源池管理能力全面升级，在新的体系下，专属资源池不再区分开发环境/训练专用和部署上线专用，取而代之的是统一的ModelArts专属资源池。新版专属资源池支持灵活配置可运行的作业类型的功能；也支持用户自行管理专属资源池网络，并进行网络打通的功能。

在ModelArts管理控制台新版专属资源池管理页面，提供了更加完备的管理能力，并展示了更加丰富的资源池信息。在本文档的后续章节中，提供了更多关于专属资源池使用和管理的详细说明以供查阅。若未使用过ModelArts的专属资源池功能，欢迎您直接尝试ModelArts新版专属资源池。若已使用过ModelArts的专属资源池功能，您可平滑切换至新版专属资源池使用。

您可通过阅读以下内容，初步了解新版专属资源池：

新版专属资源池有什么能力？

新版专属资源池是一个全面的技术和产品的改进，主要能力提升如下：

- **专属资源池类型归一**：不再区分训练、推理专属资源池。如果业务允许，您可以在一个专属资源池中同时跑训练和推理的Workload。同时，也可以通过“设置作业类型”来开启/关闭专属资源池对特定作业类型的支持。
- **自助专属池网络打通**：可以在ModelArts管理控制台自行创建和管理专属资源池所属的网络。若需要在专属资源池的任务中访问自己VPC上的资源，可通过“打通VPC”来实现。
- **更加完善的集群信息**：全新改版的专属资源池详情页面中，提供了作业、节点、资源监控等更加全面的集群信息，可帮助您及时了解集群现状，更好的规划使用资源。
- **自助管理集群GPU/NPU驱动**：每个用户对集群的驱动要求不同，在新版专属资源池详情页中，可自行选择加速卡驱动，并根据业务需要进行立即变更或平滑升级。
- **更细粒度的资源划分（Coming soon）**：您可以将已创建的专属资源池划分为多个“小池子”，并给每个小池子以不同的配额和使用权限，做到资源灵活且精细的分配和管理。

更多新的能力和体验，将在后续的版本中不断的提供，期待您有一个良好的使用旅程。

在新版专属资源池生效前创建的专属资源池，能否继续使用？

若您此前已经创建了专属资源池，这些资源池会保留不变，您在ModelArts管理控制台仍旧能看到原来的专属资源池（即弹性集群）管理入口，但不支持在此继续创建专属资源池。ModelArts支持将现有专属资源池迁移到新的体系下，此变更不需要您做任何

额外操作，我们会主动与您联系完成变更。同时，此变更不会对专属资源池上运行的Workload有任何影响。您唯一要关注的是后续需要切换到新的专属资源池（即弹性集群New）中管理，其提供了更加完善且易用的管理功能。而对于AI开发者，其提交训练任务或创建推理服务等，没有任何变化影响。

新版专属资源池和旧版专属资源池差异对比

- 旧版的开发环境/训练专用和部署上线专用专属资源池相互隔离，不能共用，且两者之间使用体验不同、提供的功能也不同。新版专属资源池将两者统一，用户可以通过设置专属资源池支持的作业类型，让资源池支持开发环境、训练作业、推理服务中的一个或多个，购买一份资源，实现多种用途。
- 新版专属资源池继承了旧版专属资源池的所有功能，并对专属资源池购买和扩缩容功能进行了大幅的体验优化，用户购买新版专属资源池可以获得更流畅、透明的购买体验。
- 新版专属资源池相比于旧版专属资源池进行了功能增强，使用新版专属资源池，用户可以享受资源池GPU/Ascend驱动自助升级、查看资源池作业排队详情、多个资源池共享一个网络等一系列新增功能，未来还会有更多新增功能将不断开放。

如果使用中遇到问题，如何获得帮助或提出反馈?

与ModelArts的其他功能一致，您可以随时在产品的侧边栏反馈问题或获取帮助。同时也建议您阅读本文档的后续章节，以便进一步了解ModelArts专属资源池相关使用方法。

专属资源池使用说明

- 若您是初次使用专属资源池，建议您可从**资源池介绍**开始，了解ModelArts提供的资源池详细说明。
- 在对ModelArts的资源池有一定了解后，若您需要创建一个自己的专属资源池，您可参考**创建资源池**来进行创建。
- 专属资源池创建成功后，可在**查看资源池详情**中查看专属资源池的详细信息。
- 若专属资源池的规格与您的业务不符，可通过**扩缩容资源池**来调整专属资源池的规格。
- 专属资源池提供了动态设置作业类型的功能，可参考**修改资源池作业类型**更新作业类型。
- ModelArts提供了自助升级专属资源池GPU/Ascend驱动的能力，可参考**资源池驱动升级**进行升级。
- 当不再需要使用专属资源池时，您可参考**删除资源池**删除专属资源池。
- 在使用专属资源池时，可能会存在各种异常，可参考**资源池异常处理**对使用专属资源池时遇到的异常情况进行处理。
- ModelArts提供了对网络的管理，同时支持打通VPC功能，具体可参见**ModelArts网络**。

9.2.2 创建资源池

本章节主要介绍创建专属资源池的详细操作。

创建专属资源池

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“专属资源池 > 弹性集群”。

说明

新用户在ModelArts管理控制台“专属资源池”中只能看到新版的“弹性集群”。使用过旧版专属资源池的老用户，可以看到两个弹性集群，其中“弹性集群 New”为新版的专属资源池。

2. 在“资源池”页签，单击“创建”，进入购买专属资源池界面，参见下表填写参数。

表 9-1 专属资源池的参数说明

参数名称	子参数	说明
名称	-	专属资源池的名称。 只能以小写字母开头，由小写字母、数字、中划线（-）组成，不能以中划线结尾。
描述	-	专属资源池的简要说明。
计费模式	-	选择计费模式，“按需计费”。
资源池类型	-	可选物理资源池和逻辑资源池。逻辑资源池与规格有关，若无逻辑规格则不显示逻辑资源池。
作业类型	-	根据业务需要，选择该资源池支持的作业类型。 ● 物理资源池：支持“开发环境”、“训练作业”和“推理服务”的作业类型。 ● 逻辑资源池：仅支持“训练作业”的作业类型。
网络	-	表示服务实例运行在指定的网络中，可以与该网络中的其它云服务资源实例互通。仅物理资源池需要设置网络。 在下拉框中选择，如果没有可用网络，单击右侧的“创建”，创建一个可用的网络。创建网络相关可以参考创建网络章节。
规格管理	规格类型	请根据界面提示选择需要使用的规格。平台分配的资源规格包含了一定的系统损耗，实际可用的资源量小于规格标称的资源。实际可用的资源量可在专属资源池创建成功后，在详情页的“节点”页签中查看。
	可用区	您可以根据实际情况选择“随机分配”或“指定AZ”。可用区是在同一区域下，电力、网络隔离的物理区域。可用区之间内网互通，不同可用区之间物理隔离。 ● 随机分配：系统自动分配可用区。 ● 指定AZ：指定资源池节点在哪个可用区域。考虑系统容灾时，推荐指定节点在同一个可用区。可设置可用区的节点数。

参数名称	子参数	说明
	节点数量	选择专属资源池的节点数，选择的节点数越多，计算性能越强。 当“可用区”选择“指定AZ”时，节点数量会根据可用区的数据自动计算，此处无须再次设置。 **说明** 单次创建时，节点数建议不大于30，否则可能触发限流导致创建失败。
	高级选项	开启后，可设置容器引擎空间大小。 容器引擎空间大小仅支持整数， 默认值与最小值为50G，不同规格的最大值不同，数值有效范围请参考界面提示。自定义设置容器引擎空间大小不会造成额外费用增加。
自定义驱动	-	选择规格为GPU/Ascend时，显示此参数。打开开关，选择驱动。
GPU驱动	-	打开“自定义驱动”开关，显示此参数。选择GPU加速卡驱动。 **说明** 规格为“hnt8”系列的，GPU驱动请选择“535.129.03”及以上版本的nvidia驱动。
高级选项	-	选中“现在配置”，可配置标签信息、网段、控制节点分布。
网段	-	可选默认和自定义。 ● 默认：系统随机分配一个不冲突的网段供用户使用，因后续不支持修改建议商用场景选择手动分配，确保网段符合用户诉求。 ● 自定义：需要自定义K8S容器网段和K8S服务网段。 – K8S容器网段：集群下容器使用的网段，决定了集群下容器的数量上限。创建后不可修改。 – K8S服务网段：同一集群下容器互相访问时使用的Service资源的网段。决定了Service资源的上限。 创建后不可修改。
集群规格	-	集群支持管理的最大节点数量，请根据业务场景选择。创建完成后支持扩容，不支持缩容。 可选默认规格和自定义规格。
控制节点分布	-	控制节点的分布位置，可选择随机分配和自定义。 ● 随机分配：随机分配控制节点可用区。 ● 自定义：需选择控制节点的可用区。 控制节点推荐尽可能随机分布在不同可用区以提高容灾能力。

3. 单击“下一步”确认规格。规格确认无误后，单击“提交”，即可创建专属资源池。
   – 当资源池创建成功后，资源池的状态会变成“运行中”，当“节点个数”中的“可用”和“总数”值大于0时，资源池才能下发任务。

- 可以将鼠标放在“创建中”字样上，查看当前创建过程详情。若点击查看详情，可跳转到“操作记录”中。
- 可以在资源池列表左上角“操作记录”中查看资源池的任务记录。

常见问题

为什么无法使用资源池节点上的全部CPU资源?

由于资源池节点上会安装系统、插件等内容，因此不能完全使用所有资源。例如：资源池节点是8U，节点分配给系统组件部分CPU，可用的资源会小于8U。

建议您在启动任务前，在该资源池的详情页中，单击“节点”页签，查看实际可用的CPU资源。

9.2.3 查看资源池详情

资源池详情页介绍

- 登录ModelArts管理控制台，在左侧导航栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”列表。
- 单击表头的标记，ModelArts支持根据资源池类型、状态筛选资源池。在列表右上角选择“名称”/“资源ID”，支持根据名称、资源ID进行筛选（在“费用中心 > 订单管理 > 我的订单”页面，单击对应订单的“详情”，可在资源信息中查看资源ID）。
- 在资源池列表中，单击某一资源池名称，进入资源池详情页，查看资源池的基本信息和其他扩展信息。
  - 当创建了多个资源池时，可在详情页单击左上角，可切换资源池。单击右上角“更多”，可进行扩缩容、删除、设置作业类型、退订、驱动升级等操作，不同资源池可进行的操作不一致，具体以控制台显示为准。
  - 在“基本信息”的“网络”中，可单击关联的资源池中的数字，查看关联的资源池。
  - 在扩展信息中可以查看监控、作业、节点、规格、事件，详细介绍见下文。

查看资源池中的作业

在资源池详情页，切换到“作业”页签。您可以查看该资源池中运行的所有作业，如果当前有作业正在排队，可以查看作业在资源池排队的位置。

说明

当前仅支持查看训练作业。

查看资源池事件

在资源池详情页，切换到“事件”页签。您可以查看资源从创建到添加节点的各个阶段的事件。产生事件的原因主要有“资源池状态变化”和“资源节点状态变化”。

在事件列表中，可单击“事件类型”列的筛选查看。

- 当资源池开始创建或者出现异常时，因资源池状态变化，会将此变化信息记录到事件中。

- 当节点的可用、异常、创建中、删除中的数量发生变化时，因资源池节点状态变化，会将此变化信息记录到事件中。

图 9-1 查看资源池事件

查看资源池节点

在资源池详情页，切换到“节点”页签。您可以查看资源池中所有的节点，并且能查看每个节点资源占用的情况。

由于集群组件会占用一部分资源，所以列表中CPU（可用/总数）呈现的资源数量不代表该节点物理资源数量，仅表示可被业务使用到的资源量。其中，CPU核数为微核，1000微核=1物理核。

- 替换节点：

  “节点”页签中提供对单个节点替换的功能。可单击操作列的“替换”，即可实现对单个节点的替换。替换节点操作不会收取费用。

  单击“操作记录”可查看当前资源池替换节点的操作记录。“运行中”表示节点在替换中。替换成功后，节点列表中会显示新的节点名称。

  替换最长时间为24小时，超时后仍然未找到合适的资源，状态会变为“失败”。

  可将鼠标悬浮在图标上，查看具体失败原因。

  说明

  - 每天累计替换的次数不超过资源池节点总数的20%，同时替换的节点数不超过资源池节点总数的5%。
  - 替换节点时需确保有空闲节点资源，否则替换可能失败。
  - 当操作记录里有节点处于重置中时，该资源池无法进行替换节点操作。

- 重置节点

  “节点”页签中提供节点重置的功能。单击操作列的“重置”，可实现对单个节点的重置。勾选多个节点的复选框，单击操作记录旁的“重置”按钮，可实现对多个节点的重置。

  下发重置节点任务时需要填写以下参数：

表 9-2 重置参数说明

参数名称	说明
操作系统	选择下拉框中支持的操作系统。
配置方式	选择重置节点的配置方式。 ● 按节点比例：重置任务包含多个节点时，同时被重置节点的最高比例。 ● 按节点数量：重置任务包含多个节点时，同时被重置节点的最大个数。

单击“操作记录”可查看当前资源池重置节点的操作记录。重置中节点状态为“重置中”，重置成功后，节点状态变为“可用”。重置节点操作不会收取费用。

说明

- 重置节点将影响相关业务的运行，请谨慎操作。
- 节点状态为“可用”的节点才能进行重置。
- 同一时间单个节点只能处于一个重置任务中，无法对同一个节点同时下发多个重置任务。
- 当操作记录里有节点处于替换中时，该资源池无法进行重置节点操作。
- 当资源池处于驱动升级状态时，该资源池无法进行重置节点操作。
- GPU和NPU规格，重置节点完成后，节点可能会出现驱动升级的现象，请耐心等待。

- 删除/退订/释放节点：

  若是“按需计费”的资源池，您可单击操作列的“删除”，即可实现对单个节点的资源释放。

  若想批量删除节点，勾选待删除节点名称前的复选框，然后单击名称上方的“删除”，即可实现对多个节点的资源释放。

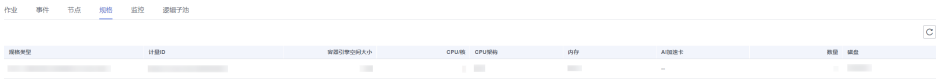
  说明

  - 删除节点可能导致该节点上运行的作业失败，请保证该节点无任务运行时再进行操作。
  - 当资源池中存在异常节点时，可通过删除操作，将资源池中指定的异常节点移除，再通过扩容专属资源池获得和之前相同的总节点个数。
  - 仅有一个节点时，无法进行删除操作。

查看资源池规格

在资源池详情页，切换到“规格”页签。您可以查看该资源池使用的资源规格以及该规格对应的数量。

图 9-2 查看资源池规格（若创建资源池时未设置容器引擎大小，则显示默认值）



查看资源池监控

在资源池详情页，切换到“监控”页签。展示了CPU使用量、内存利用率、磁盘可用容量等使用情况，均以资源池的维度呈现。当资源池中有AI加速卡时，还会显示GPU、NPU的相关监控信息。

9.2.4 扩缩容资源池

场景介绍

当专属资源池创建完成，使用一段时间后，由于用户AI开发业务的变化，对于资源池资源量的需求可能会产生变化，面对这种场景，ModelArts专属资源池提供了扩缩容功能，用户可以根据自己的需求动态调整。

- 使用扩容功能时，可以增加资源池已有规格的节点数量。

- 使用缩容功能时，可以减少资源池已有规格的节点数量。

说明

缩容操作可能影响到正在运行的业务，建议用户在业务空窗期进行缩容，或进入资源池详情页面，在指定空闲的节点上进行删除来实现缩容。

约束限制

- 只支持对状态为“运行中”的专属资源池进行扩缩容。
- 专属资源池不能缩容到0。

扩缩容专属资源池

资源池扩缩容有以下类型，分别为：

- 对已有规格增减节点数量
- 修改容器引擎空间大小

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”页签，查看资源池列表。

   说明

   在旧版资源池迁移到新版资源池的过程中，资源池状态显示为“受限”。此时，资源池无法进行扩缩容和退订。

2. 增减节点数量

   单击某个资源池操作列的“扩缩容”对资源池进行扩缩容。

   在“专属资源池扩缩容”页面，设置“资源配置 > 可用区”，可用区可选择随机分配和指定AZ。设置完成后，单击“提交”，在弹出的确认框中单击“确定”完成修改。

   - 选择随机分配时，可通过增减“目标总节点数”实现扩缩容，请用户根据本身业务诉求进行调整。增加目标节点数量即表示扩容，减少目标节点数量即表示缩容。扩缩容完成后，节点的可用区分布由系统后台随机选择。
   - 选择指定AZ时，可指定扩缩容完成后节点的可用区分布。

   图 9-3 资源配置（若创建资源池时未设置容器引擎大小，则显示默认值）

3. 修改容器引擎空间大小

   若您需要更大的容器引擎空间，您可以通过以下操作调整容器引擎空间大小。

   - 对于新建的资源，支持在新建资源池时指定容器引擎空间大小，请参见**创建资源池**中“规格管理”参数下“高级选项”。
   - 对于存量的资源，支持修改容器引擎空间大小。

- 方式一：单击某个资源池名称，进入资源池详情，单击“规格”页签，单击操作列的“调整容器引擎空间大小”，修改容器引擎空间大小。
- 方式二：单击某个资源池操作列的“扩缩容”，修改容器引擎空间大小。

须知

修改容器引擎空间大小仅作用在新建节点上，且会导致资源池内该规格下节点的dockerBaseSize不一致，可能会使得部分任务在不同节点的运行情况不一致。

9.2.5 工作空间迁移

背景信息

专属资源池的工作空间关联了企业项目，企业项目涉及到账单归集。为隔离不同子用户操作资源的权限，ModelArts提供了工作空间功能，管理员可以根据工作空间，隔离不同子用户操作工作空间内资源的权限。工作空间迁移包括资源池迁移和网络迁移，具体方法可见下文说明。

资源池工作空间迁移

1. 登录ModelArts管理控制台，选择“专属资源池 > 弹性集群”，进入资源池列表。
2. 在资源池列表中，选择目标资源池“操作 > 更多 > 工作空间迁移”。
3. 在弹出的“迁移专属资源池”中，选择要迁移的“目标工作空间”，单击“确定”。

网络工作空间迁移

1. 登录ModelArts管理控制台，选择“专属资源池 > 弹性集群”，切换到“网络”页签。
2. 在网络列表中，选择目标网络“操作 > 更多 > 工作空间迁移”。
3. 在弹出的“迁移网络”中，选择要迁移的“目标工作空间”，单击“确定”。

9.2.6 修改资源池作业类型

场景介绍

ModelArts含有许多“作业”类型（作业为统称，并非单指训练作业），其中有一部分是可以运行在专属资源池上的，包括“训练”、“推理”服务及“Notebook”开发环境。

专属资源池提供了动态设置作业类型的功能，您可以在创建资源池时、创建完成后，对资源池支持的作业类型进行编辑（新增或减少）。当前支持的“作业类型”有“训练作业”、“推理服务”和“开发环境”，用户可按需自行选择。

设置某一作业类型后，即可在此专属资源池中下发此种类型的作业，没有设置的作业类型不能下发。

⚠ 注意

为了支持不同的作业类型，后台需要在专属资源池上进行不同的初始化操作，例如安装插件、设置网络环境等。其中部分操作需要占据资源池的资源，导致用户实际可用资源减少。因此建议用户按需设置，避免不必要的资源浪费。

约束限制

专属资源池状态处于“运行中”。

操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”页面。
2. 在资源池列表中，选择某个资源池操作列“更多 > 设置作业类型”。
3. 在“设置作业类型”弹窗中，选择需要设置的作业类型。
4. 选择完成后，单击“确定”，启用作业类型。

9.2.7 资源池驱动升级

场景介绍

当专属资源池中的节点含有GPU/Ascend资源时，用户基于自己的业务，可能会有自定义GPU/Ascend驱动的需求，ModelArts面向此类客户提供了自助升级专属资源池GPU/Ascend驱动的能力。

驱动升级有两种升级方式：安全升级、强制升级。

说明

- 安全升级：不影响正在运行的业务，开始升级后会先将节点进行隔离（不能再下发新的作业），待节点上的存量作业运行完成后再进行升级，因需要等待作业完成，故升级周期可能比较长。
- 强制升级：忽略资源池中正在运行的作业，直接进行驱动升级，可能会导致运行中作业失败，需谨慎选择。

约束限制

- 专属资源池状态处于运行中，且专属池中的节点需要含有GPU/Ascend资源。
- 对于逻辑资源池，需要开启节点绑定后才能进行驱动升级，请提交工单联系工程师开启节点绑定。

驱动升级操作

1. 登录ModelArts管理控制台，在左侧导航栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”页面。
2. 在资源池列表中，选择需要进行驱动升级的资源池“操作 > 驱动升级”。
3. 在“驱动升级”弹窗中，会显示当前专属资源池的驱动类型、节点数量、当前版本、目标版本和升级方式。
   - 目标版本：在目标版本下拉框中，选择一个目标驱动版本。

- 升级方式：选择“升级方式”，可选择安全升级或强制升级。

4. 选择完成后，单击“确定”开始驱动升级。

9.2.8 删除资源池

当AI业务开发不再需要使用专属资源池时，您可以删除专属资源池，释放资源。

说明

专属资源池删除后，将导致使用此资源的开发环境、训练作业和推理服务等不可用，且删除后不可恢复，请谨慎操作。

1. 登录ModelArts管理控制台，选择左侧导航栏“专属资源池 > 弹性集群”，默认进入“专属资源池”页面。
2. 在专属资源池列表中，选择需要删除的资源池的操作列“更多 > 删除”。
3. 在“删除资源池”页面，需在文本框中输入“DELETE”，单击“确定”，删除资源池。

   可切换“训练作业”、“推理服务”、“开发环境”页签查看资源池上创建的训练作业、部署的推理服务、创建的Notebook实例。

9.2.9 资源池异常处理

资源配额限制

在使用专属资源池时（如资源扩缩容、创建VPC、创建VPC-子网、打通VPC），如果提示相关资源配额受限，请处理。

创建失败/变更失败

1. 登录ModelArts管理控制台，选择左侧导航栏“专属资源池 > 弹性集群”，默认进入“资源池”页面。
2. 您可以通过单击“创建”右侧的“操作记录”，查看当前处于失败状态的资源池信息。

图 9-4 创建失败资源池信息

3. 鼠标悬停在“状态”列的 上，即可看到该操作失败的具体原因。

说明

失败的记录默认按照操作的申请时间排序，最多显示500条并保留3天。

节点故障定位

ModelArts平台在识别到节点故障后，通过给K8S节点增加污点的方式（taint）将节点隔离避免新作业调度到该节点而受到影响，并且使本次作业不受污点影响。当前可识别的故障类型如下，可通过隔离码及对应检测方法定位故障。

表 9-3 隔离码

隔离码	分类	子类	异常中文描述	检测方法
A05 0101	GPU	显存	GPU ECC错误。	通过nvidia-smi -a查询到存在Pending Page Blacklist为Yes的记录，或多比特Register File大于0。对于Ampere架构的GPU，存在以下场景： ● 存在不可纠正的SRAM错误。 ● 存在Remapping Failure记录。 ● dmsg中存在Xid 95事件。 （参考**NVIDIA GPU Memory Error Management**） Ampere架构GPU显存错误分级： ● L1: 可纠正ECC错误（单比特ECC错误），不影响业务。观测方式：nvidia-smi -a中查询到Volatile Correctable记录。 ● L2: 不可纠正ECC错误（多比特ECC错误），当次业务受损，重启进程可恢复。观测方式：nvidia-smi -a中查询到Volatile Uncorrectable记录。 ● L3: 错误未被抑制，可能影响后续业务，需要重置卡或重启节点。观测方式：Xid事件中包含95事件。(Remapped的Pending记录只作为提示，当业务空闲时进行卡重置触发重映射即可) ● L4: 需要换卡，SRAM Uncorrectable>4或者Remapped Failed。
A05 0102	GPU	其他	nvidia-smi返回信息中包含ERR。	通过nvidia-smi -a查询到ERR!，通常为硬件问题，如电源风扇等问题。
A05 0103	GPU	其他	nvidia-smi执行错误，超时或者不存在。	执行nvidia-smi退出码非0。
A05 0104	GPU	显存	ECC错误到达64次。	通过nvidia-smi -a查询到Retired Pages中，Single Bit和Double Bit之和大于64。

隔离码	分类	子类	异常中文描述	检测方法
A05 0148	GPU	其他	infoROM告警。	执行nvidia-smi的返回信息中包含“infoROM is corrupted”告警。
A05 0109	GPU	其他	GPU其他错误。	检测到的其他GPU错误，通常为硬件问题，请联系技术人员支持。
A05 0147	IB	链路	IB网卡异常。	ibstat查看网卡非Active状态。
A05 0121	NPU	其他	npu dcmi接口检测到driver异常。	NPU驱动环境异常。
A05 0122	NPU	其他	npu dcmi device异常。	NPU设备异常，昇腾dcmi接口中返回设备存在重要或紧急告警。
A05 0123	NPU	链路	npu dcmi net异常。	NPU网络链接异常。
A05 0129	NPU	其他	NPU其他错误。	检测到的其他NPU错误，通常为不可自纠正的异常，请联系技术人员支持。
A05 0149	NPU	链路	hccn tool网口闪断检查。	NPU网络不稳定，存在闪断情况。通过“hccn_tool-i ${device_id} -link_stat -g”查看24小时内闪断5次以上。
A05 0951	NPU	显存	NPU ECC次数达到维修阈值。	NPU的HBM Double Bit Isolated Pages Count值大于等于64。
A05 0146	Runtime	其他	ntp异常。	ntpd或者chronyd服务异常。
A05 0202	Runtime	其他	节点NotReady。	节点不可达，k8sNode存在以下污点之一： ● node.kubernetes.io/unreachable ● node.kubernetes.io/not-ready
A05 0203	Runtime	掉卡	AI正常卡数和实际容量不匹配。	检测到存在GPU或NPU掉卡情况。
A05 0206	Runtime	其他	Kubelet硬盘只读。	“/mnt/paas/kubernetes/kubelet”目录为只读状态。
A05 0801	节点管理	节点运维	资源预留。	节点被标记为备机，并具有备机污点。
A05 0802	节点管理	节点运维	未知错误。	节点被标记为具有未知故障污点。
A20 0001	节点管理	驱动升级	GPU升级。	节点正在执行GPU驱动升级。

隔离码	分类	子类	异常中文描述	检测方法
A20 0002	节点管理	驱动升级	NPU升级。	节点正在执行NPU驱动升级。
A20 0008	节点管理	节点准入	准入检测。	节点正在进行节点准入检测，包括基本的节点配置检查和简单的业务验证。
A05 0933	节点管理	容错Failover	当节点具有该污点时，会将节点上容错（Failover）业务迁移走。	当节点标记该污点时，会将节点上容错（Failover）业务迁移走。
A05 0931	训练toolkit	预检容器	训练预检容器检测到GPU错误。	训练预检容器检测到GPU错误。
A05 0932	训练toolkit	预检容器	训练预检容器检测IB错误。	训练预检容器检测IB错误。

9.2.10 ModelArts 网络

ModelArts 网络与 VPC 介绍

ModelArts网络是承载ModelArts资源池节点的网络连接，对用户仅提供网络名称以及CIDR网段的选择项，为了防止在打通VPC的时候有网段的冲突，因此提供了多个CIDR网段的选项，用户可以根据自己的实际情况进行选择。

虚拟私有云VPC是一套为实例构建的逻辑隔离的、由用户自主配置和管理的虚拟网络环境。为云服务器、云容器、云数据库等资源构建隔离的、用户自主配置和管理的虚拟网络环境，提升用户资源的安全性，简化用户的网络部署。

前提条件

- 已经创建虚拟私有云。
- 已经创建子网。

创建网络

1. 登录ModelArts管理控制台，在左侧导航栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”页面。
2. 切换到“网络”页签，单击“创建”，弹出“创建网络”页面。
3. 在“创建网络”弹窗中填写网络信息。
   - 网络名称：创建网络时默认生成网络名称，也可自行修改。
   - 网段类型：可选“预置”和“自定义”。

说明

- 单用户最多可创建15个网络。
- 网段设置以后不能修改，避免与将要打通的VPC网段冲突。可能冲突的网段包括：
  - 用户的vpc网段
  - 容器网段（固定是172.16.0.0/16）
  - 服务网段（固定是10.247.0.0/16）

4. 确认无误后，单击“确定”。

打通 VPC（可选）

通过打通VPC，可以方便用户跨VPC使用资源，提升资源利用率。

1. 在“网络”页签，单击网络列表中某个网络操作列的“打通VPC”。

图 9-5 打通 VPC

2. 在打通VPC弹框中，打开“打通VPC”开关，在下拉框中选择可用的VPC和子网。

说明

需要打通的对端网络不能和当前网段重叠。

- 如果没有VPC可选，可以单击右侧的“创建虚拟私有云”，跳转到网络控制台，申请创建虚拟私有云。
- 如果没有子网可选，可以单击右侧的“创建子网”，跳转到网络控制台，创建可用的子网。
- 支持1个VPC下多个子网的打通，单击“+”即可添加子网（上限10个）。

专属资源池访问外网

若需要专属资源池访问外网，需要完成以下步骤：

步骤1 打通VPC，请参见**打通VPC（可选）**。

步骤2 为VPC配置SNAT，请参考“虚拟私有云(VPC) 使用指南 > 用户指南”的“配置SNAT服务器”章节。

----结束

删除网络

当AI业务开发不再需要使用网络时，您可以删除网络。

1. 在“网络”页签，单击某个网络操作列的“删除”。
2. 确认删除，单击“确定”即可。

9.3 弹性裸金属

9.3.1 弹性裸金属简介

弹性裸金属通过提供BMS裸机的登录使用方式，允许用户以root账号自行安装部署AI框架、应用程序等第三方软件，为用户提供专属的云上物理服务器。用户只需要选取指定服务器的规格、镜像、所需要的网络配置、密钥等基本信息，即可以快速创建弹性裸金属，获取所需要的云上物理服务器。

表 9-4 名词解释

名词	含义
BMS	裸金属服务器（Bare Metal Server）是一款兼具虚拟机弹性和物理机性能的计算类服务，为您和您的企业提供专属的云上物理服务器，为核心数据库、关键应用系统、高性能计算、大数据等业务提供卓越的计算性能以及数据安全。
镜像	弹性裸金属提供公共镜像。若使用Snt9规格，则镜像版本为Euler2.8，驱动版本为21.0.2。若使用Snt9b规格，则镜像版本为Euler2.10，驱动版本为23.0.rc2。
磁盘	弹性裸金属提供基于裸金属服务器本地硬盘的本地盘产品。 本地盘：裸金属服务器的本地硬盘设备，包括NVMe SSD、SATA等磁盘类型，适用于数据量大、对存储I/O性能、实时性等要求很高的业务场景，具有低时延、高吞吐量、高性价比等产品能力。
SSH密钥对	弹性裸金属只支持SSH密钥对的方式进行登录，可以让用户无需输入密码登录到裸金属云服务器，因此可以防止由于密码被拦截、破解造成的账户密码泄露，从而提高裸金属云服务器的安全性。 SSH密钥对，简称为密钥对，是为用户提供的远程登录Linux云服务器的认证方式，是一种区别于传统的用户名和密码登录的认证方式。 **说明** 为保证云服务器安全，未进行私钥托管的私钥只能下载一次，请妥善保管。

名词	含义
网络	• 虚拟私有云 虚拟私有云（Virtual Private Cloud，VPC），为裸金属服务器构建隔离的、用户自主配置和管理的虚拟网络环境，提升用户云中资源的安全性，简化用户的网络部署。您可以在VPC中定义安全组、VPN、IP地址段、带宽等网络特性。用户可以通过VPC方便地管理、配置内部网络，进行安全、快捷的网络变更。同时，用户可以自定义安全组内与组间的访问规则，加强裸金属服务器的安全保护。 • RoCE网络 弹性裸金属支持组建RoCE网络，可以使用户方便地进行大规模分布式计算。RoCE网络是分布式存储网络的一种类型网络，是基于以太网的RDMA（远程直接数据读取技术）技术。RDMA可以简单理解为利用相关的硬件和网络技术，服务器1的网卡可以直接读写服务器2的内存，最终达到高带宽、低延迟和低资源利用率的效果。

9.3.2 准备工作

Step1：特性开通

弹性裸金属为白名单特性，若想使用请联系局点负责人开通特性。

Step2：基础权限开通

基础权限开通需要登录管理员账号，为子用户账号开通弹性裸金属功能所需的基础权限（ModelArts FullAccess/BMS FullAccess/ECS FullAccess/VPC FullAccess/VPC Administrator/VPCEP Administrator）。

步骤1 登录统一身份认证服务IAM管理控制台。

步骤2 在“用户组”页面，单击“创建用户组”。

步骤3 填写“用户组名称”并单击“确定”。

步骤4 在操作列单击“用户组管理”，将需要配置权限的用户加入用户组中，单击“确定”。

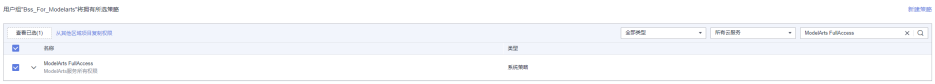
步骤5 单击用户组名称，进入用户组详情页。

步骤6 在“授权记录”页签下，单击“授权”。

图 9-6 “授权”

步骤7 在搜索栏输入“ModelArts FullAccess”，并勾选“ModelArts FullAccess”。

图 9-7 ModelArts FullAccess



步骤8 以相同的方式，依次添加：BMS FullAccess、ECS FullAccess、VPC FullAccess、VPC Administrator、VPCEP Administrator。（Server Administrator为VPC Administrator的依赖，会自动被勾选）。

图 9-8 基础权限

步骤9 单击“下一步”。

步骤10 授权范围方案选择“指定区域项目资源”，勾选相关项目。

图 9-9 作用范围

步骤11 单击“确定”，完成基础权限开通。

----结束

Step3：创建虚拟私有云

创建虚拟私有云需要登录管理员账号，IP地址段请根据现网情况合理规划。

步骤1 登录管理控制台。

步骤2 在左侧服务列表中，单击“网络 > 虚拟私有云VPC”，进入虚拟私有云页面。

步骤3 单击右上角“创建虚拟私有云”后（可采用默认参数），单击“立即创建”。

图 9-10 新建虚拟私有云

----结束

Step4：创建密钥对

步骤1 登录ModelArts管理控制台。

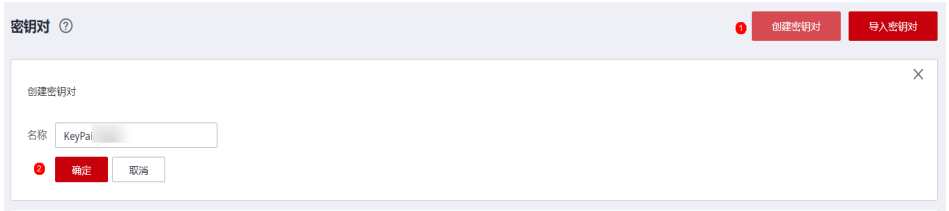
步骤2 在左侧导航栏中，选择“专属资源池 > 弹性裸金属”，进入“弹性裸金属”列表页面。

步骤3 单击“创建”，进入“创建DevServer”页面。

步骤4 单击“新建密钥对”。

步骤5 在密钥对页面中单击右上角“创建密钥对”后，单击“确定”，并将密钥对保存至本地。

图 9-11 创建密钥对



说明

为保证云服务器安全，未进行私钥托管的私钥只能下载一次，请妥善保管。

----结束

9.3.3 快速入门

本快速入门将帮助您学习如何创建和使用弹性裸金属。通过本文档，您可以了解到如何创建弹性裸金属实例，如何通过SSH登录实例以及释放实例资源。

前提条件

已开通弹性裸金属特性、创建虚拟私有云和密钥对，请见**准备工作**。

操作步骤

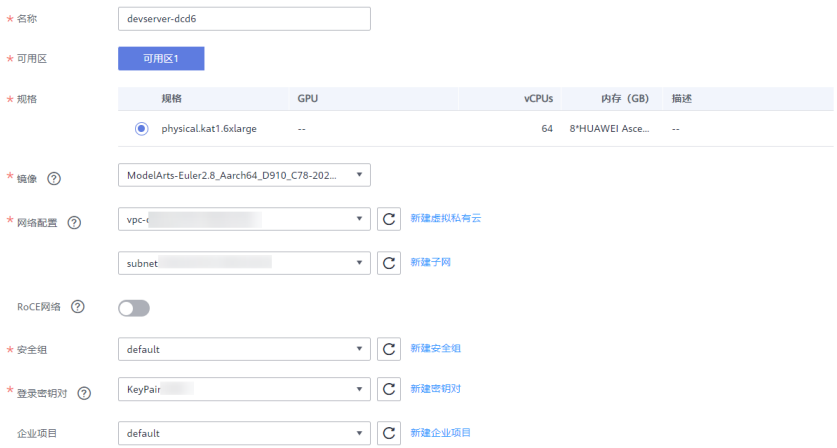
步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“专属资源池 > 弹性裸金属”，进入“弹性裸金属”列表页面。

步骤3 单击“创建”，进入“创建弹性裸金属”页面，在该页面填写相关参数信息。

- 网络配置：选择您的虚拟私有云及子网。
- 登录密钥对：选择您的密钥对。
- 其他参数保持默认即可。

图 9-12 参数信息



步骤4 单击“立即创建”，完成实例的创建。

如果需要批量购买多台资源，修改购买量后，单击“立即创建”。

说明

若ModelArts弹性裸金属创建失败，可能由多种原因导致，以下给出了几种类型的可能原因进行快速排查和定位解决。

- 资源不足：跳转到BMS页面，查看是否有要购买的规格，如果无该规格，说明该局点无该规格资源，需要联系局点负责人获取到资源后再进行购买。
- 配额不足：查看账户的资源配额是否满足，若该账号下资源配额，包括核心数、RAM等，如果未满足也会导致创建失败，需要申请配额后再进行购买。
- BMS机器内部错误：查看BMS界面，创建失败出现内部错误，此类问题请联系局点负责人定位解决。

步骤5 联系管理员完成**管理员侧网络配置**，并获取弹性公网IP和公网端口号。

步骤6 打开SSH连接工具，填写配置参数（以**MobaXterm**为例）。

单击SSH，填写弹性公网IP、登录账户名和端口号。

图 9-13 新建 SSH

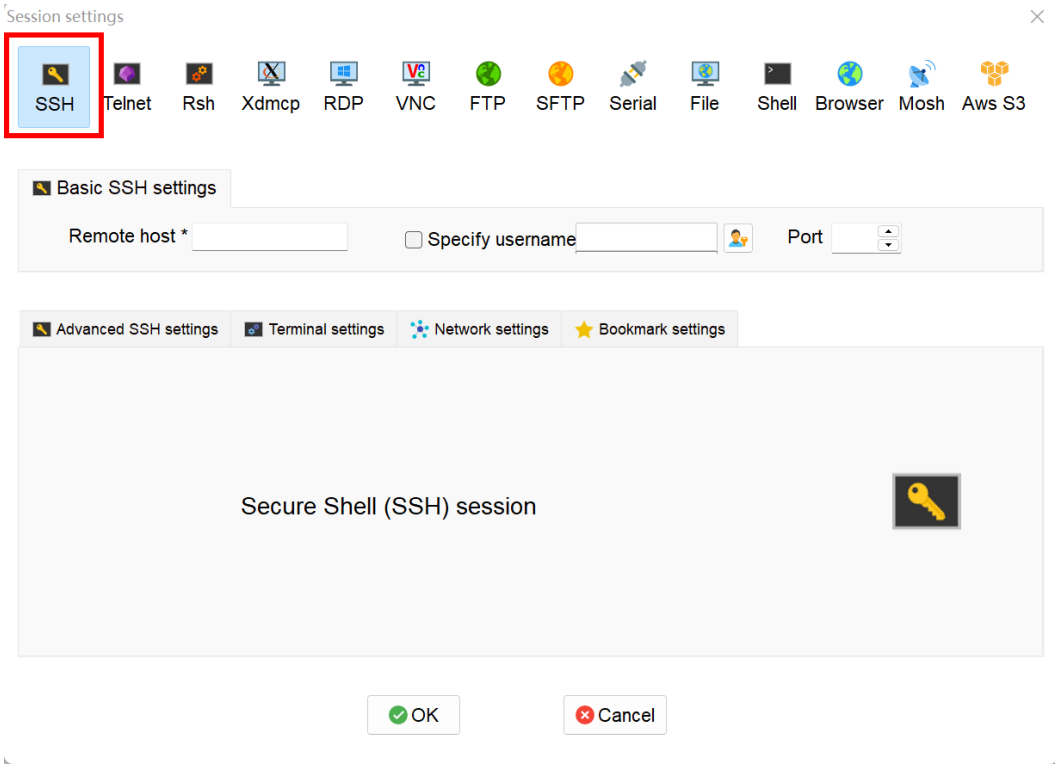
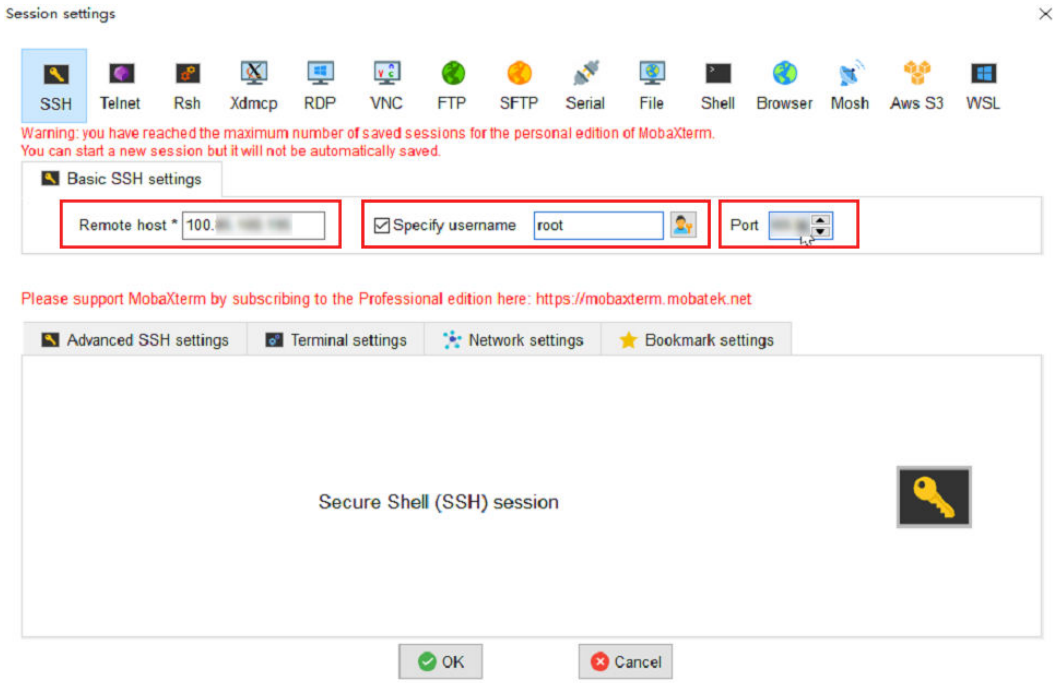


图 9-14 填写基本配置



单击“Advanced SSH settings”，选择创建实例时的密钥对文件，并单击“OK”。成功登录界面如下所示。

图 9-15 配置密钥对

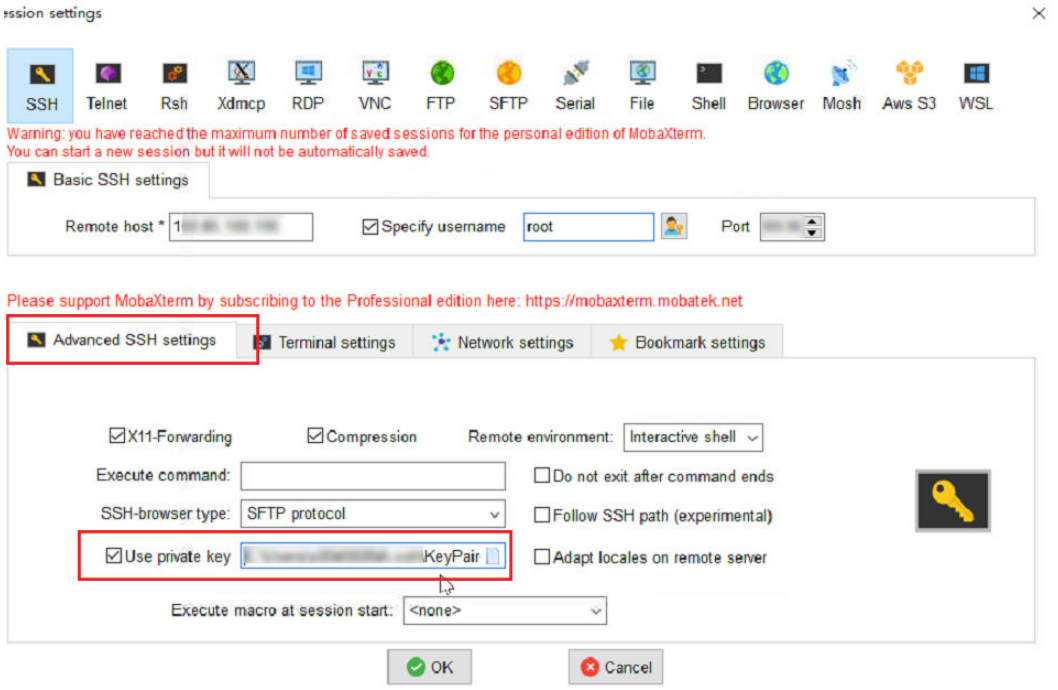
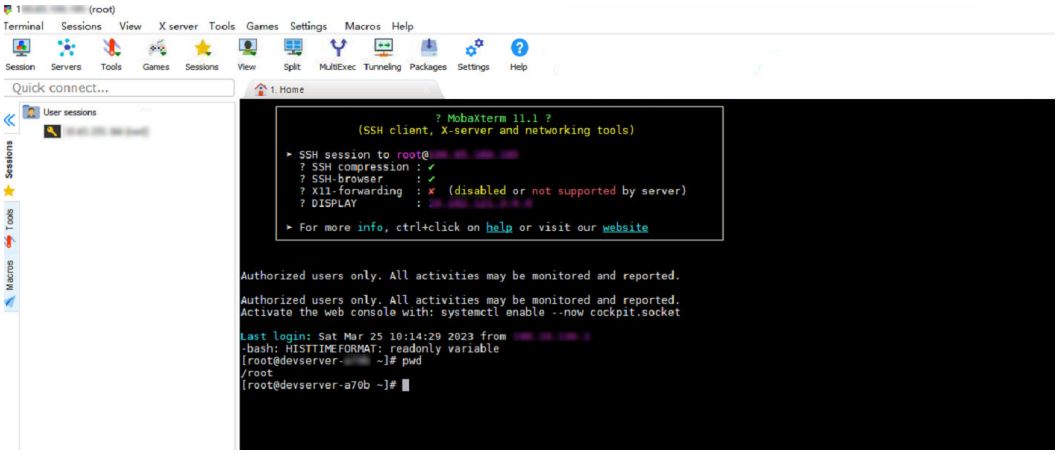
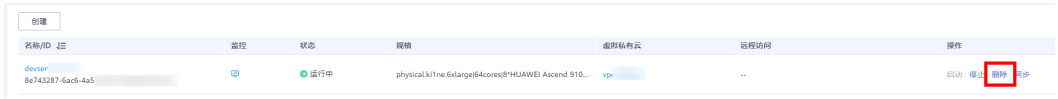


图 9-16 登录成功界面



步骤7 （可选）当需要释放实例时，在控制台页面的弹性裸金属列表中，单击操作列的“删除”，在弹出的确认对话框中，确认信息无误，然后单击“确定”，完成删除操作。

图 9-17 删除实例



----结束

9.3.4 管理弹性裸金属实例

9.3.4.1 创建弹性裸金属实例

前提条件

已完成基础权限配置、虚拟私有云创建、密钥对创建，请见**准备工作**。

创建弹性裸金属实例

步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“专属资源池 > 弹性裸金属”，进入“弹性裸金属”列表页面。

步骤3 单击“创建”，进入“创建DevServer”页面，在该页面填写相关参数信息。

表 9-5 基本信息说明

参数名称	说明
名称	弹性裸金属的名称。只能包含数字、大小写字母、下划线和中划线，长度不能超过64位且不能为空。

表 9-6 实例规格的详细参数

参数名称	说明
可用区	可用区是同一服务区内，电力和网络互相独立的地理区域，一般是一个独立的物理机房，这样可以保证可用区的独立性。是否将资源放在同一可用区内，主要取决于您对容灾能力和网络时延的要求。 ● 如果您的应用需要较高的容灾能力，建议您将资源部署在同一区域的不同可用区内。 ● 如果您的应用要求实例之间的网络延时较低，则建议您将资源创建在同一可用区内。
规格	弹性裸金属的实例规格，包含通用计算型和GPU加速型等。具体规格可有区域差异，以最终显示为准。
镜像	ModelArts服务提供的虚拟机镜像。

表 9-7 实例规格的网络资源参数

参数名称	说明
网络配置	配置弹性裸金属的网络环境。
RoCE网络	资源类型为裸金属服务器时有该参数，该参数用于配置弹性裸金属RoCE网络，该网络是支持分布式场景的必备条件，否则无需配置；若该账号下无RoCE网络，请联系相应局点负责人员解决。
网络名称	RoCE网络名称，仅打开RoCE网络后需要选择网络名称。 **说明** Snt9和Snt9b规格对应的RoCE网络类型不同，请同局点负责人员确认。
安全组	安全组是一个逻辑上的分组，为同一个VPC内具有相同安全保护需求并相互信任的弹性裸金属提供访问策略。
系统盘	系统盘和规格有关，选择支持挂载的规格才会显示此参数。可以在创建完成后在云服务器侧实现数据盘挂载或系统盘的扩容，建议取值至少100GB。

参数名称	说明
登录凭证	“密钥对”方式创建的裸金属服务器安全性更高，建议选择“密钥对”方式。如果您习惯使用“密码”方式，请增强密码的复杂度，保证密码符合要求，防止被恶意攻击。 ● 密钥对 指使用密钥对对作为登录裸金属服务器的鉴权方式。您可以选择使用已有的密钥对，或者单击“新建密钥对”创建新的密钥对。 **说明** 如果选择使用已有的密钥对，请确保您已在本地获取该文件，否则，将影响您正常登录裸金属服务器。 ● 密码 指使用设置初始密码方式作为裸金属服务器的鉴权方式，此时，您可以通过用户名密码方式登录裸金属服务器。 Linux操作系统时为root用户的初始密码，Windows操作系统时为Administrator用户的初始密码。密码复杂度需满足以下要求： – 长度为8至26个。 – 至少包含大写字母、小写字母、数字及特殊符号(!@$%^-_=+[{}]:,./?)中的3种 – 不能与用户名或倒序的用户名相同。 – 不能包含root或administrator及其逆序。
企业项目	非必选项，当账号开通企业项目服务后，可以选择对应的企业项目。

步骤4 如果需要批量购买多台资源，修改购买量。单击“立即创建”，完成实例的创建。

----结束

说明

若ModelArts弹性裸金属创建失败，可能由多种原因导致，以下给出了几种类型的可能原因进行快速排查和定位解决。

- 资源不足：跳转到BMS页面，查看是否有要购买的规格，如果无该规格，说明该局点无该规格资源，需要联系局点负责人获取到资源后再进行购买。
- 配额不足：查看账户的资源配额是否满足，若该账号下资源配额，包括核心数、RAM等，如果未满足也会导致创建失败，需要申请配额后再进行购买。
- BMS机器内部错误：查看BMS界面，创建失败出现内部错误，此类问题请联系局点负责人定位解决。

9.3.4.2 查看实例详情

步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“专属资源池 > 弹性裸金属”，进入“弹性裸金属”列表页面。

步骤3 在“弹性裸金属”列表中，单击实例名称，进入实例详情页，查看弹性裸金属实例详细信息。

表 9-8 实例规格的详细参数

参数名称	说明
名称	实例的名称。
ID	实例的唯一标识。
虚拟私有云	实例的虚拟私有云名称。
镜像	实例所使用的镜像。
访问密钥	实例的登录密钥对。
规格	实例的规格。
状态	实例的状态。
裸金属服务器	实例的裸金属服务器。
创建时间	实例的创建时间。
更新时间	实例的更新时间。

----结束

9.3.4.3 使用 SSH 远程登录

前提条件

- 创建一个弹性裸金属实例，该实例状态必须处于“运行中”。
- 已完成**管理员侧网络配置**。
- 您在创建裸金属服务器时，已经设置了密钥对登录。
- 准备好密钥对文件。

  密钥对在用户第一次创建时，自动下载，之后使用相同的密钥时不会再有下载界面（用户一定要保存好），或者每次都使用新的密钥对。
- 使用的登录工具（如MobaXterm）与待登录的裸金属服务器之间网络连通。例如，默认的22端口没有被防火墙屏蔽。

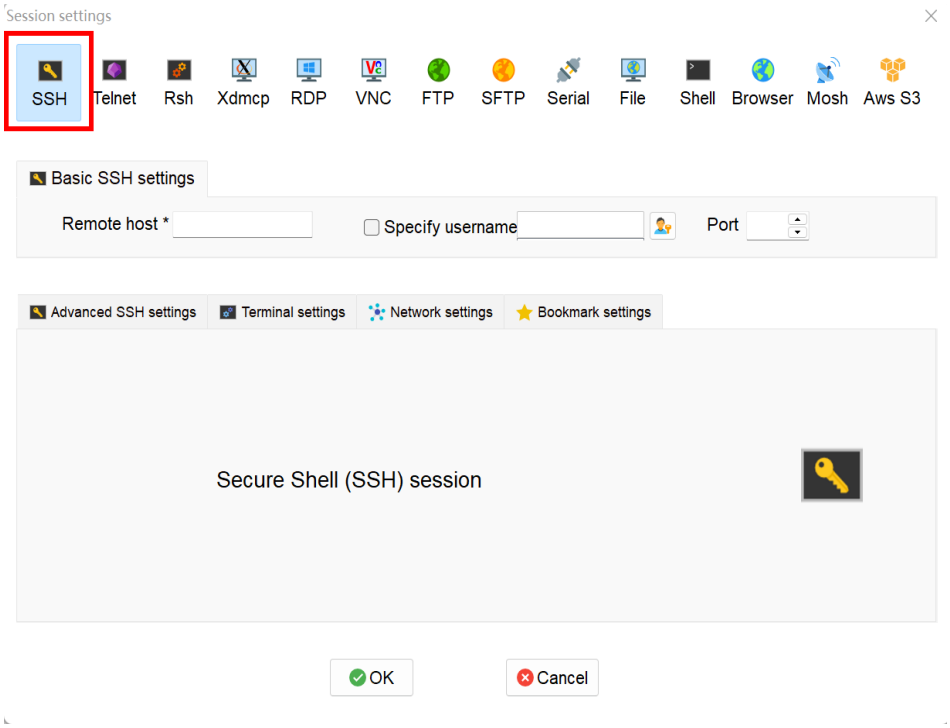
操作步骤

步骤1 联系管理员获取弹性公网IP和公网端口号。

步骤2 打开SSH连接工具，填写配置参数（以**MobaXterm**为例）。

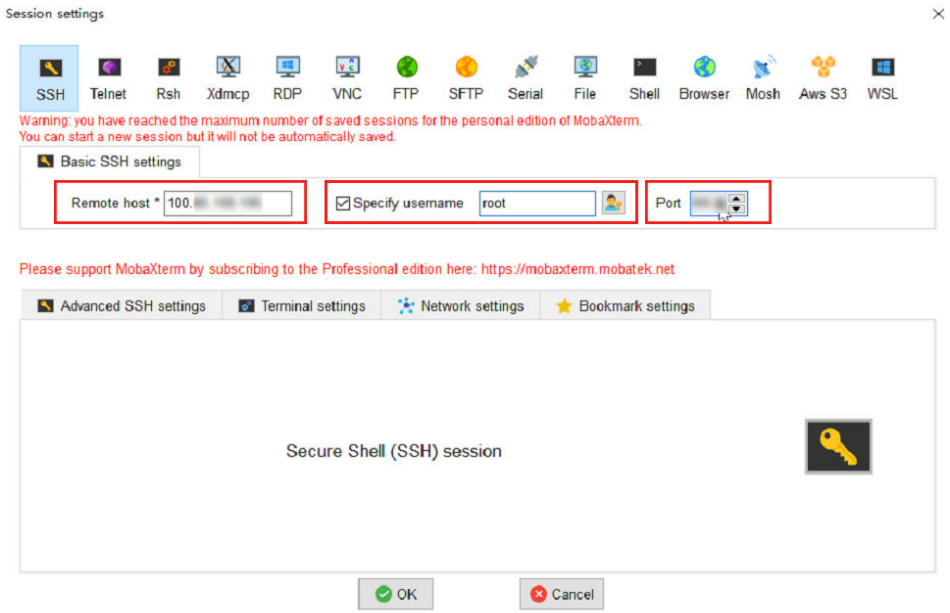
1. 单击SSH。

图 9-18 新建 SSH



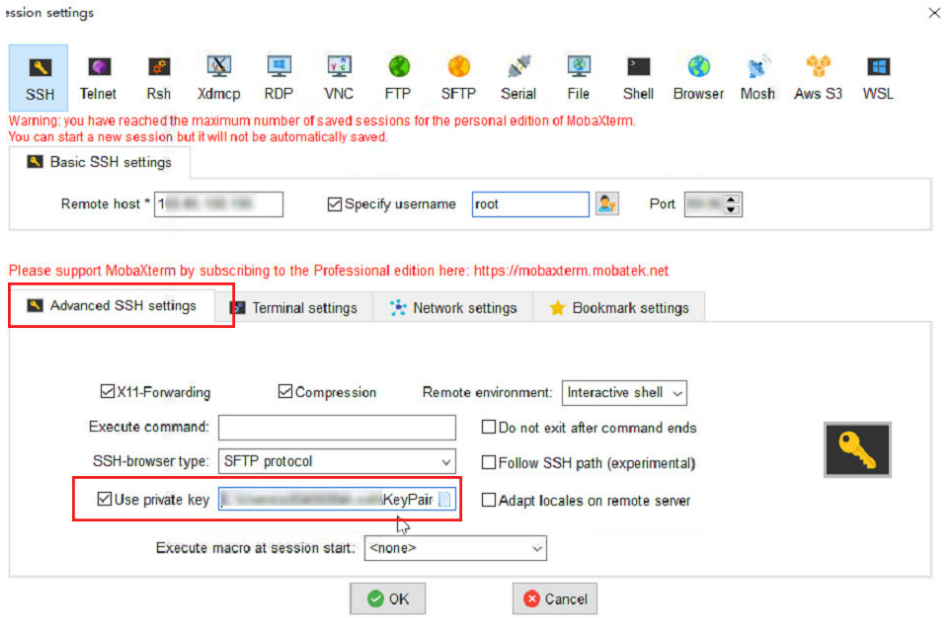
2. 填写弹性公网IP、登录账户名和端口号。

图 9-19 填写基本配置



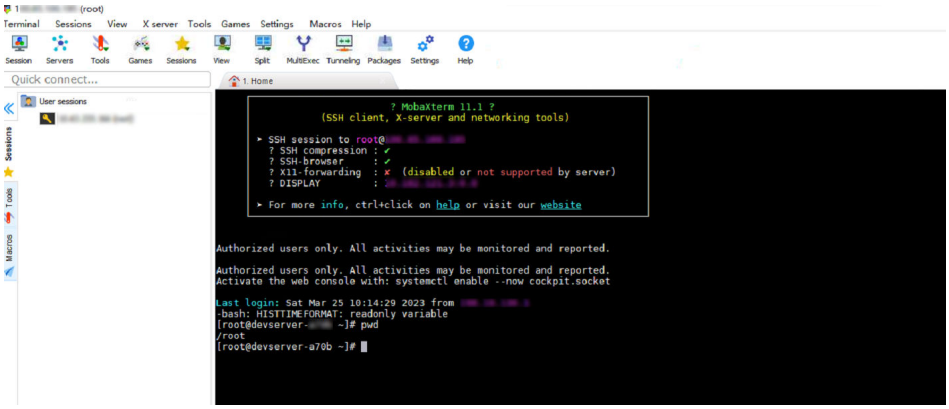
3. 单击“Advanced SSH settings”，选择创建实例时的密钥对文件，并单击“OK”。

图 9-20 配置密钥对



步骤3 登录成功界面如下所示。

图 9-21 登录成功界面



----结束

9.3.4.4 启动/停止实例

当您暂时不需要使用弹性裸金属的时候，可以通过对运行中的裸金属实例进行停止操作，停止对资源的消耗。当需要使用的时候，对于停止状态的弹性裸金属，可以通过启动操作重新使用弹性裸金属。

步骤1 登录ModelArts管理控制台。

步骤2 在左侧菜单栏中选择“专属资源池 > 弹性裸金属”。

步骤3 执行如下操作，启动或停止弹性裸金属。

- 启动弹性裸金属：单击“操作”列的“启动”。只有处于“已停止/停止失败/启动失败”状态的弹性裸金属可以执行启动操作。

- 停止弹性裸金属：单击“操作”列的“停止”，在弹出的确认对话框中，确认信息无误，然后单击“确定”。只有处于“运行中/停止失败”状态的弹性裸金属可以执行停止操作。

说明

停止服务器为“强制关机”方式，会中断您的业务，请确保服务器上的文件已保存。

----结束

9.3.4.5 同步裸金属服务器状态

当用户在云服务器页面修改了裸金属服务器状态后，可同步其状态至ModelArts的弹性裸金属实例。

步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“专属资源池 > 弹性裸金属”，进入“弹性裸金属”列表页面。

步骤3 在弹性裸金属列表中，单击操作列的“同步”，在弹出的确认对话框中，确认信息无误，然后单击“确定”，完成同步操作。

----结束

9.3.4.6 删除实例

针对不再使用的弹性裸金属实例，可以删除以释放资源（处于“启动中”状态的弹性裸金属无法被删除）。

步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“专属资源池 > 弹性裸金属”，进入“弹性裸金属”列表页面。

步骤3 在弹性裸金属列表中，单击操作列的“删除”，在弹出的确认对话框中，确认信息无误，然后单击“确定”，完成删除操作。

----结束

9.3.5 管理员侧网络配置

背景信息

弹性裸金属创建后，需要联系管理员进行网络配置后，才可进行SSH访问。本章节介绍管理员侧的网络配置步骤，下述步骤需使用管理员账号进行操作。

前提条件

已完成**弹性裸金属实例创建**。

Step1：创建弹性 IP 与 NAT 网关

步骤1 登录管理控制台。

步骤2 在左侧服务列表中，单击“网络 > 弹性公网IP EIP”，进入“弹性公网IP”页面。

步骤3 在页面右上角，单击“创建弹性公网IP”。

步骤4 参数配置可使用默认值，单击“立即申请”。

步骤5 在网络控制台左侧列表中，单击“NAT网关 > 公网NAT网关”，进入公网NAT网关页面。

步骤6 在页面右上角，单击“创建公网NAT网关”。

步骤7 选择弹性裸金属所使用“虚拟私有云”和“子网”，其余参数配置可使用默认值，单击“立即创建”。

----结束

Step2：配置 SNAT 与 DNAT 规则

步骤1 在公网NAT网关页面，单击创建的NAT网关名称，进入NAT网关详情页。

步骤2 在“SNAT规则”页签下，单击“添加SNAT规则”。

步骤3 在弹出的“添加SNAT规则页面”，配置SNAT规则：

- 使用场景：选择“虚拟私有云”。
- 子网：选择“使用已有”，选择子网。
- 弹性公网IP：勾选创建的弹性公网IP。

步骤4 单击“确定”。

步骤5 在“DNAT规则页签”下，单击“添加DNAT规则”。

步骤6 在弹出的“添加DNAT规则页面”，配置DNAT规则：

- 使用场景：选择“虚拟私有云”。
- 端口类型：选择“具体端口”。
- 支持协议：选择“TCP”。
- 弹性公网IP：选择已创建的弹性公网IP。
- 公网端口：建议选择区间为20000-30000，保证该端口号不冲突。
- 私网IP：此处填写弹性裸金属的IP地址。可单击“查看可用云主机IP > 裸金属服务器”进行查看。
- 私网端口：端口号22。

步骤7 单击“确定”。

----结束

Step3：添加安全组规则

步骤1 登录管理控制台。

步骤2 选择“计算 > 裸金属服务器”。

步骤3 进入裸金属服务器页面。

步骤4 在裸金属服务器列表，单击待变更安全组规则的裸金属服务器名称。

步骤5 系统跳转至该裸金属服务器详情页面。

步骤6 选择“安全组”页签，并单击，查看安全组规则。

步骤7 单击安全组ID。系统自动跳转至安全组页面。

步骤8 单击操作列的“配置规则”，在安全组详情页根据安全组规则生效的方向添加规则。

在入方向规则，选择默认自定义TCP协议，在协议端口处填写**Step2：配置SNAT与DNAT规则**中的DNAT规则公网端口号。

图 9-22 添加入方向规则

说明

默认有配置端口22的安全组规则，可在安全组详情页查看。若无配置可参考上述步骤配置，协议端口填22。

----结束

9.4 资源监控

9.4.1 概述

ModelArts上报的所有指标都保存在AOM中，用户可以通过AOM服务提供的指标消费和使用的能力来进行指标消费。设置指标阈值告警、告警上报等，都可以直接在AOM控制台查看，也可以使用Grafana等可视化工具来查看与分析。Grafana支持灵活而又复杂多样的监控视图和模板，为用户提供基于网页仪表面板的可视化监控效果，使用户更加直观地查看到实时资源使用情况。

9.4.2 使用 Grafana 查看 AOM 中的监控指标

9.4.2.1 操作流程

Grafana支持灵活而又复杂多样的监控视图和模板，可以满足绝大部分情况下用户的诉求。将Grafana的数据源配置完成后，就可以通过Grafana查看AOM保存的所有ModelArts的所有指标。

通过Grafana插件查看AOM中的监控指标的操作流程如下：

1. **安装配置Grafana**

   说明

   安装配置Grafana有**在Windows上安装配置Grafana**、**在Linux上安装配置Grafana**和**在Notebook上安装配置Grafana**三种方式，请您根据实际情况选择。

2. **配置Grafana数据源**
3. **使用Grafana配置Dashboards，查看指标数据**

9.4.2.2 安装配置 Grafana

9.4.2.2.1 在 Windows 上安装配置 Grafana

适用场景

本章节适用于Windows操作系统的PC。

操作步骤

1. 下载Grafana安装包。

   进入**下载链接**，单击Download the installer，等待下载成功即可。

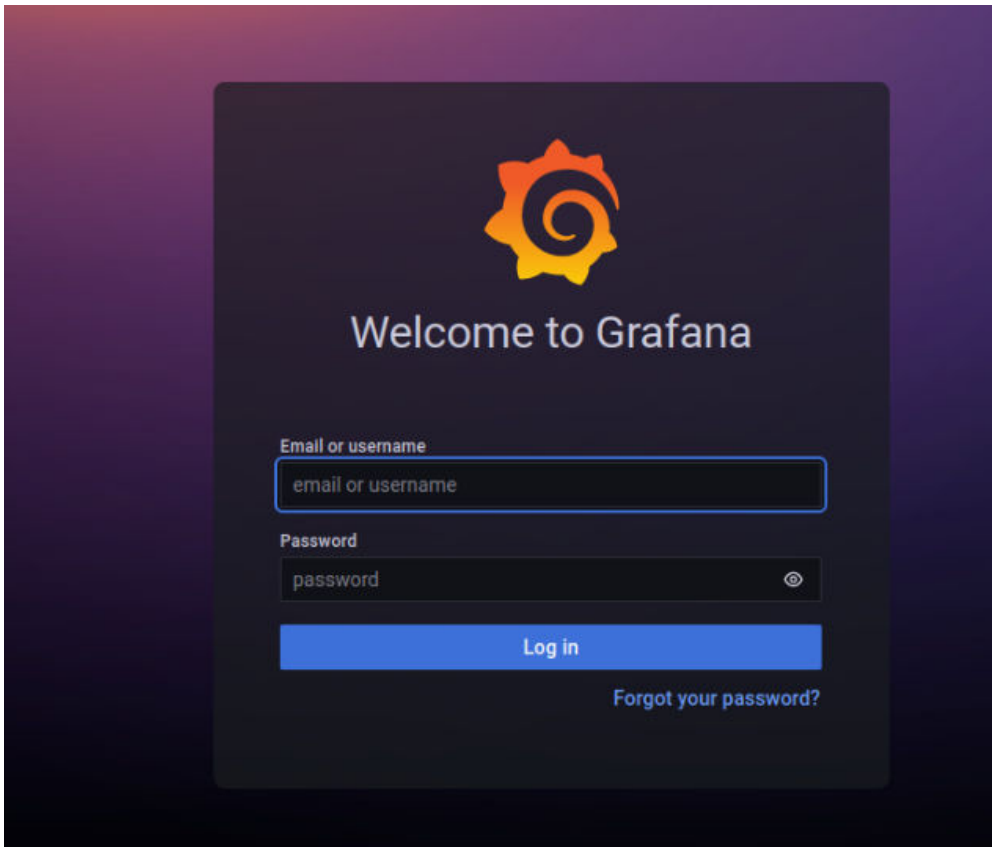
2. 安装Grafana。

   双击安装包，按照指示流程安装完成即可。

3. 在Windows的“服务”中，找到Grafana，将其开启，如果已经开启，则直接进入**4**

4. 登录Grafana。

   Grafana默认在本地的3000端口启动，打开链接http://localhost:3000，出现Grafana的登录界面。首次登录用户名和密码为admin，登录成功后请根据提示修改密码。



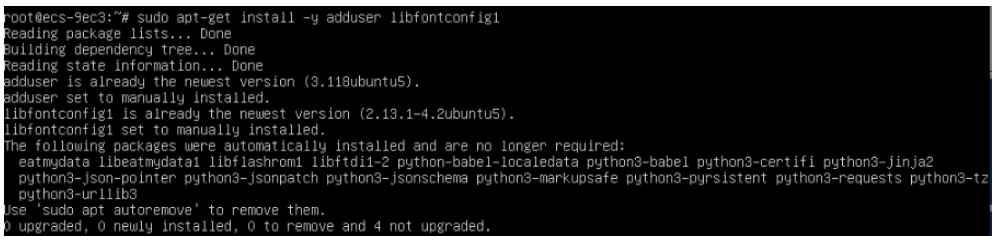
9.4.2.2.2 在 Linux 上安装配置 Grafana

前提条件

- 一台可访问外网的Ubuntu服务器。如果没有请具备以下条件：
- 准备一台ECS服务器（建议规格选8U或者以上，镜像选择Ubuntu，建议选择22.04版本，本地存储100G），具体操作请参考《弹性云服务器用户指南》> 快速入门 > 自定义购买弹性云服务器。
- 购买弹性公网IP，并绑定到购买的弹性云服务器ECS上，具体操作请参见《弹性公网IP用户指南》> 快速入门 > 为弹性云服务器申请和绑定公网IP。

操作步骤

1. 登录弹性云服务器。根据需要选择登录方式，具体操作请参考弹性云服务器用户指南 > 实例 > 登录Linux弹性云服务器> Linux弹性云服务器登录方式概述。
2. 执行如下命令安装libfontconfig1。
   ```
   sudo apt-get install -y adduser libfontconfig1
   ```
   回显如下代表执行成功：



3. 执行如下命令下载Grafana安装包。
   wget **https://dl.grafana.com/oss/release/grafana_9.3.6_amd64.deb** --no-check-certificate
   下载完成：

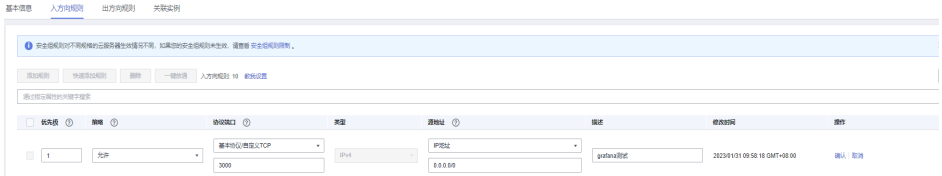
```
root@ecs-9ec3:~# wget https://dl.grafana.com/oss/release/grafana_9.3.6_amd64.deb --no-check-certificate
--2023-03-07 10:22:12--  https://dl.grafana.com/oss/release/grafana_9.3.6_amd64.deb
Resolving dl.grafana.com (dl.grafana.com)... 151.101.42.217
Connecting to dl.grafana.com (dl.grafana.com)|151.101.42.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89252050 (85M) [application/octet-stream]
Saving to: 'grafana_9.3.6_amd64.deb'

grafana_9.3.6_amd64.deb         100%[=====================================================>]  85.12M   379KB/s    in 2m 21s

2023-03-07 10:24:36 (617 KB/s) - 'grafana_9.3.6_amd64.deb' saved [89252050/89252050]
```

4. 执行如下命令安装Grafana。
   sudo dpkg -i grafana_9.3.6_amd64.deb

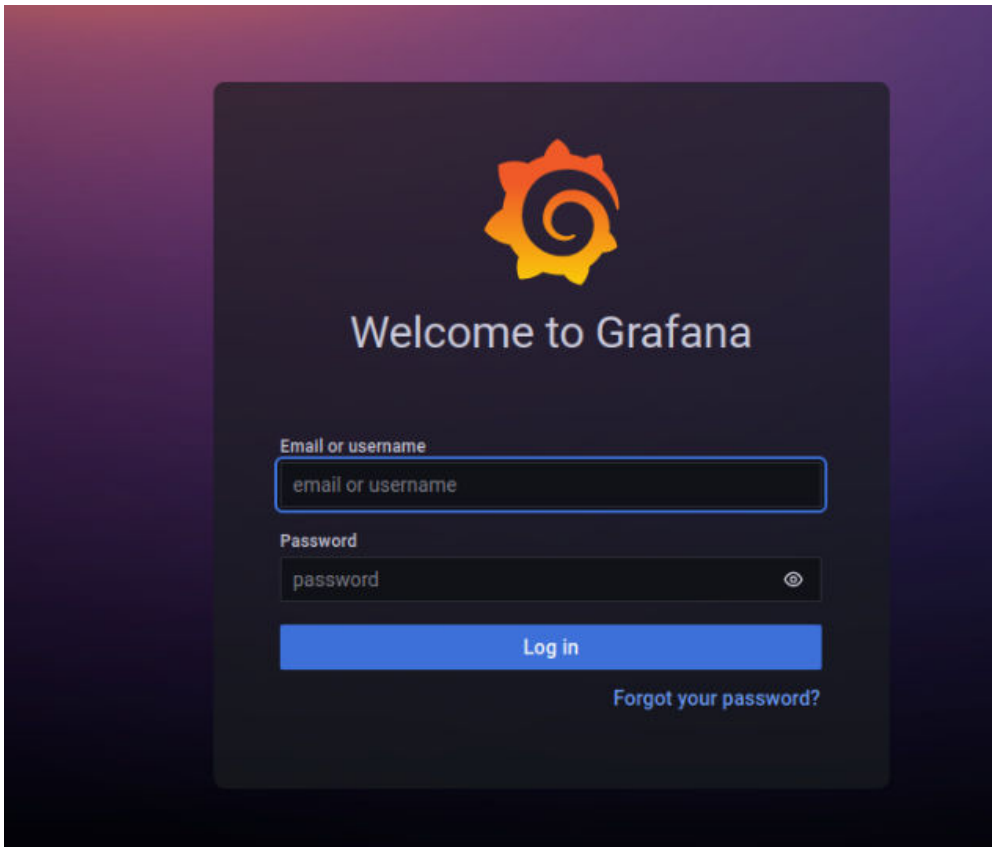
```
root@ecs-9ec3:~# sudo dpkg -i grafana_9.3.6_amd64.deb
Selecting previously unselected package grafana.
(Reading database ... 80788 files and directories currently installed.)
Preparing to unpack grafana_9.3.6_amd64.deb ...
Unpacking grafana (9.3.6) ...
Setting up grafana (9.3.6) ...
Adding system user `grafana' (UID 116) ...
Adding new user `grafana' (UID 116) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to configure grafana to start automatically using syst
emd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
 sudo /bin/systemctl start grafana-server
```

5. 执行命令启动Grafana。
   sudo /bin/systemctl start grafana-server
6. 在本地PC访问Grafana配置。
   确保ECS绑定了弹性公网IP，且对应安全组配置正确（入方向放开TCP协议的3000端口，出方向全部放通）。设置如下：
   a. 单击ECS服务器名称进入详情页，单击“安全组”页签，单击“配置规则”。

   b. 单击“入方向规则”，入方向放开TCP协议的3000端口，出方向默认全部放通。



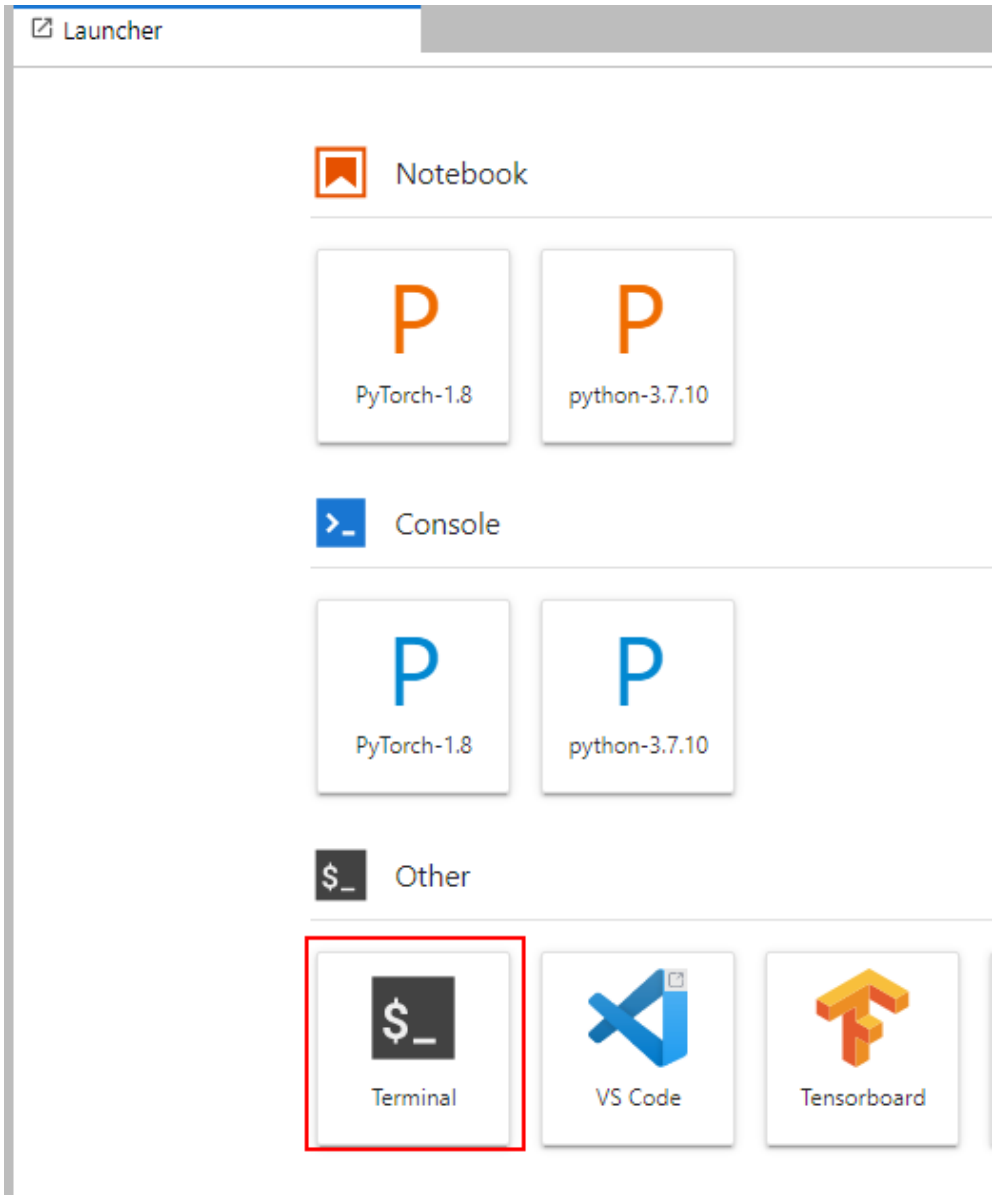
7. 在浏览器中输入“http://{弹性公网IP}:3000”，即可进行访问。首次登录用户名和密码为admin，登录成功后请根据提示修改密码。



9.4.2.2.3 在 Notebook 上安装配置 Grafana

前提条件

- 已创建CPU或GPU类型的Notebook实例，并处于运行中。
- 打开Terminal。



操作步骤

1. 在Terminal中依次执行以下命令，下载并安装Grafana。

```
mkdir -p /home/ma-user/work/grf
cd /home/ma-user/work/grf
wget https://dl.grafana.com/oss/release/grafana-9.1.6.linux-amd64.tar.gz
tar -zxvf grafana-9.1.6.linux-amd64.tar.gz
```

2. 将Grafana注册到jupyter-server-proxy。
   a. 在JupyterLab Terminal中执行以下命令：

```
mkdir -p /home/ma-user/.local/etc/jupyter
vi /home/ma-user/.local/etc/jupyter/jupyter_notebook_config.py
```

```
(PyTorch-1.8) [ma-user grf]$mkdir -p /home/ma-user/.local/etc/jupyter
(PyTorch-1.8) [ma-user grf]$vi /home/ma-user/.local/etc/jupyter/jupyter_notebook_config.py
```

b. 在打开的juputer_notebook_config.py中，增加以下代码后按ESC退出然后输入:wq保存。

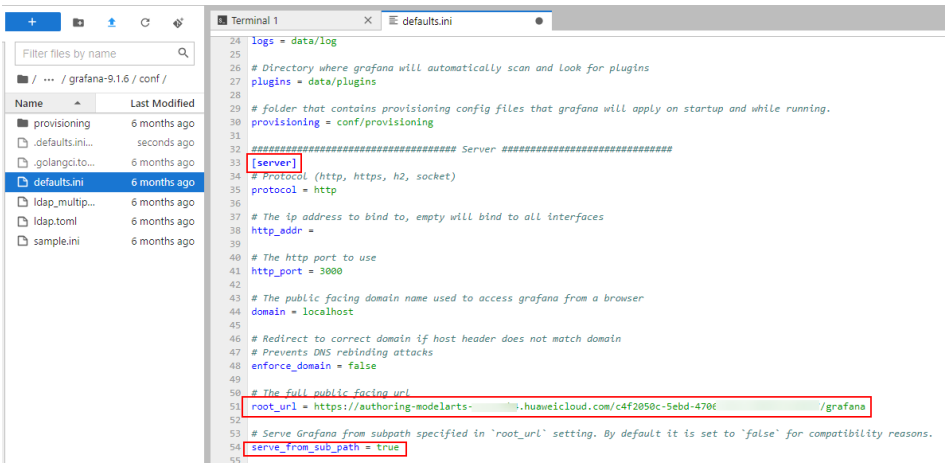
```
c.ServerProxy.servers = {
  'grafana': {
    'command': ['/home/ma-user/work/grf/grafana-9.1.6/bin/grafana-server', '--homepath', '/home/ma-user/work/grf/grafana-9.1.6', 'web'],
    'timeout': 1800,
    'port': 3000
  }
}
```

说明

如果“/home/ma-user/.local/etc/jupyter/jupyter_notebook_config.py”文件中已有“c.ServerProxy.servers”字段，新增对应的key-value键值对即可。

3. 适配JupyterLab访问地址。
   a. 在左侧导航打开“ vi /home/ma-user/work/grf/grafana-9.1.6/conf/defaults.ini”文件。
   b. 修改[server]中的“root_url”和“serve_from_sub_path”字段。

图 9-23 修改 defaults.ini 文件



其中：

- root_url的组成为：https:{jupyterlab域名}/{INSTANCE_ID}/grafana。域名和INSTANCE_ID可以从打开的jupyterLab页面地址栏获取，如下：

- Serve_from_sub_path设置为true

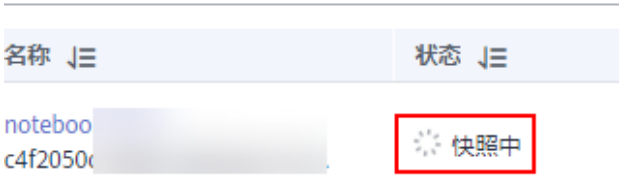
4. 保存Notebook镜像。
   a. 进入Notebook控制台，单击“开发环境 > Notebook”，在Notebook实例列表里找到对应的实例，选择“更多 > 保存镜像”。

b. 在保存镜像对话框中，设置组织、镜像名称、镜像版本和描述信息。单击“确定”保存镜像。

图 9-24 保存镜像

c. 镜像会以快照的形式保存，保存过程约5分钟，请耐心等待。此时不可再操作实例。

图 9-25 镜像保存中

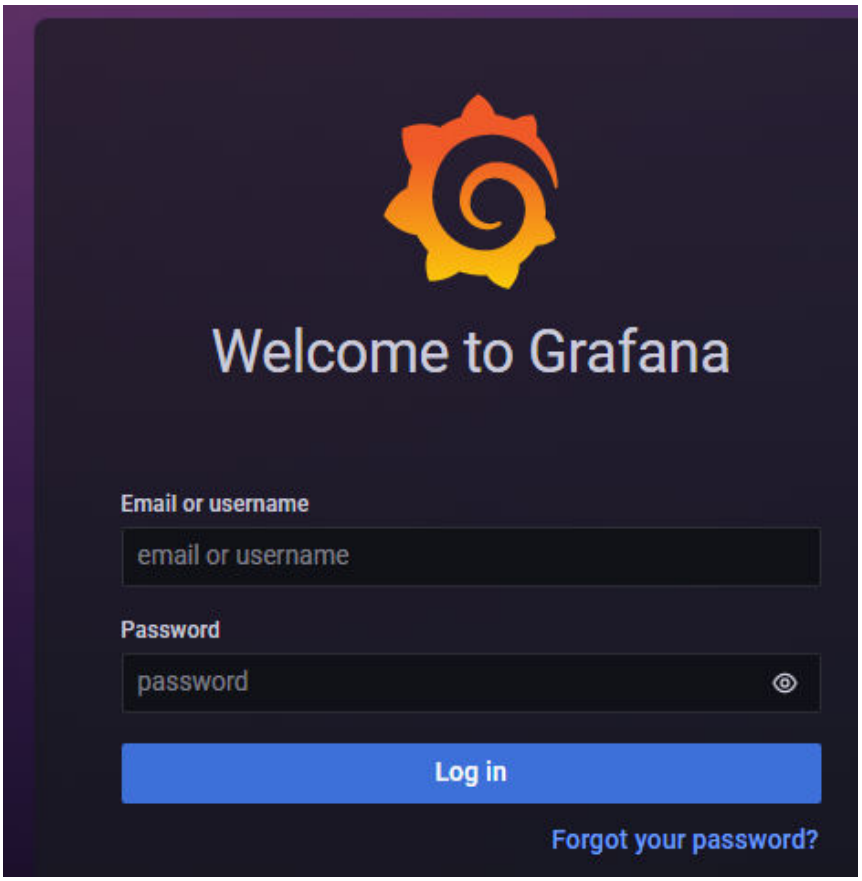


d. 镜像保存成功后，实例状态变为“运行中”，重启Notebook实例。

图 9-26 镜像保存成功

5. 打开Grafana页面。

   新打开一个浏览器窗口，在地址栏输入3中配置的root_url后。出现Grafana登录页面即代表在Notebook中安装和配置Grafana成功。首次登录用户名和密码为admin，登录成功后请根据提示修改密码。



9.4.2.3 配置 Grafana 数据源

在Grafana配置数据源后，即可通过Grafana查看ModelArts的监控数据。

前提条件

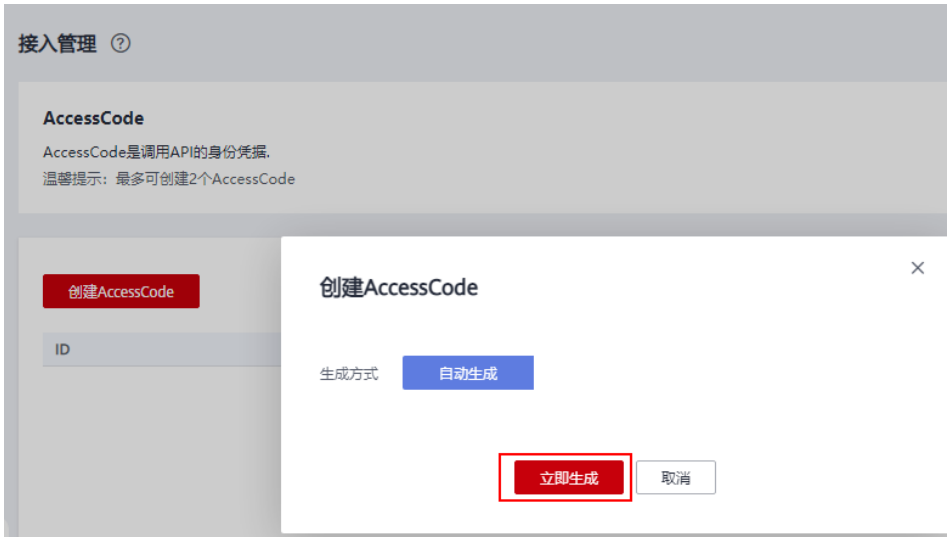
- 已**安装Grafana**。

配置 Grafana 数据源

1. 添加AccessCode
   a. 进入AOM管理控制台。

   b. 在左侧导航栏中选择“配置管理 > 接入管理”，单击“创建AccessCode”，参考下图生成AccessCode。

图 9-27 生成 AccessCode



c. 单击即可查看AccessCode，记录生成的AccessCode。

图 9-28 查看 AccessCode

2. 获取数据源URL。

   在Grafana中需要配置数据源URL，URL的组成是：https://{Endpoint}/v1/{project_id}

   - 不同服务不同区域的终端节点不同，请向企业管理员获取区域和终端节点信息。

     各服务的Endpoint信息由服务名、Region ID和外部域名三部分组成，格式为“{service_name}.{region_id}.{external_domain_name}”。其中，每一项参数的获取方式参见**表9-9**。

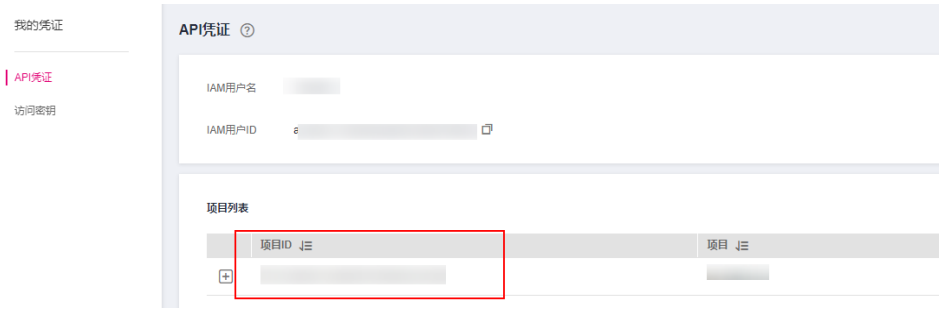
     表 9-9 Endpoint 信息的获取方式

参数	描述	获取方式
service_name	服务名缩写（不区分大小写）	AOM服务默认为“aom”。
region_id	Region ID	联系系统管理员获取。
external_domain_name	外部域名后缀	联系系统管理员获取。

   - project_id需填写对应region的项目id，通过“我的凭证”获取。

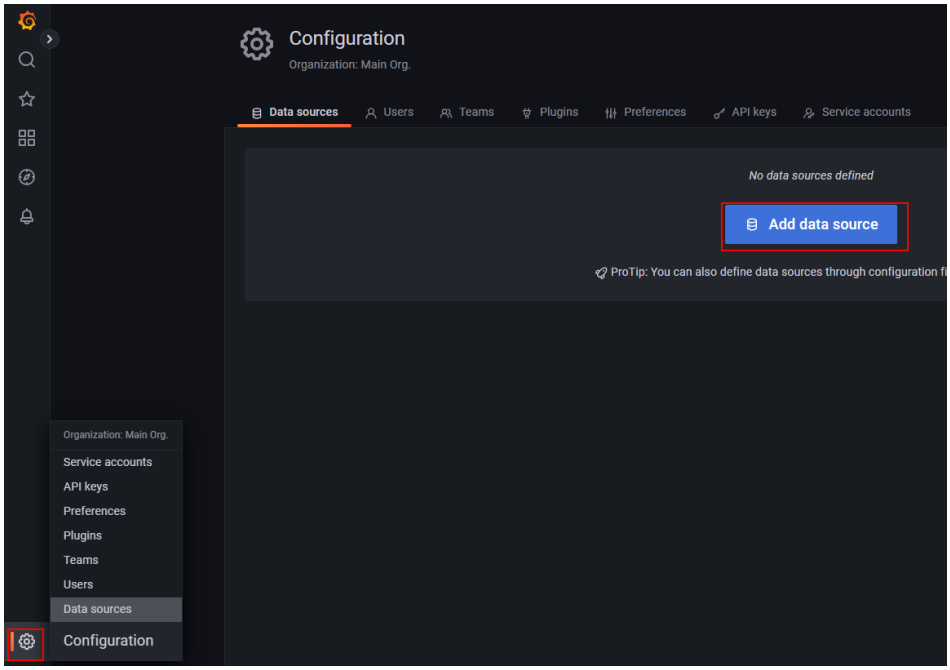
图 9-29 进入我的凭证

图 9-30 获取项目 ID



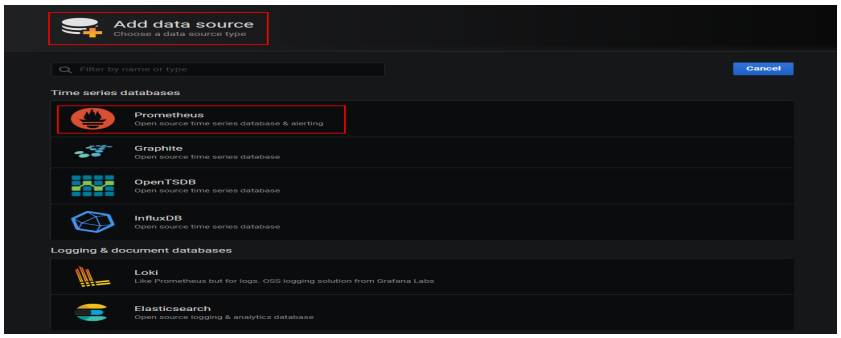
3. 在Grafana中增加数据源。
   a. 登录Grafana。首次登录用户名和密码为admin，登录成功后可根据提示修改密码。
   b. 在左侧菜单栏，选择“Configuration > Data Sources”，单击“Add data source”。

图 9-31 配置 Grafana



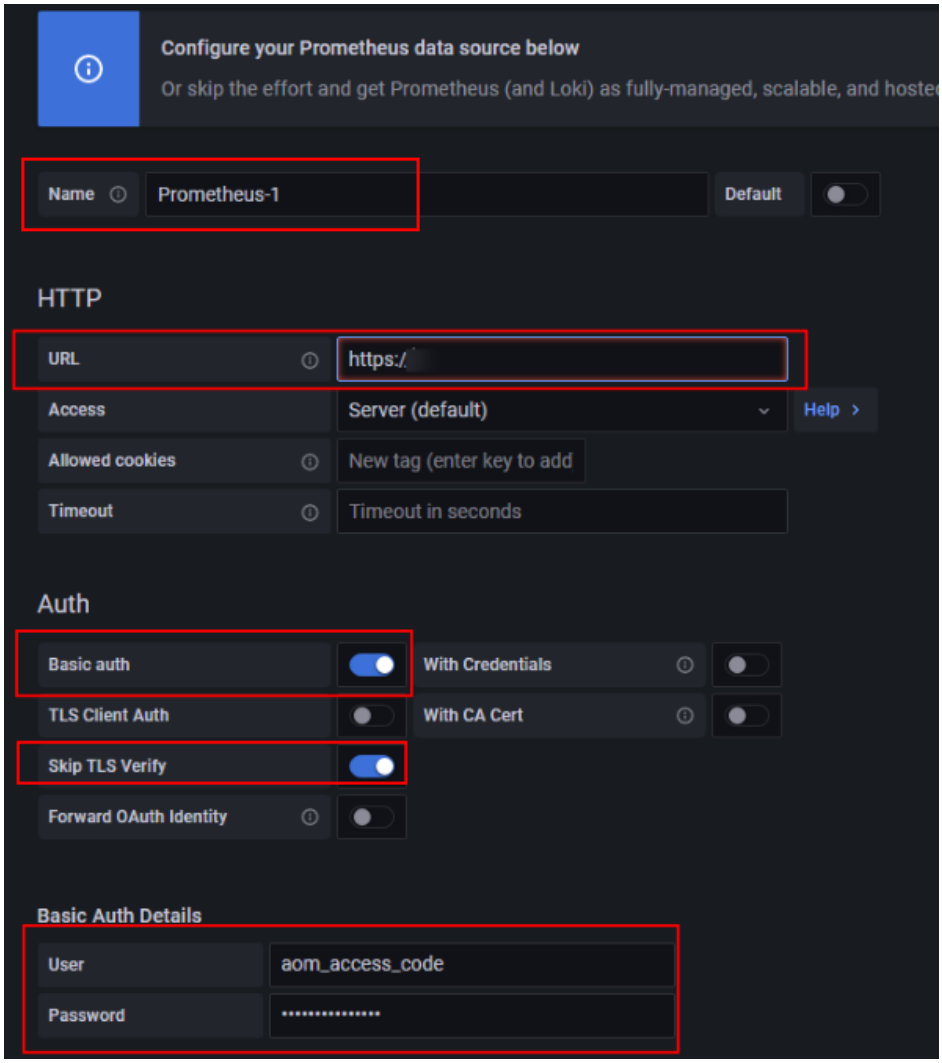
c. 单击“Prometheus”，进入Prometheus配置页面。

图 9-32 进入 Prometheus 配置页面



d. 参考下图进行配置。

图 9-33 配置 Grafana 数据源



说明

Grafana安装方式不同，Grafana版本也可能不同，图9-33仅为示例，请以实际配置界面为准。

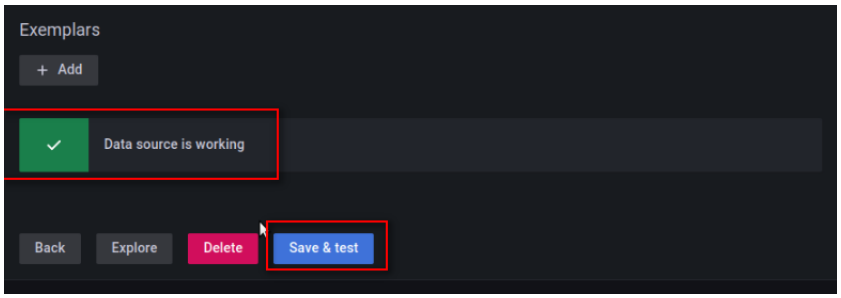
表 9-10 参数配置说明

参数名称	配置说明
Name	自定义。
URL	2.获取数据源URL。中拼接的URL：https://{Endpoint}/v1/{project_id}
Basic auth	开启
Skip TLS Verify	开启
User	aom_access_code

参数名称	配置说明
Password	1.添加AccessCode中生成的AccessCode

e. 配置完成后，单击下方的“Save & test”，展示“Data source is working”代表配置数据源成功。

图 9-34 配置数据源成功



9.4.2.4 使用 Grafana 配置 Dashboards，查看指标数据

Grafana中可以自定义配置各种视图的仪表盘，ModelArts也提供了针对集群的配置模板。本章节分别以使用ModelArts提供的模板查看指标和创建Dashboards的方式，说明如何进行仪表盘配置。Grafana的更多使用请参考Grafana官方文档。

准备工作

ModelArts提供了集群视图，节点视图，用户视图，任务视图，任务详细视图这5个模板，这些模板在Grafana官方文档可以搜索下载，您导入模板配置Dashboards时，可直接使用。

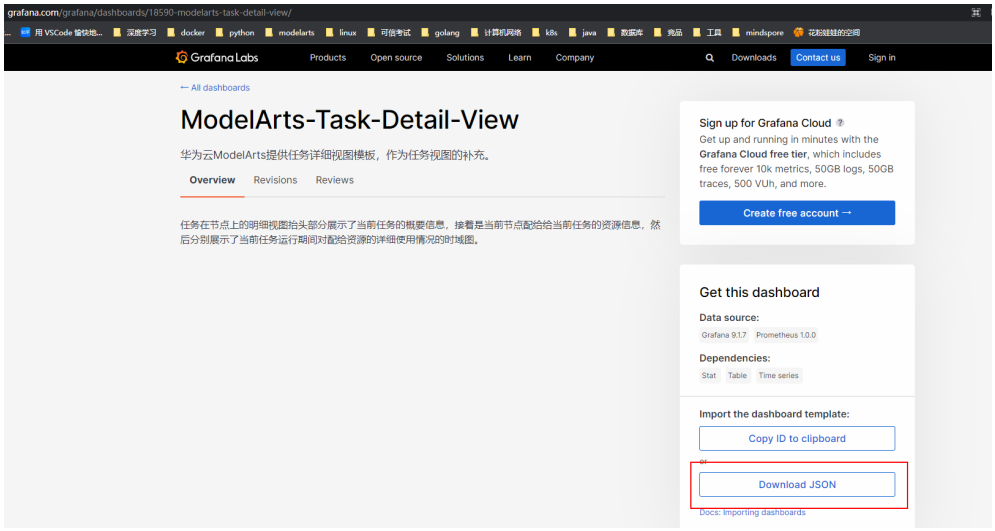
表 9-11 模板下载地址

模板名称	下载地址
集群视图	https://grafana.com/grafana/dashboards/18582-modelarts-cluster-view/
节点视图	https://grafana.com/grafana/dashboards/18583-modelarts-node-view/
用户视图	https://grafana.com/grafana/dashboards/18588-modelarts-user-view/
任务视图	https://grafana.com/grafana/dashboards/18604-modelarts-task-view/
任务详细视图	https://grafana.com/grafana/dashboards/18590-modelarts-task-detail-view/

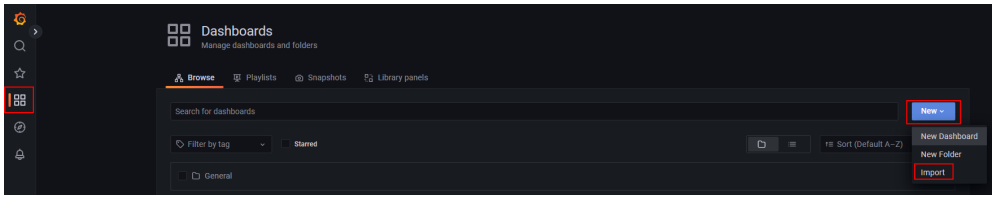
使用 ModelArts 提供的模板查看指标

1. （可选）**准备工作**展示了所有模板的下载地址，选择需要使用的模板，打开链接，单击“Download JSON”下载JSON文件，如下图。

   图 9-35 下载任务详情视图模板

2. 打开“DashBoards”，选择“New”>“Import”。

3. 导入DashBoards模板，有两种方式：
   - 方式一：上传**1**中下载的JSON文件，如**图9-36**所示。
   - 方式二：复制**准备工作**提供的模板的下载地址，单击“Load”，如**图9-37**所示。

图 9-36 上传 JSON 文件导入 DashBoards 模板

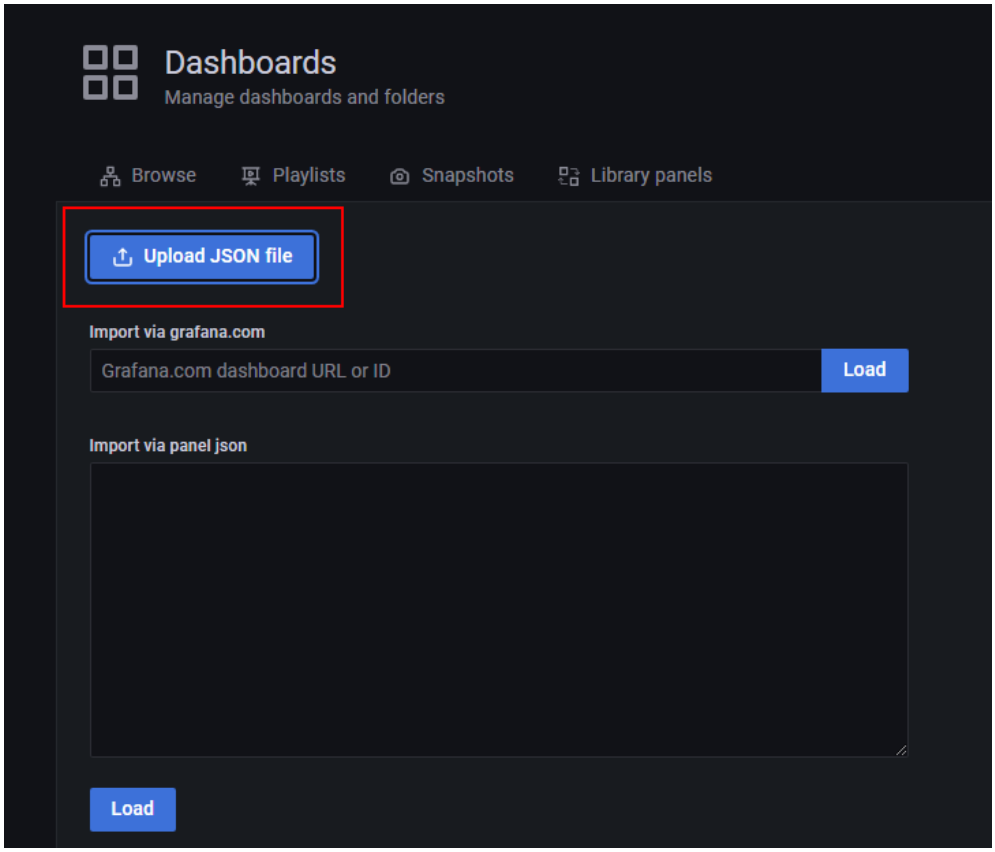
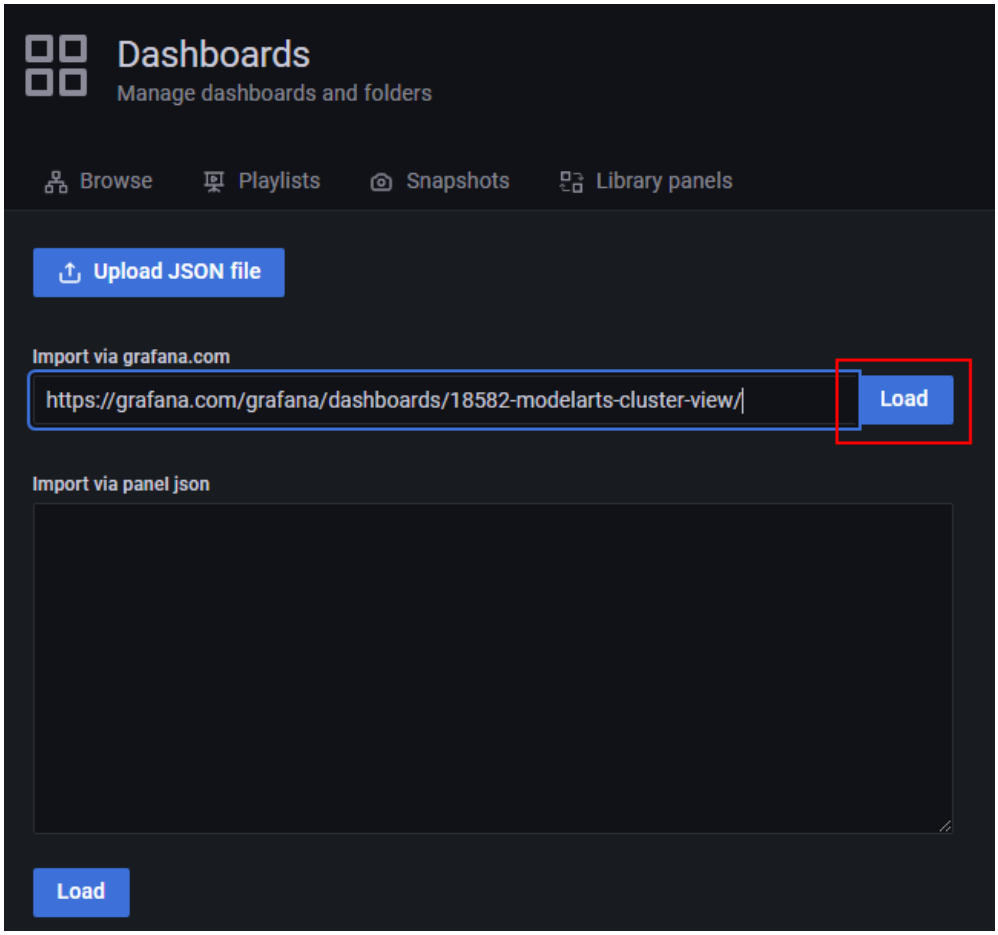
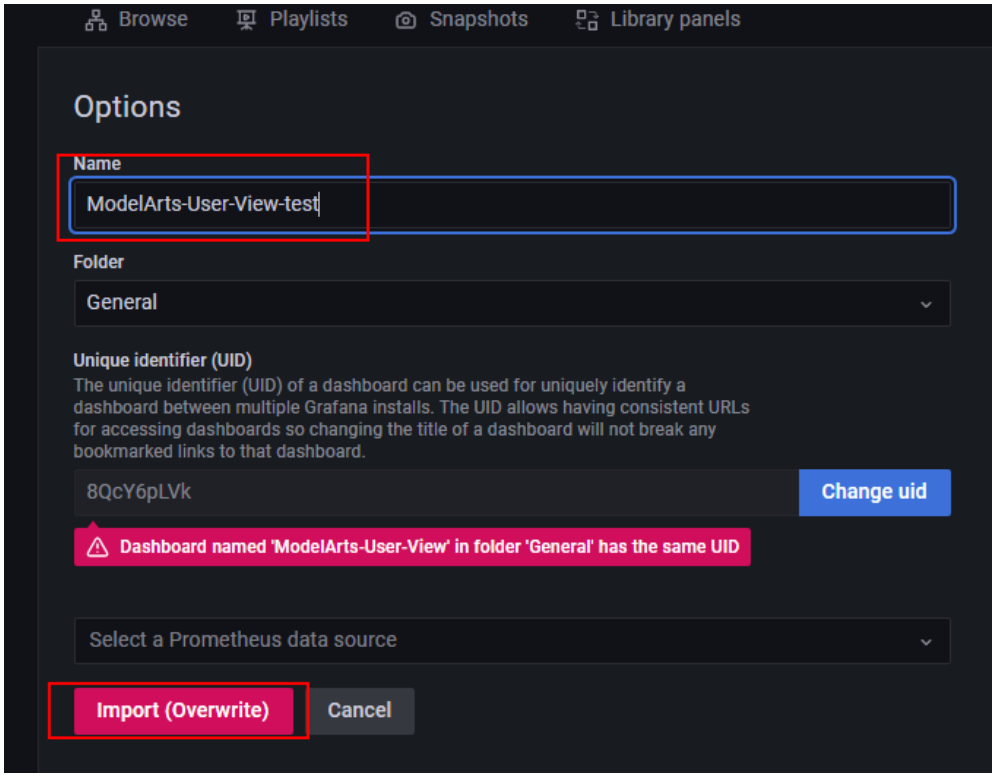


图 9-37 复制模板地址导入 DashBoards 模板



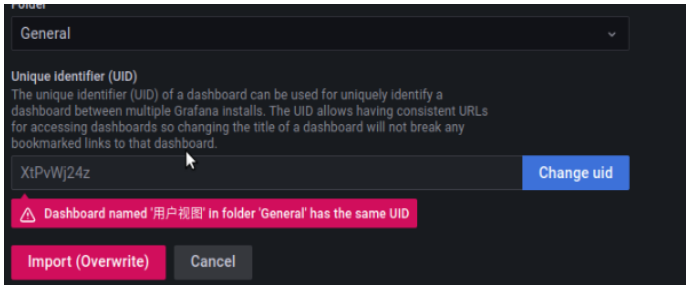
4. 修改视图名称，单击import。

图 9-38 修改视图名称

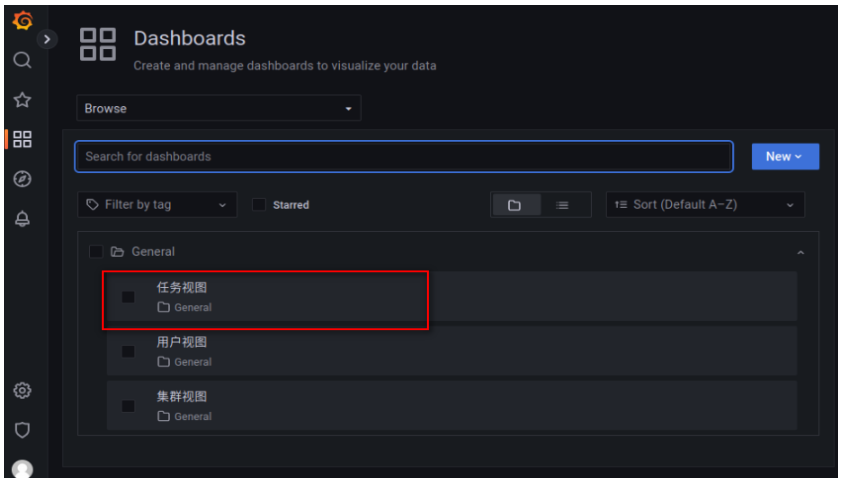


注意：如提示uid重复，则修改下json中的uid后单击“Import”。

图 9-39 修改 uid



5. 导入成功后，在Dashboards下，即可看到导入的视图，单击视图即可打开监控。

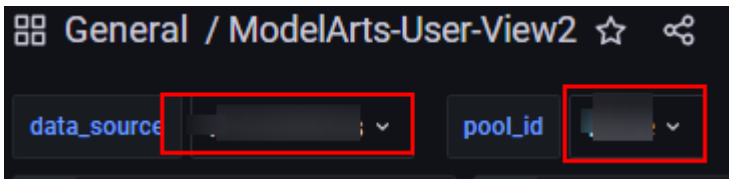


6. 模板使用

   导入成功后，单击想查看的模板即可查看响应内容。这里介绍一些常用功能的使用。

   - 切换数据源和资源池

     图 9-40 切换数据源和资源池

     单击红框中相应位置，即可出现下拉框，修改响应的数据源和资源池。

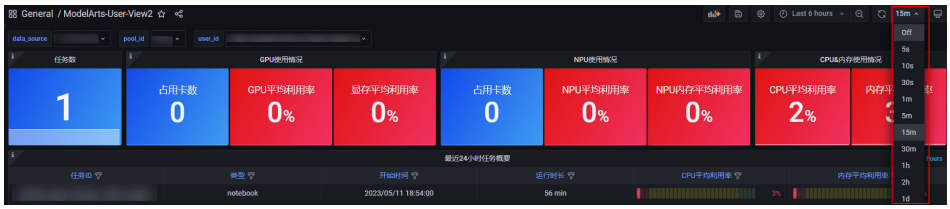
   - 刷新数据

     单击右上角的图标，即可刷新整个DashBoard的所有数据，各panel也会更新

   - 修改自动刷新时间

     图 9-41 修改自动刷新时间

     模板的默认刷新时间是15分钟，如果觉得该时间不合适，可在右上角下拉选择修改，修改后，单击保存即可生效。

   - 修改DashBoard查询数据时间范围

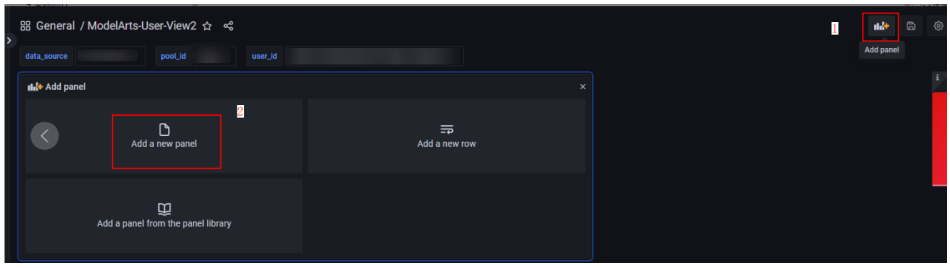
     图 9-42 修改查询数据时间范围

单击右上角图标，即可修改DashBoard整体的数据查询时间。除固定查询时间外的其他panel，都会应用该数据查询时间范围。

- 增加新panel

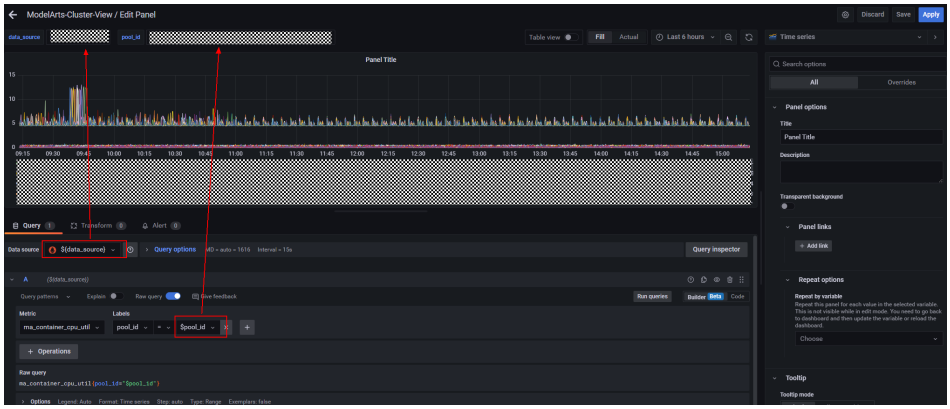
图 9-43 新增一个 panel



单击右上角的'+'图标，即可新增一个panel。

新增一个panel后，即可在其中查询相应的数据。将数据源和资源池进行如下的相应选择，即可应用当前DashBoard的对应配置。

图 9-44 使用当前 DashBoard 的配置



创建 Dashboards 查看指标

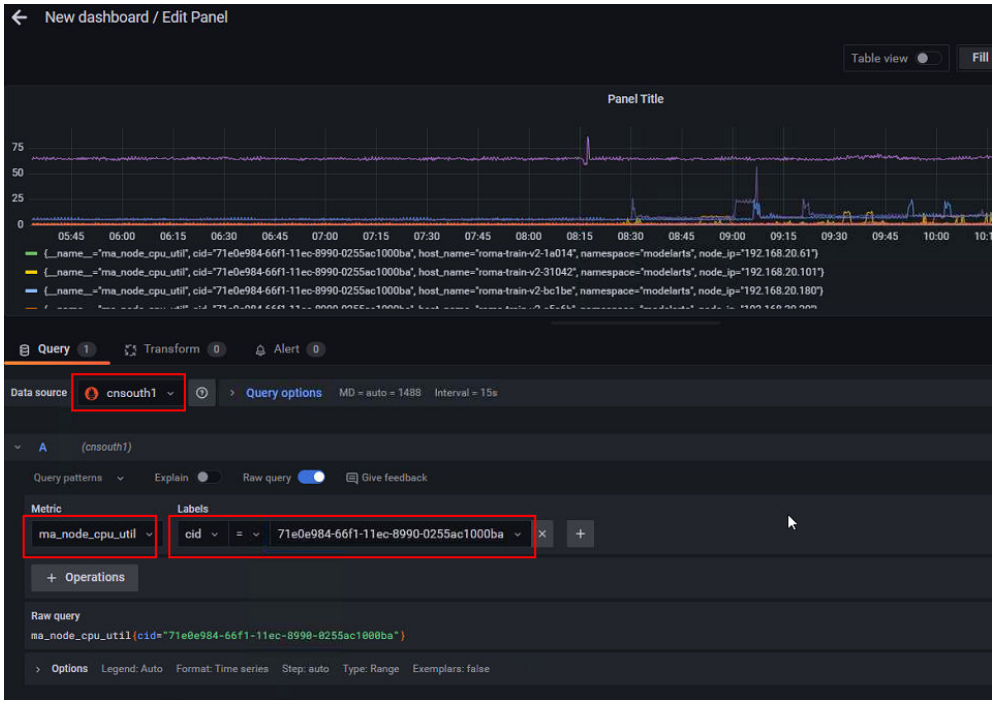
1. 打开“DashBoards”，单击“New”，选择“New Dashboards”。
2. 在New Dashboards界面，单击“Add a new panel”。
3. 在New dashboard /Edit Panel界面，填写如下参数。

   Data source：**已配置Grafana数据源**；

   Metric：指标名称，可参考**表9-12**、**表9-13**、**表9-14**获取想要查询的指标；

   Labels：填写过滤该指标的标签，请参考**表9-15**。

图 9-45 创建 Dashboards 查看指标



9.4.3 在 AOM 控制台查看 ModelArts 所有监控指标

ModelArts会定期收集资源池中各节点的关键资源（GPU、NPU、CPU、Memory等）的使用情况以及开发环境、训练作业、推理服务的关键资源的使用情况，并上报到AOM，用户可直接在AOM上查看，详细步骤如下：

1. 登录控制台，搜索AOM，进入“应用运维管理 AOM”控制台。
2. 单击“指标浏览”，进入“指标浏览”“页面”，单击“添加指标查询”。
3. 添加指标查询信息，单击添加到指标列表。
   - 添加方式：选择“全量指标”。
   - 指标名称：选择想要查询的指标，参考**表9-12**、**表9-13**、**表9-14**
   - 范围：填写过滤该指标的标签，请参考**表4的Label名字栏**。样例如下：
4. 即可出现指标信息。

表 9-12 容器级别的指标

分类	名称	指标	指标含义	单位	取值范围
CPU	CPU使用率	ma_container_cpu_util	该指标用于统计测量对象的CPU使用率。	百分比（Percent）	0～100%
	CPU内核占用量	ma_container_cpu_used_core	该指标用于统计测量对象已经使用的CPU核个数	核（Core）	≥0
	CPU内核总量	ma_container_cpu_limit_core	该指标用于统计测量对象申请的CPU核总量。	核（Core）	≥1
内存	内存总量	ma_container_memory_capacity_megabytes	该指标用于统计测量对象申请的物理内存总量。	兆字节（Megabytes）	≥0
	物理内存使用率	ma_container_memory_util	该指标用于统计测量对象已使用内存占申请物理内存总量的百分比。	百分比（Percent）	0～100%
	物理内存使用量	ma_container_memory_used_megabytes	该指标用于统计测量对象实际已经使用的物理内存，对应container_memory_working_set_bytes当前内存工作集（working set）使用量。 工作区内存使用量=活跃的匿名页和缓存，以及file-baked页<=container_memory_usage_bytes。	兆字节（Megabytes）	≥0
存储	磁盘读取速率	ma_container_disk_read_kilobytes	该指标用于统计每秒从磁盘读出的数据量。	千字节/秒（Kilobytes/Second）	≥0

分类	名称	指标	指标含义	单位	取值范围
	磁盘写入速率	ma_container_disk_write_kilobytes	该指标用于统计每秒写入磁盘的数据量。	千字节/秒（Kilobytes/Second）	≥0
GPU显存	显存容量	ma_container_gpu_mem_total_megabytes	该指标用于统计训练任务的显存容量。	兆字节（Megabytes）	>0
	显存使用率	ma_container_gpu_mem_util	该指标用于统计测量对象已使用的显存占显存容量的百分比。	百分比（Percent）	0～100%
	显存使用量	ma_container_gpu_mem_used_megabytes	该指标用于统计测量对象已使用的显存。	兆字节（Megabytes）	≥0
GPU	GPU使用率	ma_container_gpu_util	该指标用于统计测量对象的GPU使用率。	百分比（Percent）	0～100%
	GPU内存带宽利用率	ma_container_gpu_mem_copy_util	表示内存带宽利用率。以英伟达GPU V100为例，其最大内存带宽为900 GB/sec，如果当前的内存带宽为450 GB/sec，则内存带宽利用率为50%。	百分比（Percent）	0～100%
	GPU编码器利用率	ma_container_gpu_enc_util	表示编码器利用率	百分比（Percent）	%
	GPU解码器利用率	ma_container_gpu_dec_util	表示解码器利用率	百分比（Percent）	%
	GPU温度	DCGM_FI_DEV_GPU_TEMP	表示GPU温度。	摄氏度（℃）	自然数
	GPU功率	DCGM_FI_DEV_POWER_USAGE	表示GPU功率。	瓦特（W）	>0

分类	名称	指标	指标含义	单位	取值范围
	显存温度	DCGM_FI_DEV_MEMORY_TEMP	表示显存温度。	摄氏度（℃）	自然数
网络IO	下行Bps	ma_container_network_receive_bytes	该指标用于统计测试对象的入方向网络流速。	字节/秒（Bytes/Second）	≥0
	下行Pps	ma_container_network_receive_packets	每秒网卡接收的数据包个数。	个/秒（Packets/Second）	≥0
	下行错包率	ma_container_network_receive_error_packets	每秒网卡接收的错误包个数。	个/秒（Packets/Second）	≥0
	上行Bps	ma_container_network_transmit_bytes	该指标用于统计测试对象的出方向网络流速。	字节/秒（Bytes/Second）	≥0
	上行错包率	ma_container_network_transmit_error_packets	每秒网卡发送的错误包个数。	个/秒（Packets/Second）	≥0
	上行Pps	ma_container_network_transmit_packets	每秒网卡发送的数据包个数。	个/秒（Packets/Second）	≥0
NPU	NPU使用率	ma_container_npu_util	该指标用于统计测量对象的NPU使用率。（即将废止，替代指标为ma_container_npu_ai_core_util）。	百分比（Percent）	0～100%

分类	名称	指标	指标含义	单位	取值范围
	NPU显存使用率	ma_container_npu_memory_util	该指标用于统计测量对象已使用的NPU显存占NPU存储容量的百分比。（即将废止，Snt3系列替代指标为ma_container_npu_ddr_memory_util，Snt9系列替代指标为ma_container_npu_hbm_util）。	百分比（Percent）	0～100%
	NPU显存使用量	ma_container_npu_memory_used_megabytes	该指标用于统计测量对象已使用的NPU显存。（即将废止， Snt3系列替代指标为ma_container_npu_ddr_memory_usage_bytes，Snt9系列替代指标为ma_container_npu_hbm_usage_bytes）。	≥0	兆字节（Megabytes）
	NPU显存容量	ma_container_npu_memory_total_megabytes	该指标用于统计测量对象的NPU显存容量。（即将废止， Snt3系列替代指标为ma_container_npu_ddr_memory_bytes，Snt9系列替代指标为ma_container_npu_hbm_bytes）。	>0	兆字节（Megabytes）
	AI处理器错误码	ma_container_npu_ai_core_error_code	昇腾系列AI处理器错误码	-	-
	AI处理器健康状态	ma_container_npu_ai_core_health_status	昇腾系列AI处理器健康状态	-	• 1：健康 • 0：不健康

分类	名称	指标	指标含义	单位	取值范围
	AI处理器功耗	ma_container_npu_ai_core_power_usage_watts	昇腾系列AI处理器功耗	瓦特（W）	>0
	AI处理器温度	ma_container_npu_ai_core_temperature_celsius	昇腾系列AI处理器温度	摄氏度（℃）	自然数
	AI处理器AICORE利用率	ma_container_npu_ai_core_util	昇腾系列AI处理器AI Core利用率	百分比（Percent）	0～100%
	AI处理器AICORE时钟频率	ma_container_npu_ai_core_frequency_hertz	昇腾系列AI处理器AI Core时钟频率	赫兹（Hz）	>0
	AI处理器电压	ma_container_npu_ai_core_voltage_volts	昇腾系列AI处理器电压	伏特（V）	自然数
	AI处理器DDR内存总量	ma_container_npu_ddr_memory_bytes	昇腾系列AI处理器DDR内存总量	字节（Byte）	>0
	AI处理器DDR内存使用量	ma_container_npu_ddr_memory_usage_bytes	昇腾系列AI处理器DDR内存使用量	字节（Byte）	>0
	AI处理器DDR内存利用率	ma_container_npu_ddr_memory_util	昇腾系列AI处理器DDR内存利用率	百分比（Percent）	0～100%
	AI处理器HBM内存总量	ma_container_npu_hbm_bytes	昇腾系列AI处理器HBM总内存（Snt9 AI处理器专属）	字节（Byte）	>0
	AI处理器HBM内存使用量	ma_container_npu_hbm_usage_bytes	昇腾系列AI处理器HBM内存使用量（Snt9 AI处理器专属）	字节（Byte）	>0

分类	名称	指标	指标含义	单位	取值范围
	AI处理器HBM内存利用率	ma_container_npu_hbm_util	昇腾系列AI处理器HBM内存利用率（Snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存带宽利用率	ma_container_npu_hbm_bandwidth_util	昇腾系列AI处理器HBM内存带宽利用率（Snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存时钟频率	ma_container_npu_hbm_frequency_hertz	昇腾系列AI处理器HBM内存时钟频率（Snt9 AI处理器专属）	赫兹（Hz）	>0
	AI处理器HBM内存温度	ma_container_npu_hbm_temperature_celsius	昇腾系列AI处理器HBM内存温度（Snt9 AI处理器专属）	摄氏度（℃）	自然数
	AI处理器AI CPU利用率	ma_container_npu_ai_cpu_util	昇腾系列AI处理器AI CPU利用率	百分比（Percent）	0～100%
	AI处理器控制CPU利用率	ma_container_npu_ctrl_cpu_util	昇腾系列AI处理器控制CPU利用率	百分比（Percent）	0～100%
NPU RoCE网络	NPU RoCE网络上行速率	ma_container_npu_roce_tx_rate_bytes_per_second	容器所使用的NPU网络模块上行速率	字节/秒（Bytes/Second）	≥0
	NPU RoCE网络下行速率	ma_container_npu_roce_rx_rate_bytes_per_second	容器所使用的NPU网络模块下行速率	字节/秒（Bytes/Second）	≥0
Notebook业务指标	Notebook cache目录大小	ma_container_notebook_cache_dir_size_bytes	GPU和NPU类型的Notebook会在“/cache”目录上挂载一块高速本地磁盘，该指标描述该目录的总大小。	字节（Bytes）	≥0

分类	名称	指标	指标含义	单位	取值范围
	Notebook cache目录利用率	ma_container_notebook_cache_dir_util	GPU和NPU类型的Notebook会在“/cache”目录上挂载一块高速本地磁盘，该指标描述该目录的利用率。	百分比（Percent）	0～100%

表 9-13 节点指标（仅专属池上会收集）

分类	名称	指标	指标含义	单位	取值范围
CPU	CPU内核总量	ma_node_cpu_limit_core	该指标用于统计测量对象申请的CPU核总量。	核（Core）	≥1
	CPU内核占用	ma_node_cpu_used_core	该指标用于统计测量对象已经使用的CPU核数。	核（Core）	≥0
	CPU使用率	ma_node_cpu_util	该指标用于统计测量对象的CPU使用率。	百分比（Percent）	0～100%
	CPU IO等待时间	ma_node_cpu_iowait_counter	从系统启动开始累计到当前时刻，硬盘IO等待时间	jiffies	≥0
内存	物理内存使用率	ma_node_memory_util	该指标用于统计测量对象已使用内存占申请物理内存总量的百分比。	百分比（Percent）	0～100%
	物理内存容量	ma_node_memory_total_megabytes	该指标用于统计测量申请的物理内存总量。	兆字节（Megabytes）	≥0

分类	名称	指标	指标含义	单位	取值范围
网络IO	下行Bps	ma_node_network_receive_rate_bytes_seconds	该指标用于统计测试对象的入方向网络流速。	字节/秒（Bytes/Second）	≥0
	上行Bps	ma_node_network_transmit_rate_bytes_seconds	该指标用于统计测试对象的出方向网络流速。	字节/秒（Bytes/Second）	≥0
存储	磁盘读取速率	ma_node_disk_read_rate_kilobytes_seconds	该指标用于统计每秒从磁盘读出的数据量。只考虑被容器使用的数据盘。	千字节/秒（Kilobytes/Second）	≥0
	磁盘写入速率	ma_node_disk_write_rate_kilobytes_seconds	该指标用于统计每秒写入磁盘的数据量。只考虑被容器使用的数据盘。	千字节/秒（Kilobytes/Second）	≥0
	cache空间的总量	ma_node_cache_space_capacity_megabytes	该指标用于统计k8s空间的总容量。	兆字节（Megabytes）	≥0
	cache空间的使用量	ma_node_cache_space_used_capacity_megabytes	该指标用于统计k8s空间的使用量。	兆字节（Megabytes）	≥0
	容器空间的总量	ma_node_container_space_capacity_megabytes	该指标用于统计容器空间的总容量。	兆字节（Megabytes）	≥0

分类	名称	指标	指标含义	单位	取值范围
	容器空间的使用量	ma_node_container_space_used_capacity_megabytes	该指标用于统计容器空间的使用量。	兆字节（Megabytes）	≥0
	磁盘信息	ma_node_disk_info	该指标用于展示磁盘的基础信息	-	≥0
	读取次数	ma_node_disk_reads_completed_total	成功完成的读取总次数	-	≥0
	合并读取的次数	ma_node_disk_reads_merged_total	合并读取的次数	-	≥0
	读取字节数	ma_node_disk_read_bytes_total	成功读取的总字节数	字节（Bytes）	≥0
	读取花费秒数	ma_node_disk_read_time_seconds_total	所有读取所花费的总秒数	秒（Seconds）	≥0
	写入次数	ma_node_disk_writes_completed_total	成功完成的写入总数	-	≥0
	合并写入的次数	ma_node_disk_writes_merged_total	合并写入的次数	-	≥0
	写入字节数	ma_node_disk_written_bytes_total	成功写入的总字节数	字节（Bytes）	≥0
	写入花费秒数	ma_node_disk_write_time_seconds_total	所有写入操作花费的总秒数	秒（Seconds）	≥0

分类	名称	指标	指标含义	单位	取值范围
	当前IO数量	ma_node_disk_io_now	当前正在进行的I/O数量	-	≥0
	IO花费总秒数	ma_node_disk_io_time_seconds_total	执行I/O所花费的总秒数	秒（Seconds）	≥0
	IO花费加权秒数	ma_node_disk_io_time_weighted_seconds_tota	执行I/O所花费的加权秒数	秒（Seconds）	≥0
GPU	GPU使用率	ma_node_gpu_util	该指标用于统计测量对象的GPU使用率。	百分比（Percent）	0～100%
	显存容量	ma_node_gpu_mem_total_megabytes	该指标用于统计测量对象的显存容量。	兆字节（Megabytes）	>0
	显存使用率	ma_node_gpu_mem_util	该指标用于统计测量对象已使用的显存占显存容量的百分比。	百分比（Percent）	0～100%
	显存使用量	ma_node_gpu_mem_used_megabytes	该指标用于统计测量对象已使用的显存。	兆字节（Megabytes）	≥0
	共享GPU任务运行数据	node_gpu_share_job_count	针对一个GPU卡，当前运行的共享资源使用的任务数量。	个	≥0
	GPU温度	DCGM_FI_DEV_GPU_TEMP	表示GPU温度。	摄氏度（℃）	自然数

分类	名称	指标	指标含义	单位	取值范围
	GPU功率	DCGM_FI_DEV_POWER_USAGE	表示GPU功率。	瓦特（W）	>0
	显存温度	DCGM_FI_DEV_MEMORY_TEMP	表示显存温度。	摄氏度（℃）	自然数
NPU	NPU使用率	ma_node_npu_util	该指标用于统计测量对象的NPU使用率。（即将废止，替代指标为ma_node_npu_ai_core_util）。	百分比（Percent）	0～100%
	NPU显存使用率	ma_node_npu_memory_util	该指标用于统计测量对象已使用的NPU显存占NPU存储容量的百分比。（即将废止， Snt3系列替代指标为ma_node_npu_ddr_memory_util，Snt9系列替代指标为ma_node_npu_hbm_util）。	百分比（Percent）	0～100%

分类	名称	指标	指标含义	单位	取值范围
	NPU显存使用量	ma_node_npu_memory_used_megabytes	该指标用于统计测量对象已使用的NPU显存。（即将废止，Snt3系列替代指标为ma_node_npu_ddr_memory_usage_bytes，Snt9系列替代指标为ma_node_npu_hbm_usage_bytes）。	≥0	兆字节（Megabytes）
	NPU显存容量	ma_node_npu_memory_total_megabytes	该指标用于统计测量对象的NPU显存容量。（即将废止， Snt3系列替代指标为ma_node_npu_ddr_memory_bytes，Snt9系列替代指标为ma_node_npu_hbm_bytes）。	>0	兆字节（Megabytes）
	AI处理器错误码	ma_node_npu_ai_core_error_code	昇腾系列AI处理器错误码	-	-
	AI处理器健康状态	ma_node_npu_ai_core_health_status	昇腾系列AI处理器健康状态	-	• 1：健康 • 0：不健康

分类	名称	指标	指标含义	单位	取值范围
	AI处理器功耗	ma_node_npu_ai_core_power_usage_watts	昇腾系列AI处理器功耗	瓦特（W）	>0
	AI处理器温度	ma_node_npu_ai_core_temperature_celsius	昇腾系列AI处理器温度	摄氏度（℃）	自然数
	AI处理器AICORE利用率	ma_node_npu_ai_core_util	昇腾系列AI处理器AICore利用率	百分比（Percent）	0～100%
	AI处理器AICORE时钟频率	ma_node_npu_ai_core_frequency_hertz	昇腾系列AI处理器AICore时钟频率	赫兹（Hz）	>0
	AI处理器电压	ma_node_npu_ai_core_voltage_volts	昇腾系列AI处理器电压	伏特（V）	自然数
	AI处理器DDR内存总量	ma_node_npu_ddr_memory_bytes	昇腾系列AI处理器DDR内存总量	字节（Byte）	>0
	AI处理器DDR内存使用量	ma_node_npu_ddr_memory_usage_bytes	昇腾系列AI处理器DDR内存使用量	字节（Byte）	>0
	AI处理器DDR内存利用率	ma_node_npu_ddr_memory_util	昇腾系列AI处理器DDR内存利用率	百分比（Percent）	0～100%
	AI处理器HBM内存总量	ma_node_npu_hbm_bytes	昇腾系列AI处理器HBM总内存（Snt9 AI处理器专属）	字节（Byte）	>0

分类	名称	指标	指标含义	单位	取值范围
	AI处理器HBM内存使用量	ma_node_npu_hbm_usage_bytes	昇腾系列AI处理器HBM内存使用量（Snt9 AI处理器专属）	字节（Byte）	>0
	AI处理器HBM内存利用率	ma_node_npu_hbm_util	昇腾系列AI处理器HBM内存利用率（Snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存带宽利用率	ma_node_npu_hbm_bandwidth_util	昇腾系列AI处理器HBM内存带宽利用率（Snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存时钟频率	ma_node_npu_hbm_frequency_hertz	昇腾系列AI处理器HBM内存时钟频率（Snt9 AI处理器专属）	赫兹（Hz）	>0
	AI处理器HBM内存温度	ma_node_npu_hbm_temperature_celsius	昇腾系列AI处理器HBM内存温度（Snt9 AI处理器专属）	摄氏度（℃）	自然数
	AI处理器AICPU利用率	ma_node_npu_ai_cpu_util	昇腾系列AI处理器AICPU利用率	百分比（Percent）	0～100%
	AI处理器控制CPU利用率	ma_node_npu_ctrl_cpu_util	昇腾系列AI处理器控制CPU利用率	百分比（Percent）	0～100%

分类	名称	指标	指标含义	单位	取值范围
NPU RoCE网络	NPU RoCE网络上行速率	ma_node_npu_roce_tx_rate_bytes_per_second	NPU RoCE网络上行速率	字节/秒（Bytes/Second）	≥0
	NPU RoCE网络下行速率	ma_node_npu_roce_rx_rate_bytes_per_second	NPU RoCE网络下行速率	字节/秒（Bytes/Second）	≥0
	MAC上行pause帧总数	ma_node_npu_roce_mac_tx_pause_packets_total	NPU RoCE网络MAC发送的pause帧总报文数	个	≥0
	MAC下行pause帧总数	ma_node_npu_roce_mac_rx_pause_packets_total	NPU RoCE网络MAC接收的pause帧总报文数	个	≥0
	MAC上行pfc帧总数	ma_node_npu_roce_mac_tx_pfc_packets_total	NPU RoCE网络MAC发送的PFC帧总报文数	个	≥0
	MAC下行pfc帧总数	ma_node_npu_roce_mac_rx_pfc_packets_total	NPU RoCE网络MAC接收的PFC帧总报文数	个	≥0
	MAC上行坏包总数	ma_node_npu_roce_mac_tx_bad_packets_total	NPU RoCE网络MAC发送的坏包总报文数	个	≥0
	MAC下行坏包总数	ma_node_npu_roce_mac_rx_bad_packets_total	NPU RoCE网络MAC接收的坏包总报文数	个	≥0
	RoCE上行坏包总数	ma_node_npu_roce_tx_err_packets_total	NPU ROCE发送的坏包总报文数	个	≥0

分类	名称	指标	指标含义	单位	取值范围
	RoCE下行坏包总数	ma_node_npu_roce_rx_err_packets_total	NPU ROCE接收的坏包总报文数	个	≥0
infiniband或RoCE网络	网卡接收数据总量	ma_node_infiniband_port_received_data_bytes_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), received on all VLs from the port.	(counting in double words, 32 bits	≥0
	网卡发送数据总量	ma_node_infiniband_port_transmitted_data_bytes_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), transmitted on all VLs from the port.	(counting in double words, 32 bits	≥0

分类	名称	指标	指标含义	单位	取值范围
NFS挂载状态	NFS检索文件属性操作拥塞时间	ma_node_mountstats_getattr_backlog_wait	Getattr is an NFS operation that retrieves the attributes of a file or directory, such as size, permissions, owner, etc. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS检索文件属性操作往返时间	ma_node_mountstats_getattr_rtt	Getattr is an NFS operation that retrieves the attributes of a file or directory, such as size, permissions, owner, etc. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS检查文件权限操作拥塞时间	ma_node_mountstats_access_backlog_wait	Access is an NFS operation that checks the access permissions of a file or directory for a given user. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS检查文件权限操作往返时间	ma_node_mountstats_access_rtt	Access is an NFS operation that checks the access permissions of a file or directory for a given user. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS解析文件句柄操作拥塞时间	ma_node_mountstats_lookup_backlog_wait	Lookup is an NFS operation that resolves a file name in a directory to a file handle. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS解析文件句柄操作往返时间	ma_node_mountstats_lookup_rtt	Lookup is an NFS operation that resolves a file name in a directory to a file handle. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS读文件操作拥塞时间	ma_node_mountstats_read_backlog_wait	Read is an NFS operation that reads data from a file. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS读文件操作往返时间	ma_node_mountstats_read_rtt	Read is an NFS operation that reads data from a file. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS写文件操作拥塞时间	ma_node_mountstats_write_backlog_wait	Write is an NFS operation that writes data to a file. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS写文件操作往返时间	ma_node_mountstats_write_rtt	Write is an NFS operation that writes data to a file. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

表 9-14 Diagnos（IB，仅专属池上会收集）

分类	名称	指标	指标含义	单位	取值范围
infiniband或RoCE网络	PortXmitData	infiniband_port_xmit_data_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), transmitted on all VLs from the port.	计数值	自然数
	PortRcvData	infiniband_port_rcv_data_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), received on all VLs from the port.	计数值	自然数
	SymbolErrorCounter	infiniband_symbol_error_counter_total	Total number of minor link errors detected on one or more physical lanes.	计数值	自然数
	LinkErrorRecoveryCounter	infiniband_link_error_recovery_counter_total	Total number of times the Port Training state machine has successfully completed the link error recovery process.	计数值	自然数

分类	名称	指标	指标含义	单位	取值范围
	PortRcvErrors	infiniband_port_rcv_errors_total	Total number of packets containing errors that were received on the port including: Local physical errors (ICRC, VCRC, LPCRC, and all physical errors that cause entry into the BAD PACKET or BAD PACKET DISCARD states of the packet receiver state machine) Malformed data packet errors (LVer, length, VL) Malformed link packet errors (operand, length, VL) Packets discarded due to buffer overrun (overflow)	计数值	自然数
	LocalLinkIntegrityErrors	infiniband_local_link_integrity_errors_total	This counter indicates the number of retries initiated by a link transfer layer receiver.	计数值	自然数
	PortRcvRemotePhysicalErrors	infiniband_port_rcv_remote_physical_errors_total	Total number of packets marked with the EBP delimiter received on the port.	计数值	自然数
	PortRcvSwitchRelayErrors	infiniband_port_rcv_switch_relay_errors_total	Total number of packets received on the port that were discarded when they could not be forwarded by the switch relay for the following reasons: DLID mapping VL mapping Looping (output port = input port)	计数值	自然数

分类	名称	指标	指标含义	单位	取值范围
	PortXmitWait	infiniband_port_transmit_wait_total	The number of ticks during which the port had data to transmit but no data was sent during the entire tick (either because of insufficient credits or because of lack of arbitration).	计数值	自然数
	PortXmitDiscards	infiniband_port_xmit_discards_total	Total number of outbound packets discarded by the port because the port is down or congested.	计数值	自然数

表 9-15 Label 名字栏

指标对象	Label名字	Label描述
容器级别指标	modelarts_service	容器属于哪个服务，包含notebook，train和infer。
	instance_name	容器所属pod的名字。
	service_id	页面展示的实例或者job id。如开发环境为：cf55829e-9bd3-48fa-8071-7ae870dae93a， 训练作业为：9f322d5a-b1d2-4370-94df-5a87de27d36e
	node_ip	容器所属的节点IP值。
	container_id	容器ID。
	cid	集群ID。
	container_name	容器名称。
	project_id	用户所属的账号的project id。
	user_id	提交作业的用户所属的账号的user id。
	npu_id	昇腾卡的ID信息，比如davinci0（即将废止）。
	device_id	昇腾系列AI处理器的Physical ID。
	device_type	昇腾系列AI处理器类型。

指标对象	Label名字	Label描述
	pool_id	物理专属池对应的资源池id。
	pool_name	物理专属池对应的资源池name。
	logical_pool_id	逻辑子池的id。
	logical_pool_name	逻辑子池的name。
	gpu_uuid	容器使用的GPU的UUID。
	gpu_index	容器使用的GPU的索引。
	gpu_type	容器使用的GPU的型号。
	account_name	训练、推理或开发环境任务创建者的账号名。
	user_name	训练、推理或开发环境任务创建者的用户名。
	task_creation_time	训练、推理或开发环境任务的创建时间。
	task_name	训练、推理或开发环境任务的名称。
	task_spec_code	训练、推理或开发环境任务的规格。
	cluster_name	CCE集群名称。
node级别指标	cid	该node所属CCE集群的ID。
	node_ip	节点的IP。
	host_name	节点的主机名。
	pool_id	物理专属池对应的资源池ID。
	project_id	物理专属池的用户的project id。
	npu_id	昇腾卡的ID信息，比如davinci0（即将废止）。
	device_id	昇腾系列AI处理器的Physical ID。
	device_type	昇腾系列AI处理器类型。
	gpu_uuid	节点上GPU的UUID。
	gpu_index	节点上GPU的索引。
	gpu_type	节点上GPU的型号。
	device_name	infiniband或RoCE网络网卡的设备名称。
	port	IB网卡的端口号。

指标对象	Label名字	Label描述
	physical_state	IB网卡每个端口的状态。
	firmware_version	IB网卡的固件版本。
	filesystem	NFS挂载的文件系统。
	mount_point	NFS的挂载点。
Diagnos	cid	GPU所在节点所属的CCE集群ID。
	node_ip	GPU所在节点的IP。
	pool_id	物理专属池对应的资源池ID。
	project_id	物理专属池的用户的project id。
	gpu_uuid	GPU的UUID。
	gpu_index	节点上GPU的索引。
	gpu_type	节点上GPU的型号。
	device_name	网络设备或磁盘设备的名称。
	port	IB网卡的端口号。
	physical_state	IB网卡每个端口的状态。
	firmware_version	IB网卡的固件版本。

10 AI Hub

10.1 AI Hub 简介

AI Hub是在ModelArts的基础上构建的开发者生态社区，提供了数据集、算法、模型、Workflow等AI数字资产的共享，为高校科研机构、AI应用开发商、解决方案集成商、企业级/个人开发者等群体，提供安全、开放的共享及交易环节，加速AI资产的开发与落地，保障AI开发生态链上各参与方高效地实现各自的商业价值。

AI Hub中，支持数据集、算法、模型、Workflow等AI资产的共享。

- “数据”：共享了数据集。

  AI Hub的数据模块支持数据集的共享和下载。在AI Hub的“数据”中，可以查找并下载满足业务需要的数据集。也可以将自己本地的数据集发布至AI Hub中，共享给其他用户使用。

- 算法：共享了算法。

  AI Hub的算法模块支持算法的共享和订阅。在AI Hub的“算法”中，可以查找您想要的算法，订阅满足业务需要的资产，最后推送至ModelArts控制台使用。也可以将个人开发的算法分享发布至AI Hub中，共享给其他用户使用。

- “资产集市 > 模型”：共享了ModelArts模型。

  AI Hub的模型模块支持发布和订阅共享的模型。在AI Hub的“模型”中，可以查找您想要的ModelArts模型，订阅满足业务需要的资产，最后推送至ModelArts管理控制台使用。也可以将个人开发的ModelArts模型分享发布至AI Hub中，共享给其他用户使用。

- “Workflow”：共享了Workflow。

  AI Hub的Workflow模块支持Workflow的共享和订阅。在AI Hub的Workflow中，可以查找您想要的Workflow，订阅满足业务需要的资产，最后推送至ModelArts控制台使用。也可以将个人开发的Workflow分享发布至AI Hub中，共享给其他用户使用。

AI Hub 使用限制

- 订阅主要是获取AI资产的使用配额和使用权，支持在配额定义的约束下，有限地使用AI资产。

- 已发布的AI资产，如果不需要在资产列表中展示该资产，可以将资产下架。下架后，已发布资产仅发布者可见。已经被订阅的资产，即便资产下架后，基于配额资源的约束，仍然可有效使用该资产，不会因为该资产的下架而产生使用问题。

10.2 入驻 AI Hub

若需要在AI Hub共享AI资产，则需要先完成入驻AI Hub。

1. 如果没有入驻过AI Hub，在“算法”、“模型”等页面上单击“发布”按钮，将跳转至“欢迎入驻AI Hub”页面。
2. 在“欢迎入驻AI Hub”页面，填写“昵称”，单击“确定”完成入驻。
3. 注册完成后，您可以在AI Hub中发布数据集、模型等AI资产。

10.3 管理中心介绍

“管理中心”可以查看各类AI资产的发布订阅情况和个人资料等。

表 10-1 管理中心列表介绍

模块列表	功能介绍
我的算法	展示个人发布和订阅的算法列表。 ● “我的发布”：可以查看个人发布的算法信息，如浏览量、收藏量、订阅量等。通过右侧的“上架”或“下架”可以管理已发布的算法。资产下架后，已订阅该资产的用户可继续正常使用，其他用户将无法查看和订阅该资产。下架后的资产可以重新上架。 ● “我的订阅”：可以查看个人订阅的算法信息，如发布者、应用控制台、剩余配额等。通过右侧的“取消订阅”或“找回订阅”可以管理已订阅的算法。取消订阅后，ModelArts管理控制台算法管理模块-我的订阅列表中将不再展示该算法。已取消订阅的算法可以找回订阅，并在原配额约束下可以继续使用该算法。
我的模型	展示个人发布和订阅的模型列表，包括ModelArts模型和HiLens技能。 ● “我的发布”：可以查看个人发布的模型信息，如浏览量、收藏量、订阅量等。通过右侧的“上架”或“下架”可以管理已发布的模型。资产下架后，已订阅该资产的用户可继续正常使用，其他用户将无法查看和订阅该资产。下架后的资产可以重新上架。 ● “我的订阅”：可以查看个人订阅的模型信息，如发布者、应用控制台、剩余配额等。通过右侧的“取消订阅”或“找回订阅”可以管理已订阅的ModelArts模型。取消订阅后，在ModelArts管理控制台“模型管理 > 模型 > 我的订阅”列表中，将不再展示该模型。已取消订阅的模型可以找回订阅，并在原配额约束下可以继续使用该模型。

模块列表	功能介绍
我的数据	展示个人发布和下载的数据集列表。 • “我的发布”：可以查看个人发布的数据集信息，如文件大小、文件数量等。通过右侧的“重试”或“删除”可以管理已发布的数据集。 • “我的下载”：可以查看个人下载的数据集信息。单击下拉三角，可以查看数据集ID、下载方式、目标区域等信息。
我的Workflow	展示个人发布和订阅的Workflow列表。 • **“我的发布”**：可以查看个人发布的Workflow信息，浏览量、收藏量、订阅量等。通过右侧的“上架”，“下架”或“删除”可以管理已发布的Workflow。 • **“我的订阅”**：可以查看个人订阅的Workflow信息。通过右侧“取消订阅”或“找回订阅”可以管理已订阅的资产。
我的资料	查看个人基本信息，包括“账号”、“头像”、“昵称”、“邮箱”、“简介”等信息。 • 单击“编辑资料”，可以编辑“昵称”和“简介”。 • 单击“更换头像”，可以自定义替换头像。

10.4 订阅使用

10.4.1 查找和收藏资产

AI Hub共享了算法、数据集、模型、Workflow等多种AI资产，为了方便快速搜索相关资产，提供了多种快速搜索方式以及收藏功能，提升资产的查找效率。

搜索资产

在各类资产模块页面，通过如下几种搜索方式可以提高资产的查找效率，快速找到适合的算法、模型、数据集、Wrokflow等资产。

图 10-1 搜索资产

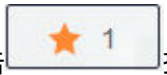
表 10-2 快速搜索方式

区域	类型	搜索方式	支持的AI资产
1	搜索官方资产	在页面单击“官方”，筛选出所有的官方资产，该类资产均可**免费**使用。	算法
2	搜索精选商品	在页面单击“精选”，筛选出所有被标记为精选的资产。	数据、算法、Workflow
3	按标签搜索	在页面单击“所有标签”，选择标签，单击“确定”，筛选出相关资产。	数据、算法、Workflow
4	按排序方式搜索	在页面的排序列表选择排序方式，调整资产排序方式快速查找所需资产。	数据、算法、Workflow

收藏免费资产

当搜索到感兴趣的免费资产时，可以收藏该资产，方便后续在“我的收藏”快速查找。

1. 单击目标资产，进入资产详情页面。
2. 在详情页面右上角，单击 

按钮收藏资产。
   收藏成功后，在各个模块的“我的收藏”页签可以快速查看收藏的资产。
3. （可选）若需要取消收藏，再次单击 

按钮即可。

10.4.2 订阅算法

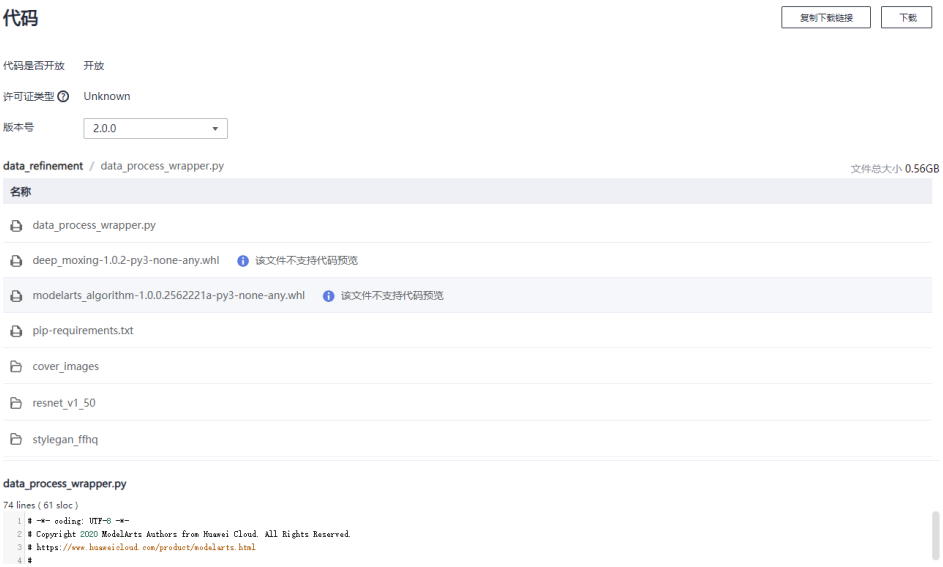
在AI Hub中，您可以查找并订阅免费的满足业务需要的算法，直接用于创建训练作业。

订阅算法

1. 登录“AI Hub”。
2. 选择算法，进入算法页面，该页面展示了所有共享的算法。
3. 搜索业务所需的算法，请参见**查找资产**。
4. 单击目标算法进入详情页面。
   - 在详情页面您可以查看算法的“描述”、“限制”、“版本”和“交付”等信息。
   - 为方便您的使用，在订阅算法时，建议您查看算法详情页“版本”页签中关于算法对应版本的“使用约束”，准备对应的数据和资源规格后进行使用。
   - 对于开放代码的算法，您也可以在详情页面预览或者下载对应代码。
     在“代码”页签，单击右侧的“下载”将完整代码下载到本地，您也可以单击下方列表中的文件名称进行预览。

目前如下后缀结尾的文件类型支持代码预览：txt、py、h、xml、html、c、properties、yml、cmake、sh、css、js、cpp、json、md、sql、bat、conf

图 10-2 下载预览代码



5. 在详情页面单击“订阅”，根据算法是否具有使用约束进行不同操作：
   - 若订阅是具有使用约束的算法，则弹出“使用约束”页面，查看并确认后单击“继续订阅”即可成功订阅。
   - 若订阅是没有使用约束的算法，则直接成功订阅。

   算法被订阅后，详情页的“订阅”按钮显示为“已订阅”，“管理中心 > 我的算法”页签里“我的订阅”。

使用算法

1. 订阅成功的算法可在ModelArts管理控制台使用，如创建训练作业等。

   方式一：从算法详情页进入管理控制台

   a. 在算法详情页单击“前往控制台”。
   b. 在弹出的“选择云服务区域”页面选择ModelArts所在的云服务区域，单击“确定”跳转至ModelArts控制台的“算法管理 > 我的订阅”页面。

图 10-3 进入算法管理

方式二：从“AI Hub”进入管理控制台

a. 在AI Hub，单击右上角“管理中心 > 我的算法”，进入“我的算法”页面。
b. 选择“我的订阅”页签，进入个人订阅的算法列表。
c. 在算法列表选择需要使用的算法，单击“应用控制台”列的“ModelArts”。
d. 在弹出的“选择云服务区域”页面选择ModelArts所在的云服务区域，单击“确定”跳转至ModelArts控制台的“算法管理 > 我的订阅”页面。

图 10-4 进入算法管理

2. 在“算法管理 > 我的订阅”页面，选择并展开订阅的目标算法。在版本列表中，单击“创建训练作业”跳转至创建训练作业页面。

取消或找回订阅的算法

当不需要使用AI Hub中订阅的算法时，可以取消订阅该算法。取消订阅后，ModelArts管理控制台“算法管理 > 我的订阅”列表中将不再展示该算法；当需要再次使用该算法时，可以找回订阅，ModelArts管理控制台“算法管理 > 我的订阅”列表中也会再次展示该算法。

1. 在AI Hub，单击右上角“管理中心 > 我的算法”，进入“我的算法”页面。
2. 选择“我的订阅”页签，进入个人订阅的算法列表。
   - **取消订阅**：仅已订阅的资产支持取消。
     单击目标资产右侧的“取消订阅”，在弹框中确认资产信息，单击“确定”取消订阅。
   - **找回订阅**：仅订阅后被取消订阅的资产支持找回。
     单击标资产右侧的“找回订阅”完成找回。

图 10-5 取消或找回订阅

10.4.3 订阅模型

在AI Hub中，您可以查找并订阅ModelArts模型。订阅成功的模型可以直接用于ModelArts模型部署。

订阅模型

1. 登录“AI Hub”。
2. 进入模型页面，该页面展示了所有共享的模型。
3. 搜索业务所需的模型，请参见**查找和收藏资产**。
4. 单击目标模型进入详情页面。

   在详情页面您可以查看模型的“描述”、“交付”、“限制”和“版本”等信息。
5. 在详情页面单击“订阅”。

   模型被订阅后，详情页的“订阅”按钮显示为“已订阅”，订阅成功的资产也会展示在“管理中心 > 我的模型”页签下“我的订阅”。

使用模型

订阅成功的模型可在ModelArts管理控制台使用，支持模型部署或安装等。

1. 将订阅成功的模型推送至应用控制台。

   方式一：从模型详情页进入管理控制台

   在模型详情页单击“前往控制台”，跳转至ModelArts控制台的“AI应用管理 > AI应用 > 我的订阅”页面。

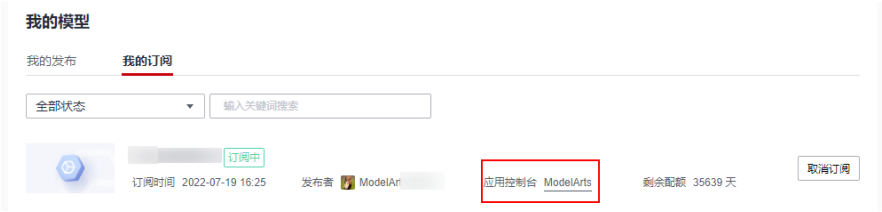
   模型对应版本列表的状态显示为“就绪”表示可以使用。

   图 10-6 推送模型

   方式二：从“AI Hub”进入管理控制台

   a. 在AI Hub，单击右上角“管理中心 > 我的模型”，进入“我的模型”页面。
   b. 选择“我的订阅”页签，进入个人订阅的模型列表。
   c. 在模型列表选择需要推送的模型，单击“应用控制台”列的ModelArts，跳转至ModelArts控制台的“AI应用管理 > AI应用 > 我的订阅”页面。

图 10-7 选择应用控制台



模型对应版本列表的状态显示为“就绪”表示可以使用。

2. 在应用控制台使用订阅的模型。

   在“AI应用管理 > AI应用 > 我的订阅”页面，选择并展开订阅的目标模型。在版本列表单击“部署”，可以将订阅的ModelArts模型部署为“在线服务”或“批量服务”，详细操作步骤请参见**服务部署简介**。

取消或找回订阅的模型

当不需要使用AI Hub中订阅的模型时，可以取消订阅该模型。取消订阅后，ModelArts管理控制台“AI应用管理 > AI应用 > 我的订阅”列表中将不再展示该模型；当需要再次使用该模型时，可以找回订阅，ModelArts管理控制台“AI应用管理 > AI应用 > 我的订阅”列表中也会再次展示该模型。

1. 在AI Hub，单击右上角“管理中心 > 我的模型”，进入“我的模型”页面。
2. 选择“我的订阅”页签，进入个人订阅的模型列表。
   - **取消订阅**：仅已订阅的资产支持取消。

     单击目标资产右侧的“取消订阅”，在弹框中确认资产信息，单击“确定”取消订阅。
   - **找回订阅**：仅订阅后被取消订阅的资产支持找回。

     单击标资产右侧的“找回订阅”完成找回。

10.4.4 下载数据

在AI Hub中，您可以查找并下载满足业务需要的数据集。

下载数据集

1. 登录“AI Hub”。
2. 选择“数据”，进入数据页面，该页面展示了所有共享的数据集。
3. 搜索业务所需的数据集，请参见**查找和收藏资产**。
4. 单击目标数据集进入详情页面。

   在详情页面查看数据集的“描述”、“版本”和“限制”等信息。
5. 在详情页面单击“下载”。根据数据集下载至OBS还是ModelArts数据集列表，填写不同配置信息：
   - **将数据集下载至OBS**
     - “下载方式”选择“对象存储服务（OBS）”。
     - “目标区域”选择您需要将该数据集下载到的区域位置。
     - “目标位置”选择OBS桶路径，桶内如有同名的文件或文件夹，将被新下载的文件或文件夹覆盖。

图 10-8 下载数据集（至 OBS）

- **将数据集下载至ModelArts**
  - “下载方式”选择“ModelArts数据集”。
  - “目标区域”选择您需要将该数据集下载到的区域位置。
  - “数据类型”：选择需要处理的文件类型。
  - “数据集输出位置”：数据集输出位置的OBS路径，此位置会存放输出的标注信息等文件，此位置不能和OBS数据源中的文件路径相同或为其子目录。
  - “数据集输入位置”：AI Hub的数据集下载到OBS的路径，此位置会作为数据集的数据存储路径，数据集输入位置不能和输出位置相同。
  - “名称”默认生成“data-xxxx”形式的数据集名称，该数据集会同步在ModelArts数据集列表中。
  - “描述”可以添加对于该数据集的相关描述。

图 10-9 下载数据集（至 ModelArts）

6. 单击“确定”，跳转至“数据 > 我的数据 > 我的下载”页面。下载成功后，数据集列表会显示“文件大小”。

使用数据集

1. 在AI Hub，单击右上角 “管理中心 > 我的数据”，进入“我的数据”页面。
2. 选择“我的下载”页签，查看所有下载的数据集。
3. 展开目标数据集，可以查看数据集详情。
   - 若是下载到OBS的数据集，可以获取“目标位置”，并参见**从OBS目录导入**数据集之后在ModelArts进行使用。

图 10-10 获取数据集目标位置

   - 若是下载到ModelArts，可以直接单击目标数据集跳转至ModelArts控制台的数据集详情页面，进行数据集相关的操作。

图 10-11 获取目标数据集

10.4.5 订阅 Workflow

在AI Hub中，您可以查找并订阅Workflow。订阅成功的Workflow通过AI Hub导入后可以直接在ModelArts控制台使用。

订阅 Workflow

1. 登录“AI Hub”。
2. 进入Workflow页面，该页面展示了所有共享的Workflow。
3. 搜索业务所需的Workflow，请参见**查找和收藏资产**。
4. 单击目标Workflow进入详情页面。

   在详情页面您可以查看Workflow的“描述”、“交付”、“版本”、“限制”等信息。
5. 在详情页面单击“订阅”。

   Workflow被订阅后，详情页的“订阅”按钮显示为“已订阅”，订阅成功的资产也会展示在“管理中心 > 我的Workflow”页签下“我的订阅”。

使用 Workflow

订阅成功的Workflow可在ModelArts管理控制台使用，支持导入工作流。

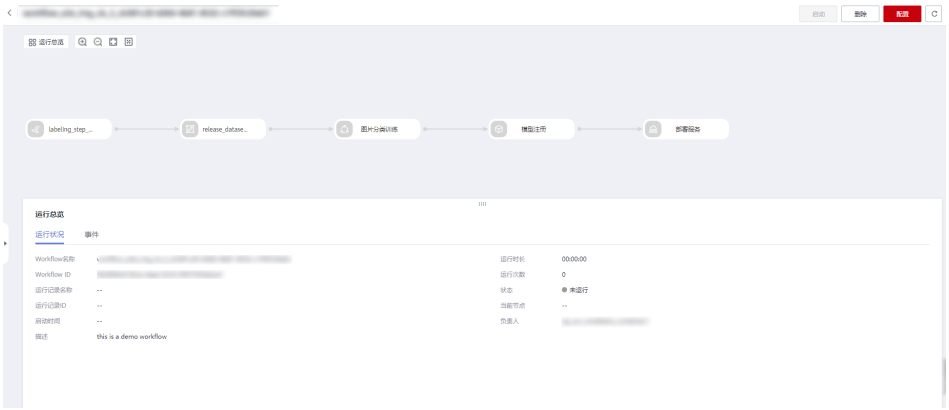
1. 将订阅成功的Workflow导入至ModelArts控制台。

   方式一：从Workflow详情页进入ModelArts控制台

   在Workflow详情页单击“运行”，在弹出来的对话框中选择需要导出的资产版本、云服务区域、工作空间信息，单击“导入”跳转至ModelArts控制台该Workflow的详情页。

图 10-12 导入 Workflow

图 10-13 ModelArts 控制台 Workflow 详情页



方式二：从“AI Hub”进入ModelArts控制台

a. 在AI Hub，单击右上角“管理中心 > 我的Workflow”，进入“我的Workflow”页面。
b. 单击“我的订阅”，进入个人订阅的Workflow列表。
c. 在“我的订阅”列表，选择需要导入的Workflow，单击“应用控制台”旁的“Workflow”。

图 10-14 选择应用控制台

d. 在弹出来的对话框中选择需要导出的“资产版本”、“云服务区域”、“工作空间”信息，单击“导入”跳转至ModelArts控制台该Workflow的详情页。

2. 在ModelArts控制台使用从Hub导入的Workflow。

   在ModelArts控制台左侧导航栏，单击Workflow。在Workflow列表中，找到从Hub导入的Workflow，单击配置进入到该Workflow。

取消或找回已订阅的 Workflow

当不需要使用AI Hub中订阅的Workflow时，可以取消订阅该Workflow。当需要再次使用该Workflow时，可以通过“找回订阅”恢复已取消的订阅。

10.5 发布分享

10.5.1 发布算法

在AI Hub中，您可以将个人开发的算法分享给他人使用。

前提条件

- 已入驻AI Hub。
- 在ModelArts的算法管理中已准备好待发布的算法。创建算法的相关操作请参见**创建算法**。

  说明

  创建算法时，算法代码存储的OBS桶内文件和文件夹不能重名，否则算法可能会发布失败。如果算法发布成功，则**代码开放**会失败。

发布算法

1. 进入“AI Hub”首页，进入算法页面。
2. 单击“发布”跳转到ModelArts控制台的“算法管理 > 我的算法”页面。
3. 单击待发布的算法名称，进入算法详情页。
4. 在算法详情页的右上角单击“发布”，进入发布资产版本页面。
5. 在发布资产版本页面，填写相关信息，发布资产。
   - 如果是发布新资产。
     i. 在“资产名称”右侧单击“创建新资产”，在弹窗中填写“名称”和“资产描述”，单击“确定”。

     说明

     发布的新资产默认是私有资产，若需要白名单或公开请跳转到AI Hub进行设置。

     ii. 填写“资产版本”和“版本描述”。
     iii. 单击“发布”。
   - 如果是更新已发布资产的版本。
     i. 在“资产名称”下拉框中选择已有资产名称。
     ii. 在“资产版本”填写新的版本号，右侧单击“查看该资产版本”可以查看历史版本信息。
     iii. 填写“版本描述”。
     iv. 单击“发布”。
6. 提交资产发布申请后，AI Hub会自动托管上架，可以前往AI Hub查看或编辑资产详情。

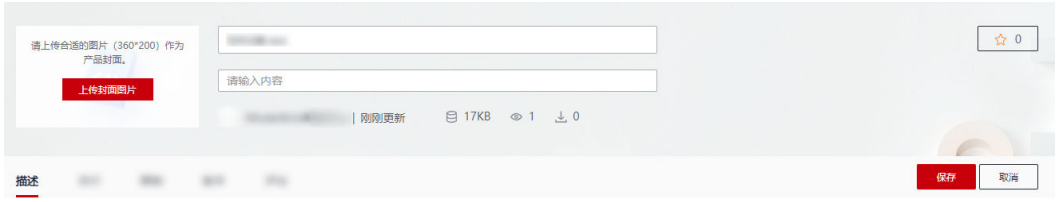
编辑资产详情

资产发布成功后，发布者可以进入详情页修改该资产的标题、封面图、描述等，让资产更吸引人。

修改封面图和二级标题

1. 在发布的资产详情页面，单击右侧的“编辑”，选择上传新的封面图，为资产编辑独特的主副标题。
2. 编辑完成之后单击“保存”。封面图和二级标题内容自动同步，您可以直接在资产详情页查看修改结果。

图 10-15 修改封面图和二级标题



编辑标签

1. 单击标签右侧的出现标签编辑框，在下拉框中勾选该资产对应的标签。
2. 单击编辑框右侧的对勾完成编辑。
   保存成功的标签信息会在资产搜索页成为过滤分类条件。

图 10-16 添加标签

编辑描述

1. 单击右侧的“编辑”，在编辑框中输入资产的描述内容，包含但不局限于背景、简介、使用方法、约束条件等。支持发布者以Markdown形式自由编辑。
2. 编辑完成之后单击“保存”。

编辑限制

支持修改资产的公开权限和时长限制。

1. 选择“限制”页签，单击右上方的“编辑”进入编辑模式：
   - 在“谁可以看”右侧的下拉框中选择公开权限。

- “公开”：表示所有使用AI Hub的用户都可以查看且使用该资产。
- “指定用户”：表示仅特定用户可以查看及使用该资产。
- “仅自己可见”：表示只有当前账号可以查看并使用该资产。

说明

公开权限只支持权限的扩大，权限从小到大为“仅自己可见<指定用户<公开”。所以若一开始创建的是公开算法，将不支持修改“谁可以看”。

- “时长限制”可以选择“不启用”或“启用”。当启用时，可以设置资产的使用时长，以及到期后是否续订。

2. 单击“保存”，完成修改。

图 10-17 编辑限制

编辑版本

1. 选择“版本”页签，单击右上方的“编辑”。
2. 在此页面可以修改版本说明或者单击对应版本“操作”列的“下线”，下架不需要的资产版本。下线操作仅对已上架成功且存在多个可用版本的资产有效。
3. 在版本框右侧单击“添加版本”跳转到ModelArts控制台的“算法管理 > 我的算法”页面，参考**发布算法**给目标算法新增版本。
4. 编辑完成后，单击右上方的“保存”完成修改。

图 10-18 编辑算法的版本

编辑论文

1. 选择“论文”页签，单击右上方的“编辑”，在编辑框内填写论文名称和其对应的访问链接，订阅者可以直接单击该链接查看论文详细内容。
2. 编辑完成后单击“保存”完成修改。

编辑代码

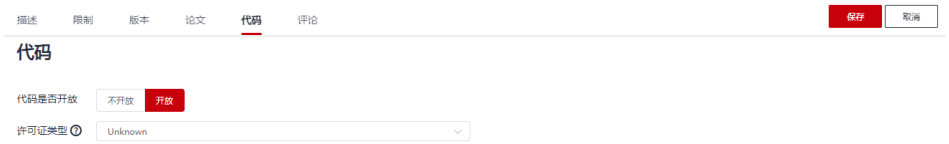
1. 选择“代码”页签，单击右上方的“编辑”，可以选择“代码是否开放”。

说明

订阅期满之前，下架代码不开放的算法不影响已订阅用户的使用。再次发布该算法代码开放后，主页列表不展示已经下架的算法，但用户可以在“管理中心> 我的算法 > 我的订阅”页面单击该算法名称查看预览代码。

2. 若开放代码可以选择修改“许可证类型”。
   单击许可证类型后面的感叹号可以了解许可证详情。
3. 编辑完成后单击“保存”完成修改。

图 10-19 编辑代码



下架算法

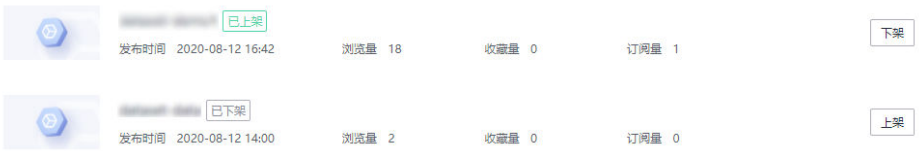
当您需要在AI Hub下架共享的资产时，可以执行如下操作：

1. 在“AI Hub”页面，选择“管理中心 > 我的算法”，进入“我的算法”页面。
2. 在“我的算法 > 我的发布”页面，单击目标资产右侧的“下架”，在弹框中确认资产信息，单击“确定”完成下架。

   说明

   资产下架后，已订阅该资产的用户在时长限制期内可继续正常使用，其他用户将无法查看和订阅该资产。

   图 10-20 下架资产

   资产下架成功后，操作列的“下架”会变成“上架”，您可以通过单击“上架”将下架的资产重新共享到AI Hub中。

10.5.2 发布模型

在AI Hub中，您可以个人开发的ModelArts模型分享给他人使用。

前提条件

- 若是发布ModelArts模型，已经在ModelArts的模型管理中准备好待发布的模型。创建模型的相关操作请参见**模型管理**。使用容器镜像导入的模型和其他训练产生的模型都支持发布至AI Hub。

发布模型

1. 进入“AI Hub”首页，进入模型页面。
2. 单击“发布”进入“发布资产到AI Hub”页面，填写相关信息。
   - 如果是发布新资产，如**图10-21**所示。
     i. “发布方式”选择“创建新资产”。

ii. 填写“资产标题”。即在AI Hub显示的资产名称。

iii. “来源”默认为“ModelArts”。

iv. 设置“ModelArts区域”。

设置可以使用该资产的ModelArts区域，以控制台实际可选值为准。

v. 选择“AI应用名称”。

从ModelArts的AI应用管理中选择待发布的模型。支持将使用容器镜像导入的模型和其他训练产生的模型发布至AI Hub。

vi. 填写“资产版本”。版本号格式为“x.x.x”。

vii. 设置“谁可以看”。

设置资产的公开权限。可选值有：

“公开”：表示所有使用AI Hub的用户都可以查看且使用该资产。

“指定用户”：表示仅特定用户可以查看及使用该资产。

“仅自己可见”：表示只有当前账号可以查看并使用该资产。

viii. 设置“时长限制”。

设置订阅者可以使用资产的时长，默认关闭，即无限期使用。若打开时长限制，除了设置资产使用的时长，还可以设置到期后是否续订。

ix. 单击“发布”。

- 如果是更新已发布资产的版本，如**图10-22**所示。

i. “发布方式”选择“添加资产版本”。

ii. 在“资产标题”下拉框中选择已有资产名称。支持搜索资产名称。

iii. 设置“ModelArts区域”。

设置可以使用该资产的ModelArts区域，以控制台实际可选值为准。

iv. 选择“AI应用名称”。

从ModelArts的AI应用管理中选择待发布的模型。支持将使用容器镜像导入的模型和其他训练产生的模型发布至AI Hub。

v. 在“资产版本”填写新的版本号。

vi. 单击“发布”。

图 10-21 发布新资产

图 10-22 添加资产版本

3. 发布成功后，自动跳转至资产详情页面。

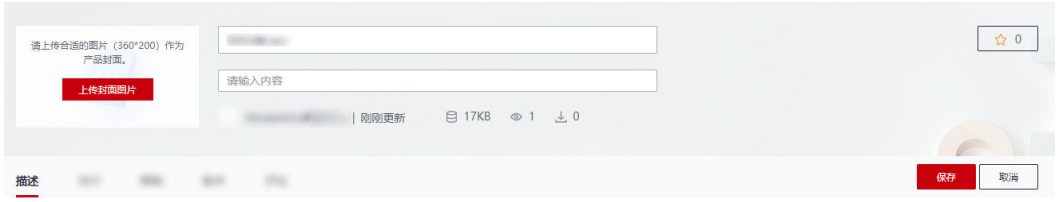
编辑资产详情

资产发布成功后，发布者可以进入详情页修改该资产的标题、封面图、描述等，让资产更吸引人。

修改封面图和二级标题

1. 在发布的资产详情页面，单击右侧的“编辑”，选择上传新的封面图，为资产编辑独特的主副标题。
2. 编辑完成之后单击“保存”。封面图和二级标题内容自动同步，您可以直接在资产详情页查看修改结果。

图 10-23 修改封面图和二级标题



编辑标签

1. 单击标签右侧的出现标签编辑框，在下拉框中勾选该资产对应的标签。
2. 单击编辑框右侧的对勾完成编辑。
   保存成功的标签信息会在资产搜索页成为过滤分类条件。

图 10-24 添加标签

编辑描述

1. 单击右侧的“编辑”，在编辑框中输入资产的描述内容，包含但不局限于背景、简介、使用方法、约束条件等。支持发布者以Markdown形式自由编辑。
2. 编辑完成之后单击“保存”。

编辑限制

支持修改资产的公开权限和时长限制。

1. 选择“限制”页签，单击右上方的“编辑”进入编辑模式：
   - 在“谁可以看”右侧的下拉框中选择公开权限。

- “公开”：表示所有使用AI Hub的用户都可以查看且使用该资产。
- “指定用户”：表示仅特定用户可以查看及使用该资产。
- “仅自己可见”：表示只有当前账号可以查看并使用该资产。

说明

公开权限只支持权限的扩大，权限从小到大为“仅自己可见<指定用户<公开”。所以若一开始创建的是公开算法或模型，将不支持修改“谁可以看”。

- “时长限制”可以选择“不启用”或“启用”。当启用时，可以设置资产的使用时长，以及到期后是否续订。

2. 单击“保存”，完成修改。

图 10-25 编辑限制

编辑版本

1. 选择“版本”页签，单击右上方的“编辑”。
2. 在此页面可以修改版本说明或者单击对应版本“操作”列的“下线”，下架不需要的资产版本。下线操作仅对已上架成功且存在多个可用版本的资产有效。
3. 在版本框右侧单击“添加版本”，弹出“发布资产到AI Hub”页面，参考**更新已发布资产的版本**添加资产版本。
4. 编辑完成后，单击右上方的“保存”完成修改。

图 10-26 编辑模型的版本

下架模型

当您需要在AI Hub下架共享的资产时，可以执行如下操作：

1. 在“AI Hub”页面，选择“管理中心 > 我的模型”，进入“我的模型”页面。
2. 在“我的模型 > 我的发布”页面，单击目标资产右侧的“下架”，在弹框中确认资产信息，单击“确定”完成下架。

说明

资产下架后，已订阅该资产的用户在时长限制期内可继续正常使用，其他用户将无法查看和订阅该资产。

图 10-27 下架资产

资产下架成功后，操作列的“下架”会变成“上架”，您可以通过单击“上架”将下架的资产重新共享到AI Hub中。

10.5.3 发布数据

在AI Hub中，您可以个人数据集分享给他人使用。

前提条件

- 已入驻AI Hub。
- 待发布的数据集已存放在ModelArts的数据集列表中或者存放在对象存储服务（OBS）中。创建或上传数据集的相关操作请参见**创建数据集**。

发布数据集

1. 进入“AI Hub”首页，进入数据页面。
2. 单击“发布”进入发布数据集页面，填写相关信息。
   - 若选择ModelArts已有的数据集发布，则参见**表10-3**配置数据集信息。

表 10-3 参数说明（ModelArts）

参数	说明
资产分类	选择“数据”。
资产标题	在AI Hub显示的资产名称，建议按照您的目的设置。
来源	选择“ModelArts”。 单个数据集最多支持20000个文件，总大小不超过30G。
ModelArts区域	选择数据集所在的区域。 以控制台实际可选值为准。
选择数据	从下拉列表中选择当前区域中需要发布的目标数据集。 当前只支持标注类型为图像分类、物体检测的图片数据集，以及自由格式的数据集。
选择版本	选择目标数据集需要发布的版本。
数据类型	至少选择一个数据集类型的标签。 可选标签：图片、音频、视频、文本、表格、其他
许可证类型	根据业务需求和数据集类型选择合适的许可证类型。 单击许可证类型后面的 可以查看许可证详情。

参数	说明
谁可以看	设置此数据集的公开权限。可选值有： ● “公开”：表示所有使用AI Hub的用户都可以查看且使用该资产。 ● “指定用户”：表示仅特定用户可以查看及使用该资产。 ● “仅自己可见”：表示只有当前账号可以查看并使用该资产。

- 若选择对象存储服务（OBS）中已有的数据集发布，则参见表10-4配置数据集信息。

  表 10-4 参数说明（OBS）

参数	说明
资产分类	默认为“数据”。
资产标题	在AI Hub显示的资产名称，建议按照您的目的设置。
来源	选择“对象存储服务（OBS）”。 单个数据集最多支持20000个文件，总大小不超过30G。
OBS区域	选择数据所在OBS桶的存储区域。
存储位置	选择待发布数据集所在对象存储服务（OBS）的路径。
数据类型	至少选择一个数据集类型的标签。 可选标签：图片、音频、视频、文本、表格、其他
许可证类型	根据业务需求和数据集类型选择合适的许可证类型。 单击许可证类型后面的  可以查看许可证详情。
谁可以看	设置此数据集的公开权限。可选值有： ● “公开”：表示所有使用AI Hub的用户都可以查看且使用该资产。 ● “指定用户”：表示仅特定用户可以查看及使用该资产。 ● “仅自己可见”：表示只有当前账号可以查看并使用该资产。

3. 单击“发布”，跳转至数据集详情页面。

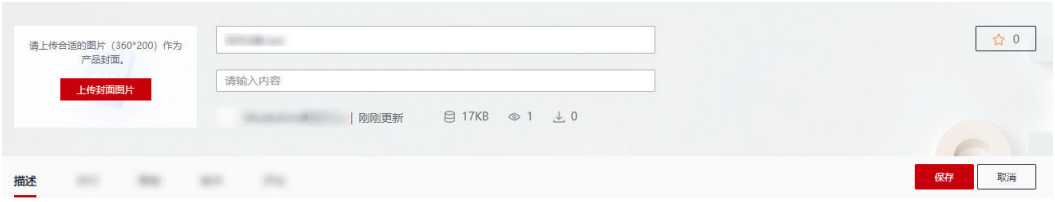
编辑资产详情

数据集发布成功后，发布者可以进入数据集的详情页修改该数据集“描述”、“版本”和“限制”等信息。

修改封面图和二级标题

1. 在发布的资产详情页面，单击右侧的“编辑”，选择上传新的封面图，为资产编辑独特的主副标题。
2. 编辑完成之后单击“保存”。封面图和二级标题内容自动同步，您可以直接在资产详情页查看修改结果。

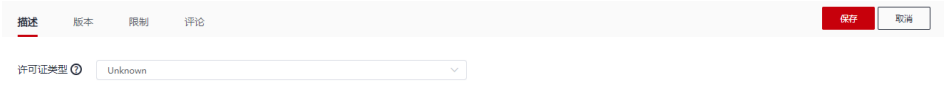
图 10-28 修改封面图和二级标题



编辑许可证类型

1. 在发布的资产详情页面，单击右侧的“编辑”。
2. 在许可证类型右侧的下拉框中选择需要更新的许可证，单击“保存”完成修改。单击许可证类型后面的感叹号可以了解许可证详情。

图 10-29 编辑许可证类型



编辑标签

1. 单击标签右侧的 出现标签编辑框。
2. 在下拉框中勾选该资产对应的标签，单击编辑框右侧的对勾完成编辑。标签信息会在资产主页成为过滤分类条件，让用户更容易找到您的资产。

图 10-30 添加标签

编辑描述

1. 单击右侧的“编辑”，在编辑框中输入资产的描述内容，包含但不局限于背景、简介、使用方法、约束条件等。支持发布者以Markdown形式自由编辑。
2. 编辑完成之后单击“保存”。

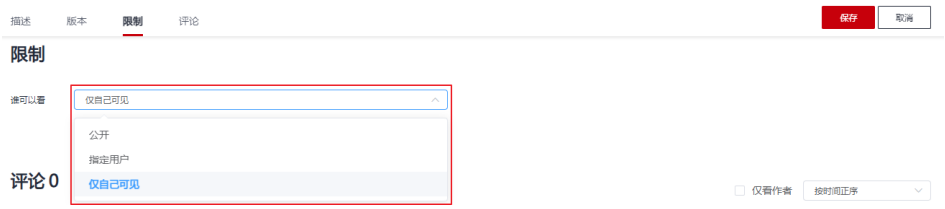
编辑版本

1. 选择“版本”页签，单击右上方的“编辑”进入编辑模式。
2. 单击“版本说明”列的，添加版本说明，单击 完成添加。
   编辑数据集的版本信息便于区分数据集信息。

编辑限制

1. 选择“限制”页签，单击右上方的“编辑”进入编辑模式。
2. 在“谁可以看”右侧的下拉框中选择公开权限。
   - “公开”：表示所有使用AI Hub的用户都可以查看且使用该资产。
   - “指定用户”：表示仅特定用户可以查看及使用该资产。
   - “仅自己可见”：表示只有当前账号可以查看并使用该资产。
3. 单击“保存”，完成修改。

图 10-31 编辑限制



重试发布数据集

若数据集发布异常，您可以重试发布。

1. 在AI Hub页面的右上角选择，“管理中心 > 我的数据”进入“我的数据”。
2. 在“我的发布”页签，查看发布异常的数据集。

图 10-32 查看发布异常的数据集

3. 根据异常状态的错误提示修改源数据后，单击目标数据集右侧的“重试”重新发布数据集。

删除发布的数据集

当您需要删除发布在AI Hub中的数据集时，可以执行如下步骤进行删除。

1. 在AI Hub页面的右上角选择“管理中心 > 我的数据”，进入“我的数据”。
2. 在“我的发布”页签，单击目标数据集右侧的“删除”，在弹窗中确认删除。

说明

由于数据集是下载至OBS使用的，所以删除已发布的数据集对使用者无影响。

11 使用自定义镜像

11.1 镜像管理

ModelArts 镜像管理简介

在AI业务开发以及运行的过程中，一般都会有复杂的环境依赖需要进行调测并固化。面对开发中的开发环境的脆弱和多轨切换问题，在ModelArts的AI开发最佳实践中，通过容器镜像的方式，将运行环境进行固化，以这种方式不仅能够进行依赖管理，而且可以方便的完成工作环境切换。配合ModelArts提供的云化容器资源使用，可以更加快速、高效的进行AI开发与模型实验的迭代等。

ModelArts默认提供了一组预置镜像供开发使用，这些镜像有以下特点：

- 零配置，即开即用，面向特定的场景，将AI开发过程中常用的依赖环境进行固化，提供合适的软件、操作系统、网络等配置策略，通过在硬件上的充分测试，确保其兼容性和性能最合适。
- 方便自定义，预置镜像已经在SWR仓库中，通过对预置镜像的扩展完成自定义镜像注册。
- 安全可信，基于安全加固最佳实践，访问策略、用户权限划分、开发软件漏洞扫描、操作系统安全加固等方式，确保镜像使用的安全性。

当用户对深度学习引擎、开发库有特殊需求场景的时候，预置镜像已经不能满足用户需求。ModelArts提供自定义镜像功能支持用户自定义运行引擎。

ModelArts底层采用容器技术，自定义镜像指的是用户自行制作容器镜像并在ModelArts上运行。自定义镜像功能支持自由文本形式的命令行参数和环境变量，灵活性比较高，便于支持任意计算引擎的作业启动需求。

ModelArts 的预置镜像使用场景

ModelArts给用户提供了一组预置镜像，用户可以直接使用预置镜像创建Notebook实例，在实例中进行依赖安装与配置后，保存为自定义镜像，可直接用于ModelArts训练，而不需要做适配。同时也可以使用预置镜像直接提交训练作业、创建AI应用等。

ModelArts提供的预置镜像版本是依据用户反馈和版本稳定性决定的。当用户的功能开发基于ModelArts提供的版本能够满足的时候，比如用户开发基于MindSpore1.X，建

议用户使用预置镜像，这些镜像经过充分的功能验证，并且已经预置了很多常用的安装包，用户无需花费过多的时间来配置环境即可使用。

ModelArts 的自定义镜像使用场景

- **Notebook中使用自定义镜像**

  当Notebook预置镜像不能满足需求时，用户可以制作自定义镜像。在镜像中自行安装与配置环境依赖软件及信息，并制作为自定义镜像，用于创建新的Notebook实例。

- **使用自定义镜像训练作业**

  如果您已经在本地完成模型开发或训练脚本的开发，且您使用的AI引擎是ModelArts不支持的框架。您可以制作自定义镜像，并上传至SWR服务。您可以在ModelArts使用此自定义镜像创建训练作业，使用ModelArts提供的资源训练模型。

- **使用自定义镜像创建AI应用**

  如果您使用了ModelArts不支持的AI引擎开发模型，也可通过制作自定义镜像，导入ModelArts创建为AI应用，并支持进行统一管理和部署为服务。

说明

用户制作的自定义镜像，使用的场景不同，镜像规则也不同，具体如下：

- 通用规则：SWR镜像类型为“私有”时，才可以共享给他人，适用于开发环境、训练作业、AI应用。
- 开发环境：SWR镜像类型为“公开”时，其它用户才可以在ModelArts镜像管理页面注册使用。
- 训练作业：SWR镜像类型为“公开”时，在使用自定义镜像创建训练作业时，在“镜像”输入框内直接填写“组织名称/镜像名称:版本名称”即可。例如：公开镜像的SWR地址为“xxx.com/test-image/tensorflow2_1_1:1.1.1”，则在创建训练作业的“镜像”输入框里内直接填“test-images/tensorflow2_1_1:1.1.1”。

自定义镜像功能关联服务介绍

使用自定义镜像功能可能涉及以下服务：

- 容器镜像服务

  容器镜像服务（Software Repository for Container，SWR）是一种支持镜像全生命周期管理的服务， 提供简单易用、安全可靠的镜像管理功能，帮助您快速部署容器化服务。您可以通过界面、社区CLI和原生API上传、下载和管理容器镜像。

  您制作的自定义镜像需要上传至SWR服务。ModelArts训练和创建AI应用使用的自定义镜像需要从SWR服务管理列表获取。

  图 11-1 获取镜像列表

- 对象存储服务

  对象存储服务（Object Storage Service，OBS）是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力。

在使用ModelArts时存在与OBS的数据交互，您需要使用的数据可以存储至OBS。

- 弹性云服务器

  弹性云服务器（Elastic Cloud Server，ECS）是由CPU、内存、操作系统、云硬盘组成的基础的计算组件。弹性云服务器创建成功后，您就可以像使用自己的本地PC或物理服务器一样，使用弹性云服务器。

  在制作自定义镜像时，您可以在本地环境或者ECS上完成自定义镜像制作。

说明

在您使用自定义镜像功能时，ModelArts可能需要访问您的容器镜像服务SWR、对象存储服务OBS等依赖服务，如果没有授权，这些功能将不能正常使用。建议您使用委托授权功能，将依赖服务操作权限委托给ModelArts服务，让ModelArts以您的身份使用依赖服务，代替您进行一些资源操作。详细操作参见使用**委托授权**。

11.2 预置镜像介绍（主流的镜像）

11.2.1 预置镜像列表

ARM + Ascend 架构的预置镜像

表 11-1 MindSpore

预置镜像	适用范围
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Notebook、训练、推理部署
mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Notebook、训练、推理部署
mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3	Notebook、训练、推理部署
mindspore1.8.0-cann5.1.2-py3.7-euler2.8.3	Notebook
mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3	Notebook
mindspore_2.2.10-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Notebook、训练、推理部署

表 11-2 TensorFlow

预置镜像	适用范围
tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3	Notebook、训练、推理部署
tensorflow1.15.0-cann5.1.2-py3.7-euler2.8.3	Notebook

预置镜像	适用范围
tensorflow1.15-cann5.1.0-py3.7-euler2.8.3	Notebook

表 11-3 Pytorch

预置镜像	适用范围
pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Notebook、训练、推理部署
pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Notebook、训练、推理部署
pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3	Notebook、训练、推理部署
pytorch_2.1.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Notebook、训练、推理部署
pytorch_1.11.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Notebook、训练、推理部署

11.2.2 预置镜像 ARM MindSpore

镜像一：mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b

表 11-4 mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b

AI引擎框架	URL	包含的依赖项	
mindspore 2.2.0 + mindspore-lite 2.2.0 + Ascend CANN Toolkit 7.0.RC1	swr.<region>.myhuaweicloud.com/atelier/mindspore_2_2_ascend:mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83	PyPI 程序包	Yum 软件包

AI引擎框架	URL	包含的依赖项	
		mindspore 2.2.0 ipykernel 6.7.0 ipython 8.17.2 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.22.0 pandas 1.2.5 pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.3.3 mindinsight 2.2.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

镜像二：mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b

表 11-5 mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b 镜像介绍

AI引擎框架	URL	包含的依赖项	
mindspore 2.1.0 + mindspore-lite 2.1.0 + Ascend CANN Toolkit 6.3.RC2	swr.{region-id}.{局点域名}/atelier/mindspore_2_0_ascend:mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b-20231009152946-e7b7e70	PyPI 程序包	Yum 软件包
		mindspore 2.1.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.21.6 pandas 1.3.5 pillow 9.5.0 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.7.3 scikit-learn 1.0.2 tornado 6.2 mindinsight 2.1.0	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

镜像三：mindspore1.8.0-cann5.1.2-py3.7-euler2.8.3

表 11-6 mindspore1.8.0-cann5.1.2-py3.7-euler2.8.3 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
mindspore 1.8.0	是 cann5.1.2	swr.{region-id}.{局点域名}/atelier/ mindspore_1_8_ascend:mindspore_1.8.0-cann_5.1.2-py_3.7-euler_2.8.3-aarch64-d910-20221009094203-4025e09	PyPI 程序包	Yum 软件包
			mindspore-ascend 1.8.0 mindinsight 1.8.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.3 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.18 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.21.2 pandas 1.1.3 pillow 9.2.0 pip 22.1.2 psutil 5.7.0 PyYAML 5.3.1 scipy 1.5.4 scikit-learn 0.24.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像四：mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3

表 11-7 mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
Mindspore 1.7.0	是（CANN 5.1）	swr.{region-id}.{局点域名}/atelier/mindspore_1_7_0:mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906 例如： swr.cn-central-231.xckpjs.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906 swr.cn-southwest-228.cdzs.cn/atelier/mindspore_1_7_0:mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906	PyPI 程序包	Yum 软件包
			mindspore-ascend 1.7.0 mindinsight 1.7.0 ipykernel 5.3.4 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.2 ma-cau-adapter 1.1.2 ma-cli 1.1.3 matplotlib 3.5.1 modelarts 1.4.7 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.21.2 pandas 1.1.3 pillow 9.2.0 pip 22.1.2 psutil 5.7.0 PyYAML 5.3.1 scipy 1.5.4 scikit-learn 0.24.0 tensorboard 1.15.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像五：mindspore_2.2.10-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c

表 11-8 mindspore_2.2.10-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c 镜像介绍

AI引擎框架	URL	包含的依赖项	
mindspore 2.2.10 + mindspore-lite 2.2.10 + Ascend CANN Toolkit 8.0.rc1	swr.<{region-id}.{局点域名}/atelier/mindspore_2_2_ascend:mindspore_2.2.10-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c-20240301174404-d5e7cea	PyPI 程序包	Yum 软件包
		ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.22.0 pandas 1.3.5 pillow 10.0.1 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

11.2.3 预置镜像 ARM Tensorflow

镜像一：tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3

表 11-9 tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
tensorflow_1.15.0	是 cann6.3.0	swr.{region-id}.{局点域名}/atelier/ tensorflow_1_15_ascend:tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64-d910-20230425164623-34300db	PyPI 程序包	Yum 软件包
			tensorflow 1.15.0 tensorboard 1.15.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cau 1.1.3 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.18 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.17.5 pandas 0.24.2 pillow 9.5.0 pip 21.0.1 psutil 5.7.0 PyYAML 5.3.1 scipy 1.3.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像二：tensorflow1.15-cann5.1.0-py3.7-euler2.8.3

表 11-10 tensorflow1.15-cann5.1.0-py3.7-euler2.8.3 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
Tensorflow 1.15	是（CANN 5.1）	swr.{region-id}.{局点域名}/atelier/tensorflow_1_15_ascend:tensorflow_1.15-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906 例如： swr.cn-central-231.xckpjs.com/atelier/tensorflow_1_15_ascend:tensorflow_1.15-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906 swr.cn-southwest-228.cdzs.cn/atelier/tensorflow_1_15_ascend:tensorflow_1.15-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906	PyPI 程序包	Yum 软件包
			tensorflow 1.15.0 tensorboard 1.15.0 ipykernel 5.3.4 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.2 ma-cau-adapter 1.1.2 ma-cli 1.1.3 matplotlib 3.5.1 modelarts 1.4.7 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.17.5 pandas 0.24.2 pillow 9.2.0 pip 22.1.2 psutil 5.7.0 PyYAML 5.3.1 scipy 1.3.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像三：tensorflow1.15.0-cann5.1.2-py3.7-euler2.8.3

表 11-11 tensorflow1.15.0-cann5.1.2-py3.7-euler2.8.3 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
tensorflow 1.15.0	是 cann5.1.2	swr.{region-id}.{局点域名}/atelier/ tensorflow_1_15_ascend:tensorflow_1.15.0-cann_5.1.2-py_3.7-euler_2.8.3-aarch64-d910-20221009094203-4025e09	PyPI 程序包	Yum 软件包
			tensorflow 1.15.0 tensorboard 1.15.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.3 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.18 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.17.5 pandas 0.24.2 pillow 9.2.0 pip 22.1.2 psutil 5.7.0 PyYAML 5.3.1 scipy 1.7.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

11.2.4 预置镜像 ARM PyTorch

镜像一：pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3

表 11-12 pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
pytorch_1.11.0	是 cann_6.3.0	swr.{region-id}.{局点域名}/atelier/ pytorch_1_11_ascend:pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64-d910-20230425164623-34300db	PyPI 程序包	Yum 软件包
			torch 1.11.0 torch-npu 1.11.0.dev20230417 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cau 1.1.3 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.18 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.21.2 pandas 0.24.2 pillow 9.5.0 pip 21.0.1 psutil 5.7.0 PyYAML 5.3.1 scipy 1.3.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像二：pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b

表 11-13 pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
pytorch_1.11.0	是 cann_6.3.2	swr.{region-id}.{局点域名}/atelier/ pytorch_1_11_ascend:pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b-20231009152946-e7b7e70	PyPI 程序包	Yum 软件包
			torch 1.11.0 torch-npu 1.11.0.dev20230417 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cau 1.1.3 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.18 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.21.2 pandas 0.24.2 pillow 9.5.0 pip 21.0.1 psutil 5.7.0 PyYAML 5.3.1 scipy 1.3.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像三：pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b

表 11-14 pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b 镜像介绍

AI引擎框架	是否使用昇腾（CANN版本）	URL	包含的依赖项	
pytorch_1.11.0	是 cann_7.0.1	swr.{region-id}.{局点域名}/atelier/pytorch_1_11_ascend:pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b-20231107190844-50a1a83	PyPI 程序包	Yum 软件包
			torch 1.11.0 torch-npu 1.11.0.dev20230417 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.4.9 ma-cau 1.1.3 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.18 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.21.2 pandas 0.24.2 pillow 9.5.0 pip 21.0.1 psutil 5.7.0 PyYAML 5.3.1 scipy 1.3.3 scikit-learn 0.20.0 tornado 6.2	ca-certificates.noarch cmake cpp curl gcc-c++ gcc gdb grep nginx python3 rpm tar unzip vim wget zip

镜像四：pytorch_2.1.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c

表 11-15 pytorch_2.1.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c 镜像介绍

AI引擎框架	URL	包含的依赖项	
pytorch 2.1.0 + mindspore-lite 2.2.10 + Ascend CANN Toolkit 8.0.rc1	swr.{region-id}.{局点域名}/atelier/pytorch_2_1_ascend:pytorch_2.1.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c-20240301174404-d5e7cea	PyPI 程序包	Yum 软件包
		ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.22.0 pandas 1.3.5 pillow 10.2.0 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

镜像五：pytorch_1.11.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c

表 11-16 pytorch_1.11.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c 镜像介绍

AI引擎框架	URL	包含的依赖项	
pytorch 1.11.0 + mindspore-lite 2.2.10 + Ascend CANN Toolkit 8.0.rc1	swr.{region-id}.{局点域名}/atelier/pytorch_1_11_ascend:pytorch_1.11.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c-20240301174404-d5e7cea	PyPI 程序包	Yum 软件包
		ipykernel 6.7.0 ipython 8.18.1 jupyter-client 7.4.9 ma-cau 1.1.7 ma-cau-adapter 1.1.3 ma-cli 1.2.3 matplotlib 3.5.1 modelarts 1.4.20 moxing-framework 2.2.3.2c7f2141 numpy 1.22.0 pandas 1.3.5 pillow 10.2.0 pip 21.0.1 psutil 5.9.5 PyYAML 6.0.1 scipy 1.10.1 scikit-learn 1.0.2 tornado 6.4	cmake cpp curl ffmpeg g++ gcc git grep python3 rpm tar unzip wget zip

11.3 Notebook 中使用自定义镜像

11.3.1 在 ModelArts 中进行镜像注册

用户的自定义镜像构建完成后，需要在ModelArts“镜像管理”页面注册后，方可在Notebook中使用。

说明

SWR镜像类型设置为“私有”时，同一账号下的子用户（IAM用户）可以注册使用。
SWR镜像类型设置为“公开”时，其它用户才可以注册使用。

镜像注册

1. 进入ModelArts控制台，单击“镜像管理 > 注册镜像”，进入“注册镜像”页面。

2. 根据界面提示填写相关信息，然后单击“立即注册”。
   - “镜像源”选择构建好的镜像。可直接拷贝完整的SWR地址，或单击 选择SWR构建好的镜像进行注册。
   - “架构”和“类型”：根据自定义镜像的实际框架选择。
3. 注册后的镜像会显示在镜像管理页面。

镜像查找

在镜像列表页面，可根据“名称”、“所属组织”、“版本”等属性进行筛选，快速查找镜像。

图 11-2 查找镜像

创建 Notebook

单击镜像名称，进入镜像详情页面，单击“创建Notebook”，会跳转到基于该自定义镜像创建Notebook的页面。

图 11-3 进入镜像详情页面

镜像同步

当用户完成镜像故障排除后，进入镜像详情页，单击操作列的“镜像同步”完成镜像状态的刷新。

图 11-4 镜像同步

11.3.2 Notebook 制作自定义镜像方法

制作自定义镜像方法：

- 方式一：使用Notebook的预置镜像创建开发环境实例，在环境中进行依赖安装与配置，配置完成后，可以通过开发环境提供的镜像保存功能，将运行实例的内容以容器镜像的方式保存下来，作为自定义镜像使用。详细操作请参考**将Notebook实例保存为自定义镜像**。
- 方式二：基于ModelArts提供的基础镜像或第三方镜像，在ECS服务器上自行编写Dockerfile构建镜像，对ModelArts基础镜像或第三方镜像进行改造，构建出符合新的自定义Docker镜像，并将镜像推送到SWR，作为自定义镜像使用。详细操作请参考**在Notebook中构建自定义镜像并使用**。

11.3.3 将 Notebook 实例保存为自定义镜像

11.3.3.1 保存 Notebook 镜像环境

通过预置的镜像创建Notebook实例，在基础镜像上安装对应的自定义软件和依赖，在管理页面上进行操作，进而完成将运行的实例环境以容器镜像的方式保存下来。

保存的镜像中，安装的依赖包不丢失，持久化存储的部分（home/ma-user/work目录的内容）不会保存在最终产生的容器镜像中。VS Code远程开发场景下，在Server端安装的插件不丢失。

说明

Notebook中保存的镜像大小不超过35G，镜像层数不能超过125层。否则镜像会保存失败。

若镜像保存时报错“The container size (xx) is greater than the threshold (25G)”，请参考**镜像保存时报错“The container size (xG) is greater than the threshold (25G)”如何解决?**处理。

前提条件

Notebook实例状态为“运行中”。

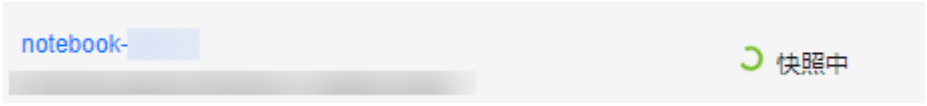
保存镜像

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“开发环境 > Notebook”，进入新版Notebook管理页面。
2. 在Notebook列表中，对于要保存的Notebook实例，单击右侧“操作”列中的“更多 > 保存镜像”，进入“保存镜像”对话框。
3. 在保存镜像对话框中，设置组织、镜像名称、镜像版本和描述信息。单击“确定”保存镜像。

   在“组织”下拉框中选择一个组织。如果没有组织，可以单击右侧的“立即创建”，创建一个组织。

   同一个组织内的用户可以共享使用该组织内的所有镜像。
4. 镜像会以快照的形式保存，保存过程约5分钟，请耐心等待。此时不可再操作实例。

图 11-5 保存镜像



须知

快照中耗费的时间仍占用实例的总运行时长，若在快照中时，实例因运行时间到期停止，将导致镜像保存失败。

5. 镜像保存成功后，实例状态变为“运行中”，用户可在“镜像管理”页面查看到该镜像详情。
6. 单击镜像的名称，进入镜像详情页，可以查看镜像版本/ID，状态，资源类型，镜像大小，SWR地址等。

11.3.3.2 基于自定义镜像创建 Notebook 实例

从Notebook中保存的镜像可以在镜像管理中查询到，可以用于创建新的Notebook实例，完全继承保存状态下的实例软件环境配置。

基于自定义镜像创建Notebook实例有两种方式：

方式一：在Notebook实例创建页面，镜像类型选择“自定义镜像”，名称选择上述保存的镜像。

图 11-6 创建基于自定义镜像的 Notebook 实例

方式二：在“镜像管理”页面，单击某个镜像的镜像详情，在镜像详情页，单击“创建Notebook”，也会跳转到基于该自定义镜像创建Notebook的页面。

11.3.4 在 Notebook 中构建自定义镜像并使用

11.3.4.1 使用场景和构建流程说明

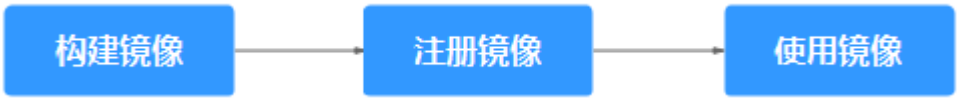
在预置镜像不满足客户使用诉求时，可以基于预置镜像自行构建容器镜像用于开发和训练。

用户在使用ModelArts开发环境时，经常需要对开发环境进行一些改造，如安装、升级或卸载一些包。但是某些包的安装升级需要root权限，运行中的Notebook实例中无root权限，所以在Notebook实例中安装需要root权限的软件，目前在预置的开发环境镜像中是无法实现的。

此时，用户可以使用ModelArts提供的基础镜像来编写Dockerfile，构建出完全适合自己的镜像。进一步可以调试该镜像，确保改造后的镜像能够在ModelArts服务中正常使用。最终将镜像进行注册，用以创建新的开发环境，满足自己的业务需求。

本案例将基于ModelArts提供的MindSpore基础镜像，并借助ModelArts命令行工具(请参考**ma-cli镜像构建命令介绍**)制作和注册镜像，构建一个面向AI开发的自定义镜像。主要流程如下图所示：

图 11-7 构建镜像流程

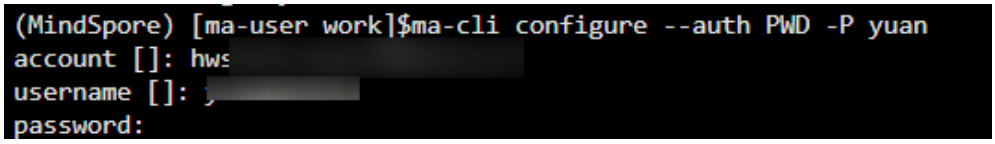


11.3.4.2 Step1 制作自定义镜像

本节描述通过加载镜像构建模板并编写Dockerfile，构建出一个新镜像。如下的步骤都需要在“开发环境 > Notebook”的Terminal中完成，请提前创建好开发环境并打开Terminal。关于Dockerfile的具体编写方法，请参考**官网**。

步骤1 首先配置鉴权信息，指定profile，根据提示输入账号、用户名及密码。鉴权更多信息请查看**配置登录信息**。

```
ma-cli configure --auth PWD -P xxx
```



步骤2 执行**env|grep -i CURRENT_IMAGE_NAME**命令查询当前实例所使用的镜像。

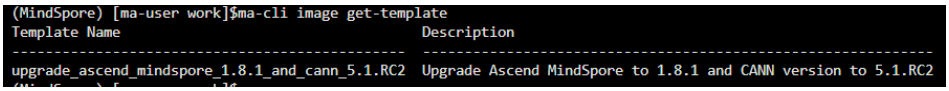
步骤3 制作新镜像。

1. 获取上步查询的基础镜像的SWR地址。

   CURRENT_IMAGE_NAME=swr.cn-south-222.ai.pcl.cn/atelier/mindspore_1_7_0:mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220906

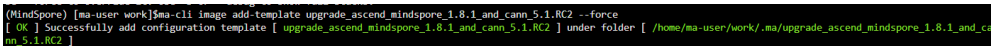
2. 加载镜像构建模板。

   执行**ma-cli image get-template**命令查询镜像模板。

   然后执行**ma-cli image add-template**命令将镜像模板加载到指定文件夹下，默认路径为当前命令所在的路径。例如：加载upgrade_ascend_mindspore_1.8.1_and_cann_5.1.RC2镜像构建模板。

   ```
   ma-cli image add-template upgrade_ascend_mindspore_1.8.1_and_cann_5.1.RC2
   ```

3. 修改Dockerfile。

   本例的Dockerfile将基于MindSpore基础镜像mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3，升级到cann 5.1.RC2和MindSpore1.8.1，构建一个面向AI任务的镜像。

   加载镜像模板后，Dockerfile文件自动加载，在“.ma/upgrade_ascend_mindspore_1.8.1_and_cann_5.1.RC2”路径下，内容参考如下，根据实际需求修改：

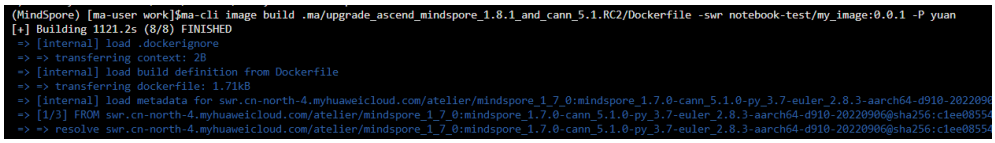
```
#The following uses Mindspore-1.7 as an example, which can be replaced with the current notebook image.（请替换成当前Notebook使用的镜像）
FROM swr.cn-south-222.ai.pcl.cn/atelier/mindspore_1_7_0:mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64-d910-20220715093657-9446c6a

ARG CANN=Ascend-cann-toolkit_5.1.RC2_linux-aarch64.run

# Set proxy to download internet resources（根据实际修改Notebook代理）
ENV HTTP_PROXY=http://proxy.modelarts.com:80 \
    http_proxy=http://proxy.modelarts.com:80 \
    HTTPS_PROXY=http://proxy.modelarts.com:80 \
    https_proxy=http://proxy.modelarts.com:80

USER root

# Download CANN-5.1.RC2 and install CANN package, which is a dependency package for mindspore-1.8.1.
# For details about the mapping between Mindpore and CANN and the download address of CANN, see the official website of Mindpore.
RUN wget https://ascend-repo.obs.cn-east-2.myhuaweicloud.com/CANN/CANN%205.1.RC2/${CANN} -P /tmp && \
    chmod +x /tmp/${CANN} && \
    sh -x /tmp/${CANN} --quiet --full && \
    rm -f /tmp/${CANN}

ENV PYTHONPATH=/usr/local/Ascend/tfplugin/latest/python/site-packages:/usr/local/Ascend/ascend-toolkit/latest/python/site-packages:/usr/local/Ascend/ascend-toolkit/latest/opp/op_impl/built-in/ai_core/tbe:/usr/local/seccomponent/lib

USER ma-user

# Update mindspore version in "MindSpore" conda env by using pip.
RUN source /home/ma-user/anaconda3/bin/activate MindSpore && \
    pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.8.1/MindSpore/ascend/aarch64/mindspore_ascend-1.8.1-cp37-cp37m-linux_aarch64.whl --upgrade && \
    echo "successfully install mindspore 1.8.1"

## [Optional] Uncomment to set default conda env
#ENV DEFAULT_CONDA_ENV_NAME=/home/ma-user/anaconda3/envs/MindSpore
```

4. 构建镜像

   使用**ma-cli image build**命令从Dockerfile构建出一个新镜像。命令更多信息请参考**镜像构建命令**。

   ```
   ma-cli image build .ma/upgrade_ascend_mindspore_1.8.1_and_cann_5.1.RC2/Dockerfile -swr notebook-test/my_image:0.0.1 -P XXX
   ```

   其中“.ma/upgrade_ascend_mindspore_1.8.1_and_cann_5.1.RC2/Dockerfile”为Dockerfile文件所在路径，“notebook-test/my_image:0.0.1”为构建的新镜像的SWR路径。“XXX”为鉴权时指定的profile。

----结束

11.3.4.3 Step2 注册新镜像

调试完成后，将新镜像注册到ModelArts镜像管理服务中，进而能够在ModelArts中使用该镜像。

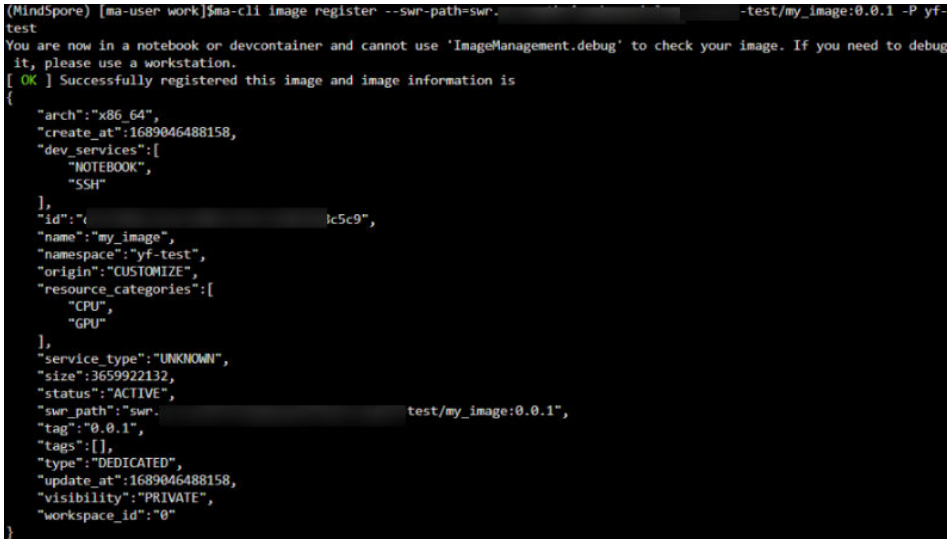
有两种方式来注册镜像。

- **方式一：使用ma-cli image register命令来注册镜像。**注册命令会返回注册好的镜像信息，包括镜像id，name等，如下图所示。该命令的更多信息可参考**注册镜像**。

  ```
  ma-cli image register --swr-path=swr.cn-south-222.ai.pcl.cn/cloud-test/mindspore_1_8:v1 -a AARCH64 -rs ASCEND -P XXX
  ```

  -a指定该镜像支持ARM架构，-rs指定镜像支持ASCEND芯片，“XXX”为鉴权时指定的profile。

  图 11-8 注册镜像

- **方式二：在ModelArts Console上注册镜像**

  登录ModelArts控制台，在左侧导航栏选择“镜像管理”，进入镜像管理页面。

  a. 单击“注册镜像”。请将完整的SWR地址复制到这里即可，或单击 可直接从SWR选择自有镜像进行注册。

  b. “架构”和“类型”根据实际情况选择，与镜像源保持一致。

11.3.4.4 Step3 创建开发环境并使用

创建开发环境

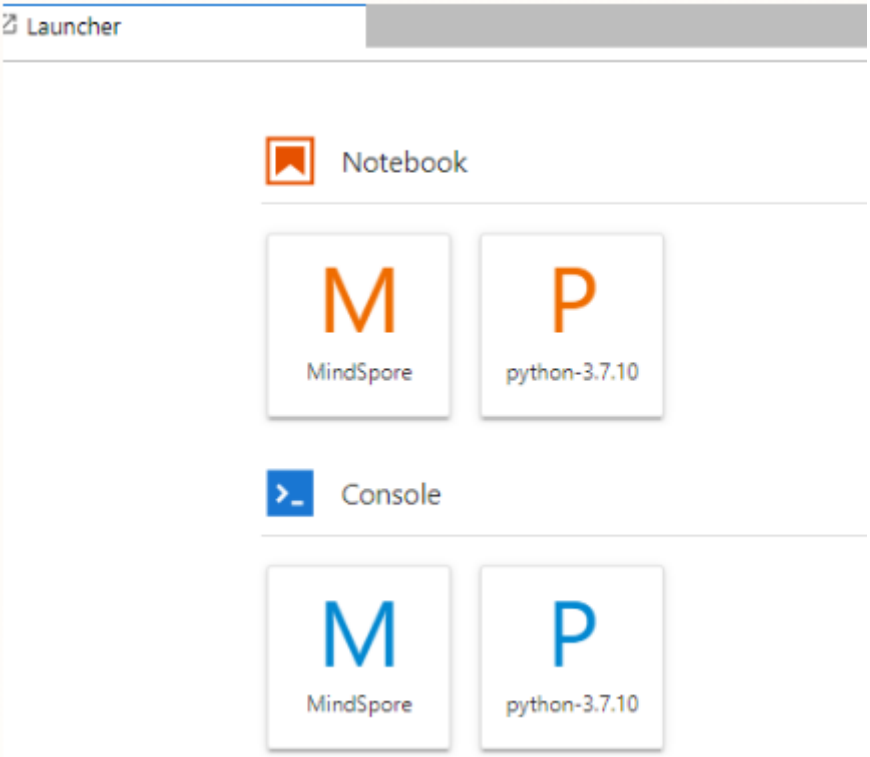
1. 镜像注册成功后，即可在ModelArts控制台的“开发环境 > Notebook”页面，创建开发环境时选择该自定义镜像。

   图 11-9 创建开发环境

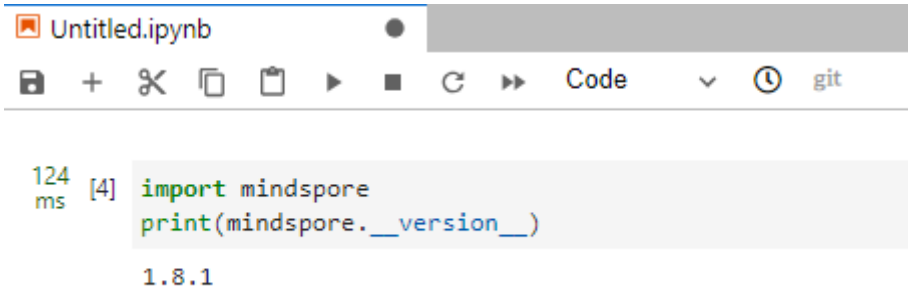
2. 打开开发环境。

图 11-10 打开开发环境



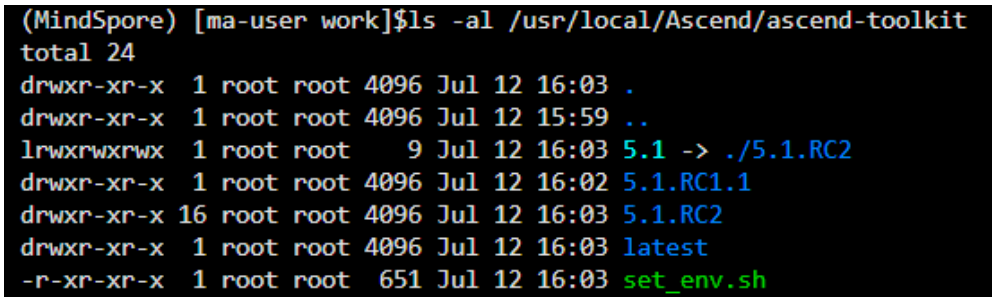
3. 单击图中的MindSpore，即可创建一个ipynb文件，导入mindspore，可以看到安装的mindspore 1.8.1已经能够使用。

图 11-11 创建一个 ipynb 文件



4. 再打开一个terminal，查看cann的版本，是Dockerfile中安装的版本。

图 11-12 查看 cann 版本



11.4 使用自定义镜像训练模型（模型训练）

11.4.1 训练管理中使用自定义镜像介绍

订阅算法和预置框架涵盖了大部分的训练场景。针对特殊场景，ModelArts支持用户构建自定义镜像用于模型训练。

自定义镜像的制作要求用户对容器相关知识有比较深刻的了解，除非订阅算法和预置框架无法满足需求，否则不推荐使用。自定义镜像需上传至容器镜像服务（SWR），才能用于ModelArts上训练。

ModelArts上使用自定义镜像训练支持2种方式：

- 使用预置框架 + 自定义镜像：

  如果先前基于预置框架且通过指定代码目录和启动文件的方式来创建的训练作业；但是随着业务逻辑的逐渐复杂，您期望可以基于预置框架修改或增加一些软件依赖的时候，此时您可以使用预置框架 + 自定义镜像的功能，即选择预置框架名称后，在预置框架版本下拉列表中选择“自定义”。
- 完全自定义镜像：

  用户遵循ModelArts镜像的规范要求制作镜像，选择自己的镜像，并且通过指定代码目录（可选）和启动命令的方式来创建的训练作业。

  说明

  当使用完全自定义镜像创建训练作业时，“启动命令”必须在“/home/ma-user”目录下执行，否则训练作业可能会运行异常。

使用预置框架 + 自定义镜像

此功能与直接基于预置框架创建训练作业的区别仅在于，镜像是由用户自行选择的。用户可以基于预置框架制作自定义镜像。基于预置框架制作自定义镜像可参考**使用基础镜像构建新的训练镜像**章节。

图 11-13 使用预置框架+自定义镜像创建算法

该功能的行为与直接基于预置框架创建的训练作业相同，例如：

- 系统将会自动注入一系列环境变量

  - PATH=${MA_HOME}/anaconda/bin:${PATH}
  - LD_LIBRARY_PATH=${MA_HOME}/anaconda/lib:${LD_LIBRARY_PATH}
  - PYTHONPATH=${MA_JOB_DIR}:${PYTHONPATH}
- 您选择的启动文件将会被系统自动以python命令直接启动，因此请确保镜像中的Python命令为您预期的Python环境。注意到系统自动注入的PATH环境变量，您可以参考下述命令确认训练作业最终使用的Python版本：
  - export MA_HOME=/home/ma-user; docker run --rm {image} ${MA_HOME}/anaconda/bin/python -V
  - docker run --rm {image} $(which python) -V
- 系统将会自动添加预置框架关联的超参

完全使用自定义镜像

图 11-14 完全使用自定义镜像创建算法

新版训练支持的自定义镜像使用说明，请参考**使用自定义镜像创建训练作业**。

完全使用自定义镜像场景下，指定的“conda env”启动训练方法如下：

由于训练作业运行时不是shell环境，因此无法直接使用“conda activate”命令激活指定的 “conda env”，需要使用其他方式以达成使用指定“conda env”来启动训练的效果。

假设您的自定义镜像中的“conda”安装于“/home/ma-user/anaconda3”目录“conda env”为“python-3.7.10”，训练脚本位于“/home/ma-user/modelarts/user-job-dir/code/train.py”。可通过以下方式使用指定的“conda env”启动训练：

- 方式一：为镜像设置正确的“DEFAULT_CONDA_ENV_NAME”环境变量与“ANACONDA_DIR”环境变量。

  您可以使用Python命令启动训练脚本。启动命令示例如下：
  ```
  python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```
- 方式二：使用“conda env python”的绝对路径。

  您可以使用“/home/ma-user/anaconda3/envs/python-3.7.10/bin/python”命令启动训练脚本。启动命令示例如下：
  ```
  /home/ma-user/anaconda3/envs/python-3.7.10/bin/python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```

- 方式三：设置PATH环境变量。

  您可以将指定的“conda env bin”目录配置到PATH环境变量中。您可以使用Python命令启动训练脚本。启动命令示例如下：

  ```
  export PATH=/home/ma-user/anaconda3/envs/python-3.7.10/bin:$PATH; python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```

- 方式四：使用“conda run -n”命令。

  您可以使用“/home/ma-user/anaconda3/bin/conda run -n python-3.7.10”命令来执行训练命令，启动命令示例如下：

  ```
  /home/ma-user/anaconda3/bin/conda run -n python-3.7.10 python /home/ma-user/modelarts/user-job-dir/code/train.py
  ```

说明

如果在训练时发生找不到“$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib ”目录下“.so”文件的相关报错，可以尝试将该目录加入到“LD_LIBRARY_PATH”，将以下命令放在上述启动方式命令前：

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib:$LD_LIBRARY_PATH;
```

例如，方式一的启动命令示例此时变为：

```
export LD_LIBRARY_PATH=$ANACONDA_DIR/envs/$DEFAULT_CONDA_ENV_NAME/lib:$LD_LIBRARY_PATH; python /home/ma-user/modelarts/user-job-dir/code/train.py
```

11.4.2 示例：从 0 到 1 制作自定义镜像并用于训练

11.4.2.1 示例：从 0 到 1 制作自定义镜像并用于开发和训练（MindSpore +Ascend）

11.4.2.1.1 场景描述

本案例介绍如何从0到1制作Ascend容器镜像，并使用该镜像在ModelArts平台上进行训练。镜像中使用的AI引擎是MindSpore，训练使用的资源是专属资源池的Ascend芯片。

约束限制

- 由于案例中需要下载商用版CANN，因此本案例仅面向有下载权限的渠道用户，非渠道用户建议参考其他自定义镜像制作教程。
- Mindspore版本与CANN版本，CANN版本与Ascend驱动/固件版本均有严格的匹配关系，版本不匹配会导致训练失败。

场景描述

目标：构建安装如下软件的容器镜像，并在ModelArts平台上使用Ascend规格资源运行训练任务。

- ubuntu-18.04
- cann-6.3.RC2 (商用版本)
- python-3.7.13
- mindspore-2.1.1

说明

- 本教程以cann-6.3.RC2.、mindspore-2.1.1为例介绍。
- 本示例仅用于示意Ascend容器镜像制作流程，且在匹配正确的Ascend驱动/固件版本的专属资源池上运行通过。

操作流程

使用自定义镜像创建训练作业时，需要您熟悉 docker 软件的使用，并具备一定的开发经验。详细步骤如下所示：

1. **Step1 创建OBS桶和文件夹**
2. **Step2 准备脚本文件并上传至OBS中**
3. **Step3 制作自定义镜像**
4. **Step4 上传镜像至SWR**
5. **Step5 在ModelArts上创建Notebook并调试**
6. **Step6 在ModelArts上创建训练作业**

11.4.2.1.2 Step1 创建 OBS 桶和文件夹

Step1 创建 OBS 桶和文件夹

在 OBS 服务中创建桶和文件夹，用于存放样例数据集以及训练代码。如下示例中，请创建命名为 “test-modelarts” 的桶，并创建如**表11-17**所示的文件夹。

表 11-17 OBS 桶文件夹列表

文件夹名称	用途
obs://test-modelarts/ascend/demo-code/	用于存储 Ascend 训练脚本文件。
obs://test-modelarts/ascend/demo-code/run_ascend/	用于存储 Ascend 训练脚本的启动脚本。
obs://test-modelarts/ascend/log/	用于存储训练日志文件。

11.4.2.1.3 Step2 准备脚本文件并上传至 OBS 中

1. 准备本案例所需训练脚本 mindspore-verification.py 文件和 Ascend 的启动脚本文件（共5个）。
   - 训练脚本文件具体内容请参见**训练脚本 mindspore-verification.py 文件**。
   - Ascend 的启动脚本文件包括以下5个，具体脚本内容请参见**Ascend 的启动脚本文件**。
     i. run_ascend.py
     ii. common.py
     iii. rank_table.py
     iv. manager.py

v. fmk.py

说明

mindspore-verification.py 和run_ascend.py脚本文件在创建训练作业时的“启动命令”参数中调用，具体请参见**•启动命令：“python ${MA_JOB_D...**。

run_ascend.py脚本运行时会调用common.py、rank_table.py、manager.py、fmk.py脚本。

2. 上传训练脚本 mindspore-verification.py 文件至OBS桶的“obs://test-modelarts/ascend/demo-code/”文件夹下。
3. 上传Ascend的启动脚本文件（共5个）至OBS桶的“obs://test-modelarts/ascend/demo-code/run_ascend/”文件夹下。

训练脚本 mindspore-verification.py 文件

mindspore-verification.py 文件内容如下：

```
import os
import numpy as np
from mindspore import Tensor
import mindspore.ops as ops
import mindspore.context as context

print('Ascend Envs')
print('------')
print('JOB_ID: ', os.environ['JOB_ID'])
print('RANK_TABLE_FILE: ', os.environ['RANK_TABLE_FILE'])
print('RANK_SIZE: ', os.environ['RANK_SIZE'])
print('ASCEND_DEVICE_ID: ', os.environ['ASCEND_DEVICE_ID'])
print('DEVICE_ID: ', os.environ['DEVICE_ID'])
print('RANK_ID: ', os.environ['RANK_ID'])
print('------')

context.set_context(device_target="Ascend")
x = Tensor(np.ones([1,3,3,4]).astype(np.float32))
y = Tensor(np.ones([1,3,3,4]).astype(np.float32))

print(ops.add(x, y))
```

Ascend 的启动脚本文件

- run_ascend.py

```
import sys
import os

from common import RunAscendLog
from common import RankTableEnv

from rank_table import RankTable, RankTableTemplate1, RankTableTemplate2

from manager import FMKManager

if __name__ == '__main__':
    log = RunAscendLog.setup_run_ascend_logger()

    if len(sys.argv) <= 1:
        log.error('there are not enough args')
        sys.exit(1)

    train_command = sys.argv[1:]
    log.info('training command')
    log.info(train_command)

    if os.environ.get(RankTableEnv.RANK_TABLE_FILE_V1) is not None:
```

```
        # new format rank table file
        rank_table_path = os.environ.get(RankTableEnv.RANK_TABLE_FILE_V1)
        RankTable.wait_for_available(rank_table_path)
        rank_table = RankTableTemplate1(rank_table_path)
    else:
        # old format rank table file
        rank_table_path_origin = RankTableEnv.get_rank_table_template2_file_path()
        RankTable.wait_for_available(rank_table_path_origin)
        rank_table = RankTableTemplate2(rank_table_path_origin)

    if rank_table.get_device_num() >= 1:
        log.info('set rank table %s env to %s' % (RankTableEnv.RANK_TABLE_FILE,
rank_table.get_rank_table_path()))
        RankTableEnv.set_rank_table_env(rank_table.get_rank_table_path())
    else:
        log.info('device num < 1, unset rank table %s env' % RankTableEnv.RANK_TABLE_FILE)
        RankTableEnv.unset_rank_table_env()

    instance = rank_table.get_current_instance()
    server = rank_table.get_server(instance.server_id)
    current_instance = RankTable.convert_server_to_instance(server)

    fmk_manager = FMKManager(current_instance)
    fmk_manager.run(rank_table.get_device_num(), train_command)
    return_code = fmk_manager.monitor()

    fmk_manager.destroy()

    sys.exit(return_code)
```

- common.py

```
import logging
import os

logo = 'Training'


# Rank Table Constants
class RankTableEnv:
    RANK_TABLE_FILE = 'RANK_TABLE_FILE'

    RANK_TABLE_FILE_V1 = 'RANK_TABLE_FILE_V_1_0'

    HCCL_CONNECT_TIMEOUT = 'HCCL_CONNECT_TIMEOUT'

    # jobstart_hccl.json is provided by the volcano controller of Cloud-Container-Engine(CCE)
    HCCL_JSON_FILE_NAME = 'jobstart_hccl.json'

    RANK_TABLE_FILE_DEFAULT_VALUE = '/user/config/%s' % HCCL_JSON_FILE_NAME

    @staticmethod
    def get_rank_table_template1_file_dir():
        parent_dir = os.environ[ModelArts.MA_MOUNT_PATH_ENV]
        return os.path.join(parent_dir, 'rank_table')

    @staticmethod
    def get_rank_table_template2_file_path():
        rank_table_file_path = os.environ.get(RankTableEnv.RANK_TABLE_FILE)
        if rank_table_file_path is None:
            return RankTableEnv.RANK_TABLE_FILE_DEFAULT_VALUE

        return os.path.join(os.path.normpath(rank_table_file_path),
RankTableEnv.HCCL_JSON_FILE_NAME)

    @staticmethod
    def set_rank_table_env(path):
        os.environ[RankTableEnv.RANK_TABLE_FILE] = path

    @staticmethod
    def unset_rank_table_env():
```

```
        del os.environ[RankTableEnv.RANK_TABLE_FILE]


class ModelArts:
    MA_MOUNT_PATH_ENV = 'MA_MOUNT_PATH'
    MA_CURRENT_INSTANCE_NAME_ENV = 'MA_CURRENT_INSTANCE_NAME'
    MA_VJ_NAME = 'MA_VJ_NAME'

    MA_CURRENT_HOST_IP = 'MA_CURRENT_HOST_IP'

    CACHE_DIR = '/cache'

    TMP_LOG_DIR = '/tmp/log/'

    FMK_WORKSPACE = 'workspace'

    @staticmethod
    def get_current_instance_name():
        return os.environ[ModelArts.MA_CURRENT_INSTANCE_NAME_ENV]

    @staticmethod
    def get_current_host_ip():
        return os.environ.get(ModelArts.MA_CURRENT_HOST_IP)

    @staticmethod
    def get_job_id():
        ma_vj_name = os.environ[ModelArts.MA_VJ_NAME]
        return ma_vj_name.replace('ma-job', 'modelarts-job', 1)

    @staticmethod
    def get_parent_working_dir():
        if ModelArts.MA_MOUNT_PATH_ENV in os.environ:
            return os.path.join(os.environ.get(ModelArts.MA_MOUNT_PATH_ENV),
ModelArts.FMK_WORKSPACE)

        return ModelArts.CACHE_DIR


class RunAscendLog:

    @staticmethod
    def setup_run_ascend_logger():
        name = logo
        formatter = logging.Formatter(fmt='[run ascend] %(asctime)s - %(levelname)s - %(message)s')

        handler = logging.StreamHandler()
        handler.setFormatter(formatter)

        logger = logging.getLogger(name)
        logger.setLevel(logging.INFO)
        logger.addHandler(handler)
        logger.propagate = False
        return logger

    @staticmethod
    def get_run_ascend_logger():
        return logging.getLogger(logo)
```

- rank_table.py

```
import json
import time
import os

from common import ModelArts
from common import RunAscendLog
from common import RankTableEnv

log = RunAscendLog.get_run_ascend_logger()
```

```
class Device:
    def __init__(self, device_id, device_ip, rank_id):
        self.device_id = device_id
        self.device_ip = device_ip
        self.rank_id = rank_id


class Instance:
    def __init__(self, pod_name, server_id, devices):
        self.pod_name = pod_name
        self.server_id = server_id
        self.devices = self.parse_devices(devices)

    @staticmethod
    def parse_devices(devices):
        if devices is None:
            return []
        device_object_list = []
        for device in devices:
            device_object_list.append(Device(device['device_id'], device['device_ip'], ''))

        return device_object_list

    def set_devices(self, devices):
        self.devices = devices


class Group:
    def __init__(self, group_name, device_count, instance_count, instance_list):
        self.group_name = group_name
        self.device_count = int(device_count)
        self.instance_count = int(instance_count)
        self.instance_list = self.parse_instance_list(instance_list)

    @staticmethod
    def parse_instance_list(instance_list):
        instance_object_list = []
        for instance in instance_list:
            instance_object_list.append(
                Instance(instance['pod_name'], instance['server_id'], instance['devices']))

        return instance_object_list


class RankTable:
    STATUS_FIELD = 'status'
    COMPLETED_STATUS = 'completed'

    def __init__(self):
        self.rank_table_path = ""
        self.rank_table = {}

    @staticmethod
    def read_from_file(file_path):
        with open(file_path) as json_file:
            return json.load(json_file)

    @staticmethod
    def wait_for_available(rank_table_file, period=1):
        log.info('Wait for Rank table file at %s ready' % rank_table_file)
        complete_flag = False
        while not complete_flag:
            with open(rank_table_file) as json_file:
                data = json.load(json_file)
                if data[RankTable.STATUS_FIELD] == RankTable.COMPLETED_STATUS:
                    log.info('Rank table file is ready for read')
                    log.info('\n' + json.dumps(data, indent=4))
                    return True
```

```
            time.sleep(period)

        return False

    @staticmethod
    def convert_server_to_instance(server):
        device_list = []
        for device in server['device']:
            device_list.append(
                Device(device_id=device['device_id'], device_ip=device['device_ip'],
rank_id=device['rank_id']))

        ins = Instance(pod_name='', server_id=server['server_id'], devices=[])
        ins.set_devices(device_list)
        return ins

    def get_rank_table_path(self):
        return self.rank_table_path

    def get_server(self, server_id):
        for server in self.rank_table['server_list']:
            if server['server_id'] == server_id:
                log.info('Current server')
                log.info('\n' + json.dumps(server, indent=4))
                return server

        log.error('server [%s] is not found' % server_id)
        return None


class RankTableTemplate2(RankTable):

    def __init__(self, rank_table_template2_path):
        super().__init__()

        json_data = self.read_from_file(file_path=rank_table_template2_path)

        self.status = json_data[RankTableTemplate2.STATUS_FIELD]
        if self.status != RankTableTemplate2.COMPLETED_STATUS:
            return

        # sorted instance list by the index of instance
        # assert there is only one group
        json_data["group_list"][0]["instance_list"] = sorted(json_data["group_list"][0]["instance_list"],
                                                             key=RankTableTemplate2.get_index)

        self.group_count = int(json_data['group_count'])
        self.group_list = self.parse_group_list(json_data['group_list'])

        self.rank_table_path, self.rank_table = self.convert_template2_to_template1_format_file()

    @staticmethod
    def parse_group_list(group_list):
        group_object_list = []
        for group in group_list:
            group_object_list.append(
                Group(group['group_name'], group['device_count'], group['instance_count'],
group['instance_list']))

        return group_object_list

    @staticmethod
    def get_index(instance):
        # pod_name example: job94dc1dbf-job-bj4-yolov4-15
        pod_name = instance["pod_name"]
        return int(pod_name[pod_name.rfind("-") + 1:])

    def get_current_instance(self):
        """
```

```
        get instance by pod name
        specially, return the first instance when the pod name is None
        :return:
        """
        pod_name = ModelArts.get_current_instance_name()
        if pod_name is None:
            if len(self.group_list) > 0:
                if len(self.group_list[0].instance_list) > 0:
                    return self.group_list[0].instance_list[0]

            return None

        for group in self.group_list:
            for instance in group.instance_list:
                if instance.pod_name == pod_name:
                    return instance
        return None

    def convert_template2_to_template1_format_file(self):
        rank_table_template1_file = {
            'status': 'completed',
            'version': '1.0',
            'server_count': '0',
            'server_list': []
        }

        logic_index = 0
        server_map = {}
        # collect all devices in all groups
        for group in self.group_list:
            if group.device_count == 0:
                continue
            for instance in group.instance_list:
                if instance.server_id not in server_map:
                    server_map[instance.server_id] = []

                for device in instance.devices:
                    template1_device = {
                        'device_id': device.device_id,
                        'device_ip': device.device_ip,
                        'rank_id': str(logic_index)
                    }
                    logic_index += 1
                    server_map[instance.server_id].append(template1_device)

        server_count = 0
        for server_id in server_map:
            rank_table_template1_file['server_list'].append({
                'server_id': server_id,
                'device': server_map[server_id]
            })
            server_count += 1

        rank_table_template1_file['server_count'] = str(server_count)

        log.info('Rank table file (Template1)')
        log.info('\n' + json.dumps(rank_table_template1_file, indent=4))

        if not os.path.exists(RankTableEnv.get_rank_table_template1_file_dir()):
            os.makedirs(RankTableEnv.get_rank_table_template1_file_dir())

        path = os.path.join(RankTableEnv.get_rank_table_template1_file_dir(),
RankTableEnv.HCCL_JSON_FILE_NAME)
        with open(path, 'w') as f:
            f.write(json.dumps(rank_table_template1_file))
            log.info('Rank table file (Template1) is generated at %s', path)

        return path, rank_table_template1_file
```

```
    def get_device_num(self):
        total_device_num = 0
        for group in self.group_list:
            total_device_num += group.device_count
        return total_device_num


class RankTableTemplate1(RankTable):
    def __init__(self, rank_table_template1_path):
        super().__init__()
        self.rank_table_path = rank_table_template1_path
        self.rank_table = self.read_from_file(file_path=rank_table_template1_path)

    def get_current_instance(self):
        current_server = None
        server_list = self.rank_table['server_list']
        if len(server_list) == 1:
            current_server = server_list[0]
        elif len(server_list) > 1:
            host_ip = ModelArts.get_current_host_ip()
            if host_ip is not None:
                for server in server_list:
                    if server['server_id'] == host_ip:
                        current_server = server
                        break
            else:
                current_server = server_list[0]

        if current_server is None:
            log.error('server is not found')
            return None
        return self.convert_server_to_instance(current_server)

    def get_device_num(self):
        server_list = self.rank_table['server_list']
        device_num = 0
        for server in server_list:
            device_num += len(server['device'])
        return device_num
```

- manager.py

```
import time
import os
import os.path
import signal

from common import RunAscendLog
from fmk import FMK


log = RunAscendLog.get_run_ascend_logger()


class FMKManager:
    # max destroy time: ~20 (15 + 5)
    # ~ 15 (1 + 2 + 4 + 8)
    MAX_TEST_PROC_CNT = 4

    def __init__(self, instance):
        self.instance = instance
        self.fmk = []
        self.fmk_processes = []
        self.get_sigterm = False
        self.max_test_proc_cnt = FMKManager.MAX_TEST_PROC_CNT

    # break the monitor and destroy processes when get terminate signal
    def term_handle(func):
        def receive_term(signum, stack):
            log.info('Received terminate signal %d, try to destroyed all processes' % signum)
            stack.f_locals['self'].get_sigterm = True
```

```
    def handle_func(self, *args, **kwargs):
        origin_handle = signal.getsignal(signal.SIGTERM)
        signal.signal(signal.SIGTERM, receive_term)
        res = func(self, *args, **kwargs)
        signal.signal(signal.SIGTERM, origin_handle)
        return res

    return handle_func

def run(self, rank_size, command):
    for index, device in enumerate(self.instance.devices):
        fmk_instance = FMK(index, device)
        self.fmk.append(fmk_instance)

        self.fmk_processes.append(fmk_instance.run(rank_size, command))

@term_handle
def monitor(self, period=1):
    # busy waiting for all fmk processes exit by zero
    # or there is one process exit by non-zero

    fmk_cnt = len(self.fmk_processes)
    zero_ret_cnt = 0
    while zero_ret_cnt != fmk_cnt:
        zero_ret_cnt = 0
        for index in range(fmk_cnt):
            fmk = self.fmk[index]
            fmk_process = self.fmk_processes[index]
            if fmk_process.poll() is not None:
                if fmk_process.returncode != 0:
                    log.error('proc-rank-%s-device-%s (pid: %d) has exited with non-zero code: %d'
                              % (fmk.rank_id, fmk.device_id, fmk_process.pid, fmk_process.returncode))
                    return fmk_process.returncode

                zero_ret_cnt += 1
        if self.get_sigterm:
            break
        time.sleep(period)

    return 0

def destroy(self, base_period=1):
    log.info('Begin destroy training processes')
    self.send_sigterm_to_fmk_process()
    self.wait_fmk_process_end(base_period)
    log.info('End destroy training processes')

def send_sigterm_to_fmk_process(self):
    # send SIGTERM to fmk processes (and process group)
    for r_index in range(len(self.fmk_processes) - 1, -1, -1):
        fmk = self.fmk[r_index]
        fmk_process = self.fmk_processes[r_index]
        if fmk_process.poll() is not None:
            log.info('proc-rank-%s-device-%s (pid: %d) has exited before receiving the term signal',
                     fmk.rank_id, fmk.device_id, fmk_process.pid)
            del self.fmk_processes[r_index]
            del self.fmk[r_index]

        try:
            os.killpg(fmk_process.pid, signal.SIGTERM)
        except ProcessLookupError:
            pass

def wait_fmk_process_end(self, base_period):
    test_cnt = 0
    period = base_period
    while len(self.fmk_processes) > 0 and test_cnt < self.max_test_proc_cnt:
        for r_index in range(len(self.fmk_processes) - 1, -1, -1):
```

```
                fmk = self.fmk[r_index]
                fmk_process = self.fmk_processes[r_index]
                if fmk_process.poll() is not None:
                    log.info('proc-rank-%s-device-%s (pid: %d) has exited',
                             fmk.rank_id, fmk.device_id, fmk_process.pid)
                    del self.fmk_processes[r_index]
                    del self.fmk[r_index]
            if not self.fmk_processes:
                break

            time.sleep(period)
            period *= 2
            test_cnt += 1

        if len(self.fmk_processes) > 0:
            for r_index in range(len(self.fmk_processes) - 1, -1, -1):
                fmk = self.fmk[r_index]
                fmk_process = self.fmk_processes[r_index]
                if fmk_process.poll() is None:
                    log.warn('proc-rank-%s-device-%s (pid: %d) has not exited within the max waiting time,
'
                             'send kill signal',
                             fmk.rank_id, fmk.device_id, fmk_process.pid)
                    os.killpg(fmk_process.pid, signal.SIGKILL)
```

- fmk.py

```
import os
import subprocess
import pathlib
from contextlib import contextmanager

from common import RunAscendLog
from common import RankTableEnv
from common import ModelArts

log = RunAscendLog.get_run_ascend_logger()


class FMK:

    def __init__(self, index, device):
        self.job_id = ModelArts.get_job_id()
        self.rank_id = device.rank_id
        self.device_id = str(index)

    def gen_env_for_fmk(self, rank_size):
        current_envs = os.environ.copy()

        current_envs['JOB_ID'] = self.job_id

        current_envs['ASCEND_DEVICE_ID'] = self.device_id
        current_envs['DEVICE_ID'] = self.device_id

        current_envs['RANK_ID'] = self.rank_id
        current_envs['RANK_SIZE'] = str(rank_size)

        FMK.set_env_if_not_exist(current_envs, RankTableEnv.HCCL_CONNECT_TIMEOUT, str(1800))

        log_dir = FMK.get_log_dir()
        process_log_path = os.path.join(log_dir, self.job_id, 'ascend', 'process_log', 'rank_' + self.rank_id)
        FMK.set_env_if_not_exist(current_envs, 'ASCEND_PROCESS_LOG_PATH', process_log_path)
        pathlib.Path(current_envs['ASCEND_PROCESS_LOG_PATH']).mkdir(parents=True, exist_ok=True)

        return current_envs

    @contextmanager
    def switch_directory(self, directory):
        owd = os.getcwd()
        try:
            os.chdir(directory)
```

```
        yield directory
    finally:
        os.chdir(owd)

def get_working_dir(self):
    fmk_workspace_prefix = ModelArts.get_parent_working_dir()
    return os.path.join(os.path.normpath(fmk_workspace_prefix), 'device%s' % self.device_id)

@staticmethod
def get_log_dir():
    parent_path = os.getenv(ModelArts.MA_MOUNT_PATH_ENV)
    if parent_path:
        log_path = os.path.join(parent_path, 'log')
        if os.path.exists(log_path):
            return log_path

    return ModelArts.TMP_LOG_DIR

@staticmethod
def set_env_if_not_exist(envs, env_name, env_value):
    if env_name in os.environ:
        log.info('env already exists. env_name: %s, env_value: %s ' % (env_name, env_value))
        return

    envs[env_name] = env_value

def run(self, rank_size, command):
    envs = self.gen_env_for_fmk(rank_size)
    log.info('bootstrap proc-rank-%s-device-%s' % (self.rank_id, self.device_id))

    log_dir = FMK.get_log_dir()
    if not os.path.exists(log_dir):
        os.makedirs(log_dir)

    log_file = '%s-proc-rank-%s-device-%s.txt' % (self.job_id, self.rank_id, self.device_id)
    log_file_path = os.path.join(log_dir, log_file)

    working_dir = self.get_working_dir()
    if not os.path.exists(working_dir):
        os.makedirs(working_dir)

    with self.switch_directory(working_dir):
        # os.setsid: change the process(forked) group id to itself
        training_proc = subprocess.Popen(command, env=envs, preexec_fn=os.setsid,
                                         stdout=subprocess.PIPE, stderr=subprocess.STDOUT)

        log.info('proc-rank-%s-device-%s (pid: %d)', self.rank_id, self.device_id, training_proc.pid)

        # https://docs.python.org/3/library/subprocess.html#subprocess.Popen.wait
        subprocess.Popen(['tee', log_file_path], stdin=training_proc.stdout)

        return training_proc
```

11.4.2.1.4 Step3 制作自定义镜像

此处介绍如何通过编写 Dockerfile 文件制作自定义镜像的操作步骤 。

目标：构建安装好如下软件的容器镜像，并使用 ModelArts 训练服务运行。

- ubuntu-18.04
- cann-6.3.RC2 (商用版本)
- python-3.7.13
- mindspore-2.1.1

说明

Mindspore 版本与 CANN 版本，CANN 版本和Ascend驱动/固件版本均有严格的匹配关系，版本不匹配会导致训练失败。

本示例仅用于示意Ascend容器镜像制作流程，且在匹配正确的Ascend驱动/固件版本的专属资源池上运行通过。

1. 准备一台 Linux **aarch64** 架构的主机，操作系统使用ubuntu-18.04。您可以准备相同规格的 弹性云服务器ECS 或者应用本地已有的主机进行自定义镜像的制作。

   购买ECS服务器的具体操作请参考**购买并登录弹性云服务器**。镜像选择公共镜像，推荐使用ubuntu18.04的镜像。

2. 安装 Docker。

   以 Linux **aarch64** 架构的操作系统为例，获取 Docker 安装包。您可以使用以下指令安装 Docker。关于安装 Docker 的更多指导内容参见 **Docker 官方文档**。

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   如果 **docker images** 命令可以执行成功，表示 Docker 已安装，此步骤可跳过。

   启动docker。

   ```
   systemctl start docker
   ```

3. 确认 Docker Engine 版本。执行如下命令。

   ```
   docker version | grep -A 1 Engine
   ```

   命令回显如下。

   ```
    Engine:
     Version:          18.09.0
   ```

   说明

   推荐使用大于等于该版本的 Docker Engine 来制作自定义镜像。

4. 准备名为 context 的文件夹。

   ```
   mkdir -p context
   ```

5. 准备可用的 pip 源文件 pip.conf 。

   ```
   [global]
   index-url = https://repo.xxx.com/repository/pypi/simple
   trusted-host = repo.xxx.com
   timeout = 120
   ```

6. 准备可用的 apt 源文件 Ubuntu-Ports-bionic.list。

   ```
   wget -O Ubuntu-Ports-bionic.list https://repo.xxx.com/repository/conf/Ubuntu-Ports-bionic.list
   ```

7. 下载 CANN 6.3.RC2-linux aarch64 与 mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl 安装文件。

   - 下载run文件“Ascend-cann-nnae_6.3.RC2_linux-aarch64.run”（**下载链接**）。
   - 下载whl 文件“mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl”（**下载链接**）。

   说明

   ModelArts当前仅支持CANN商用版本，不支持社区版。

8. 下载 Miniconda3 安装文件。

   使用地址 https://repo.anaconda.com/miniconda/Miniconda3-py37_4.10.3-Linux-aarch64.sh，下载 Miniconda3-py37-4.10.3 安装文件（对应 python 3.7.10）。

9. 将上述 pip 源文件、*.run 文件、 *.whl 文件、Miniconda3 安装文件放置在 context 文件夹内，context 文件夹内容如下。

```
context
├── Ascend-cann-nnae_6.3.RC2_linux-aarch64.run
├── mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl
├── Miniconda3-py37_4.10.3-Linux-aarch64.sh
├── pip.conf
└── Ubuntu-Ports-bionic.list
```

10. 编写容器镜像 Dockerfile 文件。

    在 context 文件夹内新建名为 Dockerfile 的空文件，并将下述内容写入其中。

```
# 容器镜像构建主机需要连通公网
FROM arm64v8/ubuntu:18.04 AS builder

# 基础容器镜像的默认用户已经是 root
# USER root

# 安装 OS 依赖（使用开源镜像站）
COPY Ubuntu-Ports-bionic.list /tmp
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    mv /tmp/Ubuntu-Ports-bionic.list /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y \
    # utils
    ca-certificates vim curl \
    # CANN 6.3.RC2
    gcc-7 g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils
net-tools libblas-dev gfortran libblas3 && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    # 修改 CANN 6.3.RC2 安装目录的父目录权限，使得 ma-user 可以写入
    chmod o+w /usr/local

RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# 设置容器镜像默认用户与工作目录
USER ma-user
WORKDIR /home/ma-user

# 使用开源镜像站提供的 pypi 配置
RUN mkdir -p /home/ma-user/.pip/
COPY --chown=ma-user:100 pip.conf /home/ma-user/.pip/pip.conf

# 复制待安装文件到基础容器镜像中的 /tmp 目录
COPY --chown=ma-user:100 Miniconda3-py37_4.10.3-Linux-aarch64.sh /tmp

# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
# 安装 Miniconda3 到基础容器镜像的 /home/ma-user/miniconda3 目录中
RUN bash /tmp/Miniconda3-py37_4.10.3-Linux-aarch64.sh -b -p /home/ma-user/miniconda3

ENV PATH=$PATH:/home/ma-user/miniconda3/bin

# 安装 CANN 6.3.RC2 Python Package 依赖
RUN pip install numpy~=1.14.3 decorator~=4.4.0 sympy~=1.4 cffi~=1.12.3 protobuf~=3.11.3 \
    attrs pyyaml pathlib2 scipy requests psutil absl-py

# 安装 CANN 6.3.RC2 至 /usr/local/Ascend 目录
COPY --chown=ma-user:100 Ascend-cann-nnae_6.3.RC2_linux-aarch64.run /tmp
RUN chmod +x /tmp/Ascend-cann-nnae_6.3.RC2_linux-aarch64.run && \
    /tmp/Ascend-cann-nnae_6.3.RC2_linux-aarch64.run --install --install-path=/usr/local/Ascend

# 安装 MindSpore 2.1.1
COPY --chown=ma-user:100 mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl /tmp
RUN chmod +x /tmp/mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl && \
    pip install /tmp/mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl

# 构建最终容器镜像
FROM arm64v8/ubuntu:18.04

# 安装 OS 依赖（使用开源镜像站）
```

```
COPY Ubuntu-Ports-bionic.list /tmp
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
    mv /tmp/Ubuntu-Ports-bionic.list /etc/apt/sources.list && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y \
    # utils
    ca-certificates vim curl \
    # CANN 6.3.RC2
    gcc-7 g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils
net-tools libblas-dev gfortran libblas3 && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list

RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# 从上述 builder stage 中复制目录到当前容器镜像的同名目录
COPY --chown=ma-user:100 --from=builder /home/ma-user/miniconda3 /home/ma-user/miniconda3
COPY --chown=ma-user:100 --from=builder /home/ma-user/Ascend /home/ma-user/Ascend
COPY --chown=ma-user:100 --from=builder /home/ma-user/var /home/ma-user/var
COPY --chown=ma-user:100 --from=builder /usr/local/Ascend /usr/local/Ascend

# 设置容器镜像预置环境变量
# 请务必设置 CANN 相关环境变量
# 请务必设置 Ascend Driver 相关环境变量
# 请务必设置 PYTHONUNBUFFERED=1, 以免日志丢失
ENV PATH=$PATH:/usr/local/Ascend/nnae/latest/bin:/usr/local/Ascend/nnae/latest/compiler/
ccec_compiler/bin:/home/ma-user/miniconda3/bin \
    LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/
lib64/common:/usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/nnae/latest/lib64:/usr/local/
Ascend/nnae/latest/lib64/plugin/opskernel:/usr/local/Ascend/nnae/latest/lib64/plugin/nnengine \
    PYTHONPATH=$PYTHONPATH:/usr/local/Ascend/nnae/latest/python/site-packages:/usr/local/
Ascend/nnae/latest/opp/built-in/op_impl/ai_core/tbe \
    ASCEND_AICPU_PATH=/usr/local/Ascend/nnae/latest \
    ASCEND_OPP_PATH=/usr/local/Ascend/nnae/latest/opp \
    ASCEND_HOME_PATH=/usr/local/Ascend/nnae/latest \
    PYTHONUNBUFFERED=1

# 设置容器镜像默认用户与工作目录
USER ma-user
WORKDIR /home/ma-user
```

关于 Dockerfile 文件编写的更多指导内容参见 **Docker 官方文档**。

11. 确认已创建完成 Dockerfile 文件。此时 context 文件夹内容如下。

```
context
├── Ascend-cann-nnae_6.3.RC2_linux-aarch64.run
├── Dockerfile
├── mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl
├── Miniconda3-py37_4.10.3-Linux-aarch64.sh
├── pip.conf
└── Ubuntu-Ports-bionic.list
```

12. 构建容器镜像。在 Dockerfile 文件所在的目录执行如下命令构建容器镜像 。

```
docker build . -t mindspore:2.1.1-cann6.3.RC2
```

构建过程结束时出现如下构建日志说明镜像构建成功。

```
Successfully tagged mindspore:2.1.1-cann6.3.RC2
```

13. 将制作完成的镜像上传至SWR服务，具体参见**Step4 上传镜像至SWR**。

11.4.2.1.5 Step4 上传镜像至 SWR

本章节介绍如何将制作好的镜像上传至SWR服务，方便后续在ModelArts上创建训练作业时调用。

1. 登录容器镜像服务控制台，选择区域。

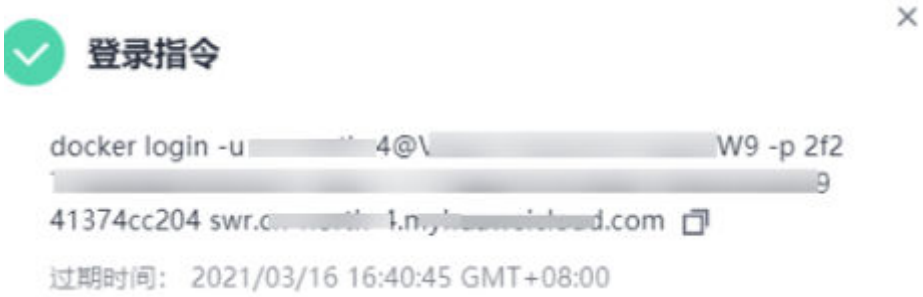
图 11-15 容器镜像服务控制台

2. 单击右上角“创建组织”，输入组织名称完成组织创建。请自定义组织名称，本示例使用“deep-learning”，下面的命令中涉及到组织名称“deep-learning”也请替换为自定义的值。

图 11-16 创建组织

3. 单击右上角“登录指令”，获取登录访问指令。

图 11-17 登录指令



4. 以root用户登录本地环境，输入登录访问指令。
5. 上传镜像至容器镜像服务镜像仓库。
   a. 使用docker tag命令给上传镜像打标签。
   ```
   #region和domain信息请替换为实际值，组织名称deep-learning也请替换为自定义的值。
   sudo docker tag mindspore:2.1.1-cann6.3.RC2 swr.{region}.{domain}/deep-learning/mindspore:2.1.1-cann6.3.RC2
   ```
   b. 使用docker push命令上传镜像。
   ```
   #region和domain信息请替换为实际值，组织名称deep-learning也请替换为自定义的值。
   sudo docker push swr.{region}.{domain}/deep-learning/mindspore:2.1.1-cann6.3.RC2
   ```

6. 完成镜像上传后，在“容器镜像服务控制台>我的镜像”页面可查看已上传的自定义镜像。

   说明

   此处的region和domain信息可以通过控制台查看或者联系局点的系统管理员获取。

   例如以下示例中，region为cn-southwest-228，domain为cdzs.cn，则镜像地址为：“swr.cn-southwest-228.cdzs.cn/deep-learning/mindspore:2.1.1-cann6.3.RC2”，请以局点实际信息为准。

11.4.2.1.6 Step5 在 ModelArts 上创建 Notebook 并调试

1. 将**上传到SWR上的镜像**注册到ModelArts的镜像管理中。

   登录ModelArts管理控制台，在左侧导航栏中选择“镜像管理 ”，单击“注册镜像”，根据界面提示注册镜像。注册后的镜像可以用于创建Notebook。

2. 在Notebook中使用自定义镜像创建Notebook并调试，调试成功后，保存镜像。

   a. 在Notebook中使用自定义镜像创建Notebook操作请参见**基于自定义镜像创建Notebook实例**。

   b. 保存Notebook镜像操作请参见**保存Notebook镜像环境**。

3. 已有的镜像调试成功后，再**使用ModelArts训练模块训练作业**。

11.4.2.1.7 Step6 在 ModelArts 上创建训练作业

1. 登录ModelArts管理控制台，在左侧导航栏中选择“训练管理 > 训练作业 New”，默认进入“训练作业”列表。
2. 在“创建训练作业”页面，填写相关参数信息，然后单击“提交”。
   - 创建方式：选择“自定义算法”
   - 启动方式：选择“自定义”
   - 镜像地址：“swr.xxx.xxx.com/deep-learning/mindspore:2.1.1-cann6.3.RC2”
   - 代码目录：设置为OBS中存放启动脚本文件的目录，例如：“obs://test-modelarts/ascend/demo-code/”
   - 启动命令：“python ${MA_JOB_DIR}/demo-code/run_ascend/run_ascend.py python ${MA_JOB_DIR}/demo-code/mindspore-verification.py”
   - 资源池：选择专属资源池
   - 类型：选择驱动/固件版本匹配的专属资源池 Ascend 规格。
   - 作业日志路径：设置为OBS中存放训练日志的路径。例如：“obs://test-modelarts/ascend/log/”
3. 在“规格确认”页面，确认训练作业的参数信息，确认无误后单击“提交”。
4. 训练作业创建完成后，后台将自动完成容器镜像下载、代码目录下载、执行启动命令等动作。

   训练作业一般需要运行一段时间，根据您的训练业务逻辑和选择的资源不同，训练时长将持续几十分钟到几小时不等。训练作业执行成功后，日志信息如**图11-18**所示。

图 11-18 专属资源池 Ascend 规格运行日志信息

```
Ascend Envs
------
JOB_ID:  modelarts-job-2436291a-8543-4ab8-84ad-2dda8f1e4f5c
RANK_TABLE_FILE:  /home/ma-user/modelarts/rank_table/jobstart_hccl.json
RANK_SIZE:  1
ASCEND_DEVICE_ID:  0
DEVICE_ID:  0
RANK_ID:  0
------
   [2. 2. 2. 2.]
   [2. 2. 2. 2.]]

  [[2. 2. 2. 2.]
   [2. 2. 2. 2.]
   [2. 2. 2. 2.]]

  [[2. 2. 2. 2.]
   [2. 2. 2. 2.]
   [2. 2. 2. 2.]]]
```

11.4.3 准备训练镜像

11.4.3.1 训练作业自定义镜像规范

针对您本地开发的模型及训练脚本，在制作镜像时，需满足ModelArts定义的规范。

规范要求

- 推荐自定义镜像使用ubuntu-18.04的操作系统，避免出现版本不兼容的问题。
- 自定义镜像的大小推荐15GB以内，最大不要超过资源池的容器引擎空间大小的一半。镜像过大会直接影响训练作业的启动时间。

  ModelArts公共资源池的容器引擎空间为50G，专属资源池的容器引擎空间的默认为50G，支持在创建专属资源池时自定义容器引擎空间。
- 自定义镜像的默认用户必须为“uid”为“1000”的用户。
- 自定义镜像中不能安装GPU或Ascend驱动程序。当用户选择GPU资源运行训练作业时，ModelArts后台自动将GPU驱动程序放置在训练环境中的 /usr/local/nvidia 目录；当用户选择Ascend资源运行训练作业时，ModelArts后台自动将Ascend驱动程序放置在 /usr/local/Ascend/driver 目录。
- X86 CPU架构，ARM CPU架构的自定义镜像分别只能运行于对应 CPU 架构的规格中。
  - 执行如下命令查看自定义镜像的 CPU 架构

    ```
    docker inspect {自定义镜像地址} | grep Architecture
    ```

    ARM CPU 架构的自定义镜像，上述命令回显示意如下

    ```
    "Architecture": "arm64"
    ```
  - 规格中带有 ARM 字样的显示，为 ARM CPU 架构。

  - 规格中未带有 ARM 字样的显示，为 X86 CPU 架构。

- ModelArts后台暂不支持下载开源安装包，建议用户在自定义镜像中安装训练所需的依赖包。

11.4.3.2 已有镜像如何适配迁移至 ModelArts 训练平台

已有镜像迁移至 训练管理需要关注如下步骤。

1. 为镜像增加训练管理的默认用户组ma-group，“gid = 100”。

   说明

   如果已存在“gid = 100”用户组，可能会报错“groupadd: GID '100' already exists”。可通过命令“cat /etc/group | grep 100”查询是否已存在 gid = 100 用户组。

   如果已存在“gid = 100”用户组，则该步骤跳过，下文Dockerfile中删除“RUN groupadd ma-group -g 100”命令。

2. 为镜像增加训练管理的默认用户ma-user，“uid = 1000”。

   说明

   如果已存在“uid = 1000”用户，可能会报错“useradd: UID 1000 is not unique”。可通过命令“cat /etc/passwd | grep 1000”查询是否已存在 uid = 1000 用户。

   如果已存在“uid = 1000”用户，则该步骤跳过，下文Dockerfile中删除“RUN useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user”命令。

3. 修改镜像中相关文件权限，使得ma-user，“uid = 1000”用户可读写。

您可以参考如下Dockerfile，修改已有镜像，使其符合新版训练管理自定义镜像的规范。

```
FROM {已有镜像}

USER root

# 如果已存在 gid = 100 用户组，则删除 groupadd 命令。
RUN groupadd ma-group -g 100
# 如果已存在 uid = 1000 用户，则删除 useradd 命令。
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# 修改镜像中相关文件权限，使得 ma-user, uid = 1000 用户可读写。
RUN chown -R ma-user:100 {Python软件包路径}

# 设置容器镜像预置环境变量。
# 请务必设置 PYTHONUNBUFFERED=1, 以免日志丢失。
ENV PYTHONUNBUFFERED=1

# 设置容器镜像默认用户与工作目录。
USER ma-user
WORKDIR /home/ma-user
```

编写好Dockerfile后，通过执行如下所示命令进行新镜像构建。

```
docker build -f Dockerfile . -t {新镜像}
```

构建成功后将新镜像上传至SWR（参考**如何登录并上传镜像到SWR**）。

11.4.3.3 使用基础镜像构建新的训练镜像

ModelArts平台提供了Tensorflow，Pytorch，MindSpore等常用深度学习任务的基础镜像，镜像里已经安装好运行任务所需软件。当基础镜像里的软件无法满足您的程序运行需求时，您可以基于这些基础镜像制作一个新的镜像并进行训练。

基于训练基础镜像构建新镜像的操作步骤

您可以参考如下步骤基于训练基础镜像来构建新镜像。

1. 安装Docker。如果**docker images**命令可以执行成功，表示Docker已安装，此步骤可跳过。

   以linux x86_64架构的操作系统为例，获取Docker安装包。您可以使用以下指令安装Docker。

   ```
   curl -fsSL get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

2. 准备名为 context 的文件夹。

   ```
   mkdir -p context
   ```

3. 准备可用的 pip 源文件 pip.conf 。

   ```
   [global]
   index-url = https://repo.xxx.com/repository/pypi/simple
   trusted-host = repo.xxx.com
   timeout = 120
   ```

4. 参考如下Dockerfile文件内容来基于ModelArts提供的训练基础镜像来构建一个新镜像。将编写好的 Dockerfile 文件放置在 context 文件夹内。

   ```
   FROM {ModelArts提供的训练基础镜像地址}

   # 配置pip
   RUN mkdir -p /home/ma-user/.pip/
   COPY --chown=ma-user:ma-group pip.conf /home/ma-user/.pip/pip.conf

   # 设置容器镜像预置环境变量
   # 将python解释器路径加入到PATH环境变量中
   # 请务必设置PYTHONUNBUFFERED=1, 以免日志丢失
   ENV PATH=${ANACONDA_DIR}/envs/${ENV_NAME}/bin:$PATH \
       PYTHONUNBUFFERED=1

   RUN /home/ma-user/anaconda/bin/pip install --no-cache-dir numpy
   ```

5. 构建新镜像。在Dockerfile文件所在的目录执行如下命令构建容器镜像training:v1。

   ```
   docker build . -t training:v1
   ```

6. 将构建好的新镜像上传至SWR（参考**如何登录并上传镜像到SWR**）。
7. 参考**使用自定义镜像创建训练作业（CPU/GPU）**章节在ModelArts上使用。

11.4.4 使用自定义镜像创建算法

针对您在本地或使用其他工具开发的算法，支持上传至ModelArts中统一管理。

创建算法入口

在ModelArts上基于自定义镜像创建算法有2个入口：

- 入口1：在ModelArts控制台“算法管理 >我的算法”入口，此处创建的算法可以在创建训练作业时直接使用，并且可以发布算法到AI Hub。
- 入口2：在ModelArts控制台 “训练管理 > 训练作业 > 创建训练作业”时，直接创建自定义算法，并提交训练作业。具体参见**使用自定义镜像创建训练作业（CPU/GPU）**。

创建算法参数设置

表 11-18 创建算法参数说明

参数	说明
启动方式	必选，选择“自定义”。
镜像	必选。容器镜像地址。 • 自有镜像或他人共享的镜像：单击右边的“选择”，可以从SWR服务选择用户的容器镜像，前提是要先上传镜像到SWR中，操作指导可参见**如何登录并上传镜像到SWR**。 • 公开镜像：支持手动输入SWR上的公开镜像地址（<用户镜像所属组织>/<镜像名称>），地址上不需要带域名信息（swr.<region>.xxx.com），系统会自动拼接域名地址。例如： modelarts-job-dev-image/pytorch_1_8:train-pytorch_1.8.0-cuda_10.2-py_3.7-euleros_2.10.1-x86_64-8.1.1
代码目录	可选，训练代码存储的OBS路径。 以选择了OBS路径“obs://obs-bucket/training-test/demo-code”作为代码目录为例，OBS路径下的内容会被自动下载至训练容器的“${MA_JOB_DIR}/demo-code”目录中，demo-code为OBS存放代码路径的最后一级目录，用户可以根据实际修改。
启动命令	必选，镜像的启动命令。在代码目录下载完成后，启动命令会被自动执行。 • 如果训练启动脚本用的是py文件，例如train.py，启动命令可以写为**python ${MA_JOB_DIR}/demo-code/train.py**。 • 如果训练启动脚本用的是sh文件，例如main.sh，启动命令可以写为**bash ${MA_JOB_DIR}/demo-code/main.sh**。 启动命令可支持使用“;”和“&&”拼接多条命令，但暂不支持换行拼接。命令中的demo-code为OBS存放代码路径的最后一级目录，用户可以根据实际修改。

输入输出管道设置

训练过程中，基于预置框架的算法需要从OBS桶或者数据集中获取数据进行模型训练，训练产生的输出结果也需要存储至OBS桶中。用户的算法代码中需解析输入输出参数实现ModelArts后台与OBS的数据交互，用户可以参考**开发自定义脚本**完成适配ModelArts训练的代码开发。

创建基于预置框架的算法时，用户需要配置算法代码中定义的输入输出参数。

- 输入配置

表 11-19 输入配置

参数	参数说明
参数名称	根据实际代码中的输入数据参数定义此处的名称。此处设置的代码路径参数必须与算法代码中解析的训练输入数据参数保持一致，否则您的算法代码无法获取正确的输入数据。 例如，算法代码中使用argparse解析的data_url作为输入数据的参数，那么创建算法时就需要配置输入数据的参数名称为“data_url”。
描述	输入参数的说明，用户可以自定义描述。
获取方式	输入参数的获取方式，默认使用“超参”，也可以选择“环境变量”。
输入约束	开启后，用户可以根据实际情况限制数据输入来源。输入来源可以选择“数据存储位置”或者“ModelArts数据集”。 如果用户选择数据来源为ModelArts数据集，还可以约束以下三种： • 标注类型。数据类型请参考**标注数据**。 • 数据格式。可选“Default”和“CarbonData”，支持多选。其中“Default”代表Manifest格式。 • 数据切分。仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。可选“仅支持切分的数据集”、“仅支持未切分数据集”和“无限制”。数据切分详细内容可参考**发布数据版本**。
添加	用户可以根据实际算法添加多个输入数据来源。

- 输出配置

表 11-20 输出配置

参数	参数说明
参数名称	根据实际代码中的训练输出参数定义此处的名称。此处设置的代码路径参数必须与算法代码中解析的训练输出参数保持一致，否则您的算法代码无法获取正确的输出路径。 例如，算法代码中使用argparse解析的train_url作为训练输出数据的参数，那么创建算法时就需要配置输出数据的参数名称为“train_url”。
描述	输出参数的说明，用户可以自定义描述。
获取方式	输出参数的获取方式，默认使用“超参”，也可以选择“环境变量”。
添加	用户可以根据实际算法添加多个输出数据路径。

定义超参

使用预置框架创建算法时，ModelArts支持用户自定义超参，方便用户查阅或修改。定义超参后会体现在启动命令中，以命令行参数的形式传入您的启动文件中。

1. 导入超参

   您可以单击“增加超参”手动添加超参。

2. 编辑超参

   超参的参数说明参见**表11-21**。

表 11-21 超参编辑参数

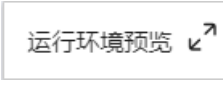
参数	说明
名称	填入超参名称。 超参名称支持64个以内字符，仅支持大小写字母、数字、下划线和中划线。
类型	填入超参的数据类型。支持String、Integer、Float和Boolean。
默认值	填入超参的默认值。创建训练作业时，默认使用该值进行训练。
约束	单击“约束”。在弹出对话框中，支持用户设置默认值的取值范围或者枚举值范围。
必需	选择是或否。 ● 选择否，则在使用该算法创建训练作业时，支持在创建训练作业页面删除该超参。 ● 选择是，则在使用该算法创建训练作业时，不支持在创建训练作业页面删除该超参。
描述	填入超参的描述说明。 超参描述支持大小写字母、数字、空格、中划线、下划线、逗号和句号。

添加训练约束

用户可以根据实际情况定义此算法的训练约束。

- 资源类型：选择适用的资源类型，支持多选。
- 多卡训练：选择是否支持多卡训练。
- 分布式训练：选择是否支持分布式训练。

运行环境预览

创建算法时，可以打开创建页面右下方的运行环境预览窗口 

，辅助您了解代码目录、启动文件、输入输出等数据配置在训练容器中的路径。

后续操作

创建算法完成后，可以使用算法快速创建训练作业，详细操作请参见**使用自定义镜像创建训练作业（CPU/GPU）**。

11.4.5 使用自定义镜像创建训练作业（CPU/GPU）

模型训练是一个不断迭代和优化的过程。在训练模块的统一管理下，方便用户试验算法、数据和超参数的各种组合，便于追踪最佳的模型与输入配置，您可以通过不同版本间的评估指标比较，确定最佳训练作业。

前提条件

- 已将用于训练的数据上传至OBS目录。
- 已在OBS创建至少1个空的文件夹，用于存储训练输出的内容。
- 已按照ModelArts规范制作自定义镜像包，自定义镜像包规范请参见**训练作业自定义镜像规范**。
- 已将自定义镜像上传至SWR服务，操作指导可参见**如何登录并上传镜像到SWR**。

创建训练作业

1. 登录ModelArts管理控制台，在左侧导航栏中选择“训练管理 > 训练作业”，进入“训练作业”列表。
2. 单击“创建训练作业”，进入创建训练作业页面，填写训练作业相关参数。

表 11-22 创建训练作业的创建方式（使用自定义镜像）

参数名称	说明
创建方式	必选，选择“自定义算法”。
启动方式	必选，选择“自定义”。
镜像	必填，填写容器镜像的地址。 容器镜像地址的填写支持如下方式。 • 选择自有镜像或他人共享的镜像：单击右边的“选择”，从容器镜像中选择用于训练的容器镜像。所需镜像需要提前上传到SWR服务中。 • 选择公开镜像：直接输入SWR服务中公开镜像的地址。地址直接填写“组织名称/镜像名称:版本名称”，不需要带域名信息（swr.<region>.xxx.com），系统会自动拼接域名地址。例如：某公开镜像的SWR地址为“swr.<region>.xxx.com/test-image/tensorflow2_1_1:1.1.1”，则此处填写“test-images/tensorflow2_1_1:1.1.1”。

参数名称	说明
代码目录	选择训练代码文件所在的OBS目录。如果自定义镜像中不含训练代码则需要配置该参数，如果自定义镜像中已包含训练代码则不需要配置。 ● 需要提前将代码上传至OBS桶中，目录内文件总大小要小于或等于5GB，文件数要小于或等于1000个，文件深度要小于或等于32。 ● 训练代码文件会在训练作业启动的时候被系统自动下载到训练容器的“${MA_JOB_DIR}/demo-code”目录中，“demo-code”为存放代码目录的最后一级OBS目录。例如，“代码目录”选择的是“/test/code”，则训练代码文件会被下载到训练容器的“${MA_JOB_DIR}/code”目录中。
运行用户ID	容器运行时的用户ID，该参数为选填参数，建议使用默认值1000。 如果需要指定uid，则uid数值需要在规定范围内，不同资源池的uid范围如下： ● 公共资源池：1000-65535 ● 专属资源池：0-65535

启动命令	必填，镜像的启动命令。
	运行训练作业时，当“代码目录”下载完成后，“启动命令”会被自动执行。
	● 如果训练启动脚本用的是py文件，例如“train.py”，则启动命令如下所示。
	`python ${MA_JOB_DIR}/demo-code/train.py`
	● 如果训练启动脚本用的是sh文件，例如“main.sh”，则启动命令如下所示。
	`bash ${MA_JOB_DIR}/demo-code/main.sh`
	启动命令支持使用“;”和“&&”拼接多条命令，命令中的“demo-code”为存放代码目录的最后一级OBS目录，以实际情况为准。

本地代码目录	指定训练容器的本地目录，启动训练时系统会将代码目录下载至此目录。 此参数可选，默认本地代码目录为“/home/ma-user/modelarts/user-job-dir”。
工作目录	训练时，系统会自动cd到此目录下执行启动文件。

表 11-23 创建训练作业的训练参数

参数名称	子参数	说明

输入	参数名称	算法代码需要通过“输入”的“参数名称”去读取训练的输入数据。
		建议设置为“data_url”，和训练代码中解析输入数据的参数保持一致。此处可以设置多条训练输入参数。训练输入参数名称不可以重名。
		例如，训练代码中使用argparse解析data_url为输入数据超参，则在此处需要配置输入数据代码参数名称为data_url。
		`import argparse`
		`# 创建解析`
		`parser = argparse.ArgumentParser(description="train mnist", formatter_class=argparse.ArgumentDefaultsHelpFormatter)`
		`# 添加参数`
		`parser.add_argument('--train_url', type=str, help='the path model saved')`
		`parser.add_argument('--data_url', type=str, help='the training data')`
		`# 解析参数`
		`args, unknown = parser.parse_known_args()`

	数据集	单击“数据集”，在ModelArts数据集列表中勾选目标数据集并选择对应的版本。 训练启动时，系统将自动下载输入路径中的数据到训练运行容器。 **说明** ModelArts数据管理模块在重构升级中，对未使用过数据管理的用户不可见。建议新用户将训练数据存放至OBS桶中使用。
	数据存储位置	单击“数据存储位置”，从OBS桶中选择训练输入数据的存储位置。 训练启动时，系统将自动下载输入路径中的数据到训练运行容器。

	获取方式	以参数名称为“data_path”的训练输入为例，说明获取方式的作用。
		● 当参数的“获取方式”为“超参”时，可以参考如下代码来读取数据。
		`import argparse`
		`parser = argparse.ArgumentParser()`
		`parser.add_argument('--data_path')`
		`args, unknown = parser.parse_known_args()`
		`data_path = args.data_path`
		● 当参数的“获取方式”为“环境变量”时，可以参考如下代码来读取数据。
		`import os`
		`data_path = os.getenv("data_path", "")`

输出	参数名称	算法代码需要通过“输出”的“参数名称”去读取训练的输出目录。 建议设置为“train_url”，和训练代码中解析输出数据的参数保持一致。此处也可以设置多条训练输出参数。训练输出参数名称不可以重名。

参数名称	子参数	说明
	数据存储位置	单击“数据存储位置”，从OBS桶中选择训练输出数据的存储位置。训练过程中，系统将自动从训练容器的本地代码目录下同步文件到数据存储位置。 **说明** 数据存储位置仅支持OBS路径。为避免数据存储冲突，建议选择一个空目录用作“数据存储位置”。

	获取方式	以参数名称为“train_url”的训练输出为例，说明获取方式的作用。
		● 当参数的“获取方式”为“超参”时，可以参考如下代码来读取数据。
		`import argparse`
		`parser = argparse.ArgumentParser()`
		`parser.add_argument('--train_url')`
		`args, unknown = parser.parse_known_args()`
		`train_url = args.train_url`
		● 当参数的“获取方式”为“环境变量”时，可以参考如下代码来读取数据。
		`import os`
		`train_url = os.getenv("train_url", "")`

	预下载至本地目录	选择是否将输出目录下的文件预下载至本地目录。 ● 不下载：表示启动训练作业时不会将输出数据的存储位置中的文件下载到训练容器的本地代码目录中。 ● 下载：表示系统会在启动训练作业时自动将输出数据的存储位置中的所有文件下载到训练容器的本地代码目录中。下载时间会随着文件变大而变长，为了防止训练时间过长，请及时清理训练容器的本地代码目录中的无用文件。如果要使用**断点续训练和增量训练**，则必须选择“下载”。
超参	-	可选，超参用于训练调优。
环境变量	-	根据业务需求增加环境变量。训练容器中预置的环境变量请参见**查看训练容器环境变量**。
故障自动重启	-	打开开关后，可以设置故障的重启次数，训练作业失败时会自动重新下发并运行训练作业。在训练作业详情页中，用户可以查看训练作业发生故障后的重启次数。 ● 此参数默认关闭。 ● 打开故障自动重启后，需要设置重启次数，重启次数取值范围1~3次，重启次数设置后无法修改。

3. 选择训练资源的规格。训练参数的可选范围与已有自定义镜像的使用约束保持一致。根据需要选择公共资源池或专属资源池，参数说明请参**创建训练作业**。
4. 单击“提交”，完成训练作业的创建。

   训练作业一般需要运行一段时间。

   要查看训练作业实时情况，您可以前往训练作业列表，单击训练作业的名称，进入训练作业详情页，查看训练作业的基本情况，具体请参考**查看作业详情**。

11.4.6 使用自定义镜像创建训练作业（Ascend）

如果Ascend-Powered-Engine预置镜像无法满足您的需求，您可以构建一个自定义镜像，通过自定义镜像创建训练作业。Ascend自定义镜像训练作业创建流程与CPU/GPU 一致，但是需要额外关注：

- Ascend HCCL RANK_TABLE_FILE 文件说明

  Ascend HCCL RANK_TABLE_FILE 文件提供Ascend分布式训练作业的集群信息，用于Ascend芯片分布式通信，可以被HCCL集合通信库解析。该文件格式有两个版本，分别为模板一、模板二。当前ModelArts提供的是模板二格式。完整的Ascend HCCL RANK_TABLE_FILE 文件内容请参见**昇腾AI处理器资源配置文件。**

  ModelArts训练环境的Ascend HCCL RANK_TABLE_FILE 文件名为jobstart_hccl.json。获取方式参考**表11-24** 。

  - ModelArts训练环境 jobstart_hccl.json文件内容（模板二）示例

    ```
    {
       "group_count": "1",
       "group_list": [{
          "device_count": "1",
          "group_name": "job-trainjob",
          "instance_count": "1",
          "instance_list": [{
             "devices": [{
                "device_id": "4",
                "device_ip": "192.1.10.254"
             }],
             "pod_name": "jobxxxxxxxx-job-trainjob-0",
             "server_id": "192.168.0.25"
          }]
       }],
       "status": "completed"
    }
    ```

    jobstart_hccl.json文件中的status字段的值在训练脚本启动时，并不一定为completed状态。因此需要训练脚本等待status字段的值等于completed之后，再去读取文件的剩余内容。

    如果算法开发者期望使用模板一格式的jobstart_hccl.json文件，可以使用训练脚本，在等待status字段的值等于completed之后，将模板二格式jobstart_hccl.json文件转换为模板一格式的 jobstart_hccl.json 文件。

  - 转换后的jobstart_hccl.json 文件格式（模板一）示例

    ```
    {
       "server_count": "1",
       "server_list": [{
          "device": [{
             "device_id": "4",
             "device_ip": "192.1.10.254",
             "rank_id": "0"
          }],
          "server_id": "192.168.0.25"
       }],
       "status": "completed",
       "version": "1.0"
    }
    ```

    说明

    转换功能的实现，可参考**示例：从 0 到 1 制作自定义镜像并用于开发和训练（MindSpore+Ascend）**中所述的Ascend训练脚本的启动脚本。

- RANK_TABLE_FILE环境变量

表 11-24 RANK_TABLE_FILE 环境变量说明

环境变量	说明
RANK_TABLE_FILE	该环境变量指示Ascend HCCL RANK_TABLE_FILE文件所在目录，值为/user/config。 算法开发者可通过 “${RANK_TABLE_FILE}/jobstart_hccl.json”，路径获取该文件。

11.4.7 自定义镜像训练作业失败定位思路

问题现象

使用自定义镜像训练作业时，训练失败。

定位思路

1. 确定镜像来源
   - 确认该自定义镜像的基础镜像是否来源于ModelArts提供的基础镜像，推荐用户使用ModelArts的基础镜像构建自定义镜像，具体请参见。
   - 如镜像来源于第三方，设法找到自定义镜像的制作者咨询，制作者一般对镜像如何使用更加了解。
2. 确定自定义镜像大小

   自定义镜像的大小推荐15GB以内，最大不要超过资源池的容器引擎空间大小的一半。镜像过大会直接影响训练作业的启动时间。

   ModelArts公共资源池的容器引擎空间为50G，专属资源池的容器引擎空间的默认为50G，支持在创建专属资源池时自定义容器引擎空间。
3. 确定错误类型
   - 提示找不到文件等错误，请参见。
   - 提示找不到包等错误，请参见。
   - Ascend启动脚本和初始化脚本问题。

     确认相关脚本是否来源于官方文档并且是否严格按照官方文档使用。比如确认脚本名称是否正常、脚本路径是否正常。
   - 驱动版本与底层驱动不兼容

     当对自定义镜像的驱动进行升级时，请确定底层驱动是否兼容。当前支持哪种驱动版本，请从中获取。
   - 文件权限不足

     该问题可能为自定义镜像的用户与作业容器的用户不同导致的。请修改dockerfile文件：

     ```
     RUN if id -u ma-user > /dev/null 2>&1 ; \
     then echo 'MA 用户已存在' ; \
     else echo 'MA 用户不存在' && \
     groupadd ma-group -g 1000 && \
     useradd -d /home/ma-user -m -u 1000 -g 1000 -s /bin/bash ma-user ; fi && \
     chmod 770 /home/ma-user && \
     chmod 770 /root && \
     usermod -a -G root ma-user
     ```
   - 其他现象，可以在已有的查找。

建议与总结

用户使用自定义镜像训练作业时，建议按照制作镜像。文档中同时提供了端到端的示例供用户参考。

11.5 使用自定义镜像创建 AI 应用（推理部署）

11.5.1 创建 AI 应用的自定义镜像规范

针对您本地开发的模型，在制作AI应用的自定义镜像时，需满足ModelArts定义的规范。

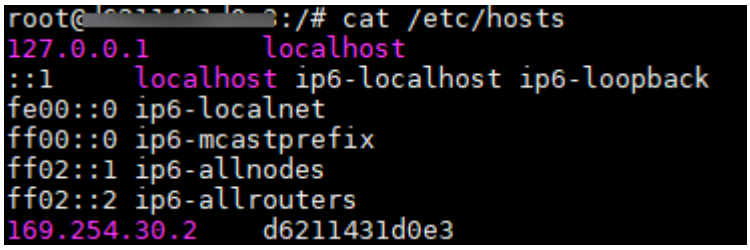
- 自定义镜像中不能包含恶意代码。
- 自定义镜像大小不超过30GB。
- **镜像对外接口**

  设置镜像的对外服务接口，推理接口需与config.json文件中apis定义的url一致，当镜像启动时可以直接访问。下面是mnist镜像的访问示例，该镜像内含mnist数据集训练的模型，可以识别手写数字。其中listen_ip为容器IP，您可以通过启动自定义镜像，在容器中获取容器IP。

  - 请求示例

    ```
    curl -X POST \ http://{listen_ip}:8080/ \ -F images=@seven.jpg
    ```

    图 11-19 listen_ip 获取示例

  - 返回示例

    ```
    {"mnist_result": 7}
    ```

- **（可选）健康检查接口**

  如果在滚动升级时要求不中断业务，那么必须在config.json文件中配置健康检查的接口，供ModelArts调用，在config.json文件中配置。当业务可提供正常服务时，健康检查接口返回健康状态，否则返回异常状态。

  须知

  - 如果要实现无损滚动升级，必须配置健康检查接口。
  - 自定义镜像如果需要在“在线服务”模块使用OBS外部存储挂载功能，需要新建一个OBS挂载专属目录如“/obs-mount/”，避免选择存量目录覆盖已有文件。OBS挂载仅开放对挂载目录文件新增、查看、修改功能不支持删除挂载目录文件对象，若需要删除文件请到OBS并行文件系统中手动删除。

  健康检查接口示例如下。

- URI
  GET /health
- 请求示例**curl -X GET \ http://{listen_ip}:8080/health**
- 响应示例
  {"health": "true"}
- 状态码

  表 11-25 状态码

状态码	编码	状态码说明
200	OK	请求成功

- **日志文件输出**

  为保证日志内容可以正常显示，日志信息需要打印到标准输出。
- **镜像启动入口**

  如果需要部署批量服务，镜像的启动入口文件需要为“/home/run.sh”，采用CMD设置默认启动路径，例如Dockerfile如下：

  CMD ["sh", "/home/run.sh"]
- **镜像依赖组件**

  如果需要部署批量服务，镜像内需要安装python、jre/jdk、zip等组件包。
- **（可选）保持Http长链接，无损滚动升级**

  如果需要支持滚动升级的过程中不中断业务，那么需要将服务的Http的“keep-alive”参数设置为200s。以gunicorn服务框架为例，gunicorn缺省情形下不支持keep-alive，需要同时安装gevent并配置启动参数“--keep-alive 200 -k gevent”。不同服务框架参数设置有区别，请以实际情况为准。
- **（可选）处理SIGTERM信号，容器优雅退出**

  如果需要支持滚动升级的过程中不中断业务，那么需要在容器中捕获SIGTERM信号，并且在收到SIGTERM信号之后等待60秒再优雅退出容器。提前优雅退出容器可能会导致在滚动升级的过程中业务概率中断。要保证容器优雅退出，从收到SIGTERM信号开始，业务需要将收到的请求全部处理完毕再结束，这个处理时长最多不超过90秒。例如run.sh如下所示：

```
#!/bin/bash
gunicorn_pid=""

handle_sigterm() {
  echo "Received SIGTERM, send SIGTERM to $gunicorn_pid"
  if [ $gunicorn_pid != "" ]; then
      sleep 60
      kill -15 $gunicorn_pid  # 传递 SIGTERM 给gunicorn进程
      wait $gunicorn_pid            # 等待gunicorn进程完全终止
  fi
}

trap handle_sigterm TERM
```

11.5.2 从 0-1 制作自定义镜像并创建 AI 应用

针对ModelArts目前不支持的AI引擎，您可以针对该引擎构建自定义镜像，并将镜像导入ModelArts，创建为AI应用。本文详细介绍如何使用自定义镜像完成AI应用的创建，并部署成在线服务。

操作流程如下：

1. **本地构建镜像**：在本地制作自定义镜像包，镜像包规范可参考**创建AI应用的自定义镜像规范**。
2. **本地验证镜像并上传镜像至SWR服务**：验证自定义镜像的API接口功能，无误后将自定义镜像上传至SWR服务。
3. **将自定义镜像创建为AI应用**：将上传至SWR服务的镜像导入ModelArts的AI应用管理。
4. **将AI应用部署为在线服务**：将导入的模型部署上线。

本地构建镜像

以linux x86_x64架构的主机为例，您可以应用本地已有的主机进行自定义镜像的制作。

1. 登录主机后，安装Docker，可参考**Docker官方文档**。也可执行以下命令安装docker。
```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```
2. 获取基础镜像。本示例以Ubuntu18.04为例。
```
docker pull ubuntu:18.04
```
3. 新建文件夹“self-define-images”，在该文件夹下编写自定义镜像的“Dockerfile”文件和应用服务代码“test_app.py”。本样例代码中，应用服务代码采用了flask框架。

   文件结构如下所示
```
self-define-images/
  --Dockerfile
  --test_app.py
```
   - “Dockerfile”
```
From ubuntu:18.04
# 配置源，安装 python、python3-pip 和 Flask
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
  sed -i "s@http://.*security.ubuntu.com@http://repo.xxx.com@g" /etc/apt/sources.list && \
  sed -i "s@http://.*archive.ubuntu.com@http://repo.xxx.com@g" /etc/apt/sources.list && \
  apt-get update && \
  apt-get install -y python3 python3-pip && \
  pip3 install  --trusted-host https://repo.xxx.com -i https://repo.xxx.com/repository/pypi/simple
Flask

# 复制应用服务代码进镜像里面
COPY test_app.py /opt/test_app.py

# 指定镜像的启动命令
CMD python3  /opt/test_app.py
```
   - “test_app.py”
```
from flask import Flask, request
import json
app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
    print("----------- in hello func ----------")
    data = json.loads(request.get_data(as_text=True))
    print(data)
    username = data['name']
    rsp_msg = 'Hello, {}!'.format(username)
    return json.dumps({"response":rsp_msg}, indent=4)

@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
```

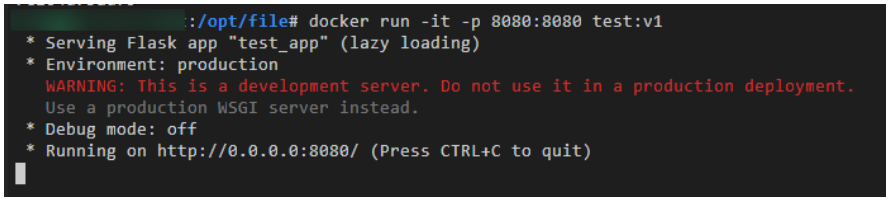
```
    print("---------- in goodbye func ----------")
    return '\nGoodbye!\n'


@app.route('/', methods=['POST'])
def default_func():
    print("---------- in default func ----------")
    data = json.loads(request.get_data(as_text=True))
    return '\n called default func !\n {} \n'.format(str(data))

# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

4. 进入“self-define-images”文件夹，执行以下命令构建自定义镜像“test:v1”。
```
docker build -t test:v1 .
```
5. 您可以使用“docker images”查看您构建的自定义镜像。

本地验证镜像并上传镜像至 SWR 服务

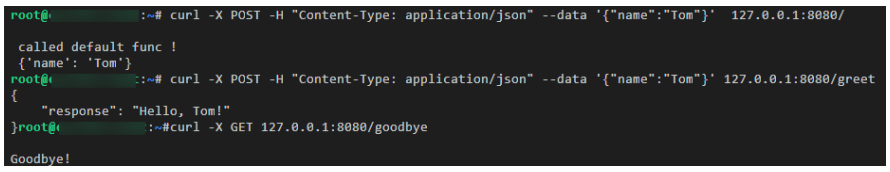
1. 在本地环境执行以下命令启动自定义镜像
```
docker run -it -p 8080:8080 test:v1
```

图 11-20 启动自定义镜像



2. 另开一个终端，执行以下命令验证自定义镜像的三个API接口功能。
```
curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}'  127.0.0.1:8080/
curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/greet
curl -X GET 127.0.0.1:8080/goodbye
```
如果验证自定义镜像功能成功，结果如下图所示。

图 11-21 校验接口



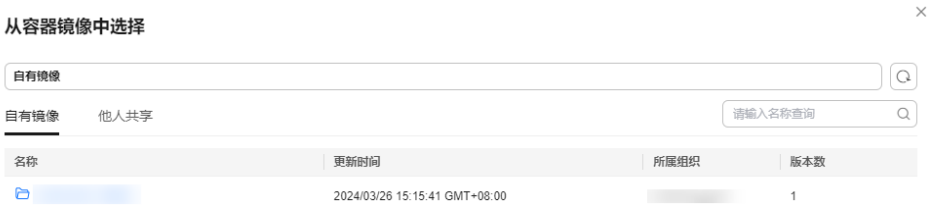
3. 上传自定义镜像至SWR服务。上传镜像的详细操作可参考**如何登录并上传镜像到SWR**。
4. 完成自定义镜像上传后，您可以在“容器镜像服务>我的镜像>自有镜像”列表中看到已上传镜像。

将自定义镜像创建为 AI 应用

参考**从容器镜像中选择元模型**导入元模型，您需要特别关注以下参数：

- 元模型来源：选择“从容器镜像中选择”
  - 容器镜像所在的路径：选择已制作好的自有镜像

图 11-22 选择已制作好的自有镜像



  - 容器调用接口：指定模型启动的协议和端口号。请确保协议和端口号与自定义镜像中提供的协议和端口号保持一致。
  - 镜像复制：选填，选择是否将容器镜像中的模型镜像复制到ModelArts中。
  - 健康检查：选填，用于指定模型的健康检查。仅当自定义镜像中配置了健康检查接口，才能配置“健康检查”，否则会导致AI应用创建失败。
- apis定义：选填，用于编辑自定义镜像的apis定义。模型apis定义需要遵循ModelArts的填写规范，参见**模型配置文件编写说明**。

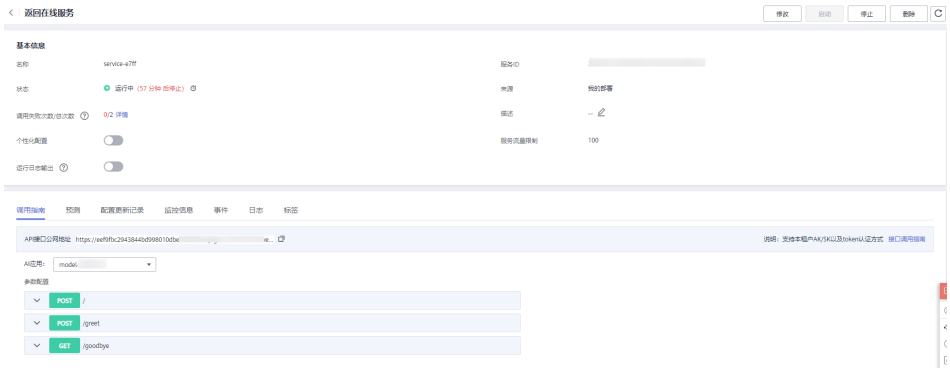
  本样例的配置文件如下所示：

```
[{
    "url": "/",
    "method": "post",
    "request": {
        "Content-type": "application/json"
    },
    "response": {
        "Content-type": "application/json"
    }
  },
{
    "url": "/greet",
    "method": "post",
    "request": {
        "Content-type": "application/json"
    },
    "response": {
        "Content-type": "application/json"
    }
  },
{
    "url": "/goodbye",
    "method": "get",
    "request": {
        "Content-type": "application/json"
    },
    "response": {
        "Content-type": "application/json"
    }
  }
]
```

将 AI 应用部署为在线服务

1. 参考**部署为在线服务**将AI应用部署为在线服务。
2. 在线服务创建成功后，您可以在服务详情页查看服务详情。

图 11-23 调用指南



3. 您可以通过“预测”页签访问在线服务。

图 11-24 访问在线服务

11.6 FAQ

11.6.1 如何登录并上传镜像到 SWR

本章节介绍如何上传镜像到容器镜像服务SWR。

Step1 登录 SWR

1. 登录容器镜像服务控制台，选择区域。
2. 单击右上角“创建组织”，输入组织名称完成组织创建。您可以自定义组织名称，本示例使用“deep-learning”，实际操作时请重新命名一个组织名称。后续所有命令中使用到组织名称deep-learning时，均需要替换为此处实际创建的组织名称。
3. 单击右上角“登录指令”，获取登录访问指令。
4. 以root用户登录ECS环境，输入登录指令。

图 11-25 在 ECS 中执行登录指令

Step2 上传镜像到 SWR

此小节介绍如何上传镜像至容器镜像服务SWR的镜像仓库。

1. 登录SWR后，使用docker tag命令给上传镜像打标签。下面命令中的组织名称deep-learning，请替换为Step1中实际创建的组织名称，以下所有命令中的deep-learning都需要替换。
   ```
   sudo docker tag tf-1.13.2:latest swr.xxx.com/deep-learning/tf-1.13.2:latest
   ```
2. 使用docker push命令上传镜像。
   ```
   sudo docker push swr.xxx.com/deep-learning/tf-1.13.2:latest
   ```

   图 11-26 上传镜像

3. 完成镜像上传后，在“容器镜像服务控制台>我的镜像”页面可查看已上传的自定义镜像。

   “swr.xxx.com/deep-learning/tf-1.13.2:latest”即为此自定义镜像的“SWR_URL”。

11.6.2 如何给镜像设置环境变量

在Dockerfile中，可使用ENV 指令来设置环境变量，具体信息请参考**Dockerfile指导**。

11.6.3 如何通过 docker 启动 Notebook 保存后的镜像

Notebook保存后的镜像有Entrypoint参数，如**图11-27**。Entrypoint参数中指定的可执行文件或命令会覆盖镜像的默认启动命令，Entrypoint中指定的执行命令内容不在镜像中预置，在本地环境通过docker run启动通过Notebook保存的镜像，报错创建容器任务失败，启动文件或目录不存在，如**图11-28**。

因此需要设置--entrypoint参数，覆盖Entrypoint中指定的程序。使用--entrypoint参数指定的启动文件或命令启动镜像。命令示例如下：

```
docker run -it -d --entrypoint /bin/bash image:tag
```

图 11-27 Entrypoint 参数

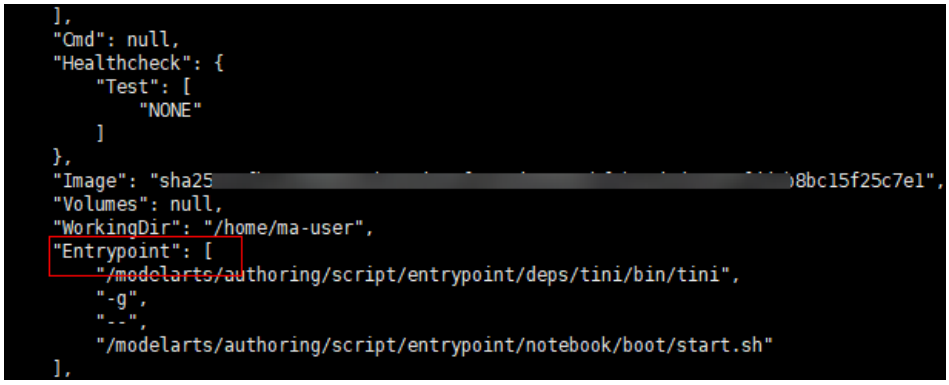
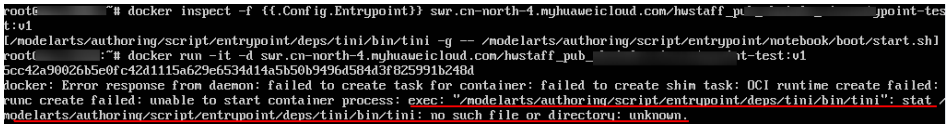


图 11-28 启动镜像报错



11.6.4 如何在 Notebook 开发环境中配置 Conda 源

用户可以在Notebook开发环境中自行安装开发依赖包，方便使用。常见的依赖安装支持pip和Conda，pip源已经配置好，可以直接使用安装，Conda源需要多一步配置。

本章节介绍如何在Notebook开发环境中配置Conda源。

配置 Conda 源

Conda软件已经预置在镜像中，具体操作可以参见 **https://mirror.tuna.tsinghua.edu.cn/help/anaconda/**。

常用 Conda 命令

全部Conda命令建议参考**Conda官方文档**。这里仅对常用命令做简要说明。

表 11-26 常用 Conda 命令

命令说明	命令
获取帮助	conda --help conda update --help #获取某一命令的帮助，如update
查看conda版本	conda -V
更新conda	conda update conda #更新 conda conda update anaconda #更新 anaconda

命令说明	命令
环境管理	conda env list #显示所有的虚拟环境 conda info -e #显示所有的虚拟环境 conda create -n myenv python=3.7 #创建一个名为myenv环境，指定Python版本是3.7 conda activate myenv #激活名为myenv的环境 conda deactivate #关闭当前环境 conda remove -n myenv --all #删除一个名为myenv的环境 conda create -n newname --clone oldname #克隆oldname环境为newname环境
package管理	conda list #查看当前环境下已安装的package conda list -n myenv #指定myenv环境下安装的package conda search numpy #查找名为numpy的package的所有信息 conda search numpy=1.12.0 --info #查看版本为1.12.0的numpy的信息 conda install numpy pandas #安装numpy和pandas两个package，此命令可同时安装一个或多个包 conda install numpy=1.12.0 #安装指定版本的numpy #install，update及remove命令使用-n指定环境，install及update命令使用-c指定源地址 conda install -n myenv numpy #在myenv的环境中安装名字为numpy的package conda install -c https://conda.anaconda.org/anaconda numpy #使用源 https://conda.anaconda.org/anaconda 安装numpy conda update numpy pandas #更新numpy和pandas两个package，此命令可同时更新一个或多个包 conda remove numpy pandas #卸载numpy和pandas两个package，此命令可同时卸载一个或多个包 conda update --all #更新当前环境下所有的package
清理conda	conda clean -p # 删除无用的包 conda clean -t # 删除压缩包 conda clean -y --all # 删除所有的安装包及cache

安装完外部库后保存镜像环境

ModelArts的新版Notebook提供了镜像保存功能。支持一键将运行中的Notebook实例保存为镜像，将准备好的环境保存下来，可以作为自定义镜像，方便后续使用。保存镜像，安装的依赖包不会丢失。安装完依赖包后，推荐保存镜像，避免安装的依赖包丢失。具体操作请参见**保存Notebook镜像环境**。

11.6.5 自定义镜像软件版本匹配注意事项

如果您的自定义镜像涉及NCCL、CUDA、OFED等软件库，当您制作自定义镜像时，您需要确保镜像中的软件库和ModelArts的软件库相匹配。您镜像中的软件版本需要满足以下要求：

- NCCL版本 ≥ 2.7.8。
- OFED版本 ≥ MLNX_OFED_LINUX-5.4-3.1.0.0。
- CUDA版本需要参考专属资源池的GPU驱动版本，自主进行适配，GPU驱动版本可在专属资源池详情页面查看。

12 权限管理

12.1 ModelArts 权限管理基本概念

ModelArts作为一个完备的AI开发平台，支持用户对其进行细粒度的权限配置，以达到精细化资源、权限管理之目的。这类特性在大型企业用户的使用场景下很常见，但对个人用户则显得复杂而意义不足，所以建议个人用户在使用ModelArts时，参照**个人用户快速配置ModelArts访问权限**来进行初始权限设置。

说明

您是否需要阅读本文档?

如果下述问题您的任何一个回答为“是”，则需要阅读此文档

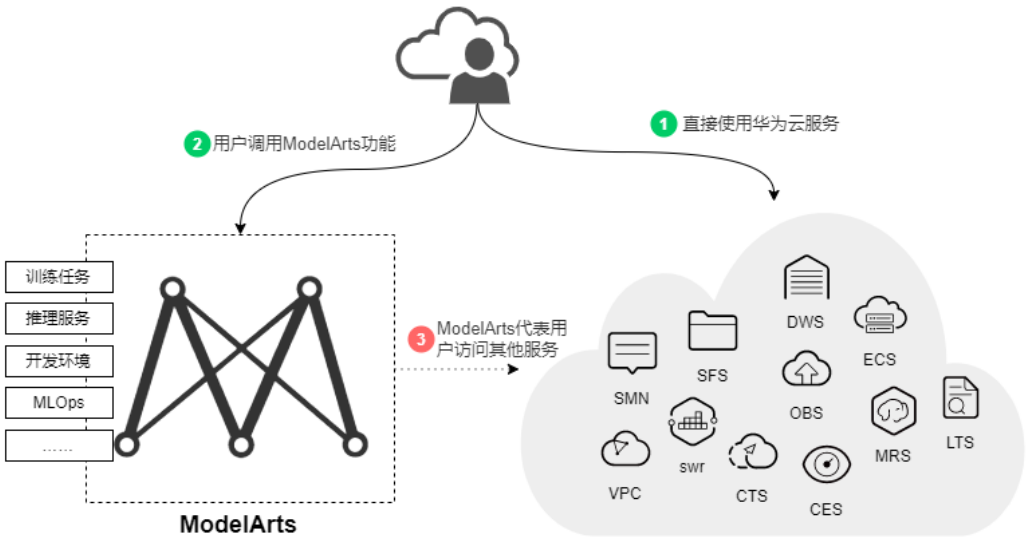
- 您是企业用户，且
  - 存在多个部门，且需要限定不同部门的用户只能访问其专属资源、功能
  - 存在多种角色（如管理员、算法开发者、应用运维），希望限制不同角色只能使用特定功能
  - 逻辑上存在多套“环境”且相互隔离（如开发环境、预生产环境、生产环境），并限定不同用户在不同环境上的操作权限
  - 其他任何需要对特定子用户（组）做出特定权限限制的情况
- 您是个人用户，但已经在IAM创建多个子用户，且期望限定不同子用户所能使用的ModelArts功能、资源不同
- 希望了解ModelArts的权限控制能力细节，期望理解其概念和实操方法

ModelArts的大部分权限管理能力均基于统一身份认证服务（Identity and Access Management，简称IAM）来实现，在您继续往下阅读之前，强烈建议您先行熟悉IAM基本概念，如果能完整理解IAM的所有概念，将更加有助于您理解本文档。

为了支持客户对ModelArts的权限做精细化控制，提供了3个方面的能力来支撑，分别是：权限、委托和工作空间。下面分别讲解。

理解 ModelArts 的权限与委托

图 12-1 权限管理抽象



ModelArts与其他服务类似，对外暴露的每个功能，都通过IAM的权限来进行控制。比如，用户（此处指IAM子用户，而非租户）希望在ModelArts创建训练作业，则该用户必须拥有 "modelarts:trainJob:create" 的权限才可以完成操作（无论界面操作还是API调用）。关于如何给一个用户赋权（准确讲是需要先将用户加入用户组，再面向用户组赋权），可以参考IAM的文档《权限管理》。

而ModelArts还有一个特殊的地方在于，为了完成AI计算的各种操作，AI平台在任务执行过程中需要访问用户的其他服务，典型的就是训练过程中，需要访问OBS读取用户的训练数据。在这个过程中，就出现了ModelArts“代表”用户去访问其他云服务的情形。从安全角度出发，ModelArts代表用户访问任何云服务之前，均需要先获得用户的授权，而这个动作就是一个“委托”的过程。用户授权ModelArts再代表自己访问特定的云服务，以完成其在ModelArts平台上执行的AI计算任务。

综上，对于**图1 权限管理抽象**可以做如下解读：

- 用户访问任何云服务，均是通过标准的IAM权限体系进行访问控制。用户首先需要具备相关云服务的权限（根据您具体使用的功能不同，所需的相关服务权限多寡亦有差异）。
- **权限**：用户使用ModelArts的任何功能，亦需要通过IAM权限体系进行正确权限授权。
- **委托**：ModelArts上的AI计算任务执行过程中需要访问其他云服务，此动作需要获得用户的委托授权。

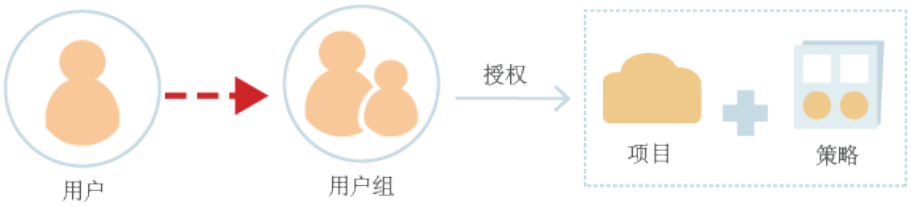
ModelArts 权限管理

默认情况下，管理员创建的IAM用户没有任何权限，需要将其加入用户组，并给用户组授予策略，才能使得用户组中的用户获得对应的权限，这一过程称为授权。授权后，用户就可以基于授予的权限对云服务进行操作。

⚠ 注意

ModelArts部署时通过物理区域划分，为项目级服务，授权时“选择授权范围方案”可以选择“指定区域项目资源”，如果授权时指定了区域对应的项目，则该权限仅对此项目生效；简单的做法是直接选择“所有资源”。

ModelArts也支持企业项目，所以选择授权范围方案时，也可以指定企业项目。



IAM在对用户组授权的时候，并不是直接将具体的某个权限进行赋权，而是需要先将权限加入到“策略”当中，再把策略赋给用户组。为了方便用户的权限管理，各个云服务都提供了一些预置的“系统策略”供用户直接使用。如果预置的策略不能满足您的细粒度权限控制要求，则可以通过“自定义策略”来进行精细控制。

表12-1列出了ModelArts的所有预置系统策略。

表 12-1 ModelArts 系统策略

策略名称	描述	类型
ModelArts FullAccess	ModelArts管理员用户，拥有所有ModelArts服务的权限	系统策略
ModelArts CommonOperations	ModelArts操作用户，拥有所有ModelArts服务操作权限除了管理专属资源池的权限	系统策略
ModelArts Dependency Access	ModelArts服务的常用依赖服务的权限	系统策略

通常来讲，只给管理员开通“ModelArts FullAccess”，如果不需要太精细的控制，直接给所有用户开通“ModelArts CommonOperations”即可满足大多数小团队的开发场景诉求。如果您希望通过自定义策略做深入细致的权限控制，请阅读IAM。

说明

ModelArts的权限不会凌驾于其他服务的权限之上，当您给用户进行ModelArts赋权时，系统不会自动对其他相关服务的相关权限进行赋权。这样做的好处是更加安全，不会出现预期外的“越权”，但缺点是，您必须同时给用户赋予不同服务的权限，才能确保用户可以顺利完成某些ModelArts操作。

举例，如果用户需要用OBS中的数据进行训练，当已经为IAM用户配置ModelArts训练权限时，仍需同时为其配置对应的OBS权限（读、写、列表），才可以正常使用。其中OBS的列表权限用于支持用户从ModelArts界面上选择要进行训练的数据路径；读权限主要用于数据的预览以及训练任务执行时的数据读取；写权限则是为了保存训练结果和日志。

- 对于个人用户或小型组织，一个简单做法是为IAM用户配置“作用范围”为“全局级服务”的“Tenant Administrator”策略，这会使用户获得除了IAM以外的所有用户权限。在获得便利的同时，由于用户的权限较大，会存在相对较大的安全风险，需谨慎使用。（对于个人用户，其默认IAM账号就已经属于admin用户组，且具备Tenant Administrator权限，无需额外操作）
- 当您需要限制用户操作，仅为ModelArts用户配置OBS相关的最小化权限项，对于其他云服务，也可以进行精细化权限控制，具体请参考对应的云服务文档。

ModelArts 委托授权

前文已经介绍，ModelArts在执行AI计算任务过程中，需要“代表”用户去访问其他云服务，而此动作需要提前获得用户的授权。在IAM权限体系下，此类授权动作是通过“委托”来完成。

为了简化用户的委托授权操作，ModelArts增加了自动配置委托授权的支持，用户仅需在ModelArts控制台的“全局配置”页面中，为自己或特定用户配置委托即可。

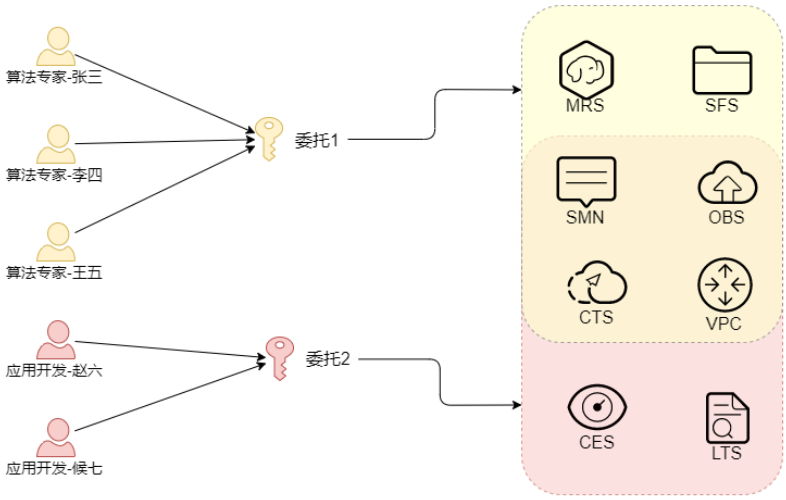
说明

- 只有具备IAM委托管理权限的用户才可以进行此项操作，通常是IAM admin用户组的成员才具备此权限。
- 目前ModelArts的委托授权操作是分区域操作的，这意味着您需要在每个您所用到的区域均执行委托授权操作。

在ModelArts控制台的“全局配置”页面，单击“添加授权”后，系统会引导您为特定用户或所有用户进行委托配置，通常默认会创建一个名为“modelarts_agency_<用户名>_随机ID”的委托条目。在权限配置的区域，您可以选择ModelArts提供的预置配置，也可以自定义选择您所授权的策略。当然如果这两种形态对于您的诉求均过于粗犷，您也可以直接在IAM管理页面里创建完全由您进行精细化配置的委托（需要委托给ModelArts服务），然后在此页面的委托选择里使用“已有委托”“”（而非“新增委托”）。

至此，您应该已经发现了一个细节，ModelArts在使用委托时，是将其与用户进行关联的，用户与委托的关系是多对1的关系。这意味着，如果两个用户需要配置的委托一致，那么不需要为每个用户都创建一个独立的委托项，只需要将两个用户都“指向”同一个委托项即可。

图 12-2 用户与委托对应关系



说明

每个用户必须关联委托才可以使用ModelArts，但即使委托所赋之权限不足，在API调用之初也不会报错，只有到系统具体使用到该功能时，才会发生问题。例如，用户在创建训练任务时打开了“消息通知”，该功能依赖SMN委托授权，但只有训练任务运行过程中，真正需要发送消息时，系统才会“出错”，而有些错误系统会选择“忽略”，另一些错误则可能导致任务直接失败。当您做深入的“权限最小化”限制时，请确保您在ModelArts上将要执行的操作仍旧有足够的权限。

严格授权模式

严格授权模式是指在IAM中创建的子用户必须由账号管理员显式在IAM中授权，才能访问ModelArts服务，管理员用户可以通过授权策略为普通用户精确添加所需使用的ModelArts功能的权限。

相对的，在非严格授权模式下，子用户不需要显式授权就可以使用ModelArts，管理员需要在IAM上为子用户配置Deny策略来禁止子用户使用ModelArts的某些功能。

账号的管理员用户可以在“全局配置”页面修改授权模式。

须知

如无特殊情况，建议优先使用严格授权模式。在严格授权模式下，子用户要使用ModelArts的功能都需经过授权，可以更精确的控制子用户的权限范围，达成权限最小化的安全策略。

用工作空间限制资源访问

工作空间是ModelArts面向企业客户提供的一个高阶功能，用于进一步将用户的资源划分在多个**逻辑隔离**的空间中，并支持以空间维度进行访问的权限限定。目前工作空间功能是“受邀开通”状态，作为企业用户您可以通过您对口的技术支持经理申请开通。

在开通工作空间后，系统会默认为您创建一个“default”空间，您之前所创建的所有资源，均在该空间下。当您创建新的工作空间之后，相当于您拥有了一个新的

“ModelArts分身”，您可以通过菜单栏的左上角进行工作空间的切换，不同工作空间中的工作互不影响。

创建工作空间时，必须绑定一个企业项目。多个工作空间可以绑定到同一个企业项目，但一个工作空间**不可以**绑定多个企业项目。借助工作空间，您可以对不同用户的资源访问和权限做更加细致的约束，具体为如下两种约束：

- 只有被授权的用户才能访问特定的工作空间（在创建、管理工作空间的页面进行配置），这意味着，像数据集、算法等AI资产，均可以借助工作空间做访问的限制。
- 在前文提到的权限授权操作中，如果“选择授权范围方案”时设定为“指定企业项目资源”，那么该授权仅对绑定至该企业项目的工作空间生效。

说明

- 工作空间的约束与权限授权的约束是叠加生效的，意味着对于一个用户，必须同时拥有工作空间的访问权和训练任务的创建权限（且该权限覆盖至当前的工作空间），他才可以在这个空间里提交训练任务。
- 对于已经开通企业项目但没有开通工作空间的用户，其所有操作均相当于在“default”企业项目里进行，请确保对应权限已覆盖了名为default的企业项目。
- 对于未开通企业项目的用户，不受上述约束限制。

本章小结

对于ModelArts的权限管理，总结了如下几条关键点：

- 如果您是个人用户，则不需要考虑细粒度权限问题，您的账户默认具备使用ModelArts的所有权限。
- ModelArts平台的所有功能均通过IAM体系进行了权限管控，您可以通过标准的IAM**授权**动作，来对特定用户进行精细化的权限管控。
- 对于所有用户（包括个人用户），需要完成对ModelArts的**委托**授权（ModelArts > 全局配置 > 添加授权），才能使用特定的功能，否则会造成您的操作出现不可预期的错误。
- 对于开通了企业项目的用户，可以进一步申请开通ModelArts的**工作空间**，通过组合使用基础授权和工作空间，来达成更加复杂的权限控制目的。

12.2 权限控制方式

12.2.1 IAM

介绍ModelArts所有功能涉及到的IAM权限配置。

IAM 权限简介

如果您需要对云服务平台上创建的ModelArts资源，为企业中的员工设置不同的访问权限，以达到不同员工之间的权限隔离，您可以使用统一身份认证服务（Identity and Access Management，简称IAM）进行精细的权限管理。该服务提供用户身份认证、权限分配、访问控制等功能，可以帮助您安全的控制云服务资源的访问。如果已经能满足您的要求，不需要通过IAM对用户进行权限管理，您可以跳过本章节，不影响您使用ModelArts服务的其他功能。

IAM是云服务平台提供权限管理的基础服务，无需付费即可使用，您只需要为您账号中的资源进行付费。

通过IAM，您可以通过授权控制他们对云服务资源的访问范围。例如您的员工中有负责软件开发的人员，您希望他们拥有ModelArts的使用权限，但是不希望他们拥有删除ModelArts等高危操作的权限，那么您可以使用IAM进行权限分配，通过授予用户仅能使用ModelArts，但是不允许删除ModelArts的权限，控制他们对ModelArts资源的使用范围。

关于IAM的详细介绍，请参见**IAM产品简介**。

角色与策略权限管理

ModelArts服务支持角色与策略授权。默认情况下，管理员创建的IAM用户没有任何权限，需要将其加入用户组，并给用户组授予策略或角色，才能使得用户组中的用户获得对应的权限，这一过程称为授权。授权后，用户就可以基于被授予的权限对云服务进行操作。

ModelArts部署时通过物理区域划分，为项目级服务。授权时，“授权范围”需要选择“指定区域项目资源”，然后在指定区域对应的项目中设置相关权限，并且该权限仅对此项目生效；如果“授权范围”选择“所有资源”，则该权限在所有区域项目中都生效。访问ModelArts时，需要先切换至授权区域。

如**表12-2**所示，包括了ModelArts的所有系统策略权限。如果系统预置的ModelArts权限，不满足您的授权要求，可以创建自定义策略，可参考**策略JSON格式字段介绍**。

表 12-2 ModelArts 系统策略

策略名称	描述	类型
ModelArts FullAccess	ModelArts管理员用户，拥有所有ModelArts服务的权限。	系统策略
ModelArts CommonOperations	ModelArts操作用户，拥有所有ModelArts服务操作权限除了管理专属资源池的权限。	系统策略
ModelArts Dependency Access	ModelArts服务的常用依赖服务的权限。	系统策略

ModelArts对其他云服务有依赖关系，因此在ModelArts控制台的各项功能需要配置相应的服务权限后才能正常查看或使用，依赖服务及其预置的权限如下。

表 12-3 ModelArts 控制台依赖服务的角色或策略

控制台功能	依赖服务	需配置角色/策略
数据管理	对象存储服务OBS	OBS Administrator
	数据湖探索DLI	DLI FullAccess
	MapReduce服务MRS	MRS Administrator

控制台功能	依赖服务	需配置角色/策略
	数据仓库服务 GaussDB(DWS)	DWS Administrator
	云审计服务CTS	CTS Administrator
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
开发环境	对象存储服务OBS	OBS Administrator
	凭据管理服务CSMS	CSMS ReadOnlyAccess
	云审计服务CTS	CTS Administrator
	弹性云服务器ECS	ECS FullAccess
	容器镜像服务SWR	SWR Administrator
	弹性文件服务SFS	SFS Turbo FullAccess
	应用运维管理服务 AOM	AOM FullAccess
	密钥管理服务KMS	KMS CMKFullAccess
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
训练管理	对象存储服务OBS	OBS Administrator
	消息通知服务SMN	SMN Administrator
	云审计服务CTS	CTS Administrator
	弹性文件服务SFS Turbo	SFS Turbo ReadOnlyAccess
	容器镜像服务SWR	SWR Administrator
	应用运维管理服务 AOM	AOM FullAccess
	密钥管理服务KMS	KMS CMKFullAccess
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
Workflow	对象存储服务OBS	OBS Administrator
	云审计服务CTS	CTS Administrator
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
自动学习	对象存储服务OBS	OBS Administrator

控制台功能	依赖服务	需配置角色/策略
	云审计服务CTS	CTS Administrator
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
AI应用管理	对象存储服务OBS	OBS Administrator
	企业项目管理服务EPS	EPS FullAccess
	云审计服务CTS	CTS Administrator
	容器镜像服务SWR	SWR Administrator
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
部署上线	对象存储服务OBS	OBS Administrator
	云监控服务CES	CES ReadOnlyAccess
	消息通知服务SMN	SMN Administrator
	企业项目管理服务EPS	EPS FullAccess
	云审计服务CTS	CTS Administrator
	云日志服务LTS	LTS FullAccess
	虚拟私有云VPC	VPC FullAccess
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
AI Gallery	对象存储服务OBS	OBS Administrator
	云审计服务CTS	CTS Administrator
	容器镜像服务SWR	SWR Administrator
	AI开发平台ModelArts	ModelArts CommonOperations ModelArts Dependency Access
专属资源池	云审计服务CTS	CTS Administrator
	云容器引擎CCE	CCE Administrator
	裸金属服务器BMS	BMS FullAccess
	镜像服务IMS	IMS FullAccess
	数据加密服务DEW	DEW KeypairReadOnlyAccess
	虚拟私有云VPC	VPC FullAccess
	弹性云服务器ECS	ECS FullAccess
	弹性文件服务SFS	SFS Turbo FullAccess

控制台功能	依赖服务	需配置角色/策略
	对象存储服务OBS	OBS Administrator
	应用运维管理服务 AOM	AOM FullAccess
	标签管理服务TMS	TMS FullAccess
	AI开发平台ModelArts	ModelArts FullAccess
	费用中心	BSS Administrator

如果系统预置的权限，不满足您的授权要求，可以创建自定义策略。自定义策略中可以添加的授权项（Action）请参考**ModelArts资源权限项**。

目前云服务平台支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

下面为您介绍常用的ModelArts自定义策略样例。

- 示例1：授权镜像管理的权限。

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "modelarts:image:register",
                "modelarts:image:listGroup"
            ]
        }
    ]
}
```

- 示例2：拒绝用户创建、更新、删除专属资源池。

  拒绝策略需要同时配合其他策略使用，否则没有实际作用。用户被授予的策略中，一个授权项的作用如果同时存在Allow和Deny，则遵循**Deny优先原则**。

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "modelarts:*:*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "swr:*:*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "smn:*:*"
            ],
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "modelarts:pool:create",
                "modelarts:pool:update",
                "modelarts:pool:delete"
            ],
            "Effect": "Deny"
        }
    ]
}
```

- 示例3：多个授权项策略。

  一个自定义策略中可以包含多个授权项，且除了可以包含本服务的授权项外，还可以包含其他服务的授权项，可以包含的其他服务必须跟本服务同属性，即都是项目级服务或都是全局级服务。多个授权语句策略描述如下：

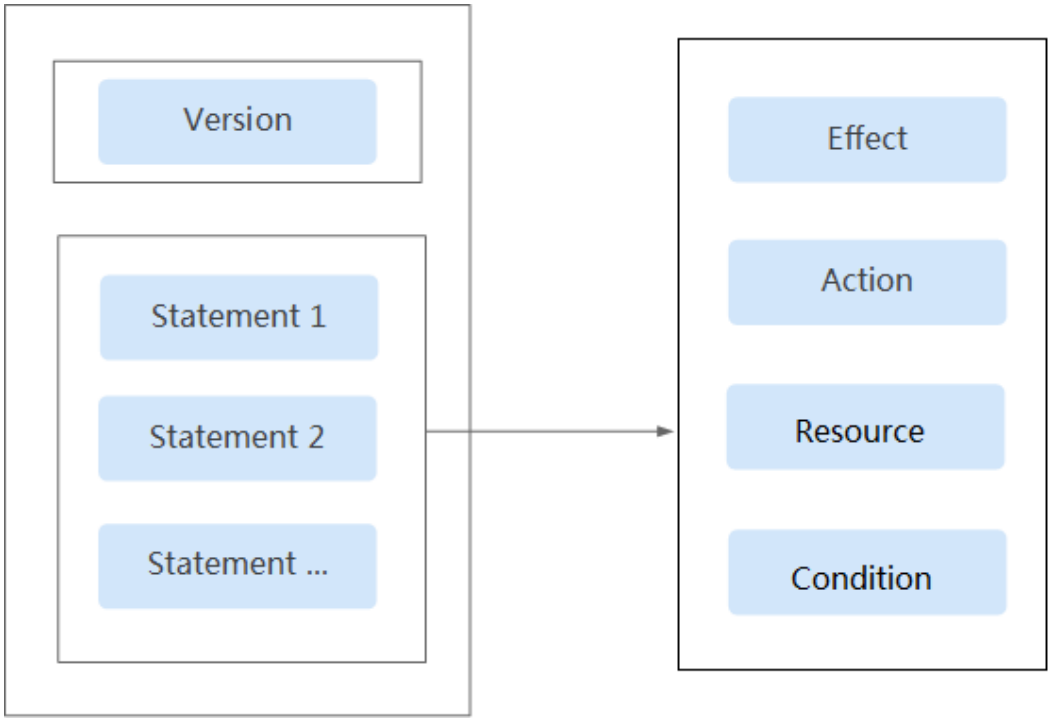
```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "modelarts:service:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lts:logs:list"
            ]
        }
    ]
}
```

策略 JSON 格式字段介绍

策略结构

策略结构包括Version（策略版本号）和Statement（策略权限语句）两部分，其中Statement可以有多个，表示不同的授权项。

图 12-3 策略结构



策略参数

下面介绍策略参数详细说明。了解策略参数后，您可以根据场景自定义策略。

表 12-4 策略参数说明

参数		含义	值
Version		策略的版本。	1.1：代表基于策略的访问控制。
Statement：策略的授权语句	Effect：作用	定义Action中的操作权限是否允许执行。	• Allow：允许执行。 • Deny：不允许执行。 **说明** 当同一个Action的Effect既有Allow又有Deny时，遵循Deny优先的原则。
	Action：授权项	操作权限。	格式为“服务名:资源类型:操作”。授权项支持通配符号*，通配符号*表示所有。 示例： "modelarts:notebook:list"：表示查看Notebook实例列表权限，其中modelarts为服务名，notebook为资源类型，list为操作。 您可以在对应服务“API参考”资料中查看该服务所有授权项。

参数		含义	值
	Condition：条件	使策略生效的特定条件，包括条件键和运算符。	格式为“条件运算符:{条件键：[条件值1,条件值2]}”。 如果您设置多个条件，同时满足所有条件时，该策略才生效。 示例: "StringEndWithIfExists":{"g:UserName":["specialCharacter"]}：表示当用户输入的用户名以"specialCharacter"结尾时该条statement生效。
	Resource: 资源类型	策略所作用的资源。	格式为“服务名:<region>:<account-id>:资源类型:资源路径”，资源类型支持通配符号*，通配符号*表示所有。 **说明** ModelArts的授权不支持指定具体资源路径。

ModelArts 资源类型

管理员可以按ModelArts的资源类型选择授权范围。ModelArts支持的资源类型如下表：

表 12-5 ModelArts 资源类型（角色与策略授权）

资源类型	说明
notebook	开发环境的Notebook实例
exemlProject	自动学习项目
exemlProjectInf	自动学习项目的在线推理服务
exemlProjectTrain	自动学习项目的训练作业
exemlProjectVersion	自动学习项目的版本
workflow	Workflow项目
pool	专属资源池
network	专属资源池网络连接
trainJob	训练作业
trainJobLog	训练作业的运行日志
trainJobInnerModel	系统预置模型
model	模型
service	在线服务
nodeservice	边缘服务

资源类型	说明
workspace	工作空间
dataset	数据集
dataAnnotation	数据集的标注信息
aiAlgorithm	训练算法
image	镜像
devserver	弹性裸金属

ModelArts 资源权限项

参考《 ModelArts API参考 》中的权限策略和授权项。

12.2.2 委托和依赖

功能依赖

功能依赖策略项

您在使用ModelArts服务中开发算法、管理训练作业过程中，需要和其他云服务交互，比如需要在提交训练作业时选择指定数据集OBS路径和日志存储OBS路径。因此管理员在为用户配置细粒度授权策略时，需要同时配置依赖的权限项，用户才能使用完整的功能。

说明

- 如果您使用根用户（与账户同名的缺省子用户）使用ModelArts，根用户默认拥有所有权限，不再需要单独授权。
- 请用户确保当前用户具备委托授权中包含的依赖策略项权限。例如，用户给ModelArts的委托需要授权SWR Admin权限，需要保证用户本身具备SWR Admin权限。

表 12-6 基本配置

业务场景	依赖的服务	依赖策略项	支持的功能
全局配置	IAM	iam:users:listUsers	查询用户列表（仅管理员需要）
基本功能	IAM	iam:tokens:assume	使用委托获取用户临时认证凭据（必需）

表 12-7 管理工作空间

业务场景	依赖的服务	依赖策略项	支持的功能
工作空间	IAM	iam:users:listUsers	按用户进行工作空间授权
	ModelArts	modelarts:*:*delete*	删除工作空间时，同时清理空间内的资源

表 12-8 管理开发环境 Notebook

业务场景	依赖的服务	依赖策略项	支持的功能
开发环境实例生命周期管理	ModelArts	modelarts:notebook:create modelarts:notebook:list modelarts:notebook:get modelarts:notebook:update modelarts:notebook:delete modelarts:notebook:start modelarts:notebook:stop modelarts:notebook:updateStopPolicy modelarts:image:delete modelarts:image:list modelarts:image:create modelarts:image:get modelarts:pool:list modelarts:tag:list modelarts:network:get aom:metric:get aom:metric:list aom:alarm:list	实例的启动、停止、创建、删除、更新等依赖的权限。

业务场景	依赖的服务	依赖策略项	支持的功能
动态挂载存储配置	ModelArts	modelarts:notebook:listMountedStorages modelarts:notebook:mountStorage modelarts:notebook:getMountedStorage modelarts:notebook:umountStorage	动态挂载存储配置。
	OBS	obs:bucket:ListAllMyBuckets obs:bucket:ListBucket	
镜像管理	ModelArts	modelarts:image:register modelarts:image:listGroup	在镜像管理中注册和查看镜像。
保存镜像	SWR	SWR Admin	SWR Admin为SWR最大权限，用于： • 开发环境运行的实例，保存成镜像。 • 使用自定义镜像创建开发环境Notebook实例。
使用SSH功能	ECS	ecs:serverKeypairs:list ecs:serverKeypairs:get ecs:serverKeypairs:delete ecs:serverKeypairs:create	为开发环境Notebook实例配置登录密钥。
	DEW	kps:domainKeypairs:get kps:domainKeypairs:list	
挂载SFS Turbo盘	SFS Turbo	SFS Turbo FullAccess	子用户对SFS目录的读写操作权限。专属池Notebook实例挂载SFS（公共池不支持），且挂载的SFS不是当前子用户创建的。
查看所有实例	ModelArts	modelarts:notebook:listAllNotebooks	ModelArts开发环境界面上，查询所有用户的实例列表，适用于给开发环境的实例管理员配置该权限。
	IAM	iam:users:listUsers	

业务场景	依赖的服务	依赖策略项	支持的功能
VSCode插件（本地）/ PyCharm Toolkit（本地）	ModelArts	modelarts:notebook:listAllNotebooks modelarts:trainJob:create modelarts:trainJob:list modelarts:trainJob:update modelarts:trainJobVersion:delete modelarts:trainJob:get modelarts:trainJob:logExport modelarts:workspace:getQuotas（如果开通了**工作空间**功能，则需要配置此权限。）	从本地VSCode连接云上的Notebook实例、提交训练作业等。

业务场景	依赖的服务	依赖策略项	支持的功能
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUploadParts obs:object:AbortMultipartUpload obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:ModifyObjectMetaData	
	IAM	iam:projects:listProjects	从本地PyCharm查询IAM项目列表，完成连接配置。

表 12-9 管理训练作业

业务场景	依赖的服务	依赖策略项	支持的功能
训练管理	ModelArts	modelarts:trainJob:* modelarts:trainJobLog:* modelarts:aiAlgorithm:* modelarts:image:list modelarts:network:get modelarts:workspace:get	创建训练作业和查看训练日志。
		modelarts:workspace:getQuota	查询工作空间配额。如果开通了**工作空间**功能，则需要配置此权限。
		modelarts:tag:list	在训练作业中使用标签管理服务TMS。
	IAM	iam:credentials:listCredentials iam:agencies:listAgencies	使用配置的委托授权项。
	SFS Turbo	sfsturbo:shares:getShare sfsturbo:shares:getAllShares	在训练作业中使用SFS Turbo。
	SWR	swr:repository:listTags swr:repository:getRepository swr:repository:listRepositories	使用自定义镜像运行训练作业。
	SMN	smn:topic:publish smn:topic:list	通过SMN通知训练作业状态变化事件。

业务场景	依赖的服务	依赖策略项	支持的功能
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:GetObjectVersion obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion obs:object:ListMultipartUploadParts obs:object:AbortMultipartUpload obs:object:GetObjectAcl obs:object:GetObjectVersionAcl obs:bucket:PutBucketAcl obs:object:PutObjectAcl obs:object:ModifyObjectMetaData	使用OBS桶中的数据集运行训练作业。

表 12-10 使用 Workflow

业务场景	依赖的服务	依赖策略项	支持的功能
使用数据集	ModelArts	modelarts:dataset:getDataset modelarts:dataset:createDataset modelarts:dataset:createDatasetVersion modelarts:dataset:createImportTask modelarts:dataset:updateDataset modelarts:processTask:createProcessTask modelarts:processTask:getProcessTask modelarts:dataset:listDatasets	在工作流中使用ModelArts数据集

业务场景	依赖的服务	依赖策略项	支持的功能
管理AI应用	ModelArts	modelarts:model:list modelarts:model:get modelarts:model:create modelarts:model:delete modelarts:model:update	在工作流中管理ModelArts AI应用
部署上线	ModelArts	modelarts:service:get modelarts:service:create modelarts:service:update modelarts:service:delete modelarts:service:getLogs	在工作流中管理ModelArts在线服务
训练作业	ModelArts	modelarts:trainJob:get modelarts:trainJob:create modelarts:trainJob:list modelarts:trainJobVersion:list modelarts:trainJobVersion:create modelarts:trainJob:delete modelarts:trainJobVersion:delete modelarts:trainJobVersion:stop	在工作流中管理ModelArts训练作业
工作空间	ModelArts	modelarts:workspace:get modelarts:workspace:getQuotas	在工作流中使用ModelArts工作空间

业务场景	依赖的服务	依赖策略项	支持的功能
管理数据	OBS	obs:bucket:ListAllMybuckets（获取桶列表） obs:bucket:HeadBucket（获取桶元数据） obs:bucket:ListBucket（列举桶内对象） obs:bucket:GetBucketLocation（获取桶区域位置） obs:object:GetObject（获取对象内容、获取对象元数据） obs:object:GetObjectVersion（获取对象内容、获取对象元数据） obs:object:PutObject（PUT上传、POST上传、复制对象、追加写对象、初始化上传段任务、上传段、合并段） obs:object:DeleteObject（删除对象、批量删除对象） obs:object:DeleteObjectVersion（删除对象、批量删除对象） obs:object:ListMultipartUploadParts（列举已上传的段） obs:object:AbortMultipartUpload（取消多段上传任务） obs:object:GetObjectAcl（获取对象ACL） obs:object:GetObjectVersionAcl（获取对象ACL） obs:bucket:PutBucketAcl（设置桶ACL） obs:object:PutObjectAcl（设置对象ACL）	在工作流中使用OBS数据
工作流运行	IAM	iam:users:listUsers（查询用户列表） iam:agencies:getAgency（查询指定委托详情） iam:tokens:assume（获取委托Token）	在工作流运行时，调用ModelArts其他服务
集成DLI	DLI	dli:jobs:get（查询作业详情） dli:jobs:list_all（查询作业列表） dli:jobs:create（创建新作业）	在工作流中集成DLI

业务场景	依赖的服务	依赖策略项	支持的功能
集成MRS	MRS	mrs:job:get（查询作业详情） mrs:job:submit（创建并执行作业） mrs:job:list（查询作业列表） mrs:job:stop（停止作业） mrs:job:batchDelete（批量删除作业） mrs:file:list（查询文件列表）	在工作流中集成MRS

表 12-11 管理 AI 应用

业务场景	依赖的服务	依赖策略项	支持的功能
管理AI应用	SWR	swr:repository:deleteRepository swr:repository:deleteTag swr:repository:getRepository swr:repository:listTags	从自定义镜像导入 从OBS导入时使用自定义引擎

业务场景	依赖的服务	依赖策略项	支持的功能
	OBS	obs:bucket:ListAllMybuckets（获取桶列表） obs:bucket:HeadBucket（获取桶元数据） obs:bucket:ListBucket（列举桶内对象） obs:bucket:GetBucketLocation（获取桶区域位置） obs:object:GetObject（获取对象内容、获取对象元数据） obs:object:GetObjectVersion（获取对象内容、获取对象元数据） obs:object:PutObject（PUT上传、POST上传、复制对象、追加写对象、初始化上传段任务、上传段、合并段） obs:object:DeleteObject（删除对象、批量删除对象） obs:object:DeleteObjectVersion（删除对象、批量删除对象） obs:object:ListMultipartUploadParts（列举已上传的段） obs:object:AbortMultipartUpload（取消多段上传任务） obs:object:GetObjectAcl（获取对象ACL） obs:object:GetObjectVersionAcl（获取对象ACL） obs:bucket:PutBucketAcl（设置桶ACL） obs:object:PutObjectAcl（设置对象ACL）	从OBS导入模型 模型转换指定OBS路径

表 12-12 管理部署上线

业务场景	依赖的服务	依赖策略项	支持的功能
在线服务	LTS	lts:logs:list（查询日志列表）	查询和展示LTS日志

业务场景	依赖的服务	依赖策略项	支持的功能
	OBS	obs:bucket:GetBucketPolicy （获取桶策略） obs:bucket:HeadBucket （获取桶元数据） obs:bucket:ListAllMyBuckets （获取桶列表） obs:bucket:PutBucketPolicy （设置桶策略） obs:bucket:DeleteBucketPolicy （删除桶策略）	服务运行时容器挂载外部存储卷
批量服务	OBS	obs:object:GetObject（获取对象内容、获取对象元数据） obs:object:PutObject（PUT上传、POST上传、复制对象、追加写对象、初始化上传段任务、上传段、合并段） obs:bucket:CreateBucket（创建桶） obs:bucket:ListBucket（列举桶内对象） obs:bucket:ListAllMyBuckets（获取桶列表）	创建批量服务，批量推理。
边缘服务	CES	ces:metricData:list（查询指标数据）	查看服务的监控指标
	IEF	ief:deployment:delete（删除应用部署）	管理边缘服务

表 12-13 管理数据集

业务场景	依赖的服务	依赖策略项	支持的功能
管理数据集和标注	OBS	obs:bucket:ListBucket（列举桶内对象） obs:object:GetObject（获取对象内容、获取对象元数据） obs:object:PutObject（PUT上传、POST上传、复制对象、追加写对象、初始化上传段任务、上传段、合并段） obs:object:DeleteObject（删除对象、批量删除对象） obs:bucket:HeadBucket（获取桶元数据） obs:bucket:GetBucketAcl（获取桶ACL） obs:bucket:PutBucketAcl（设置桶ACL） obs:bucket:GetBucketPolicy（获取桶策略） obs:bucket:PutBucketPolicy（设置桶策略） obs:bucket:DeleteBucketPolicy（删除桶策略） obs:bucket:PutBucketCORS（设置桶的CORS配置、删除桶的CORS配置） obs:bucket:GetBucketCORS（获取桶的CORS配置） obs:object:PutObjectAcl（设置对象ACL）	管理OBS中的数据集 标注OBS数据 创建数据管理作业
管理表格数据集	DLI	dli:database:displayAllDatabases dli:database:displayAllTables dli:table:describe_table	在数据集中管理DLI数据
管理表格数据集	DWS	dws:openAPICluster:list dws:openAPICluster:getDetail	在数据集中管理DWS数据
管理表格数据集	MRS	mrs:job:submit mrs:job:list mrs:cluster:list mrs:cluster:get	在数据集中管理MRS数据

业务场景	依赖的服务	依赖策略项	支持的功能
智能标注	ModelArts	modelarts:service:list modelarts:model:list modelarts:model:get modelarts:model:create modelarts:trainJobInnerModel:list modelarts:workspace:get modelarts:workspace:list	使用智能标注
团队标注	IAM	iam:projects:listProjects（查询租户项目） iam:users:listUsers（查询用户列表） iam:agencies:createAgency（创建委托） iam:quotas:listQuotasForProject（查询指定项目的配额）	管理标注团队

表 12-14 资源管理

业务场景	依赖的服务	依赖策略项	支持的功能
资源池管理	BSS	bss:coupon:view bss:order:view bss:balance:view bss:discount:view bss:renewal:view bss:bill:view bss:contract:update bss:order:pay bss:unsubscribe:update bss:renewal:update bss:order:update	资源池的创建、续费、退订等与计费相关的功能。依赖权限需要配置在IAM项目视图中。
	CCE	cce:cluster:list cce:cluster:get	获取CCE集群列表、集群详情、集群证书等信息。依赖权限需要配置在IAM项目视图中。

业务场景	依赖的服务	依赖策略项	支持的功能
	KMS	kms:cmk:list kms:cmk:getMaterial	获取用户创建的密钥对列表信息。依赖权限需要配置在IAM项目视图中。
	AOM	aom:metric:get	获取资源池的监控数据。依赖权限需要配置在IAM项目视图中。
	OBS	obs:bucket:ListAllMybuckets obs:bucket:HeadBucket obs:bucket:ListBucket obs:bucket:GetBucketLocation obs:object:GetObject obs:object:PutObject obs:object:DeleteObject obs:object:DeleteObjectVersion	获取AI诊断日志。依赖权限需要配置在IAM项目视图中。
	ECS	ecs:availabilityZones:list	查询可用区列表。依赖权限需要配置在IAM项目视图中。

业务场景	依赖的服务	依赖策略项	支持的功能
网络管理	VPC	vpc:routes:create vpc:routes:list vpc:routes:get vpc:routes:delete vpc:peerings:create vpc:peerings:accept vpc:peerings:get vpc:peerings:delete vpc:routeTables:update vpc:routeTables:get vpc:routeTables:list vpc:vpcs:create vpc:vpcs:list vpc:vpcs:get vpc:vpcs:delete vpc:subnets:create vpc:subnets:get vpc:subnets:delete vpcep:endpoints:list vpcep:endpoints:create vpcep:endpoints:delete vpcep:endpoints:get vpc:ports:create vpc:ports:get vpc:ports:update vpc:ports:delete vpc:networks:create vpc:networks:get vpc:networks:update vpc:networks:delete	ModelArts网络资源创建和删除、VPC网络打通。依赖权限需要配置在IAM项目视图中。
	SFS Turbo	sfsturbo:shares:addShareNic sfsturbo:shares:deleteShareNic sfsturbo:shares:showShareNic sfsturbo:shares:listShareNics	用户的网络和SFS Turbo资源打通。依赖权限需要配置在IAM项目视图中。

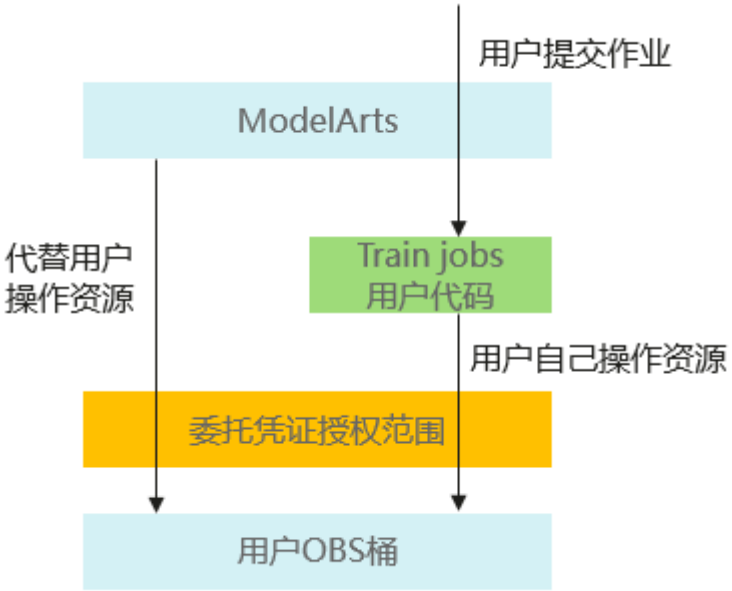
业务场景	依赖的服务	依赖策略项	支持的功能
边缘资源池	IEF	ief:node:list ief:group:get ief:application:list ief:application:get ief:node:listNodeCert ief:node:get ief:IEFInstance:get ief:deployment:list ief:group:listGroupInstanceState ief:IEFInstance:list ief:deployment:get ief:group:list	边缘池增删改查管理

委托授权

用户在使用ModelArts服务运行作业过程中，为了简化用户的操作，ModelArts后台可以代替用户完成一些工作，如训练作业启动前自动下载用户OBS桶中的数据集到作业空间、自动转储训练作业日志到用户OBS桶中。

ModelArts服务不会保存用户的Token认证凭据，在后台异步作业中操作用户的资源（如OBS桶）前，需要用户通过IAM委托向ModelArts显式授权，ModelArts在需要时使用用户的委托获取临时认证凭据用于操作用户资源，见“**添加授权**”。

图 12-4 委托授权



如图12-4所示，用户向ModelArts授权后，ModelArts使用委托授权的临时凭证访问和操作用户资源，协助用户自动化一些繁琐和耗时的操作。同时，委托凭证会同步到用户的作业中（Notebook实例和训练作业），客户在作业中可以使用委托凭证自行访问自己的资源。

在ModelArts服务中委托授权有两种方式：

1、一键式委托授权

ModelArts提供了一键式自动授权功能，用户可以在ModelArts的全局配置功能中，快速完成委托授权，由ModelArts为用户自动创建委托并配置到ModelArts服务中。

这种方式为保证使用业务过程中有足够的权限，基于依赖服务的预置系统策略指定授权范围，创建的委托的权限比较大，基本覆盖了依赖服务的全部权限。如果您需要对委托授权的权限范围进行精确控制，请使用第二种方式。

2、定制化委托授权

管理员在IAM中为不同用户创建不同的委托授权策略，再到ModelArts中为用户配置已创建好的委托。管理员在IAM中为用户创建委托时，根据用户的实际权限范围为委托指定最小权限范围，控制用户在使用ModelArts过程中可访问的资源内容。

委托授权的越权风险

可以看到用户的委托授权是独立的，理论上用户的委托授权范围是可以超出用户自身用户组的授权策略的授权范围，如果配置不当就会出现用户越权的问题。

为了控制委托授权的越权风险，ModelArts服务的全局配置功能要求只有租户管理员才能为用户配置委托，由管理员保证委托授权的安全性。

委托授权的最小化

管理员在配置委托授权时，应严格控制授权的范围。

ModelArts为用户异步自动化完成作业的准备、清理等操作，所需的委托授权内容是基础授权范围。如果用户只使用ModelArts的部分功能，管理员可以依据委托授权表格的说明屏蔽不使用的基础权限项。相反地，如果用户需要在作业中使用基础授权范围外的资源权限，管理员也可以为用户在委托授权中增加新的权限项。总之，委托授权的范围应该基于实际业务场景所需权限范围来进行定制，保持委托授权范围的最小化。

委托基础授权范围

当您需要定制委托授权的权限列表时，请参考下面表格，根据实际业务选择授权项。

表 12-15 开发环境基础委托授权

业务场景	依赖的服务	委托授权项	说明	配置建议
JupyterLab	OBS	obs:object:DeleteObject obs:object:GetObject obs:object:GetObjectVersion obs:bucket:CreateBucket obs:bucket:ListBucket obs:bucket:ListAllMyBuckets obs:object:PutObject obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:PutBucketCORS	通过ModelArts的Notebook，在JupyterLab中使用OBS上传下载数据。	建议配置。
开发环境监控功能	AOM	aom:alarm:put	调用AOM的接口，获取Notebook相关的监控数据和事件，展示在ModelArts的Notebook中。	建议配置。

表 12-16 训练作业基础委托授权

业务场景	依赖的服务	委托授权项	说明
训练作业	OBS	obs:bucket:ListBucket obs:object:GetObject obs:object:PutObject	训练作业启动前下载数据、模型、代码。 训练作业运行中上传日志、模型。

表 12-17 部署上线基础委托授权

业务场景	依赖的服务	委托授权项	说明
在线服务	LTS	lts:groups:create lts:groups:list lts:topics:create lts:topics:delete lts:topics:list	在线服务配置LTS日志上报

业务场景	依赖的服务	委托授权项	说明
批量服务	OBS	obs:bucket:ListBucket obs:object:GetObject obs:object:PutObject	运行批量服务
边缘服务	IEF	ief:deployment:list ief:deployment:create ief:deployment:update ief:deployment:delete ief:node:createNodeCert ief:iefInstance:list ief:node:list	通过IEF部署边缘服务

表 12-18 数据管理基础委托授权

业务场景	依赖的服务	委托授权项	说明
数据集和数据标注	OBS	obs:object:GetObject obs:object:PutObject obs:object:DeleteObject obs:object:PutObjectAcl obs:bucket:ListBucket obs:bucket:HeadBucket obs:bucket:GetBucketAcl obs:bucket:PutBucketAcl obs:bucket:GetBucketPolicy obs:bucket:PutBucketPolicy obs:bucket:DeleteBucketPolicy obs:bucket:PutBucketCORS obs:bucket:GetBucketCORS	管理OBS桶中的数据集
数据标注	ModelArts推理	modelarts:service:get modelarts:service:create modelarts:service:update	基于ModelArts推理进行智能数据标注

表 12-19 专属资源池管理基础委托授权

业务场景	依赖的服务	委托授权项	说明
资源池网络管理（新版本）	VPC	vpc:routes:create vpc:routes:list vpc:routes:get vpc:routes:delete vpc:peerings:create vpc:peerings:accept vpc:peerings:get vpc:peerings:delete vpc:routeTables:update vpc:routeTables:get vpc:routeTables:list vpc:vpcs:create vpc:vpcs:list vpc:vpcs:get vpc:vpcs:delete vpc:subnets:create vpc:subnets:get vpc:subnets:delete vpcep:endpoints:list vpcep:endpoints:create vpcep:endpoints:delete vpcep:endpoints:get vpc:ports:create vpc:ports:get vpc:ports:update vpc:ports:delete vpc:networks:create vpc:networks:get vpc:networks:update vpc:networks:delete	ModelArts网络资源创建和删除、VPC网络打通。此处依赖权限需要配置在IAM项目视图中。
	SFS Turbo	sfsturbo:shares:addShareNic sfsturbo:shares:deleteShareNic sfsturbo:shares:showShareNic sfsturbo:shares:listShareNics	用户的网络和SFS Turbo资源打通。依赖权限需要配置在IAM项目视图中。

业务场景	依赖的服务	委托授权项	说明
资源池管理	ECS	ecs:availabilityZones:list	查询可用区列表。依赖权限需要配置在IAM项目视图中。

12.2.3 工作空间

ModelArts的用户需要为不同的业务目标开发算法、管理和部署模型，此时可以创建多个工作空间，把不同应用开发过程的输出内容划分到不同工作空间中，便于管理和使用。

工作空间支持3种访问控制：

- PUBLIC：租户（主账号和所有子账号）内部公开访问。
- PRIVATE：仅创建者和主账号可访问。
- INTERNAL：创建者、主账号、指定IAM子账号可访问当授权类型为INTERNAL时需要指定可访问的子账号的账号名，可选择多个。

每个账号每个IAM项目都会分配1个默认工作空间，默认工作空间的访问控制为PUBLIC。

通过工作空间的访问控制能力，可限制仅允许部分人访问对应的工作空间。通过此功能可实现类似如下场景：

- **教育场景**：老师可给每个学生分配1个INTERNAL的工作空间并且限制该工作空间被指定学生访问，这样可使得学生可独立完成在ModelArts上的实验。
- **企业场景**：管理者可创建用于生产任务的工作空间并限制仅让运维人员使用，用于日常调试的工作空间并限制仅让开发人员使用。通过这种方式让不同的企业角色只能在指定工作空间下使用资源。

目前工作空间功能是“受邀开通”状态，作为企业用户您可以通过您对口的技术支持申请开通。

12.3 典型场景配置实践

12.3.1 个人用户快速配置 ModelArts 访问权限

ModelArts使用过程中涉及到OBS、SWR等服务交互，需要用户配置委托授权，允许ModelArts访问这些依赖服务。如果没有授权，ModelArts的部分功能将不能正常使用。

约束与限制

- 只有主账号可以使用委托授权，可以为当前账号授权，也可以为当前账号下的所有IAM用户授权。
- 多个IAM用户或账号，可使用同一个委托。

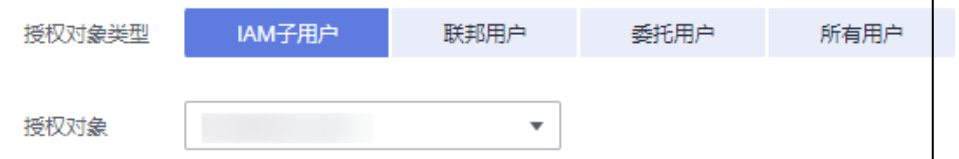
- 一个账号下，最多可创建50个委托。
- 对于首次使用ModelArts新用户，请直接新增委托即可。一般用户新增普通用户权限即可满足使用要求。如果有精细化权限管理的需求，可以自定义权限按需设置。
- 如果未获得委托授权，当打开“访问授权”页面时，ModelArts会提醒您当前用户未配置授权，需联系此IAM用户的管理员账号进行委托授权。

添加授权

1. 登录ModelArts管理控制台，在左侧导航栏选择“全局配置”，进入“全局配置”页面。
2. 单击“添加授权”，进入“访问授权”配置页面，根据参数说明进行配置。

表 12-20 参数说明

参数	说明
“授权对象类型”	包括IAM子用户、联邦用户、委托用户和所有用户。 ● IAM子用户：由主账号在IAM中创建的用户，是服务的使用人员，具有独立的身份凭证（密码和访问密钥），根据账号授予的权限使用资源。 ● 联邦用户：又称企业虚拟用户。 ● 委托用户：IAM中创建的一个委托。I ● 所有用户：该选项表示会将委托的权限授权到当前账号下的所有子账号、包括未来创建的子账号，授权范围较大，需谨慎使用。个人用户选择“所有用户”即可。

参数	说明
“授权对象”	“授权对象类型”选择“所有用户”时不涉及此参数。 • IAM子用户：选择指定的IAM子用户，给指定的IAM子用户配置委托授权。 **图 12-5** 选择 IAM 子用户  • 联邦用户：输入联邦用户的用户名或用户ID。 **图 12-6** 选择联邦用户  • 委托用户：选择委托名称。使用账号A创建一个权限委托，在此处将该委托授权给账号B拥有的委托。在使用账号B登录控制台时，可以在控制台右上角的个人账号切换角色到账号A，使用账号A的委托权限。 **图 12-7** 委托用户切换角色 
“委托选择”	• 已有委托：列表中如果已有委托选项，则直接选择一个可用的委托为上述选择的用户授权。单击委托名称查看该委托的权限详情。 • 新增委托：如果没有委托可选，可以在新增委托中创建委托权限。对于首次使用ModelArts的用户，需要新增委托。
“新增委托 > 委托名称”	系统自动创建委托名称，用户可以手动修改。

参数	说明
“新增委托 > 授权方式”	• 角色授权：IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。该机制以服务为粒度，提供有限的服务相关角色用于授权。由于各服务之间存在业务依赖关系，因此给用户授予角色时，可能需要一并授予依赖的其他角色，才能正确完成业务。角色并不能满足用户对精细化授权的要求，无法完全达到企业对权限最小化的安全管控要求。 • 策略授权：IAM最新提供的一种细粒度授权的能力，可以精确到具体服务的操作、资源以及请求条件等。基于策略的授权是一种更加灵活的授权方式，能够满足企业对权限最小化的安全管控要求。
“新增委托 > 权限配置 > 普通用户”	普通用户包括用户使用ModelArts完成AI开发的所有必要功能权限，如数据的访问、训练任务的创建和管理等。一般用户选择此项即可。 可以单击“查看权限列表”，查看普通用户权限。
“新增委托 > 权限配置 > 自定义”	如用户有精细化权限管理的需求，可使用自定义模式灵活按需配置ModelArts创建的委托权限。可以根据实际需要在权限列表中勾选要配置的权限。

3. 单击“创建”，即可完成委托配置。

查看授权的权限列表

用户可以在“全局配置”页面的授权列表中，查看已经配置的委托授权内容。单击授权内容列的“查看权限”，可以查看该授权的权限详情。

图 12-8 查看权限

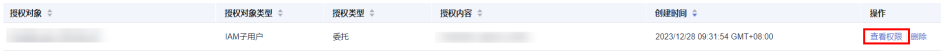


图 12-9 普通用户权限列表

权限名称	使用模块	描述
OBS Administrator	数据管理 | 开发环境 | 训练管理 | AI应用管理 | 部署上线 | AI ...	对象存储服务管理员
DLI FullAccess	数据管理	数据湖探索的所有执行权限
MRS Administrator	数据管理	MapReduce服务（MRS）管理员，拥有该服务下的所有权限
DWS Administrator	数据管理	数据仓库服务（DWS）管理员，拥有该服务下的所有权限
VPC Administrator	训练管理 | AI应用管理 | 部署上线 | 专属资源池	虚拟私有云管理员
CES ReadOnlyAccess	AI应用管理 | 部署上线	云监控服务只读权限
SMN Administrator	训练管理 | AI应用管理 | 部署上线	消息通知服务管理员
EPS FullAccess	训练管理 | AI应用管理 | 部署上线	企业项目管理服务所有权限
CTS Administrator	数据管理 | 开发环境 | 训练管理 | AI应用管理 | 部署上线 | AI ...	云审计服务（CTS）管理员，拥有该服务下的所有权限
SFS ReadOnlyAccess	训练管理	弹性文件服务只读权限
LTS FullAccess	AI应用管理 | 部署上线	云日志服务所有权限
ModelArts CommonOperations	数据管理 | 开发环境 | 训练管理 | AI应用管理 | 部署上线 | AI ...	ModelArts服务普通用户权限（不包括创建、更新、删除专...

12.3.2 管理员和开发者权限分离

对于中小规模团队，管理员希望对ModelArts资源进行主导分配，全局控制，而对于普通开发者只需关注自己实例的生命周期控制。对于开发者账号，一般不会具有

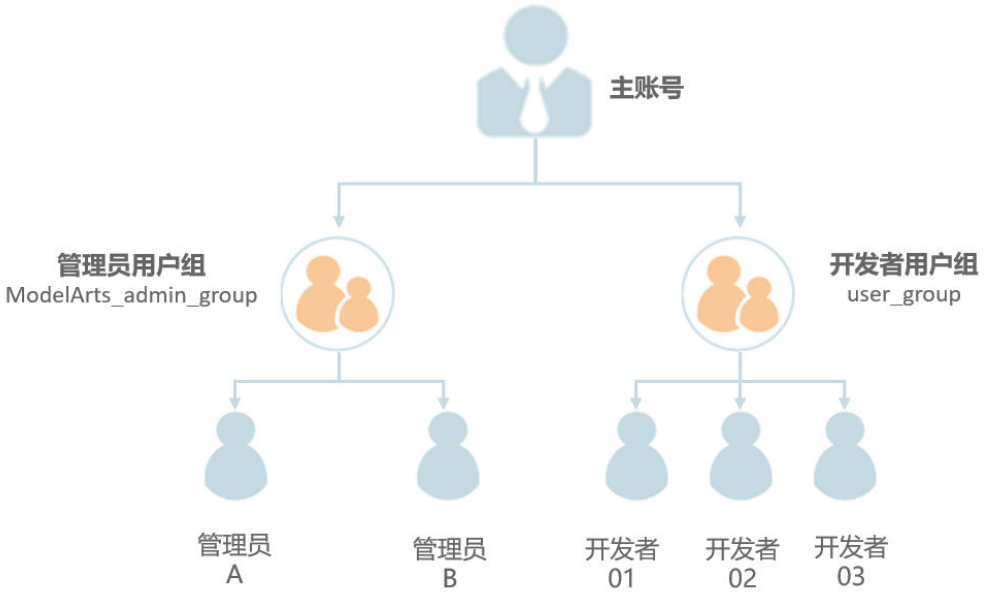
te_admin的权限，相应的权限也需要主账号进行统一配置。本章节以使用Notebook进行项目开发为例，通过自定义策略配置实现管理员和开发者分离。

场景描述

以使用Notebook进行项目开发为例，管理员账号需要拥有ModelArts专属资源池的完全控制权限，以及Notebook所有实例的访问和操作权限。

普通开发者使用开发环境，只需关注对自己Notebook实例的操作权限，包括对自己实例的创建、启动、停止、删除等权限以及周边依赖服务的权限。普通开发者不需要ModelArts专属资源池的操作权限，也不需要查看其他用户的Notebook实例。

图 12-10 账号关系示意图



配置管理员权限

管理员账号需要拥有ModelArts专属资源池的完全控制权限，以及Notebook所有实例的访问和操作权限。可以通过以下配置流程实现管理员权限配置。

步骤1 使用主账号创建一个管理员用户组ModelArts_admin_group，将管理员账号加入用户组ModelArts_admin_group中。

步骤2 创建自定义策略。

1. 使用管理员账号登录控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入IAM服务。
2. 创建自定义策略1，赋予用户IAM和OBS服务权限。在统一身份认证服务控制台的左侧菜单栏中，选择“权限管理> 权限”。单击右上角“创建自定义策略”，在“策略名称”中填入“Policy1_IAM_OBS”，策略配置方式选择JSON视图，输入策略内容，单击“确定”。

   自定义策略“Policy1_IAM_OBS”的具体内容如下，赋予用户IAM和OBS操作权限。可以直接复制粘贴。

```
{
    "Version": "1.1",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:users:listUsers",
                "iam:projects:listProjects",
                "obs:object:PutObject",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:bucket:HeadBucket",
                "obs:object:DeleteObject",
                "obs:bucket:CreateBucket",
                "obs:bucket:ListBucket"
                ]
        }
    ]
}
```

3. 重复2.2创建自定义策略2，赋予用户依赖服务ECS、SWR、MRS和SMN的操作权限，ModelArts的操作权限。“策略名称”为“Policy2_AllowOperation”，策略配置方式选择JSON视图，输入策略内容，单击“确定”。

   自定义策略“Policy2_AllowOperation”的具体内容如下，赋予用户依赖服务ECS、SWR、MRS和SMN的操作权限，ModelArts的操作权限。可以直接复制粘贴。

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:serverKeypairs:list",
                "ecs:serverKeypairs:get",
                "ecs:serverKeypairs:delete",
                "ecs:serverKeypairs:create",
                "swr:repository:getNamespace",
                "swr:repository:listNamespaces",
                "swr:repository:deleteTag",
                "swr:repository:getRepository",
                "swr:repository:listTags",
                "swr:instance:createTempCredential",
                "mrs:cluster:get",
                "modelarts:*:*"
            ]
        }
    ]
}
```

步骤3 将2创建的自定义策略授权给管理员用户组ModelArts_admin_group。

1. 在统一身份认证服务控制台的左侧菜单栏中，选择“用户组”。在用户组页面单击对应用户组名称ModelArts_admin_group操作列的“授权”，勾选策略“Policy1_IAM_OBS”和“Policy2_AllowOperation”。单击“下一步”。
2. 选择授权范围方案为所有资源，单击“确定”。

步骤4 给管理员用户配置ModelArts委托授权，允许ModelArts服务在运行时访问OBS等依赖服务。

1. 使用主账号登录ModelArts的管理控制台，在左侧导航栏单击“全局配置”，进入“全局配置”页面。
2. 单击“添加授权”。在“访问授权”页面，在“授权对象类型”下面选择“IAM子用户”，“授权对象”选择管理员的账号，选择“新增委托”，“权限配置”选择“普通用户”。管理员不做权限控制，此处默认使用普通用户委托即可。
3. 单击“创建”。

步骤5 测试管理员用户权限。

1. 使用管理员用户登录ModelArts管理控制台。在登录页面，请使用“IAM用户登录”方式进行登录。

   首次登录会提示修改密码，请根据界面提示进行修改。

2. 在ModelArts控制台的左侧导航栏中，选择“专属资源池”，单击创建，未提示权限不足，表明管理员用户的权限配置成功。

----结束

配置开发者权限

开发者权限需要通过IAM的细粒度授权控制实现，可以通过以下配置流程实现开发者权限配置。

步骤1 使用主账号创建一个开发者用户组user_group，将开发者账号加入用户组user_group中。

步骤2 创建自定义策略。

1. 使用主账号登录控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入IAM服务。
2. 创建自定义策略3，拒绝用户操作ModelArts专属资源池并拒用户查看其他用户的Notebook。

   在统一身份认证服务控制台的左侧菜单栏中，选择“权限管理> 权限”。单击右上角“创建自定义策略”，“策略名称”为“Policy3_DenyOperation”，策略配置方式选择JSON视图，输入策略内容，单击“确定”。

   自定义策略“Policy3_DenyOperation”的具体内容如下，可以直接复制粘贴。

```
{
   "Version": "1.1",
   "Statement": [
         {
          "Effect": "deny",
          "Action": [
              "modelarts:pool:create",
              "modelarts:pool:update",
              "modelarts:pool:delete",
       "modelarts:notebook:listAllNotebooks"
            ]

       }
   ]
}
```

步骤3 将自定义策略授权给开发者用户组user_group。

1. 在统一身份认证服务控制台的左侧菜单栏中，选择“用户组”。在用户组页面单击对应用户组名称user_group操作列的“授权”，勾选策略“Policy1_IAM_OBS”、“Policy2_AllowOperation”和“Policy3_DenyOperation”。单击“下一步”。
2. 选择授权范围方案为所有资源，单击“确定”。

步骤4 给开发者用户配置ModelArts委托授权，允许ModelArts服务在运行时访问OBS等依赖服务。

1. 使用主账号登录ModelArts的管理控制台，在左侧导航栏单击“全局配置”，进入“全局配置”页面。
2. 单击“添加授权”。在“访问授权”页面，在“授权对象类型”下面选择“IAM子用户”，“授权对象”选择开发者的账号，“委托选择”选择“新增委托”，“委托名称”设置为“ma_agency_develop_user”，“权限配置”选择“自定

义”，“权限名称”勾选“OBS Administrator”。开发者用户只需要配置OBS的委托授权即可，允许开发者用户在使用Notebook时，与OBS服务交互。

3. 单击“创建”。
4. 在“全局配置”页面，再次单击“添加授权”，进入“访问授权”页面，为其他开发者用户配置委托。

   “授权对象类型”选择“IAM子用户”，“授权对象”选择开发者的账号，“委托选择”选择“已有委托”，“委托名称”勾选上一步创建的“ma_agency_develop_user”，

步骤5 测试开发者用户权限。

1. 使用user_group用户组中任意一个子用户登录ModelArts管理控制台。在登录页面，请使用“IAM用户登录”方式进行登录。

   首次登录会提示修改密码，请根据界面提示进行修改。
2. 在ModelArts左侧菜单栏中，选择“专属资源池”，单击创建，界面未提示权限不足，表明开发者用户的权限配置成功。

----结束

12.3.3 查看所有子账号的 Notebook 实例

当子用户被授予“listAllNotebooks”和“listUsers”权限时，在Notebook页面上，单击“查看所有”，可以看到IAM项目下所有子用户创建的Notebook实例 。

说明

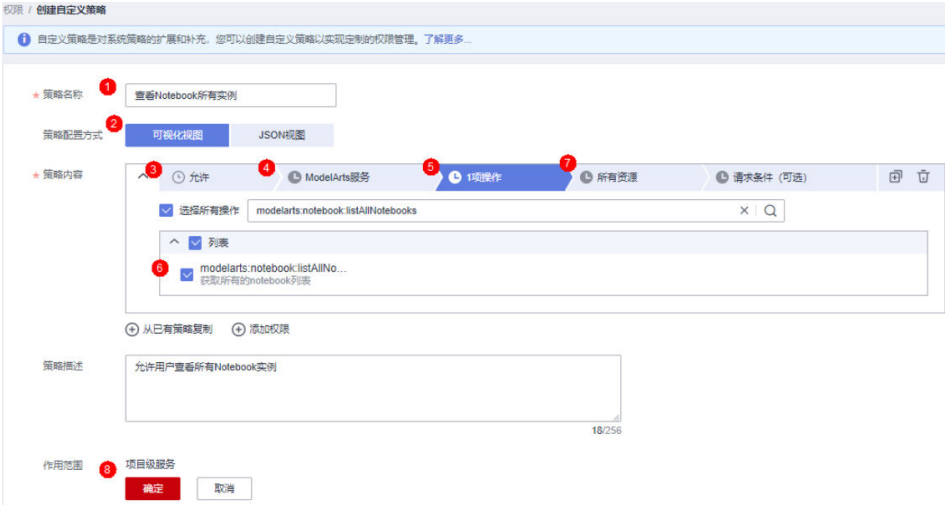
配置该权限后，可以查看子用户的Notebook，也可以在Notebook中访问子用户的OBS、SWR等。

给子账号配置查看所有 Notebook 实例的权限

1. 使用主用户账号登录ModelArts管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。
2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”，需要设置两条策略。

   策略1：设置查看Notebook所有实例，如图12-11所示，单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：查看Notebook所有实例。
   - “策略配置方式”：选择可视化视图。
   - “策略内容”：允许，云服务中搜索ModelArts服务并选中，操作列中搜索关键词modelarts:notebook:listAllNotebooks并选中，所有资源选择默认值。

图 12-11 创建自定义策略



策略2：设置查看Notebook实例创建者信息的策略。

- “策略名称”：设置自定义策略名称，例如：查看所有子用户信息。
- “策略配置方式”：选择可视化视图。
- “策略内容”：允许，云服务中搜索IAM服务并选中，操作列中搜索关键词iam:users:listUsers并选中，所有资源选择默认值。

3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的两条自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限查看该用户组内成员创建的所有Notebook实例。

   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

子用户启动其他用户的 SSH 实例

子用户可以看到所有用户的Notebook实例后，如果要通过SSH方式远程连接其他用户的Notebook实例，需要将SSH密钥对更新成自己的，否则会报错ModelArts.6789，更新密钥对具体操作请参见**修改Notebook SSH远程连接配置**。

具体的错误信息提示：ModelArts.6789：当前用户没有权限使用ssh密钥对xxx，请更新实例密钥对并重试。

12.3.4 使用 Cloud Shell 登录训练容器

使用场景

允许用户使用ModelArts控制台提供的Cloud Shell登录运行中的训练容器。

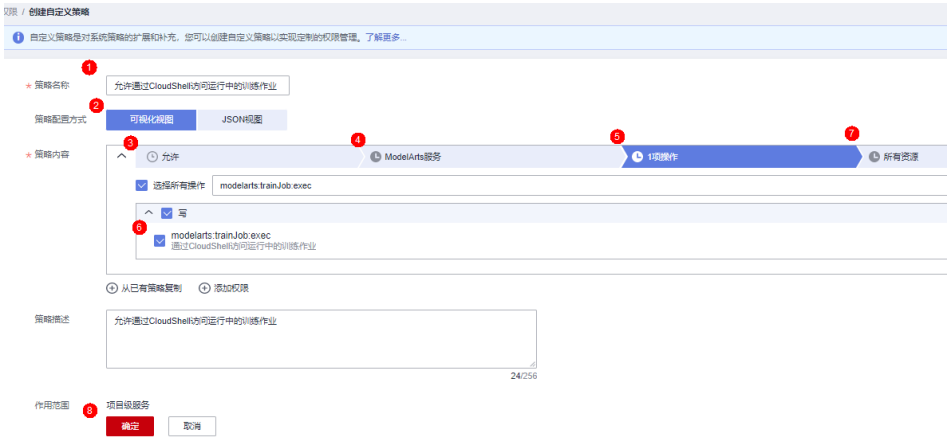
约束限制

仅专属资源池均支持使用Cloud Shell，且训练作业必须处于“运行中”状态。

前提条件：给子账号配置允许使用 Cloud Shell 的权限

1. 使用主用户账号登录管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。
2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”按如下要求设置完成后单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：允许通过Cloud Shell访问运行中的训练作业。
   - “策略配置方式”：选择可视化视图。
   - “策略内容”：允许，云服务中搜索ModelArts服务并选中，操作列中搜索关键词modelarts:trainJob:exec并选中，所有资源选择默认值。

   图 12-12 创建自定义策略

3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限通过Cloud Shell登录运行中的训练作业容器。

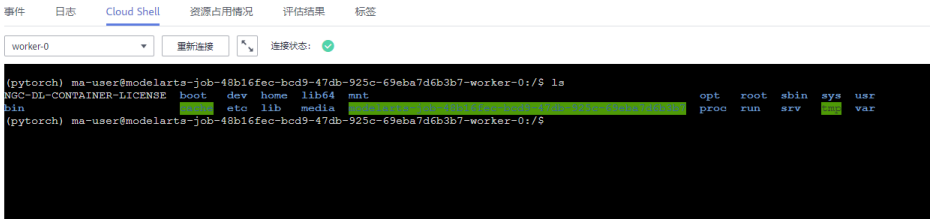
   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

如何使用 Cloud Shell

1. 参考**前提条件：给子账号配置允许使用Cloud Shell的权限**，完成配置。
2. 在ModelArts管理控制台的左侧导航栏中选择“训练管理 > 训练作业”。
3. 在训练作业列表中，单击作业名称进入训练作业详情页面。
4. 在训练作业详情页面，单击“Cloud Shell”页签，登录训练容器。

   连接成功后，Cloud Shell界面提示如下。

图 12-13 Cloud Shell 界面

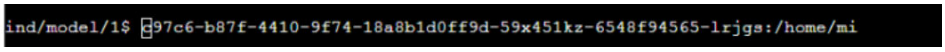


当作业处于非运行状态或权限不足时会导致无法使用Cloud Shell，请根据提示定位原因即可。

说明

部分用户登录Cloud Shell界面时，可能会出现路径显示异常情况，此时在Cloud Shell中单击回车键即可恢复正常。

图 12-14 路径异常



12.3.5 限制用户使用公共资源池

本章节介绍如何控制ModelArts用户权限，限制用户使用ModelArts公共资源池的资源创建训练作业、创建开发环境实例，部署推理服务等。

场景介绍

对于ModelArts专属资源池的用户，不允许使用公共资源池创建训练作业、创建Notebook实例或者部署推理服务时，可以通过权限控制限制用户使用公共资源池。

涉及配置的自定义权限策略项如下;

- modelarts:notebook:create：此策略项表示创建Notebook实例。
- modelarts:trainJob:create：此策略项表示创建训练作业。
- modelarts:service:create：此策略项表示创建推理服务。

给子账号配置权限：限制使用公共资源池

1. 使用主用户账号登录管理控制台，单击右上角用户名，在下拉框中选择“统一身份认证”，进入统一身份认证（IAM）服务。
2. 在统一身份认证服务页面的左侧导航选择“权限管理 > 权限”，单击右上角的“创建自定义策略”，设置策略，单击“确定”。
   - “策略名称”：设置自定义策略名称，例如：不允许用户使用公共资源池创建。
   - “策略配置方式”：选择可视化视图或者JSON视图均可。
   - “策略内容”：拒绝，云服务中搜索“ModelArts”服务并选中，“操作”中查找写操作“modelarts:trainJob:create”、“modelarts:notebook:create”和“modelarts:service:create”并选中。“所有资源”选择“默认值”。“请求条件”中单击“添加条件”，设置“条件键”为“modelarts:poolType”，“运算符”为“StringEquals”，“值”为“public”。

JSON视图的策略内容如下：

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "modelarts:trainJob:create",
                "modelarts:notebook:create",
                "modelarts:service:create"
            ],
            "Condition": {
                "StringEquals": {
                    "modelarts:poolType": [
                        "public"
                    ]
                }
            }
        }
    ]
}
```

3. 在统一身份认证服务页面的左侧导航选择“用户组”，在用户组页面查找待授权的用户组名称，在右侧的操作列单击“授权”，勾选步骤2创建的两条自定义策略，单击“下一步”，选择授权范围方案，单击“确定”。

   此时，该用户组下的所有用户均有权限查看该用户组内成员创建的所有Notebook实例。

   如果没有用户组，也可以创建一个新的用户组，并通过“用户组管理”功能添加用户，并配置授权。如果指定的子用户没有在用户组中，也可以通过“用户组管理”功能增加用户。

4. 在用户的委托授权中同步增加此策略，避免在租户面通过委托token突破限制。

   在统一身份认证服务页面的左侧导航中选择委托，找到该用户组在ModelArts上使用的委托名称，单击右侧的“修改”操作，选择“授权记录”页签，单击“授权”，选中上一步创建的自定义策略“不允许用户使用公共资源池”，单击“下一步”，选择允许使用的资源区域，单击“确定”。

验证

使用子账号用户登录ModelArts控制台，选择“训练管理 > 训练作业”，单击“创建训练作业”，在创建训练页面，资源池规格只能选择专属资源池。

使用子账号用户登录ModelArts控制台，选择“开发环境 > Notebook”，单击“创建”，在创建Notebook页面，资源池规格只能选择专属资源池。

使用子账号用户登录ModelArts控制台，选择“部署上线 > 在线服务”，单击“部署”，在部署服务页面，资源池规格只能选择专属资源池。

12.4 FAQ

12.4.1 使用 ModelArts 时提示“权限不足”，如何解决?

当您使用ModelArts时如果提示权限不足，请您按照如下指导对相关服务和用户进行授权，并对用户权限进行检查操作。

由于ModelArts的使用权限依赖OBS服务的授权，您需要为用户授予OBS的系统权限。

- 如果您需要授予用户关于OBS的所有权限和ModelArts的基础操作权限，请参见**配置基础操作权限**。
- 如果您需要对用户使用OBS和ModelArts的权限进行精细化管理，进行自定义策略配置，请参见**创建ModelArts自定义策略**。

配置基础操作权限

使用ModelArts的基本功能，您需要为用户配置“作用范围”为“项目级服务”的“ModelArts CommonOperations”权限，由于ModelArts依赖OBS权限，您还需要为用户授予“作用范围”为“全局级服务”的“OBS Administrator”策略。

具体操作步骤如下：

步骤1 创建用户组。

登录IAM管理控制台，单击“用户组>创建用户组”。在“创建用户组”界面，输入“用户组名称”单击“确定”。

步骤2 配置用户组权限。

在用户组列表中，单击步骤1新建的用户组右侧的“授权”，在用户组“授权”页面，您需要配置的权限如下：

1. 配置“作用范围”为“项目级服务”的“ModelArts CommonOperations”权限，如下图所示，然后单击“确定”完成授权。

   说明

   区域级项目授权后只在授权区域生效，如果需要所有区域都生效，则所有区域都需要进行授权操作。

2. 配置“作用范围”为“全局级服务”的“OBS Administrator”权限，然后单击“确定”完成授权。

步骤3 在IAM控制台创建用户，并将其加入步骤1中创建的用户组。

步骤4 新创建的用户登录控制台，切换至授权区域，验证权限：

- 在“服务列表”中选择ModelArts，进入ModelArts主界面，选择不同类型的专属资源池，在页面单击“创建”，如果无法进行创建（当前权限仅包含ModelArts CommonOperations），表“ModelArts CommonOperations”已生效。
- 在“服务列表”中选择除ModelArts外（假设当前策略仅包含ModelArts CommonOperations）的任一服务，如果提示权限不足，表示“ModelArts CommonOperations”已生效。
- 在“服务列表”中选择ModelArts，进入ModelArts主界面，单击“数据管理>数据集>创建数据 > 集”，如果可以成功访问对应的OBS路径，表示全局级服务的“OBS Administrator”已生效。

----结束

创建 ModelArts 自定义策略

如果系统预置的ModelArts权限不满足您的授权要求，或者您需要管理用户操作OBS的操作权限，可以创建自定义策略。

目前支持可视化视图创建自定义策略和JSON视图创建自定义策略，本章节将使用JSON视图方式的策略，以为ModelArts用户授予开发环境的使用权限并且配置ModelArts用户OBS相关的最小化权限项为例，指导您进行自定义策略配置。

说明

如果一个自定义策略中包含多个服务的授权语句，这些服务必须是同一属性，即都是全局级服务或者项目级服务。

由于OBS为全局服务，ModelArts为项目级服务，所以需要创建两条“作用范围”别为“全局级服务”以及“项目级服务”的自定义策略，然后将两条策略同时授予用户。

1. 创建ModelArts相关OBS的最小化权限的自定义策略。

   登录IAM控制台，在“权限管理>权限”页面，单击“创建自定义策略”。参数配置说明如下：

   - “策略名称”支持自定义。
   - “策略配置方式”为“JSON视图”。
   - “策略内容”请参见**ModelArts依赖的OBS权限自定义策略样例**。

2. 创建ModelArts开发环境的使用权限的自定义策略。参数配置说明如下：
   - “策略名称”支持自定义。
   - “策略配置方式”为“JSON视图”。
   - “策略内容”请参见**ModelArts开发环境使用权限的自定义策略样例**，ModelArts自定义策略中可以添加的授权项（Action）请参见《ModelArts API参考》>权限策略和授权项> 策略及授权项说明。
3. 在IAM控制台创建用户组之后，将步骤**1**中创建的自定义策略授权给该用户组。
4. 在IAM控制台创建用户，并将其加入**3**中创建的用户组。
5. 新创建的用户登录控制台，切换至授权区域，验证权限：
   - 在“服务列表”中选择ModelArts，进入ModelArts主界面，单击“数据管理 > 数据集”，如果无法进行创建（当前仅包含开发环境的使用权限），表示仅为ModelArts用户授予开发环境的使用权限已生效。
   - 在“服务列表”中选择除ModelArts，进入ModelArts主界面，单击“开发环境>Notebook>创建”，如果可以成功访问“存储配置”项对应的OBS路径，表示为用户配置的OBS相关权限已生效。

ModelArts 依赖的 OBS 权限自定义策略样例

如下示例为ModelArts依赖OBS服务的最小化权限项，包含OBS桶和OBS对象的权限。授予示例中的权限您可以通过ModelArts正常访问OBS不受限制。

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "obs:bucket:ListAllMybuckets",
                "obs:bucket:HeadBucket",
                "obs:bucket:ListBucket",
                "obs:bucket:GetBucketLocation",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:object:PutObject",
                "obs:object:DeleteObject",
                "obs:object:DeleteObjectVersion",
                "obs:object:ListMultipartUploadParts",
                "obs:object:AbortMultipartUpload",
                "obs:object:GetObjectAcl",
                "obs:object:GetObjectVersionAcl",
                "obs:bucket:PutBucketAcl",
                "obs:object:PutObjectAcl"
            ],
```

```
            "Effect": "Allow"
        }
    ]
}
```

ModelArts 开发环境使用权限的自定义策略样例

```
{
    "Version": "1.1",
    "Statement": [

        {
            "Effect": "Allow",
            "Action": [
                "modelarts:notebook:list",
                "modelarts:notebook:create" ,
                "modelarts:notebook:get" ,
                "modelarts:notebook:update" ,
                "modelarts:notebook:delete" ,
                "modelarts:notebook:action" ,
                "modelarts:notebook:access"
            ]
        }
    ]
}
```

13 最佳实践

13.1 MindSpore 模型快速迁移到 ModelArts 训练

本案例介绍如何将本地开发好的MindSpore模型代码，通过PyCharm ToolKit连接到ModelArts进行云上调试和训练。

开始使用样例前，请仔细阅读**准备工作**罗列的要求，提前完成准备工作。本案例的步骤如下所示：

步骤1：安装和登录PyCharm ToolKit

步骤2：使用PyCharm进行本地开发调试

步骤3：使用ModelArts Notebook进行开发调试

步骤4：使用PyCharm提交训练作业至ModelArts

步骤5：清除相应资源

准备工作

- 本地已安装PyCharm 2019.2或以上版本，社区版或专业版均可，请单击**PyCharm工具下载地址**获取工具并在本地完成安装。
  - 使用PyCharm ToolKit远程连接Notebook开发环境，仅限PyCharm专业版。
  - 使用PyCharm ToolKit提交训练作业，社区版和专业版都支持。
- 已创建当前使用账号的访问密钥，并获得对应的AK和SK。如果未创建，请参见**创建访问密钥（AK和SK）**。
- 当前账号已完成访问授权的配置。如未完成，请参考**使用委托授权**。

环境说明

- Python 3.7.6
- PyCharm 2023.1.3 (Professional Edition)

  说明

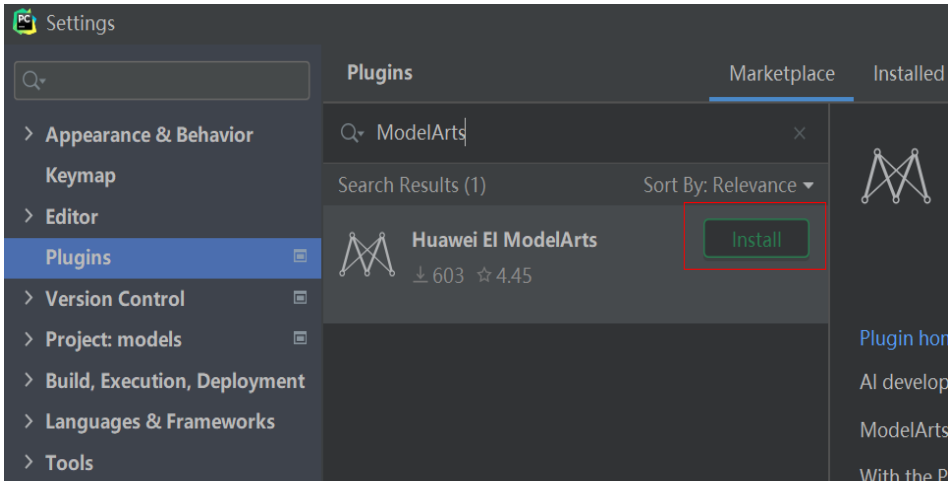
  本案例使用PyCharm版本为PyCharm 2023.1.3 (Professional Edition)，不同版本PyCharm之间部分界面可能不同，仅供参考。

步骤 1：安装和登录 PyCharm ToolKit

1. 安装PyCharm ToolKit

   在PyCharm中选择“File>Settings>Plugins”，在Marketplace里搜索“ModelArts”，单击“Install”即可完成安装。

   图 13-1 通过 Marketplace 安装

2. 登录PyCharm ToolKit

   a. 打开“Edit Credential”界面。

      安装完插件后，会在IDE菜单栏出现“ModelArts”，单击后选择“Edit Credential”。

   b. 联系局点运营公司获取YAML配置文件和host信息。

      把host信息追加到本地PC的hosts文件中，本地PC的hosts文件一般位于“C:\Windows\System32\drivers\etc”。

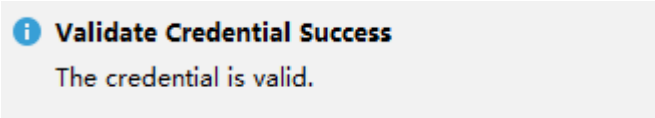
      在“Edit Credential”对话框中，单击“Config”，导入的YAML配置文件，导入后会显示“Import successful”的提示，此时可以看到局点信息已配置成功。

   c. 验证登录信息

      将创建访问密钥（AK和SK）输入到ToolKit对应位置，单击OK按钮进行登录，出现下图提示即为登录成功。

      如果未创建，请参见**创建访问密钥（AK和SK）**。

      图 13-2 成功登录提示

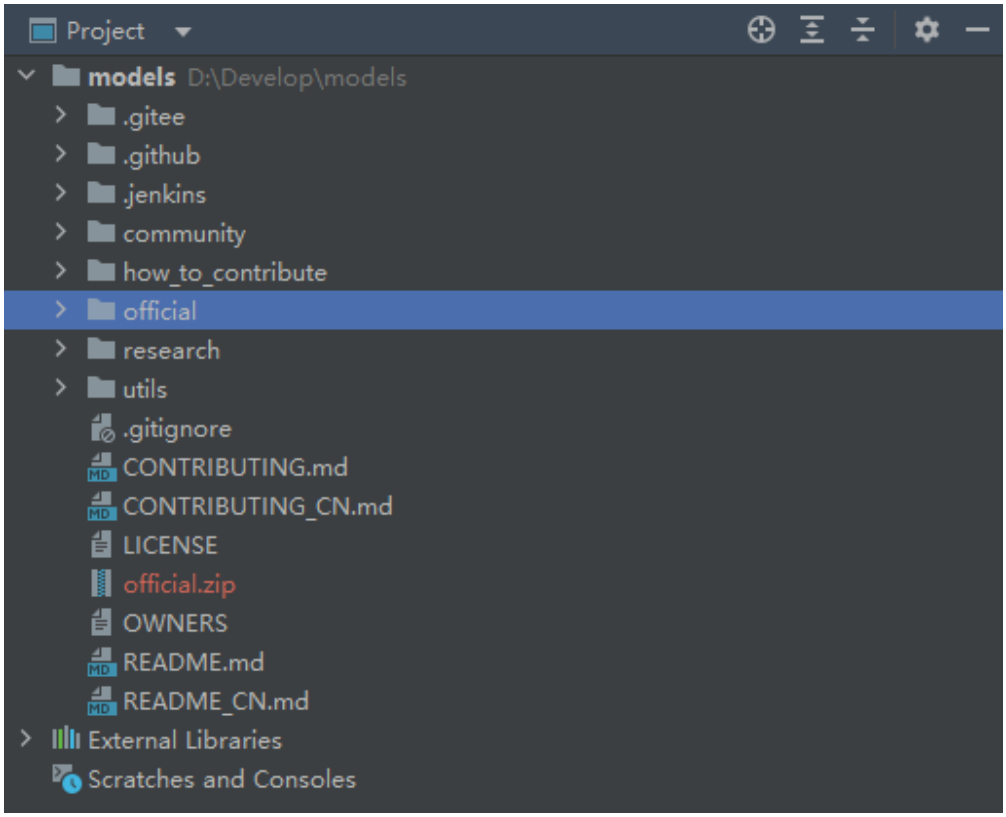
步骤 2：使用 PyCharm 进行本地开发调试

1. 下载代码至本地

   本案例中，以图像分类模型resnet50模型为例，路径为“./models/official/cv/resnet/”

   ```
   # 在本地电脑Terminal下载代码至本地
   git clone https://gitee.com/mindspore/models.git -b v1.5.0
   ```

图 13-3 下载代码至本地



2. 配置本地PC开发环境

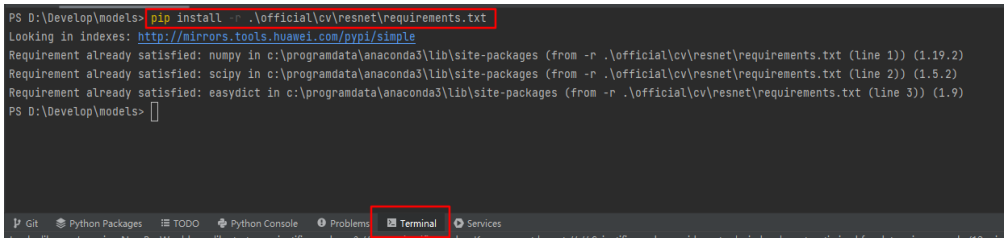
   修改“models/official/cv/resnet/requirements.txt”文件，改为：

   ```
   numpy==1.17.5
   scipy==1.5.4
   easydict==1.9
   ```

   执行pip命令安装：

   ```
   # 在PyCharm的Terminal安装mindspore
   pip install mindspore==1.7.0
   # 在PyCharm的Terminal安装resnet依赖
   pip install -r .\official\cv\resnet\requirements.txt
   ```

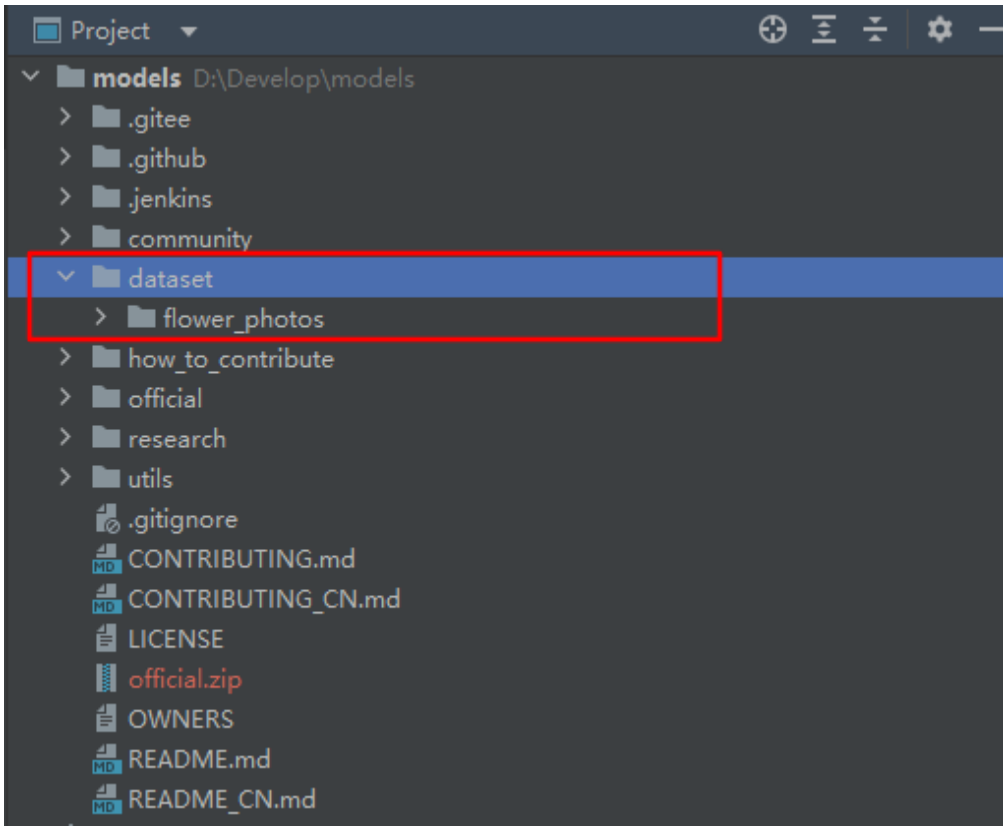
图 13-4 安装 resnet 依赖



3. 准备数据集

   本样例使用的数据集为类别数为五类的花卉识别数据集，**下载数据集**并解压数据到工程目录。新建dataset文件夹，将解压后数据集保存在dataset文件夹下。

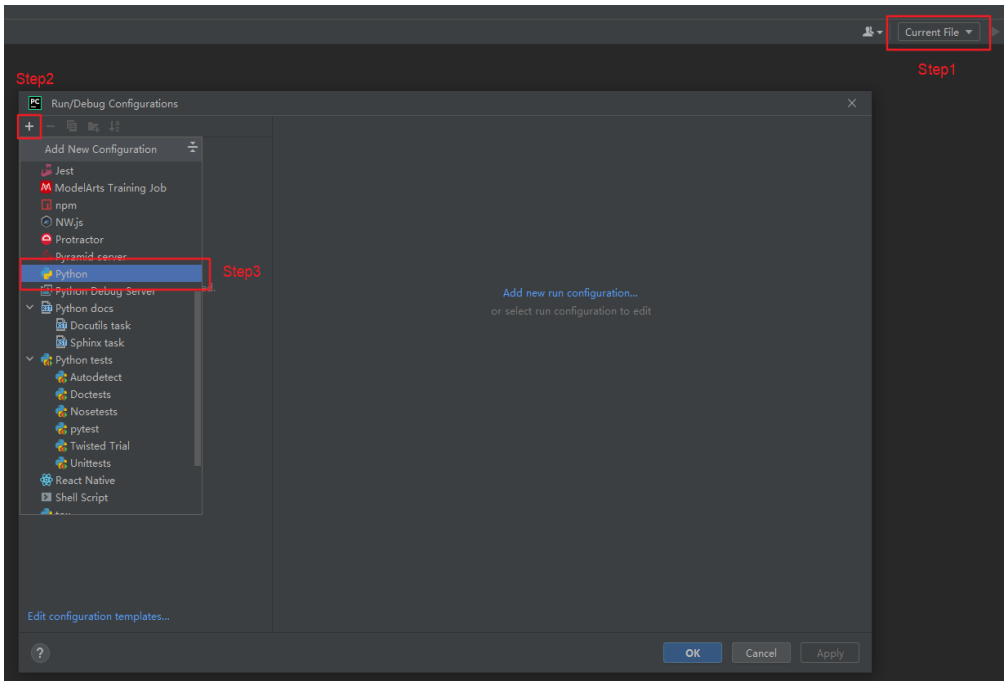
图 13-5 准备数据集



4. 配置PyCharm解释器和入参

   单击右上角“Current File”，选择“Edit Configuration”，打开“Run/Debug Configuration”对话框。在对话框中单击“+”，选择“Python”。

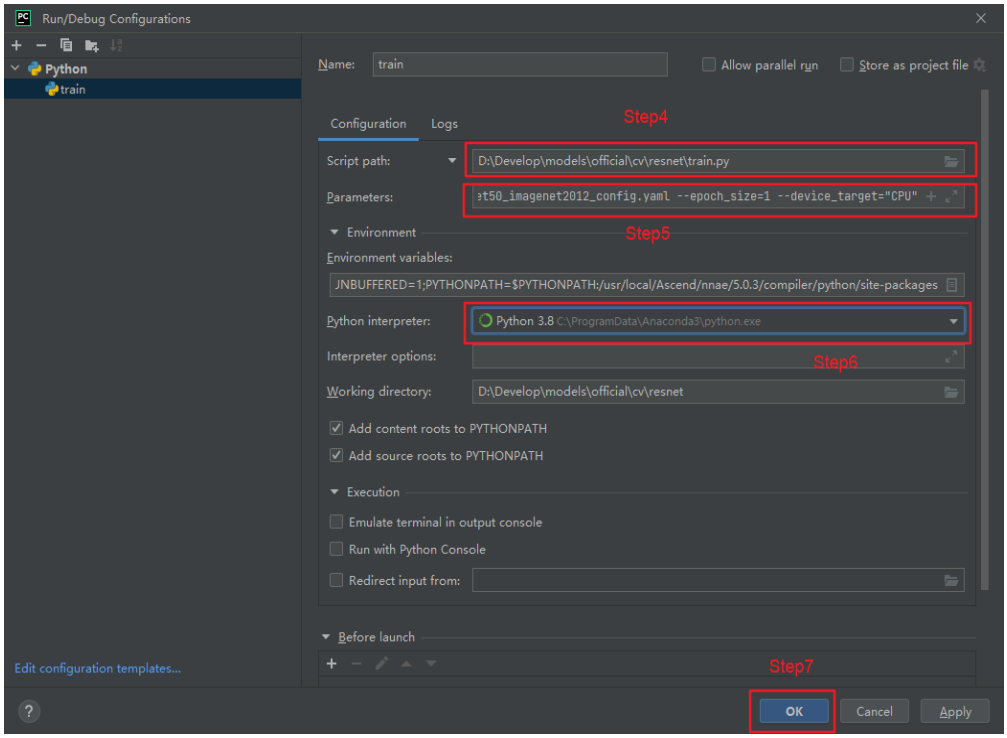
图 13-6 前往 PyCharm 解释器



“Script path”选择train.py文件，“Parameters”命令如下所示，并选择Python解释器，然后单击“OK”：

```
--net_name=resnet50 --dataset=imagenet2012 --data_path=../../../dataset/flower_photos/ --class_num=5 --config_path=./config/resnet50_imagenet2012_config.yaml --epoch_size=1 --device_target="CPU"
```

图 13-7 配置 PyCharm 解释器



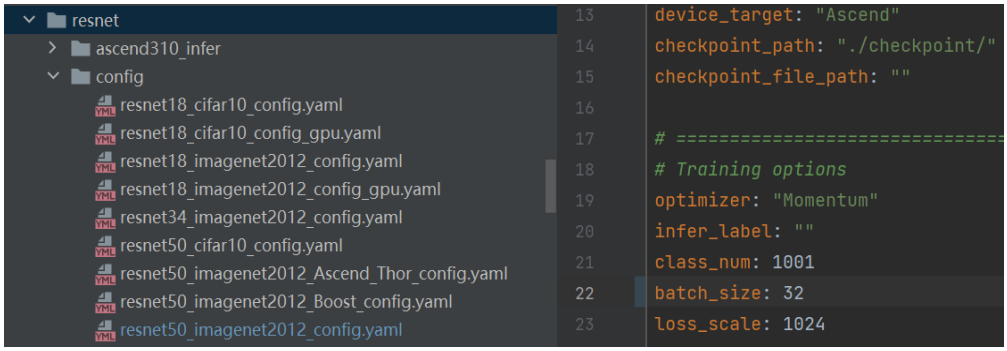
说明

根据README说明文档，配置Parameter参数device_target="CPU"表示CPU环境运行，device_target="Ascend"表示在Ascend环境运行。

5. 本地代码开发调测

一般本地CPU算力较低并且内存较小，可能出现内存溢出的报错，因此可以把“models/official/cv/resnet/config/resnet50_imagenet2012_config.yaml”的“batch_size”由“256”改为“32”，使得训练作业可以快速运行。

图 13-8 修改 batch_size



AI开发过程中的数据集开发及模型开发是和硬件规格无关的，而且这一部分的开发耗时是最长的，因此可以先在本地PC的CPU环境进行数据集和模型开发调试。

说明

本例中，因为样例代码已经支持在CPU上进行训练，因此用户能够在CPU上完成整个训练流程。如果代码只支持在GPU或者Ascend上训练，那么可能会报错，需要使用Notebook进行调试。

设置断点后单击“调试”，可实现代码逐步调试，查看中间变量值。

图 13-9 “调试”按钮

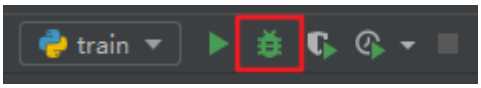
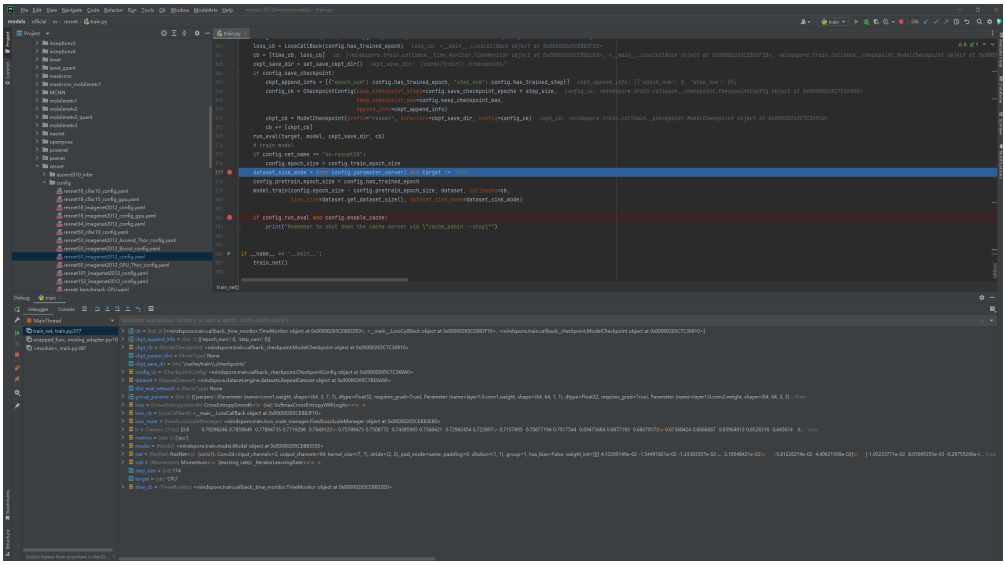


图 13-10 通过设置断点实现代码调试



可单击“运行”按钮，通过日志观察是否能正常训练。

图 13-11 “运行”按钮

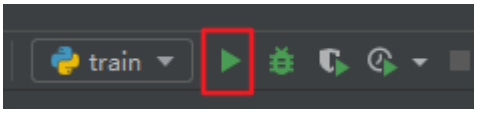
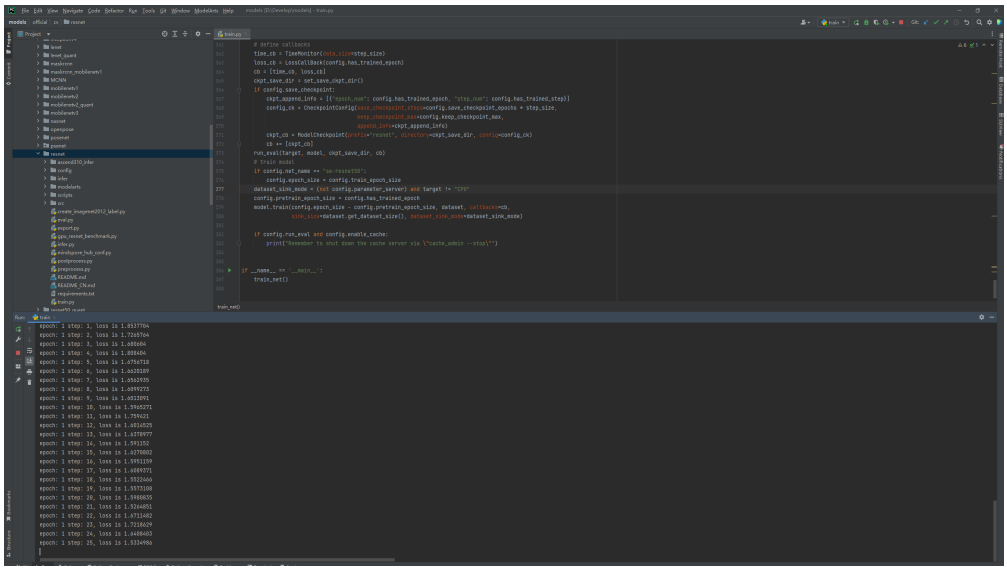


图 13-12 训练日志



步骤 3：使用 ModelArts Notebook 进行开发调试

使用ModelArts Notebook进行开发调试具有如下优势：

- 环境保持一致
- 配置一键完成
- 代码远程Debug
- 资源按需使用

说明

只有PyCharm专业版支持本章节，社区版可以直接跳转至**步骤4：使用PyCharm提交训练作业至ModelArts**完成创建训练作业。

1. 连接Notebook开发环境

   a. 创建或打开Ascend规格的Notebook。创建Notebook详细操作请参见**创建Notebook实例**，Notebook规格相关信息如下所示：

   “镜像”：mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3。

   “资源选择”：公共资源池。

   “类型”：ASCEND。

   “规格”：选Ascend类型的，以界面实际可选值为准。

   “存储配置”：EVS存储。

   “SSH远程开发”：开启。

   “密钥对”：选择已有密钥对，或单击密钥对右侧的“立即创建”创建密钥对。

2. 通过ToolKit连接Notebook

   a. 在IDE菜单栏中选择“ModelArts>Notebook>Remote Config”，在打开的界面中选择要连接的Notebook实例。

说明

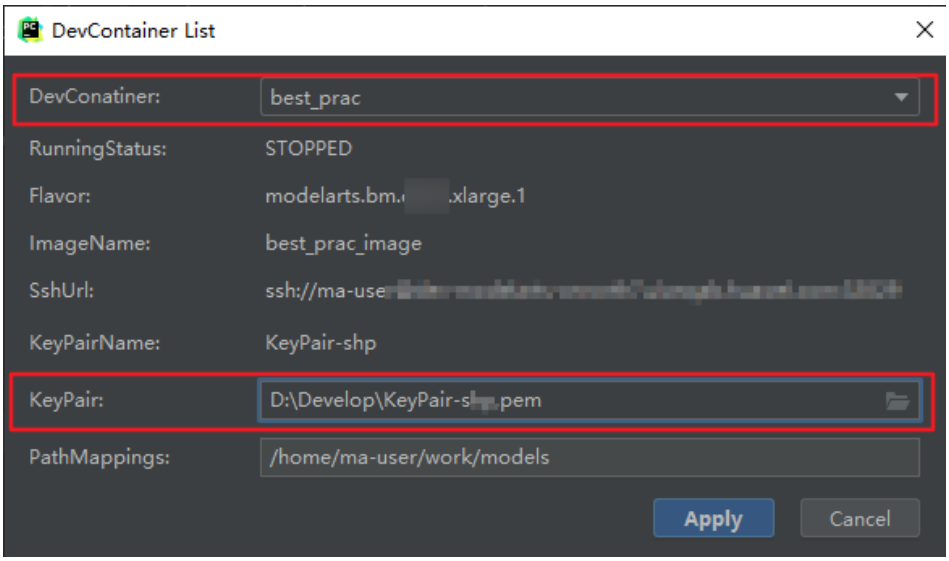
此处如果看不到Connect to Remote选项，请先参考**创建Notebook实例**章节，创建Notebook实例，并开启该实例的SSH远程开发功能。

也可能是PyCharm ToolKit的版本不正确，请按照文档要求下载新版本的PyCharm ToolKit。

下载前请先清除浏览器缓存，如果之前下载过老版本的PyCharm ToolKit，浏览器会有缓存，可能会导致新版本下载失败。

b. 在KeyPair中选择该Notebook实例对应的密钥，选择完成后，单击Apply进行远程Notebook一键配置，等待一段时间后，会出现重启IDE的确认框，单击确认重启，重启后即可生效。

图 13-13 ToolKit 连接 Notebook 配置界面



说明

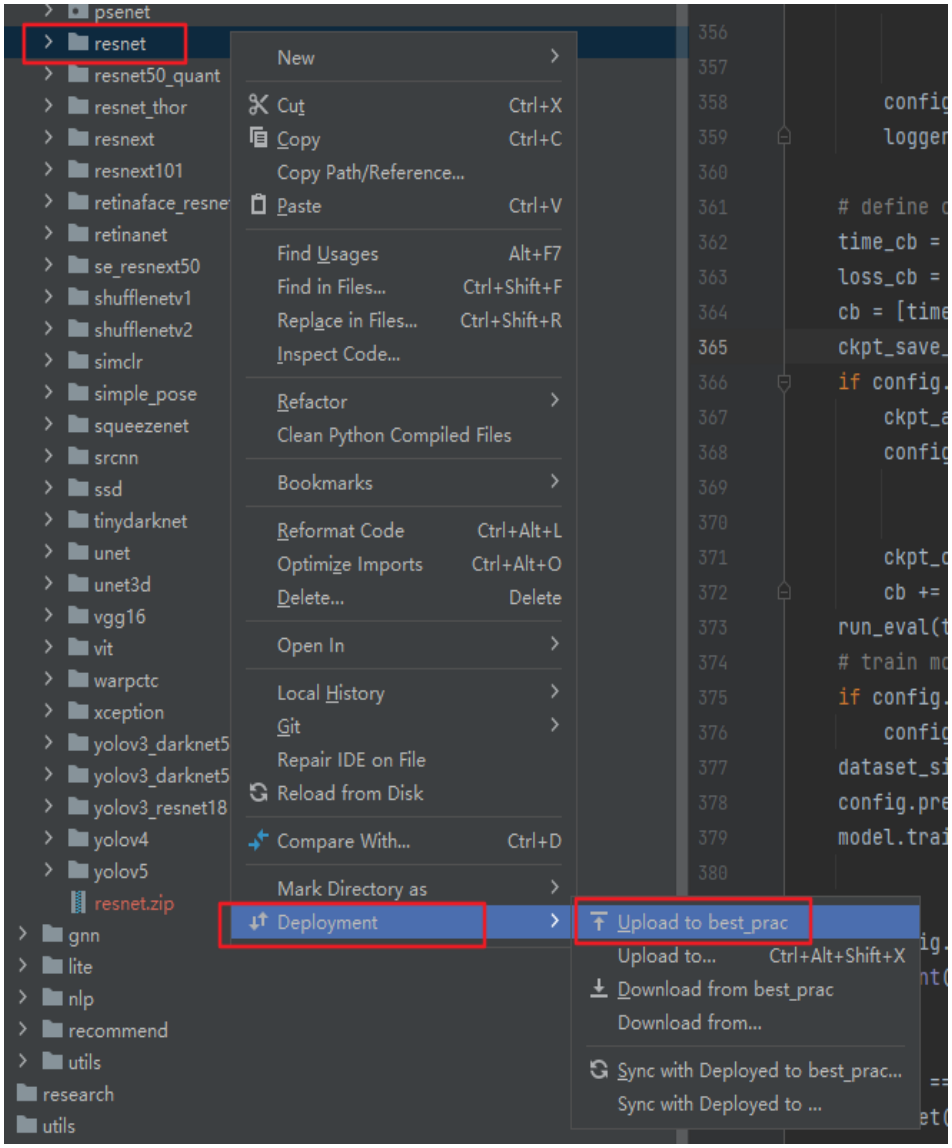
- KeyPair: 需要选择保存在本地的Notebook对应的keypair认证。即创建Notebook时创建的密钥对文件，创建时会直接保存到浏览器默认的下载文件夹中。
- PathMappings: 该参数为本地IDE项目和Notebook对应的同步目录，默认为“/home/ma-user/work/project”，可根据自己实际情况更改。

3. 同步代码和数据至Notebook

a. 将代码同步至Notebook

选择resnet文件夹，右键选择“Deployment>Upload to”上传代码至Notebook。

图 13-14 同步代码至 Notebook



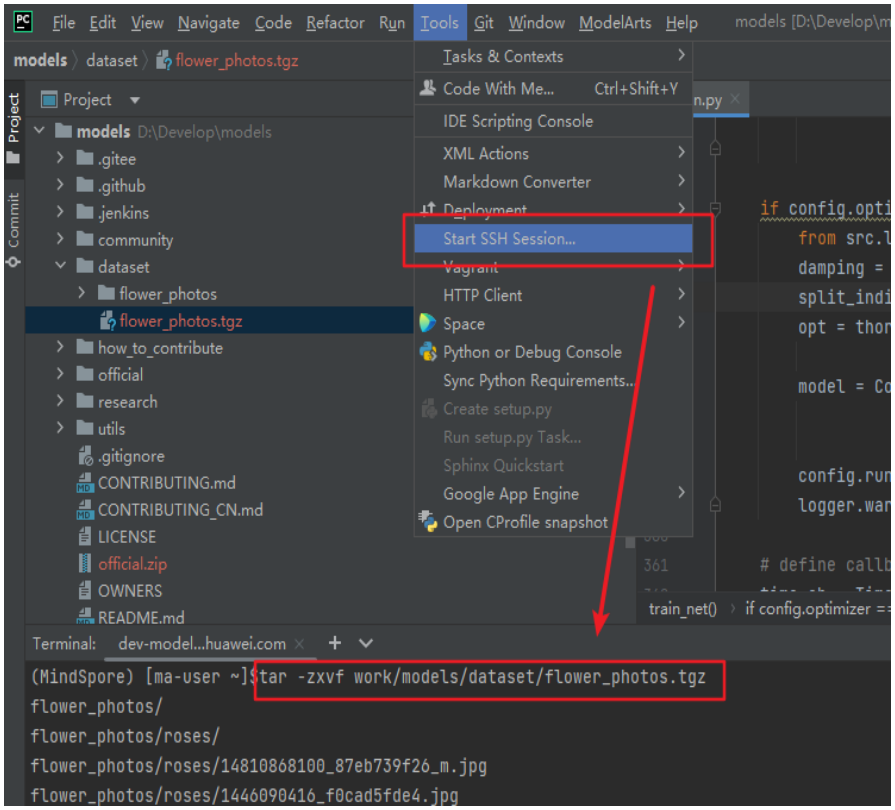
b. 将数据同步至Notebook

- （推荐）方法1：数据集压缩包上传至Notebook后解压

  把数据集压缩包右键选择“Deployment>Upload to”的方式上传至Notebook后，在Notebook中对数据集进行解压操作，解压命令如下：

  ```
  tar -zxvf work/models/dataset/flower_photos.tgz
  ```

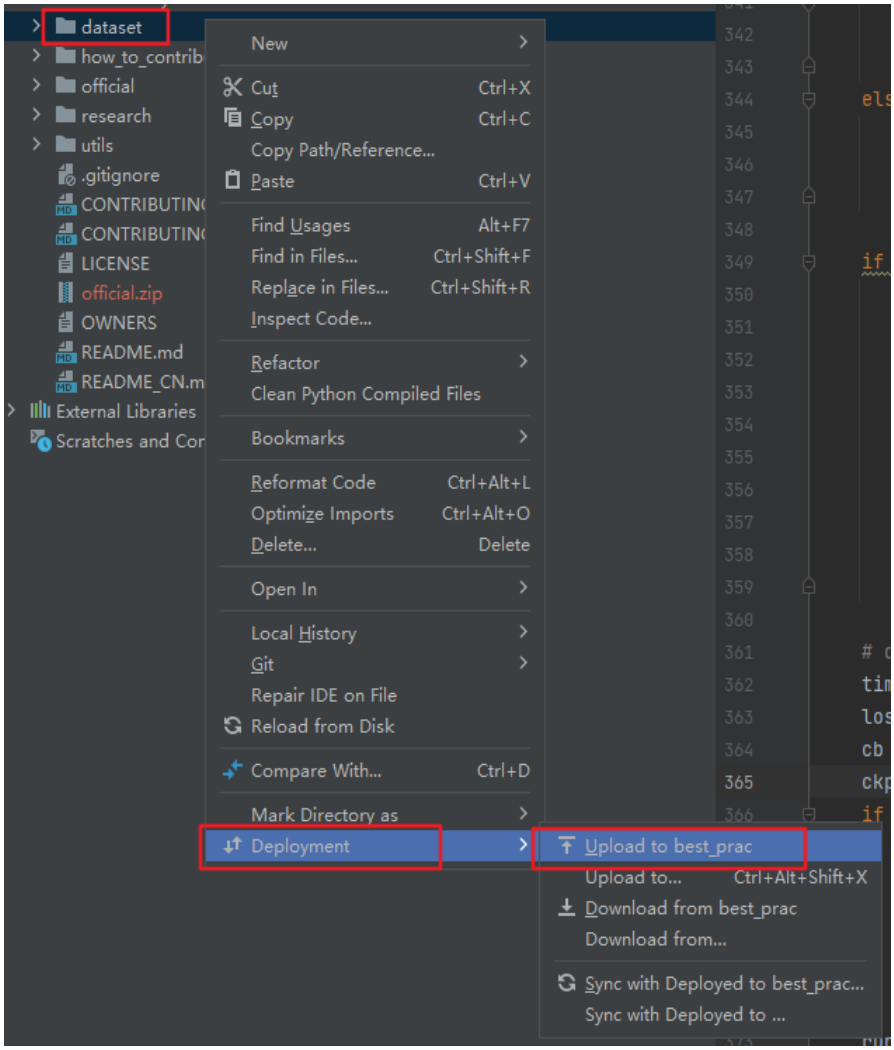
图 13-15 数据集压缩包上传至 Notebook 后解压



- 方法2：文件夹直接上传至Notebook

  类似上传代码至Notebook，直接上传数据文件夹。（由于本案例数据集中图片数量较多，通过IDE进行上传比较耗时，推荐使用方法1进行上传）

图 13-16 文件夹直接上传至 Notebook



注意

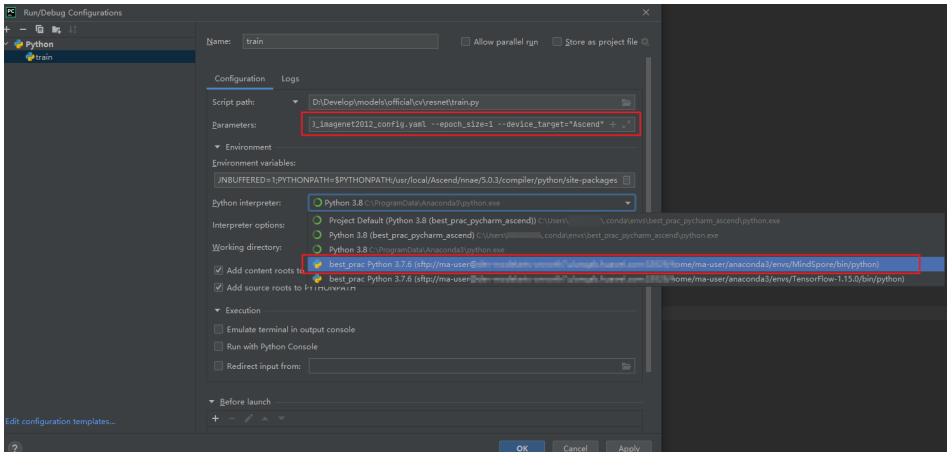
- 当数据集比较大达到数GB时，建议先将数据集先上传至OBS再通过OBS上传至Notebook，PyCharm只适合做小文件的同步上传。
- 调试时建议使用较小的数据集子集，方便数据同步与数据加载。

4. 配置Python解释器

   修改Parameters参数，并选择Python解释器。

```
--net_name=resnet50 --dataset=imagenet2012 --data_path=../../../dataset/flower_photos/ --class_num=5 --config_path=./config/resnet50_imagenet2012_config.yaml --epoch_size=1 --device_target="Ascend"
```

图 13-17 配置 python 解释器

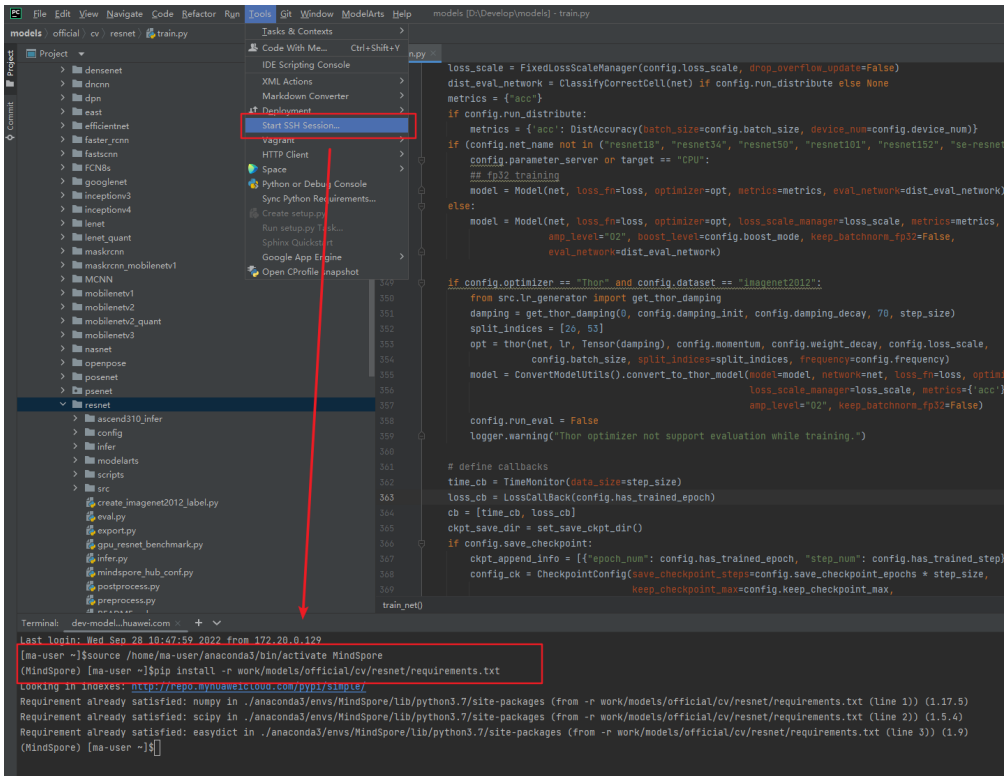


5. Notebook安装依赖

   打开“Tool>Start SSH Section”，安装依赖软件。

```
# 进入MindSpore环境
source /home/ma-user/anaconda3/bin/activate MindSpore
# 安装resnet依赖
pip install -r work/models/official/cv/resnet/requirements.txt
```

图 13-18 Notebook 安装依赖



6. 调试与运行

   配置完解释器后，PyCharm可以直接使用远端Notebook中的python解释器和硬件规格，满足用户在本地体验到真实的硬件环境并进行全流程的调试和验证。

注意

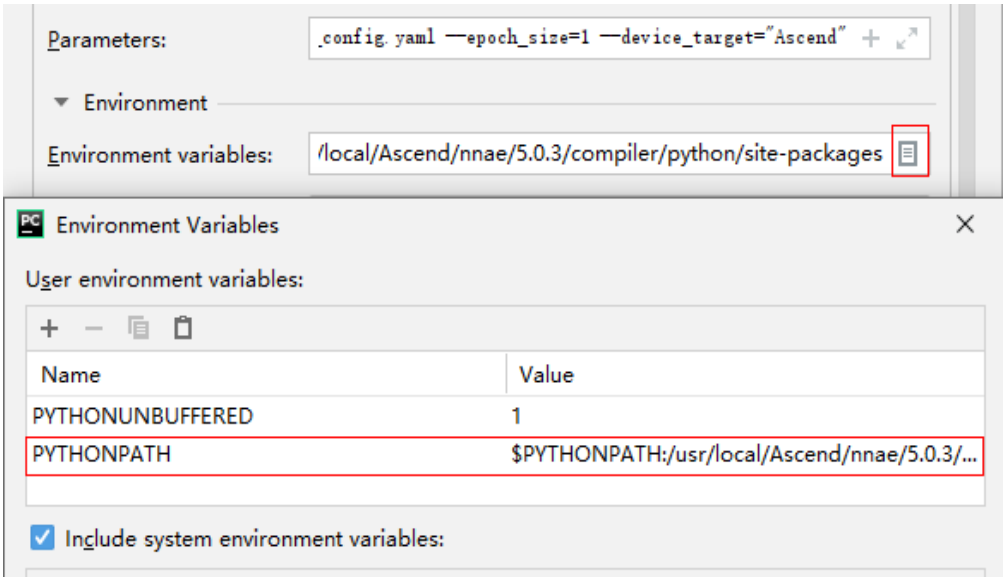
基于Ascend的样例中，可能会抛出异常。

```
ModuleNotFoundError: No module named 'te'
```

原因是：PyCharm的PYTHONPATH会将Notebook中的环境变量中指定的“PYTHONPATH”进行覆盖，因此，还需要将te包所在的路径添加到PyCharm的“PYTHONPATH”中。

te包的路径通过“pip show te”查看，例如te包返回对应的路径为：“/usr/local/Ascend/nnae/5.0.3/compiler/python/site-package”，则“PYTHONPATH”对应的“Value”为“$PYTHONPATH:/usr/local/Ascend/nnae/5.0.3/compiler/python/site-package”

图 13-19 将 te 包所在的路径添加到 PyCharm 的 PYTHONPATH 中



7. 保存开发环境镜像

   成功完成Notebook调测后，此时的Notebook已经包含了模型训练所有的依赖环境，因此可以将已经调测完成的开发环境保存成一个镜像，选择“Notebook>更多>保存镜像”。此时Notebook会冻结，需要等待几分钟（只需要保存一次）。

   保存后的镜像可以在“ModelArts>镜像管理”中进行查看，对应完整的镜像名称为“详情->SWR地址”。

图 13-20 查看保存后的镜像

说明

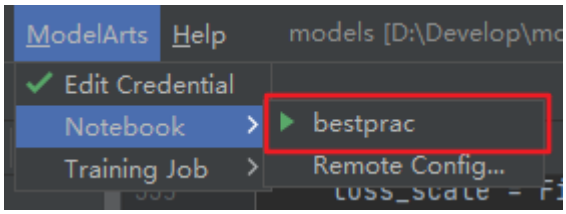
Notebook在代码调试完成及保存镜像后就可以关闭了，减少资源浪费。

8. 连接、停止、启动和断开Notebook实例

- 连接Notebook实例

  当Notebook实例为绿色三角形状态时，表示该实例运行中（但未与PyCharm连接）。此时单击该实例名称，实例会变为绿色勾状态，表示PyCharm已与实例连接成功。

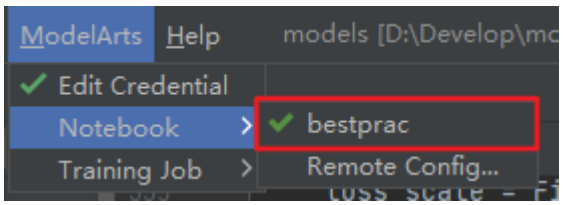
  图 13-21 实例运行中状态

- 停止Notebook实例

  当Notebook实例为绿色勾状态时，表示该实例运行中且与PyCharm连接成功。此时单击该实例名称，实例会变为黄色感叹号状态，表示停止Notebook实例。

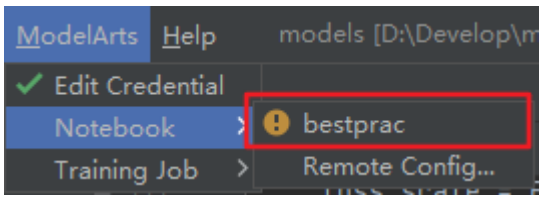
  图 13-22 实例运行中且与 PyCharm 连接成功状态

- 启动Notebook实例

  当Notebook实例为黄色感叹号状态时，表示该实例已停止。此时单击该实例名称，实例会变为绿色勾状态，表示启动Notebook实例且与PyCharm连接成功。（默认启动时间为4小时）

  图 13-23 实例已停止状态

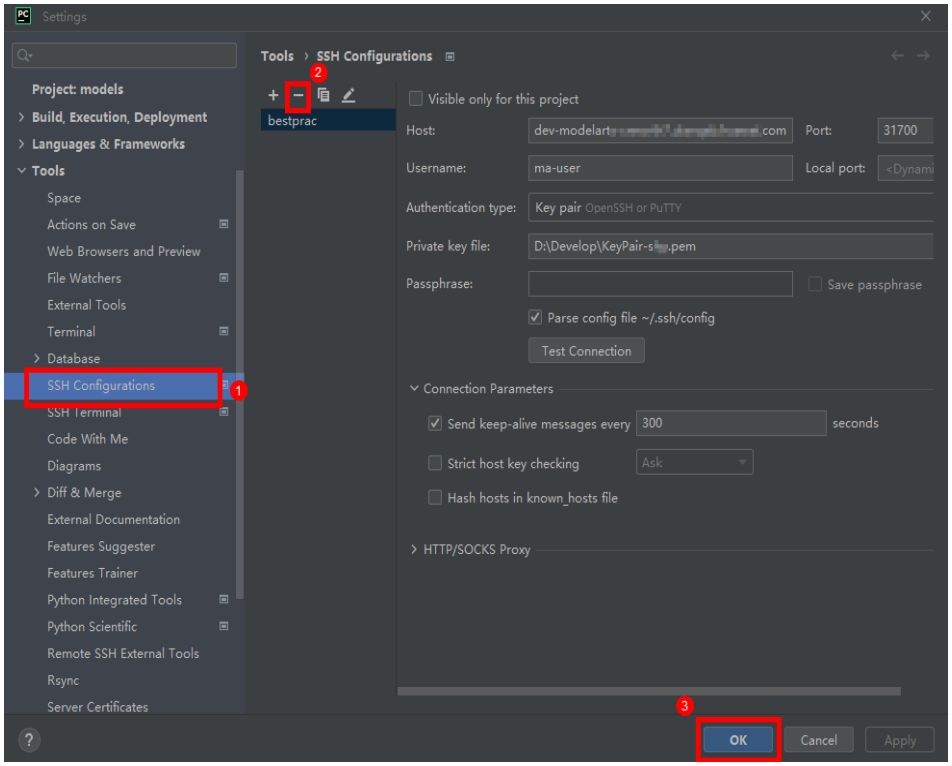
- 断开PyCharm ToolKit中的Notebook实例SSH连接

  选择“File>Settings>Tool>SSH Configurations”，单击需要断开的实例，选择“-”，单击“OK”，则IDE菜单栏“ModelArts>Notebook”中的Notebook实例连接断开。

⚠ 注意

该步骤会使Notebook实例不在PyCharm ToolKit中呈现，但Notebook实例仍然存在于控制台。如果想删除Notebook实例以释放资源，请登录ModelArts管理控制台，在Notebook管理页面进行删除。

图 13-24 断开 PyCharm ToolKit 中的 Notebook 实例 SSH 连接



步骤 4：使用 PyCharm 提交训练作业至 ModelArts

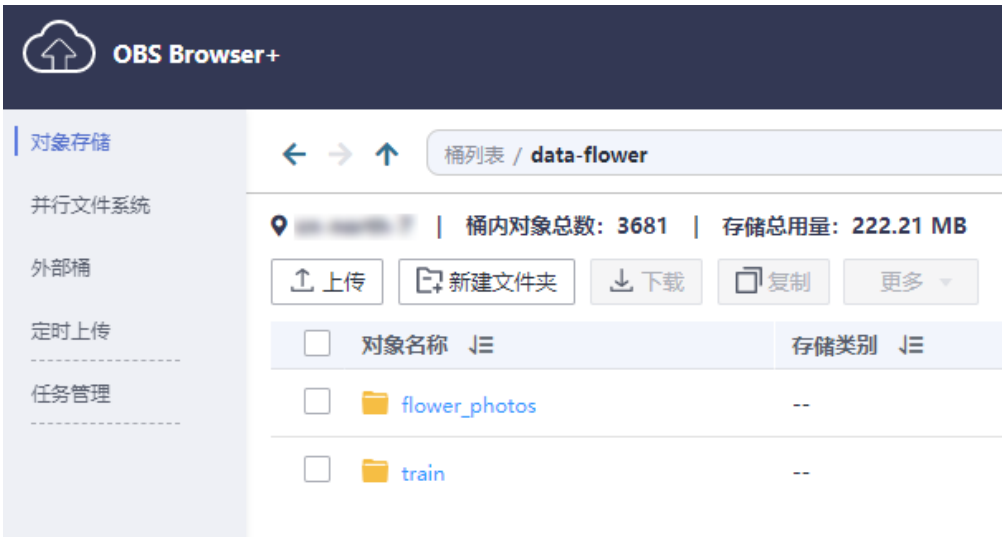
ModelArts训练平台提供了海量的算力规格和训练优化，支持将本地调试好的代码以及之前保存的开发环境镜像直接在PyCharm中提交训练作业。

1. 创建OBS桶并上传数据

   由于训练作业是在ModelArts端运行，因此需要把训练数据和训练代码上传至云端Notebook。可借助OBS Browser+把下载好的训练数据上传至OBS。

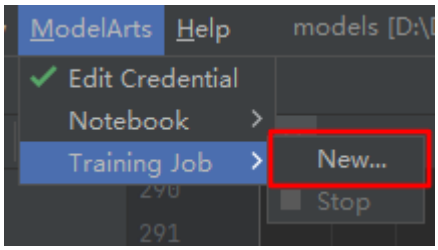
   新建data-flower桶，把训练数据flower_photos文件夹通过OBS Browser+上传至对应的OBS，并新建train文件夹用来存放训练作业相关数据。

图 13-25 上传数据至 OBS



2. 创建训练作业
在IDE菜单栏选择“ModelArts>Training Job>New...”创建训练作业。

图 13-26 创建训练作业



创建训练作业界面各参数名称及含义如下表所示。

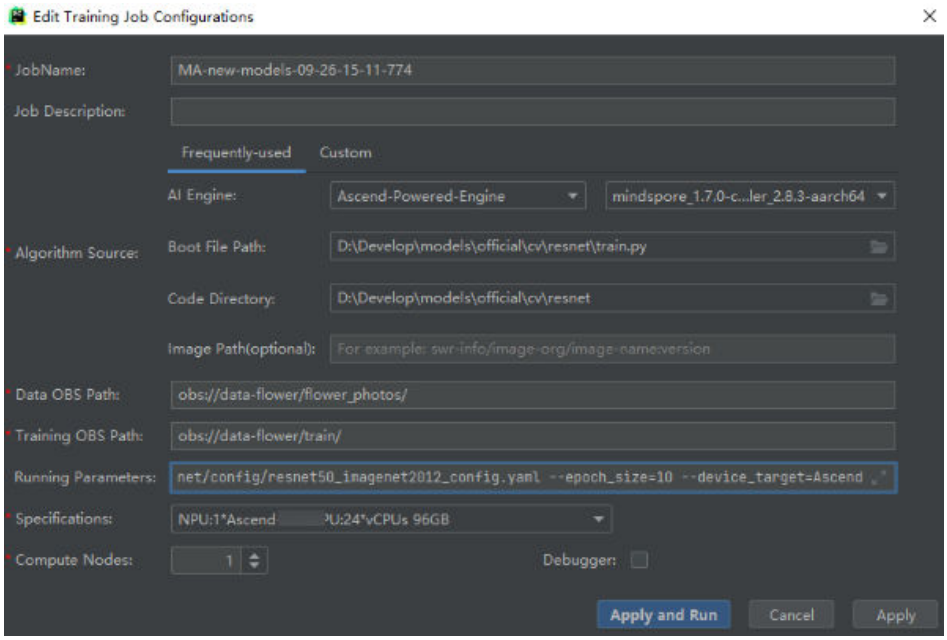
表 13-1 参数名称及含义

参数名称	含义
JobName	训练作业的名称，默认为当前的时间。
AI Engine	训练引擎，这里选择“mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64”
Boot File Path	本地训练启动代码。
Code Directory	本地代码目录
Image Path(optional)	可选项，输入自定义镜像swr路径地址（使用的自定义镜像和预置的训练镜像引擎一致）
Data OBS Path	OBS上的数据集路径（需要提前把数据上传到OBS中）
Training OBS Path	OBS路径（该路径必须是存在的），用于保存代码和训练模型及日志的输出
Running Parameters	训练脚本接收的参数。
Specifications	计算规格，这里选择Ascend类型的，以界面实际可选值为准。
Compute Node	节点数（单机训练默认为1）

PyCharm中支持两种方式创建训练作业：使用预置镜像训练作业、自定义镜像创建训练作业。

- 使用预置镜像创建训练作业
在RunningParameters中填入如下训练参数，其余参数按实际路径填写。
```
--net_name=resnet50 --dataset=imagenet2012 --enable_modelarts=True --class_num=5 --config_path=/home/ma-user/modelarts/user-job-dir/resnet/config/resnet50_imagenet2012_config.yaml --epoch_size=10 --device_target=Ascend
```
填写完训练作业参数后，单击“Apply and Run”即完成训练作业创建。

图 13-27 使用预置镜像创建训练作业



- 使用自定义镜像创建训练作业

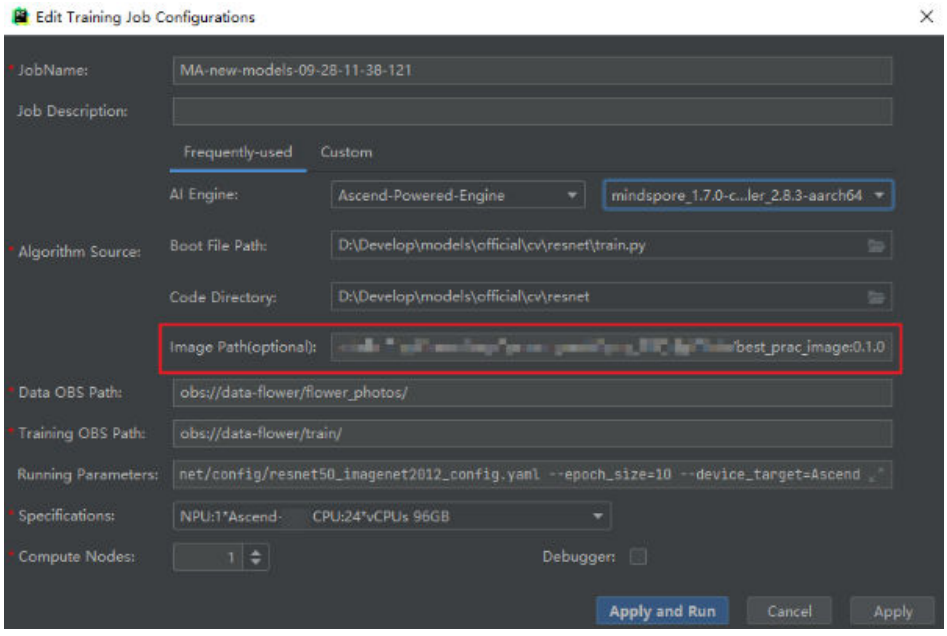
  使用自定义镜像创建训练作业和使用预置镜像创建训练作业的差别，在于Image Path处填入了自定义镜像的地址。填写完训练作业参数后，单击“Apply and Run”即完成训练作业创建。

  说明

  在选择AI Engine预置镜像时，需要和自定义镜像保持一致，该设置的作用为通过预置镜像的启动命令启动自定义镜像。

  例如自定义镜像中用到Mindspore，则预置镜像中可选择包含Mindspore的镜像。

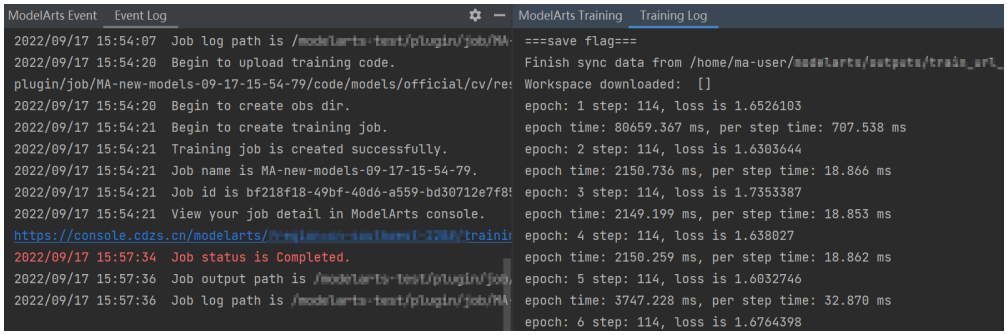
图 13-28 使用自定义镜像创建训练作业



3. 查看训练日志

在单击“Apply and Run”按钮后，训练的日志可以在PyCharm窗口中实时展示。也可以单击Event Log中的控制台链接，转调到网页端中查看训练日志。

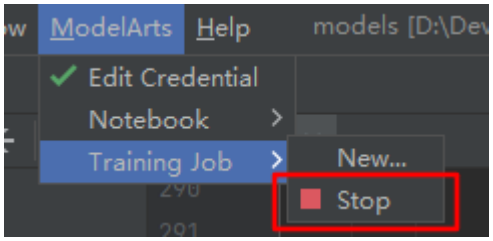
图 13-29 在 PyCharm 中查看训练日志



4. 终止训练作业

   如果想要在中途终止训练，可以在PyCharm中单击“ModelArts>Training Job>Stop”，或者直接在网页端单击终止。

图 13-30 终止训练作业



步骤 5：清除相应资源

在完成试用后，建议您删除相关资源，如在线服务、训练作业及其OBS目录。

- 停止Notebook：在“Notebook”页面，单击对应实例操作列的“停止”。
- 在PyCharm菜单栏中，选择“ModelArts > Stop Training Job”停止此训练作业。
- 进入OBS管理控制台，删除创建的OBS桶。先逐个删除桶内文件夹和文件，再执行删除桶的操作。

13.2 使用自定义引擎创建 AI 应用

使用自定义引擎创建AI应用，用户可以通过选择自己存储在SWR服务中的镜像作为AI应用的引擎，指定预先存储于OBS服务中的文件目录路径作为模型包，来创建AI应用，轻松地应对ModelArts平台预置引擎无法满足个性化诉求的场景。

ModelArts将自定义引擎类型的AI应用部署为服务时，会先将AI应用相关的SWR镜像下载至集群中，用“uid=1000, gid=100”的用户启动SWR镜像为容器，然后将OBS文件下载到容器中的“/home/mind/model”目录下，最后执行SWR镜像中预置的启动命令。ModelArts平台将暴露在容器“8080”端口的服务注册到APIG，用户可以通过提供的APIG（API网关）URL访问到该服务。

自定义引擎创建 AI 应用的规范

使用自定义引擎创建AI应用，用户的SWR镜像、OBS模型包和文件大小需要满足以下规范：

- SWR镜像规范：
  - 镜像必须内置一个用户名为“ma-user”，组名为“ma-group”的普通用户，且必须确保该用户的uid=1000、gid=100。内置用户的dockerfile指令如下：
    ```
    groupadd -g 100 ma-group && useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user
    ```
  - 明确设置镜像的启动命令。在dockerfile文件中指定cmd，dockerfile指令示例如下：
    ```
    CMD sh /home/mind/run.sh
    ```
    启动入口文件run.sh需要自定义。示例如下：
    ```
    #!/bin/bash

    # 自定义脚本内容
    ...

    # run.sh调用app.py启动服务器，app.py请参考https示例
    python app.py
    ```
  - 提供的服务必须使用https协议， 且暴露在“8080”端口。请参考**https示例**。
  - （可选）在“8080”端口，提供URL路径为“/health”的健康检查服务（健康检查的URL路径必须为 "/health"）。
- OBS模型包规范

  模型包的名字必须为model。模型包规范请参见**模型包规范介绍**。
- 文件大小规范

  当使用公共资源池时，SWR的镜像大小（指下载后的镜像大小，非SWR界面显示的压缩后的镜像大小）和OBS模型包大小总和不大于30G。

https 示例

使用Flask启动https，Webserver代码示例如下：

```
from flask import Flask, request
import json

app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
    print("---------- in hello func ----------")
    data = json.loads(request.get_data(as_text=True))
    print(data)
    username = data['name']
    rsp_msg = 'Hello, {}!'.format(username)
    return json.dumps({"response":rsp_msg}, indent=4)

@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
    print("---------- in goodbye func ----------")
    return '\nGoodbye!\n'


@app.route('/', methods=['POST'])
def default_func():
```

```
    print("---------- in default func ----------")
    data = json.loads(request.get_data(as_text=True))
    return '\n called default func !\n {} \n'.format(str(data))

@app.route('/health', methods=['GET'])
def healthy():
    return "{\"status\": \"OK\"}"

# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080, ssl_context='adhoc')
```

在本地机器调试

自定义引擎的规范可以在安装有docker的本地机器上通过以下步骤提前验证：

1. 将自定义引擎镜像下载至本地机器，假设镜像名为custom_engine:v1。
2. 将模型包文件夹复制到本地机器，假设模型包文件夹名字为model。
3. 在模型包文件夹的同级目录下验证如下命令拉起服务：
   ```
   docker run --user 1000:100 -p 8080:8080 -v model:/home/mind/model    custom_engine:v1
   ```

   说明

   该指令无法完全模拟线上，主要是由于-v挂载进去的目录是root权限。在线上，模型文件从OBS下载到/home/mind/model目录之后，文件owner将统一修改为ma-user。
4. 在本地机器上启动另一个终端，执行以下验证指令，得到符合预期的推理结果。
   ```
   curl https://127.0.0.1:8080/${推理服务的请求路径}
   ```

推理部署示例

本节将详细说明以自定义引擎方式创建AI应用的步骤。

1. 创建AI应用并查看AI应用详情

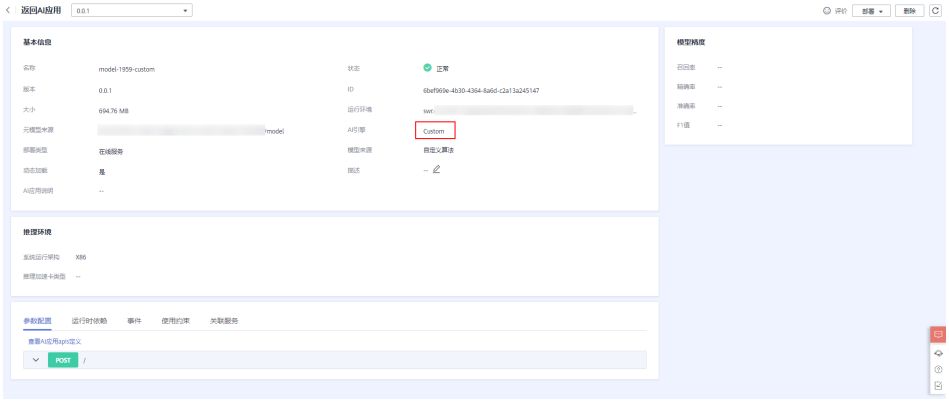
   登录ModelArts管理控制台，进入“AI应用管理>AI应用”页面中，单击“创建AI应用”，进入AI应用创建页面，设置相关参数如下：
   - 元模型来源：选择“从对象存储服务（OBS）中选择”。
   - 选择元模型：从OBS中选择一个模型包。
   - AI引擎：选择“Custom”。
   - 引擎包：从容器镜像中选择一个镜像。

     其他参数保持默认值。

   单击“立即创建”，跳转到AI应用列表页，查看AI应用状态，当状态变为“正常”，AI应用创建成功。

   单击AI应用名称，进入AI应用详情页面，查看AI应用详情信息。

图 13-31 查看 AI 应用详情



2. 部署服务并查看详情

   在AI应用详情页面，单击右上角“部署>在线服务”，进入服务部署页面，AI应用和版本默认选中，选择合适的“计算节点规格”（例如CPU：2核 8GB），其他参数可保持默认值，单击“下一步”，跳转至服务列表页，当服务状态变为“运行中”，服务部署成功。

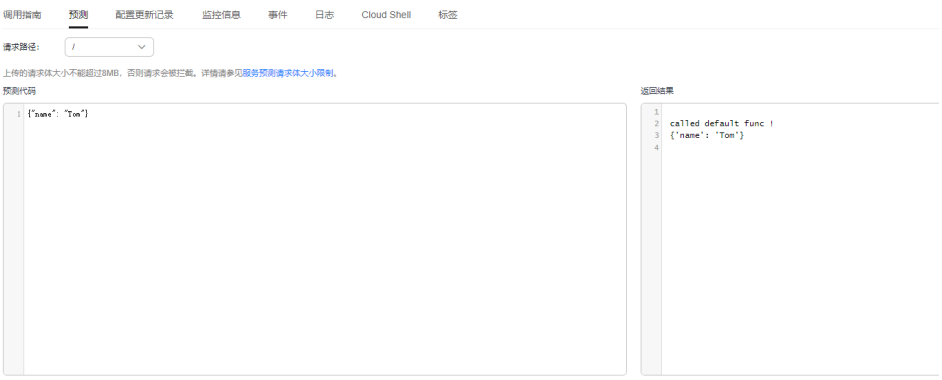
   单击服务名称，进入服务详情页面，查看服务详情信息，单击“日志”页签，查看服务日志信息。

图 13-32 查看服务日志信息

3. 服务预测

   在服务详情页面，单击“预测”页签，进行服务预测。

图 13-33 服务预测



13.3 使用大模型创建 AI 应用部署在线服务

背景说明

目前大模型的参数量已经达到千亿甚至万亿，随之大模型的体积也越来越大。千亿参数大模型的体积超过200G，在版本管理、生产部署上对平台系统产生了新的要求。例如：导入AI应用管理时，需要支持动态调整租户存储配额；模型加载、启动慢，部署时需要灵活的超时配置；当负载异常重启，模型需要重新加载，服务恢复时间长的问题亟待解决。

为了应对如上诉求，ModelArts推理平台针对性给出解决方案，用于支持大模型场景下的AI应用管理和服务部署。

约束与限制

- 需要申请单个AI应用大小配额和添加使用节点本地存储缓存的白名单
- 需要使用自定义引擎Custom，配置动态加载
- 需要使用专属资源池部署服务
- 专属资源池磁盘空间需大于1T

操作事项

1. **申请扩大AI应用的大小配额和使用节点本地存储缓存白名单**
2. **上传模型数据并校验上传对象的一致性**
3. **创建专属资源池**
4. **创建AI应用**
5. **部署在线服务**

申请扩大 AI 应用的大小配额和使用节点本地存储缓存白名单

服务部署时，默认情况下，动态加载的模型包位于临时磁盘空间，服务停止时已加载的文件会被删除，再次启动时需要重新加载。为了避免反复加载，平台允许使用资源池节点的本地存储空间来加载模型包，并在服务停止和重启时仍有效（通过哈希值保证数据一致性）

使用大模型要求用户采用自定义引擎，并开启动态加载的模式导入模型。基于此，需要执行以下操作：

- 如果模型超过默认配额值，需要提工单申请扩大单个AI应用的大小配额。单个AI应用大小配额默认值为20GB。
- 需要提工单申请添加使用节点本地存储缓存的白名单。

上传模型数据并校验上传对象的一致性

为了动态加载时保证数据完整性，需要在上传模型数据至OBS时，进行上传对象的一致性校验。obsutil、OBS Browser+以及OBS SDK都支持在上传对象时进行一致性校验，您可以根据自己的业务选择任意一种方式进行校验。详见对象存储服务 OBS帮助文档《校验上传对象的一致性》章节。

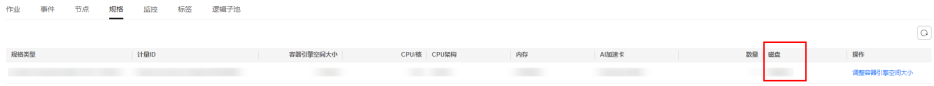
以OBS Browser+为例，如图13-34。使用OBS Browser+上传数据，开启MD5校验，动态加载并使用节点本地的持久化存储时，检查数据一致性。

图 13-34 OBS Browser+配置 MD5 校验

创建专属资源池

使用本地的持久化存储功能，需使用专属资源池，且专属资源池磁盘空间大小必须超过1T。您可以通过专属资源池详情页面，规格页签，查看专属资源池磁盘信息。当服务部署失败，提示磁盘空间不足时，请参考**服务部署、启动、升级和修改时，资源不足如何处理?**

图 13-35 查看专属资源池磁盘信息



创建 AI 应用

使用大模型创建AI应用，选择从对象存储服务（OBS）中导入，需满足以下参数配置：

1. 采用自定义引擎，开启动态加载

   使用大模型要求用户使用自定义引擎，并开启动态加载的模式导入模型。用户可以制作自定义引擎，满足大模型场景下对镜像依赖包、推理框架等的特殊需求。自定义引擎的制作请参考**14.2-使用自定义引擎创建AI应用**。

   当用户使用自定义引擎时，默认开启动态加载，模型包与镜像分离，在服务部署时动态将模型加载到服务负载。

2. 配置健康检查

   大模型场景下导入的AI应用，要求配置健康检查，避免在部署时服务显示已启动但实际不可用。

图 13-36 采用自定义引擎，开启动态加载并配置健康检查示例图

部署在线服务

部署服务时，需满足以下参数配置：

1. 自定义部署超时时间

   大模型加载启动的时间一般大于普通的模型创建的服务，请配置合理的“部署超时时间”，避免尚未启动完成被认为超时而导致部署失败。

2. 添加环境变量

   部署服务时，增加如下环境变量，会将负载均衡的请求亲和策略配置为集群亲和，避免未就绪的服务实例影响预测成功率。

   ```
   MODELARTS_SERVICE_TRAFFIC_POLICY: cluster
   ```

图 13-37 自定义部署超时时间和添加环境变量示例图

建议部署多实例，增加服务可靠性。

13.4 从 OBS 导入模型创建 AI 应用部署在线服务

本案例介绍如何将制作好的模型包上传至对象存储服务（OBS）平台，从OBS导入模型创建为AI应用，将AI应用部署为在线服务，最后进行服务预测。

开始使用样例前，请仔细阅读**准备工作**罗列的要求，提前完成准备工作。本案例的步骤如下所示：

步骤1：将模型包上传至OBS

步骤2：从OBS导入模型创建为AI应用

步骤3：将AI应用部署为在线服务

步骤4：在线服务预测

准备工作

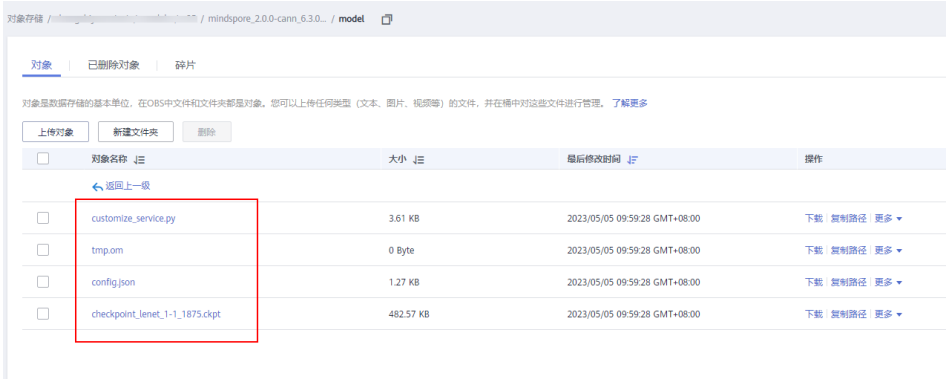
根据**模型包规范介绍**，准备所需文件，模型包里面必需包含“model”文件夹，“model”文件夹下面放置模型文件，模型配置文件，模型推理代码文件。

步骤 1：将模型包上传至 OBS

1. 登录对象存储服务（OBS）管理控制台。
2. 创建桶及放置模型包的目录，单击“上传对象”，将**准备工作**制作的模型包上传至OBS桶目录下。例如MindSpore模型包结构如下：

```
OBS桶/目录名
├── resnet
|   ├── model 必选： 固定子目录名称，用于放置模型相关文件
|   |   ├── <<自定义python包>> 可选：用户自有的Python包，在模型推理代码中可以直接引用
|   |   ├── checkpoint_lenet_1-1_1875.ckpt 必选；使用mindspore训练出来的ckpt格式的模型文件
|   |   ├── config.json 必选：模型配置文件，文件名称固定为config.json, 只允许放置一个
|   |   ├── customize_service.py 必选：模型推理代码，文件名称固定为customize_service.py, 只允许放置一个，customize_service.py依赖的文件可以直接放model目录下
|   |   ├── tmp.om 必选；空的.om文件，保证模型包可以正常导入
```

图 13-38 上传模型包



步骤 2：从 OBS 导入模型创建为 AI 应用

1. 登录ModelArts管理控制台，选择“AI应用管理 > AI应用”，在“我的AI应用”页签单击“创建”，进入AI应用创建页面。
2. 元模型来源选择“从对象存储服务（OBS）中选择”，选择**步骤1：将模型包上传至OBS**上传的模型包导入。“AI引擎”、“运行环境”及“运行时依赖”将会自动识别。示例如下：

图 13-39 从 obs 导入模型

3. 完成其他配置项后，单击“立即创建”，下发创建AI应用任务。
4. 在AI应用列表页，查看AI应用状态，当“状态”变为“正常”，AI应用创建成功。

步骤 3：将 AI 应用部署为在线服务

1. 选择**步骤2：从OBS导入模型创建为AI应用**创建的AI应用，单击列表前面的小三角符号，打开AI应用版本详情，操作列选择“部署 > 在线服务”，跳转至在线服务部署页面。

图 13-40 部署在线服务

2. 在线服务部署页面，AI应用及版本号将会自动识别。建议开启自动停止功能，在线服务将在您选择的时间点后自动停止。按需完成其他配置，完成后单击“立即创建”，下发在线服务部署任务。示例如下：

图 13-41 AI 应用及配置

3. 查询在线服务列表页服务状态，当“状态”变为“运行中”，在线服务部署成功。

步骤 4：在线服务预测

1. 单击**步骤3：将AI应用部署为在线服务**部署的在线服务的名称，进入服务详情页面。
2. 单击“预测”页签，进行预测。预测示例如下：

图 13-42 在线服务预测

步骤 5：清理资源

完成预测后，建议您清理相关资源，如在线服务和OBS目录。

- 停止或删除在线服务：在“在线服务”页面，单击对应服务操作列的“停止”或“删除”。
- 进入OBS管理控制台，删除创建的OBS桶。先逐个删除桶内文件夹和文件，再执行删除桶的操作。

13.5 WebSocket 在线服务全流程开发

背景说明

WebSocket是一种网络传输协议，可在单个TCP连接上进行全双工通信，位于OSI模型的应用层。WebSocket协议在2011年由IETF标准化为RFC 6455，后由RFC 7936补充规范。Web IDL中的WebSocket API由W3C标准化。

WebSocket使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在WebSocket API中，浏览器和服务器只需要完成一次握手，两者之间就可以建立持久性的连接，并进行双向数据传输。

前提条件

- 用户需有一定的Java开发经验，熟悉jar打包流程。
- 用户需了解WebSocket协议的基本概念及调用方法。
- 用户需熟悉Docker制作镜像的方法。

约束与限制

- WebSocket协议只支持部署在线服务。

- 只支持自定义镜像导入AI应用部署的在线服务。

准备工作

ModelArts使用WebSocket完成推理需要用户自己准备自定义镜像，该自定义镜像需要在单机环境下能够提供完整的WebSocket服务，如完成WebSocket的握手，client向server发送数据，server向client发送数据等。模型的推理过程在自定义镜像中完成，如下载模型，加载模型，执行预处理，完成推理，拼装响应体等。

操作步骤

WebSocket在线服务开发操作步骤如下：

- **上传镜像至容器镜像服务**
- **使用镜像创建AI应用**
- **使用AI应用部署在线服务**
- **WebSocket在线服务调用**

上传镜像至容器镜像服务

将准备好的本地镜像上传到容器镜像服务（SWR）。上传镜像的详细操作可参考**如何登录并上传镜像到SWR**。

使用镜像创建 AI 应用

1. 登录ModelArts管理控制台，进入“AI应用管理 > AI应用”页面，单击“创建”，跳转至创建AI应用页面。
2. 完成AI应用配置，部分配置如下：
   - 元模型来源：选择“从容器镜像中选择”。
   - 容器镜像所在的路径：选择**上传镜像至容器镜像服务**上传的路径。
   - 容器调用接口：根据实际情况配置容器调用接口。
   - 健康检查：保持默认。如果镜像中配置了健康检查则按实际情况配置健康检查。

   图 13-43 AI 应用配置参数

3. 单击“立即创建”，进入AI应用列表页，等AI应用状态变为“正常”，表示AI应用创建成功。

使用 AI 应用部署在线服务

1. 登录ModelArts管理控制台，进入“部署上线 >在线服务”页面，单击“部署”，跳转至在线服务部署页面。
2. 完成服务的配置，部分配置如下：
   - 选择AI应用及版本：选择**使用镜像创建AI应用**创建完成的AI应用及版本
   - 升级为WebSocket：打开开关

   图 13-44 升级为 WebSocket

3. 单击“下一步”，确认配置后“提交”，完成在线服务的部署。返回在线服务列表页，查看服务状态变为“运行中”，表示服务部署成功。

WebSocket 在线服务调用

WebSocket协议本身不提供额外的认证方式。不管自定义镜像里面是ws还是wss，经过ModelArts平台出去的WebSocket协议都是wss的。同时wss只支持客户端对服务端的单向认证，不支持服务端对客户端的双向认证。

可以使用ModelArts提供的以下认证方式：

- **token认证**

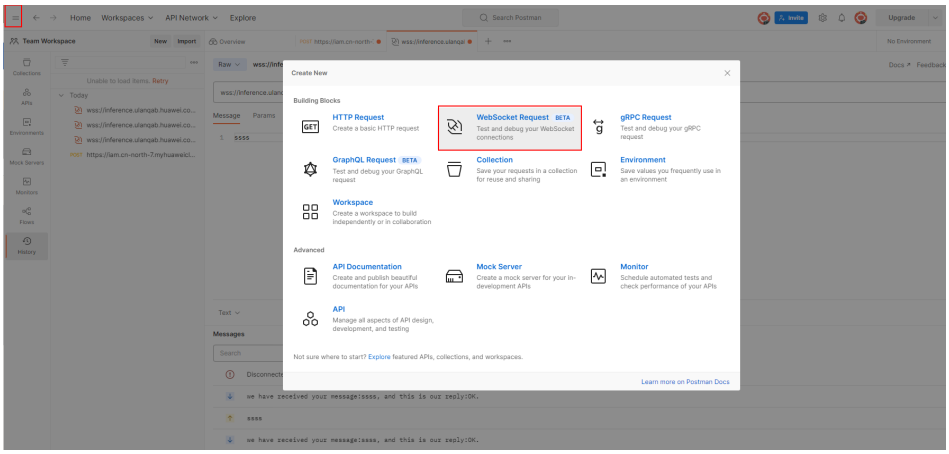
WebSocket服务调用步骤如下（以图形界面的软件Postman进行预测，token认证为例）：

1. **WebSocket连接的建立**
2. **WebSocket客户端和服务端双向传输数据**

步骤1 WebSocket连接的建立

1. 打开Postman（需选择8.5 以上版本，以10.12.0为例）工具，单击左上角，选择“File>New”，弹出新建对话框，选择“WebSocket Request”（当前为beta版本）功能：

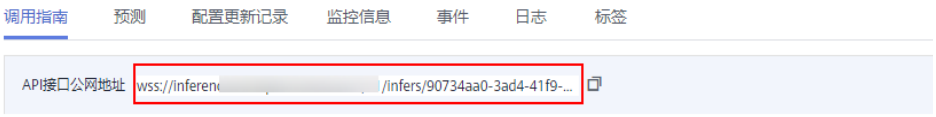
图 13-45 选择 WebSocket Request 功能



2. 在新建的窗口中填入WebSocket连接信息：

   左上角选择Raw，不要选择Socket.IO（一种WebSocket实现，要求客户端跟服务端都要基于Socket.IO），地址栏中填入从服务详情页“调用指南”页签中获取“API接口调用公网地址”后面的地址。如果自定义镜像中有更细粒度的地址，则在地址后面追加该URL。如果有queryString，那么在params栏中添加参数。在header中添加认证信息（不同认证方式有不同header，跟https的推理服务相同）。选择单击右上的connect按钮，建立WebSocket连接。

图 13-46 获取 API 接口调用公网地址

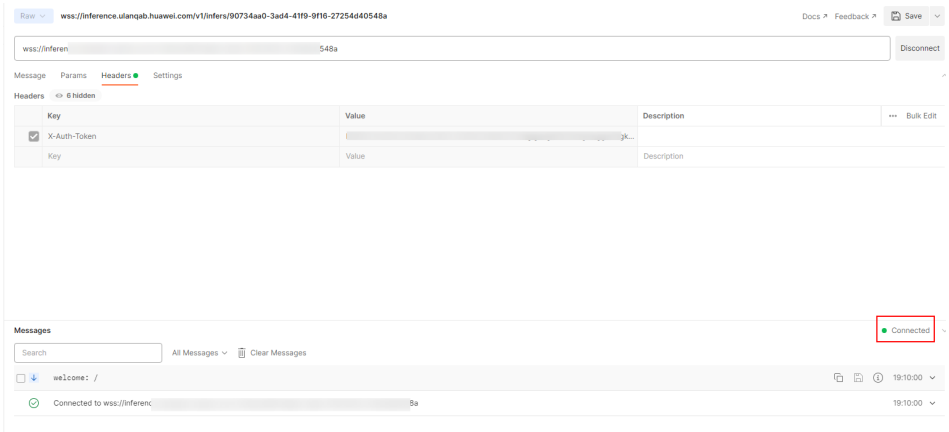


说明

- 如果信息正确，右下角连接状态处会显示：CONNECTED；
- 如果无法建立连接，如果是401状态码，检查认证信息；
- 如果显示WRONG_VERSION_NUMBER等关键字，检查自定义镜像的端口和ws跟wss的配置是否正确。

连接成功后结果如下：

图 13-47 连接成功



须知

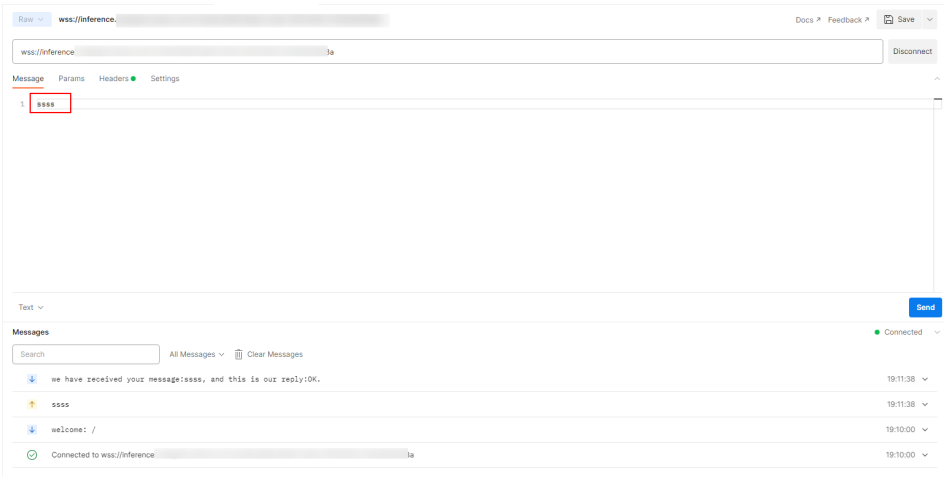
优先验证自定义镜像提供的websocket服务的情况，不同的工具实现的websocket服务会有不同，可能出现连接建立后维持不住，可能出现请求一次后连接就中断需要重新连接的情况，ModelArts平台只保证，未上ModelArts前自定义镜像的websocket的形态跟上了ModelArts平台后的websocket形态相同（除了地址跟认证方式不同）。

步骤2 WebSocket客户端和服务端双向传输数据

连接建立后，WebSocket使用TCP完成全双工通信。WebSocket的客户端可以往服务端发送数据，客户端有不同的实现，同一种语言也存在不同的lib包的实现，这里不考虑实现的不同种类。

客户端发送的内容在协议的角度不限定格式，Postman支持Text/Json/XML/HTML/Binary，以text为例，在输入框中输入要发送的文本，单击右侧中部的Send按钮即可将请求发往服务端，当文本内容过长，可能会导致postman工具卡住。

图 13-48 发送数据



----结束

14 常见问题

14.1 一般性问题

14.1.1 什么是 ModelArts

ModelArts是面向AI开发者的一站式开发平台，提供海量数据预处理及半自动化标注、大规模分布式训练、自动化模型生成及模型按需部署能力，帮助用户快速创建和部署AI应用，管理全周期AI工作流。

“一站式”是指AI开发的各个环节，包括数据处理、模型训练、创建AI应用、AI应用部署都可以在ModelArts上完成。从技术上看，ModelArts底层支持各种异构计算资源，开发者可以根据需要灵活选择使用，而不需要关心底层的技术。同时，ModelArts支持Tensorflow主流开源的AI开发框架，也支持开发者使用自研的算法框架，匹配您的使用习惯。

ModelArts的理念就是让AI开发变得更简单、更方便。

14.1.2 ModelArts 与其他服务的关系

与对象存储服务的关系

ModelArts使用对象存储服务（Object Storage Service，简称OBS）存储数据和模型，实现安全、高可靠和低成本的存储需求。OBS的更多信息请参见《对象存储服务控制台指南》。

与云容器引擎的关系

ModelArts使用云容器引擎（Cloud Container Engine，简称CCE）部署模型为在线服务，支持服务的高并发和弹性伸缩需求。CCE的更多信息请参见《云容器引擎用户指南》。

与容器镜像服务的关系

当使用ModelArts不支持的AI框架构建模型时，可通过构建的自定义镜像导入ModelArts进行训练或推理。您可以通过容器镜像服务（Software Repository for

Container，简称SWR）制作并上传自定义镜像，然后再通过容器镜像服务导入ModelArts。SWR的更多信息请参见《容器镜像服务（SWR）用户指南》。

与云监控的关系

ModelArts使用云监控服务（Cloud Eye Service， 简称CES）监控在线服务和对应模型负载，执行自动实时监控、告警和通知操作。CES的更多信息请参见《云监控服务用户指南》。

与云审计的关系

ModelArts使用云审计服务（Cloud Trace Service，简称CTS）记录ModelArts相关的操作事件，便于日后的查询、审计和回溯。CTS的更多信息请参见《云审计服务用户指南》。

14.1.3 ModelArts 与 DLS 服务的区别?

深度学习服务（DLS）是一站式深度学习平台服务，内置大量优化的网络模型，以便捷、高效的方式帮助用户轻松使用深度学习技术，通过灵活调度按需服务化方式提供模型训练与评估。

但是，DLS服务仅提供深度学习技术，而ModelArts集成了深度学习和机器学习技术，同时ModelArts是一站式的AI开发平台，从数据标注、算法开发、模型训练及部署，管理全周期的AI流程。直白点解释，ModelArts包含并支持DLS中的功能特性。当前，DLS服务已下线，深度学习技术相关的功能可以直接在ModelArts中使用，如果您是DLS服务客户，也可以将DLS的数据迁移至ModelArts中使用。

14.1.4 支持哪些型号的 Ascend 芯片?

目前支持Ascend Snt3和Snt9。

- **模型训练**：支持使用Snt9训练模型。其中ModelArts提供了可直接使用Snt9训练的算法。
- **模型推理**：在ModelArts中将模型部署上线为在线服务时，支持使用Snt3规格资源进行模型推理。

14.1.5 如何获取访问密钥?

获取访问密钥

⚠ 注意

使用云星账号的用户，请联系管理员获取AK/SK。本章节操作方式不适用于云星用户。

1. 登录控制台，进入“我的凭证”页面，选择“访问密钥>新增访问密钥”。
2. 进入“新增访问密钥”页面，输入描述信息，单击“确定”。
3. 根据提示，单击“立即下载”，保存密钥文件。密钥文件会直接保存到浏览器默认的下载文件夹中。即可查看访问密钥（Access Key Id和Secret Access Key）。

14.1.6 如何上传数据至 OBS?

使用ModelArts进行AI模型开发时，您需要将数据上传至对象存储服务（OBS）桶中。您可以登录OBS管理控制台创建OBS桶，并在您创建的OBS桶中创建文件夹，然后再进行数据的上传，OBS上传数据的详细操作请参见《对象存储服务用户指南》。

14.1.7 提示“上传的 AK/SK 不可用”，如何解决?

问题分析

AK与SK是用户访问OBS时需要使用的密钥对，AK与SK是一一对应，且一个AK唯一对应一个用户。如提示不可用，可能是由于账号欠费或AK与SK不正确等原因。

解决方案

1. 使用当前账号登录OBS管理控制台，确认当前账号是否能访问OBS。
   - 是，请执行步骤2。
2. - 是，
   - 否，请根据“**如何管理访问密钥**”操作指导更换为当前账号的AK/SK。

14.1.8 使用 ModelArts 时提示“权限不足”，如何解决?

当您使用ModelArts时如果提示权限不足，请您按照如下指导对相关服务和用户进行授权，并对用户权限进行检查操作。

由于ModelArts的使用权限依赖OBS服务的授权，您需要为用户授予OBS的系统权限。

- 如果您需要授予用户关于OBS的所有权限和ModelArts的基础操作权限，请参见**配置基础操作权限**。
- 如果您需要对用户使用OBS和ModelArts的权限进行精细化管理，进行自定义策略配置，请参见**创建ModelArts自定义策略**。

配置基础操作权限

使用ModelArts的基本功能，您需要为用户配置“作用范围”为“项目级服务”的“ModelArts CommonOperations”权限，由于ModelArts依赖OBS权限，您还需要为用户授予“作用范围”为“全局级服务”的“OBS Administrator”策略。

具体操作步骤如下：

步骤1 创建用户组。

登录IAM管理控制台，单击“用户组>创建用户组”。在“创建用户组”界面，输入“用户组名称”单击“确定”。

步骤2 配置用户组权限。

在用户组列表中，单击步骤1新建的用户组右侧的“授权”，在用户组“授权”页面，您需要配置的权限如下：

1. 配置“作用范围”为“项目级服务”的“ModelArts CommonOperations”权限，如下图所示，然后单击“确定”完成授权。

> **说明**
>
> 区域级项目授权后只在授权区域生效，如果需要所有区域都生效，则所有区域都需要进行授权操作。

2. 配置“作用范围”为“全局级服务”的“OBS Administrator”权限，然后单击“确定”完成授权。

步骤3 在IAM控制台创建用户，并将其加入步骤1中创建的用户组。

步骤4 新创建的用户登录控制台，切换至授权区域，验证权限：

- 在“服务列表”中选择ModelArts，进入ModelArts主界面，选择不同类型的专属资源池，在页面单击“创建”，如果无法进行创建（当前权限仅包含ModelArts CommonOperations），表“ModelArts CommonOperations”已生效。
- 在“服务列表”中选择除ModelArts外（假设当前策略仅包含ModelArts CommonOperations）的任一服务，如果提示权限不足，表示“ModelArts CommonOperations”已生效。
- 在“服务列表”中选择ModelArts，进入ModelArts主界面，单击“数据管理>数据集>创建数据 > 集”，如果可以成功访问对应的OBS路径，表示全局级服务的“OBS Administrator”已生效。

----结束

创建 ModelArts 自定义策略

如果系统预置的ModelArts权限不满足您的授权要求，或者您需要管理用户操作OBS的操作权限，可以创建自定义策略。

目前支持可视化视图创建自定义策略和JSON视图创建自定义策略，本章节将使用JSON视图方式的策略，以为ModelArts用户授予开发环境的使用权限并且配置ModelArts用户OBS相关的最小化权限项为例，指导您进行自定义策略配置。

> **说明**
>
> 如果一个自定义策略中包含多个服务的授权语句，这些服务必须是同一属性，即都是全局级服务或者项目级服务。
>
> 由于OBS为全局服务，ModelArts为项目级服务，所以需要创建两条“作用范围”别为“全局级服务”以及“项目级服务”的自定义策略，然后将两条策略同时授予用户。

1. 创建ModelArts相关OBS的最小化权限的自定义策略。

   登录IAM控制台，在“权限管理>权限”页面，单击“创建自定义策略”。参数配置说明如下：

   - “策略名称”支持自定义。
   - “策略配置方式”为“JSON视图”。
   - “策略内容”请参见**ModelArts依赖的OBS权限自定义策略样例**。

2. 创建ModelArts开发环境的使用权限的自定义策略。参数配置说明如下：
   - “策略名称”支持自定义。
   - “策略配置方式”为“JSON视图”。
   - “策略内容”请参见**ModelArts开发环境使用权限的自定义策略样例**，ModelArts自定义策略中可以添加的授权项（Action）请参见《ModelArts API参考》>权限策略和授权项> 策略及授权项说明。

3. 在IAM控制台创建用户组之后，将步骤**1**中创建的自定义策略授权给该用户组。

4. 在IAM控制台创建用户，并将其加入3中创建的用户组。
5. 新创建的用户登录控制台，切换至授权区域，验证权限：
   - 在“服务列表”中选择ModelArts，进入ModelArts主界面，单击“数据管理 > 数据集”，如果无法进行创建（当前仅包含开发环境的使用权限），表示仅为ModelArts用户授予开发环境的使用权限已生效。
   - 在“服务列表”中选择除ModelArts，进入ModelArts主界面，单击“开发环境>Notebook>创建”，如果可以成功访问“存储配置”项对应的OBS路径，表示为用户配置的OBS相关权限已生效。

ModelArts 依赖的 OBS 权限自定义策略样例

如下示例为ModelArts依赖OBS服务的最小化权限项，包含OBS桶和OBS对象的权限。授予示例中的权限您可以通过ModelArts正常访问OBS不受限制。

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "obs:bucket:ListAllMybuckets",
                "obs:bucket:HeadBucket",
                "obs:bucket:ListBucket",
                "obs:bucket:GetBucketLocation",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:object:PutObject",
                "obs:object:DeleteObject",
                "obs:object:DeleteObjectVersion",
                "obs:object:ListMultipartUploadParts",
                "obs:object:AbortMultipartUpload",
                "obs:object:GetObjectAcl",
                "obs:object:GetObjectVersionAcl",
                "obs:bucket:PutBucketAcl",
                "obs:object:PutObjectAcl"
            ],
            "Effect": "Allow"
        }
    ]
}
```

ModelArts 开发环境使用权限的自定义策略样例

```
{
    "Version": "1.1",
    "Statement": [

        {
            "Effect": "Allow",
            "Action": [
                "modelarts:notebook:list",
                "modelarts:notebook:create" ,
                "modelarts:notebook:get" ,
                "modelarts:notebook:update" ,
                "modelarts:notebook:delete" ,
                "modelarts:notebook:action" ,
                "modelarts:notebook:access"
            ]
        }
    ]
}
```

14.1.9 如何用 ModelArts 训练基于结构化数据的模型?

针对高阶用户，ModelArts在开发环境提供创建Notebook进行代码开发的功能，在训练作业提供创建大数据量训练任务的功能；用户在开发、训练流程中使用Scikit_Learn、XGBoost或Spark_MLlib引擎均可。

14.1.10 在 ModelArts 中如何查看 OBS 目录下的所有文件?

在使用Notebook或训练作业时，需要查看目录下的所有文件，您可以通过如下方式实现：

- 通过OBS管理控制台进行查看。

  使用当前账户登录OBS管理控制台，去查找对应的OBS桶、文件夹、文件。
- 通过接口判断路径是否存在。在已有的Notebook实例，或者创建一个Notebook，执行如下命令，检查路径是否存在。

  ```
  import moxing as mox
  mox.file.list_directory('obs://bucket_name', recursive=True)
  ```

  如果文件较多，请您耐心等待，最终文件路径信息会在提示信息之后显示。

14.1.11 ModelArts 数据集保存到容器的哪里?

ModelArts的数据集和数据存储位置对应的数据都保存在OBS中。

14.1.12 ModelArts 支持哪些 AI 框架?

ModelArts的开发环境Notebook、训练作业、模型推理（即AI应用管理和部署上线）支持的AI框架及其版本，不同模块的呈现方式存在细微差异，各模块支持的AI框架请参见如下描述。

统一镜像列表

ModelArts提供了ARM+Ascend规格的统一镜像，包括MindSpore、PyTorch。适用于开发环境，模型训练，服务部署，请参考**统一镜像列表**。

表 14-1 MindSpore

预置镜像	适配芯片	适用范围
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook、训练、推理部署
mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook、训练、推理部署
mindspore_2.2.10-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Ascend snt9c	Notebook、训练、推理部署

表 14-2 PyTorch

预置镜像	适配芯片	适用范围
pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend snt9b	Notebook、训练、推理部署
pytorch_2.1.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Ascend snt9c	Notebook、训练、推理部署
pytorch_1.11.0-cann_8.0.rc1-py_3.9-hce_2.0.2312-aarch64-snt9c	Ascend snt9c	Notebook、训练、推理部署

开发环境 Notebook

开发环境的Notebook，根据不同的工作环境，对应支持的镜像和版本有所不同。

表 14-3 新版 Notebook 支持的镜像

镜像名称	镜像描述	适配芯片	支持SSH远程开发访问	支持在线JupyterLab访问
pytorch_1.11.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是
pytorch_2.1.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是
mindspore_2.2.0-cann_7.0.1-py_3.9-euler_2.10.7-aarch64-snt9b	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine	Ascend	是	是
mindspore_2.1.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置MindSpore	Ascend	是	是
pytorch_1.11.0-cann_6.3.2-py_3.7-euler_2.10.7-aarch64-snt9b	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是

镜像名称	镜像描述	适配芯片	支持SSH远程开发访问	支持在线Jupyter Lab访问
mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是
mindstudio5.0.rc1-ascend-cann5.1.rc1-euler2.8.3-aarch64	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	否
mindspore1.8.0-cann5.1.2-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是
tensorflow1.15-cann5.1.0-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. TensorFlow are preset in the AI engine.	Ascend	是	是
mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3	Ascend+ARM算法开发和训练基础镜像，AI引擎预置MindSpore	Ascend	是	是
pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3	Ascend+ARM算法开发和训练基础镜像，AI引擎预置PyTorch	Ascend	是	是

镜像名称	镜像描述	适配芯片	支持SSH远程开发访问	支持在线JupyterLab访问
tensorflow1.15-mindspore1.7.0-cann5.1.0-euler2.8-aarch64	Ascend+ARM algorithm development and training. TensorFlow and MindSpore are preset in the AI engine.	Ascend	是	是
tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是
tensorflow1.15.0-cann5.1.2-py3.7-euler2.8.3	Ascend+ARM algorithm development and training. MindSpore are preset in the AI engine.	Ascend	是	是

训练作业

创建训练作业时，训练支持的AI引擎及对应版本如下所示。

预置引擎命名格式如下：

```
<训练引擎名称_版本号>-[cpu | <cuda_版本号 | cann_版本号 >]-<py_版本号>-<操作系统名称_版本号>-<x86_64 | aarch64>
```

表 14-4 训练作业支持的 AI 引擎

工作环境	系统架构	系统版本	AI引擎与版本	支持的cuda或Ascend版本
Ascend-Powered-Engine	aarch64	Euler2.8	mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0

工作环境	系统架构	系统版本	AI引擎与版本	支持的cuda或Ascend版本
PyTorch	aarch64	Euler2.8	pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0
TensorFlow	aarch64	Euler2.8	tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64	cann_6.3.0

说明

不同区域支持的AI引擎有差异，请以实际环境为准。

推理支持的 AI 引擎

在ModelArts创建AI应用时，若使用预置镜像“从模板中选择”或“从OBS中选择”导入模型，则支持如下常用引擎及版本的模型包。

说明

- 标注“推荐”的Runtime来源于统一镜像，后续统一镜像将作为主流的推理基础镜像。
- 推荐将旧版镜像切换为统一镜像，旧版镜像后续将会逐渐下线。
- 待下线的基本镜像不再维护。
- 统一镜像Runtime的命名规范：<AI引擎名字及版本> - <硬件及版本：cpu或cuda或cann> - <python版本> - <操作系统版本> - <CPU架构>

表 14-5 支持的常用引擎及其 Runtime

模型使用的引擎类型	支持的运行环境（Runtime）
TensorFlow	tensorflow_1.15.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64
MindSpore	mindspore_2.0.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64
Pytorch	pytorch_1.11.0-cann_6.3.0-py_3.7-euler_2.8.3-aarch64

14.1.13 ModelArts 训练和推理分别对应哪些功能?

ModelArts训练包括自动学习、训练管理、专属资源池-训练/开发环境功能。

ModelArts推理包括AI应用管理、部署上线功能。

14.1.14 ModelArts AI 识别可以单独针对一个标签识别吗?

标注多个标签进行训练而成的模型，最后部署成在线服务之后也是对标注的多个标签去进行识别的。如果只需要快速识别一种标签，建议单独训练识别此标签的模型使用，并选择较大的部署上线的规格也可以提供识别速度。

14.1.15 为什么资源充足还是在排队?

- 如果是公共资源池，一般是由于其他用户占用资源导致，请耐心等待或根据**训练作业一直在等待中（排队）？**方法降低排队时间。
- 如果是专属资源池，建议您进行以下排查：
  a. 排查专属资源池中是否存在其他作业（包括推理作业、训练作业、开发环境作业等）。
     可通过总览页面，快速判断是否有其他模块的作业或实例在运行中，并进入到相关作业或实例上，判断是否使用了专属资源池。如判断相关作业或实例可停止，则可以停止，释放出更多的资源。
  b. 单击进入专属资源池详情页面，查看作业列表。
     观察队头是否有其他作业在排队，如果已有作业在排队，则新建的作业需要继续等待。
  c. 如果通过排查计算，发现资源确实足够，则考虑可能由于资源碎片化导致的。
     例如，集群共2个节点，每个节点都空闲了4张卡，总剩余卡数为8张卡，但用户的作业要求为1节点8张卡，因此无法调度上。

14.2 数据管理（旧版）

14.2.1 添加图片时，图片大小有限制吗?

在数据管理功能中，针对“物体检测”或“图像分类”的数据集，在数据集中上传更多的图片时，是有限制的。要求单张图片大小不超过8MB，且只支持JPG、JPEG、PNG和BMP四种格式的图片。

解决方案:

- 方法1：使用导入功能。将图片上传至OBS任意目录，通过“从OBS目录导入”方式导入到已有数据集。
- 方法2：使用同步数据源功能。将图片上传到数据集输入目录下（或者其子目录），单击数据集详情页中的“同步数据源”将新增图片导入。需注意的是，同步数据源同时也会将OBS已删除的文件从数据集也删除，请谨慎操作。
- 方法3：新建数据集。将图片上传至OBS任意目录，可以直接使用这些图片目录作为数据集的输入目录，新建一个数据集。

14.2.2 数据集图片无法显示，如何解决?

问题现象

创建的数据集，在进行标注时无法显示图片，单击单张图片也无法查看。或者数据集中提示图片加载异常。

原因分析

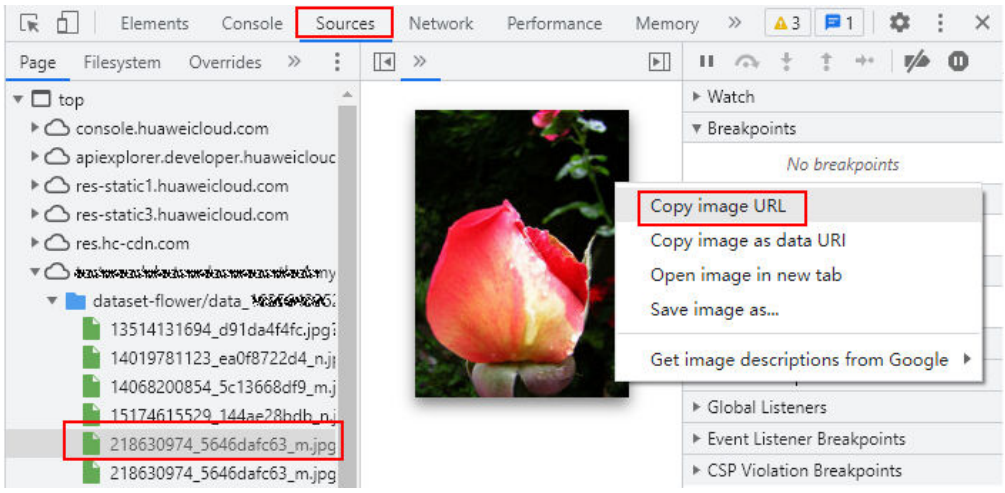
- 可能由于用户本地网络原因，无法正常访问OBS导致图片无法正常加载。
- 可能由于没有OBS桶的访问权限导致，请检查数据集输入位置所在的OBS桶，是否具有访问权限。

- 可能是OBS桶加密或者OBS文件加密导致。
- 可能跟OBS桶的存储类别有关，并行文件系统不支持图像处理，所以无法展示缩略图。

解决方案

1. 以Chrome浏览器为例，“F12”打开浏览器Console，锁定该图片，获取图片链接并复制。

   图 14-1 F12 获取图片链接



2. 在新的浏览器页面输入该链接，会出现提示“您的连接不是私密连接”，在该页面单击“高级”，然后选择继续前往目标链接页面。
3. 图片访问成功后再次返回ModelArts管理控制台访问数据集，即可成功查看图片。

14.2.3 如何将多个物体检测的数据集合并成一个数据集?

可以在OBS桶中创建一个父级目录，目录下面设置不同的文件夹，将多个数据集分别导出到这些文件夹里面，最后用父目录创数据集即可。

登录ModelArts管理控制台，选择“数据管理>数据集”进入数据集概览页，单击右上角“导出”，将对应的数据集到导出至OBS父级目录下的子文件夹中。

14.2.4 导入数据集失败

导入数据集失败可能原因为OBS桶类型选择错误，请您选择标准存储类型的桶导入。

14.2.5 表格类型的数据集如何标注

表格类型的数据集适合表格等结构化数据处理。数据格式支持csv。不支持标注，支持对部分表格数据进行预览，但是最多支持100条数据预览。

14.2.6 本地标注的数据，导入 ModelArts 需要做什么？

ModelArts支持通过导入数据集的操作，导入更多数据。本地标注的数据，当前支持从OBS目录导入或从Manifest文件导入两种方式。导入之后您还可以在ModelArts数据管理模块中对数据进行重新标注或修改标注情况。

14.2.7 为什么通过 Manifest 文件导入失败?

问题现象

针对已发布的数据集，使用此数据集的Manifest文件，重新导入，此时出现导入失败的错误。

原因分析

针对已发布的数据集，其对应的OBS目录下，发生了数据变化，如删除图片，导致此Manifest文件与当前OBS目录下的数据情况不符。使用此Manifest文件再次导入时，出现错误。

解决方案

- 方法1（推荐），建议将此数据集重新发布版本，然后再使用新版本的Manifest文件导入。
- 方法2，修改您本地的Manifest文件，查找OBS目录下的数据变更，根据变更同步修改Manifest。确保Manifest文件与OBS目录下的数据现状相同，然后使用修改后的Manifest文件导入。

14.2.8 标注结果存储在哪里?

ModelArts管理控制台，提供了数据可视化能力，您可以在控制台中查看详细数据以及标注信息。如需了解标注结果的存储路径，请参见如下说明。

背景说明

针对ModelArts中的数据集，在创建数据集时，需指定“数据集输入位置”和“数据集输出位置”。两个参数填写的均是OBS路径。

- “数据集输入位置”即原始数据存储的OBS路径。
- “数据集输出位置”，指在ModelArts完成数据标注后，执行数据集发布操作后，在此指定路径下，按数据集版本，生成相关目录。包含ModelArts中使用的Manifest文件（包含数据及标注信息）。详细文件说明可参见“发布数据集”章节。

查看步骤

1. 在ModelArts管理控制台，进入“数据管理>数据集”。
2. 选择需查看数据集，单击名称左侧小三角，展开数据集详情。可获得“数据集输出位置”指定的OBS路径。

   说明

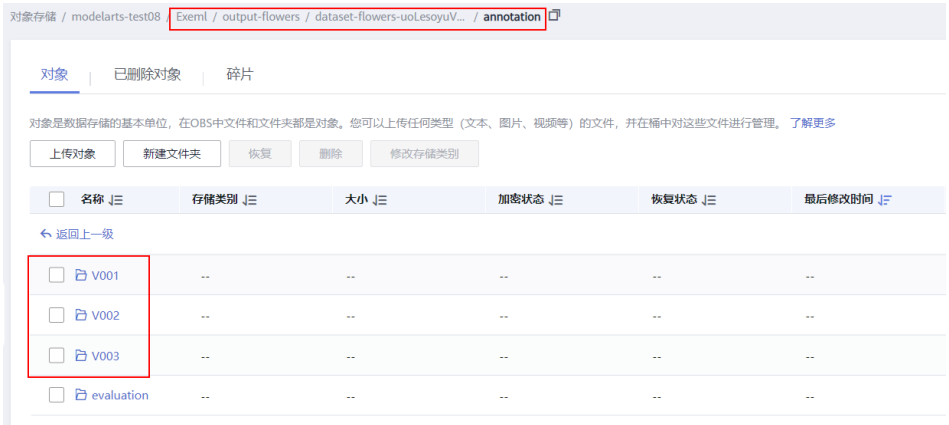
   获取标注信息前，需确保数据集已发布，至少有一个以上数据集版本。

   图 14-2 数据集详情

3. 进入OBS管理控制台，根据上述步骤获得的路径，找到对应版本号目录，即可获取数据集对应的标注结果。

   图 14-3 获取标注结果

14.2.9 如何将标注结果下载至本地?

ModelArts数据集中的标注信息和数据在发布后，将以manifest格式存储在“数据集输出位置”对应的OBS路径下。

路径获取方式：

1. 在ModelArts管理控制台，进入“数据管理>数据集”。
2. 选择需查看数据集，单击名称左侧小三角，展开数据集详情。可获得“数据集输出位置”指定的OBS路径。
3. 进入OBS管理控制台，根据上述步骤获得的路径，找到对应版本号目录，即可获取数据集对应的标注结果。

如需将标注结果下载至本地，可前往manifest文件存储的OBS中，单击“下载”，即可将标注结果存储至本地。

图 14-4 下载标注结果

14.2.10 团队标注时，为什么团队成员收不到邮件?

团队标注时，成员收不到邮件的可能原因如下：

- 当数据集中的所有数据已完成标注，即“未标注”数据为空时，创建的团队标注任务，因为没有数据需要标注，不会给团队成员发送标注邮件。在发起团队标注任务时，请确保数据集中存在“未标注”数据。
- 只有当创建团队标注任务时，标注人员才会收到邮件。创建标注团队及添加标注团队的成员并不会发送邮件。
- 请确保您的邮箱已完成配置且配置无误。可参考，完成邮箱配置。
- 团队成员自检其邮箱是否有拦截设置。

14.2.11 可以两个账号同时进行一个数据集的标注吗?

可以多人同时标注，但多人同时对同一张图片标注的话，只会以最后一个保存的人的标注结果为最终标注结果。建议轮流标注并及时保存标注结果。

14.2.12 标注过程中，已经分配标注任务后，能否将一个 labeler 从标注任务中删除? 删除后对标注结果有什么影响? 如果不能删除 labeler，能否删除将他的标注结果从整体标注结果中分离出来?

目前不支持从标注任务中删除labeler。

labeler的标注必须通过审核后，才能同步到最终结果，不支持单独分离操作。

14.2.13 数据标注中，难例集如何定义? 什么情况下会被识别为难例?

难例是指难以识别的样本，目前只有图像分类和检测支持难例。

14.2.14 物体检测标注时，支持叠加框吗?

支持。

“物体检测”类型的数据集，在标注时，可在一张图片中添加多个标注框以及标签。需注意的是，标注框不能超过图片边缘。

14.2.15 如何将两个数据集合并?

目前不支持直接合并。

但是可以参考如下操作方式，将两个数据集的数据合并在一个数据集中。

例如需将数据集A和数据集B进行合并。

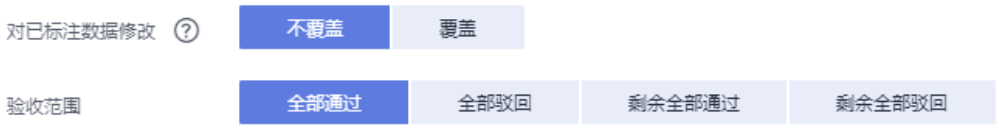
1. 分别将数据集A和数据集B进行发布。
2. 发布后可获得数据集A和数据集B的Manifest文件。可通过数据集的“数据集输出位置”获得此文件。
3. 创建一个空数据集C，即无任何输出，其输入位置选择一个空的OBS文件夹。
4. 在数据集C中，执行导入数据操作，将数据集A和数据集B的Manifest文件导入。

   导入完成后，即将数据集A和数据集B的数据分别都合并至数据集C中。如需使用合并后的数据集，再针对数据集C执行发布操作即可。

14.2.16 智能标注是否支持多边形标注?

不支持。目前智能标注针对矩形框的标注类型，其他标注形式的样本，在智能标注的训练过程中，会跳过这部分。

14.2.17 团队标注的完成验收的各选项表示什么意思?



- 全部通过：被驳回的样本，也会通过。
- 全部驳回：已经通过的样本，需要重新标注，下次验收时重新进行审核。
- 剩余全部通过：已经驳回的会驳回，其余会自动验收通过。
- 剩余全部驳回：样本抽中的通过的，不需要标注了，未通过和样本未抽中的需要重新标注验收。

14.2.18 同一个账户，图片展示角度不同是为什么?

有的图片存在旋转角度等属性，不同的浏览器的处理策略不同，对浏览器的兼容性如表1和表2所示。

- L代表last，L3-产品版本上线时最新的3个稳定浏览器版本。
- 如果您当前使用的浏览器版本过低，将在一定程度上影响页面的显示效果，系统会提示您尽快对浏览器进行升级。
- 如果您当前使用的浏览器不支持访问管理控制台，系统会建议您对浏览器进行升级或安装支持的浏览器。

14.2.19 智能标注完成后新加入数据是否需要重新训练?

智能标注完成后，需要对标注结果进行确认。

- 如果未确认标注结果，直接加入新数据，重新智能标注，会将待确认的数据和新加入的数据全部重新训练。
- 如果确认标注结果后，再加入新数据，只重新训练标注新的数据。

14.2.20 为什么在 ModelArts 数据标注平台标注数据提示标注保存失败?

问题现象

以Chrome浏览器为例，同一张图片，第一次标注时，右上角弹窗提示标注保存失败，第二次提交相同的标注结果，又提示标注成功，此问题概率性发生。“F12”打开浏览器Console，单击network查看请求列表，请求状态显示为(failed)net::ERR_*ADDRESS*_IN_USE。

原因分析

可能是用户本地网络的原因，网速不稳定或者网络配置有问题，均可能导致保存失败。

解决方案

1. 切换为稳定的网络后重试。

2. 初始化网络配置，使用管理员权限启动CMD，输入netsh winsock reset指令，完成后重启电脑，再登录数据标注平台重试。

14.2.21 标注多个标签，是否可针对一个标签进行识别?

数据标注时若标注多个标签进行训练而成的模型，最后部署成在线服务之后也是对标注的多个标签去进行识别的。如果只需要快速识别一种标签，建议单独训练识别此标签的模型使用，并选择较大的部署上线的规格也可以提供识别速度。

14.2.22 使用数据处理的数据扩增功能后，新增图片没有自动标注

物体检测支持扩增后的图片自动标注，图像分类暂不支持。

14.2.23 视频数据集无法显示和播放视频

若无法显示和播放视频，请检查视频格式类型，目前只支持MP4格式。

14.2.24 使用样例的有标签的数据或者自己通过其他方式打好标签的数据放到 OBS 桶里，在 modelarts 中同步数据源以后看不到已标注，全部显示为未标注

OBS桶设置了自动加密会导致此问题，需要新建OBS桶重新上传数据，或者取消桶加密后，重新上传数据。

14.2.25 如何使用 soft NMS 方法降低目标框堆叠度

目前AI市场订阅的算法中，yolo3可以使用该方法降低目标框堆叠度，yolo5 算法中没有看到相关支持的信息，需要在自定义算法进行使用。

14.2.26 ModelArts 标注数据丢失，看不到标注过的图片的标签

原因是删除了默认的标注作业，导致标签被删除。

14.2.27 如何将某些图片划分到验证集或者训练集?

目前只能指定切分比例，随机将样本划分到训练集或者验证集，不支持指定。

切分比例的指定：

在发布数据集时，仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。

一般默认不启用该功能。启用后，需设置对应的训练验证比例。

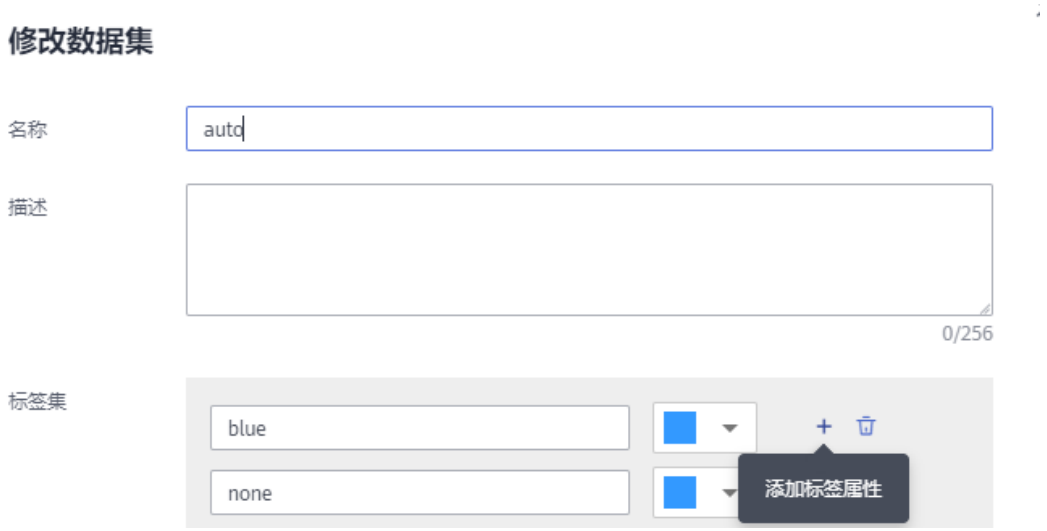
输入“训练集比例”，数值只能是0~1区间内的数。设置好“训练集比例”后，“验证集比例”自动填充。“训练集比例”加“验证集比例”等于1。

“训练集比例”即用于训练模型的样本数据比例；“验证集比例”即用于验证模型的样本数据比例。“训练验证比例”会影响训练模板的性能。

14.2.28 物体检测标注时除了位置、物体名字，是否可以设置其他标签，比如是否遮挡、亮度等?

可以通过修改数据集给标签添加自定义属性来设置一些自定义的属性。

图 14-5 修改数据集



14.2.29 ModelArts 数据管理支持哪些格式?

不同类型的数据集支持不同的功能。

数据集类型	标注类型	创建数据集	导入数据	导出数据	发布数据集	修改数据集	管理版本	自动分组	数据特征
文件型	图像分类	支持	支持	支持	支持	支持	支持	支持	支持
	物体检测	支持	支持	支持	支持	支持	支持	支持	支持
	图像分割	支持	支持	支持	支持	支持	支持	支持	-
	声音分类	支持	支持	-	支持	支持	支持	-	-
	语音内容	支持	支持	-	支持	支持	支持	-	-
	语音分割	支持	支持	-	支持	支持	支持	-	-
	文本分类	支持	支持	-	支持	支持	支持	-	-
	命名实体	支持	支持	-	支持	支持	支持	-	-
	文本三元组	支持	支持	-	支持	支持	支持	-	-
	视频	支持	支持	-	支持	支持	支持	-	-

数据集类型	标注类型	创建数据集	导入数据	导出数据	发布数据集	修改数据集	管理版本	自动分组	数据特征
	自由格式	支持	-	支持	支持	支持	支持	-	-
表格型	表格	支持	支持	-	支持	支持	支持	-	-

14.2.30 旧版数据集中的数据是否会被清理?

旧版数据集中创建的数据不会被清理，旧版数据集中会自动关联一个数据标注任务。

但是在新版数据集中创建的数据，在旧版的数据集列表不会展示。

14.2.31 数据集版本管理找不到新建的版本

版本列表是可以缩放的，请缩小页面后查找。

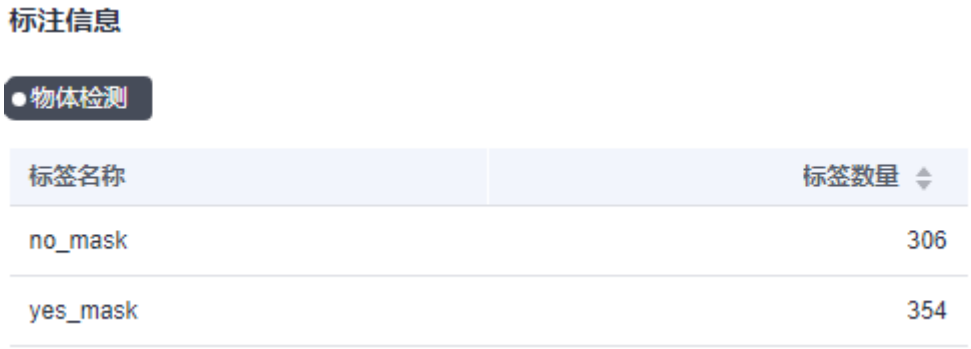
单击数据集名称，进入数据集概览页，在概览页选择“版本管理”，可对页面进行缩小。

14.2.32 如何查看数据集大小

数据管理目前只统计数据集的样本数量，无法查看数据集大小。

14.2.33 如何查看新版数据集的标注详情

1. 登录ModelArts管理控制台，左侧菜单栏选择“数据管理>数据集”。
2. 按照数据集名称，找到您想查看的数据集，单击该数据集名称，进入数据集概览页。
3. 在“概览”页签下，标注信息框，单击“查看标注详情”即可。



14.2.34 标注数据如何导出

只有“图像分类”、“物体检测”、“图像分割”类型的数据集支持导出功能。

- “图像分类”只支持导出txt格式的标注文件。
- “物体检测”只支持导出Pascal VOC格式的XML标注文件。
- “图像分割”只支持导出Pascal VOC格式的XML标注文件以及Mask图像。

其他类型的数据集可以使用。

14.2.35 找不到新创建的数据集

目前旧版数据集页面不展示新版数据集，新版数据集查看需跳转到新版的页面。

14.2.36 数据集配额不正确

当前每个账号支持的数据集配额为100，新版数据集页面显示所有已创建的数据集，但是旧版数据集页面不显示新版数据集。所以旧版页面存在显示不完整的情况，可以前往新版数据集页面查看。

14.2.37 数据集如何切分

在发布数据集时，仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。

一般默认不启用该功能。启用后，需设置对应的训练验证比例。

输入“训练集比例”，数值只能是0~1区间内的数。设置好“训练集比例”后，“验证集比例”自动填充。“训练集比例”加“验证集比例”等于1。

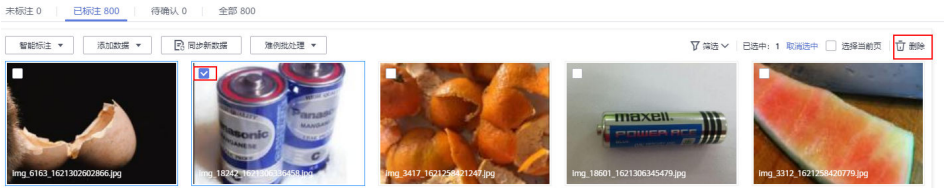
“训练集比例”即用于训练模型的样本数据比例；“验证集比例”即用于验证模型的样本数据比例。“训练验证比例”会影响训练模板的性能。

14.2.38 如何删除数据集图片

1. 登录ModelArts管理控制台，左侧菜单栏选择“数据管理>数据标注”，进入数据标注列表，单击需要删除图片的数据集，进入标注详情页。
2. 在“全部”、“未标注”或“已标注”页面中，依次选中需要删除的图片，或者“选择当前页”选中该页面所有图片，然后单击 删除。在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成图片删除操作。

   其中，被选中的图片，其左上角将显示为勾选状态。如果当前页面无选中图片时， 按钮为灰色，无法执行删除操作。

图 14-6 删除数据集图片



14.2.39 从 AI Hub 下载到桶里的数据集，再在 ModelArts 里创建数据集，显示样本数为 0

首先需要确认从AI Hub下载的数据格式，比如压缩包、excel文件等会被忽略，支持格式详情：

数据集类型	标注类型	创建数据集	导入数据	导出数据	发布数据集	修改数据集	管理版本	自动分组	数据特征
文件型	图像分类	支持	支持	支持	支持	支持	支持	支持	支持
	物体检测	支持	支持	支持	支持	支持	支持	支持	支持
	图像分割	支持	支持	支持	支持	支持	支持	支持	-
	声音分类	支持	支持	-	支持	支持	支持	-	-
	语音内容	支持	支持	-	支持	支持	支持	-	-

数据集类型	标注类型	创建数据集	导入数据	导出数据	发布数据集	修改数据集	管理版本	自动分组	数据特征
	语音分割	支持	支持	-	支持	支持	支持	-	-
	文本分类	支持	支持	-	支持	支持	支持	-	-
	命名实体	支持	支持	-	支持	支持	支持	-	-
	文本三元组	支持	支持	-	支持	支持	支持	-	-
	视频	支持	支持	-	支持	支持	支持	-	-
	自由格式	支持	-	支持	支持	支持	支持	-	-
表格型	表格	支持	支持	-	支持	支持	支持	-	-

14.3 Notebook（新版）

14.3.1 规格限制

14.3.1.1 是否支持 sudo 提权?

出于安全考虑，Notebook不支持sudo提权操作。

14.3.1.2 是否支持 apt-get?

目前ModelArts开发环境的Terminal不支持使用“apt-get”。您可以使用**自定义镜像**来实现。

14.3.1.3 是否支持 Keras 引擎?

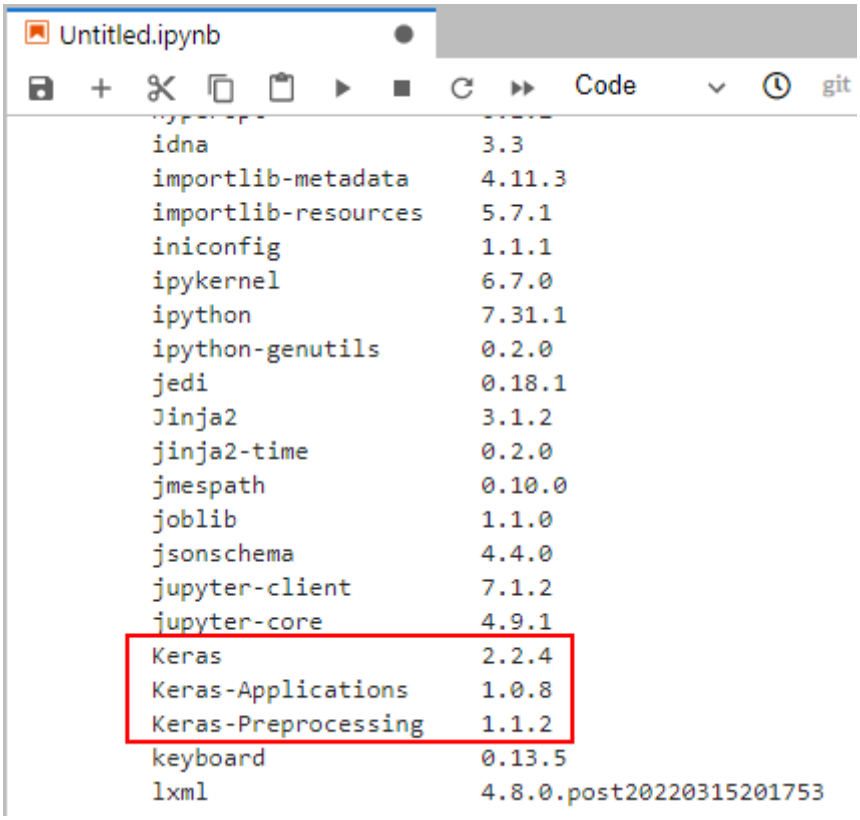
开发环境中的Notebook支持。训练作业和模型部署（即推理）暂时不支持。

Keras是一个用Python编写的高级神经网络API，它能够以TensorFlow、CNTK或者Theano作为后端运行。Notebook开发环境支持“tf.keras”。

如何查看 Keras 版本

1. 在ModelArts管理控制台，创建一个Notebook实例，镜像选择“TensorFlow-1.13”或“TensorFlow-1.15”。
2. 打开Notebook，在JupyterLab中执行**!pip list**查看Keras的版本。

图 14-7 查看 Keras 引擎版本



14.3.1.4 是否支持 caffe 引擎?

ModelArts的python2环境支持使用caffe，目前python3环境无法使用caffe。

14.3.1.5 是否支持本地安装 MoXing?

不支持，目前MoXing只支持在ModelArts里面使用。

14.3.1.6 Notebook 支持远程登录吗?

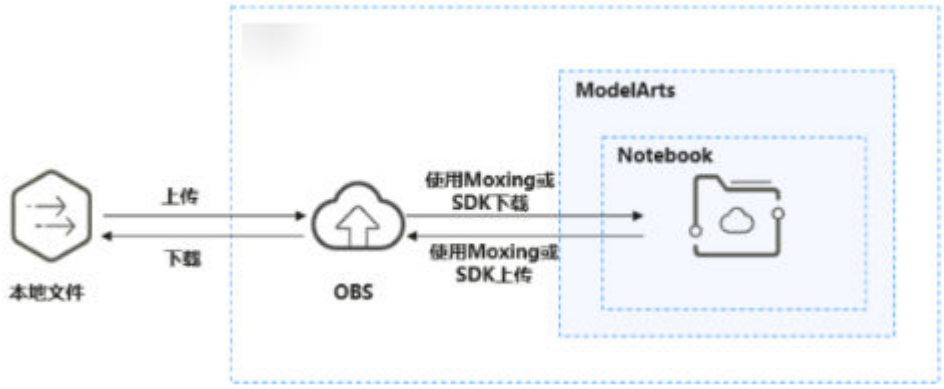
支持。创建Notebook时，可以开启SSH远程开发选项。在本地IDE通过或远程登录Notebook实例。

14.3.2 文件上传下载

14.3.2.1 如何在 Notebook 中上传下载 OBS 文件?

在Notebook中可以通过调用ModelArts的Moxing接口或者SDK接口与OBS交互，将Notebook中的文件上传至OBS，或者下载OBS中的文件至Notebook中。

图 14-8 Notebook 中上传下载 OBS 文件



方法一：在 Notebook 中通过 Moxing 上传下载 OBS 文件

MoXing是ModelArts自研的分布式训练加速框架，构建于开源的深度学习引擎TensorFlow、PyTorch等之上，使用MoXing API可让模型代码的编写更加简单、高效。

MoXing提供了一套文件对象API，可以用来读写OBS文件。

示例代码：

```
import moxing as mox

#下载一个OBS文件夹sub_dir_0，从OBS下载至Notebook
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/home/ma-user/work/sub_dir_0')
#下载一个OBS文件obs_file.txt，从OBS下载至Notebook
mox.file.copy('obs://bucket_name/obs_file.txt', '/home/ma-user/work/obs_file.txt')

#上传一个OBS文件夹sub_dir_0，从Notebook上传至OBS
mox.file.copy_parallel('/home/ma-user/work/sub_dir_0', 'obs://bucket_name/sub_dir_0')
#上传一个OBS文件obs_file.txt，从Notebook上传至OBS
mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```

方法二：在 Notebook 中通过 SDK 上传下载 OBS 文件

使用ModelArts SDK接口将OBS中的文件下载到Notebook后进行操作。

示例代码：将OBS中的文件file1.txt下载到Notebook的/home/ma-user/work/路径下。其中，桶名称、文件夹和文件的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.download_file(src_obs_file="obs://bucket-name/dir1/file1.txt", dst_local_dir="/home/ma-user/work/")
```

使用ModelArts SDK接口将OBS中的文件夹下载到Notebook后进行操作。

示例代码：将OBS中的文件夹dir1下载到Notebook的/home/ma-user/work/路径下。其中，桶名称和文件夹的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.download_dir(src_obs_dir="obs://bucket-name/dir1/", dst_local_dir="/home/ma-user/work/")
```

使用ModelArts SDK接口将Notebook中的文件上传到OBS后进行操作。

示例代码：将Notebook中的file1.txt文件上传到OBS桶路径obs://bucket-name/dir1/中。其中，桶名称、文件夹和文件的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.upload_file(src_local_file='/home/ma-user/work/file1.txt', dst_obs_dir='obs://bucket-name/dir1/')
```

使用ModelArts SDK接口将Notebook中的文件夹上传到OBS。

示例代码：将Notebook中的文件夹“/work/”上传至“bucket-name”桶的“dir1”文件夹下，路径为“obs://bucket-name/dir1/work/”。其中，桶名称和文件夹的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.upload_dir(src_local_dir='/home/ma-user/work/', dst_obs_dir='obs://bucket-name/dir1/')
```

14.3.2.2 如何上传本地文件至 Notebook?

新版Notebook中JupyterLab的文件上传方式请参见**上传本地文件至JupyterLab**。

14.3.2.3 如何导入大文件到 Notebook 中?

- **大文件（大于100MB的文件）**

  针对大文件，建议使用OBS服务上传文件。使用OBS客户端，将本地文件上传至OBS桶中，然后使用ModelArts SDK从OBS下载文件至Notebook本地。

  使用ModelArts SDK或Moxing接口从OBS下载文件请参见**如何在Notebook中上传下载OBS文件?** 。

- **文件夹**

  将文件夹压缩成压缩包，上传方式与大文件相同。将文件上传至Notebook后，可在Terminal中解压压缩包。

  ```
  unzip xxx.zip  #在xxx.zip压缩包所在路径直接解压
  ```

  解压命令的更多使用说明可以在主流搜索引擎中查找Linux解压命令操作。

14.3.2.4 upload 后，数据将上传到哪里?

如果您创建的Notebook使用OBS存储实例时，单击“upload”后，数据将直接上传到该Notebook实例对应的OBS路径下，即创建Notebook时指定的OBS路径。

14.3.2.5 如何下载 Notebook 中的文件到本地?

新版Notebook中JupyterLab下载文件到本地的方式，请参见**从JupyterLab下载文件至本地**。

14.3.2.6 如何将开发环境 Notebook A 的数据复制到 Notebook B 中?

目前不支持直接将Notebook A的数据复制到Notebook B，如果需要复制数据，可参考如下步骤操作：

1. 将Notebook A的数据上传至OBS；
2. 下载OBS中的数据至Notebook B。

文件的上传下载详细操作请参考**如何在Notebook中上传下载OBS文件?** 。

14.3.2.7 在 Notebook 中上传文件失败，如何解决?

问题现象

- 文件上传很快，但是上传失败。
- 上传文件到Notebook时，界面一直在转圈；使用Moxing命令上传，报错；上传OBS文件时，打开OBS浏览器也不显示桶，一直在“获取数据中”。
- 在JupyterLab界面通过 ModelArts Upload Files按钮上传文件时，显示“获取数据失败”。

  图 14-9 OBS 文件上传界面

  查看Notebook日志（通常在/home/ma-user/log/下，notebook-<date>.log），报错“List objects failed, obs_client resp: {'status' : 403, 'reason' : 'Forbidden' , 'errorCode' : 'AccessDenied'”。

可能原因

第一种问题现象是通过内网上传时，文件大小受限，需要解决内网的问题。

其它问题现象的可能原因如下：

- 无OBS访问授权。
- 无OBS桶或文件的访问权限。
- OBS桶被删除。

解决方案

- **检查委托授权**

  请前往全局配置，查看是否具有OBS访问授权。如果没有，请参考配置访问授权（全局配置）。
- **请确认是否有OBS桶的访问权限**

  进入OBS控制台页面，可以看到所有的OBS桶列表，进入需要访问的桶，确认是否有权限访问，如果无权限则会报错。
- **进入OBS控制台页面，确认OBS桶是否存在。**

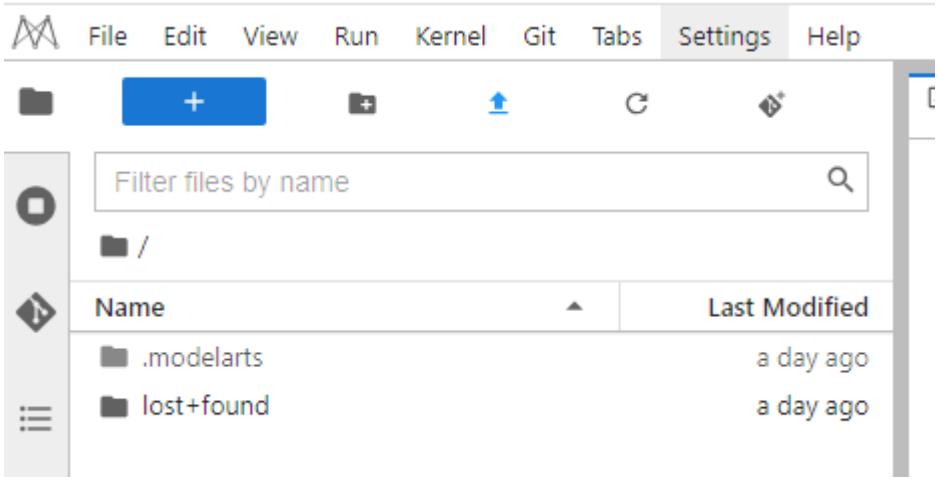
14.3.2.8 动态挂载 OBS 并行文件系统成功，但是在 Notebook 的 JupyterLab 中无法看到本地挂载点

问题现象

在Notebook中动态挂载OBS并行文件系统，本地挂载目录为/data/demo-yf/，实际在JupyterLab左侧导航看不到此目录。

图 14-10 本地挂载目录

图 14-11 Notebook 的 JupyterLab

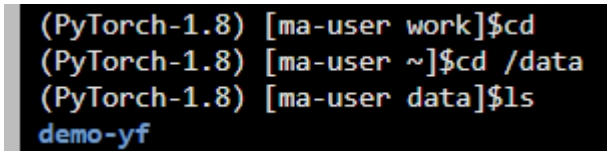


原因分析

本地挂载目录是在Notebook容器的“~/data”目录下创建的demo-yf文件夹，而JupyterLab左侧导航默认路径为“~/work”目录，相当于/data和/work是同一层级，所以在JupyterLab中看不到。

打开Terminal后，默认为~work目录，执行如下命令进入~data目录查看本地挂载路径：

```
(PyTorch-1.8) [ma-user work]$cd
(PyTorch-1.8) [ma-user ~]$cd /data
(PyTorch-1.8) [ma-user data]$ls
```



14.3.3 数据存储

14.3.3.1 如何对 OBS 的文件重命名?

由于OBS管理控制台不支持对OBS的文件重命名，当您需要对OBS文件进行重命名时需要通过调用MoXing API实现，在已有的或者新创建的Notebook中，执行如下命令，通过接口对OBS中的文件进行重命名。

具体操作如下：

如下示例为将文件“obs_file.txt”重命名为“obs_file_2.txt”。

```
import moxing as mox
mox.file.rename('obs://bucket_name/obs_file.txt', 'obs://bucket_name/obs_file_2.txt')
```

14.3.3.2 Notebook 停止或者重启后，“/cache”下的文件还存在么？如何避免重启?

“/cache”目录下存储的是临时文件，在Notebook实例停止或重启后，不会被保存。存储在“/home/ma-user/work”目录下的数据，在Notebook实例停止或重启后，会被保留。

为避免重启，请勿在开发环境中进行重型作业训练，如大量占用资源的作业。

14.3.3.3 如何使用 pandas 库处理 OBS 桶中的数据?

步骤1 参考**下载OBS文件到Notebook中**的指导，将OBS中的数据下载至Notebook本地处理。

步骤2 参考**pandas用户指南**处理pandas数据。

----结束

14.3.4 环境配置相关

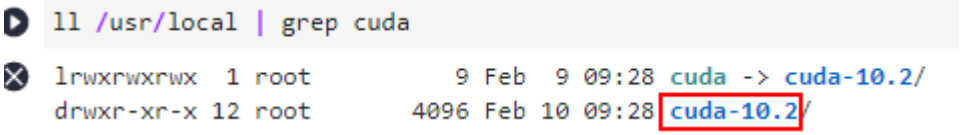
14.3.4.1 如何查看 Notebook 使用的 cuda 版本?

执行如下命令查看环境中的cuda版本。

```
ll /usr/local | grep cuda
```

举例：

图 14-12 查看当前环境的 cuda 版本



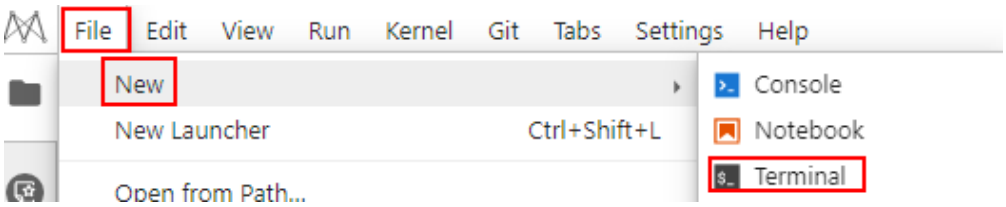
如**图1**所示，当前环境中cuda版本为10.2

14.3.4.2 如何打开 ModelArts 开发环境的 Terminal 功能?

1. 登录ModelArts管理控制台，选择“开发环境>Notebook”。
2. 创建Notebook实例，实例处于“运行中”，单击“操作”列的“打开”，进入“JupyterLab”开发页面。

3. 选择“Files > New > Terminal”，进入到Terminal界面。

图 14-13 进入 Terminal 界面



14.3.4.3 如何在 Notebook 中安装外部库?

ModelArts Notebook中已安装Jupyter、Python程序包等多种环境，包括TensorFlow、MindSpore、PyTorch、Spark等。您也可以使用pip install在Notobook或Terminal中安装外部库。

在 Notebook 中安装

例如，通过JupyterLab在“TensorFlow-1.8”的环境中安装Shapely。

1. 打开一个Notebook实例，进入到Launcher界面。
2. 在“Notebook”区域下，选择“TensorFlow-1.8”，新建一个ipynb文件。
3. 在新建的Notobook中，在代码输入栏输入如下命令。

   !pip install Shapely

在 Terminal 中安装

例如，通过terminal在“TensorFlow-1.8”的环境中使用**pip**安装Shapely。

1. 打开一个Notebook实例，进入到Launcher界面。
2. 在“Other”区域下，选择“Terminal”，新建一个terminal文件。
3. 在代码输入栏输入以下命令，获取当前环境的kernel，并激活需要安装依赖的python环境。

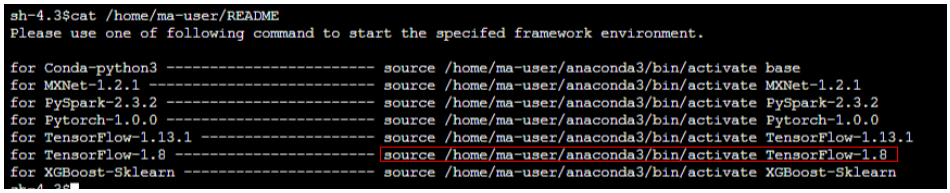
   cat /home/ma-user/README

   source /home/ma-user/anaconda3/bin/activate TensorFlow-1.8

   说明

   如果需要在其他python环境里安装，请将命令中“TensorFlow-1.8”替换为其他引擎。

图 14-14 激活环境



4. 在代码输入栏输入以下命令安装Shapely。

   pip install Shapely

14.3.4.4 如何获取本机外网 IP?

本机的外网IP地址可以在主流搜索引擎中搜索“IP地址查询”获取。

图 14-15 查询外网 IP 地址

14.3.4.5 如何解决“在 IOS 系统里打开 ModelArts 的 Notebook，字体显示异常”的问题?

问题现象

在IOS系统里打开ModelArts的Notebook时，字体显示异常。

解决方法

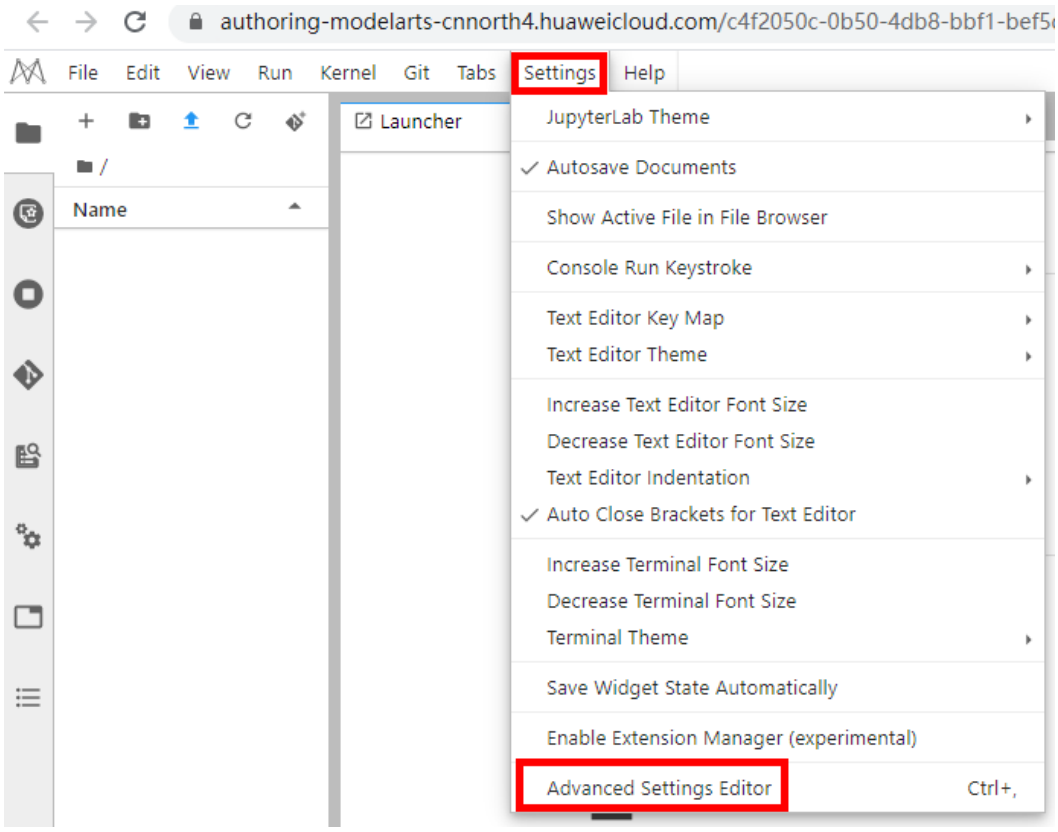
设置Terminal的“fontFamily”为“Menlo”。

操作步骤

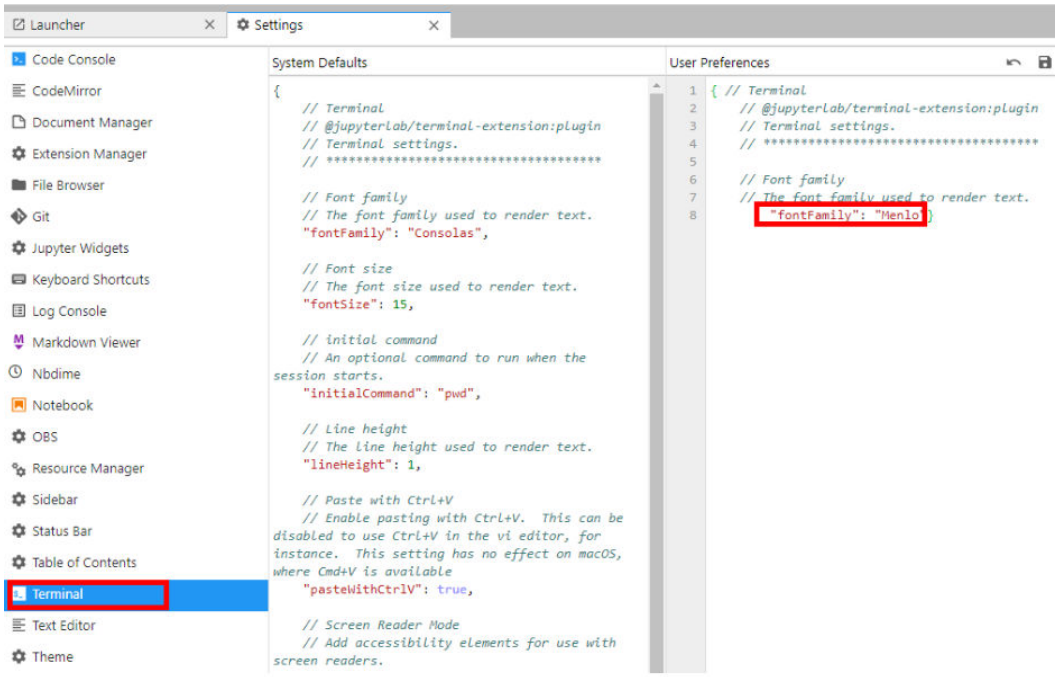
步骤1 登录ModelArts管理控制台，选择“开发环境>Notebook”。

步骤2 创建Notebook实例之后，在Notebook列表中，单击目标Notebook“操作”列的“打开”，进入“JupyterLab”开发页面。

步骤3 在“JupyterLab”页面，选择菜单栏的“Settings>Advanced Settings Editor”，进入“Settings”页面。



步骤4 单击Terminal，将“fontFamily”设置为“Menlo”。



----结束

14.3.4.6 Notebook 有代理吗？如何关闭？

Notebook有代理。

执行**env|grep proxy**命令查询Notebook代理。

执行**unset https_proxy unset http_proxy**命令关闭代理。

14.3.5 Notebook 实例常见错误

14.3.5.1 创建 Notebook 实例后无法打开页面，如何处理?

如果您在创建Notebook实例之后，打开Notebook时，因报错导致无法打开页面，您可以根据以下对应的错误码来排查解决。

打开 Notebook 显示黑屏

Notebook打开后黑屏，由于代理问题导致，切换代理。

打开 Notebook 显示空白

- 打开Notebook时显示空白，请清理浏览器缓存后尝试重新打开。
- 检查浏览器是否安装了过滤广告组件，如果是，请关闭该组件。

报错 404

如果是IAM用户在创建实例时出现此错误，表示此IAM用户不具备对应存储位置（OBS桶）的操作权限。

解决方法：

1. 使用账号登录OBS，并将对应OBS桶的访问权限授予该IAM用户。
2. IAM用户获得权限后，登录ModelArts管理控制台，删除该实例，然后重新使用此OBS路径创建Notebook实例。

报错 503

如果出现503错误，可能是由于该实例运行代码时比较耗费资源。建议先停止当前Notebook实例，然后重新启动。

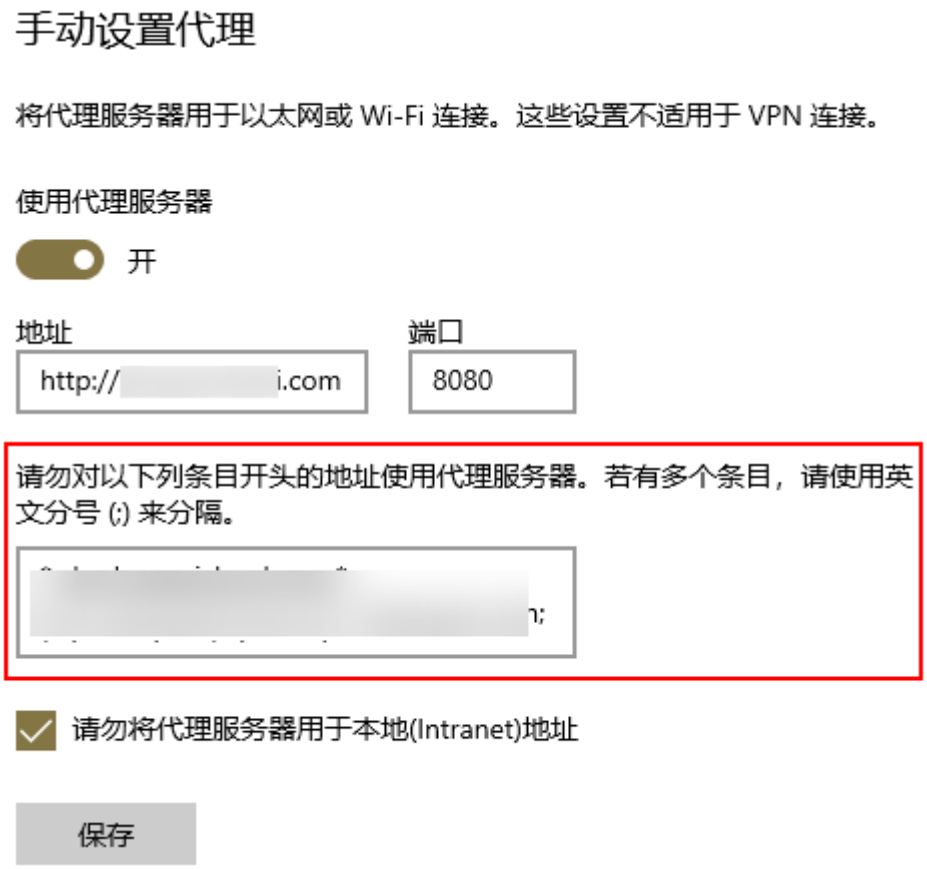
报错 500

Notebook JupyterLab页面无法打开，报错500，可能是工作目录work下的磁盘空间满了，请排查并清理磁盘空间。

报错 This site can't be reached

创建完Notebook后，单击操作列的“打开”，报错如下：

解决方案：复制页面的域名，添加到windows代理“请勿对以下列条目开头的地址使用代理服务器”中，然后保存就可以正常打开。



14.3.5.2 使用 pip install 时出现“没有空间”的错误

问题现象

在Notebook实例中，使用**pip install**时，出现“No Space left...”的错误。

解决办法

建议使用**pip install --no-cache **** 命令安装，而不是使用**pip install ****。加上“--no-cache”参数，可以解决很多此类报错。

14.3.5.3 使用 pip install 提示 Read timed out

问题现象

在Notebook实例中，使用**pip install**时，提示“ReadTimeoutError...”或者“Read timed out...”的错误。

```
sh-4.3$pip install torch==1.7.0 torchvision==0.8.0 torchaudio==0.7.0 matplotlib pyyaml tqdm sklearn h5py tensorboard pandas
Looking in indexes: http://pip-notebook.modelarts.com:8888/repository/pypi/simple/
Collecting torch==1.7.0
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeou
tError("HTTPConnectionPool(host='pip-notebook.modelarts.com', port=8888): Read timed out. (read timeout=15)")': /repository/pypi
/packages/torch/1.7.0/torch-1.7.0-cp37-cp37m-manylinux1_x86_64.whl
  WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeou
tError("HTTPConnectionPool(host='pip-notebook.modelarts.com', port=8888): Read timed out. (read timeout=15)")': /repository/pypi
/packages/torch/1.7.0/torch-1.7.0-cp37-cp37m-manylinux1_x86_64.whl
  WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeou
tError("HTTPConnectionPool(host='pip-notebook.modelarts.com', port=8888): Read timed out. (read timeout=15)")': /repository/pypi
/packages/torch/1.7.0/torch-1.7.0-cp37-cp37m-manylinux1_x86_64.whl
^CERROR: Operation cancelled by user
WARNING: You are using pip version 21.0.1; however, version 21.1.2 is available.
You should consider upgrading via the '/opt/conda/bin/python -m pip install --upgrade pip' command.
```

解决办法

建议先尝试使用**pip install --upgrade pip**，再使用**pip install**。

14.3.5.4 出现“save error”错误，可以运行代码，但是无法保存

如果当前Notebook还可以运行代码，但是无法保存，保存时会提示“save error”错误。大多数原因是WAF安全拦截导致的。

当前页面，即用户的输入或者代码运行的输出有一些字符被拦截，认为有安全风险。出现此问题时，请提交工单，联系专业的工程师帮您核对并处理问题。

14.3.5.5 使用 SSH 工具连接 Notebook，服务器的进程被清理了，GPU 使用率显示还是 100%

原因是代码运行卡死导致被进程清理，GPU显存没有释放；或者代码运行过程中内存溢出导致程序被清理，需要释放下显存，清理GPU，然后重新启动。为了避免进程结束引起的代码未保存，建议您每隔一段时间保存下代码输出至OBS桶或者容器./work目录下。

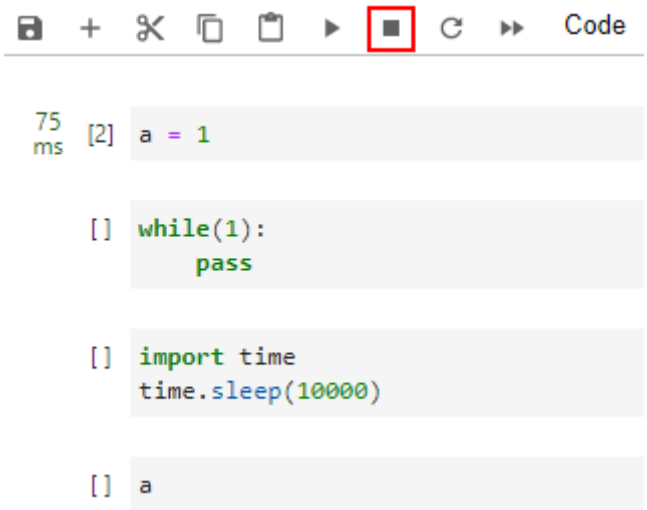
14.3.6 代码运行常见错误

14.3.6.1 Notebook 无法执行代码，如何处理?

当Notebook出现无法执行时，您可以根据如下几种情况判断并处理。

1. 如果只是Cell的执行过程卡死或执行时间过长，如图14-16中的第2个和第3个Cell，导致第4个Cell无法执行，但整个Notebook页面还有反应，其他Cell也还可以单击，则直接单击下图中红色方框处的“interrupt the kernel”，停止所有Cell的执行，同时会保留当前Notebook中的所有变量空间。

   图 14-16 停止所有 Cell

2. 如果整个Notebook页面也已经无法使用，单击任何地方都无反应，则关闭Notebook页面，关闭ModelArts管理控制台页面。然后，重新打开管理控制台，打开之前无法使用的Notebook，此时的Notebook仍会保留无法使用之前的所有变量空间。

3. 如果重新打开的Notebook仍然无法使用，则进入ModelArts管理控制台页面的Notebook列表页面，“停止”此无法使用的Notebook。待Notebook处于“停止”状态后，再单击“启动”，重新启动此Notebook，并打开Notebook。此时，Notebook仍会保留无法使用之前的所有变量空间。

14.3.6.2 运行训练代码，出现 dead kernel，并导致实例崩溃

在Notebook实例中运行训练代码，如果数据量太大或者训练层数太多，亦或者其他原因，导致出现“内存不够”问题，最终导致该容器实例崩溃。

出现此问题后，系统将自动重启Notebook，来修复实例崩溃的问题。此时只是解决了崩溃问题，如果重新运行训练代码仍将失败。如果您需要解决“内存不够”的问题，建议您创建一个新的Notebook，使用更高规格的资源池，比如专属资源池来运行此训练代码。已经创建成功的Notebook不支持选用更高规格的资源规格进行扩容。

14.3.6.3 如何解决训练过程中出现的 cudaCheckError 错误?

问题现象

Notebook中，运行训练代码出现如下错误。

```
cudaCheckError() failed : no kernel image is available for execution on the device
```

原因分析

因为编译的时候需要设置setup.py中编译的参数arch和code和电脑的显卡匹配。

解决方法

对于Tesla V100的显卡，GPU算力为**-gencode arch=compute_70,code=[sm_70,compute_70]**，设置setup.py中的编译参数即可解决。

14.3.6.4 开发环境提示空间不足，如何解决?

当提示空间不足时，推荐使用EVS类型的Notebook实例。

参考**如何在Notebook中上传下载OBS文件?** 操作指导，针对原有的Notebook，首先将代码和数据上传至OBS桶中。然后创建一个EVS类型的Notebook，将此OBS中的文件下载至Notebook本地（指新建的EVS类型Notebook）。

14.3.6.5 如何处理使用 opencv.imshow 造成的内核崩溃?

问题现象

当在Notebook中使用opencv.imshow后，会造成Notebook崩溃。

原因分析

opencv的cv2.imshow在jupyter这样的client/server环境下存在问题。 而matplotlib不存在这个问题。

解决方法

参考如下示例进行图片显示。注意opencv加载的是BGR格式， 而matplotlib显示的是RGB格式。

Python语言：

```
from matplotlib import pyplot as plt
import cv2
img = cv2.imread('图片路径')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('my picture')
plt.show()
```

14.3.6.6 使用 Windows 下生成的文本文件时报错找不到路径?

问题现象

当在Notebook中使用Windows下生成的文本文件时，文本内容无法正确读取，可能报错找不到路径。

原因分析

Notebook是Linux环境，和Windows环境下的换行格式不同，Windows下是CRLF，而Linux下是LF。

解决方法

可以在Notebook中转换文件格式为Linux格式。

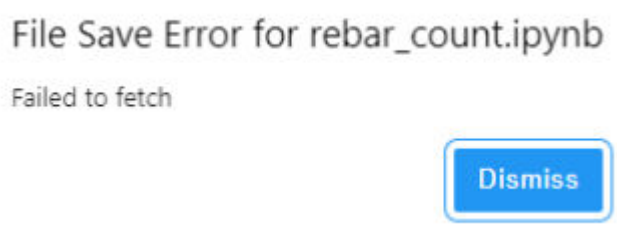
shell语言：

```
dos2unix  文件名
```

14.3.6.7 JupyterLab 中文件保存失败，如何解决?

问题现象

JupyterLab中保存文件时报错如下：



原因分析

- 浏览器安装了第三方插件proxy进行了拦截，导致无法进行保存。
- 在Notebook中的运行文件超过指定大小就会提示此报错。
- jupyter页面打开时间太长。

- 网络环境原因，是否有连接网络代理。

解决方法

- 关掉插件然后重新保存。
- 减少文件大小。
- 重新打开jupyter页面。
- 请检查网络。

14.3.7 VS Code 连接开发环境失败常见问题

14.3.7.1 在 ModelArts 控制台界面上单击 VS Code 接入并在新界面单击打开，未弹出 VS Code 窗口

原因分析

未安装VS Code或者安装版本过低。

解决方法

下载并安装VS Code（Windows用户请单击“Win”，其他用户请单击“其他”下载），安装完成后单击“刷新”完成连接。

14.3.7.2 在 ModelArts 控制台界面上单击 VS Code 接入并在新界面单击打开，VS Code 打开后未进行远程连接

须知

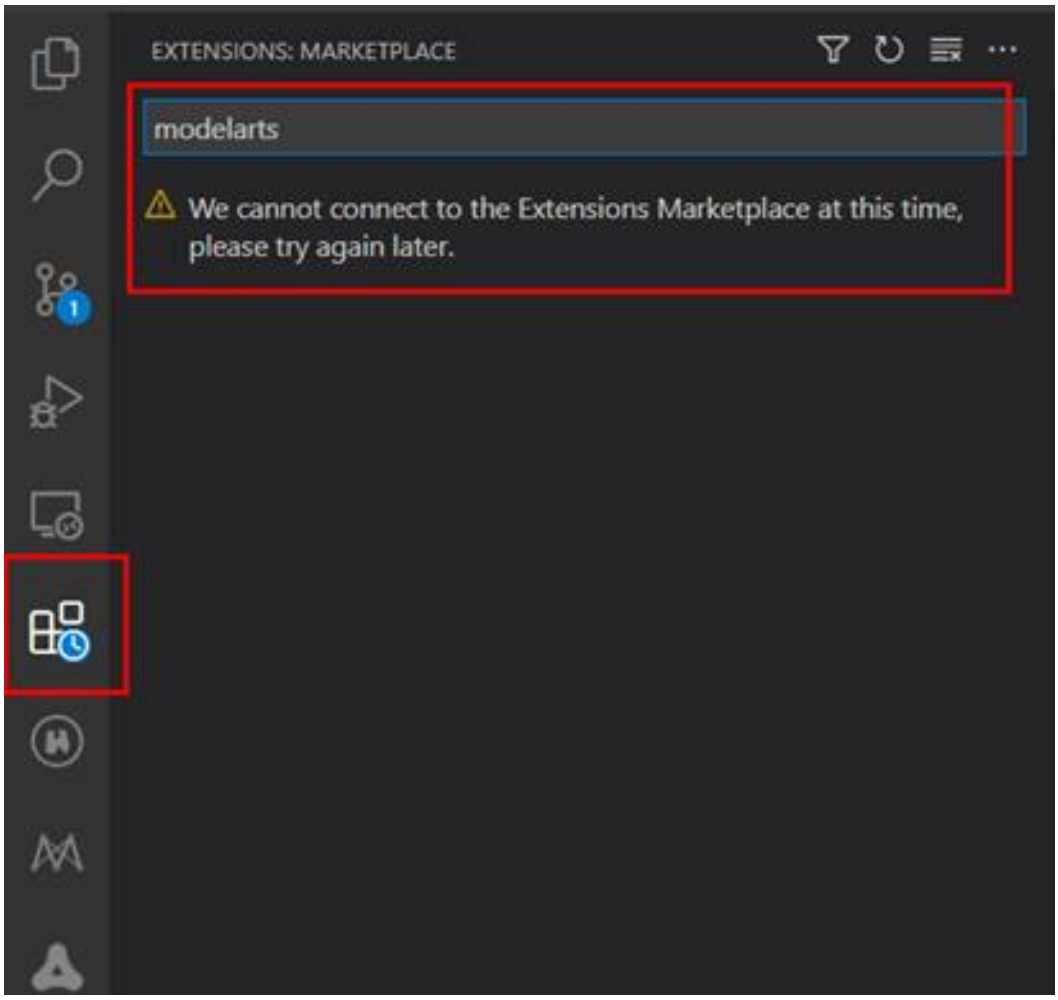
若本地为Linux系统，见原因分析二。

原因分析一

自动安装VS Code插件失败。

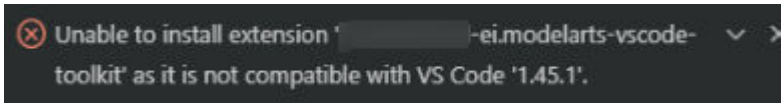
解决方法一

方法一：检查VS Code网络是否正常。在VS Code插件市场上搜索ModelArts，若显示如下则网络异常，请切换代理或使用其他网络。



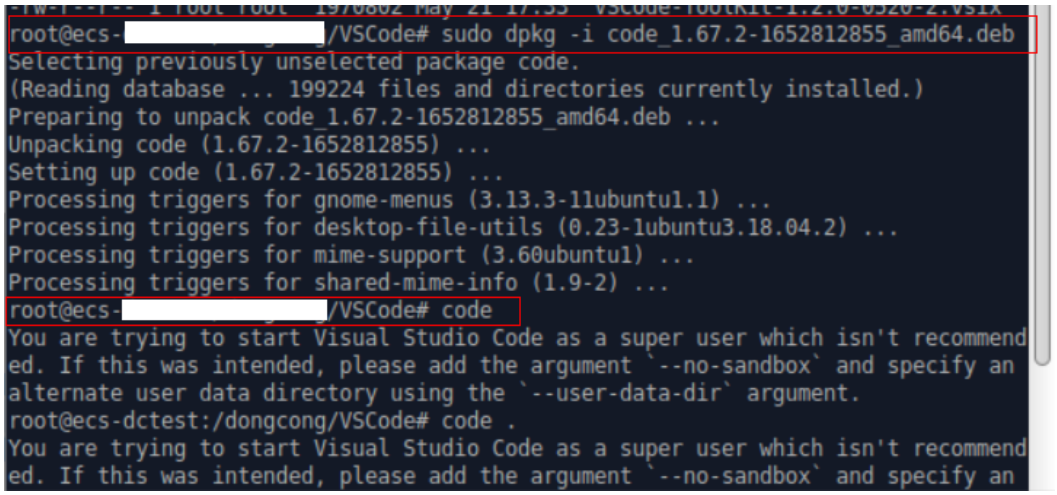
操作完成后再次执行搜索，若显示如下则网络正常，请回到ModelArts控制台界面再次单击界面上的“VS Code接入”按钮。

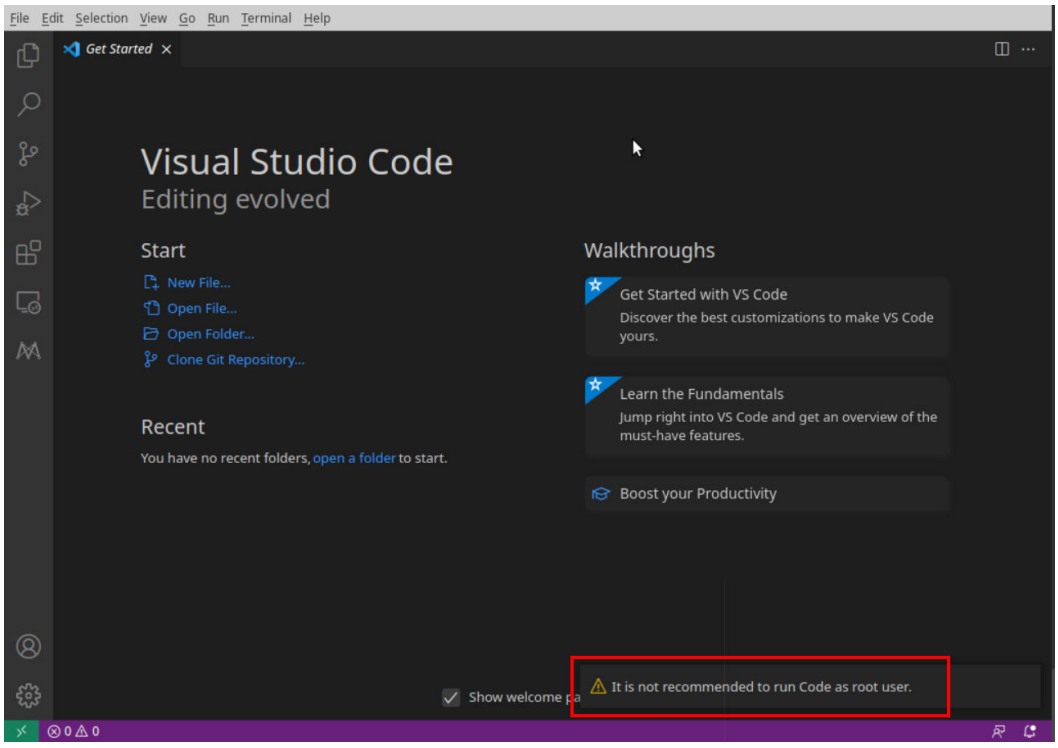
方法二：出现如下图报错，是由于VS Code版本过低，建议升级VS Code版本为1.57.1或者最新版。



原因分析二

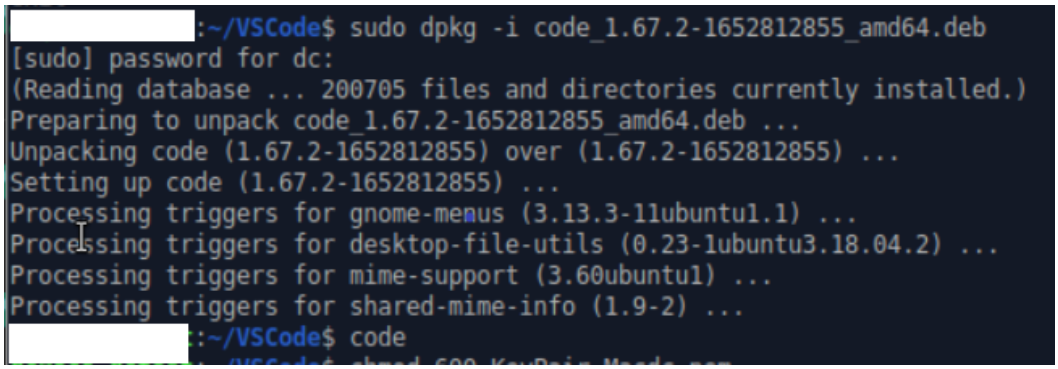
本地系统为Linux，由于使用root用户安装VS Code，打开VS Code显示信息It is not recommended to run Code as root user

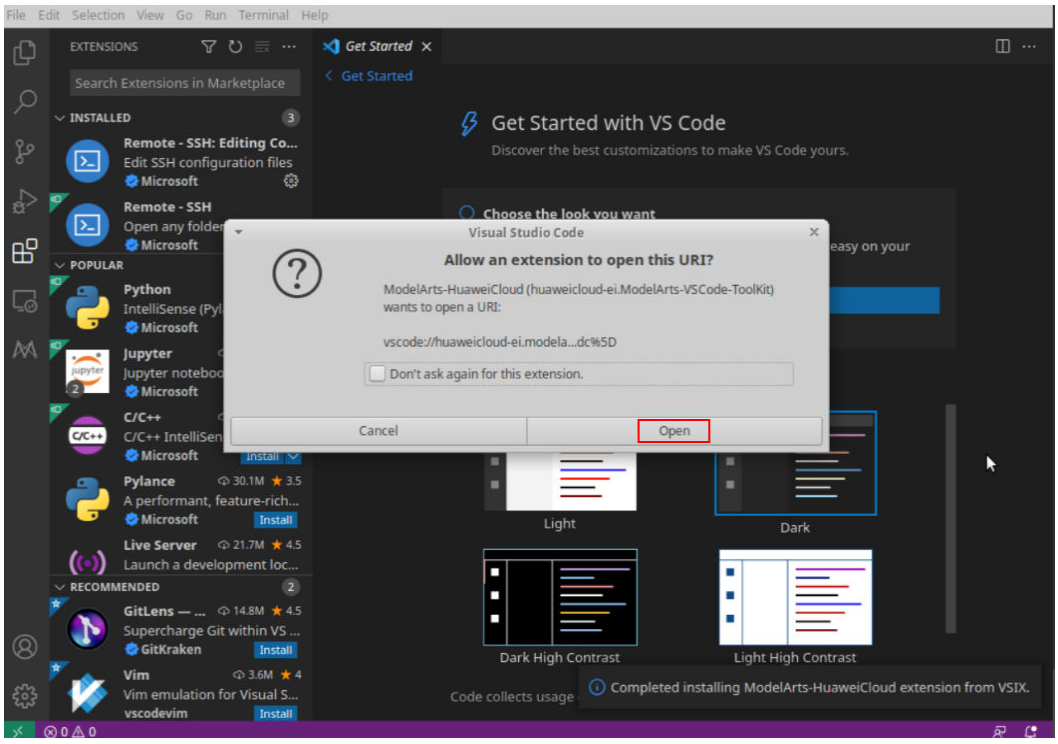




解决方法二

请使用非root用户安装VS Code后，回到ModelArts控制台界面再次单击界面上的“VS Code接入”按钮。

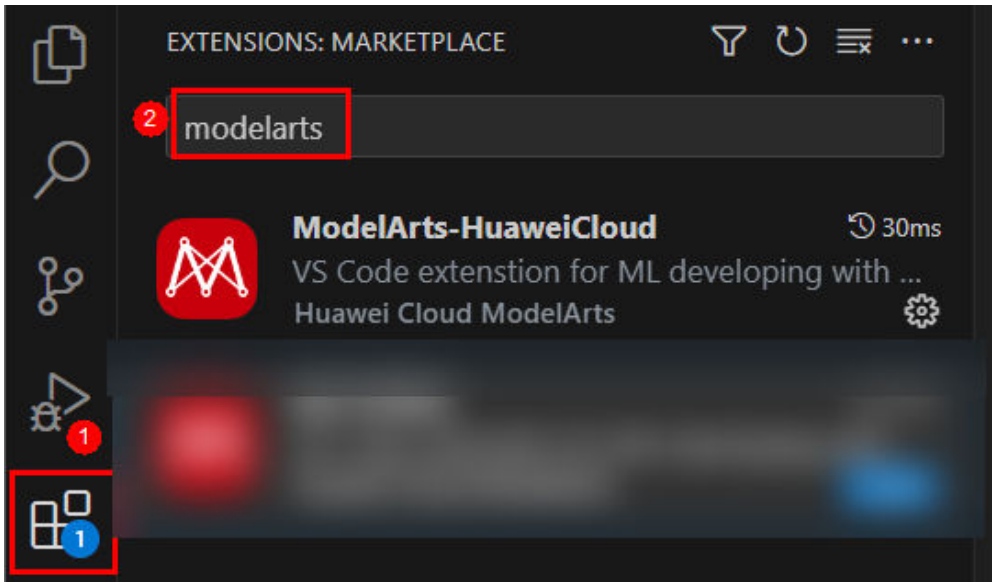




14.3.7.3 VS Code 连接开发环境失败时，请先进行基础问题排查

VS Code连接开发环境失败时，请参考以下步骤进行基础排查：

步骤1 排查插件包是否为最新版：在extensions中搜索，看是否需要升级。



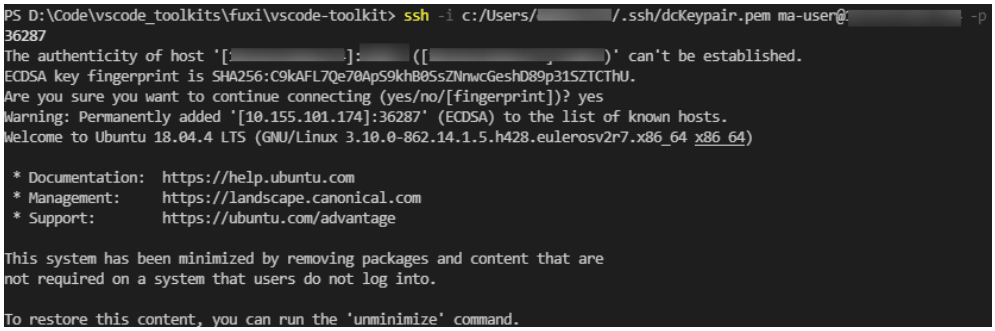
步骤2 检查实例状态是否为运行中，如果是，请执行下一步继续排查。

步骤3 在VS Code的Terminal中执行如下命令，连接到远端开发环境。

```
ssh -tt -o StrictHostKeyChecking=no -i ${IdentityFile} ${User}@${HostName} -p ${Port}
```

参数说明：

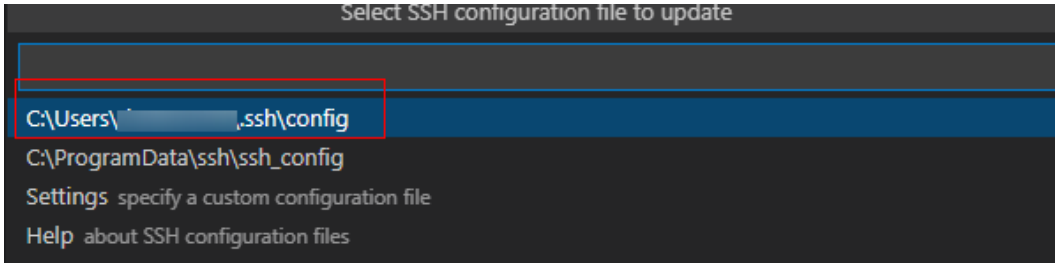
- IdentityFile：本地密钥路径
- User：用户名，例如：ma-user
- HostName：IP地址
- Port：端口号



如可以连接上，请执行下一步继续排查。

步骤4 检查配置是否正确，如果正确请执行下一步继续排查。

打开config文件进行检查，如图所示：



```
HOST remote-dev
    hostname <instance connection host>
    port <instance connection port>
    user ma-user
    IdentityFile ~/.ssh/test.pem
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
    ForwardAgent yes
```

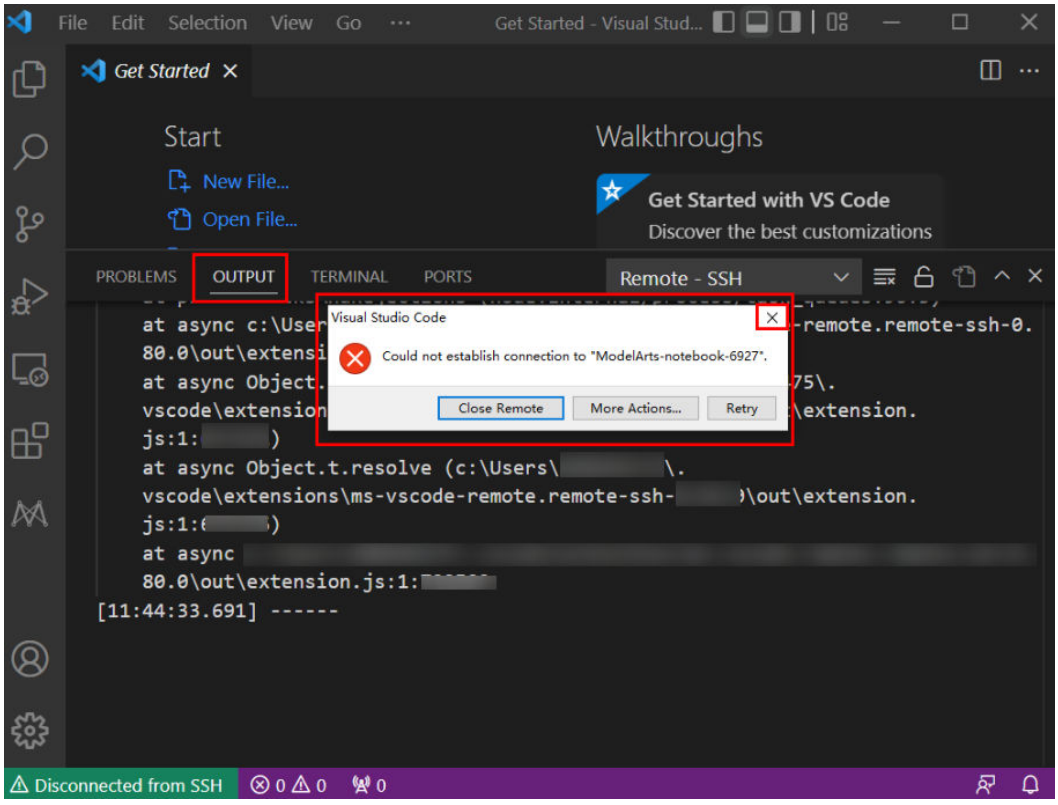
步骤5 查看密钥文件，建议放在C:\Users\xx.ssh下，并确保密钥文件无中文字符。

步骤6 如果还未解决，请参考后续章节的FAQ处理。

----结束

14.3.7.4 远程连接出现弹窗报错：Could not establish connection to xxx

问题现象



原因分析

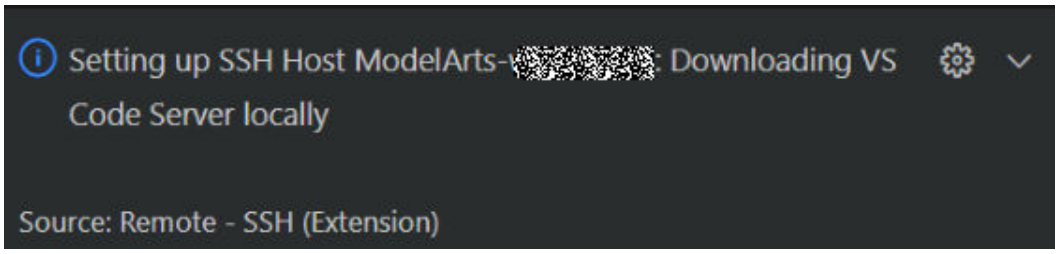
执行VS Code Remote SSH连接失败。

解决方法

单击弹窗右上角关闭弹窗，查看OUTPUT中的具体报错信息，并参考后续章节列举的几种常见报错解决问题。

14.3.7.5 连接远端开发环境时，一直处于"Setting up SSH Host xxx: Downloading VS Code Server locally"超过 10 分钟以上，如何解决?

问题现象



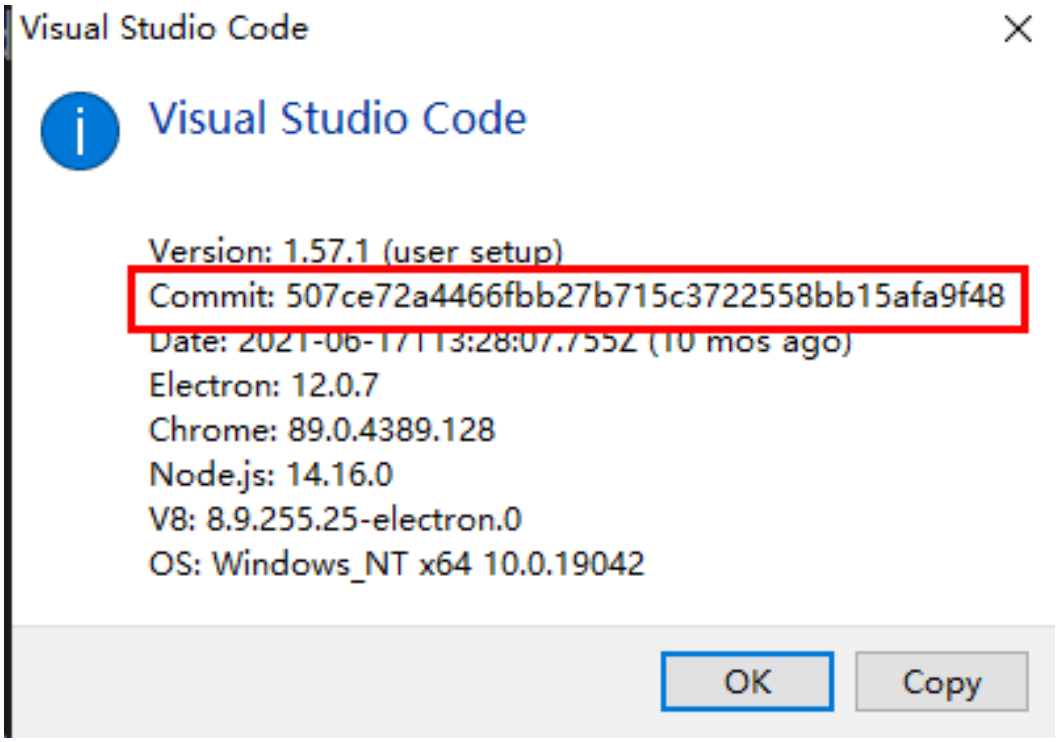
原因分析

当前本地网络原因，导致远程自动安装VS Code Server时间过长。

解决方法

手动安装vscode-server。

步骤1 获取VS Code的commitID



步骤2 下载相应版本vscode-server压缩包，请根据开发环境cpu架构选择arm版本或x86版本。

说明

替换下面链接中${commitID}为**步骤1 获取VS Code的commitID**中commitID。

- arm版本，下载vscode-server-linux-arm64.tar.gz

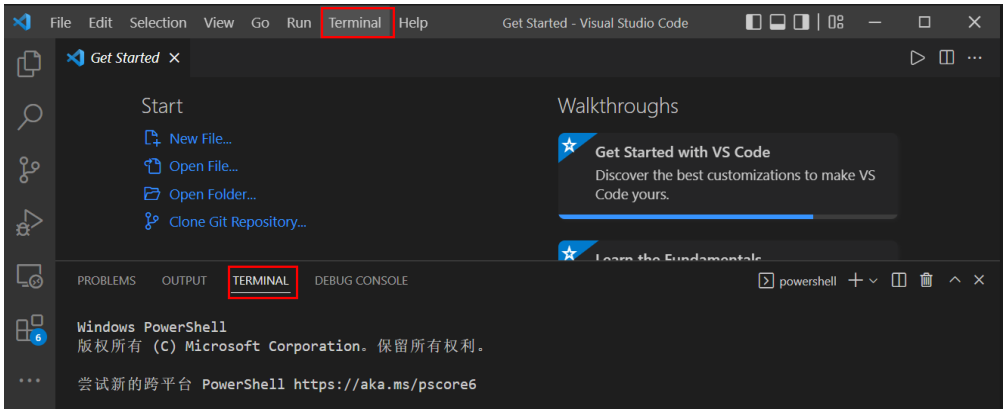
https://update.code.visualstudio.com/commit:${commitID}/server-linux-arm64/stable

- x86版本，下载vscode-server-linux-x64.tar.gz

  https://update.code.visualstudio.com/commit:${commitID}/server-linux-x64/stable

步骤3 进入远程环境。

打开VS Code中的Terminal。



在VS Code的Terminal中执行如下命令，连接到远端开发环境。

```
ssh -tt -o StrictHostKeyChecking=no -i ${IdentityFile} ${User}@${HostName} -p ${Port}
```

参数说明：

- IdentityFile：本地密钥路径

- User：用户名，例如：ma-user

- HostName：IP地址

- Port：端口号

步骤4 手动安装vscode-server。

在VS Code的Terminal中执行如下命令，清空残留的vscode-server，注意替换命令中${commitID}为**步骤1 获取VS Code的commitID**中commitID。

```
rm -rf /home/ma-user/.vscode-server/bin/${commitID}/*
mkdir -p /home/ma-user/.vscode-server/bin/${commitID}
```

上传vscode-server压缩包到开发环境。执行如下命令：

```
exit
scp -i xxx.pem -P 31205 本地vscode-server压缩包路径 ma-user@xxx:/home/ma-user/.vscode-server/bin
ssh -tt -o StrictHostKeyChecking=no -i ${IdentityFile} ${User}@${HostName} -p ${Port}
```

参数说明：

- IdentityFile：本地密钥路径

- User：用户名，例如：ma-user

- HostName：IP地址

- Port：端口号

以arm版本为例，将vscode-server压缩包解压至$HOME/.vscode-server/bin文件夹，注意替换命令中${commitID}为**步骤1 获取VS Code的commitID**中commitID。

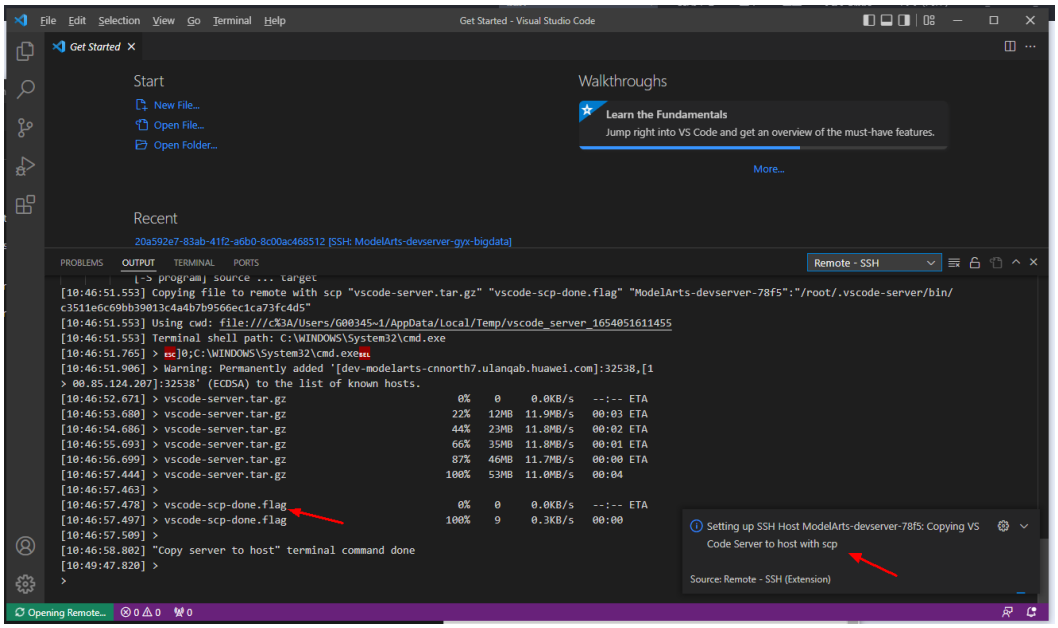
```
cd /home/ma-user/.vscode-server/bin
tar -zxf vscode-server-linux-arm64.tar.gz
mv vscode-server-linux-arm64/* ${commitID}
```

步骤5 重新远程连接。

----结束

14.3.7.6 连接远端开发环境时，一直处于"Setting up SSH Host xxx: Copying VS Code Server to host with scp"超过 10 分钟以上，如何解决?

问题现象



原因分析

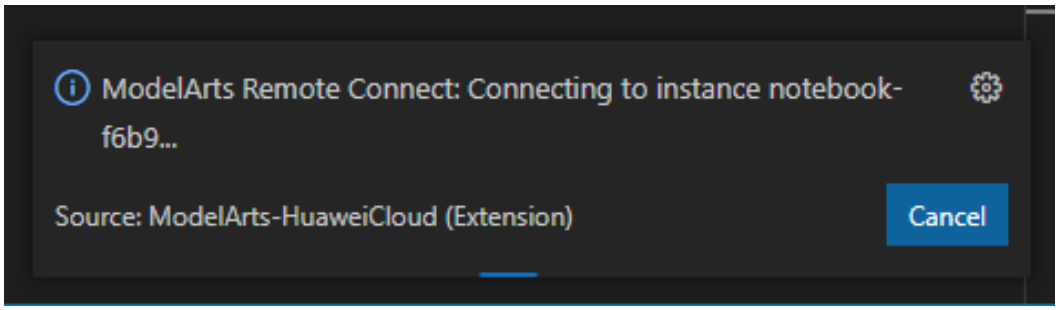
通过查看日志发现本地vscode-scp-done.flag显示成功上传，但远端未接收到。

解决方法

关闭VS Code所有窗口后，回到ModelArts控制台界面再次单击界面上的“VS Code接入”按钮。

14.3.7.7 连接远端开发环境时，一直处于"ModelArts Remote Connect: Connecting to instance xxx..."超过 10 分钟以上，如何解决?

问题现象

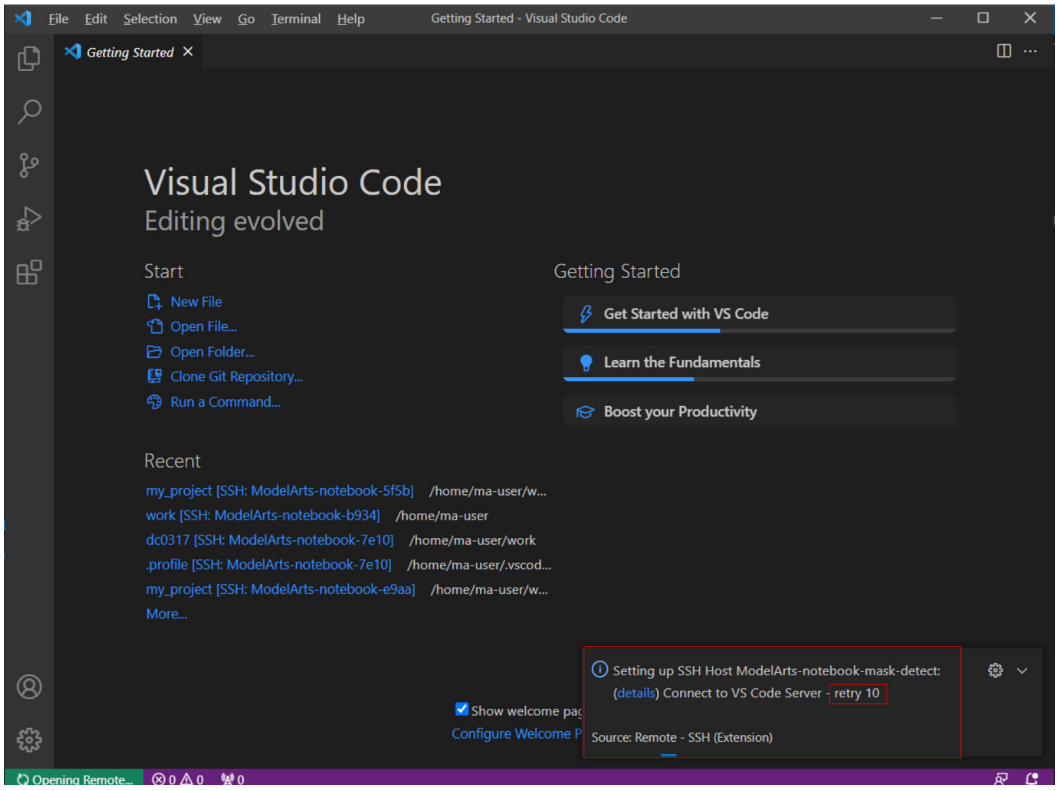


解决方法

单击“Canel”，并回到ModelArts控制台界面再次单击界面上的“VS Code接入”按钮。

14.3.7.8 远程连接处于 retry 状态如何解决?

问题现象



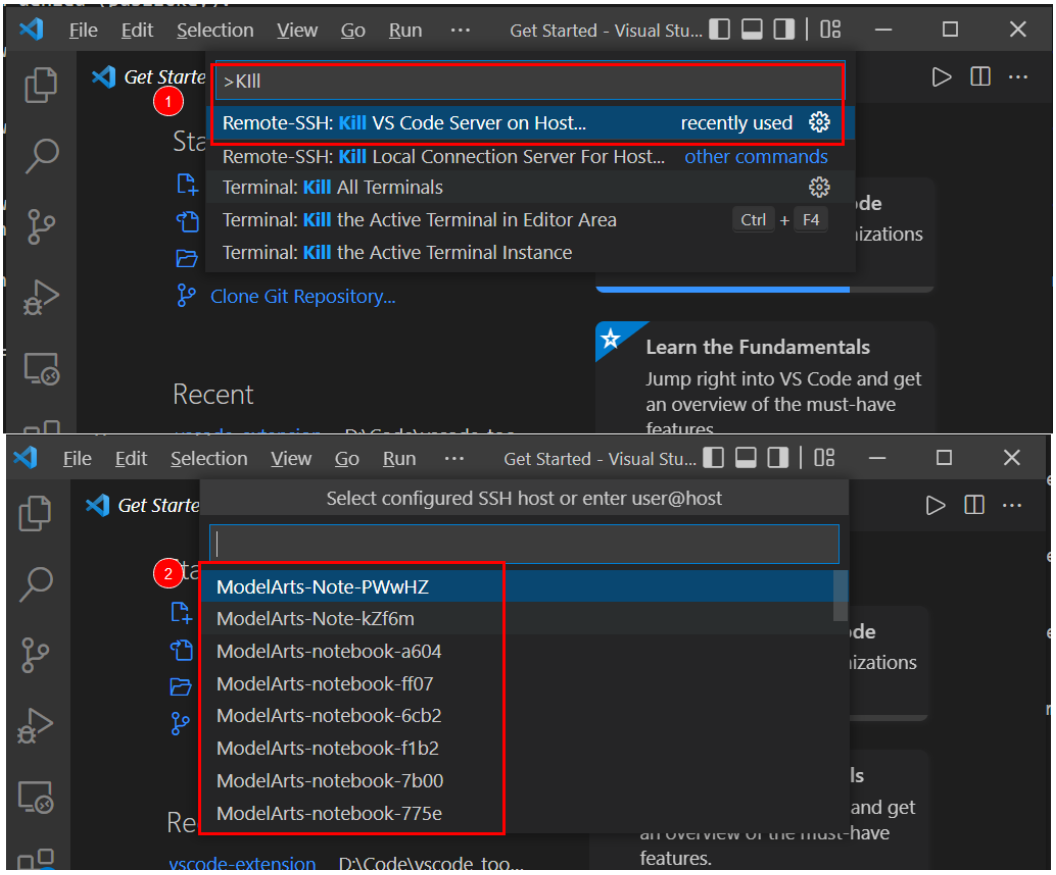
原因分析

之前下载VS Code server失败，有残留信息，导致本次无法下载。

解决方法

方法一（本地）：打开命令面板（Windows： Ctrl+Shift+P，macOS：Cmd+Shift+P），搜索“Kill VS Code Server on Host”，选择出问题的实例进行自动清除，然后重新进行连接。

图 14-17 清除异常的实例



方法二（远端）：在VS Code的Terminal中删除“/home/ma-user/.vscode-server/bin/”下正在使用的文件，然后重新进行连接。

```
ssh -tt -o StrictHostKeyChecking=no -i ${IdentityFile} ${User}@${HostName} -p ${Port}
rm -rf /home/ma-user/.vscode-server/bin/
```

参数说明：

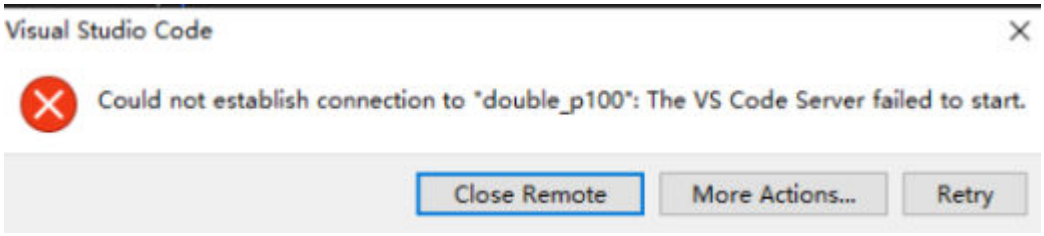
- IdentityFile：本地密钥路径
- User：用户名，例如：ma-user
- HostName：IP地址
- Port：端口号

说明

vscode-server相关问题也可以使用上述的解决方法。

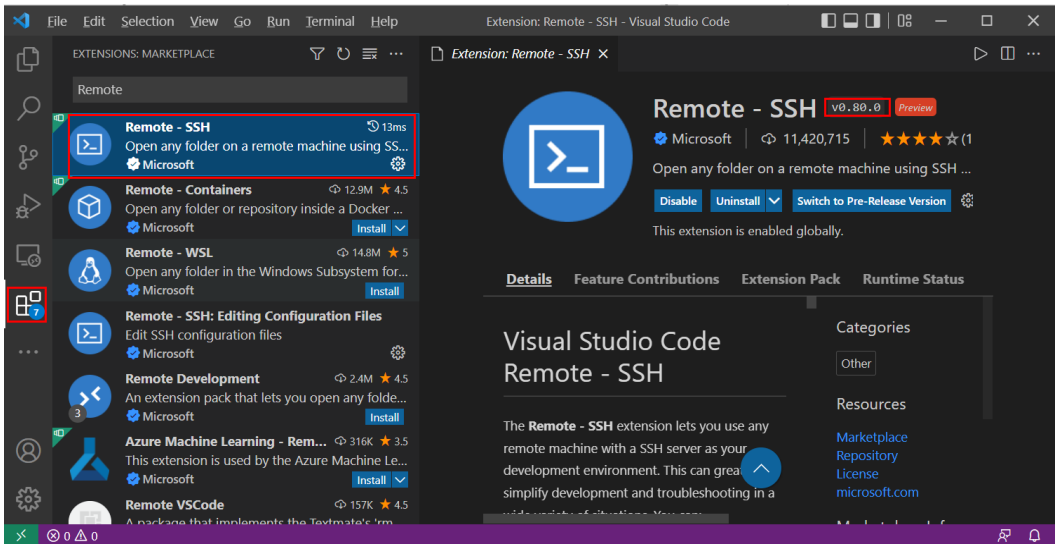
14.3.7.9 报错“The VS Code Server failed to start”如何解决?

问题现象



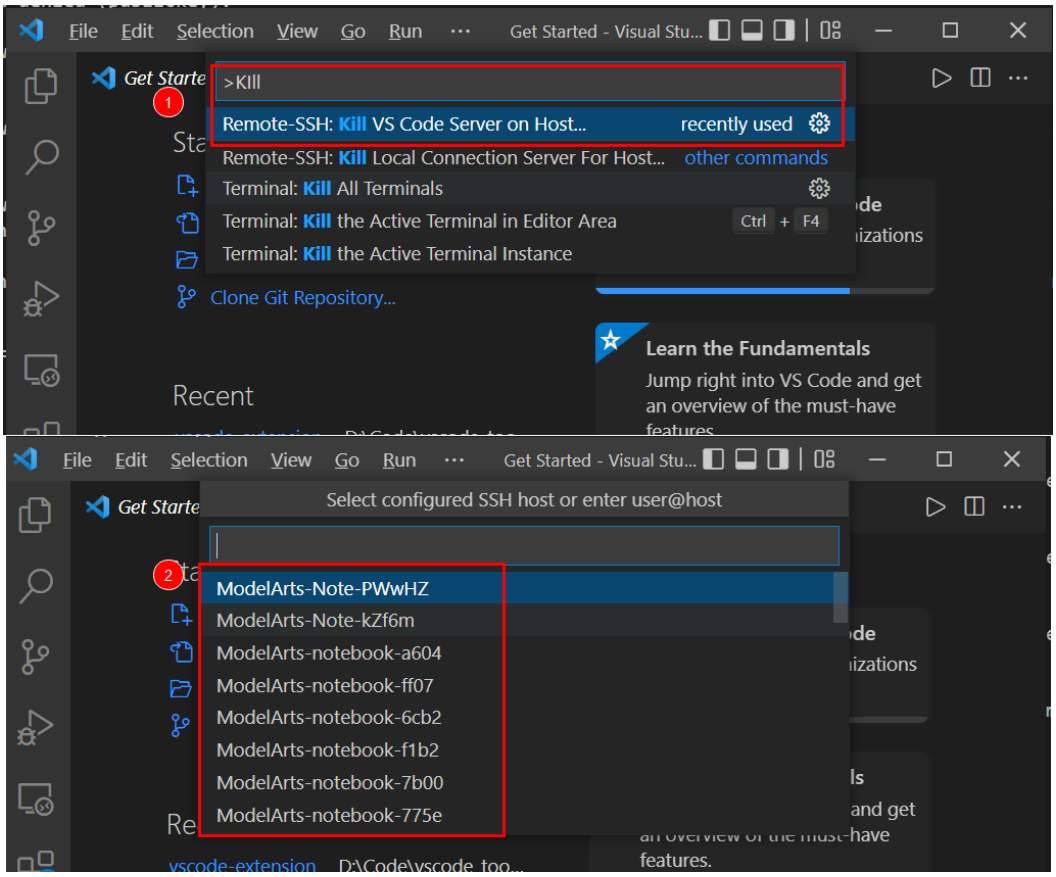
解决方法

步骤1 检查VS Code版本是否为1.78.2或更高版本，如果是，请查看Remote-SSH版本，若低于v0.76.1，请升级Remote-SSH。



步骤2 打开命令面板（Windows： Ctrl+Shift+P，macOS：Cmd+Shift+P），搜索“Kill VS Code Server on Host”，选择出问题的实例进行自动清除，然后重新进行连接。

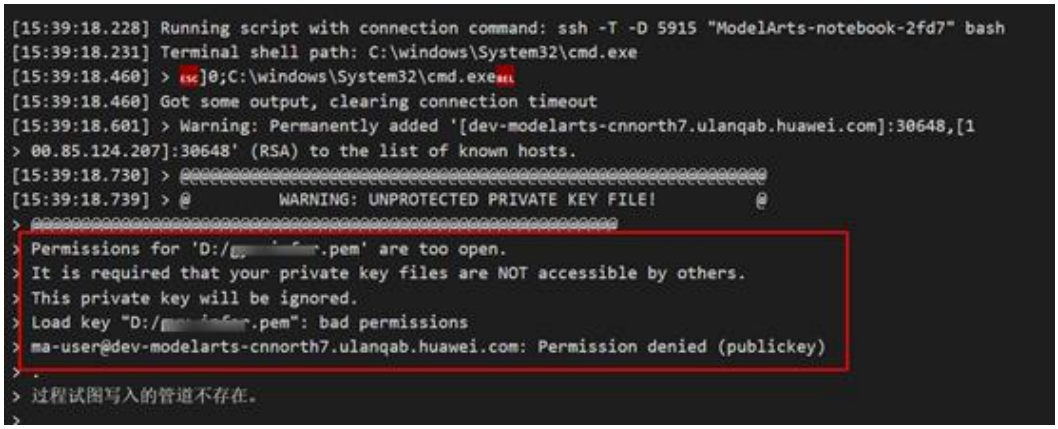
图 14-18 清除异常的实例



----结束

14.3.7.10 报错“Permissions for 'x:/xxx.pem' are too open”如何解决?

问题现象



原因分析

原因分析一：密钥文件未放在指定路径，详情请参考**安全限制**或**VS Code文档**。请参考解决方法一处理。

原因分析二：当操作系统为macOS/Linux时，可能是密钥文件或放置密钥的文件夹权限问题，请参考解决方法二处理。

解决方法

解决方法一：

请将密钥放在如下路径或其子路径下：

Windows：C:\Users\{{user}}

macOS/Linux： Users/{{user}}

解决方法二：

请**检查文件和文件夹权限**。

Local SSH file and folder permissions

macOS / Linux:

On your local machine, make sure the following permissions are set:

Folder / File	Permissions
`.ssh` in your user folder	`chmod 700 ~/.ssh`
`.ssh/config` in your user folder	`chmod 600 ~/.ssh/config`
`.ssh/id_rsa.pub` in your user folder	`chmod 600 ~/.ssh/id_rsa.pub`
Any other key file	`chmod 600 /path/to/key/file`

Windows:

The specific expected permissions can vary depending on the exact SSH implementation you are using. We recommend using the out of box Windows 10 OpenSSH Client.

In this case, make sure that all of the files in the `.ssh` folder for your remote user on the SSH host is owned by you and no other user has permissions to access it. See the Windows OpenSSH wiki for details.

For all other clients, consult your client's documentation for what the implementation expects.

14.3.7.11 报错“Bad owner or permissions on C:\Users\Administrator/.ssh/config”或“Connection permission denied (publickey)”如何解决?

问题现象

报错“Bad owner or permissions on C:\Users\Administrator/.ssh/config”或“Connection permission denied (publickey). Please make sure the key file is correctly selected and the file permission is correct. You can view the instance keypair information on ModelArts console.”

原因分析

文件夹“.ssh”的权限不仅是Windows当前用户拥有，或者当前用户权限不足，故修改权限即可。

解决方案

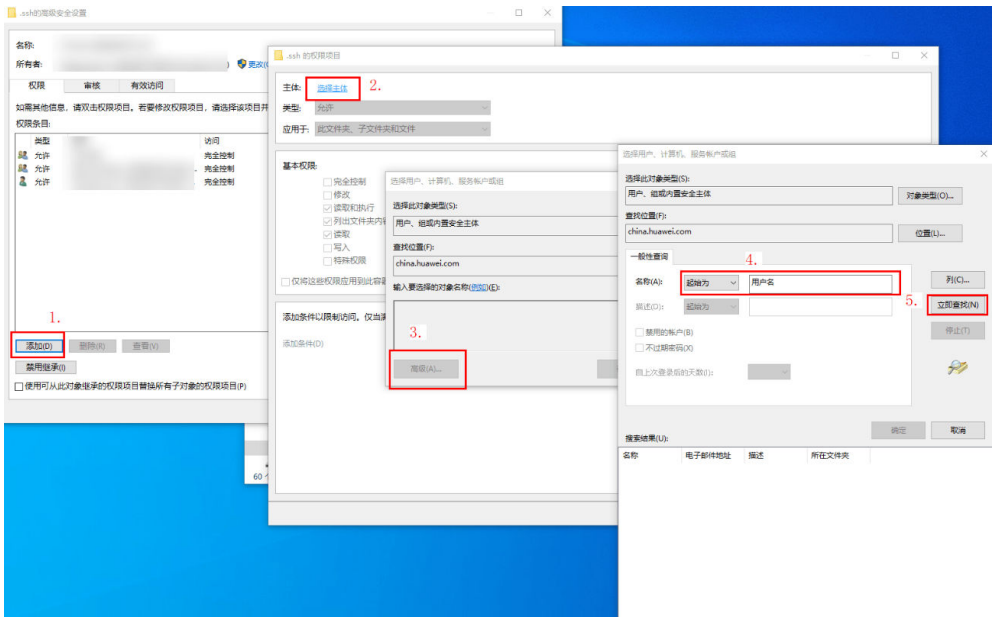
1. 找到.ssh文件夹。一般位于“C:\Users”，例如“C:\Users\xxx”。

   说明

   “C:\Users”目录下的文件名必须和Windows登录用户名完全一致。

2. 右键单击.ssh文件夹，选择“属性”。然后单击“安全”页签。
3. 单击“高级”，在弹出的高级安全设置界面单击“禁用继承”， 在弹出的“阻止继承”窗口单击“从此对象中删除所有继承的权限”。此时所有用户都将被删除。
4. 添加所有者：在同一窗口中，单击“添加”，在弹出的新窗口中，单击“主体”后面的“选择主体”，弹出“选择用户，计算机，服务帐户或组”窗口，单击“高级”，输入用户名， 单击“立即查找”按钮，显示用户搜索结果列表。 选择您的用户帐户，然后单击“确定”（大约四个窗口）以关闭所有窗口。

   图 14-19 添加所有者

5. 完成所有操作后，再次关闭并打开VS Code并尝试连接到远程SSH主机。备注：此时密钥需放到.ssh文件夹中。

14.3.7.12 报错“ssh: connect to host xxx.pem port xxxxx: Connection refused”如何解决?

问题现象

原因分析

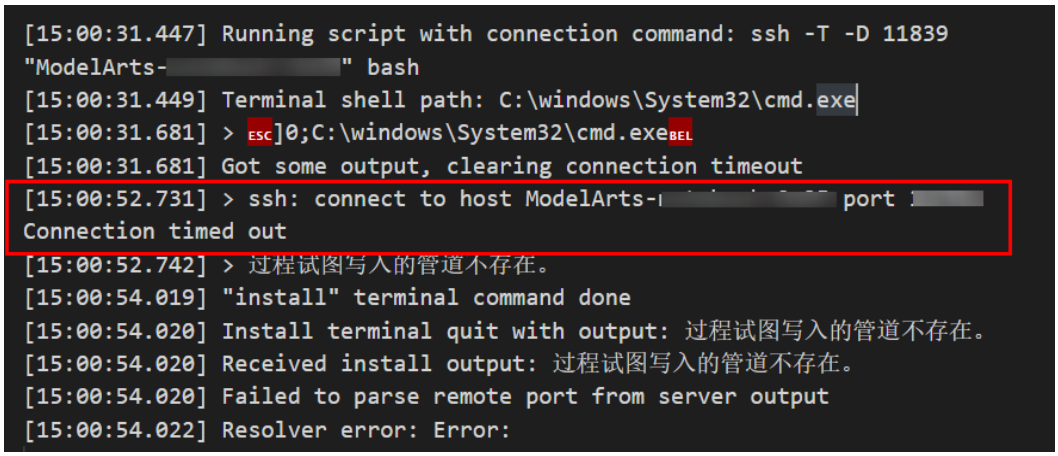
实例处于非运行状态。

解决方法

请前往ModelArts控制台查看实例是否处于运行状态，如果实例已停止，请执行启动操作，如果实例处于其他状态比如“错误”，请尝试先执行停止然后执行启动操作。待实例变为“运行中”后，再次执行远程连接。

14.3.7.13 报错"ssh: connect to host ModelArts-xxx port xxx: Connection timed out"如何解决?

问题现象



原因分析

原因分析一：实例配置的白名单IP与本地网络访问IP不符。

解决方法：请修改白名单为本地网络访问IP或者去掉白名单配置。

原因分析二：本地网络不通。

解决方法：检查本地网络以及网络限制。

14.3.7.14 报错“Load key "C:/Users/xx/test1/xxx.pem": invalid format”如何解决?

问题现象

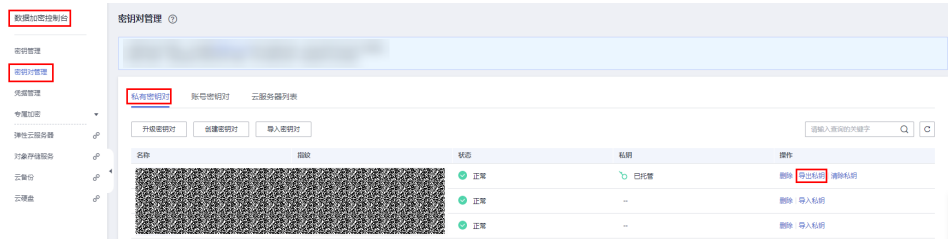
原因分析

密钥文件内容不正确或格式不正确。

解决方法

请使用正确的密钥文件进行远程访问，如果本地没有正确的密钥文件或文件已损坏，可以尝试：

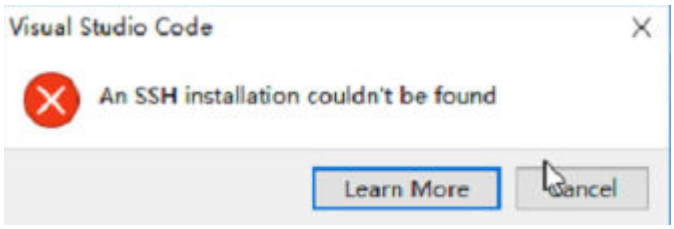
1. 登录控制台，搜索“数据加密服务 DEW”，选择“密钥对管理 > 私钥”页签，查看并下载正确的密钥文件。



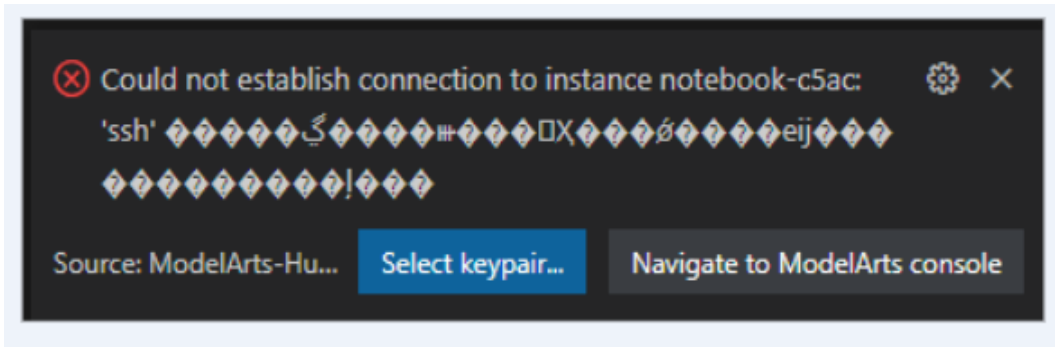
2. 如果密钥不支持下载且已无法找到之前下载的密钥，建议创建新的开发环境实例并创建新的密钥文件。后续版本会支持运行中的开发环境动态更换密钥文件。

14.3.7.15 报错“An SSH installation couldn't be found”或者“Could not establish connection to instance xxx: 'ssh' ...”如何解决?

问题现象



或



VS Code连接Notebook一直提示选择证书，且提示信息除标题外，都是乱码。选择证书后，如上图所示仍然没有反应且无法进行连接。

原因分析

当前环境未装OpenSSH或者OpenSSH未安装在默认路径下，详情请参考**VS Code文档**。

解决方法

- 若当前环境未安装OpenSSH，请**下载并安装OpenSSH**。

  Installing a supported SSH client

OS	Instructions
Windows 10 1803+ / Server 2016/2019 1803+	Install the Windows OpenSSH Client.
Earlier Windows	Install Git for Windows.
macOS	Comes pre-installed.
Debian/Ubuntu	Run `sudo apt-get install openssh-client`
RHEL / Fedora / CentOS	Run `sudo yum install openssh-clients`

  VS Code will look for the `ssh` command in the PATH. Failing that, on Windows it will attempt to find `ssh.exe` in the default Git for Windows install path. You can also specifically tell VS Code where to find the SSH client by adding the `remote.SSH.path` property to `settings.json`.

  当通过“可选功能”未能成功安装时，请手动**下载OpenSSH安装包**，然后执行以下步骤：

步骤1 下载zip包并解压放入“C:\Windows\System32”。

步骤2 以管理员身份打开CMD，在“C:\Windows\System32\OpenSSH-xx”目录下，执行以下命令：

```
powershell.exe -ExecutionPolicy Bypass -File install-sshd.ps1
```

步骤3 添加环境变量：将“C:\Program Files\OpenSSH-xx”（路径中包含ssh可执行exe文件）添加到环境系统变量中。

步骤4 重新打开CMD，并执行ssh，结果如下图即说明安装成功，如果还未装成功则执行**5**和**6**。

```
C:\windows\system32>ssh
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command]
```

步骤5 OpenSSH默认端口为22端口，开启防火墙22端口号，在CMD执行以下命令：

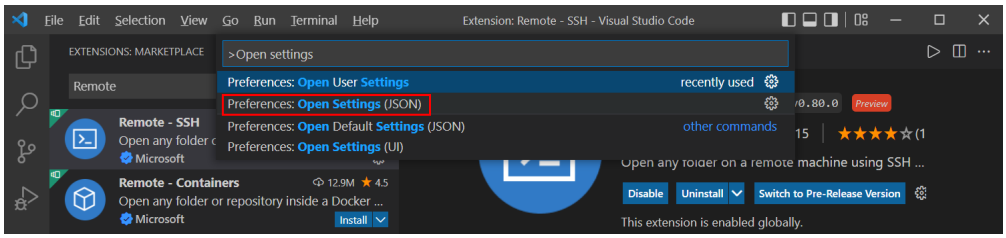
```
netsh advfirewall firewall add rule name=sshd dir=in action=allow protocol=TCP localport=22
```

步骤6 启动OpenSSH服务，在CMD执行以下命令：

```
Start-Service sshd
```

----结束

- 若OpenSSH未安装在默认路径下，打开命令面板（Windows： Ctrl+Shift+P，macOS：Cmd+Shift+P），
  搜索“Open settings”。

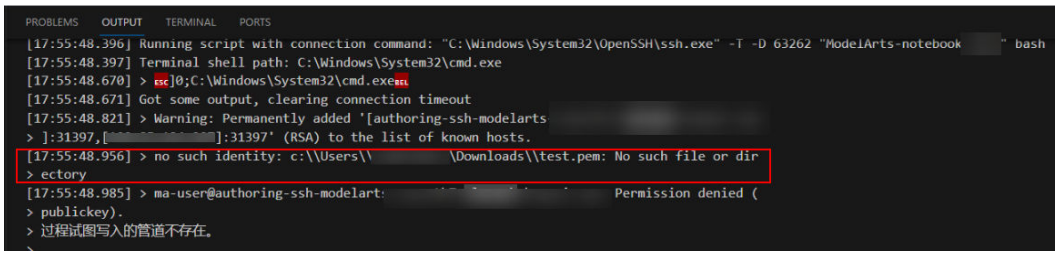


然后将remote.SSH.path属性添加到settings.json中，例如："remote.SSH.path": "*本地OpenSSH的安装路径*"

```
{
    "extensions.autoCheckUpdates": false,
    "extensions.autoUpdate": false,
    "remote.SSH.remotePlatform": {
        "ModelArts-notebook-                ": "linux"
    },
    "remote.SSH.path": "D:/OpenSSH-Win64/ssh.exe"
}
```

14.3.7.16 报错“no such identity: C:/Users/xx /test.pem: No such file or directory”如何解决?

问题现象



原因分析

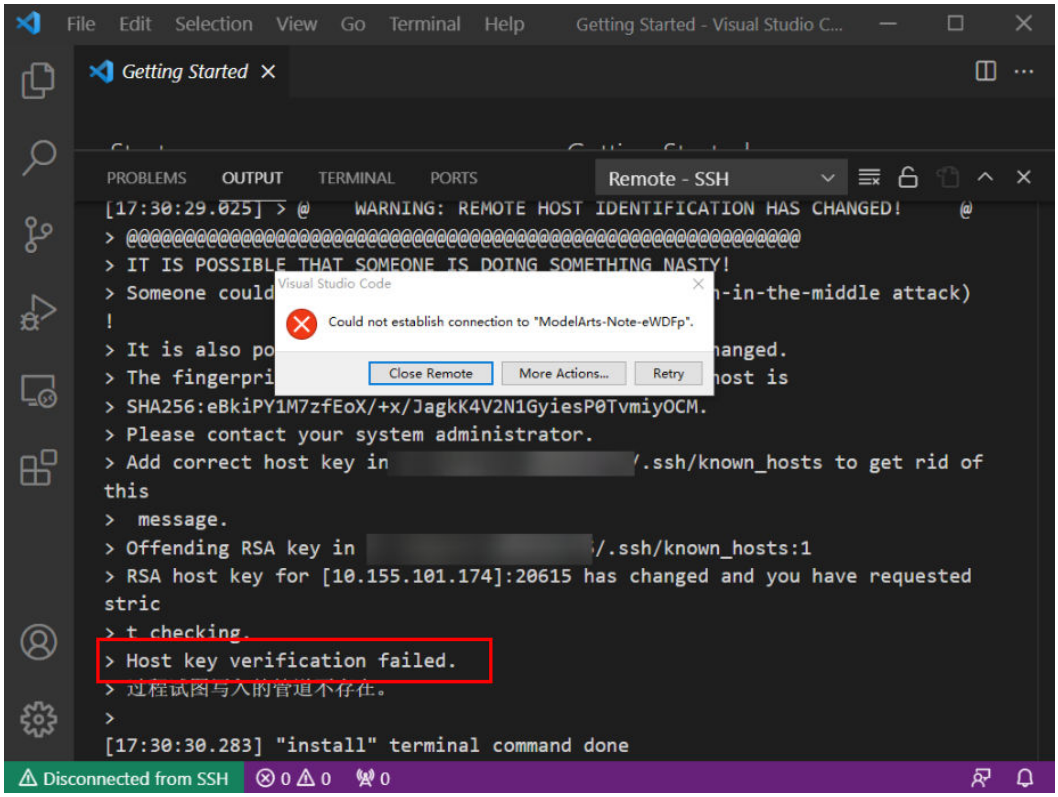
密钥文件不存在于该路径下，或者该路径下密钥文件名被修改。

解决方法

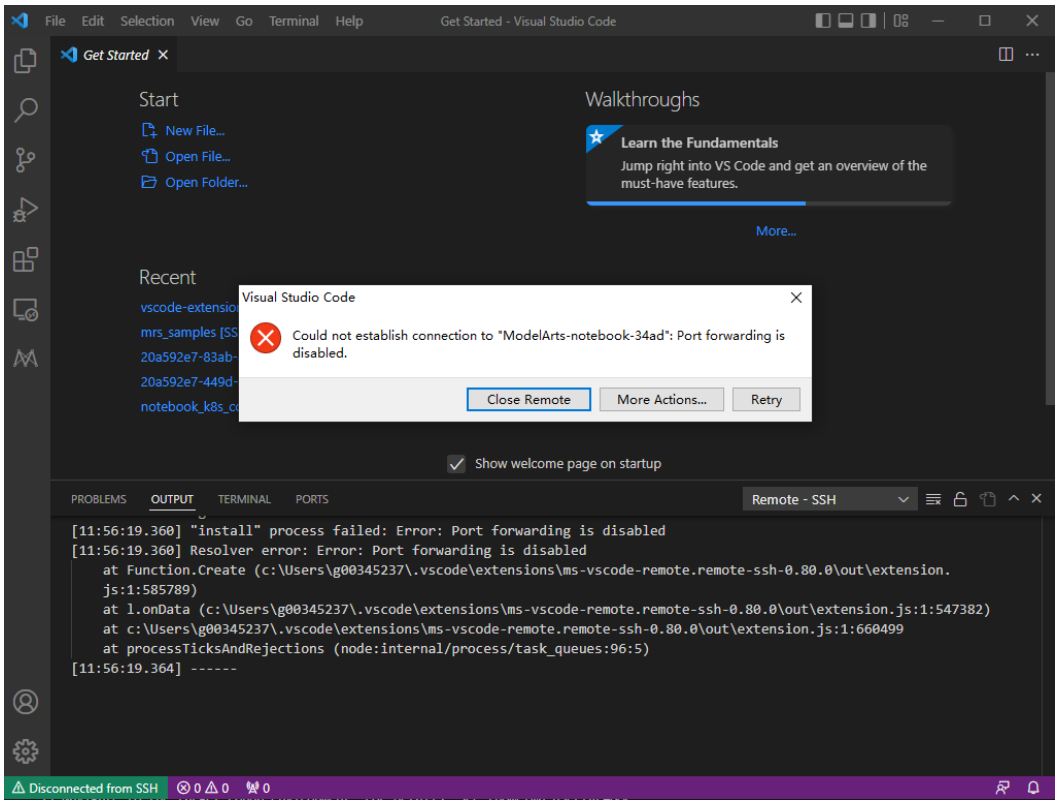
重新选择密钥路径。

14.3.7.17 报错“Host key verification failed.'或者'Port forwarding is disabled.”如何解决?

问题现象



或



原因分析

Notebook实例重新启动后，公钥发生变化，OpenSSH核对公钥发出警告。

解决方法

- 在VS Code中使用命令方式进行远程连接时，增加参数"-o StrictHostKeyChecking=no"

  ```
  ssh -tt -o StrictHostKeyChecking=no -i ${IdentityFile} ${User}@${HostName} -p ${Port}
  ```

  参数说明：

  - IdentityFile：本地密钥路径
  - User：用户名，例如：ma-user
  - HostName：IP地址
  - Port：端口号

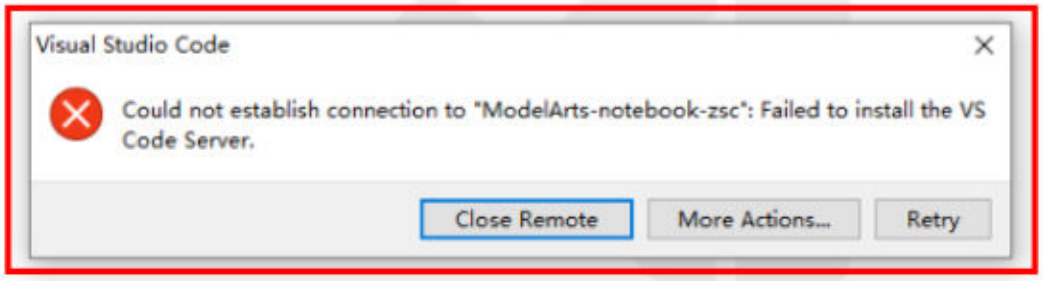
- 在VS Code中手工配置远程连接时，在本地的ssh config文件中增加配置参数“StrictHostKeyChecking no”和“UserKnownHostsFile=/dev/null”

  ```
  Host xxx
    HostName x.x.x.x #IP地址
    Port 22522
    User ma-user
    IdentityFile C:/Users/my.pem
    StrictHostKeyChecking no
    UserKnownHostsFile=/dev/null
    ForwardAgent yes
  ```

提示：增加参数后SSH登录时会忽略known_hosts文件，有安全风险。

14.3.7.18 报错“Failed to install the VS Code Server.”或“tar: Error is not recoverable: exitng now.”如何解决?

问题现象



或

```
[17:53:24.382] > vscode-scp-done.flag                          100%    9     0.2KB/s   00:00
[17:53:24.756] > Found flag and server on host
[17:53:24.765] > d3aeabcaa9c5%%2%%
> tar --version:
[17:53:24.789] > tar (GNU tar) 1.30
> Copyright (C) 2017 Free Software Foundation, Inc.
> License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
> This is free software: you are free to change and redistribute it.
> There is NO WARRANTY, to the extent permitted by law.
>
> Written by John Gilmore and Jay Fenlason.
[17:53:24.796] > tar: This does not look like a tar archive
>
> gzip: stdin: unexpected end of file
> tar: Child returned status 1
> tar: Error is not recoverable: exiting now
[17:53:24.804] >
> ERROR: tar exited with non-0 exit code: 0
> Already attempted local download, failing
> d3aeabcaa9c5: start
> exitCode==37==
```

原因分析

可能为/home/ma-user/work磁盘空间不足。

解决方法

删除/home/ma-user/work路径下无用文件。

14.3.7.19 VS Code 连接远端 Notebook 时报错如“XHR failed”

原因分析

可能是所在环境的网络问题，请排查。

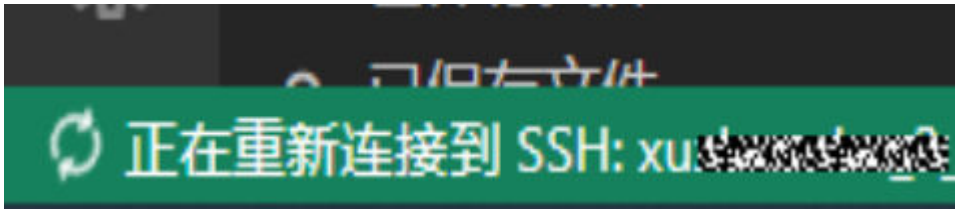
解决方法

参见**XHR failed常见排查思路**，进行排查。

14.3.7.20 VS Code 连接后长时间未操作，连接自动断开

问题现象

VS Code SSH连接后，长时间未操作，窗口未关闭，再次使用发现VS Code在重连环境，无弹窗报错。左下角显示如下图：



查看VS Code Remote-SSH日志发现，连接在大约2小时后断开了：

```
>
[21:32:39.136] Got some output, clearing connection timeout
[21:48:58.059] > 这会是正常的
[21:49:12.060] >
[22:40:58.740] >
>                这会断开了
[23:32:49.341] > Connection reset by 139.159.152.36 port 32528
>
```

原因分析

用户SSH交互操作停止后一段时间，防火墙对空闲链接进行了断开操作，SSH默认配置中不存在超时主动断连的动作，但是防火墙会关闭超时空闲连接（参考：http://bluebiu.com/blog/linux-ssh-session-alive.html），后台的实例运行是一直稳定的，重连即可再次连上。

解决方法

如果想保持长时间连接不断开，可以通过配置SSH定期发送通信消息，避免防火墙认为链路空闲而关闭。

- 客户端配置（用户可根据需要自行配置，不配置默认是不给服务端发心跳包），如图1，图2所示。

  图 14-20 打开 VS Code ssh config 配置文件

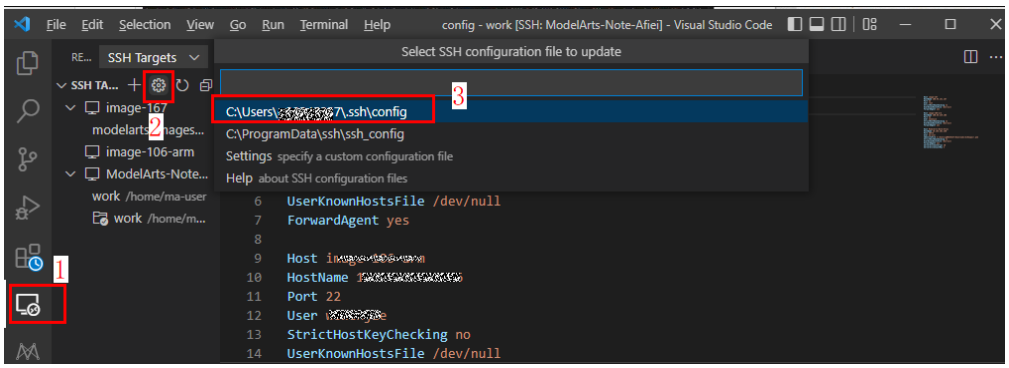
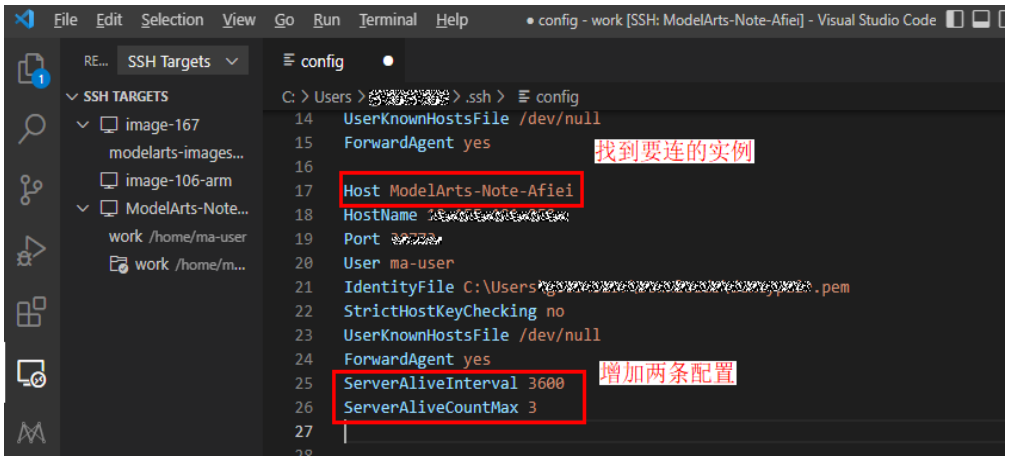


图 14-21 增加配置信息



配置信息示例如下：

```
Host ModelArts-xx
    ......
    ServerAliveInterval 3600  # 增加这个配置，单位是秒，每1h向服务端主动发个包
    ServerAliveCountMax 3  # 增加这个配置，3次发包均无响应会断开连接
```

比如防火墙配置是2小时空闲就关闭连接，那我们客户端配置ServerAliveInterval小于2小时（比如1小时），就可以避免防火墙将连接断开。

- 服务器端配置（Notebook当前已经配置，24h应该是长于防火墙的断连时间配置，该配置无需用户手工修改，写在这里仅是帮助理解ssh配置原理）配置文件路径：/home/ma-user/.ssh/etc/sshd_config

```
/modelarts/authoring(MindSpore) [ma-user work]$cat /home/ma-user/.ssh/etc/sshd_config |grep Client
ClientAliveInterval 1440m
ClientAliveCountMax 3
```

每24h向client端主动发个包，3次发包均无响应会断开连接

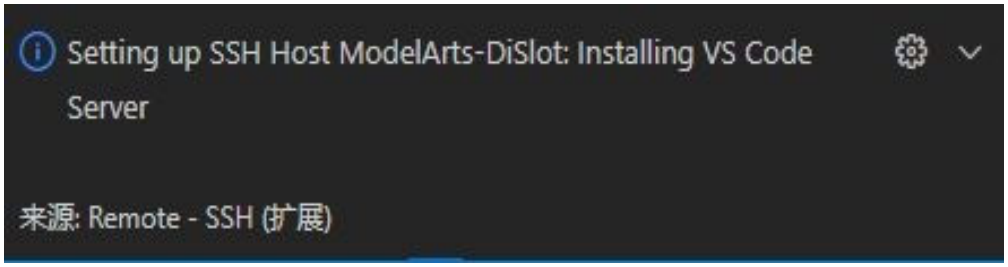
参考：https://unix.stackexchange.com/questions/3026/what-do-options-serveraliveinterval-and-clientaliveinterval-in-sshd-config-d

- 对于业务有影响的需要进行长链接保持的场景，尽量将日志写在单独的日志文件中，将脚本后台运行，例如：

```
nohup train.sh > output.log 2>&1 & tail -f output.log
```

14.3.7.21 VS Code 自动升级后，导致远程连接时间过长

问题现象



原因分析

由于VS Code自动升级，导致连接时需要重新下载新版vscode-server。

解决方法

禁止VS Code自动升级。单击左下角选择Settings项，搜索Update: Mode，将其设置为none。

图 14-22 打开 Settings

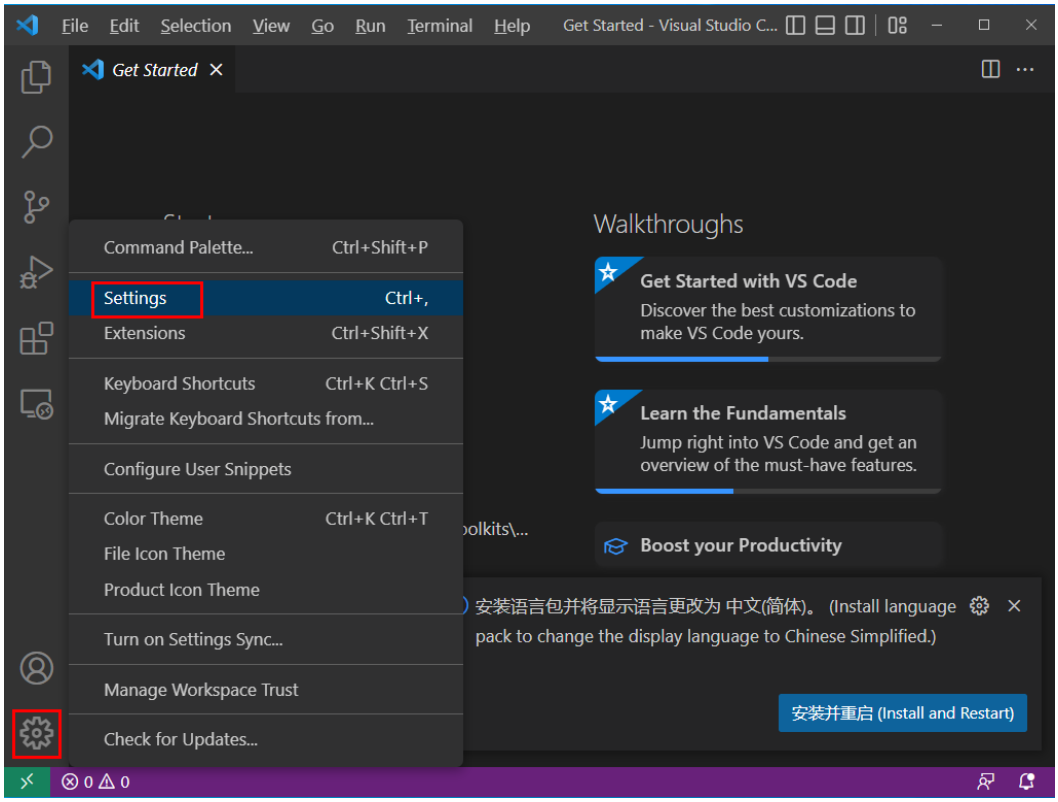
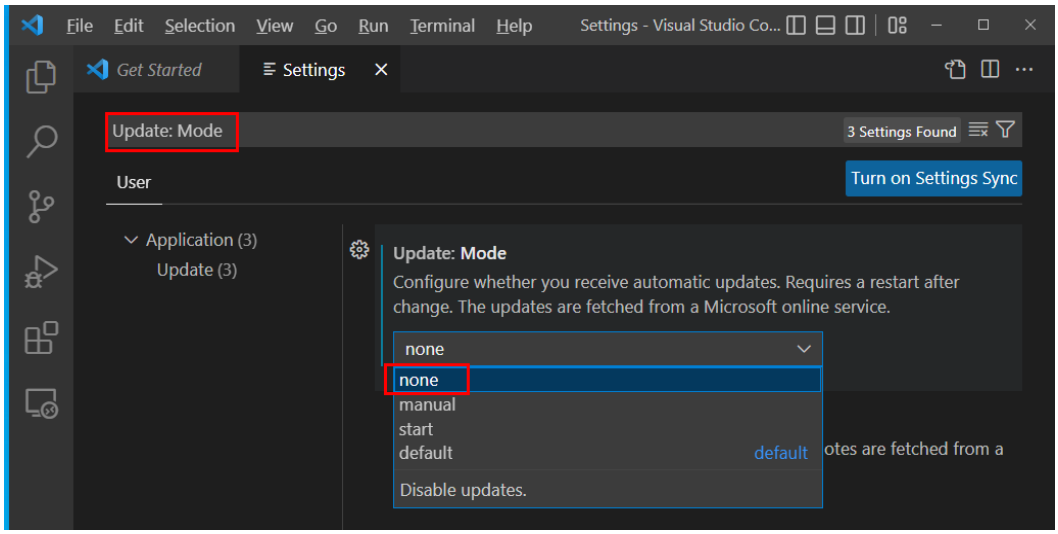
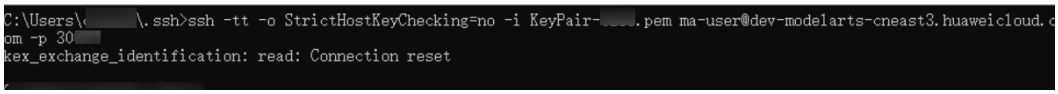


图 14-23 设置“Update: Mode”为“none”



14.3.7.22 使用 SSH 连接，报错“Connection reset”如何解决?

问题现象



原因分析

可能是用户网络限制原因。比如部分企业网络的SSH是默认屏蔽的。

解决方法

用户重新进行申请SSH权限。

14.3.7.23 使用 MobaXterm 工具 SSH 连接 Notebook 后，经常断开或卡顿，如何解决?

问题现象

MobaXterm成功连接到开发环境后，过一段时间会自动断开。

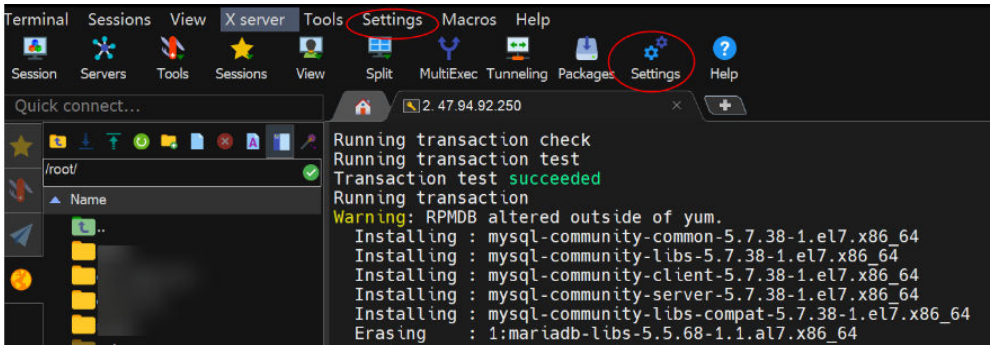
可能原因

配置MobaXterm工具时，没有勾选“SSH keepalive”或专业版MobaXterm工具的“Stop server after”时间设置太短。

解决方案

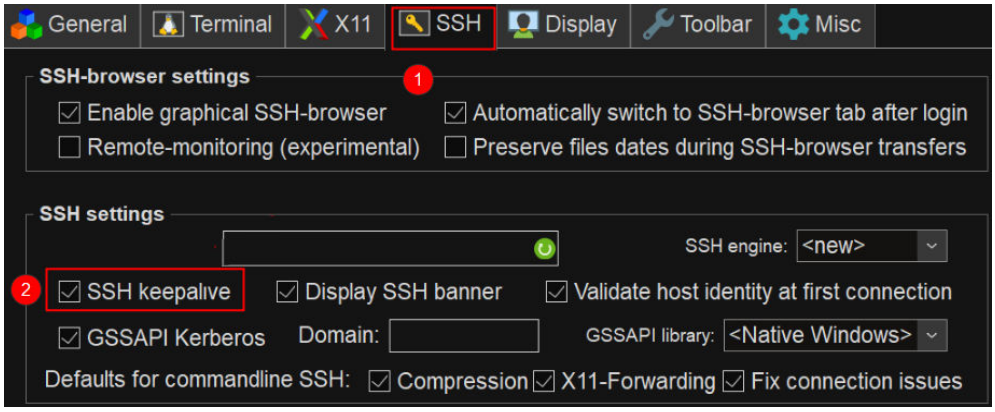
步骤1 打开MobaXterm，单击菜单栏“Settings”，如**图1 打开“Settings”**所示。

图 14-24 打开“Settings”



步骤2 在打开的“MobaXterm Configuration”配置页面，选择“SSH”选项卡，勾选“SSH keepalive”，如**图2 勾选“SSH keepalive”**所示。

图 14-25 勾选“SSH keepalive”

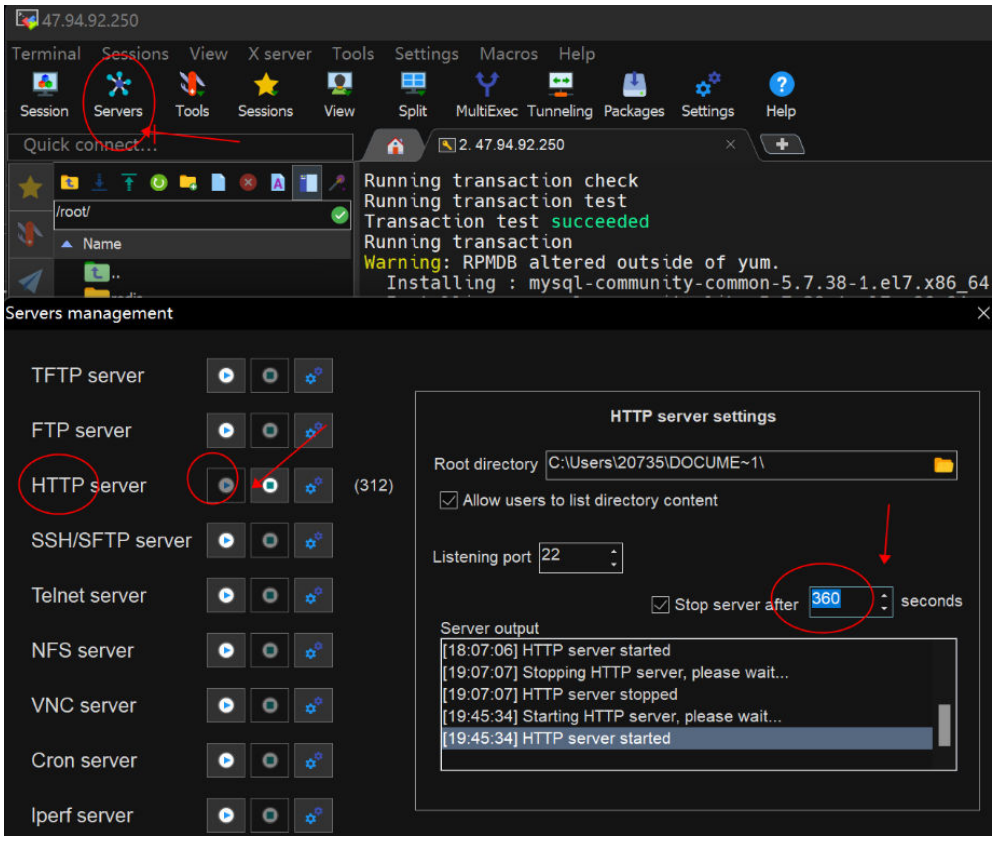


说明

如果使用的是专业版的MobaXterm工具，请执行**步骤3**。

步骤3 如果使用的是专业版的MobaXterm工具，请参考**图3 设置“Stop server after”**，此参数默认值为360s，将其设置为3600s或更大值。

图 14-26 设置“Stop server after”



----结束

14.3.8 更多功能咨询

14.3.8.1 在 Notebook 中，如何使用昇腾多卡进行调试?

昇腾多卡训练任务是多进程多卡模式，跑几卡需要起几个python进程。昇腾底层会读取环境变量：RANK_TABLE_FILE，开发环境已经设置，用户无需关注。比如跑八卡，可以如下片段代码：

```
export RANK_SIZE=8
current_exec_path=$(pwd)
echo 'start training'
for((i=0;i<=$RANK_SIZE-1;i++));
do
echo 'start rank '$i
mkdir ${current_exec_path}/device$i
cd ${current_exec_path}/device$i
echo $i
export RANK_ID=$i
dev=`expr $i + 0`
echo $dev
export DEVICE_ID=$dev
python train.py > train.log 2>&1 &
done
```

其中，train.py中设置环境变量DEVICE_ID：

```
devid = int(os.getenv('DEVICE_ID'))
context.set_context(mode=context.GRAPH_MODE, device_target="Ascend", device_id=devid)
```

14.3.8.2 使用 Notebook 不同的资源规格，为什么训练速度差不多?

如果用户的代码中训练任务是单进程的，使用Notebook 8核64GB，72核512GB训练的速度是基本一致的，例如用户用的是2核4GB的资源，使用4核8GB，或者8核64GB效果是一样的。

如果用户的代码中训练任务是多进程的，使用Notebook 72核512GB训练速度要优于8核64GB。

14.3.8.3 使用 MoXing 时，如何进行增量训练?

在使用MoXing构建模型时，如果您对前一次训练结果不满意，可以在更改部分数据和标注信息后，进行增量训练。

“mox.run”添加增量训练参数

在完成标注数据或数据集的修改后，您可以在“mox.run”中，修改“log_dir”参数，并新增“checkpoint_path”参数。其中“log_dir”参数建议设置为一个新的目录，“checkpoint_path”参数设置为上一次训练结果输出路径，如果是OBS目录，路径填写时建议使用“obs://”开头。

如果标注数据中的标签发生了变化，在运行“mox.run”前先执行**如果标签发生变化**的操作。

```
mox.run(input_fn=input_fn,
        model_fn=model_fn,
        optimizer_fn=optimizer_fn,
        run_mode=flags.run_mode,
        inter_mode=mox.ModeKeys.EVAL if use_eval_data else None,
        log_dir=log_dir,
        batch_size=batch_size_per_device,
        auto_batch=False,
        max_number_of_steps=max_number_of_steps,
```

```
    log_every_n_steps=flags.log_every_n_steps,
    save_summary_steps=save_summary_steps,
    save_model_secs=save_model_secs,
    checkpoint_path=flags.checkpoint_url,
    export_model=mox.ExportKeys.TF_SERVING)
```

如果标签发生变化

当数据集中的标签发生变化时，需要执行如下语句。此语句需在“mox.run”之前运行。

语句中的“logits”，表示根据不同网络中分类层权重的变量名，配置不同的参数。此处填写其对应的关键字。

```
mox.set_flag('checkpoint_exclude_patterns', 'logits')
```

如果使用的是MoXing内置网络，其对应的关键字需使用如下API获取。此示例将打印Resnet_v1_50的关键字，为“logits”。

```
import moxing.tensorflow as mox

model_meta = mox.get_model_meta(mox.NetworkKeys.RESNET_V1_50)
logits_pattern = model_meta.default_logits_pattern
print(logits_pattern)
```

您也可以通过如下接口，获取MoXing支持的网络名称列表。

```
import moxing.tensorflow as mox
print(help(mox.NetworkKeys))
```

打印出来的示例如下所示：

```
Help on class NetworkKeys in module
moxing.tensorflow.nets.nets_factory:

class NetworkKeys(builtins.object)
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  ALEXNET_V2 = 'alexnet_v2'
 |
 |  CIFARNET = 'cifarnet'
 |
 |  INCEPTION_RESNET_V2 = 'inception_resnet_v2'
 |
 |  INCEPTION_V1 = 'inception_v1'
 |
 |  INCEPTION_V2 = 'inception_v2'
 |
 |  INCEPTION_V3 = 'inception_v3'
 |
 |  INCEPTION_V4 = 'inception_v4'
 |
 |  LENET = 'lenet'
 |
 |  MOBILENET_V1 = 'mobilenet_v1'
 |
 |  MOBILENET_V1_025 = 'mobilenet_v1_025'
 |
 |  MOBILENET_V1_050 = 'mobilenet_v1_050'
```

```
MOBILENET_V1_075 = 'mobilenet_v1_075'
MOBILENET_V2 = 'mobilenet_v2'
MOBILENET_V2_035 = 'mobilenet_v2_035'
MOBILENET_V2_140 = 'mobilenet_v2_140'
NASNET_CIFAR = 'nasnet_cifar'
NASNET_LARGE = 'nasnet_large'
NASNET_MOBILE = 'nasnet_mobile'
OVERFEAT = 'overfeat'
PNASNET_LARGE = 'pnasnet_large'
PNASNET_MOBILE = 'pnasnet_mobile'
PVANET = 'pvanet'
RESNET_V1_101 = 'resnet_v1_101'
RESNET_V1_110 = 'resnet_v1_110'
RESNET_V1_152 = 'resnet_v1_152'
RESNET_V1_18 = 'resnet_v1_18'
RESNET_V1_20 = 'resnet_v1_20'
RESNET_V1_200 = 'resnet_v1_200'
RESNET_V1_50 = 'resnet_v1_50'
RESNET_V1_50_8K = 'resnet_v1_50_8k'
RESNET_V1_50_MOX = 'resnet_v1_50_mox'
RESNET_V1_50_OCT = 'resnet_v1_50_oct'
RESNET_V2_101 = 'resnet_v2_101'
RESNET_V2_152 = 'resnet_v2_152'
RESNET_V2_200 = 'resnet_v2_200'
RESNET_V2_50 = 'resnet_v2_50'
RESNEXT_B_101 = 'resnext_b_101'
RESNEXT_B_50 = 'resnext_b_50'
RESNEXT_C_101 = 'resnext_c_101'
RESNEXT_C_50 = 'resnext_c_50'
VGG_16 = 'vgg_16'
VGG_16_BN = 'vgg_16_bn'
VGG_19 = 'vgg_19'
VGG_19_BN = 'vgg_19_bn'
VGG_A = 'vgg_a'
```

```
VGG_A_BN = 'vgg_a_bn'

XCEPTION_41 = 'xception_41'

XCEPTION_65 = 'xception_65'

XCEPTION_71 = 'xception_71'
```

14.3.8.4 在 Notebook 中如何查看 GPU 使用情况

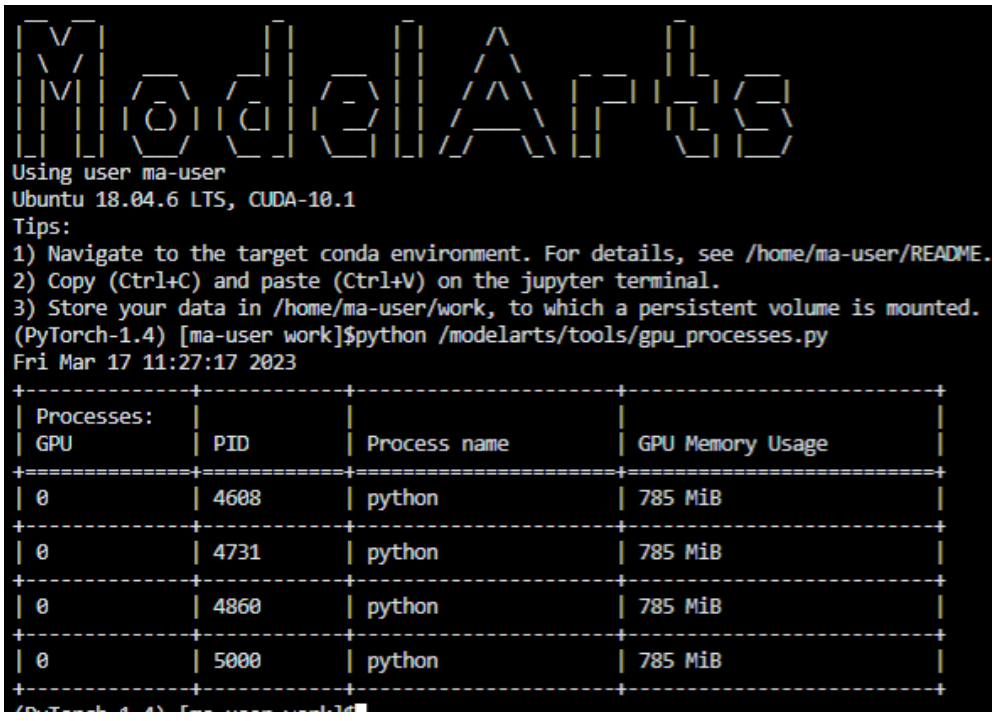
创建Notebook时，当您选择的类型为GPU时，查看GPU使用情况具体操作如下：

1. 登录ModelArts管理控制台，选择“开发环境>Notebook”。
2. 在Notebook列表中，单击目标Notebook“操作”列的“打开”，进入“Jupyter”开发页面。
3. 在Jupyter页面的“Files”页签下，单击“New”，然后选择“Terminal”，进入到Terminal界面。
4. 执行如下命令查看GPU使用情况。
   ```
   nvidia-smi
   ```
5. 查看当前Notebook实例中有哪些进程使用GPU。

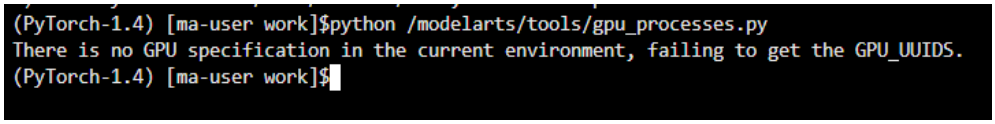
   方法一：

   ```
   python /modelarts/tools/gpu_processes.py
   ```

   如果当前进程使用GPU

   如果当前没有进程使用GPU

   方法二：

   打开文件“/resource_info/gpu_usage.json”，可以看到有哪些进程在使用GPU。

```
{
    <notebook name>: {
        <GPU0 UUID>: [
            {
                "pid": 2263,
                "processName": "python",
                "gpuMemoryUsage": "4935Mi"
            },
            {...}
        ]
        <GPU1 UUID>: [...]
    }
}
```

如果当前没有进程使用GPU，该文件可能不存在或为空。

14.3.8.5 如何在代码中打印 GPU 使用信息

用户可通过shell命令或python命令查询GPU使用信息。

使用 shell 命令

1. 执行nvidia-smi命令。

   依赖CUDA nvcc

   ```
   watch -n 1 nvidia-smi
   ```

   ```
   Every 1.0s: nvidia-smi

   Mon Oct 25 15:20:11 2021
   +-----------------------------------------------------------------------------+
   | NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
   |-------------------------------+----------------------+----------------------+
   | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
   | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
   |===============================+======================+======================|
   |   0  Tesla V100-SXM2...  On   | 00000000:5F:00.0 Off |                    0 |
   | N/A   31C    P0    43W / 300W |      0MiB / 32510MiB |      0%      Default |
   +-------------------------------+----------------------+----------------------+
   |   1  Tesla V100-SXM2...  On   | 00000000:B5:00.0 Off |                    0 |
   | N/A   34C    P0    44W / 300W |      0MiB / 32510MiB |      0%      Default |
   +-------------------------------+----------------------+----------------------+

   +-----------------------------------------------------------------------------+
   | Processes:                                                       GPU Memory |
   |  GPU       PID   Type   Process name                             Usage      |
   |=============================================================================|
   |  No running processes found                                                 |
   +-----------------------------------------------------------------------------+
   ```

2. 执行gpustat命令。

   ```
   pip install gpustat
   gpustat -cp -i
   ```

   ```
   notebook-6a654129-698e-4635-b6be-67aedbdd4c54  Mon Oct 25 15:19:11 2021  440.33.01
   [0] Tesla V100-SXM2-32GB | 31'C,   0 % |     0 / 32510 MB |
   [1] Tesla V100-SXM2-32GB | 34'C,   0 % |     0 / 32510 MB |
   ```

   使用Ctrl+C可以退出。

使用 python 命令

1. 执行nvidia-ml-py3命令（常用）。

```
!pip install nvidia-ml-py3
import nvidia_smi
nvidia_smi.nvmlInit()
deviceCount = nvidia_smi.nvmlDeviceGetCount()
for i in range(deviceCount):
    handle = nvidia_smi.nvmlDeviceGetHandleByIndex(i)
    util = nvidia_smi.nvmlDeviceGetUtilizationRates(handle)
    mem = nvidia_smi.nvmlDeviceGetMemoryInfo(handle)
    print(f"|Device {i}| Mem Free: {mem.free/1024**2:5.2f}MB / {mem.total/1024**2:5.2f}MB | gpu-util:
{util.gpu:3.1%} | gpu-mem: {util.memory:3.1%} |")
```

```
Output:
|Device 0| Mem Free: 32510.44MB / 32510.50MB | gpu-util: 0.0% | gpu-mem: 0.0% |
|Device 1| Mem Free: 32510.44MB / 32510.50MB | gpu-util: 0.0% | gpu-mem: 0.0% |
```

2. 执行nvidia_smi + wapper + prettytable命令。

   用户可以将GPU信息显示操作看作一个装饰器，在模型训练过程中就可以实时的显示GPU状态信息。

```
def gputil_decorator(func):
    def wrapper(*args, **kwargs):
        import nvidia_smi
        import prettytable as pt

        try:
            table = pt.PrettyTable(['Devices','Mem Free','GPU-util','GPU-mem'])
            nvidia_smi.nvmlInit()
            deviceCount = nvidia_smi.nvmlDeviceGetCount()
            for i in range(deviceCount):
                handle = nvidia_smi.nvmlDeviceGetHandleByIndex(i)
                res = nvidia_smi.nvmlDeviceGetUtilizationRates(handle)
                mem = nvidia_smi.nvmlDeviceGetMemoryInfo(handle)
                table.add_row([i, f"{mem.free/1024**2:5.2f}MB/{mem.total/1024**2:5.2f}MB",
f"{res.gpu:3.1%}", f"{res.memory:3.1%}"])

        except nvidia_smi.NVMLError as error:
            print(error)

        print(table)
        return func(*args, **kwargs)
    return wrapper
```

```
Output:

+---------+-----------------------+----------+----------+
| Devices |        Mem Free       | GPU-util | GPU-mem |
+---------+-----------------------+----------+----------+
|    0    | 32510.44MB/32510.50MB |   0.0%   |   0.0%   |
|    1    | 32510.44MB/32510.50MB |   0.0%   |   0.0%   |
+---------+-----------------------+----------+----------+
```

3. 执行pynvml命令。

   nvidia-ml-py3可以直接查询nvml c-lib库，而无需通过nvidia-smi。因此，这个模块比nvidia-smi周围的包装器快得多。

```
from pynvml import *
nvmlInit()
handle = nvmlDeviceGetHandleByIndex(0)
info = nvmlDeviceGetMemoryInfo(handle)
print("Total memory:", info.total)
print("Free memory:", info.free)
print("Used memory:", info.used)
```

```
Output:
Total memory: 34089730048
Free memory: 34089664512
Used memory: 65536
```

4. 执行gputil命令。

```
!pip install gputil
import GPUtil as GPU
GPU.showUtilization()
```

```
Output:

| ID | GPU | MEM |
------------------
|  0 |  0% | 25% |
|  1 |  0% |  0% |
```

```
import GPUtil as GPU
GPUs = GPU.getGPUs()
for gpu in GPUs:
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total
{3:.0f}MB".format(gpu.memoryFree, gpu.memoryUsed, gpu.memoryUtil*100, gpu.memoryTotal))
```

```
Output:
GPU RAM Free: 32510MB | Used: 0MB | Util   0% | Total 32510MB
GPU RAM Free: 32510MB | Used: 0MB | Util   0% | Total 32510MB
```

注：用户在使用pytorch/tensorflow等深度学习框架时也可以使用框架自带的api进行查询。

14.3.8.6 Ascend 上如何查看实时性能指标?

Ascend芯片上查看实时性能指标：npu-smi info，类似GPU的nvidia-smi。

14.3.8.7 JupyterLab 目录的文件、Terminal 的文件和 OBS 的文件之间的关系

- JupyterLab目录的文件与Terminal中work目录下的文件相同。即用户在Notebook中新建的，或者是从OBS目录中同步的文件。
- 挂载OBS存储的Notebook，JupyterLab目录的文件可以与OBS的文件进行同步，使用JupyterLab文件上传下载功能。Terminal的文件与JupyterLab目录的文件相同。
- 挂载EVS存储的Notebook，JupyterLab目录的文件可使用Moxing接口或SDK接口，读取OBS中的文件。Terminal的文件与JupyterLab目录的文件相同。

14.3.8.8 如何迁移旧版 Notebook 数据到新版 Notebook

旧版Notebook功能已经下线，本章节介绍如何将旧版Notebook存储中的数据迁移至新版Notebook继续使用。

新旧版 Notebook 使用存储差异说明

表 14-6 新版旧版 Notebook 支持的存储配置说明

存储类型	旧版 Notebook	新版 Notebook	说明
OBS对象存储服务	支持	不支持	OBS为存储系统，而非文件系统。 在旧版Notebook使用过程中，OBS数据的远程复制以及本地复制容易混淆，造成用户对数据操作难以控制。因此，在新版Notebook中去掉了挂载OBS能力，用户可以通过代码操作OBS数据，灵活的获取和操作数据。
OBS并行文件系统	不支持	支持	新版Notebook提供了动态挂载OBS并行文件系统能力，用户在启动的Notebook详情页面中，可以添加数据存储。不涉及新旧版Notebook间的数据迁移。
云硬盘EVS	支持	支持	新旧版Notebook均支持使用，需要迁移保存旧版Notebook中的数据。
弹性文件服务SFS	不支持	支持	在专属资源池场景下使用。旧版Notebook中该功能已经下线，不涉及迁移。
文件存储EFS	不支持	支持	仅新版Notebook支持。

旧版 Notebook 使用 OBS 存储

旧版Notebook使用OBS存储时，数据是存放在OBS上，不需要迁移数据。直接在新版Notebook中使用该目录下数据即可。具体操作方式参见： **如何在Notebook中读写OBS文件**。

图 14-27 旧版 Notebook 使用 OBS 存储

旧版 Notebook 使用 EVS 存储

旧版Notebook使用的是云硬盘EVS存储，建议将EVS中的数据保存并迁移，在新版Notebook中使用。

- EVS存储中数据量较少：建议将数据下载到本地，创建新版Notebook后再上传。
- EVS存储中数据量较大：建议将EVS中的数据上传至OBS桶中保存。创建新版Notebook时，通过OBS桶读取数据使用。

Notebook中的数据上传下载的具体操作请参见: Notebook中的数据上传下载。

图 14-28 旧版 Notebook 使用 EVS 存储

14.3.8.9 ModelArts 中创建的数据集，如何在 Notebook 中使用

ModelArts上创建的数据集存放在OBS中，可以将OBS中的数据下载到Notebook中使用。

Notebook中读取OBS数据方式请参见**如何在Notebook中上传下载OBS文件?** 。

14.3.8.10 pip 介绍及常用命令

pip是通用的python包的管理工具。它提供了对Python包的查找、下载、安装和卸载的功能。

pip常用命令如下：

```
pip --help#获取帮助
pip install SomePackage==XXXX #指定版本安装
pip install SomePackage #最新版本安装
pip uninstall SomePackage #卸载软件版本
```

其他命令请使用**pip --help**命令查询。

14.3.8.11 开发环境中不同 Notebook 规格资源“/cache”目录的大小

创建Notebook时，可以根据业务数据量的大小选择资源。

ModelArts会挂载硬盘至“/cache”目录，用户可以使用此目录来储存临时文件。“/cache”与代码目录共用资源，不同资源规格有不同的容量。

映射规则：当前不支持CPU配置cache盘；GPU与昇腾资源为单卡时，cache目录保持500G大小限制；除单卡外，cache盘大小与卡数有关，计算方式为卡数*500G，上限为3T。详细表14-7所示。

表 14-7 不同 Notebook 规格资源“/cache”目录的大小

规格类别	cache盘大小
GPU-0.25卡	500G*0.25
GPU-0.5卡	500G*0.5
GPU-单卡	500G
GPU-双卡	500G*2
GPU-四卡	500G*4
GPU-八卡	3T
昇腾-单卡	500G
昇腾-双卡	500G*2
昇腾-四卡	500G*4
昇腾-八卡	3T
CPU	--

14.3.8.12 资源超分对 Notebook 实例有什么影响?

Notebook超分，是指一个节点中CPU、内存共享的场景。为了充分利用资源，在专属池中存在超分情况。

举例：一个专属池中有1个8U64G的CPU节点，如创建2U8G规格的Notebook，因为超分最多可启动 8U/(2U*0.6)= 6.67个Notebook实例。这里的0.6就是超分比率。即启动该Notebook实例最少需要1.2U的CPU，运行Notebook时最大使用到2U的资源；内存同理，最少需要4.8G的内存，运行时最大使用到8U的内存。

超分情况下会存在实例终止的风险。如1个8U的节点上同时启动了6个2U的实例，如果其中一个实例CPU使用增大到超过节点的上限（ 8U ）时，k8S会将使用资源最多的实例终止掉。

因此超分会带来实例重启的风险，请不要超分使用。

14.4 训练作业

14.4.1 功能咨询

14.4.1.1 欠拟合的解决方法有哪些?

1. 模型复杂化。
   - 对同一个算法复杂化。例如回归模型添加更多的高次项，增加决策树的深度，增加神经网络的隐藏层数和隐藏单元数等。
   - 弃用原来的算法，使用一个更加复杂的算法或模型。例如用神经网络来替代线性回归，用随机森林来代替决策树。
2. 增加更多的特征，使输入数据具有更强的表达能力。
   - 特征挖掘十分重要，尤其是具有强表达能力的特征，可以抵过大量的弱表达能力的特征。
   - 特征的数量并非重点，质量才是，总之强表达能力的特征最重要。
   - 能否挖掘出强表达能力的特征，还在于对数据本身以及具体应用场景的深刻理解，这依赖于经验。
3. 调整参数和超参数。
   - 神经网络中：学习率、学习衰减率、隐藏层数、隐藏层的单元数、Adam优化算法中的β1和β2参数、batch_size数值等。
   - 其他算法中：随机森林的树数量，k-means中的cluster数，正则化参数λ等。
4. 增加训练数据作用不大。

   欠拟合一般是因为模型的学习能力不足，一味地增加数据，训练效果并不明显。
5. 降低正则化约束。

   正则化约束是为了防止模型过拟合，如果模型压根不存在过拟合而是欠拟合了，那么就考虑是否降低正则化参数λ或者直接去除正则化项。

14.4.1.2 旧版训练迁移至新版训练需要注意哪些问题?

新版训练和旧版训练的差异主要体现在以下3点:

- **新旧版创建训练作业方式差异**
- **新旧版训练代码适配的差异**
- **新旧版训练预置引擎差异**

新旧版创建训练作业方式差异

- 旧版训练支持使用“算法管理”、“常用框架”、“自定义”（即自定义镜像）方式创建训练作业。
- 新版训练支持使用“自定义算法”、“我的算法”方式来创建训练作业。

新版训练的创建方式有了更明确的类别划分，选择方式和旧版训练存在区别。

- 旧版中使用“算法管理”中已保存的算法创建训练作业的用户，可以在新版训练中使用“我的算法”创建训练作业。
- 旧版中使用“常用框架”创建训练作业的用户，可以在新版训练中使用“自定义算法”创建训练作业（启动方式选择“预置框架”）。
- 旧版中使用“自定义”（即自定义镜像）创建训练作业的用户，可以在新版训练中使用“自定义算法”创建训练作业（启动方式选择“自定义”）。

新旧版训练代码适配的差异

旧版训练中，用户需要在输入输出数据上做如下配置：

```
#解析命令行参数
import argparse
parser = argparse.ArgumentParser(description='MindSpore Lenet Example')
parser.add_argument('--data_url', type=str, default="./Data",
                    help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='if is test, must provide\
                    path where the trained ckpt file')
args = parser.parse_args()
...
#下载数据参数至容器本地，在代码中使用local_data_path代表训练输入位置
mox.file.copy_parallel(args.data_url, local_data_path)
...
#上传容器本地数据至obs路径
mox.file.copy_parallel(local_output_path, args.train_url)
```

新版训练中，用户配置输入输出数据，无需书写下载数据的代码，在代码中把arg.data_url和arg.train_url当做本地路径即可，详情参考开发自定义脚本。

```
#解析命令行参数
import argparse
parser = argparse.ArgumentParser(description='MindSpore Lenet Example')
parser.add_argument('--data_url', type=str, default="./Data",
                    help='path where the dataset is saved')
parser.add_argument('--train_url', type=str, default="./Model", help='if is test, must provide\
                    path where the trained ckpt file')
args = parser.parse_args()
...
# 下载的代码无需设置，后续涉及训练数据和输出路径数据使用data_url和train_url即可
#下载数据参数至容器本地，在代码中使用local_data_path代表训练输入位置
#mox.file.copy_parallel(args.data_url, local_data_path)
...
#上传容器本地数据至obs路径
#mox.file.copy_parallel(local_output_path, args.train_url)
```

新旧版训练预置引擎差异

- 新版的预置训练引擎默认安装Moxing2.0.0及以上版本。
- 新版的预置训练引擎统一使用了Python3.7及以上版本。
- 新版镜像修改了默认的HOME目录，由“/home/work”变为“/home/ma-user”，请注意识别训练代码中是否有“/home/work”的硬编码。
- 提供预置引擎类型有差异。新版的预置引擎在常用的训练引擎上进行了升级。

  如果您需要使用旧版训练引擎，单击显示旧版引擎即可选择旧版引擎。新旧版支持的预置引擎差异请参考**表14-8**。

表 14-8 新旧版预置引擎差异

工作环境	预置训练I引擎与版本	旧版训练	新版训练
Ascend-Powered-Engine	Mindspore-1.3.0	√	x
	Mindspore-1.7.0	x	√
	Tensorflow-1.15	√	√

14.4.1.3 ModelArts 训练好后的模型如何获取?

使用自动学习产生的模型只能在ModelArts上部署上线，无法下载至本地使用。

使用自定义算法或者订阅算法训练生成的模型，会存储至用户指定的OBS路径中，供用户下载。

14.4.1.4 模型可视化作业中各参数的意义?

可视化作业通过TensorBoard提供能力，TensorBoard功能介绍请参见**TensorBoard官网资料**。

14.4.1.5 如何在 ModelArts 上获得 RANK_TABLE_FILE 进行分布式训练?

ModelArts会帮用户生成RANK_TABLE_FILE文件，可通过环境变量查看文件位置。

- 在Notebook中打开terminal，可以运行如下命令查看RANK_TABLE_FILE：
  ```
  env | grep RANK
  ```
- 在训练作业中，您可以在训练启动脚本的首行加入如下代码，把RANK_TABLE_FILE的值打印出来：
  ```
  os.system('env | grep RANK')
  ```

14.4.1.6 如何查询自定义镜像的 cuda 和 cudnn 版本?

查询cuda版本：

```
cat /usr/local/cuda/version.txt
```

查询cudnn版本：

```
cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2
```

14.4.1.7 Moxing 安装文件如何获取?

Moxing安装文件不支持下载和用户自主安装。在ModelArts的Notebook和训练作业镜像中预置了Moxing安装包，用户可以直接引用。

14.4.1.8 多节点训练 TensorFlow 框架 ps 节点作为 server 会一直挂着，ModelArts 是怎么判定训练任务结束? 如何知道是哪个节点是 worker 呢?

TensorFlow框架分布式训练的情况下，会启动ps与worker任务组，worker任务组为关键任务组，会以worker任务组的进程退出码，判断训练作业是否结束。

通过task name判断的哪个节点是worker。下发的训练作业是一个volcano job，里边会有两个task：一个是ps、一个是worker。两个task的启动命令不同，会自动生成超参--task_name，ps的--task_name=ps，worker的 --task_name=worker。

14.4.1.9 训练作业的自定义镜像如何安装 Moxing?

为避免自动安装Moxing会影响用户自定义镜像中的包环境，所以自定义镜像需要用户手动安装Moxing。Moxing安装包会在作业启动后放在“/home/ma-user/modelarts/package/”目录下。可在使用Moxing功能前执行如下代码，进行Moxing的安装。

```
import os
os.system("pip install /home/ma-user/modelarts/package/moxing_framework-*.whl")
```

说明

本案例仅适用于训练作业环境。

14.4.2 训练过程读取数据

14.4.2.1 在 ModelArts 上训练模型，输入输出数据如何配置?

ModelArts支持用户上传自定义算法创建训练作业。上传自定义算法前，请完成算法的开发并上传至OBS桶。创建算法请参考使用预置框架创建算法。创建训练作业请参考创建训练作业指导。

解析输入路径参数、输出路径参数

运行在ModelArts的模型读取存储在OBS服务的数据，或者输出至OBS服务指定路径，输入和输出数据需要配置3个地方：

1. 训练代码中需解析输入路径参数和输出路径参数。ModelArts推荐以下方式实现参数解析。

```
import argparse
# 创建解析
parser = argparse.ArgumentParser(description="train mnist",
                                 formatter_class=argparse.ArgumentDefaultsHelpFormatter)
# 添加参数
parser.add_argument('--train_url', type=str,
                    help='the path model saved')
parser.add_argument('--data_url', type=str, help='the training data')
# 解析参数
args, unknown = parser.parse_known_args()
```

   完成参数解析后，用户使用“data_url”、“train_url”代替算法中数据来源和数据输出所需的路径。

2. 在使用预置框架创建算法时，根据1中的代码参数设置定义的输入输出参数。
   - 训练数据是算法开发中必不可少的输入。“输入”参数建议设置为“data_url”，表示数据输入来源，也支持用户根据1的算法代码自定义代码参数。
   - 模型训练结束后，训练模型以及相关输出信息需保存在OBS路径。“输出”数据默认配置为模型输出，代码参数为“train_url”，也支持用户根据1的算法代码自定义输出路径参数。
3. 在创建训练作业时，填写输入路径和输出路径。

   训练输入选择对应的OBS路径或者数据集路径，训练输出选择对应的OBS路径。

14.4.2.2 如何提升训练效率，同时减少与 OBS 的交互?

场景描述

在使用ModelArts进行自定义深度学习训练时，训练数据通常存储在对象存储服务（OBS）中，且训练数据较大时（如200GB以上），每次都需要使用GPU资源池进行训练，且训练效率低。

希望提升训练效率，同时减少与对象存储OBS的交互。可通过如下方式进行调整优化。

优化原理

对于ModelArts提供的GPU资源池，每个训练节点会挂载500GB的NVMe类型SSD提供给用户免费使用。此SSD挂载到“/cache”目录，“/cache”目录下的数据生命周期与训练作业生命周期相同，当训练作业运行结束以后“/cache”目录下面所有内容会被

清空，腾出空间，供下一次训练作业使用。因此，可以在训练过程中将数据从OBS复制到“/cache”目录，然后每次从“/cache”目录读取数据，直到训练结束。训练结束以后“/cache”目录的内容会自动被清空。

优化方式

以TensorFlow代码为例。

优化前代码如下所示：

```
...
tf.flags.DEFINE_string('data_url', '', 'dataset directory.')
FLAGS = tf.flags.FLAGS
mnist = input_data.read_data_sets(FLAGS.data_url, one_hot=True)
```

优化后的代码示例如下，将数据复制至“/cache”目录。

```
...
tf.flags.DEFINE_string('data_url', '', 'dataset directory.')
FLAGS = tf.flags.FLAGS
import moxing as mox
TMP_CACHE_PATH = '/cache/data'
mox.file.copy_parallel('FLAGS.data_url', TMP_CACHE_PATH)
mnist = input_data.read_data_sets(TMP_CACHE_PATH, one_hot=True)
```

14.4.2.3 大量数据文件，训练过程中读取数据效率低?

当数据集存在较多数据文件（即海量小文件），数据存储在OBS中，训练过程需反复从OBS中读取文件，导致训练过程一直在等待文件读取，效率低。

解决方法

1. 建议将海量小文件，在本地压缩打包。例如打包成.zip格式。
2. 将此压缩后的文件上传至OBS。
3. 训练时，可直接从OBS下载此压缩文件至/cache目录。此操作仅需执行一次，无需训练过程反复与OBS交互导致训练效率低。

   如下示例，可使用mox.file.copy_parallel将zip文件下载至本地/cache目录并解压，然后再读取做训练。

   ```
   ...
   tf.flags.DEFINE_string('<obs_file_path>/data.zip', '', 'dataset directory.')
   FLAGS = tf.flags.FLAGS
   import os
   import moxing as mox
   TMP_CACHE_PATH = '/cache/data'
   mox.file.copy_parallel('FLAGS.data_url', TMP_CACHE_PATH)
   zip_data_path = os.path.join(TMP_CACHE_PATH, '*.zip')
   unzip_data_path = os.path.join(TEMP_CACHE_PATH, 'unzip')
   #也可以采用zipfile等Python包来做解压
   os.system('unzip '+ zip_data_path + ' -d ' + unzip_data_path)
   mnist = input_data.read_data_sets(unzip_data_path, one_hot=True)
   ```

14.4.2.4 使用 Moxing 时如何定义路径变量?

问题描述

```
mox.file.copy_parallel(src_obs_dir=input_storage,'obs://dyyolov8/yolov5_test/yolov5-7.0/datasets'),
```

mox这个函数怎么定义以变量的形式填写OBS路径?

解决方案

变量定义参考如下示例：

```
input_storage = './test.py'
import moxing as mox
mox.file.copy_parallel(input_storage,'obs://dyyolov8/yolov5_test/yolov5-7.0/datasets')
```

14.4.3 编写训练代码

14.4.3.1 训练模型时引用依赖包，如何创建训练作业?

ModelArts支持训练模型过程中安装第三方依赖包。在训练代码目录下放置“pip-requirements.txt”文件后，在训练启动文件被执行前系统会执行如下命令，以安装用户指定的Python Packages。

```
pip install -r pip-requirements.txt
```

仅使用**预置框架**创建的训练作业支持在训练模型时引用依赖包。

说明

pip-requirements.txt文件命名支持以下4种格式，文档中以pip-requirements为例说明。

- pip-requirement.txt
- pip-requirements.txt
- requirement.txt
- requirements.txt

- 代码目录位置请参考**在代码目录下提供安装文件**。
- pip-requirements文件写法请参考**安装文件规范**。

在代码目录下提供安装文件

- 如果使用“我的算法”创建训练作业，则在创建算法时，可以把相关文件放置在配置的“代码目录”下，算法的“启动方式”必须选择“预置框架”。
- 如果使用“自定义算法”创建训练作业，则可以把相关文件放置在配置的“代码目录”下，“启动方式”必须选择“预置框架”。

需要在创建训练作业前将相关文件上传至OBS路径下，文件打包要求请参见**安装文件规范**。

安装文件规范

请根据依赖包的类型，在代码目录下放置对应文件：

- **依赖包为开源安装包时**

  说明

  暂时不支持直接从github的源码中安装。

  在“代码目录”中创建一个命名为“pip-requirements.txt”的文件，并且在文件中写明依赖包的包名及其版本号，格式为“包名==版本号”。

  例如，“代码目录”对应的OBS路径下，包含模型文件，同时还存在“pip-requirements.txt”文件。“代码目录”的结构如下所示：

```
|---模型启动文件所在OBS文件夹
    |---model.py              #模型启动文件。
    |---pip-requirements.txt   #定义的配置文件，用于指定依赖包的包名及版本号。
```

“pip-requirements.txt”文件内容如下所示：

```
alembic==0.8.6
bleach==1.4.3
click==6.6
```

- **依赖包为whl包时**

  如果训练后台不支持下载开源安装包或者使用用户编译的whl包时，由于系统无法自动下载并安装，因此需要在“代码目录”放置此whl包，同时创建一个命名为“pip-requirements.txt”的文件，并且在文件中指定此whl包的包名。依赖包必须为“.whl”格式的文件。

  例如，“代码目录”对应的OBS路径下，包含模型文件、whl包，同时还存在“pip-requirements.txt”文件。“代码目录”的结构如下所示：

  ```
  |---模型启动文件所在OBS文件夹
      |---model.py              #模型启动文件。
      |---XXX.whl               #依赖包。依赖多个时，此处放置多个。
      |---pip-requirements.txt   #定义的配置文件，用于指定依赖包的包名。
  ```

  “pip-requirements.txt”文件内容如下所示：

  ```
  numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl
  tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
  ```

14.4.3.2 训练作业常用文件路径是什么？

训练环境的当前目录以及代码目录在容器的位置一般通过环境变量${MA_JOB_DIR}读取，${MA_JOB_DIR}变量对应的实际值是/home/ma-user/modelarts/user-job-dir。

14.4.3.3 如何安装 C++的依赖库？

在训练作业的过程中，会使用到第三方库。以C++为例，请参考如下操作步骤进行安装：

1. 将源码下载至本地并上传到OBS。
2. 将上传到OBS的源码使用Moxing复制到开发环境Notebook中。

   以下为使用EVS挂载的开发环境，将数据复制至notebook中的代码示例：

   ```
   import moxing as mox
   mox.file.make_dirs('/home/ma-user/work/data')
   mox.file.copy_parallel('obs://bucket-name/data', '/home/ma-user/work/data')
   ```

3. 在Jupyter页面的“Files”页签下，单击“New”，打开“Terminal”。执行如下命令进入目标路径，确认源码已下载，即“data”文件是否存在。

   ```
   cd /home/ma-user/work
   ls
   ```

4. 在“Terminal”环境进行编译，具体编译方式请您根据业务需求进行。
5. 将编译结果使用Moxing复制至OBS中 。代码示例如下：

   ```
   import moxing as mox
   mox.file.make_dirs('/home/ma-user/work/data')
   mox.file.copy_parallel('/home/ma-user/work/data', 'obs://bucket-name/file)
   ```

6. 在训练时，将OBS中的编译结果使用Moxing复制到容器中使用。代码示例如下：

   ```
   import moxing as mox
   mox.file.make_dirs('/cache/data')
   mox.file.copy_parallel('obs://bucket-name/data', '/cache/data')
   ```

14.4.3.4 训练作业中如何判断文件夹是否拷贝完毕?

您可以在训练作业启动文件的脚本中，通过如下方式获取拷贝和被拷贝文件夹大小，根据结果判断是否拷贝完毕：

```
import moxing as mox
mox.file.get_size('obs://bucket_name/obs_file',recursive=True)
```

其中，“get_size”为获取文件或文件夹的大小。“recursive=True”表示类型为文件夹，“True”表示是文件夹，“False”为文件。

如果输出结果为一致，表示文件夹拷贝已完毕。如果输出结果不一致，表示拷贝未结束。

14.4.3.5 如何在训练中加载部分训练好的参数?

在训练作业时，需要从预训练的模型中加载部分参数，初始化当前模型。请您通过如下方式加载：

1. 通过如下代码，您可以查看所有的参数。
   ```
   from moxing.tensorflow.utils.hyper_param_flags import mox_flags
   print(mox_flags.get_help())
   ```
2. 通过如下方式控制载入模型时需要恢复的参数名。其中，“checkpoint_include_patterns”为需要恢复的参数，“checkpoint_exclude_patterns”为不需要恢复的参数。
   ```
   checkpoint_include_patterns: Variables names patterns to include when restoring checkpoint. Such as: conv2d/weights.
   checkpoint_exclude_patterns: Variables names patterns to include when restoring checkpoint. Such as: conv2d/weights.
   ```
3. 通过以下方式控制需要训练的参数列表。其中，“trainable_include_patterns”为需要训练的参数列表，“trainable_exclude_patterns”为不需要训练的参数列表。
   ```
   --trainable_exclude_patterns: Variables names patterns to exclude for trainable variables. Such as: conv1,conv2.
   --trainable_include_patterns: Variables names patterns to include for trainable variables. Such as: logits.
   ```

14.4.3.6 训练作业的启动文件如何获取训练作业中的参数?

训练作业参数有两种来源，包括后台自动生成的参数和用户手动输入的参数。具体获取方式如下：

1. 创建训练作业时，“输入”支持配置训练的输入参数名称（一般设置为“data_url”），以及输入数据的存储位置，“输出”支持配置训练的输出参数名称（一般设置为“train_url”），以及输出数据的存储位置。
2. 训练作业运行成功之后，在训练作业列表中，您可以单击作业名称，查看该作业的详情。可从日志中获取参数的传入方式，如图14-29所示。

图 14-29 查看日志

3. 如果需在训练中获取“train_url”、“data_url”和“test”参数的值，可在训练作业的启动文件中添加以下代码获取：

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('--data_url', type=str, default=None, help='test')
parser.add_argument('--train_url', type=str, default=None, help='test')
parser.add_argument('--test', type=str, default=None, help='test')
```

14.4.3.7 训练作业中使用 os.system('cd xxx')无法进入相应的文件夹?

当在训练作业的启动脚本中使用**os.system('cd xxx')**无法进入相应的文件夹时，建议使用如下方法：

```
import os
os.chdir('/home/work/user-job-dir/xxx')
```

14.4.3.8 训练作业如何调用 shell 脚本，是否可以执行.sh 文件?

ModelArts支持调用shell脚本，可以使用python调用“.sh”。具体操作步骤如下：

1. 上传“.sh”脚本至OBS桶，例如“.sh”所在存储位置为 “ /bucket-name/code/test.sh”。
2. 在本地创建“.py”文件，例如“test.py”。由于后台会自动将代码目录下载至容器的“/home/work/user-job-dir/”目录下，因此您可以在启动文件“test.py”中通过如下方式调用“.sh”文件：

```
import os
os.system('bash /home/work/user-job-dir/code/test.sh')
```

3. 将“test.py”文件上传至OBS中，则该文件存储位置为“/bucket-name/code/test.py”。
4. 创建训练作业时，指定的代码目录为“/bucket-name/code/”，启动文件目录为“/bucket-name/code/test.py”。

训练作业创建完成之后就可以使用python调用“.sh”文件。

14.4.3.9 训练代码中，如何获取依赖文件所在的路径?

由于用户本地开发的代码需要上传至ModelArts后台，训练代码中涉及到依赖文件的路径时，用户设置有误的场景较多。因此推荐通用的解决方案：使用os接口得到依赖文件的绝对路径，避免报错。

以下示例展示如何通过os接口获得其他文件夹下的依赖文件路径。

文件目录结构：

```
project_root                #代码根目录
 └─bootfile.py              #启动文件
 └─otherfileDirectory       #其他依赖文件所在的目录
     └─otherfile.py         #其他依赖文件
```

在启动文件代码中，建议用户参考以下方式获取其他依赖文件所在路径，即示例中的“otherfile_path”。

```
import os
current_path = os.path.dirname(os.path.realpath(__file__)) # 获得启动文件bootfile.py的路径
project_root = os.path.dirname(current_path) # 通过启动文件路径获得工程的根目录，对应ModelArts训练控制台上设置的代码目录
otherfile_path = os.path.join(project_root, "otherfileDirectory", "otherfile.py") # 通过工程的根目录得到依赖文件路径
```

14.4.3.10 自定义 python 包中如果引用 model 目录下的文件，文件路径怎么写

如果容器中的文件实际路径不清楚，可以使用Python获取当前文件路径的方法获取。

```
os.getcwd() #获取文件当前工作目录路径（绝对路径）
os.path.realpath(__ file __) #获得文件所在的路径（绝对路径）
```

也可在搜索引擎寻找其他获取文件路径的方式，使用获取到的路径进行文件读写。

14.4.4 创建训练作业

14.4.4.1 创建训练作业时提示“对象目录大小/数量超过限制”，如何解决?

问题分析

创建训练作业选择的代码目录有大小和文件个数限制。

解决方法

将代码目录中除代码以外的文件删除或存放到其他目录，保证代码目录大小不超过128MB，文件个数不超过4096个。

14.4.4.2 训练作业参数填写应该注意什么?

训练作业参数填写需要您注意以下几点：

- 如果已配置算法来源和数据来源，则下方的运行参数，将根据选择的对象自动填写“data_url”，无法直接在运行参数中直接修改。

  图 14-30 自动填充的运行参数

- 在创建训练作业配置运行参数时，只需要填写对应的参数与参数值，如图14-31所示。

  图 14-31 配置运行参数

- 训练作业中的参数值为OBS桶路径时，需要使用数据对应的路径，且以“obs://”开头。如图14-32所示。

图 14-32 配置参数 OBS 路径

- 在代码中创建OBS文件夹时，需要调用MoXing的API，具体方法如下：

```
import moxing as mox
mox.file.make_dirs('obs://bucket_name/sub_dir_0/sub_dir_1')
```

14.4.4.3 训练环境中不同规格资源“/cache”目录的大小

在创建训练作业时可以根据训练作业的大小选择资源。

ModelArts会挂载硬盘至“/cache”目录，用户可以使用此目录来储存临时文件。“/cache”与代码目录共用资源，不同资源规格有不同的容量。

说明

- k8s磁盘的驱逐策略是90%，所以可以正常使用的磁盘大小应该是“cache目录容量 x 0.9”。
- 裸机的本地磁盘为物理磁盘，无法扩容，若存储的数据量大，建议使用SFS存放数据，SFS支持扩容。

- GPU规格的资源

表 14-9 GPU cache 目录容量

GPU规格	cache目录容量
V100	800G
8*V100	3T
P100	800G

- CPU规格的资源

表 14-10 CPU cache 目录容量

CPU规格	cache目录容量
2 核 8GiB	50G
8 核 32GiB	50G

14.4.4.4 训练作业的“/cache”目录是否安全?

ModelArts训练作业的程序运行在容器中，容器挂载的目录地址是唯一的，只有运行时的容器能访问到。因此训练作业的“/cache”是安全的。

14.4.4.5 训练作业一直在等待中（排队）？

训练作业状态一直在等待中状态表示当前所选的资源池规格资源紧张，作业需要进行排队，请耐心等待。如想降低排队时间，根据您所选资源池的类型，有以下建议：

1. 公共资源池：

   公共资源池资源较少，高峰期如举办相关活动时会存在资源不足情况。有以下方法可以尝试：

   - 如果使用的是免费规格，可以换成收费规格，免费规格资源较少，排队概率高。
   - 规格选择卡数尽量少，如可以选择1卡，相比于选择8卡排队几率大大降低。
   - 可以尝试使用其他Region。
   - 如果有长期的资源使用诉求，可以购买独占使用的专属资源池。

2. 专属资源池：

   - 如有多个可用的专属资源池，可尝试选择其他较为空闲的资源池。
   - 可清理当前资源池下的其他资源，如停止长时间不使用的Notebook。
   - 在非高峰期时提交训练作业。
   - 如长期长时间排队可以联系该专属资源池的账号管理员，管理员可根据使用情况对资源池进行扩容。

相关问题：**为什么资源充足还是在排队？**

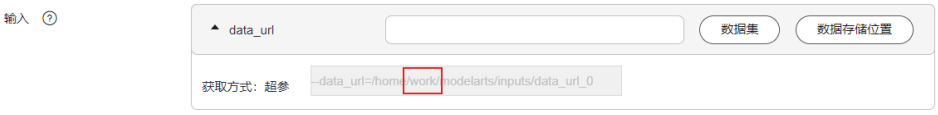
14.4.4.6 创建训练作业时，超参目录为什么有的是/work 有的是/ma-user?

问题描述

创建训练作业时，输入输出参数的超参目录有的是/work，有的是/ma-user。

图 14-33 目录是/ma-user

图 14-34 目录是/work

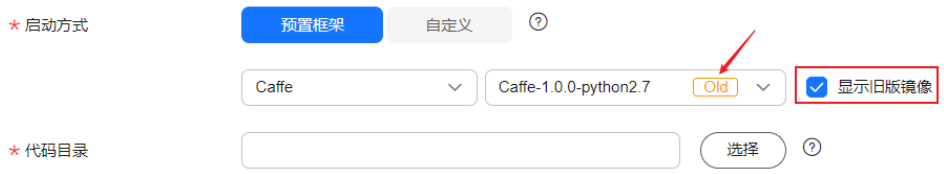


解决方案

这是创建训练作业选用的算法有差异导致的。

- 如果选择的算法是使用旧版镜像创建的，那么创建训练作业时输入输出参数的超参目录就是/work。

图 14-35 创建算法



- 如果选择的算法不是使用旧版镜像创建的，那么创建训练作业时输入输出参数的超参目录就是/ma-user。

14.4.5 管理训练作业版本

14.4.5.1 训练作业是否支持定时或周期调用?

ModelArts训练作业不支持定时周期化调用。当您的作业处于“运行中”状态时，可以按照业务需求进行调用。

14.4.6 查看作业详情

14.4.6.1 如何查看训练作业资源占用情况?

在ModelArts管理控制台，选择“训练管理>训练作业”，进入训练作业列表页面。在训练作业列表中，单击目标作业名称，查看该作业的详情。您可以在“资源占用情况”页签查看到如下指标信息。

- CPU：CPU使用率（cpuUsage）百分比（Percent）。
- MEM：物理内存使用率（memUsage）百分比（Percent）。
- GPU：GPU使用率（gpuUtil）百分比（Percent）。
- GPU_MEM：显存使用率（gpuMemUsage）百分比（Percent）。

14.4.6.2 如何访问训练作业的后台?

ModelArts不支持访问训练作业后台。

14.4.6.3 两个训练作业的模型都保存在容器相同的目录下是否有冲突?

ModelArts训练作业之间的存储目录相互不影响，每个环境之间彼此隔离，看不到其他作业的数据。

14.4.6.4 训练输出的日志只保留 3 位有效数字，是否支持更改 loss 值?

在训练作业中，训练输出的日志只保留3位有效数字，当loss过小的时候，显示为0.000。具体日志如下：

```
INFO:tensorflow:global_step/sec: 0.382191
INFO:tensorflow:step: 81600(global step: 81600) sample/sec: 12.098 loss: 0.000
INFO:tensorflow:global_step/sec: 0.382876
INFO:tensorflow:step: 81700(global step: 81700) sample/sec: 12.298 loss: 0.000
```

由于当前不支持更改loss值，您可以通过将loss的值乘以1000来规避此问题。

14.4.6.5 训练好的模型是否可以下载或迁移到其他账号？如何获取下载路径？

通过训练作业训练好的模型可以下载，然后将下载的模型上传存储至其他账号对应区域的OBS中。

获取模型下载路径

1. 登录ModelArts管理控制台，在左侧导航栏中选择“训练管理 > 训练作业”，进入“训练作业”列表。
2. 在训练作业列表中，单击目标训练作业名称，查看该作业的详情。
3. 在左侧获取“输出位置”下的路径，即为训练模型的下载路径。

模型迁移到其他账号

您可以通过如下两种方式将训练的模型迁移到其他账号。

- 将训练好的模型下载至本地后，上传至目标账号对应区域的OBS桶中。
- 通过对模型存储的目标文件夹或者目标桶配置策略，授权其他账号进行读写操作。详请参见对象存储服务OBS文档中的“自定义创建桶策略（可视化视图）”。

14.5 推理部署

14.5.1 模型管理

14.5.1.1 导入模型

14.5.1.1.1 如何将 Keras 的.h5 格式模型导入到 ModelArts 中

ModelArts不支持直接导入“.h5”格式的模型。您可以先将Keras的“.h5”格式转换为TensorFlow的格式，然后再导入ModelArts中。

从Keras转TensorFlow操作指导请参见其**官网指导**。

14.5.1.1.2 导入模型时，模型配置文件中的安装包依赖参数如何编写？

问题描述

从OBS中或者从容器镜像中导入模型时，开发者需要编写模型配置文件。模型配置文件描述模型用途、模型计算框架、模型精度、推理代码依赖包以及模型对外API接口。配置文件为JSON格式。配置文件中的“dependencies”，表示配置模型推理代码需要的依赖包，需要提供依赖包名、安装方式和版本约束的信息，详细参数见**模型配置文件编写说明**。导入模型时，模型配置文件中的安装包依赖参数“dependencies”如何编写？

解决方案

安装包存在前后依赖关系。例如您在安装“mmcv-full”之前，需要完成“Cython”、“pytest-runner”、“pytest”的安装，在配置文件中，您需要把“Cython”、“pytest-runner”、“pytest”写在“mmcv-full”的前面。

示例如下：

```
"dependencies": [
   {
   "installer": "pip",
   "packages": [
      {
         "package_name": "Cython"
      },
      {
         "package_name": "pytest-runner"
      },
      {
         "package_name": "pytest"
      },
      {
         "restraint": "ATLEAST",
         "package_version": "5.0.0",
         "package_name": "Pillow"
      },
      {
         "restraint": "ATLEAST",
         "package_version": "1.4.0",
         "package_name": "torch"
      },
      {
         "restraint": "ATLEAST",
         "package_version": "1.19.1",
         "package_name": "numpy"
      },
      {
         "package_name": "mmcv-full"
      }
    ]
  }
]
```

当"mmcv-full"安装失败，原因可能是基础镜像中没有安装gcc，无法编译导致安装失败，此时需要用户使用线下wheel包安装。

示例如下：

```
"dependencies": [
   {
   "installer": "pip",
   "packages": [
      {
         "package_name": "Cython"
      },
      {
         "package_name": "pytest-runner"
      },
      {
         "package_name": "pytest"
      },
      {
         "restraint": "ATLEAST",
         "package_version": "5.0.0",
         "package_name": "Pillow"
      },
      {
         "restraint": "ATLEAST",
         "package_version": "1.4.0",
         "package_name": "torch"
      },
      {
         "restraint": "ATLEAST",
         "package_version": "1.19.1",
         "package_name": "numpy"
```

```
    },
    {
      "package_name": "mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl"
    }
   ]
  }
]
```

模型配置文件的“dependencies”支持多个“dependency”结构数组以list形式填入。

示例如下：

```
"dependencies": [
  {
  "installer": "pip",
  "packages": [
    {
      "package_name": "Cython"
    },
    {
      "package_name": "pytest-runner"
    },
    {
      "package_name": "pytest"
    },
    {
      "package_name": "mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl"
    }
   ]
  },
  {
  "installer": "pip",
  "packages": [
    {
      "restraint": "ATLEAST",
      "package_version": "5.0.0",
      "package_name": "Pillow"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.4.0",
      "package_name": "torch"
    },
    {
      "restraint": "ATLEAST",
      "package_version": "1.19.1",
      "package_name": "numpy"
    },
   ]
  }
]
```

14.5.1.1.3 创建 AI 应用使用 Mindspore 引擎，报错 ModelArts.0107

在超算生态环境，Mindspore模型包规范对OM模型有校验。需要在模型包中添加一个空的.om文件，即可正常导入模型包。

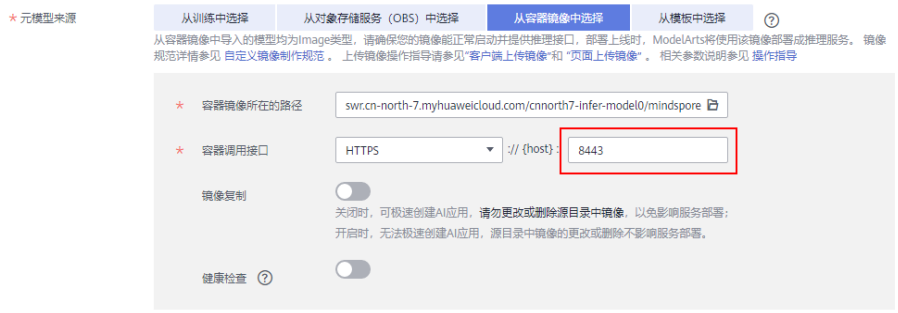
14.5.1.1.4 使用自定义镜像创建在线服务，如何修改默认端口

当模型配置文件中定义了具体的端口号，例如：8443，创建AI应用没有配置端口（默认端口号为8080），或者配置了其他端口号，均会导致服务部署失败。您需要把AI应用中的端口号配置为8443，才能保证服务部署成功。

修改默认端口号，具体操作如下：

1. 登录ModelArts控制台，左侧菜单选择“AI应用管理 > AI应用”；
2. 单击“创建”，进入创建AI应用界面，元模型选择“从容器镜像中选择”，选择自定义镜像；
3. 配置“容器调用接口”和端口号，端口号与模型配置文件中的端口保持一致；

图 14-36 修改端口号



4. 设置完成后，单击“立即创建”，等待AI应用状态变为“正常”；
5. 重新部署在线服务。

14.5.1.1.5 ModelArts 平台是否支持多模型导入

ModelArts平台从对象存储服务（OBS）中导入模型包适用于单模型场景。如果有多模型复合场景，推荐使用自定义镜像方式，通过从容器镜像（SWR）中选择元模型的方式创建AI应用部署服务。制作自定义镜像请参考**从0-1制作自定义镜像并创建AI应用**。

14.5.1.1.6 导入 AI 应用对于镜像大小限制

ModelArts部署使用的是容器化部署，容器运行时有空间大小限制，当用户的模型文件或者其他自定义文件，系统文件超过容器引擎空间大小时，会提示镜像内空间不足。

当前，公共资源池容器引擎空间的大小最大支持50G，专属资源池容器引擎空间的默认为50G，专属资源池容器引擎空间可在创建资源池时自定义设置，设置专属资源池容器引擎空间不会造成额外费用增加。

如果使用的是OBS导入或者训练导入，则包含基础镜像、模型文件、代码、数据文件和下载安装软件包的大小总和。

如果使用的是自定义镜像导入，则包含解压后镜像和镜像下载文件的大小总和。

14.5.2 部署上线

14.5.2.1 功能咨询

14.5.2.1.1 ModelArts 支持将模型部署为哪些类型的服务?

目前，仅支持在线服务和批量服务。

14.5.2.1.2 在线服务和批量服务有什么区别?

- 在线服务
  将模型部署为一个Web服务，您可以通过管理控制台或者API接口访问在线服务。

- 批量服务

  批量服务可对批量数据进行推理，完成数据处理后自动停止。

批量服务一次性推理批量数据，处理完服务结束。在线服务提供API接口，供用户调用推理。

14.5.2.1.3 服务预测请求体大小限制是多少?

服务部署完成且服务处于运行中后，可以往该服务发送推理的请求，请求的内容根据模型的不同可以是文本，图片，语音，视频等内容。

当使用调用指南页签中显示的调用地址（APIG网关服务的地址）预测时，对请求体的大小限制是12MB，超过12MB时，请求会被拦截。

如果是从ModelArts console的预测页签进行的预测，由于console的网络链路的不同，此时要求请求体的大小不超过8MB。

因此，尽量避免请求体大小超限。如果有高并发的大流量推理请求，请提工单联系专业服务支持。

14.5.2.1.4 部署服务如何选择计算节点规格?

部署服务时，用户需要指定节点规格进行服务部署，界面目前显示的节点规格是ModelArts根据用户的AI应用和资源池的节点规格计算得到，用户可以选择ModelArts提供的规格，也可以使用自定义规格（公共资源池不支持）。

计算节点规格主要是根据用户AI应用实际需要的资源进行选择，如AI应用正常运行需要3U10G的资源，那么需要选择大于3U10G的计算节点规格。确保服务能够部署成功正常运行。

图 14-37 选择计算节点规格

规格的使用注意事项如下：

1、权限控制

通用的计算节点规格是未做权限控制的，如modelarts.vm.cpu.2u，只要资源池有资源，就可以选择使用。ModelArts默认提供两个规格：CPU：modelarts.vm.cpu.2u和GPU：modelarts.vm.gpu.p4。

一些特殊的规格需要联系系统管理员增加权限。

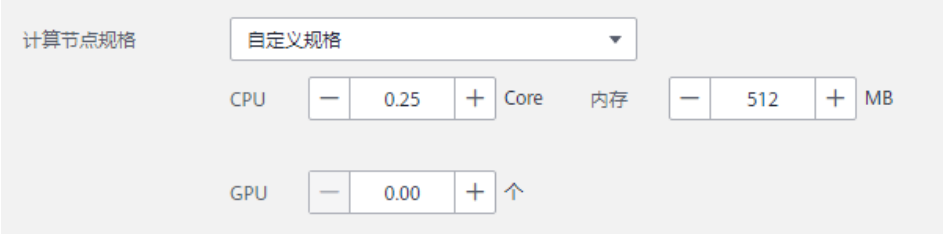
2、公共资源池的规格无法选择

共享池的资源是有限的，显示置灰表示当前规格的资源已经被用完。请选择未置灰的规格，也可以创建自己的专属资源池。

3、自定义规格

只有在专属资源池部署服务时，支持自定义资源规格。公共资源池部署服务不支持。

图 14-38 自定义规格



14.5.2.1.5 部署 GPU 服务支持的 Cuda 版本是多少?

默认支持Cuda版本为10.2，如果需要更高的版本，可以提工单申请技术支持。

14.5.2.2 在线服务

14.5.2.2.1 部署在线服务时，自定义预测脚本 python 依赖包出现冲突，导致运行出错

导入模型时，需同时将对应的推理代码及配置文件放置在模型文件夹下。使用Python编码过程中，推荐采用相对导入方式（Python import）导入自定义包。

如果ModelArts推理框架代码内部存在同名包，而又未采用相对导入，将会出现冲突，导致部署或预测失败。

14.5.2.2.2 在线服务的 API 接口组成规则是什么?

AI应用部署成在线服务后，用户可以获取API接口用于访问推理。

API接口组成规则如下：

```
https://域名/版本/infer/服务ID
```

示例如下：

```
https://6ac81cdfac4f4a30be95xxxbb682.apig.xxx.xxx.com/v1/infers/
468d146d-278a-4ca2-8830-0b6fb37d3b72
```

图 14-39 API 接口

14.5.2.2.3 配置了合理的服务部署超时时间，服务还是部署失败，无法启动

服务部署成功的标志是模型启动完成，如果没有配置健康检查，就无法检测到模型是否真实的启动。

在自定义镜像健康检查接口中，用户可以实现实际业务是否成功的检测。在创建AI应用时配置健康检查延迟时间，保证容器服务的初始化。

因此，推荐在创建AI应用时配置健康检查，并设置合理的延迟检测时间， 实现实际业务的是否成功的检测，确保服务部署成功。

14.6 API/SDK

14.6.1 ModelArts 的 API 或 SDK 支持模型下载到本地吗?

ModelArts的API和SDK不支持模型下载到本地，但训练作业输出的模型是存放在对象存储服务（OBS）里面的，您可以通过OBS的API或SDK下载存储在OBS中的文件。

14.6.2 ModelArts 的 SDK 支持哪些安装环境?

ModelArts的SDK支持在Notebook或本地环境中使用，但是不同环境下的不同架构，支持情况不同，如表14-11所示。

表 14-11 SDK 安装环境

开发环境	架构	是否支持
Notebook	ARM	是
	X86	是
本地环境	ARM	否
	X86	是

14.6.3 ModelArts 通过 OBS 的 API 访问 OBS 中的文件，算内网还是公网?

在同一区域，ModelArts通过OBS的API访问OBS中的文件属于内网通信，不消耗公网流量费。

若是通过互联网从OBS下载数据到本地，这时候会产生OBS公网流量费。

14.6.4 调用 API 提交训练作业后，能否绘制作业的资源占用率曲线?

调用API提交训练作业后，您可登录ModelArts控制台，在“训练管理 > 训练作业”中，单击“名称/ID”进入“训练作业详情”页面的“资源占用情况”模块，查看作业的资源占用率曲线。

14.6.5 使用 SDK 如何查看旧版专属资源池列表?

可参考如下代码查看旧版专属资源池列表：

```
from modelarts.session import Session
from modelarts.estimator import Estimator
algo_info = Estimator(modelarts_session=Session()).get_job_pool_list()print(algo_info)
```

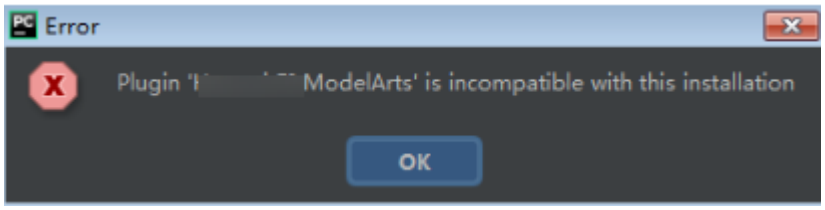
14.7 PyCharm Toolkit 使用

14.7.1 安装 ToolKit 工具时出现错误，如何处理?

问题现象

在安装ToolKit工具过程中，出现如下错误。

图 14-40 错误提示



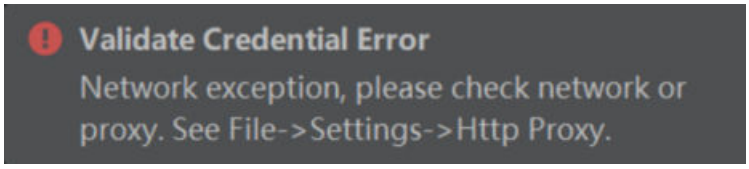
解决措施

此问题是因为插件版本和PyCharm版本不一致导致的，需要获取和PyCharm同一版本的插件安装，即2019.2或以上版本。

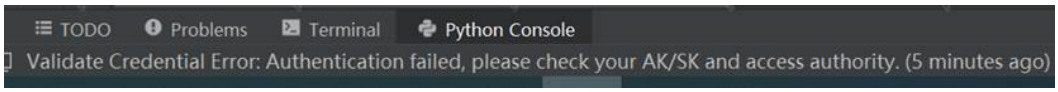
14.7.2 PyCharm ToolKit 工具中 Edit Credential 时，出现错误

问题现象

PyCharm ToolKit工具中Edit Credential时，提示Validate Credential error。



或



原因分析

- 可能原因一：Region等信息配置不正确
- 可能原因二：未配置hosts文件或者hosts文件信息配置不正确

- 可能原因三：网络代理设置
- 可能原因四：AK/SK不正确
- 可能原因五：电脑时间设置错误

解决措施

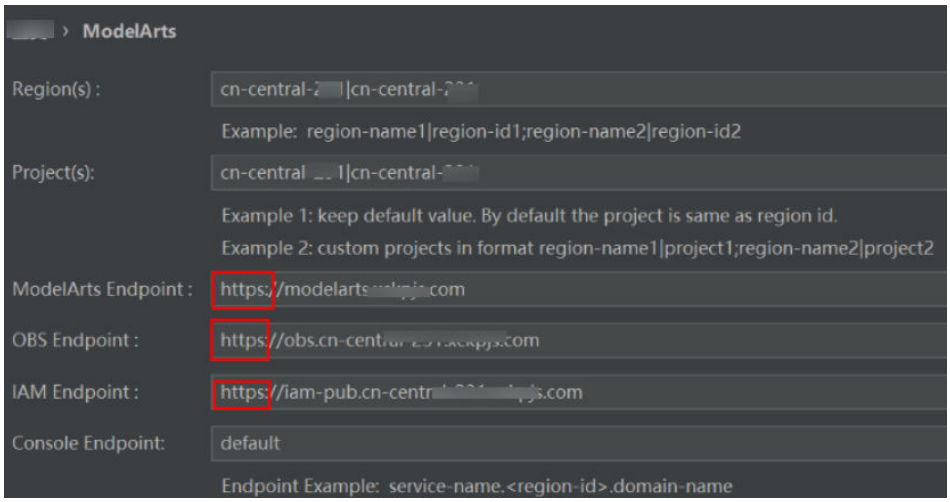
一、Region等信息配置不正确

配置正确的Region、Projects、Endpoint信息，具体请参考**使用YAML文件配置Toolkit**。

例如：Endpoint配置不正确也会导致认证失败。

错误示例：Endpoint参数前面带了https，正确的配置中不需要有https。

图 14-41 配置 ToolKit



二、未配置hosts文件或者hosts文件信息配置不正确

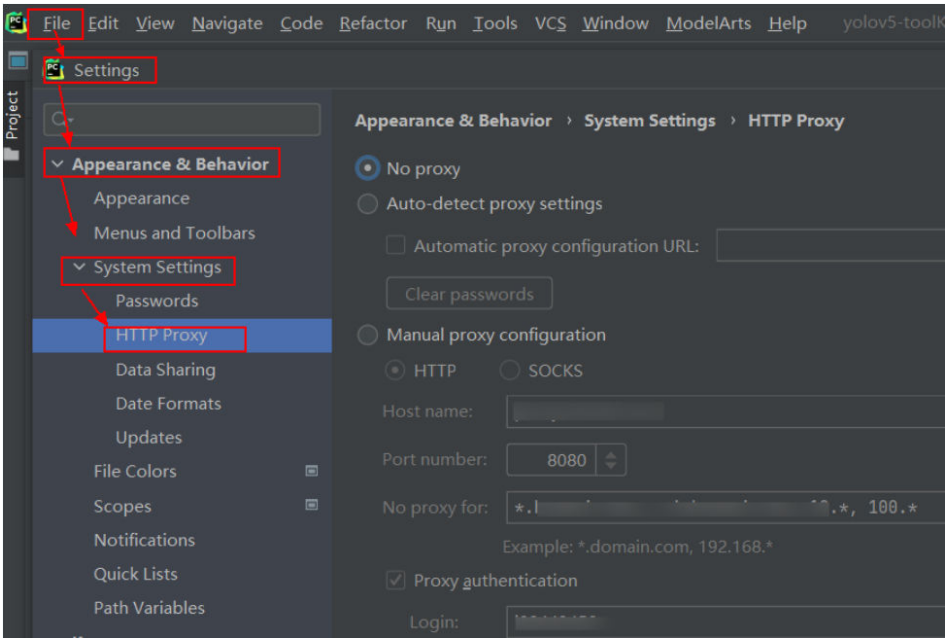
在本地PC的hosts文件中配置域名和IP地址的对应关系，具体请参考**配置域名和IP**。

三、网络代理设置

如果用户使用的网络有代理设置要求，请检查代理配置是否正确。也可以使用手机热点网络连接进行测试排查。

检查代理配置是否正确。

图 14-42 PyCharm 网络代理设置



四、AK/SK不正确

获取到的AK/SK信息不正确，请确认获取到正确的AK/SK信息再进行尝试，具体请参考**创建访问密钥（AK和SK）**。

云星账号的AK/SK请联系局点的技术支持获取。

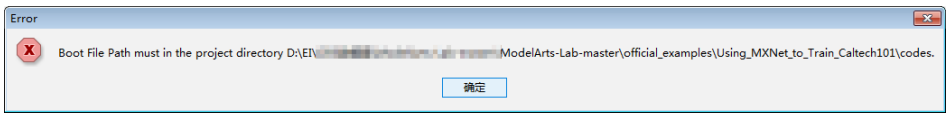
五、电脑时间设置错误

请设置电脑时间为正确时间。

14.7.3 为什么无法启动训练?

如果启动脚本选择了不属于本工程的代码，则无法启动训练，错误信息如下图所示。建议将启动脚本添加至本工程，或者是打开启动脚本所在工程后，再启动训练作业。

图 14-43 错误信息



14.7.4 提交训练作业时，出现 xxx isn't existed in train_version 错误

问题现象

提交训练作业时，出现xxx isn't existed in train_version错误，如下所示。

图 14-44 xxx isn't existed in train_version 错误

原因分析

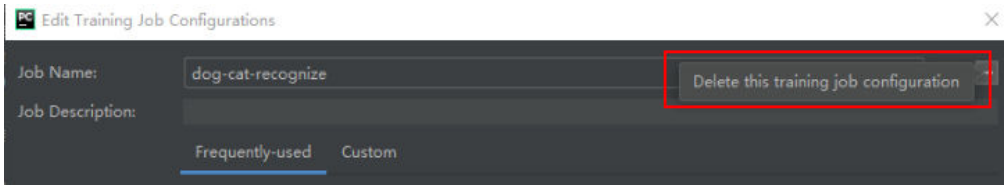
出现上述错误，是因为用户通过PyCharm ToolKit提交了训练作业后，又登录了ModelArts管理控制台删除了该训练作业导致的。

由于PyCharm ToolKit本地记录了云端ModelArts的训练作业ID，如果手动在ModelArts管理控制台删除，本地提交作业时候会报找不到该ID的作业。

解决措施

如果已经在ModelArts管理控制台删除了作业，需要在ToolKit也同步删除掉本地的配置信息，单击“Edit Training Configuration”，找到作业名称，单击右上角的减号并确认删除。

图 14-45 删除配置信息



在弹出的确认对话框中，确认信息无误后，单击“是”删除对应配置信息。删除后您可以创建新的训练作业配置并提交训练作业。

14.7.5 提交训练作业报错“Invalid OBS path”

当运行训练作业时，出现“Invalid OBS path”错误。

图 14-46 训练作业报错

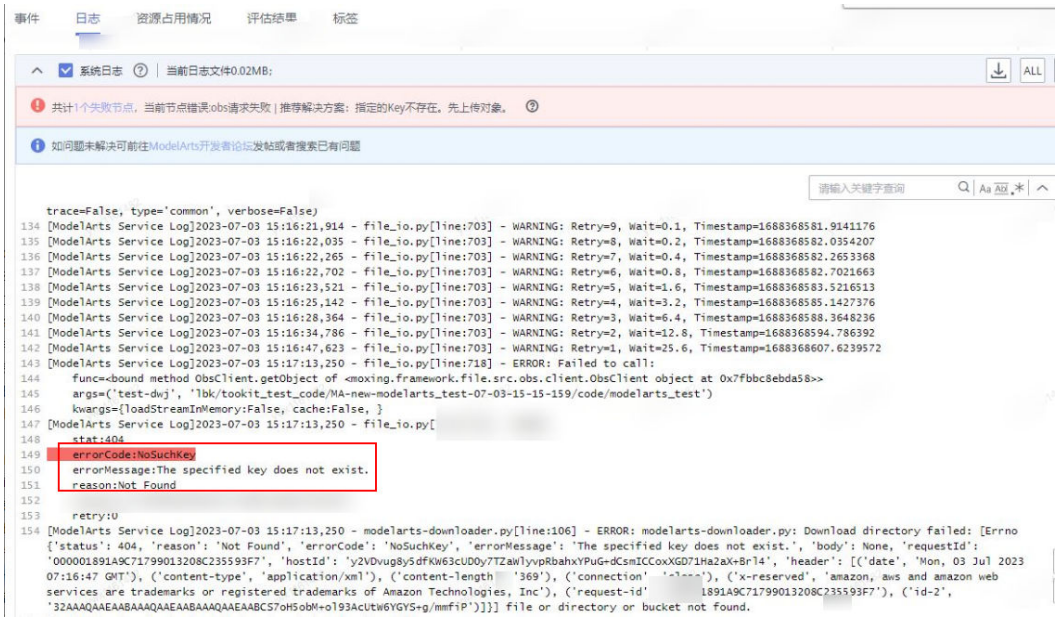
该问题可以通过以下方法排查：

- 如果是首次使用ModelArts的新用户，需要登录ModelArts管理控制台，在全局配置页面完成访问授权配置，推荐使用委托授权的方式。完成全局配置后，再次提交作业即可。
- 请排查所填写的Data Path in OBS是否存在，文件夹下是否有数据文件，如果没有，需要在OBS创建目录并上传训练数据到该目录。

14.7.6 使用 PyCharm Toolkit 提交训练作业报错 NoSuchKey

问题现象

使用PyCharm Toolkit提交训练作业时，报错，查看日志如下：



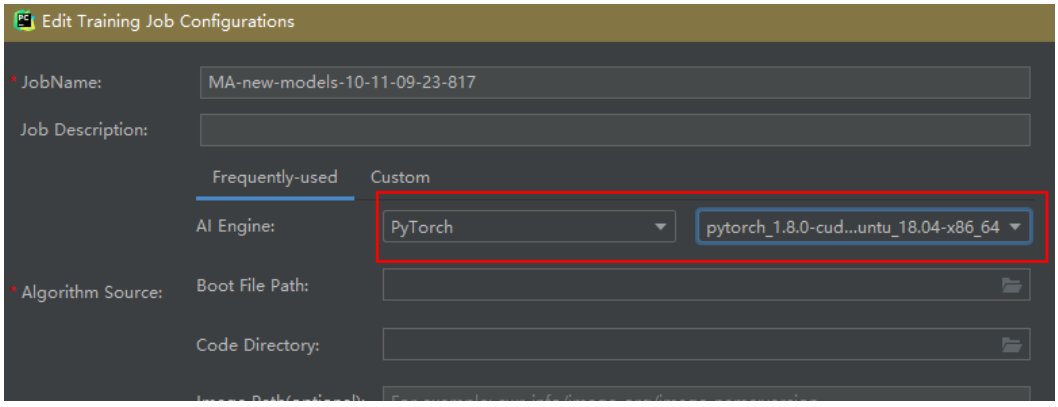
原因分析

检查配置后发现，是镜像版本太低，旧版的镜像与当前训练作业不兼容。

解决措施

使用PyCharm Toolkit提交训练作业时，常用框架选择训练作业支持的版本，具体支持哪些版本请参考**训练作业支持的AI引擎**。PyTorch的举例：不要选PyTorch-1.0.0、PyTorch-1.3.0、PyTorch-1.4.0。选择如下图：

图 14-47 选择训练作业支持的 AI 框架



14.7.7 部署上线时，出现错误

在部署上线前，您需要基于训练后的模型编写配置文件和推理代码。

如果您的模型存储路径下，缺少配置文件“confi.json”，或者缺少推理代码“customize_service.py”时，将出现错误，错误信息如下图所示。

解决方案：

请参考**模型包规范**写配置文件和推理代码，并存储至需部署的模型所在OBS目录下。

图 14-48 错误信息

14.7.8 如何查看 PyCharm ToolKit 的错误日志

PyCharm ToolKit的错误日志记录在PyCharm的“idea.log”中，以Windows为例，该文件的路径在“C:\Users\xxx\.IdeaIC2019.2\system\log\idea.log”。

在日志中搜索“modelarts”，可以查看所有和PyCharm ToolKit相关的日志。

14.7.9 如何通过 PyCharm ToolKit 创建多个作业同时训练?

PyCharm ToolKit一次只能运行一个作业，运行第二个作业时需要手动将第一个作业停止。

14.7.10 使用 PyCharm ToolKit ，提示 Error occurs when accessing to OBS

问题现象

查看PyCharm ToolKit的日志，报错信息为：Error occurs when accessing to OBS。

原因分析

可能是用户无OBS权限。

解决方法

判断用户是否有OBS权限。

步骤1 登录ModelArts控制台，进入“数据管理 > 数据集 ”，单击“创建数据集”，如果可以成功访问对应的OBS路径，表示用户有OBS权限。如果没有OBS权限，请执行**2**配置OBS权限。

步骤2 如没有OBS权限，请配置OBS权限配置。

----结束

15 故障排除

15.1 通用问题

15.1.1 ModelArts 中提示 OBS 路径错误

问题现象

- 在ModelArts中引用OBS桶路径时，提示找不到用户创建的OBS桶或提示ModelArts.2791：非法的OBS路径。
- 在对OBS桶操作时，出现Error: stat:403错误。
- Notebook中下载OBS文件时提示Permission denied。

原因分析

- 没有他人OBS桶的访问权限。
- ModelArts上没有配置委托授权。
- OBS文件加密上传导致。ModelArts不支持OBS加密文件。
- OBS桶的权限和访问ACL设置不正确导致。
- 创建训练作业时，代码目录和启动文件设置有误。

处理办法

检查您的账号是否有该OBS桶的访问权限

如果在使用Notebook时，需要访问其他账号的OBS桶，请查看您的账号是否有该OBS桶的访问权限。

检查委托授权

请前往全局配置，查看是否具有OBS访问授权。如果没有，请参考配置访问授权（全局配置）。

检查OBS桶是否为非加密桶

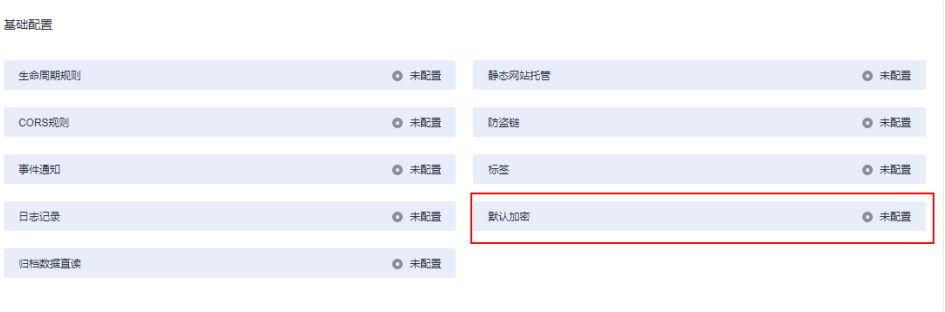
1. 进入OBS管理控制台，单击桶名称进入概览页。

2. 确保此OBS桶的加密功能关闭。如果此OBS桶为加密桶，可单击“默认加密”选项进行修改。

   说明

   创建OBS桶时，桶的存储类别请勿选择“归档存储”和“深度归档存储”，归档存储的OBS桶会导致模型训练失败。

   图 15-1 查看 OBS 桶是否加密



检查OBS文件是否为加密文件

1. 进入OBS管理控制台，单击桶名称进入概览页。
2. 单击左侧菜单栏对象，进入对象列表。单击存放文件的对象名称，并找到具体的文件，可在文件列表的“加密状态”列查看文件是否加密。文件加密无法取消，请先解除桶加密，重新上传图片或文件。

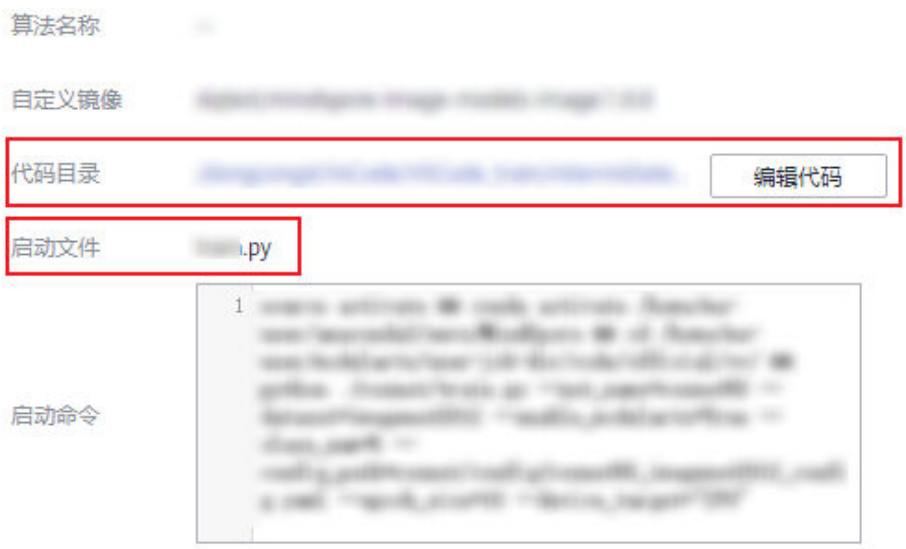
检查OBS桶的ACLs设置

1. 进入OBS管理控制台，查找对应的OBS桶，单击桶名称进入概览页。
2. 在左侧菜单栏选择“访问权限控制>桶ACLs”，检查当前账号是否具备读写权限，如果没有权限，请联系桶的拥有者配置权限。
3. 在左侧菜单栏选择“访问权限控制>桶策略”，检查当前OBS桶是否允许子用户访问。

检查训练作业的代码目录和启动文件地址

1. 进入ModelArts管理控制台，在“作业管理 > 训练作业”中查找到对应的“运行失败”的训练作业，单击作业“名称/ID”进入详情页。
2. 在详情页左侧栏中，查看代码目录和启动文件选择是否正确，且OBS文件名称中不能有空格。
   - 代码目录：需要选择到OBS目录。如果选择了文件，会提示非法的OBS路径。
   - 启动文件：需要选择以“.py”结尾的文件。如果选择的文件不是以“.py”结尾，会提示非法的OBS路径。

图 15-2 查看训练作业的代码目录和启动文件



如果还不能解决问题，请参考案例进行进一步排查。

15.2 自动学习

15.2.1 准备数据

15.2.1.1 数据集版本发布失败

出现此问题时，表示数据不满足数据管理模块的要求，导致数据集发布失败，无法执行自动学习的下一步流程。

请根据如下几个要求，检查您的数据，将不符合要求的数据排除后再重新启动自动学习的训练任务。

ModelArts.4710 OBS 权限问题

ModelArts在跟OBS交互时，由于权限相关的问题导致。当界面提示“OBS service Error Message”信息时，表示是由于OBS权限导致的问题，请参考如下步骤排除故障。如果界面错误提示不包含此信息，则是因为后台服务故障导致，建议联系技术支持。

1. 检查当前账号是否具备OBS权限。

   如果当前账号是个IAM用户（即子账号），需确认当前账号是否具备OBS服务操作权限。

   请参考**对象存储服务 OBS用户指南**的“ 产品介绍>OBS权限管理”，为当前IAM用户配置“作用范围”为“全局级服务”的“Tenant Administrator”策略，即拥有OBS服务所有操作权限。

   如果需要限制此IAM用户操作，仅为此用户配置OBS相关的最小化权限项，具体操作请参见**创建ModelArts自定义策略**。

2. 检查OBS桶是否具备权限。

说明

下方步骤描述中所致的OBS桶，指创建自动学习项目时，指定的OBS桶，或者是创建项目时选择的数据集，其数据存储所在的OBS桶。

- 检查当前账号具备OBS桶的读写权限（桶ACLs）
  - 进入OBS管理控制台，选择当前自动学习项目使用的OBS桶，单击桶名称进入概览页。
  - 在左侧菜单栏选择“访问权限控制>桶ACL”，检查当前账号是否具备读写权限，如果没有权限，请联系桶的拥有者配置权限。
- 确保此OBS桶是非加密桶
  i. 进入OBS管理控制台，选择当前自动学习项目使用的OBS桶，单击桶名称进入概览页。
  ii. 确保此OBS桶的加密功能关闭。如果此OBS桶为加密桶，可单击“默认加密”选项进行修改。

图 15-3 OBS 桶是否加密

- 确保归档数据直读功能关闭
  i. 进入OBS管理控制台，选择当前自动学习项目使用的OBS桶，单击桶名称进入概览页。
  ii. 确保此OBS桶的归档数据直读功能关闭。如果此功能开启，可单击“归档数据直读”选项进行修改。

图 15-4 关闭归档数据直读功能

ModelArts.4711 数据集标注样本数满足算法要求

每个类别至少包含5张以上图片。

ModelArts.4342 标注信息不满足切分条件

出现此故障时，建议根据如下建议，修改标注数据后重试。

- 多标签的样本（即一张图片包含多个标签），至少需要有2张。如果启动训练时，设置了数据集切分功能，如果多标签的数据少于2张，会导致数据集切分失败。建议检查您的标注信息，保证标注多标签的图片，超过2张。
- 数据集切分后，训练集和验证集包含的标签类别不一样。出现这种情况的原因：多标签场景下时，做随机数据切分后，包含某一类标签的样本均被划分到训练集，导致验证集无该标签样本。由于这种情况出现的概率比较小，可尝试重新发布版本来解决。

ModelArts.4371 数据集版本已存在

出现此错误码时，表示数据集版本已存在，请重新发布数据集版本。

ModelArts.4712 数据集正在执行导入或同步等其他任务

如果自动学习中使用的数据集，正在执行导入或同步数据的任务时，此时进行训练将出现此错误。建议等待其他任务完成后，再启动自动学习的训练任务。

15.2.1.2 数据集版本不合格

出现此问题时，表示数据集版本发布成功，但是不满足自动学习训练作业要求，因此出现数据集版本不合格的错误提示。

标注信息不满足训练要求

针对不同类型的自动学习项目，训练作业对数据集的要求如下。

- 图像分类：用于训练的图片，至少有2种以上的分类（即2种以上的标签），每种分类的图片数不少于5张。

- 物体检测：用于训练的图片，至少有1种以上的分类（即1种以上的标签），每种分类的图片数不少于5张。
- 预测分析：由于预测分析任务的数据集不在数据管理中进行统一管理，即使数据不满足要求，不在此环节出现故障信息。
- 声音分类：用于训练的音频，至少有2种以上的分类（即2种以上的标签），每种分类的音频数不少于5个。
- 文本分类：用于训练的文本，至少有2种以上的分类（即2种以上的标签），每种分类的文本数不少于20个。

15.2.2 模型训练

15.2.2.1 自动学习训练作业创建失败

出现此问题，一般是因为后台服务故障导致的，建议稍等片刻，然后重新创建训练作业。如果重试超过3次仍无法解决，请。

15.2.2.2 自动学习训练作业失败

训练作业创建成功，但是在运行过程中，由于一些故障导致作业运行失败。

首次请检查您的账户是否欠费。如果账号状态正常。请针对不同类型的作业进行排查。

- 针对**图像分类**、**声音分类**、**文本分类**的作业，排查思路请参见**确保OBS中的数据存在**、**检查OBS的访问权限**、**检查图片是否符合要求**。
- 针对**物体检测**作业，排查思路请参见**确保OBS中的数据存在**、**检查OBS的访问权限**、**检查图片是否符合要求**、**检查标注框是否符合要求（物体检测）**。
- 针对**预测分析**作业，排查思路请参见**确保OBS中的数据存在**、**检查OBS的访问权限**、**预测分析作业失败的排查思路**。

确保 OBS 中的数据存在

如果存储在OBS中的图片或数据被删除，且未同步至ModelArts自动学习或数据集中，则会导致任务失败。

建议前往OBS检查，确保数据存在。针对图像分类、声音分类、文本分类、物体检测等类型，可在自动学习的数据标注页面，单击“同步数据源”，将OBS中的数据重新同步至ModelArts中。

检查 OBS 的访问权限

如果OBS桶的访问权限设置无法满足训练要求时，将会出现训练失败。请排查如下几个OBS的权限设置。

- 当前账号具备OBS桶的读写权限（桶ACLs）
  a. 进入OBS管理控制台，选择当前自动学习项目使用的OBS桶，单击桶名称进入概览页。
  b. 在左侧菜单栏选择“访问权限控制>桶ACLs”，检查当前账号是否具备读写权限，如果没有权限，请联系桶的拥有者配置权限。
- 确保此OBS桶是非加密桶

a. 进入OBS管理控制台，选择当前自动学习项目使用的OBS桶，单击桶名称进入概览页。
b. 确保此OBS桶的加密功能关闭。如果此OBS桶为加密桶，可单击“默认加密”选项进行修改。

图 15-5 OBS 桶是否加密

- 确保归档数据直读功能关闭
  a. 进入OBS管理控制台，选择当前自动学习项目使用的OBS桶，单击桶名称进入概览页。
  b. 确保此OBS桶的归档数据直读功能关闭。如果此功能开启，可单击“归档数据直读”选项进行修改。

图 15-6 关闭归档数据直读功能

- 确保OBS中的文件是非加密状态
  上传图片或文件时不要选择KMS加密，否则会导致数据集读取失败。文件加密无法取消，请先解除桶加密，重新上传图片或文件。

图 15-7 OBS 桶中的文件未加密

检查图片是否符合要求

目前自动学习不支持四通道格式的图片。请检查您的数据，排除或删除四通道格式的图片。

检查标注框是否符合要求（物体检测）

目前物体检测仅支持矩形标注框。请确保所有图片的标注框为矩形框。

如果使用非矩形框，可能存在以下报错：

```
Error bandbox.
```

针对其他类型的项目（图像分类、声音分类等），无需关注此问题。

预测分析作业失败的排查思路

1. 检查用于预测分析的数据是否满足要求。

   由于预测分析任务未使用数据管理的功能发布数据集，因此当数据不满足训练作业要求时，会出现训练作业运行失败的错误。

   建议检查用于训练的数据，是否满足预测分析作业的要求。要求如下所示，如果数据满足要求，执行下一步检查。如果不满足要求，请根据要求仅需数据调整后再重新训练。

   - 文件规范：名称由以字母数字及中划线下划线组成，以'.csv'结尾，且文件不能直接放在OBS桶的根目录下，应该存放在OBS桶的文件夹内。如：“/obs-xxx/data/input.csv”。
   - 文件内容：文件保存为“csv”文件格式，文件内容以换行符（即字符“\n”，或称为LF）分隔各行，行内容以英文逗号（即字符“,”）分隔各列。文件内容不能包含中文字符，列内容不应包含英文逗号、换行符等特殊字符，不支持引号语法，建议尽量以字母及数字字符组成。
   - 训练数据：训练数据列数一致，总数据量不少于100条不同数据（有一个特征取值不同，即视为不同数据）。训练数据列内容不能有时间戳格式（如：yy-mm-dd、yyyy-mm-dd等）的数据。确保指定标签列的取值至少有两个且无数据缺失，除标签列外数据集中至少还应包含两个有效特征列（列的取值至少有两个且数据缺失比例低于10%）。训练数据的csv文件不能包含表头，否则会导致训练失败。当前由于特征筛选算法限制，标签列建议放在数据集最后一列，否则可能导致训练失败。

2. 由于ModelArts会自动对数据进行一些过滤，过滤后再启动训练作业。当预处理后的数据不满足训练要求时，也会导致训练作业运行失败。

   对于数据集中列的过滤策略如下所示：

   - 如果某一列空缺的比例大于系统设定的阈值（0.9），此列数据在训练时将被剔除。
   - 如果某一列只有一种取值（即每一行的数据都是一样的），此列数据在训练时将被剔除。

- 对于非纯数值列，如果此列的取值个数等于行数（即每一行的数值都是不一样的），此列数据在训练时将被剔除。

经过上述过滤后，如果数据集不再满足第一点中关于训练数据的要求，则会导致训练失败或无法进行。建议完善数据后，再启动训练。

3. 数据集文件有以下限制：
   a. 如果您使用2u8g规格，测试建议数据集文件应小于10MB。当文件大小符合限制要求，如果存在极端的数据规模（行数列数之积）时，仍可能会导致训练失败，建议的数据规模低于10000。

      如果您使用8u32g规格，测试建议数据集文件应小于100MB。当文件大小符合限制要求，如果存在极端的数据规模（行数列数之积）时，仍可能会导致训练失败，建议的数据规模低于1000000。
4. 如果上述排查操作仍无法解决，请。

15.2.3 部署上线

15.2.3.1 部署上线任务提交失败

当出现此错误时，一般情况是由于账号的配额受限导致的。

在自动学习项目中，启动部署后，会自动将模型部署为一个在线服务，如果由于配额限制（即在线服务的个数超出配额限制），导致无法将模型部署为服务。此时会在自动学习项目中提示“部署上线任务提交失败”的错误。

修改建议

- 方法1：进入“部署上线>在线服务”页面，将不再使用的服务删除，释放资源。
- 方法2：如果您部署的在线服务仍需继续使用，建议申请增加配额。

15.2.3.2 部署上线失败

出现此问题，一般是因为后台服务故障导致的，建议稍等片刻，然后重新部署在线服务。如果重试超过3次仍无法解决，请获取如下信息，并协助解决故障。

- 获取服务ID。

  进入“部署上线>在线服务”页面，在服务列表中找到自动学习任务中部署的在线服务，自动学习部署的服务都是以“exeML-”开头的。单击服务名称进入服务详情页面，在“基本信息”区域，获取“服务ID”的值。

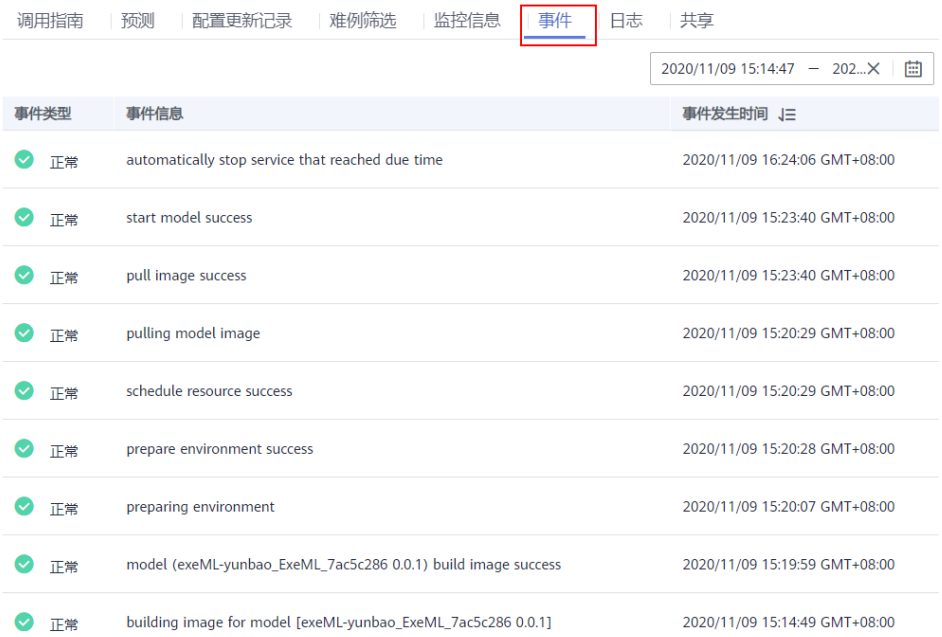
  图 15-8 获取服务 ID

- 获取在线服务事件信息。

  进入服务详情页面后，单击“事件”页签，将事件信息表截图后反馈给技术支持人员。

图 15-9 获取事件信息



15.2.4 模型发布

15.2.4.1 模型发布任务提交失败

出现此问题，一般是因为后台服务故障导致的，建议稍等片刻，然后重新创建训练作业。如果重试超过3次仍无法解决，请。

15.2.4.2 模型发布失败

出现此问题，一般是因为后台服务故障导致的，建议稍等片刻，然后重新创建训练作业。如果重试超过3次仍无法解决，请获取如下信息，并协助解决故障。

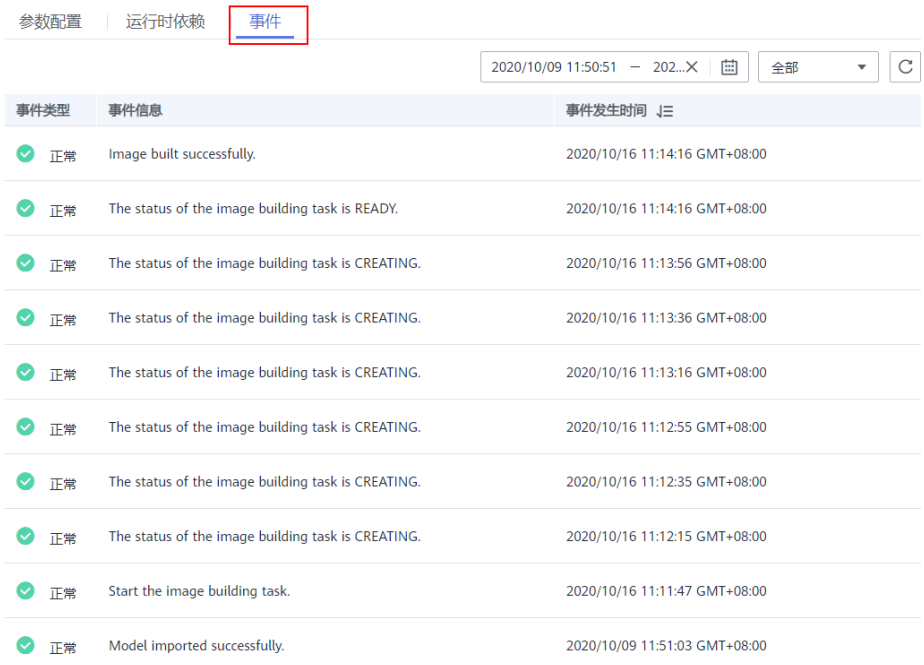
- 获取模型ID。

  进入“AI应用管理>AI应用”页面，在AI应用列表中找到自动学习任务中自动创建的模型，自动学习产生的模型都是以“exeML-”开头的。单击模型名称进入模型详情页面，在“基本信息”区域，获取“ID”的值。

图 15-10 获取模型 ID

- 获取模型事件信息。
  进入模型详情页面后，单击“事件”页签，将事件信息表截图后反馈给技术支持人员。

  图 15-11 获取事件信息



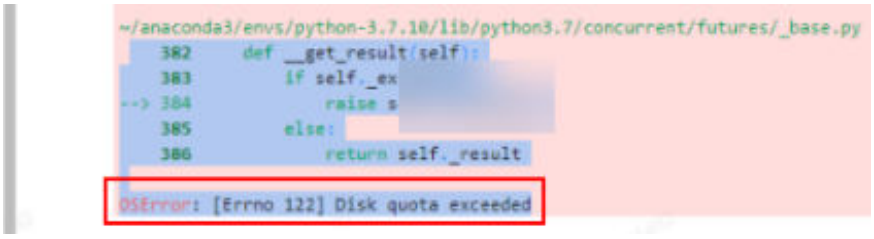
15.3 开发环境

15.3.1 环境配置故障

15.3.1.1 Notebook 提示磁盘空间已满

问题现象

- 在使用Notebook时，提示磁盘空间已满：No Space left on Device。
- 在Notebook执行代码时，出现如下报错，提示：Disk quato exceeded。



原因分析

- 在JupyterLab浏览器左侧导航删除文件后，会默认放入回收站占用内存，导致磁盘空间不足。

- 磁盘配额不足。

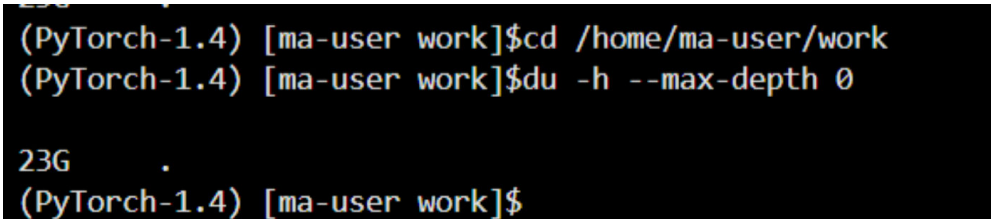
处理方法

查看虚拟机所使用的存储空间，再查看回收站文件占用内存，根据实际删除回收站里不需要的大文件。

1. 在Notebook实例详情页，查看实例的存储容量。

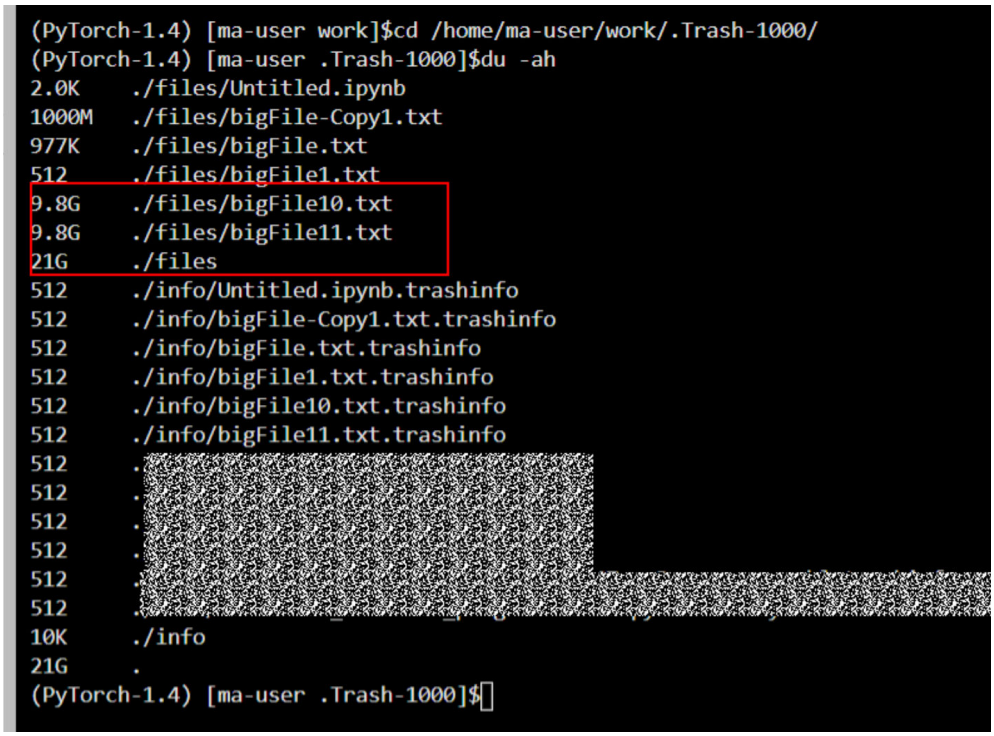
2. 执行如下命令，排查虚拟机所使用的存储空间，一般接近存储容量，请排查回收站占用内存。

```
cd /home/ma-user/work
du -h --max-depth 0
```



3. 执行如下命令，排查回收站占用内存（回收站文件默认在/home/ma-user/work/.Trash-1000/files下）。

```
cd /home/ma-user/work/.Trash-1000/
du -ah
```



4. 根据实际删除回收站不需要的大文件。（注：请谨慎操作，文件删除后不可恢复）

```
rm {文件路径}
```

```
(PyTorch-1.4) [ma-user .Trash-1000]$pwd
/home/ma-user/work/.Trash-1000
(PyTorch-1.4) [ma-user .Trash-1000]$rm /home/ma-user/work/.Trash-1000/files/bigFile10.txt
(PyTorch-1.4) [ma-user .Trash-1000]$rm /home/ma-user/work/.Trash-1000/files/bigFile11.txt
```

说明

如果删除的文件夹或者文件中带有空格，需要给文件夹或文件加上单引号。如图示例：

```
(PyTorch-1.8) [ma-user files]$rm -rf ./'1 6'
(PyTorch-1.8) [ma-user files]$ll
```

5. 执行如下命令，再次检查虚拟机所使用的存储空间。

```
cd /home/ma-user/work
du -h --max-depth 0
```

6. 如果Notebook实例的存储配置采用的是云硬盘EVS，可在Notebook详情页申请扩容磁盘。

建议与总结

建议在使用Notebook时注意磁盘空间大小，随时删除不需要的文件。以免因磁盘空间问题导致训练失败。

15.3.1.2 Notebook 中使用 Conda 安装 Keras 2.3.1 报错

问题现象

使用Conda安装Keras 2.3.1版本报错。

```
conda install keras=2.3.1
```

```
/home/ma-user/anaconda3/lib/python3.7/site-packages/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.26.12)
  RequestsDependencyWarning)
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: -
Found conflicts! Looking for incompatible packages.
This can take several minutes.  Press CTRL-C to abort.
                                                                                failed

# >>>>>>>>>>>>>>>>>>>>>> ERROR REPORT <<<<<<<<<<<<<<<<<<<<<<

    Traceback (most recent call last):
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/cli/install.py", line 265, in install
        should_retry_solve=(_should_retry_unfrozen or repodata_fn != repodata_fns[-1]),
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 117, in solve_for_transaction
        should_retry_solve)
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 158, in solve_for_diff
        force_remove, should_retry_solve)
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 275, in solve_final_state
        ssc = self._add_specs(ssc)
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 696, in _add_specs
        raise UnsatisfiableError({})
    conda.exceptions.UnsatisfiableError:
    Did not find conflicting dependencies. If you would like to know which
    packages conflict ensure that you have enabled unsatisfiable hints.

    conda config --set unsatisfiable_hints True
```

原因分析

可能是Conda网络不通，请使用**pip install**命令安装。

解决方法

执行 **!pip install keras==2.3.1**命令安装Keras。

```
!pip install keras==2.3.1
```

```
Looking in indexes: http://repo.myhuaweicloud.com/repository/pypi/simple
Collecting keras==2.3.1
  Using cached http://repo.myhuaweicloud.com/repository/pypi/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bdc0fe9e589a60ceb/Keras-2.3.1-py2.py3-none-any.whl (377 kB
Requirement already satisfied: keras-applications>=1.0.6 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.0.8)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.1.2)
Requirement already satisfied: numpy>=1.9.1 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.21.5)
Requirement already satisfied: pyyaml in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (5.1)
Requirement already satisfied: scipy>=0.14 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.7.3)
Requirement already satisfied: h5py in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (2.8.0)
Requirement already satisfied: six>=1.9.0 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.16.0)
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: Keras 2.2.4
    Uninstalling Keras-2.2.4:
      Successfully uninstalled Keras-2.2.4
Successfully installed keras-2.3.1
```

15.3.1.3 Notebook 中安装依赖包报错 ERROR: HTTP error 404 while getting xxx

问题现象

在Notebook中安装依赖包时报错，报错截图如下：

```
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /home/ma-user/anaconda3/envs/llama2/lib/python3.10/site-packages (from aiohttp->datasets) (3
.1.0)
Collecting multidict<7.0,>=4.5 (from aiohttp->datasets)
  Using cached http://repo.myhuaweicloud.com/repository/pypi/packages/df/93/34efbfa7aa778b04b365960f52f7071d7942ce386572aac8940ae032dd48/multidict-6.0.2-cp
310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (114 kB)
Collecting async-timeout<5.0,>=4.0.0a3 (from aiohttp->datasets)
  ERROR: HTTP error 404 while getting http://repo.myhuaweicloud.com/repository/pypi/packages/a7/fa/e01228c2938de91d47b307831c62ab9e4001e747789d0b05baf779a6
488c/async_timeout-4.0.3-py3-none-any.whl.metadata
ERROR: 404 Client Error: Not Found for url: http://repo.myhuaweicloud.com/repository/pypi/packages/a7/fa/e01228c2938de91d47b307831c62ab9e4001e747789d0b05ba
f779a6488c/async_timeout-4.0.3-py3-none-any.whl.metadata
(llama2) [ma-user work]$pip install datasets
```

原因分析

pypi源没有这个包或源不可用。

解决方案

使用别的源下载。

```
pip install -i 源地址 包名
```

15.3.1.4 Notebook 中已安装对应库，仍报错 import numba ModuleNotFoundError: No module named 'numba'

问题现象

在Notebook中使用**!pip install numba**命令安装了numba库且运行正常（且已保存为自定义镜像）， 然后使用DataArts执行此脚本的任务时提示没有这个库。

原因分析

客户创建了多个虚拟环境，numba库安装在了python-3.7.10中，如图15-12所示。

图 15-12 查询创建的虚拟环境

```
[ma-user work]$conda info --envs
/home/ma-user/anaconda3/lib/python3.7/site-packages/requests/__init__.
d version!
  RequestsDependencyWarning)
# conda environments:
#
base                     /home/ma-user/anaconda3
PyTorch-1.8           *  /home/ma-user/anaconda3/envs/PyTorch-1.8
python-3.7.10            /home/ma-user/anaconda3/envs/python-3.7.10
```

解决方案

在Terminal中执行**conda deactivate**命令退出当前虚拟环境，默认进入base环境。执行**pip list**命令查询已安装的包，然后安装需要的依赖进行保存，最后切换至指定的虚拟环境后再运行脚本。

```
Using user ma-user
Ubuntu 18.04.6 LTS, CUDA-10.2
Tips:
1) Navigate to the target conda environment. For details, see /home/ma-user/README.
2) Copy (Ctrl+C) and paste (Ctrl+V) on the jupyter terminal.
3) Store your data in /home/ma-user/work, to which a persistent volume is mounted.
(PyTorch-1.8) [ma-user work]$conda deactivate
(base) [ma-user work]$conda deactivate
[ma-user work]$pip list
Package                       Version
----------------------------- -------------------------
absl-py                       1.3.0
addict                        2.4.0
APScheduler                   3.9.1
```

15.3.2 实例故障

15.3.2.1 创建 Notebook 失败，查看事件显示 JupyterProcessKilled

问题现象

创建Notebook失败，查看事件显示JupyterProcessKilled。

原因分析

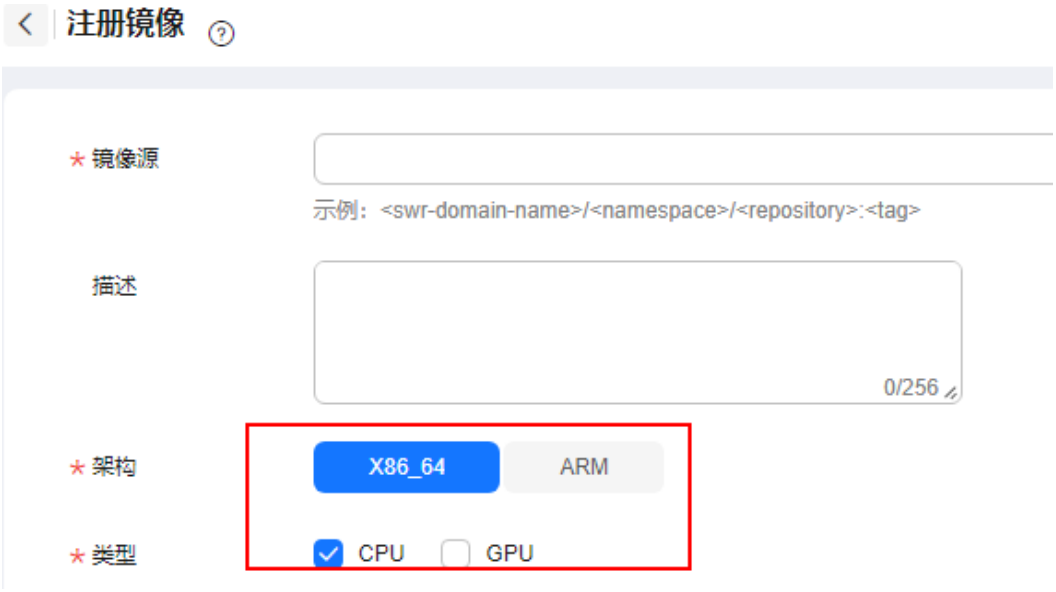
出现此故障是因为Jupyter进程被清理掉了，一般情况Notebook会自动重启的，如果没有自动重启，创建一直失败，请确认是否是自定义镜像的问题。

解决方案

排查是否是自定义镜像的问题。

自定义镜像构建完成，在ModelArts镜像管理注册时，架构和类型需要和源镜像保持一致。

图 15-13 注册镜像



15.3.2.2 创建 Notebook 实例后无法打开页面，如何处理?

如果您在创建Notebook实例之后，打开Notebook时，因报错导致无法打开页面，您可以根据以下对应的错误码来排查解决。

打开 Notebook 显示黑屏

Notebook打开后黑屏，由于代理问题导致，切换代理。

打开 Notebook 显示空白

- 打开Notebook时显示空白，请清理浏览器缓存后尝试重新打开。
- 检查浏览器是否安装了过滤广告组件，如果是，请关闭该组件。

报错 404

如果是IAM用户在创建实例时出现此错误，表示此IAM用户不具备对应存储位置（OBS桶）的操作权限。

解决方法：

1. 使用账号登录OBS，并将对应OBS桶的访问权限授予该IAM用户。
2. IAM用户获得权限后，登录ModelArts管理控制台，删除该实例，然后重新使用此OBS路径创建Notebook实例。

报错 503

如果出现503错误，可能是由于该实例运行代码时比较耗费资源。建议先停止当前Notebook实例，然后重新启动。

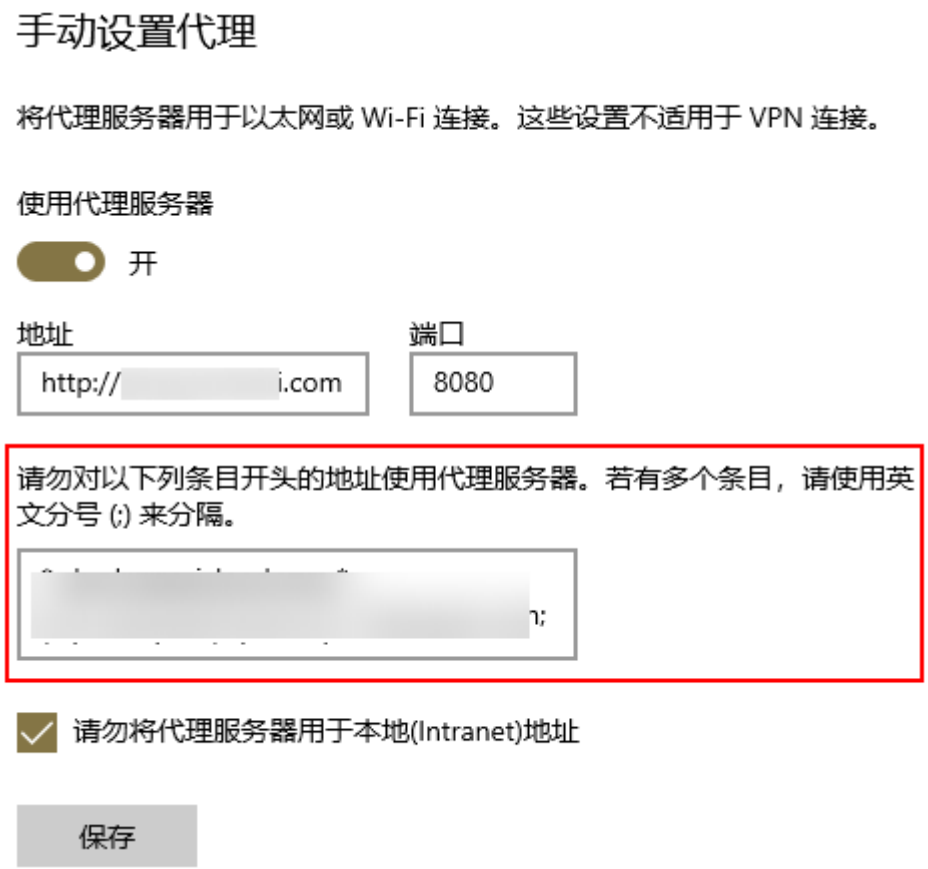
报错 500

Notebook JupyterLab页面无法打开，报错500，可能是工作目录work下的磁盘空间满了，请排查并清理磁盘空间。

报错 This site can't be reached

创建完Notebook后，单击操作列的“打开”，报错如下：

解决方案：复制页面的域名，添加到windows代理“请勿对以下列条目开头的地址使用代理服务器”中，然后保存就可以正常打开。



15.3.2.3 使用 pip install 时出现“没有空间”的错误

问题现象

在Notebook实例中，使用**pip install**时，出现“No Space left...”的错误。

解决办法

建议使用**pip install --no-cache **** 命令安装，而不是使用**pip install ****。加上“--no-cache”参数，可以解决很多此类报错。

15.3.2.4 出现“save error”错误，可以运行代码，但是无法保存

如果当前Notebook还可以运行代码，但是无法保存，保存时会提示“save error”错误。大多数原因是WAF安全拦截导致的。

当前页面，即用户的输入或者代码运行的输出有一些字符被拦截，认为有安全风险。出现此问题时，请提交工单，联系专业的工程师帮您核对并处理问题。

15.3.2.5 出现 ModelArts.6333 错误，如何处理?

问题现象

在使用Notebook过程中，界面出现“ModelArts.6333”报错信息。

原因分析

可能由于实例过负载引起故障，Notebook正在自动恢复中，请刷新页面并等待几分钟。常见原因是内存占用满。

处理方法

当出现此错误时，Notebook会自动恢复，您可以刷新页面，等待几分钟。

由于出现此错误，常见原因是内存占用满导致的，您可以尝试使用如下方法，从根本上解决错误。

- 方法1：将Notebook更换为更高规格的资源。
- 方法2：可以参考如下方法调整代码中的参数，减少内存占用。如果代码调整后仍然出现内存不足的情况，请使用方法1。
  a. 调用**sklearn**方法**silhouette_score(addr_1,siteskmeans.labels)**，可以指定参数**sample_size**来减少内存占用。
  b. 调用**train**方法的时候可以尝试减少**batch_size**等参数。

15.3.2.6 打开 Notebook 实例提示 token 不存在或者 token 丢失如何处理?

问题现象

把已打开的Notebook url发送给他人使用，他人无法打开，报错“……lost token or incorrect token……”。

原因分析

原因是由于其他人没有此账号的令牌导致。

解决方案

在此url后面加上Notebook拥有者的token。

15.3.3 代码运行故障

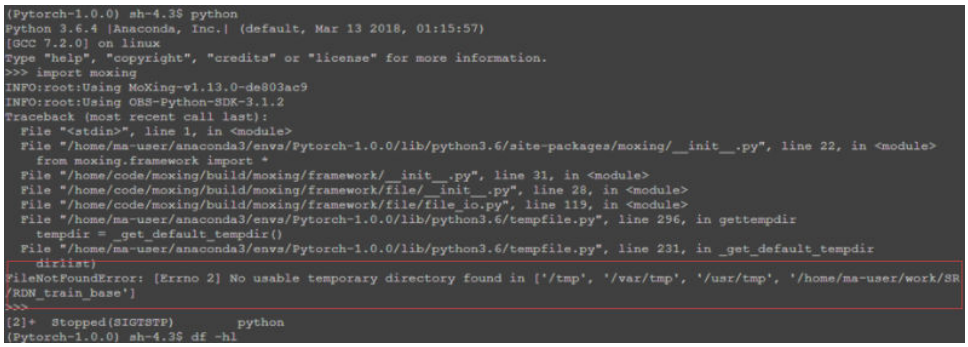
15.3.3.1 Notebook 运行代码报错，在'/tmp'中到不到文件

问题现象

使用Notebook运行代码，报错：

```
FileNotFoundError: [Error 2] No usable temporary directory found in ['/tmp', '/var/tmp', '/usr/tmp', 'home/ma-user/work/SR/RDN_train_base']
```

图 15-14 运行代码报错



原因分析

根据报错提示，需要排查是否将大量数据被保存在“/tmp”中。

处理方法

1. 进入到“Terminal”界面。在“/tmp”目录下，执行命令**du -sh ***，查看该目录下的空间占用情况。

   ```
   sh-4.3$cd /tmp
   sh-4.3$du -sh *
   4.0K    core-js-banners
   0       npm-19-41ed4c62
   6.7M    v8-compile-cache-1000
   ```

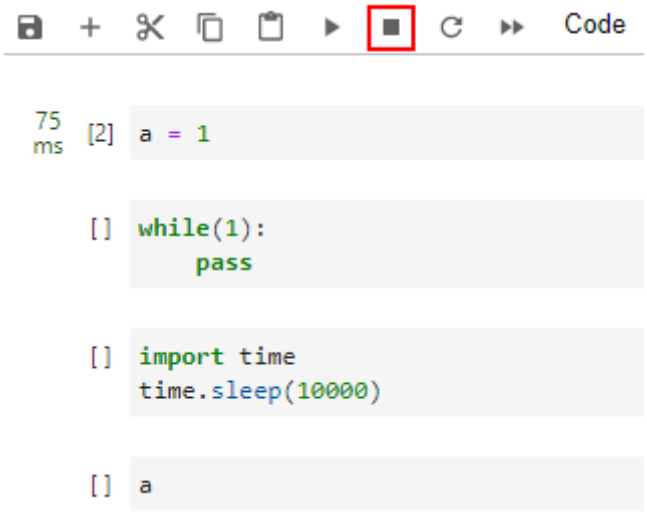
2. 请删除不用的大文件。
   a. 删除示例文件“test.txt”： **rm -f /home/ma-user/work/data/test.txt**
   b. 删除示例文件夹“data”： **rm -rf /home/ma-user/work/data/**

15.3.3.2 Notebook 无法执行代码，如何处理?

当Notebook出现无法执行时，您可以根据如下几种情况判断并处理。

1. 如果只是Cell的执行过程卡死或执行时间过长，如图15-15中的第2个和第3个Cell，导致第4个Cell无法执行，但整个Notebook页面还有反应，其他Cell也还可以单击，则直接单击下图中红色方框处的“interrupt the kernel”，停止所有Cell的执行，同时会保留当前Notebook中的所有变量空间。

   图 15-15 停止所有 Cell

2. 如果整个Notebook页面也已经无法使用，单击任何地方都无反应，则关闭Notebook页面，关闭ModelArts管理控制台页面。然后，重新打开管理控制台，打开之前无法使用的Notebook，此时的Notebook仍会保留无法使用之前的所有变量空间。
3. 如果重新打开的Notebook仍然无法使用，则进入ModelArts管理控制台页面的Notebook列表页面，“停止”此无法使用的Notebook。待Notebook处于“停止”状态后，再单击“启动”，重新启动此Notebook，并打开Notebook。此时，Notebook仍会保留无法使用之前的所有变量空间。

15.3.3.3 运行训练代码，出现 dead kernel，并导致实例崩溃

在Notebook实例中运行训练代码，如果数据量太大或者训练层数太多，亦或者其他原因，导致出现“内存不够”问题，最终导致该容器实例崩溃。

出现此问题后，系统将自动重启Notebook，来修复实例崩溃的问题。此时只是解决了崩溃问题，如果重新运行训练代码仍将失败。如果您需要解决“内存不够”的问题，建议您创建一个新的Notebook，使用更高规格的资源池，比如专属资源池来运行此训练代码。已经创建成功的Notebook不支持选用更高规格的资源规格进行扩容。

15.3.3.4 如何解决训练过程中出现的 cudaCheckError 错误?

问题现象

Notebook中，运行训练代码出现如下错误。

```
cudaCheckError() failed : no kernel image is available for execution on the device
```

原因分析

因为编译的时候需要设置setup.py中编译的参数arch和code和电脑的显卡匹配。

解决方法

对于Tesla V100的显卡，GPU算力为**-gencode arch=compute_70,code=[sm_70,compute_70]**，设置setup.py中的编译参数即可解决。

15.3.3.5 开发环境提示空间不足，如何解决?

当提示空间不足时，推荐使用EVS类型的Notebook实例。

参考**如何在Notebook中上传下载OBS文件?** 操作指导，针对原有的Notebook，首先将代码和数据上传至OBS桶中。然后创建一个EVS类型的Notebook，将此OBS中的文件下载至Notebook本地（指新建的EVS类型Notebook）。

15.3.3.6 如何处理使用 opencv.imshow 造成的内核崩溃?

问题现象

当在Notebook中使用opencv.imshow后，会造成Notebook崩溃。

原因分析

opencv的cv2.imshow在jupyter这样的client/server环境下存在问题。 而matplotlib不存在这个问题。

解决方法

参考如下示例进行图片显示。注意opencv加载的是BGR格式， 而matplotlib显示的是RGB格式。

Python语言：

```
from matplotlib import pyplot as plt
import cv2
img = cv2.imread('图片路径')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('my picture')
plt.show()
```

15.3.3.7 使用 Windows 下生成的文本文件时报错找不到路径?

问题现象

当在Notebook中使用Windows下生成的文本文件时，文本内容无法正确读取，可能报错找不到路径。

原因分析

Notebook是Linux环境，和Windows环境下的换行格式不同，Windows下是CRLF，而Linux下是LF。

解决方法

可以在Notebook中转换文件格式为Linux格式。

shell语言：

```
dos2unix 文件名
```

15.3.3.8 创建 Notebook 文件后，右上角的 Kernel 状态为“No Kernel”如何处理?

问题现象

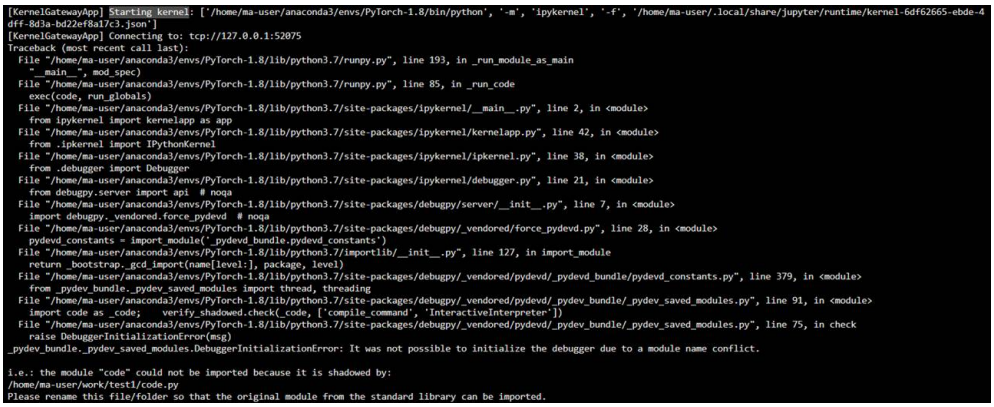
现象：创建Notebook文件后，右上角的Kernel状态为“No Kernel”。

原因分析

可能因为用户工作目录下的code.py和创建kernel依赖的import code文件名称冲突。

解决方案

1. 查看“/home/ma-user/log/”下以“kernelgateway”开头的最新日志文件，搜索“Starting kernel”附近的日志。若看到如下类似的堆栈，可看到是因为用户工作目录下的“code.py”和创建kernel依赖的import code文件名冲突：



2. 重命名当前工作目录下和创建kernel依赖的库文件冲突的文件名称。

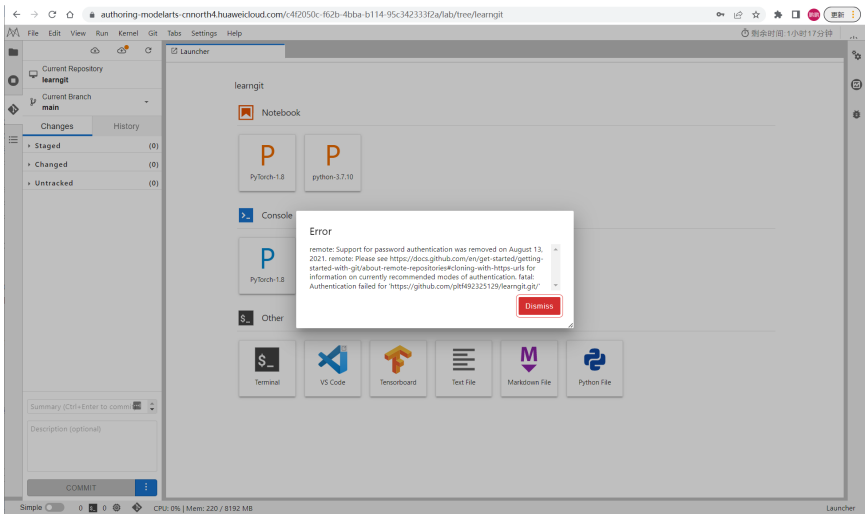
   常见容易冲突的文件：code.py、select.py。

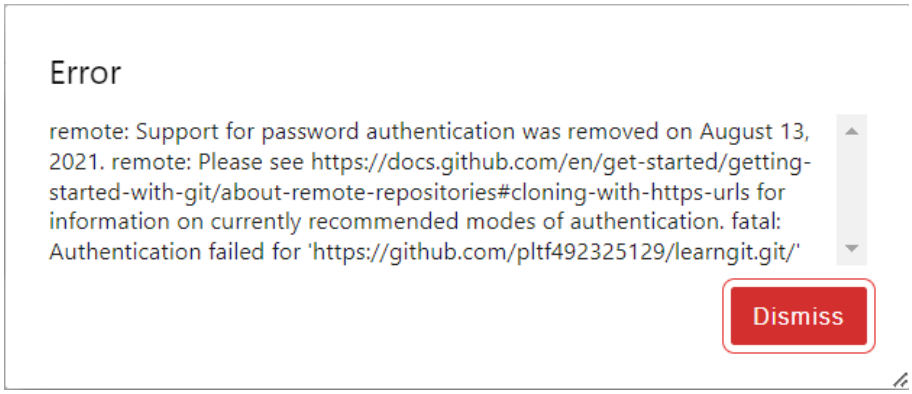
15.3.4 JupyterLab 插件故障

15.3.4.1 git 插件密码失效如何解决?

问题现象

在JupyterLab中使用git插件时，当git clone私有仓库和git push文件时会出现如下报错：



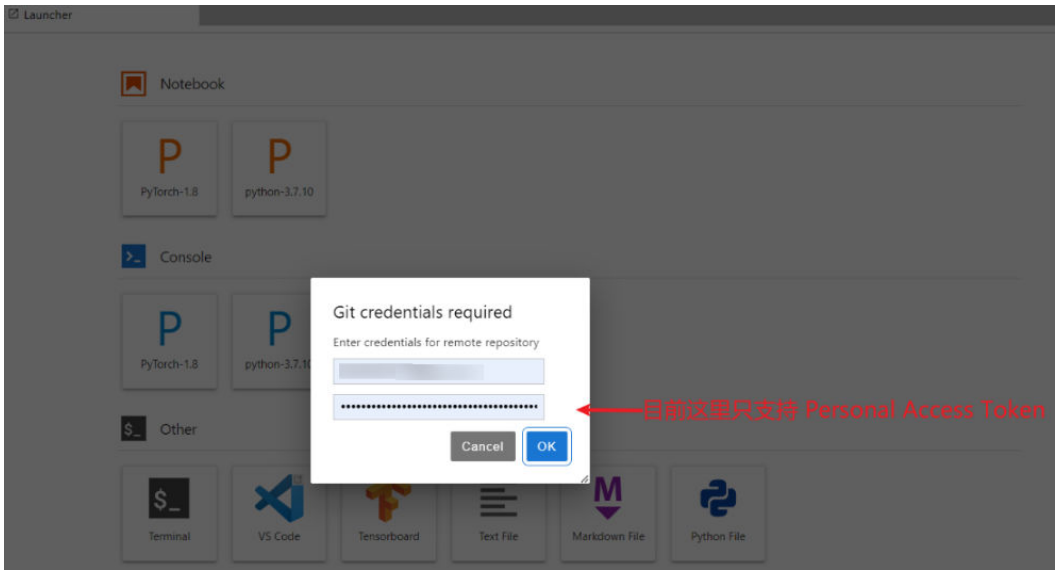


原因分析

原因为Github已取消密码授权方式，此时在git clone私有仓库和git push文件时需要在授权方式框中输入token。

解决方案

使用token替换原先的密码授权方式，在git clone私有仓库和git push文件时，需要在授权方式框中输入token（见下图）；具体获取token方式请参考**查看GitHub中Personal Access Token信息**。



15.3.5 镜像保存故障

15.3.5.1 镜像保存时报错“there are processes in 'D' status, please check process status using 'ps -aux' and kill all the 'D' status processes”或“Buildimge,False,Error response from daemon，Cannot pause container xxx”如何解决?

问题现象

- 在Notebook里保存镜像时报错“there are processes in 'D' status, please check process status using 'ps -aux' and kill all the 'D' status processes”。
- 在Notebook里保存镜像时报错“Buildimge,False,Error response from daemon: Cannot pause container xxx”。

原因分析

执行镜像保存时，Notebook中存在状态为D的进程，会导致镜像保存失败。

解决方案

1. 在Terminal里执行**ps -aux**命令检查进程。

```
(PyTorch-1.8) [ma-user work]$ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ma-user        1  0.0  0.0   4532   392 ?        Ss   10:47   0:00 /modelarts/authoring/scrip
ma-user        8  0.0  0.0  22028  2196 ?        S    10:47   0:00 /bin/bash /modelarts/autho
ma-user      103  0.0  0.2 137000 76276 ?        SN   10:47   0:02 /modelarts/authoring/noteb
ma-user      115  0.0  0.0  13444   808 ?        S    10:47   0:00 /bin/bash /modelarts/autho
ma-user      116  0.0  0.0   7940   660 ?        S    10:47   0:00 tee /home/ma-user/log/note
ma-user      119  1.5  0.3 3800480 130936 ?      Sl   10:47   0:47 /modelarts/authoring/noteb
ma-user     3134  0.0  0.0  38536 18876 pts/0    SNs  10:58   0:00 /bin/bash -l
ma-user    11045  0.0  0.0   4388   392 pts/0    DN+  11:37   0:00 ./d_process
ma-user    11046  0.0  0.0   4388   392 pts/0    SN+  11:37   0:00 ./d_process
ma-user    11069  4.2  0.0  22148  2408 pts/1    SNs  11:37   0:00 /bin/bash -l
ma-user    11128  0.0  0.0   7936   656 ?        S    11:37   0:00 sleep 3
ma-user    11131  0.0  0.0  37796  1616 pts/1    RN+  11:37   0:00 ps -aux
(PyTorch-1.8) [ma-user work]$
```

2. 执行**kill -9 <pid>**命令将相关进程结束后，再次执行镜像保存即可。

15.3.5.2 镜像保存时报错“container size %dG is greater than threshold %dG”如何解决?

问题现象

在Notebook里保存镜像时报错“container size %dG is greater than threshold %dG”。

原因分析

Notebook容器当前的大小超过了阈值。

解决方案

需要减少容器大小。Notebook容器的大小分为两部分：镜像大小和容器中新安装文件的大小。因此有两种方法来解决该问题：

- 减少容器中新安装文件的大小
  a. 删除用户在Notebook新安装的内容，比如用户在Notebook中下载了很多文件，可以将这些文件删除。这种方法仅适用于除/home/ma-user/work和/cache目录外的其他目录，因为持久化存储的部分（home/ma-user/work目录的内容）不会保存在最终产生的容器镜像中、“/cache”目录下存储的是临时文件，不占用容器空间。
  b. 如果没有文件可以删除，或者不清楚哪些可以删除，那么可以使用相同的镜像重新创建一个Notebook，使用新建的Notebook时，注意减少软件包的安装或文件的下载等操作，也可以减少容器大小；
- 减少镜像文件的大小
  如果无法确认哪些包或文件可以不安装，那么可以选择一个较小的镜像来重建Notebook，然后在其中再安装需要的软件或文件。目前公共镜像中占用空间最小的是mindspore1.7.0-py3.7-ubuntu18.04。

15.3.5.3 保存镜像时报错“too many layers in your image”如何解决?

问题现象

保存镜像时报错“too many layers in your image”。

原因分析

用户创建Notebook时所选用的镜像是经过多次保存的自定义镜像或用户自行注册的镜像，基于该镜像所创建的Notebook已经无法再执行镜像保存的操作了。

解决方法

使用公共镜像或其他的自定义镜像来创建Notebook，完成镜像保存操作。

15.3.5.4 镜像保存时报错“The container size (xG) is greater than the threshold (25G)”如何解决?

问题现象

镜像保存时报错“The container size (30G) is greater than the threshold (25G)”，镜像创建失败。

原因分析

镜像保存本质是通过在资源集群节点上的agent中进行了docker commit，再配合一系列自动化操作来上传和更新管理数据等。每次Commit都会带来额外的一些开销，层数越多镜像越大，如果多次保存后就会有存储显示没那么大，但是镜像已经很大了。镜像超大会导致加载的各种问题，所以这里做了限制。这种场景下，建议找到原始镜像重新构建环境进行保存。

解决方法

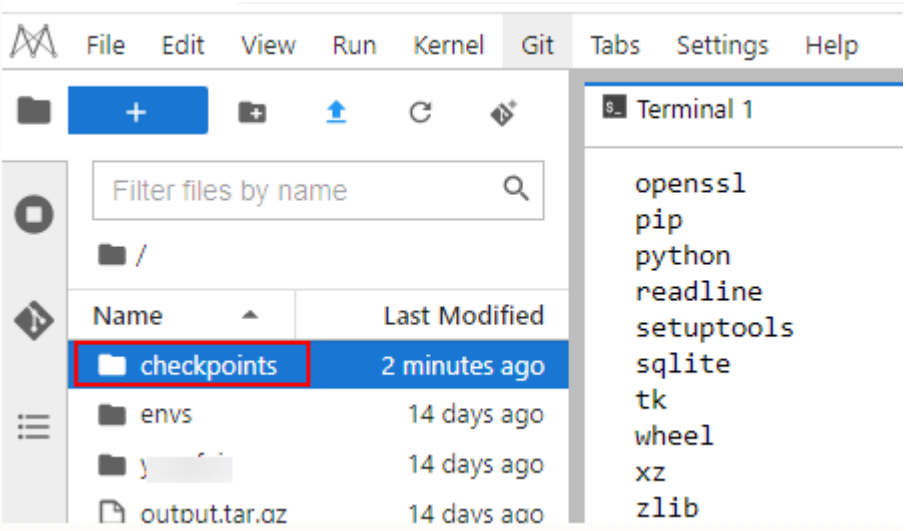
找到原始镜像重新构建环境。建议使用干净的基础镜像，最小化的安装运行依赖内容，并进行安装后的软件缓存清理，然后保存镜像。

15.3.6 其他故障

15.3.6.1 Notebook 中无法打开“checkpoints”文件夹

checkpoints是Notebook的关键字，若用户创建文件夹命名为checkpoints，则在JupyterLab上无法打开、重命名和删除。此时可以在Terminal里使用命令行打开checkpoints，或者新建文件夹将checkpoints里的数据移动到新的文件夹下。

图 15-16 JupyterLab 浏览器左侧导航无法打开 checkpoints



操作步骤：

打开Terminal，用命令行进行操作。

方法一：执行**cd checkpoints命令**打开checkpoints文件夹。

方法二：新建一个文件夹，移动checkpoints文件夹的数据到新建的文件夹下。

1. 执行**mkdir *xxx***命令，新建一个文件夹，例如“xxx”（不要用checkpoints关键字命名）
2. 然后移动checkpoints文件夹的数据到新建的文件夹下，删除根目录下checkpoints文件夹即可。

```
mv checkpoints/* xxx
rm -r checkpoints
```

15.3.6.2 创建新版 Notebook 无法使用已购买的专属资源池，如何解决？

问题现象

已购买专属资源池，但创建Notebook时该资源池不可选择，无法创建Notebook。

提示当前专属资源池未初始化开发环境，请到专属资源池页面初始化开发环境。

原因分析

新购买的专属资源池，需要初始化环境才能用于创建Notebook。

解决方法

请到专属资源池页面初始化开发环境。

步骤1 进入“专属资源池”页面，单击“操作”列的“更多 > 设置作业类型”。

步骤2 在“设置作业类型”页面，勾选“开发环境”，单击“确定”。此时“开发环境”的状态为“环境初始化中”，等到状态为“已启用”，即可使用新购买的专属资源池。

图 15-17 设置“作业类型”为“开发环境”

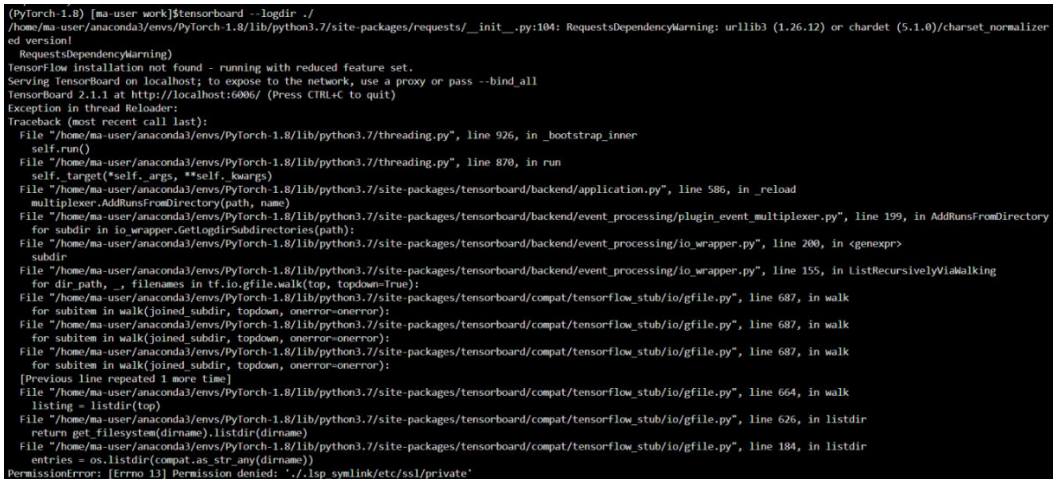
图 15-18 开发环境初始化

----结束

15.3.6.3 在 Notebook 中使用 tensorboard 命令打开日志文件报错 Permission denied

问题现象

在Notebook的Terminal中执行**tensorboard --logdir ./**命令，报错[Errno 13] Permission denied……。



原因分析

当前目录下包含没有权限的文件。

解决方法

建议用户新建一个文件夹（例如：tb_logs），将tensorboard的日志文件（例如：tb.events）放到新建的文件夹下，然后执行tensorboard命令。示例命令如下：

```
mkdir -p ./tb_logs
mv tb.events ./tb_logs
tensorboard --logdir ./tb_logs
```

15.4 训练作业

15.4.1 OBS 操作相关故障

15.4.1.1 读取文件报错，如何正确读取文件

问题现象

- 创建训练作业如何读取“json”和“npy”文件。
- 训练作业如何使用cv2库读取文件。
- 如何在MXNet环境下使用torch包。
- 训练作业读取文件，出现如下报错：
  ```
  NotFoundError (see above for traceback): Unsucessful TensorSliceReader constructor: Failed to find any matching files for xxx://xxx
  ```

原因分析

在ModelArts中，用户的数据都是存放在OBS桶中，而训练作业运行在容器中，无法通过访问本地路径的方式访问OBS桶中的文件。

处理方法

读取文件报错，您可以使用Moxing将数据拷贝至容器中，再直接访问容器中的数据。请参见步骤**1**。

您也可以根据不同的文件类型，进行读取。请参见**读取“json”文件**、**读取“npy”文件**、**使用cv2库读取文件**和**在MXNet环境下使用torch包**。

1. 读取文件报错，您可以使用Moxing将数据拷贝至容器中，再直接访问容器中的数据。具体方式如下：
   ```
   import moxing as mox
   mox.file.make_dirs('/cache/data_url')
   mox.file.copy_parallel('obs://bucket-name/data_url', '/cache/data_url')
   ```
2. **读取**“json”**文件**，请您在代码中尝试如下方法：

```
json.loads(mox.file.read(json_path, binary=True))
```

3. **使用“numpy.load”读取**“npy”**文件**，请您在代码中尝试如下方法：
   - 使用MoXing API读取OBS中的文件
     ```
     np.load(mox.file.read(_SAMPLE_PATHS['rgb'], binary=True))
     ```
   - 使用MoXing的file模块对OBS文件进行读写
     ```
     with mox.file.File(_SAMPLE_PATHS['rgb'], 'rb') as f:
     np.load(f)
     ```
4. **使用cv2库读取文件**，请您尝试如下方法：
   ```
   cv2.imdecode(np.fromstring(mox.file.read(img_path), np.uint8), 1)
   ```
5. **在MXNet环境下使用torch包**，请您尝试如下方法先进行导包：
   ```
   import os
   os.sysytem('pip install torch')
   import torch
   ```

15.4.1.2 TensorFlow-1.8 作业连接 OBS 时反复出现提示错误

问题现象

基于TensorFlow-1.8启动训练作业，并在代码中使用“tf.gfile”模块连接OBS，启动训练作业后会频繁打印如下日志信息：

```
Connection has been released. Continuing.
Found secret key
```

原因分析

这是TensorFlow-1.8中会出现的情况，该日志是Info级别的，并不是错误信息，可以通过设置环境变量来屏蔽INFO级别的日志信息。环境变量的设置一定要在import tensorflow或者import moxing之前。

处理方法

您需要通过在代码中设置环境变量“TF_CPP_MIN_LOG_LEVEL”来屏蔽INFO级别的日志信息。具体操作如下：

```
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import tensorflow as tf
import moxing.tensorflow as mox
```

“TF_CPP_MIN_LOG_LEVEL”与日志等级对应关系为：

```
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"]='1'    # 默认的显示等级，显示所有信息
os.environ["TF_CPP_MIN_LOG_LEVEL"]='2'    # 只显示warning和Error
os.environ["TF_CPP_MIN_LOG_LEVEL"]='3'    # 只显示Error
```

15.4.1.3 TensorFlow 在 OBS 写入 TensorBoard 到达 5GB 时停止

问题现象

ModelArts训练作业出现如下报错：

```
Encountered Unknown Error EntityTooLarge
Your proposed upload exceeds the maximum allowed object size.:
If the signature check failed. This could be because of a time skew. Attempting to adjust the signer
```

原因分析

OBS限制单次上传文件大小为5GB，TensorFlow保存summary可能是本地缓存，在每次触发flush时将该summary文件覆盖OBS上的原文件。当超过5GB后，由于达到了OBS单次导入文件大小的上限，导致无法继续写入。

处理方法

如果在运行训练作业的过程中出现该问题，建议处理方法如下：

1. 推荐使用本地缓存的方式来解决，使用如下方法：

```
import moxing.tensorflow as mox
mox.cache()
```

15.4.1.4 保存模型时出现 Unable to connect to endpoint 错误

问题现象

训练作业保存模型时日志报错，具体信息如下：

InternalError (see above for traceback): : Unable to connect to endpoint

原因分析

OBS连接不稳定可能会出现报错，“Unable to connect to endpoint”。

处理方法

对于OBS连接不稳定的现象，通过增加代码来解决。您可以在代码最前面增加如下代码，让TensorFlow对ckpt和summary的读取和写入可以通过本地缓存的方式中转解决：

```
import moxing.tensorflow as mox

mox.cache()
```

15.4.1.5 OBS 拷贝过程中提示“BrokenPipeError: Broken pipe”

问题现象

训练作业在使用moxing拷贝数据时出现如下报错。

图 15-19 错误日志

```
  readable=readable)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 358, in _make_put_request
  chunkedMode, methodName=methodName, readable=readable)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 390, in _make_request_with_retry
  raise e
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 369, in _make_request_with_retry
  _redirectLocation, skipAuthentication=skipAuthentication)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 436, in _make_request_internal
  conn = self._send_request(connect_server, method, path, header_config, entity, port, scheme, redirect, chunkedMode)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 586, in _send_request
  entity(util.conn_delegate(conn))
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/util.py", line 250, in entity
  conn.send(chunk)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/util.py", line 154, in send
  self.conn.send(data)
File "/home/work/anaconda/lib/python3.6/http/client.py", line 986, in send
  self.sock.sendall(data)
File "/home/work/anaconda/lib/python3.6/ssl.py", line 972, in sendall
  v = self.send(byte_view[count:])
File "/home/work/anaconda/lib/python3.6/ssl.py", line 941, in send
  return self._sslobj.write(data)
File "/home/work/anaconda/lib/python3.6/ssl.py", line 642, in write
  return self._sslobj.write(data)
BrokenPipeError: [Errno 32] Broken pipe
```

原因分析

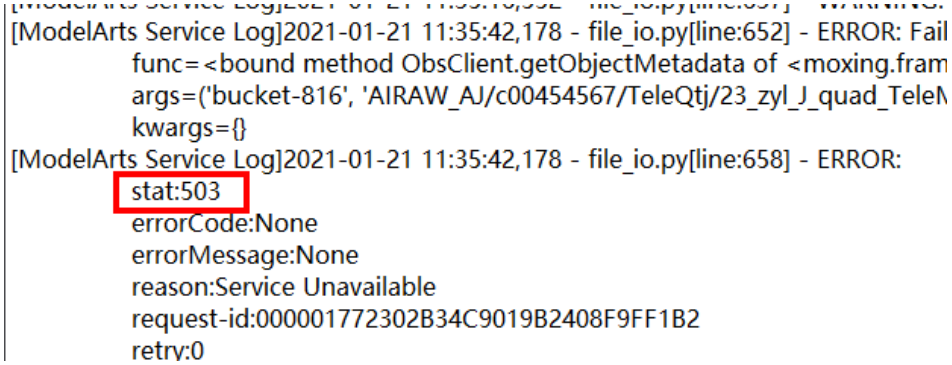
出现该问题的可能原因如下：

- 在大规模分布式作业上，每个节点都在拷贝同一个桶的文件，导致OBS桶限流。
- OBS Client连接数过多，进程/线程之间的轮询，导致一个OBS Clinet与服务端连接30S内无响应，超过超时时间，服务端断开了连接。

处理方法

1. 如果是限流问题，日志中还会有如下错误，OBS相关的错误码解释请参见《OBS服务SDK参考》的“Python>异常处理>OBS服务端错误码”，这种情况建议提工单。

   图 15-20 错误日志

2. 如果是client数太多，尤其对于5G以上文件，OBS接口不支持直接调用，需要分多个线程分段拷贝，目前OBS侧服务端超时时间是30S，可以通过如下设置减少进程数。

   ```
   # 设置进程数
   os.environ['MOX_FILE_LARGE_FILE_TASK_NUM']=1
   import moxing as mox

   # 拷贝文件
   mox.file.copy_parallel(src_url=your_src_dir, dst_url=your_target_dir, threads=0, is_processing=False)
   ```

   说明

   创建训练作业时，可通过环境变量“MOX_FILE_PARTIAL_MAXIMUM_SIZE”设置用户需要分段下载的大文件阈值（单位为Bytes），超过该阈值的文件将使用并发下载模式进行分段下载。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.1.6 日志提示“ValueError: Invalid endpoint: obs.xxxx.com”

问题现象

训练作业中使用Tensorboard直接写入到OBS路径，出现如下类似报错。

图 15-21 错误日志

```
Traceback (most recent call last):
  File "/home/work/anaconda/lib/python3.6/threading.py", line 916, in _bootstrap_inner
    self.run()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/event_file_writer.py", line 219, in run
    self._record_writer.flush()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/event_file_writer.py", line 69, in flush
    self._py_recordio_writer.flush()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/record_writer.py", line 187, in flush
    self._writer.flush()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/record_writer.py", line 89, in flush
    s3 = boto3.client('s3', endpoint_url=os.environ.get('S3_ENDPOINT'))
  File "/home/work/anaconda/lib/python3.6/site-packages/boto3/__init__.py", line 91, in client
    return _get_default_session().client(*args, **kwargs)
  File "/home/work/anaconda/lib/python3.6/site-packages/boto3/session.py", line 263, in client
    aws_session_token=aws_session_token, config=config)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/session.py", line 835, in create_client
    client_config=config, api_version=api_version)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/client.py", line 85, in create_client
    verify, credentials, scoped_config, client_config, endpoint_bridge)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/client.py", line 287, in _get_client_args
    verify, credentials, scoped_config, client_config, endpoint_bridge)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/args.py", line 107, in get_client_args
    client_cert=new_config.client_cert)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/endpoint.py", line 261, in create_endpoint
    raise ValueError("Invalid endpoint: %s" % endpoint_url)
ValueError: Invalid endpoint: obs.myhuaweicloud.com
```

原因分析

出现该问题的可能原因：

直接在OBS上写tensorboard文件，存在不稳定的风险。

处理方法

建议先将Tensorboard文件写到本地，然后再拷贝回OBS。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.1.7 日志提示“errorMessage:The specified key does not exist”

问题现象

在用moxing访问OBS路径时，出现如下错误：

```
ERROR:root:
stat:404
errorCode:NoSuchKey
errorMessage:The specified key does not exist.
```

原因分析

出现该问题的可能原因如下：

桶中的对象不存在，请检查OBS路径中的内容是否存在。具体错误码请参见《OBS服务SDK参考》的“Python>异常处理>OBS服务端错误码”。

处理方法

1. 检查OBS路径及内容格式是否正常。
2. 必现的问题，使用本地Pycharm远程连接Notebook调试。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2 云上迁移适配故障

15.4.2.1 无法导入模块

问题现象

ModelArts训练作业导入模块时日志报错：

```
Traceback (most recent call last):File "project_dir/main.py", line 1, in <module>from module_dir import module_file
ImportError: No module named module_dir
ImportError: No module named xxx
```

原因分析

- 训练作业导入模块时日志出现前两条报错信息，可能原因如下：

  代码如果在本地运行，需要将“project_dir”加入到PYTHONPATH或者将整个“project_dir”安装到“site-package”中才能运行。但是在ModelArts可以将“project_dir”加入到“sys.path”中解决该问题。

  使用**from module_dir import module_file**来导包，代码结构如下：

  ```
  project_dir
  |- main.py
  |- module_dir
  |  |- __init__.py
  |  |- module_file.py
  ```

- 训练作业导入模块时日志出现“ImportError: No module named xxx”的报错，可以判断是环境中没有包含用户依赖的python包。

处理方法

- 训练作业导入模块时日志出现前两条报错信息，处理方法如下：

  a. 首先保证被导入的module中有“__init__.py”存在，创建“module_dir”的“__init__.py”，如原因分析中的结构所示。

  b. 由于无法知晓“project_dir”在容器中的位置，所以利用绝对路径，在“main.py”中将“project_dir”添加到“sys.path”中，再导入：

  ```
  import os
  import sys
  # __file__为获取当前执行脚本main.py的绝对路径
  # os.path.dirname(__file__)获取main.py的父目录，即project_dir的绝对路径
  current_path = os.path.dirname(__file__)
  sys.path.append(current_path)
  # 在sys.path.append执行完毕之后再导入其他模块
  from module_dir import module_file
  ```

- 训练作业导入模块时日志出现“ImportError: No module named xxx”的报错，请添加如下代码安装依赖包：

```
import os
os.system('pip install xxx')
```

15.4.2.2 训练作业日志中提示“No module named .*”

用户请按照以下思路进行逐步排查：

1. **检查依赖包是否存在**
2. **检查依赖包路径是否能被识别**
3. **检查训练作业使用的资源规格是否正确**
4. **建议与总结**

检查依赖包是否存在

如果依赖包不存在，您可以使用以下两种方式完成依赖包的安装。

- 方式一（推荐使用）：在创建我的算法时，需要在“代码目录”下放置相应的文件或安装包。

  请根据依赖包的类型，在代码目录下放置对应文件：

  - **依赖包为开源安装包时**

    在“代码目录”中创建一个命名为“pip-requirements.txt”的文件，并且在文件中写明依赖包的包名及其版本号，格式为“包名==版本号”。

    例如，“代码目录”对应的OBS路径下，包含模型文件，同时还存在“pip-requirements.txt”文件。“代码目录”的结构如下所示：

    ```
    |---模型启动文件所在OBS文件夹
        |---model.py              #模型启动文件。
        |---pip-requirements.txt   #定义的配置文件，用于指定依赖包的包名及版本号。
    ```

    “pip-requirements.txt”文件内容如下所示：

    ```
    alembic==0.8.6
    bleach==1.4.3
    click==6.6
    ```

  - **依赖包为whl包时**

    如果训练后台不支持下载开源安装包或者使用用户编译的whl包时，由于系统无法自动下载并安装，因此需要在“代码目录”放置此whl包，同时创建一个命名为“pip-requirements.txt”的文件，并且在文件中指定此whl包的包名。依赖包必须为“.whl”格式的文件。

    例如，“代码目录”对应的OBS路径下，包含模型文件、whl包，同时还存在“pip-requirements.txt”文件。“代码目录”的结构如下所示：

    ```
    |---模型启动文件所在OBS文件夹
        |---model.py              #模型启动文件。
        |---XXX.whl               #依赖包。依赖多个时，此处放置多个。
        |---pip-requirements.txt   #定义的配置文件，用于指定依赖包的包名。
    ```

    “pip-requirements.txt”文件内容如下所示：

    ```
    numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl
    tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
    ```

- 方式二：可以在启动文件添加如下代码安装依赖包：

  ```
  import os
  os.system('pip install xxx')
  ```

方式一在训练作业启动前即可完成相关依赖包的下载与安装，而方式二是运行启动文件过程中进行依赖包的下载与安装。

检查依赖包路径是否能被识别

代码如果在本地运行，需要将“project_dir”加入到PYTHONPATH或者将整个“project_dir”安装到“site-package”中才能运行。但是在ModelArts可以将“project_dir”加入到“sys.path”中解决该问题。

使用**from module_dir import module_file**来导包，代码结构如下：

```
project_dir
|- main.py
|- module_dir
|  |- __init__.py
|  |- module_file.py
```

检查训练作业使用的资源规格是否正确

训练作业报错No module named npu_bridge.npu_init

```
from npu_bridge.npu_init import *
ImportError: No module named npu_bridge.npu_init
```

检查下训练作业使用的规格是否支持NPU，有可能是训练时使用了GPU规格，导致发生了NPU相关调用报错。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.3 如何安装第三方包，安装报错的处理方法

问题现象

- ModelArts**如何安装自定义库函数**，例如“apex”。
- ModelArts训练环境安装第三方包时出现如下报错：
  ```
  xxx.whl is not a supported wheel on this platform
  ```

原因分析

由于安装的文件名格式不支持，导致出现“xxx.whl is not a supported wheel on this platform”报错，具体解决方法请参见**2**。

处理方法

1. **安装第三方包**
   a. pip中存在的包，使用如下代码：
      ```
      import os
      os.system('pip install xxx')
      ```
   b. pip源中不存在的包，此处以“apex”为例，请您用如下方式将安装包上传到OBS桶中。 该样例已将安装包上传至“obs://cnnorth4-test/codes/mox_benchmarks/apex-master/”中，将在启动文件中添加以下代码进行安装。
      ```
      try:
          import apex
      except Exception:
          import os
          import moxing as mox
      ```

```
    mox.file.copy_parallel('obs://cnnorth4-test/codes/mox_benchmarks/apex-master/', '/cache/apex-master')
    os.system('pip --default-timeout=100 install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" /cache/apex-master')
```

2. **安装报错**

   “xxx.whl”文件无法安装，需要您按照如下步骤排查：

   a. 当出现“xxx.whl”文件无法安装，在启动文件中添加如下代码，查看当前pip命令支持的文件名和版本。

   ```
   import pip
   print(pip.pep425tags.get_supported())
   ```

   获取到支持的文件名和版本如下：

   ```
   [('cp36', 'cp36m', 'manylinux1_x86_64'), ('cp36', 'cp36m', 'linux_x86_64'), ('cp36', 'abi3', 'manylinux1_x86_64'), ('cp36', 'abi3', 'linux_x86_64'), ('cp36', 'none', 'manylinux1_x86_64'), ('cp36', 'none', 'linux_x86_64'), ('cp35', 'abi3', 'manylinux1_x86_64'), ('cp35', 'abi3', 'linux_x86_64'), ('cp34', 'abi3', 'manylinux1_x86_64'), ('cp34', 'abi3', 'linux_x86_64'), ('cp33', 'abi3', 'manylinux1_x86_64'), ('cp33', 'abi3', 'linux_x86_64'), ('cp32', 'abi3', 'manylinux1_x86_64'), ('cp32', 'abi3', 'linux_x86_64'), ('py3', 'none', 'manylinux1_x86_64'), ('py3', 'none', 'linux_x86_64'), ('cp36', 'none', 'any'), ('cp3', 'none', 'any'), ('py36', 'none', 'any'), ('py3', 'none', 'any'), ('py35', 'none', 'any'), ('py34', 'none', 'any'), ('py33', 'none', 'any'), ('py32', 'none', 'any'), ('py31', 'none', 'any'), ('py30', 'none', 'any')]
   ```

   b. 将“faiss_gpu-1.5.3-cp36-cp36m-manylinux2010_x86_64.whl”更改为“faiss_gpu-1.5.3-cp36-cp36m-manylinux1_x86_64.whl”，并安装，执行命令如下：

   ```
   import moxing as mox
   import os

   mox.file.copy('obs://wolfros-net/zp/AI/code/faiss_gpu-1.5.3-cp36-cp36m-manylinux2010_x86_64.whl','/cache/faiss_gpu-1.5.3-cp36-cp36m-manylinux1_x86_64.whl')
   os.system('pip install /cache/faiss_gpu-1.5.3-cp36-cp36m-manylinux1_x86_64.whl')
   ```

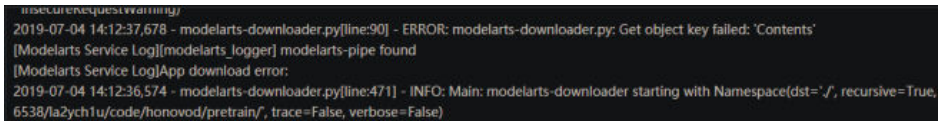
15.4.2.4 下载代码目录失败

问题现象

训练作业运行时下载失败，出现如下报错，请参见图15-22：

```
ERROR：modelarts-downloader.py: Get object key failed: 'Contents'
```

图 15-22 获取内容失败



原因分析

在创建训练作业时指定的代码目录不存在导致训练失败。

处理方法

请您根据报错原因排查创建训练作业时指定的代码目录，即OBS桶的路径是否正确。有两种方法判断是否存在。

- 使用当前账户登录OBS管理控制台，去查找对应的OBS桶、文件夹、文件是否存在。

- 通过接口判断路径是否存在。在代码中执行如下命令，检查路径是否存在。
  ```
  import moxing as mox
  mox.file.exists('obs://obs-test/ModelArts/examples/')
  ```

15.4.2.5 训练作业日志中提示“No such file or directory”

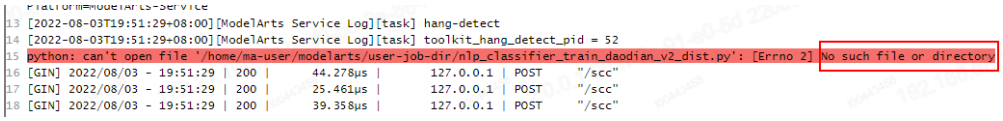
问题现象

训练作业运行失败，日志中提示“No such file or directory”。

例如：找不到训练输入的数据路径时，会提示“No such file or directory”。

例如：找不到训练启动文件时，也会提示“No such file or directory”。

图 15-23 日志提示找不到启动文件（示例）



原因分析

- 找不到训练输入数据路径，可能是报错的路径填写不正确。用户请按照以下思路进行逐步排查：
  a. **检查报错的路径是否为OBS路径**
  b. **检查报错的路径是否存在**
- 找不到启动文件，可能是训练作业启动命令的路径填写不正确，参考**使用自定义镜像创建训练作业时，检查启动文件路径**排查解决。
- 可能为多个进程或者worker读写同一个文件。如果使用了SFS，则考虑是否多个节点同时写同一个文件。分析代码中是否存在多进程写同一文件的情况。建议避免作业中存在多进程，多节点并发读写同一文件的情况。

检查报错的路径是否为 OBS 路径

使用ModelArts时，用户数据需要存放在自己OBS桶中，但是训练代码运行过程中不能使用OBS路径读取数据。

原因：

训练作业创建成功后，由于在运行容器直连OBS服务进行训练性能很差，系统会自动下载训练数据至运行容器的本地路径。所以，在训练代码中直接使用OBS路径会报错。例如训练代码的OBS路径为obs://bucket-A/training/，训练代码会被自动下载至${MA_JOB_DIR}/training/。

假设训练代码的OBS目录为obs://bucket-A/XXX/{training-project}/，“{training-project}”是存放训练代码的文件夹名称。训练时会自动下载OBS中{training-project}目录下的数据到训练容器的本地路径$MA_JOB_DIR/{training-project}/。

如果报错路径为训练数据路径，需要在以下两个地方完成适配，具体适配方法请参考自定义算法适配章节的输入输出配置部分：

1. 在创建算法时，您需要在输入路径配置中设置代码路径参数，默认为“data_url”。

2. 您需要在训练代码中添加超参，默认为“data_url”。使用“data_url”当做训练数据输入的本地路径。

检查报错的路径是否存在

由于用户本地开发的代码需要上传至ModelArts后台，训练代码中涉及到依赖文件的路径时，用户设置有误的场景较多。

推荐通用的解决方案：使用os接口得到依赖文件的绝对路径，避免报错。

示例：

```
|---project_root                #代码根目录
   |---BootfileDirectory        #启动文件所在的目录
     |---bootfile.py           #启动文件
   |---otherfileDirectory       #其他依赖文件所在的目录
     |---otherfile.py          #其他依赖文件
```

在启动文件中，建议用户参考以下方式获取依赖文件所在路径，即示例中的otherfile_path。

```
import os
current_path = os.path.dirname(os.path.realpath(__file__)) # BootfileDirectory, 启动文件所在的目录
project_root = os.path.dirname(current_path) # 工程的根目录，对应ModelArts训练控制台上设置的代码目录
otherfile_path = os.path.join(project_root, "otherfileDirectory", "otherfile.py")
```

使用自定义镜像创建训练作业时，检查启动文件路径

以OBS路径“obs://obs-bucket/training-test/demo-code”为例，训练代码会被自动下载至训练容器的“${MA_JOB_DIR}/demo-code”目录中，demo-code为OBS存放代码路径的最后一级目录，可以根据实际修改。

使用自定义镜像创建训练作业时，在代码目录下载完成后，镜像的启动命令会被自动执行。启动命令的填写规范如下：

- 如果训练启动脚本用的是py文件，例如train.py，运行命令可以写为python ${MA_JOB_DIR}/demo-code/train.py。
- 如果训练启动脚本用的是sh文件，例如main.sh，运行命令可以写为bash ${MA_JOB_DIR}/demo-code/main.sh。

其中demo-code为OBS存放代码路径的最后一级目录，可以根据实际修改。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.6 训练过程中无法找到 so 文件

问题现象

ModelArts训练作业运行时，日志中遇到如下报错，导致训练失败：

```
libcudart.so.9.0 cannot open shared object file no such file or directory
```

原因分析

编译生成so文件的cuda版本与训练作业的cuda版本不一致。

处理方法

编译环境的cuda版本与训练环境不一致，训练作业运行就会报错。例如：使用cuda版本为10的开发环境tf-1.13中编译生成的so包，在cuda版本为9.0训练环境中tf-1.12训练会报该错。

编译环境和训练环境的cuda版本不一致时，可参考如下处理方法：

1. 在业务执行前加如下命令，检查是否能找到so文件。如果已经找到so文件，执行2；如果没有找到，执行3。
   ```
   import os;
   os.system(find /usr -name *libcudart.so*);
   ```
2. 设置环境变量LD_LIBRARY_PATH，设置完成后，重新下发作业即可。
   例如so文件的存放路径为：/use/local/cuda/lib64，LD_LIBRARY_PATH设置如下：
   ```
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
   ```
3. 执行如下命令，查看训练环境的cuda版本，确认当前cuda版本是否支持so文件。
   ```
   os.system("cat /usr/local/cuda/version.txt")
   ```
   a. 支持。当前cuda版本无so文件，需外部导入so文件（自行在浏览器下载），再设置LD_LIBRARY_PATH，具体见2。
   b. 不支持。尝试更换引擎，重新下发作业。或者使用自定义镜像创建作业，可参考**使用自定义镜像创建作业**。

15.4.2.7 ModelArts 训练作业无法解析参数，日志报错

问题现象

ModelArts训练作业无法解析参数，遇到如下报错，导致无法正常运行：

```
error: unrecognized arguments: --data_url=xxx://xxx/xxx
error: unrecognized arguments: --init_method=tcp://job
absl.flags._exceptions.UnrecognizedFlagError:Unknown command line flag 'task_index'
```

原因分析

- 运行参数中未定义该参数。
- 在训练环境中，系统可能会传入在Python脚本里没有定义的其他参数名称，导致参数无法解析，日志报错。

处理方法

1. 参数定义中增加该参数的定义，代码示例如下：
   ```
   parser.add_argument('--init_method', default='tcp://xxx',help="init-method")
   ```
2. 通过使用解析方式**args, unparsed = parser.parse_known_args()**代替**args = parser.parse_args()**解决该问题。代码示例如下：
   ```
   import argparse
   parser = argparse.ArgumentParser()
   parser.add_argument('--data_url', type=str, default=None, help='obs path of dataset')
   args, unparsed = parser.parse_known_args()
   ```

15.4.2.8 训练输出路径被其他作业使用

问题现象

在创建训练作业时出现如下报错：操作失败！Other running job contain train_url: /bucket-20181114/code_hxm/

原因分析

根据报错信息判断，在创建训练作业时，同一个“训练输出路径”在被其他作业使用。

处理方法

一个“训练输出路径”只能被一个处于“运行中”、“排队中”或“初始化”状态的作业使用。

当出现此报错时，建议检查并重新填写训练作业的“训练输出路径”，以避免创建作业失败。

15.4.2.9 使用自定义镜像创建训练作业，找不到启动文件

问题现象

使用旧版训练的自定义镜像创建训练作业，出现如下报错，提示找不到运行的主文件：no such file or directory。

原因分析

根据报错提示可以判断是运行命令的启动文件目录不正确导致运行失败。

处理方法

需要排查执行命令的启动文件目录是否正确，具体操作如下：

在ModelArts管理控制台，使用训练的自定义镜像创建训练作业时，“算法来源”选择“自定义”页签。

若训练代码启动脚本在OBS路径为“obs://bucket-name/app/code/train.py”，创建作业时配置代码目录为“/bucket-name/app/code/”。

代码目录配置完成后，执行如下命令，那么“run_train.sh”将选中的“code”文件夹下载到旧版训练容器的“/home/work/user-job-dir”目录中。

```
bash /home/work/run_train.sh  #旧版训练命令，run_train.sh训练启动引导脚本，打包在ModelArts提供的基础镜像中。
```

运行命令就可以设置为：

```
bash /home/work/run_train.sh python /home/work/user-job-dir/code/train.py {python_file_parameter}  #旧版训练
```

15.4.2.10 Pytorch1.0 引擎提示“RuntimeError: std::exception”

问题现象

在使用pytorch1.0镜像时，必现如下报错：

```
“RuntimeError: std::exception”
```

原因分析

出现该问题的可能原因如下：

pytorch1.0镜像中的libmkldnn软连接与原生torch的冲突，具体可参看文档。

处理方法

1. 按照issues中的说明，应该是环境中的库冲突了，因此在启动脚本最开始之前，添加如下代码。

```
import os
os.system("rm /home/work/anaconda3/lib/libmkldnn.so")
os.system("rm /home/work/anaconda3/lib/libmkldnn.so.0")
```

2. 必现的问题，使用本地Pycharm远程连接Notebook调试。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.11 MindSpore 日志提示“ retCode=0x91, [the model stream execute failed]”

问题现象

使用mindspore进行训练时，出现如下报错：

```
[ERROR] RUNTIME(3002)model execute error, retCode=0x91, [the model stream execute failed]
```

原因分析

出现该问题的可能原因如下：

数据读入的速度跟不上模型迭代的速度。

处理方法

1. 减少预处理shuffle操作。

```
dataset = dataset.shuffle(buffer_size=x)
```

2. 关闭数据预处理开关，可能会影响性能。

```
NPURunConfig(enable_data_pre_proc=Flase)
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.12 使用 moxing 适配 OBS 路径，pandas 读取文件报错

问题现象

使用moxing适配OBS路径，然后用较高版本的pandas读取OBS文件报出如下错误：

```
1.‘can't decode byte xxx in position xxx’
2.‘OSError:File isn't open for writing’
```

原因分析

出现该问题的可能原因如下：

moxing对高版本的pandas兼容性不够。

处理方法

1. 在适配OBS路径后，读取文件模式从‘r’改成‘rb’，然后将mox.file.File的'_write_check_passed'属性值改为‘True’，参考如下代码。

```
import pandas as pd
import moxing as mox

mox.file.shift('os', 'mox')  # 将os的open操作替换未mox.file.File适配OBS路径的操作

param = {'encoding': 'utf-8'}
path = 'xxx.csv'
with open(path, 'rb') as f:
    f._wirte_check_passed = True
    df = pd.read_csv(ff, **param)
```

2. 必现的问题，使用本地Pycharm远程连接Notebook调试。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.13 日志提示“Please upgrade numpy to >= xxx to use this pandas version”

问题现象

在安装其他包的时候，有依赖冲突，对numpy库有其他要求，但是发现numpy卸载不了。出现如下类似错误：

```
your numpy version is 1.14.5.Please upgrade numpy to >= 1.15.4 to use this pandas version
```

原因分析

出现该问题的可能原因如下：

conda和pip包混装，有一些包卸载不掉。

处理方法

参考如下代码，三步走。

1. 先卸载numpy中可以卸载的组件。

2. 删除你环境中site-packages路径下的numpy文件夹。
3. 重新进行安装需要的版本。

```
import os
os.system("pip uninstall -y numpy")
os.system('rm -rf /home/work/anaconda/lib/python3.6/site-packages/numpy/')
os.system("pip install numpy==1.15.4")
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.14 重装的包与镜像装 CUDA 版本不匹配

问题现象

在现有镜像基础上，重新装了引擎版本，或者编译了新的CUDA包，出现如下错误：

```
1.“RuntimeError: cuda runtime error (11) : invalid argument at /pytorch/aten/src/THC/
THCCachingHostAllocator.cpp:278”
2.“libcudart.so.9.0 cannot open shared object file no such file or directory”
3.“Make sure the device specification refers to a valid device, The requested device appeares to be a
GPU,but CUDA is not enabled”
```

原因分析

出现该问题的可能原因如下：

新安装的包与镜像中带的CUDA版本不匹配。

处理方法

必现的问题，使用本地Pycharm远程连接Notebook调试安装。

1. 先远程登录到所选的镜像，使用“nvcc -V”查看目前镜像自带的CUDA版本。
2. 重装torch等，需要注意选择与上一步版本相匹配的版本。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.15 创建训练作业提示错误码 ModelArts.2763

问题现象

创建训练作业时，提示ModelArts.2763 : 选择的支持实例无效，请检查请求中信息的合法性。

原因分析

用户选择的训练规格资源和算法不匹配。

例如：算法支持的是GPU规格，创建训练作业时选择了ASCEND规格的资源类型。

处理方法

1. 查看算法代码中设置的训练资源规格。
2. 检查创建训练作业时所选的资源规格是否正确，重新创建训练作业选择正确的资源规格。

15.4.2.16 训练作业日志中提示 “AttributeError: module '***' has no attribute '***'”

问题现象

训练日志中出现AttributeError: module '***' has no attribute '***'错误。如：AttributeError: module 'torch' has no attribute 'concat'。

原因分析

出现该问题的可能原因如下：

- 对应python包使用错误，该python包确实没有对应的变量或者方法
- 第三方pip源中的python包版本更新，导致在训练作业中安装的python包的版本可能也会发生变化。如训练作业之前无此问题，后面一直有此问题，则考虑是此原因。

处理方法

- 通过Notebook调试。
- 安装时指定版本。如：pip install xxx==1.x.x
- 第三方pip源可能随时更新，可通过制作自定义镜像，来避免该影响。可参见文档训练管理中使用自定义镜像介绍。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.2.17 系统容器异常退出

问题现象

在训练创建后出现“系统容器异常退出”的故障。

图 15-24 错误日志

```
[ModelArts Service Log]2022-10-11 19:17:35,178 - file_io.py[line:728] - WARNING: Retry=4, Wait=3.2, Timestamp=1665487055.178172, Function=getObject, args=
('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:17:38,405 - file_io.py[line:728] - WARNING: Retry=3, Wait=6.4, Timestamp=1665487058.4054542, Function=getObject, args=
('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:17:44,832 - file_io.py[line:728] - WARNING: Retry=2, Wait=12.8, Timestamp=1665487064.8322, Function=getObject, args=
('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:17:57,663 - file_io.py[line:728] - WARNING: Retry=1, Wait=25.6, Timestamp=1665487077.6639552, Function=getObject,
args=('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:18:23,266 - file_io.py[line:741] - ERROR: Failed to call:
    func=<bound method ObsClient.getObject of <moxing.framework.file.src.obs.client.ObsClient object at 0x7fb332a2a910>>
    args=('modelarts-cn-north-4-test', 'modelarts/code-test')
    kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:18:23,267 - file_io.py[line:748] - ERROR:
    stat:404
    errorCode:NoSuchKey
    errorMessage:The specified key does not exist.
    reason:Not Found
    request-id:00000183C6C4010C66D399E000C0E366
    retry:0
[ModelArts Service Log]2022-10-11 19:18:23,267 - modelarts-downloader.py[line:90] - ERROR: modelarts-downloader.py: Download directory failed: [Errno
{'status': 404, 'reason': 'Not Found', 'errorCode': 'NoSuchKey', 'errorMessage': 'The specified key does not exist.', 'body': None, 'requestId':
'00000183C6C4010C66D399E000C0E366', 'hostId': 'tMe7zkB1evwTBxJLL7XbKfb68lgFdhz/M+xo5tgabLcrgw90DRbnKiHVYoJmYHPT', 'header': [('x-reserved', 'amazon, aws
and amazon web services are trademarks or registered trademarks of Amazon Technologies, Inc'), ('request-id', '00000183C6C4010C66D399E000C0E366'), ('id-2',
'32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSdfbsDDEx4QVqrcVsoqSC/Io8A16gNg'), ('content-type', 'application/xml'), ('date', 'Tue, 11 Oct 2022 11:17:57 GMT'),
('content-length', '310')]}] file or directory or bucket not found.
```

原因分析

出现该问题的可能原因如下：

1. OBS相关错误。
   a. OBS文件不存在。The specified key does not exist。
   b. 用户OBS权限不足。
   c. OBS限流。
   d. OBS其他问题。
2. 磁盘空间不足。

处理方法

1. 如果是OBS相关错误。
   a. OBS文件不存在。The specified key does not exist。
      参考**日志提示“errorMessage:The specified key does not exist”**章节处理。
   b. 用户OBS权限不足。
      参考 **5.5.1 日志提示“reason:Forbidden”**。
   c. OBS限流。
      参考**5.1.1 OBS拷贝过程中提示“BrokenPipeError: Broken pipe”**。
   d. OBS其他问题。
      请参考或者采集request id后向OBS客服进行咨询。
2. 如果是空间不足。
   参考 **常见的磁盘空间不足的问题和解决办法**章节处理。

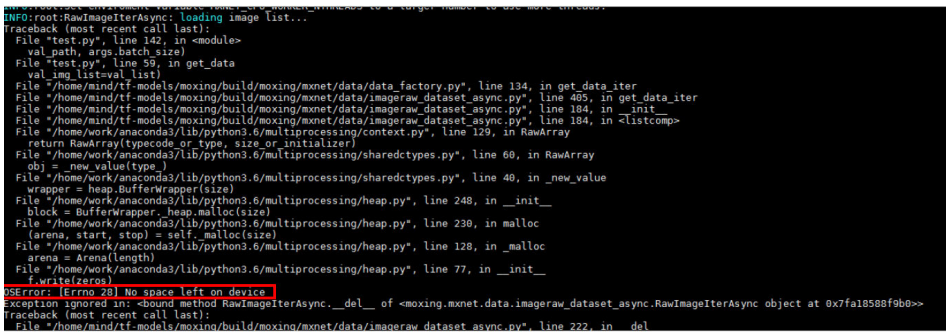
15.4.3 硬盘限制故障

15.4.3.1 下载或读取文件报错，提示超时、无剩余空间

问题现象

训练过程中拷贝数据/代码/模型时出现如下报错：

图 15-25 错误日志



原因分析

出现该问题的可能原因如下。

- 磁盘空间不足。
- 分布式作业时，有些节点的docker base size配置未生效，容器内“/”根目录空间未达到50GB，只有默认的10GB，导致作业训练失败。
- 实际存储空间足够，却依旧报错“No Space left on device”。

  同一目录下创建较多文件，为了加快文件检索速度，内核会创建一个索引表，短时间内创建较多文件时，会导致索引表达到上限，进而报错。

  说明

  触发条件和下面的因素有关：

  - 文件名越长，文件数量的上限越小
  - blocksize越小，文件数量的上限越小。（ blocksize，系统默认 4096B。总共有三种大小：1024B、2048B、4096B ）
  - 创建文件越快，越容易触发（机制大概是：有一个缓存，这块大小和上面的1和2有关，目录下文件数量比较大时会启动，使用方式是边用边释放 ）

处理方法

1. 可以参照**日志提示"write line error"**文档进行修复。
2. 如果是分布式作业有的节点有错误，有的节点正常，建议提工单请求隔离有问题的节点。
3. 如果是触发了欧拉操作系统的限制，有如下建议措施。
   - 分目录处理，减少单个目录文件量。
   - 减慢创建文件的速度。
   - 关闭ext4文件系统的dir_index属性，具体可参考：https://access.redhat.com/solutions/29894，（ 可能会影响文件检索性能 ）。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.3.2 拷贝数据至容器中空间不足

问题现象

ModelArts训练作业运行时，日志中遇到如下报错，导致数据无法复制至容器中。

```
OSError:[Errno 28] No space left on device
```

原因分析

数据下载至容器的位置空间不足。

处理方法

1. 请排查是否将数据下载至“/cache”目录下，GPU规格资源的每个节点会有一个“/cache”目录，空间大小为4TB。并确认该目录下并发创建的文件数量是否过大，占用过多存储空间会出现inode耗尽的情况，导致空间不足。
2. 请排查是否使用的是GPU资源。如果使用的是CPU规格的资源，“/cache”与代码目录共用10G，会造成内存不足，请更改为使用GPU资源。
3. 请在代码中添加环境变量来解决。

```
import os
os.system('export TMPDIR=/cache')
```

15.4.3.3 Tensorflow 多节点作业下载数据到/cache 显示 No space left

问题现象

创建训练作业，Tensorflow多节点作业下载数据到/cache显示：“No space left”。

原因分析

TensorFlow多节点任务会启动parameter server（简称ps）和worker两种角色，ps和worker会被调度到相同的机器上。由于训练数据对于ps没有用，因此在代码中ps相关的逻辑不需要下载训练数据。如果ps也下载数据到“/cache”，实际下载的数据会翻倍。例如只下载了2.5TB的数据，程序就显示空间不够而失败，因为/cache只有4TB的可用空间。

处理方法

在使用Tensorflow多节点作业下载数据时，正确的下载逻辑如下：

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--job_name", type=str, default="")
args = parser.parse_known_args()

if args[0].job_name != "ps":
    copy.....................
```

15.4.3.4 日志文件的大小达到限制

问题现象

ModelArts训练作业在运行过程中报错，提示日志文件的大小已达到限制：

```
modelarts-pope: log length overflow(max:1073741824; already: 107341771; new:90), process will continue running silently
```

原因分析

根据报错信息，可以判断是日志文件的大小已达到限制。出现该报错之后，日志不再增加，后台将继续运行。

处理方法

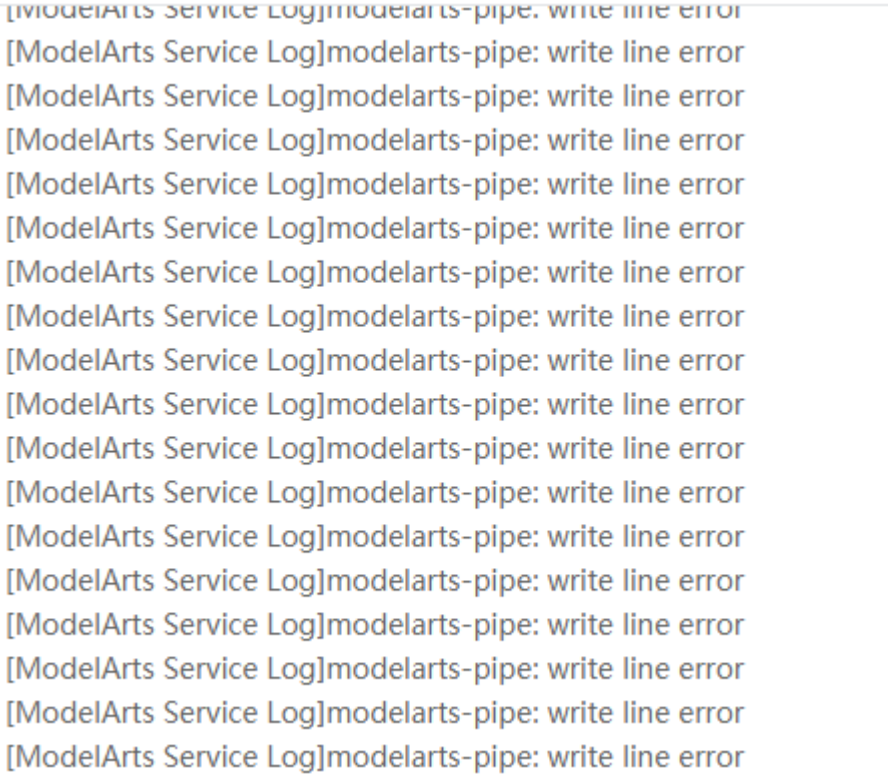
请您在启动文件中减少无用日志输出。

15.4.3.5 日志提示"write line error"

问题现象

在程序运行过程中，刷出大量错误日志"write line error"。并且问题是必现问题，每次运行到同一地方的时候，出现错误，具体见下面截图：

图 15-26 错误日志



原因分析

出现该问题的可能原因如下：

- 程序运行过程中，产生了core文件，core文件占满了"/"根目录空间。
- 本地数据、文件保存将"/cache"目录3.5T空间用完了。

说明

云上训练磁盘空间一般指如下两个目录的磁盘空间：

1. “/”根目录，是docker中配置项“base size”，默认是10G，云上统一改为50G。
2. “/cache”目录满了，一般是3.5T存储空间满了。

处理方法

1. 如果在训练作业的工作目录下有core文件生成，可以在启动脚本最前面加上如下代码，来关闭core文件产生。

```
import os
os.system("ulimit -c 0")
```

2. 排查数据集大小，checkpoint保存文件大小，是否占满了磁盘空间。
3. 必现的问题，使用本地Pycharm远程连接Notebook调试。

建议与总结

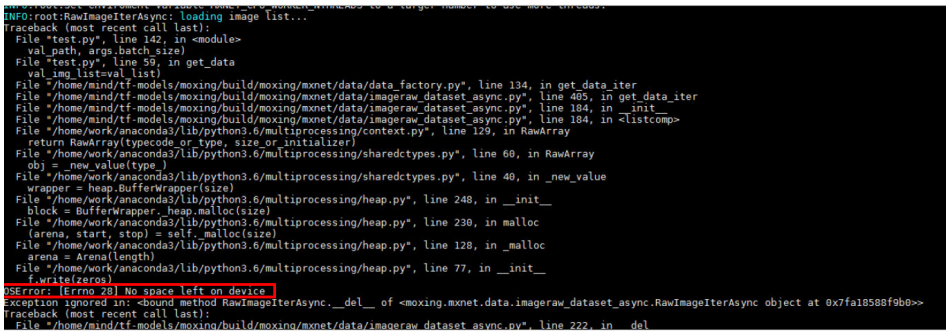
在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.3.6 日志提示“No space left on device”

问题现象

训练过程中拷贝数据/代码/模型时出现如下报错：

图 15-27 错误日志



原因分析

出现该问题的可能原因如下。

- 磁盘空间不足。
- 分布式作业时，有些节点的docker base size配置未生效，容器内“/”根目录空间未达到50G，只有默认的10GB，导致作业训练失败。
- 实际存储空间足够，却依旧报错“No Space left on device”。

  同一目录下创建较多文件，为了加快文件检索速度，内核会创建一个索引表，短时间内创建较多文件时，会导致索引表达到上限，进而报错。

  说明

  触发条件和下面的因素有关：

  - 文件名越长，文件数量的上限越小
  - blocksize越小，文件数量的上限越小。（ blocksize，系统默认 4096B。总共有三种大小：1024B、2048B、4096B ）
  - 创建文件越快，越容易触发（机制大概是：有一个缓存，这块大小和上面的1和2有关，目录下文件数量比较大时会启动，使用方式是边用边释放 ）

处理方法

1. 可以参照**日志提示"write line error"**文档进行修复。
2. 如果是分布式作业有的节点有错误，有的节点正常，建议提工单请求隔离有问题的节点。

3. 如果是触发了欧拉操作系统的限制，有如下建议措施。
   - 分目录处理，减少单个目录文件量。
   - 减慢创建文件的速度。
   - 关闭ext4文件系统的dir_index属性，具体可参考：https://access.redhat.com/solutions/29894，（可能会影响文件检索性能）。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.3.7 OOM 导致训练作业失败

问题现象

因为OOM导致的训练作业失败，会有如下几种现象：

1. 错误码返回137，如下图所示：
2. 日志中有报错，含有“killed”相关字段，例如如下截图：

图 15-28 错误日志信息

```
Traceback (most recent call last):
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 261, in <module>
    main()
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 251, in main
    loss,acc = train_and_test(e, opt.alpha_start)
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 243, in train_and_test
    acc = test(epoch, alpha_start, False)
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 222, in test
    output = net(images, epoch, alpha_start)
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/modules/module.py", line 541, in __call__
    result = self.forward(*input, **kwargs)
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/parallel/data_parallel.py", line 152, in forward
    outputs = self.parallel_apply(replicas, inputs, kwargs)
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/parallel/data_parallel.py", line 162, in parallel_apply
    return parallel_apply(replicas, inputs, kwargs, self.device_ids[:len(replicas)])
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/parallel/parallel_apply.py", line 75, in parallel_apply
    thread.start()
  File "/home/ma-user/anaconda/lib/python3.6/threading.py", line 851, in start
    self._started.wait()
  File "/home/ma-user/anaconda/lib/python3.6/threading.py", line 551, in wait
    signaled = self._cond.wait(timeout)
  File "/home/ma-user/anaconda/lib/python3.6/threading.py", line 295, in wait
    waiter.acquire()
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/utils/data/_utils/signal_handling.py", line 66, in handler
    error_if_any_worker_fails()
RuntimeError: DataLoader worker (pid 38077) is killed by signal: Killed.
```

3. 日志中有报错“RuntimeError: CUDA out of memory. ”，如下图所示：

图 15-29 错误日志信息

```
Traceback (most recent call last):
  File "memory_test.py", line 47, in <module>
    tmp_tensor = torch.empty(int(total_memory * 0.45), dtype=torch.int8, device='cuda')
RuntimeError: CUDA out of memory. Tried to allocate 14.29 GiB (GPU 0; 14.29 GiB total capacity; 0 bytes
already allocated; 14.29 GiB free; 0 bytes reserved in total by PyTorch)
```

4. Tensorflow引擎日志中出现“Dst tensor is not initialized”。

原因分析

按照之前支撑的经验，出现该问题的可能原因如下：

- 绝大部分都是确实是显存不够用。
- 还有较少数原因是节点故障，跑到特定节点必现OOM，其他节点正常。

处理方法

1. 如果是正常的OOM，就需要修改一些超参，释放一些不需要的tensor。
   a. 修改网络参数，比如batch_size、hide_layer、cell_nums等。
   b. 释放一些不需要的tensor，使用过的，如下：
   ```
   del tmp_tensor
   torch.cuda.empty_cache()
   ```
2. 必现的问题，使用本地Pycharm远程连接Notebook调试超参。
3. 如果还存在问题，可能需要提工单进行定位，甚至需要隔离节点修复。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

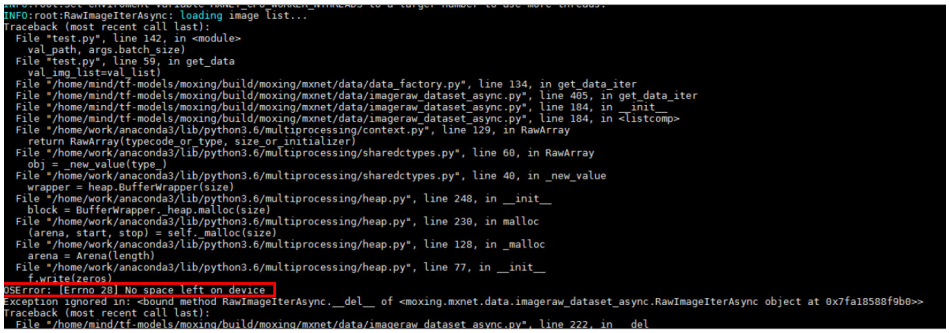
15.4.3.8 常见的磁盘空间不足的问题和解决办法

该章节用于统一整体所有的常见的磁盘空间不足的问题和解决办法。减少相关问题文档的重复内容。

问题现象

训练过程中拷贝数据/代码/模型时出现如下报错：

图 15-30 错误日志



原因分析

出现该问题的可能原因如下：

- 本地数据、文件保存将"/cache"目录空间用完。
- 数据处理过程中对数据进行解压，导致数据大小膨胀，将"/cache"目录空间用完。
- 数据未保存至/cache目录或者/home/ma-user/目录（/cache会软连接成/home/ma-user/），导致数据占满系统目录。系统目录仅支持系统功能基本运行，无法支持大数据存储。
- 部分训练任务会在训练过程中生成checkpoint文件，并进行更新。如更新过程中，未删除历史的checkpoint文件，会导致/cache目录逐步被用完。

- 实际存储空间足够，却依旧报错“No Space left on device”。可能是inode不足，或者是触发操作系统的文件索引缓存问题，导致操作系统无法创建文件，造成用户磁盘占满。

  说明

  触发条件和下面的因素有关：
  - 文件名越长，文件数量的上限越小。
  - blocksize越小，文件数量的上限越小。 blocksize系统默认为4096B，总共有三种大小：1024B、2048B、4096B。
  - 创建文件越快，越容易触发（机制大概是：有一个缓存，这块大小和上面的1和2有关，目录下文件数量比较大时会启动，使用方式是边用边释放）。
- 程序运行过程中，产生了core文件，core文件占满了"/"根目录空间。

处理方法

1. 排查数据集大小、数据集解压后的大小，checkpoint保存文件大小，是否占满了磁盘空间。
2. 如数据大小已超过/cache目录大小，则可以考虑通过SFS来额外挂载数据盘进行扩容。
3. 将数据和checkpoint保存在/cache目录或者/home/ma-user/目录。
4. 检查checkpoint相关逻辑，保证历史checkpoint不会不断积压，导致/cache目录用完。
5. 如文件大小小于/cache目录大小并且文件数量超过50w，则考虑为inode不足或者触发了操作系统的文件索引相关问题。需要：
   - 分目录处理，减少单个目录文件量。
   - 减慢创建文件的速度。如数据解压过程中，sleep 5s后再进行下一个数据的解压。
6. 如果训练作业的工作目录下有core文件生成，可以在启动脚本最前面加上如下代码，来关闭core文件产生。并推荐先在开发环境中进行代码调试。

```
import os
os.system("ulimit -c 0")
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.4 外网访问限制

15.4.4.1 日志提示“ Network is unreachable”

问题现象

在使用pytorch时，将torchvision.models中的pretrained置为了True，日志中出现如下报错：

```
‘OSError: [Errno 101] Network is unreachable’
```

原因分析

出现该问题的可能原因如下：

因为安全性问题，ModelArts内部训练机器不能访问外网。

处理方法

1. 将pretrained改成false，提前下载好预训练模型，加载下载好的预训练模型位置即可，可参考如下代码。

```
import torch
import torchvision.models as models

model1 = models.resnet34(pretrained=False, progress=True)
checkpoint = '/xxx/resnet34-333f7ec4.pth'
state_dict = torch.load(checkpoint)
model1.load_state_dict(state_dict)
```

2. 必现的问题，使用本地Pycharm远程连接Notebook调试。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.4.2 运行训练作业时提示 URL 连接超时

问题现象

训练作业在运行时提示URL连接超时，具体报错如下：

```
urllib.error.URLERROR:<urlopen error [Errno 110] Connection timed out>
```

原因分析

由于安全性问题在ModelArts上不能联网下载。

处理方法

如果在运行训练作业时提示连接超时，请您将需要联网下载的数据提前下载至本地，并上传至OBS中。

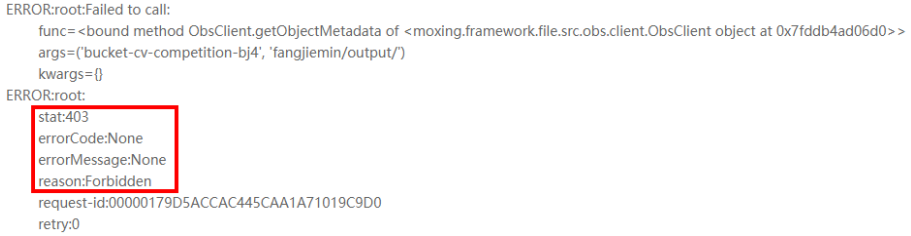
15.4.5 权限问题

15.4.5.1 训练作业访问 OBS 时，日志提示“stat:403 reason:Forbidden”

问题现象

训练作业访问OBS时，出现如下报错：

图 15-31 报错信息



原因分析

出现该问题的可能原因如下:

- OBS服务的权限出现问题，导致无法正常读取数据

处理方法

请检查OBS权限配置，如未解决问题可参考OBS文档的“已配置OBS权限，仍然无法访问OBS（403 AccessDenied）”。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

- OBS服务相关报错可根据错误信息（包括errorCode、errorMessage等）判断具体错误原因。具体错误码请参考《OBS服务SDK参考》的“Python>异常处理>OBS服务端错误码”章节:

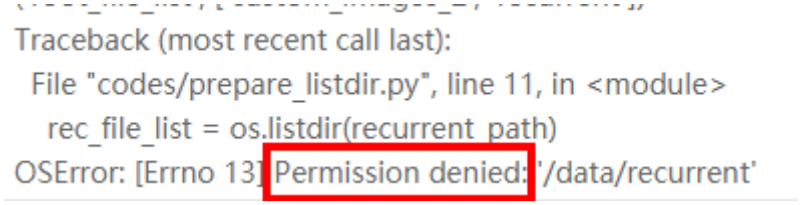
15.4.5.2 日志提示"Permission denied"

问题现象

训练作业访问挂载的EFS，或者是执行.sh启动脚本时，出现如下错误:

- [Errno 13]Permission denied: '/xxx/xxxx'

  图 15-32 错误日志

- bash: /bin/ln: Permission denied
- 自定义镜像中，bash:/home/ma-user/.pip/pip.conf：Permission Denied
- 自定义镜像中，tee: /xxx/xxxx: Permission denied cp: cannot stat '': No such file or directory

原因分析

出现该问题的可能原因如下:

- [Errno 13]Permission denied: '/xxx/xxxx'
  - 上传数据时文件所属与文件权限未修改，导致训练作业以work用户组访问时没有权限了。
  - 在代码目录中的.sh拷贝到容器之后，需要添加“x”可执行权限。
- bash: /bin/ln: Permission denied

  因安全问题，不支持用户开通使用ln命令。

- bash:/home/ma-user/.pip/pip.conf：Permission Denied
  因从V1切换到V2时，ma-user的uid仍是1102未改变导致。
- tee: /xxx/xxxx: Permission denied cp: cannot stat '': No such file or directory
  可能原因是用户使用的启动脚本为旧版本的run_train.sh，脚本里面有某些环境变量在新版本下发的作业中并不存在这些环境变量导致。
- 可能原因是使用Python file接口并发读写同一文件。

处理方法

1. 对挂载盘的数据加权限，可以改为与训练容器内相同的用户组（1000），假如/nas盘是挂载路径，执行如下代码。
   ```
   chown -R 1000：1000 /nas
   或者
   chmod 777 -R /nas
   ```
2. 如果是自定义镜像中拉取的.sh脚本没有执行权限，可以在自定义脚本启动前执行"chmod +x xxx.sh"添加可执行权限。
3. ModelArts控制台上创建训练作业自定义镜像入口，默认以1000 uid用户来启动v2容器镜像，将ma-user的uid从1102改为1000，改变方式如下（假若需要sudo权限，可取消sudoers行的注释）：

   ```
   FROM {your-v1-custom-docker-image or other docker-image}

   USER root

   # prepare moxing_framework and seccomponent package
   # and chmod/chown moxing_framework package to the right permission or owner (ma-user)

   RUN groupadd ma-group -g 1000 && \
       useradd -d /home/ma-user -m -u 1000 -g 1000 -s /bin/bash ma-user && \
       chmod 770 /home/ma-user && \
       # usermod -a -G work ma-user && \
       # alien -i seccomponent-1.0.2-2.0.release.x86_64.rpm && \
       chmod 770 /root && \
       # or silver bullet of files permission
       # chmod -R 777 /root && \
       usermod -a -G root ma-user

   # ENV LD_LIBRARY_PATH=/usr/local/seccomponent/lib:$LD_LIBRARY_PATH

   # RUN echo "ma-user  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

   # RUN pip install moxing_framework-2.0.0.rc2.4b57a67b-py2.py3-none-any.whl

   USER ma-user
   WORKDIR /home/ma-user
   ```

4. v1训练作业环境变量迁移v2说明：
   - v1的DLS_TASK_NUMBER环境变量，可以使用v2的MA_NUM_HOSTS环境变量替换，即选择的训练节点数。
   - v1的DLS_TASK_INDEX环境变量，当前可以使用v2的VC_TASK_INDEX环境变量替换，下一步使用MA_TASK_INDEX替换，建议使用demo script中的方式获取，以保证兼容性。
   - v1的BATCH_CUSTOM0_HOSTS环境变量，可以使用v2的${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}:6666替换。

- 一般而言，v1的BATCH_CUSTOM{N}_HOSTS环境变量，可以使用v2的${MA_VJ_NAME}-${MA_TASK_NAME}-{N}.${MA_VJ_NAME}:6666替换。

5. 分析代码中是否存在并发读写同一文件的逻辑，如有则进行修改。

   如用户使用多卡的作业，那么可能每张卡都会有同样的读写数据的代码，可参考如下代码修改。

```
import moxing as mox
from mindspore.communication import init, get_rank, get_group_size
init()
rank_id = get_rank()
# 仅让0号卡进行数据下载
if rank_id % 8 == 0:
    mox.file.copy_parallel('obs://bucket-name/dir1/dir2/', '/cache')
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.6 GPU 相关问题

15.4.6.1 日志提示"No CUDA-capable device is detected"

问题现象

在程序运行过程中，出现如下类似错误。

```
1.‘failed call to cuInit: CUDA_ERROR_NO_DEVICE:  no CUDA-capable device is detected’
2.‘No CUDA-capable device is detected although requirements are installed’
```

原因分析

出现该问题的可能原因如下：

- 用户/训练系统，将CUDA_VISIBLE_DEVICES传错了，检查一下CUDA_VISIBLE_DEVICES变量是否正常。
- 用户选择了1/2/4卡这些规格的作业，然后设置了CUDA_VISIBLE_DEVICES=‘1’这种类似固定的卡ID号，与实际选择的卡ID不匹配。

处理方法

1. 尽量代码里不要去修改CUDA_VISIBLE_DEVICES变量，用系统默认里面自带的。
2. 如果必须指定卡ID，需要注意一下1/2/4规格下，指定的卡ID与实际分配的卡ID不匹配的情况。
3. 如果上述方法还出现了错误，可以去notebook里面调试打印CUDA_VISIBLE_DEVICES变量，或者用以下代码测试一下，查看结果是否返回的是True。

```
import torch
torch.cuda.is_available()
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.6.2 日志提示“RuntimeError: connect() timed out”

问题现象

使用pytorch进行分布式训练时，出现如下错误：

图 15-33 错误日志

```
INFO - 03/23/21 17:20:50 - 0:00:04 - Building data done with 1331166 images loaded.
Traceback (most recent call last):
  File "swav-master/main_swav.py", line 500, in <module>
    main()
  File "swav-master/main_swav.py", line 191, in main
    mp.spawn(main_worker, nprocs=args.ngpu, args=())
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/multiprocessing/spawn.py", line 171, in spawn
    while not spawn_context.join():
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/multiprocessing/spawn.py", line 118, in join
    raise Exception(msg)
Exception:

-- Process 2 terminated with the following error:
Traceback (most recent call last):
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/multiprocessing/spawn.py", line 19, in _wrap
    fn(i, *args)
  File "/cache/user-job-dir/swav-master/main_swav.py", line 231, in main_worker
    rank=args.rank)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/distributed/distributed_c10d.py", line 397, in init_process_group
    store, rank, world_size = next(rendezvous_iterator)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/distributed/rendezvous.py", line 168, in _env_rendezvous_handler
    store = TCPStore(master_addr, master_port, world_size, start_daemon)
RuntimeError: connect() timed out.
```

原因分析

出现该问题的可能原因如下：

如果在此之前是有进行数据拷贝的，每个节点拷贝的速度不是同一个时间完成的，然后有的节点没有拷贝完，其他节点进行torch.distributed.init_process_group()导致超时。

处理方法

如果是多个节点拷贝不同步，并且没有barrier的话导致的超时，可以在拷贝数据之前，先进行torch.distributed.init_process_group()，然后再根据local_rank()==0去拷贝数据，之后再调用torch.distributed.barrier()等待所有rank完成拷贝。具体可参考如下代码：

```
import moxing as mox
import torch

torch.distributed.init_process_group()
if local_rank == 0:
    mox.file.copy_parallel(src,dst)

torch.distributed.barrier()
```

建议与总结

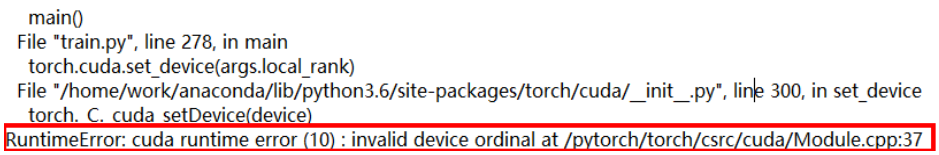
在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.6.3 日志提示“cuda runtime error (10) : invalid device ordinal at xxx”

问题现象

训练作业失败，日志报出如下错误：

图 15-34 错误日志



原因分析

可以从以下角度排查：

- 请检查CUDA_VISIBLE_DEVICES设置的值是否与作业规格匹配。例如您选择4卡规格的作业，实际可用的卡ID为0、1、2、3，但是您在进行cuda相关的运算时，例如"tensor.to(device="cuda:7")"，将张量搬到了7号GPU卡上，超过了实际可用的ID号。
- 如果cuda相关运算设置的卡ID号在所选规格范围内，但是依旧出现了上述报错。可能是该资源节点中存在GPU卡损坏的情况，导致实际能检测到的卡少于所选规格。

处理方法

1. 建议直接根据系统分卡情况下传进去的CUDA_VISIBLE_DEVICES去设置，不用手动指定默认的。
2. 如果发现资源节点中存在GPU卡损坏，请联系技术支持处理。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.6.4 日志提示“RuntimeError: Cannot re-initialize CUDA in forked subprocess”

问题现象

在使用pytorch启动多进程的时候，出现如下报错：

```
RuntimeError: Cannot re-initialize CUDA in forked subprocess
```

原因分析

出现该问题的可能原因如下：

multiprocessing启动方式有误。

处理方法

可以参考**官方文档**，如下：

```
"""run.py:"""
#!/usr/bin/env python
import os
import torch
import torch.distributed as dist
import torch.multiprocessing as mp

def run(rank, size):
```

```
    """ Distributed function to be implemented later. """
    pass

def init_process(rank, size, fn, backend='gloo'):
    """ Initialize the distributed environment. """
    os.environ['MASTER_ADDR'] = '127.0.0.1'
    os.environ['MASTER_PORT'] = '29500'
    dist.init_process_group(backend, rank=rank, world_size=size)
    fn(rank, size)


if __name__ == "__main__":
    size = 2
    processes = []
    mp.set_start_method("spawn")
    for rank in range(size):
        p = mp.Process(target=init_process, args=(rank, size, run))
        p.start()
        processes.append(p)

    for p in processes:
        p.join()
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.6.5 训练作业找不到 GPU

问题现象

训练作业运行出现如下报错：

```
failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
```

原因分析

根据错误信息判断，报错原因为训练作业运行程序读取不到GPU。

处理方法

根据报错提示，请您排查代码，是否已添加以下配置，设置该程序可见的GPU：

```
os.environ['CUDA_VISIBLE_DEVICES'] = '0,1,2,3,4,5,6,7'
```

其中，0为服务器的GPU编号，可以为0，1，2，3等，表明对程序可见的GPU编号。若未进行添加配置则该编号对应的GPU不可用。

15.4.7 业务代码问题

15.4.7.1 日志提示“pandas.errors.ParserError: Error tokenizing data. C error: Expected .* fields”

问题现象

使用pandas读取csv数据表时，日志报出如下错误导致训练作业失败：

```
pandas.errors.ParserError: Error tokenizing data. C error: Expected 4 field
```

原因分析

csv中文件的每一行的列数不相等。

处理方法

可以使用以下方法处理：

- 校验csv文件，将多出字段的行删除。
- 在代码中忽略错误行，参考如下：
  ```
  import pandas as pd
  pd.read_csv(filePath,error_bad_lines=False)
  ```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.2 日志提示“max_pool2d_with_indices_out_cuda_frame failed with error code 0”

问题现象

pytroch1.3镜像中，去升级了pytroch1.4的版本，导致之前在pytroch1.3跑通的代码报错如下：

```
“RuntimeError:max_pool2d_with_indices_out_cuda_frame failed with error code 0”
```

原因分析

出现该问题的可能原因如下：

pytorch1.4引擎与之前pytorch1.3版本兼容性问题。

处理方法

1. 在images之后添加contigous。
   ```
   images = images.cuda()
   pred = model(images.permute(0, 3, 1, 2).contigous())
   ```
2. 将版本回退至pytorch1.3。
3. 必现的问题，使用本地Pycharm远程连接Notebook调试。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.3 训练作业失败，返回错误码 139

问题现象

训练作业运行失败，返回错误码139，如下图所示：

原因分析

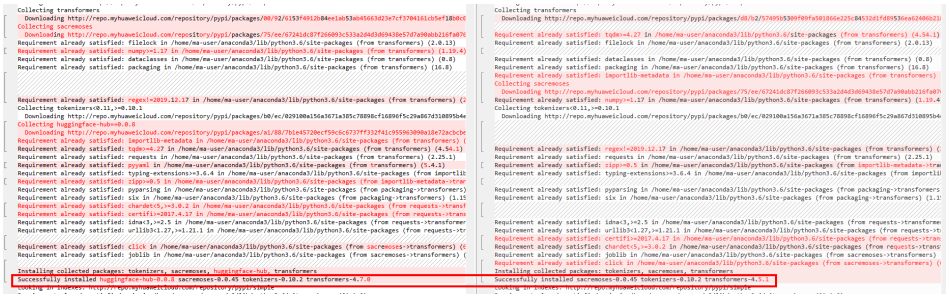
出现该问题的可能原因如下

- pip源中的pip包更新了，之前能跑通的代码，在包更新之后产生了不兼容的情况，例如transformers包，导致import的时候出现了错误。
- 用户代码问题，出现了内存越界、非法访问内存空间的情况。
- 未知系统问题导致，建议先尝试重建作业，重建后仍然失败，建议提工单定位。

处理方法

1. 如果存在之前能跑通，什么都没修改，过了一阵跑不通的情况，先去排查跑通和跑不通的日志是否存在pip源更新了依赖包，如下图，安装之前跑通的老版本即可。

   图 15-35 PIP 安装对比图

2. 推荐您使用本地Pycharm远程连接Notebook调试。
3. 如果上述情况都解决不了，请联系技术支持工程师。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.4 训练作业失败，如何使用开发环境调试训练代码?

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.5 日志提示“ '(slice(0, 13184, None), slice(None, None, None))' is an invalid key”

问题现象

训练过程中出现如下报错：

```
TypeError: '(slice(0, 13184, None), slice(None, None, None))' is an invalid key
```

原因分析

出现该问题的可能原因如下：

切分数据时，选择的数据不对。

处理方法

尝试如下代码：

```
X = dataset.iloc[:,:-1].values
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.6 日志报错“DataFrame.dtypes for data must be int, float or bool”

问题现象

训练过程中出现如下报错：

```
DataFrame.dtypes for data must be int, float or bool
```

原因分析

出现该问题的可能原因如下：

训练数据中出现了非int、float、bool类型数据。

处理方法

可参考如下代码，将错误列进行转换：

```
from sklearn import preprocessing
lbl = preprocessing.LabelEncoder()
train_x['acc_id1'] = lbl.fit_transform(train_x['acc_id1'].astype(str))
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.7 日志提示“CUDNN_STATUS_NOT_SUPPORTED.”

问题现象

在pytorch训练时，出现如下报错：

```
RuntimeError: cuDNN error: CUDNN_STATUS_NOT_SUPPORTED. This error may appear if you passed in a non-contiguous input.
```

原因分析

出现该问题的可能原因如下：

数据输入不连续，cuDNN不支持的类型。

处理方法

1. 禁用cuDNN，在训练前加入如下代码。

   ```
   torch.backends.cudnn.enabled = False
   ```

2. 将输入数据转换成contiguous。

```
images = images.cuda()
images = images.permute(0, 3, 1, 2).contigous()
```

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.8 日志提示“Out of bounds nanosecond timestamp”

问题现象

在使用pandas.to_datetime转换时间时，出现如下报错：

```
pandas._libs.tslibs.np_datetime.OutOfBoundsDatetime: Out of bounds nanosecond timestamp: 1-01-02 13:20:00
```

原因分析

出现该问题的可能原因如下：

时间值越界，请参考**官方文档**。

处理方法

校验时间数据，pandas以纳秒表示时间戳。

- 最小时间：1677-09-22 00:12:43.145225
- 最大时间：2262-04-11 23:47:16.854775807，需注意上下界限。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.9 日志提示“Unexpected keyword argument passed to optimizer”

问题现象

在使用keras时，升级版本>=2.3.0之后，之前跑通的代码出现如下报错：

```
TypeError: Unexpected keyword argument passed to optimizer: learning_rate
```

原因分析

出现该问题的可能原因是“learning_rate”的参数名称写错了。keras官方文档中说明参数“lr”已重命名为“learning_rate”，在训练代码中必须写成“learning_rate”才能调用成功。keras官方文档请参见**https://github.com/keras-team/keras/releases/tag/2.3.0**。

处理方法

将训练代码里的参数名称“lr”改成“learning_rate”。

建议与总结

在创建训练作业前，推荐您先使用ModelArts开发环境调试训练代码，避免代码迁移过程中的错误。

15.4.7.10 日志提示“no socket interface found”

问题现象

在pytorch镜像运行分布式作业时，设置NCCL日志级别，代码如下：

```
import os
os.environ["NCCL_DEBUG"] = "INFO"
```

会出现如下错误：

图 15-36 错误日志

```
job0879f61e-job-base-pda-2-0:712:712 [0] bootstrap.cc:37 NCCL WARN Bootstrap : no socket interface found
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO init.cc:128 -> 3
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO bootstrap.cc:76 -> 3
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO bootstrap.cc:245 -> 3
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO bootstrap.cc:266 -> 3
Traceback (most recent call last):
  File "train_net.py", line 1923, in <module>
    main_worker(args)
  File "train_net.py", line 355, in main_worker
    network = torch.nn.parallel.DistributedDataParallel(network, device_ids=device_ids, find_unused_parameters=True)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/nn/parallel/distributed.py", line 298, in __init__
    self.broadcast_bucket_size)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/nn/parallel/distributed.py", line 480, in _distributed_broadcast_coalesced
    dist._broadcast_coalesced(self.process_group, tensors, buffer_size)
RuntimeError: NCCL error in: /pytorch/torch/lib/c10d/ProcessGroupNCCL.cpp:374, internal error
Traceback (most recent call last):
```

原因分析

可能原因如下：

- 原因1：未设置环境变量NCCL_IB_TC、NCCL_IB_GID_INDEX、NCCL_IB_TIMEOUT，因此会导致通信速度慢且不稳定，最后造成IB通信断连，偶发上述现象。
- 原因2：NCCL_SOCKET_IFNAME设置错误。当用户的NCCL版本低于2.14时，则需要手动设置NCCL_SOCKET_IFNAME环境变量。

处理方法

- 针对原因1，需要在代码中补充如下环境变量。

  ```
  import os
  os.environ["NCCL_IB_TC"] = "128"
  os.environ["NCCL_IB_GID_INDEX"] = "3"
  os.environ["NCCL_IB_TIMEOUT"] = "22"
  ```

- 针对原因2，需要在代码中设置环境变量NCCL_SOCKET_IFNAME。

  ```
  import os
  os.environ["NCCL_SOCKET_IFNAME"] = "eth0"
  ```

  说明

  只有当用户的NCCL版本低于2.14时，才需要进行以上设置。

15.4.7.11 日志提示“Runtimeerror: Dataloader worker (pid 46212) is killed by signal: Killed BP”

问题现象

训练作业日志运行出现如下报错：Runtimeerror: Dataloader worker (pid 46212) is killed by signal: Killed BP。

原因分析

由于batch size过大，导致Dataloader进程退出。

处理方法

请调小batch size的数值。

15.4.7.12 日志提示“AttributeError: 'NoneType' object has no attribute 'dtype'”

问题现象

代码在Notebook的keras镜像中可以正常运行，在训练模块使用tensorflow.keras训练报错时，出现如下报错：AttributeError: 'NoneType' object has no attribute 'dtype'。

原因分析

训练镜像的numpy版本与Notebook中不一致。

处理方法

在代码中打印出numpy的版本，查看是否为1.18.5版本，若非该版本号则在代码开始处执行：

```
import os
os.system('pip install numpy==1.18.5')
```

如果依旧有报错情况，将以上代码修改为：

```
import os
os.system('pip install numpy==1.18.5')
os.system('pip install keras==2.6.0')
os.system('pip install tensorflow==2.6.0')
```

15.4.7.13 日志提示“No module name 'unidecode'”

问题现象

从mindspore开源gitee中master分支下下载的tacotron2模型，修改配置文件后上传ModelArts准备训练，日志报错提示：No module name 'unidecode'。

原因分析

requirements.txt的Unidecode名字写错了，应该把U改成小写，所以导致训练作业的环境没有装上unidecode模块。

处理方法

将requirements.txt中的Unidecode改为unidecode。

建议与总结

您可以在训练代码里添加一行：

```
os.system('pip list')
```

然后运行训练作业，查看日志中是否有所需要的模块。

15.4.7.14 分布式 Tensorflow 无法使用“tf.variable”

问题现象

多机或多卡使用“tf.variable”会造成以下错误：WARNING:tensorflow:Gradient is None for variable:v0/tower_0/UNET_v7/sub_pixel/Variable:0.Make sure this variable is used in loss computation.

图 15-37 分布式 Tensorflow 无法使用

```
WARNING:tensorflow:Gradient is None for varaible: v0/tower_0/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0/tower_0/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_1/tower_1/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_1/tower_1/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_2/tower_2/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_2/tower_2/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_3/tower_3/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_3/tower_3/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_4/tower_4/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_4/tower_4/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_5/tower_5/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_5/tower_5/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_6/tower_6/UNET_v7/sub_pixel/Variable:0. Make sure this variable is used in loss computation.
WARNING:tensorflow:Gradient is None for varaible: v0_6/tower_6/UNET_v7/sub_pixel/Variable_1:0. Make sure this variable is used in loss computation.
```

原因分析

分布式Tensorflow不能使用“tf.variable”要使用“tf.get_variable”。

处理方法

请您将“启动文件”中的“tf.variable”替换为“tf.get_variable”。

15.4.7.15 MXNet 创建 kvstore 时程序被阻塞，无报错

问题现象

使用**kv_store = mxnet.kv.create('dist_async')**方式创建“kvstore”时程序被阻塞。如，执行如下代码，如果无法输出“end”，表明程序阻塞。

```
print('start')
kv_store = mxnet.kv.create('dist_async')
print('end')
```

原因分析

worker阻塞的原因可能是连不上server。

处理方法

将如下代码放在“启动文件”里“import mxnet”之前可以看到节点间相互通信状态，同时ps能够重新发送。

```
import os
os.environ['PS_VERBOSE'] = '2'
os.environ['PS_RESEND'] = '1'
```

其中，“os.environ['PS_VERBOSE'] = '2'”为打印所有的通信信息。
“os.environ['PS_RESEND'] = '1'”为在“PS_RESEND_TIMEOUT”毫秒后没有收到ACK消息，Van实例会重发消息。

15.4.7.16 日志出现 ECC 错误，导致训练作业失败

问题现象

训练作业日志运行出现如下报错：RuntimeError: CUDA error: uncorrectable ECC error encountered

原因分析

由于ECC错误，导致作业运行失败。

处理方法

当ECC错误且计数超过64时，系统会自动隔离故障节点，重启训练作业确认故障是否解决。如果未隔离的节点导致训练作业再次失败或卡死，请联系技术支持处理。

15.4.7.17 超过最大递归深度导致训练作业失败

问题现象

ModelArts训练作业报错：

```
RuntimeError: maximum recursion depth exceeded in __instancecheck__
```

原因分析

递归深度超过了Python默认的递归深度，导致训练失败。

处理方法

如果超过最大递归深度，建议您在启动文件中增大递归调用深度，具体操作如下：

```
import sys
sys.setrecursionlimit(1000000)
```

15.4.7.18 使用预置算法训练时，训练失败，报“bndbox”错误

问题现象

使用预置算法创建训练作业，训练失败，日志中出现如下报错。

```
KeyError: 'bndbox'
```

原因分析

用于训练的数据集中，使用了“非矩形框”标注。而预置使用算法不支持“非矩形框”标注的数据集。

处理方法

此问题有两种解决方法：

- 方法1：使用常用框架自行编码开发模型，支持“多边形”标注的数据集。
- 方法2：修改数据集，使用矩形标注。然后再启动训练作业。

15.4.7.19 训练作业状态显示“审核作业初始化”

问题现象

当创建训练作业的“算法来源”选择“自定义”镜像创建训练作业时，训练作业状态显示审核作业初始化。

原因分析

自定义镜像首次运行时，需要先审核镜像，通过审核之后才可创建作业，即当前状态为审核作业初始化。

15.4.7.20 训练作业进程异常退出

问题现象

训练作业运行失败，日志中出现如下类似报错：

```
[Modelarts Service Log]Training end with return code: 137
```

原因分析

日志显示训练进程的退出码为137。训练进程表示用户的代码启动后的进程，所以这里的退出码是用户的训练作业代码返回的。常见的错误码还包括247、139等。

- 退出码137或者247

  可能是内存溢出造成的。请减少数据量、减少batch_size，优化代码，合理聚合、复制数据。

  说明

  请注意，数据文件大小不等于内存占用大小，需仔细评估内存使用情况。

- 退出码139

请排查安装包的版本，可能存在包冲突的问题。

排查办法

根据错误信息判断，报错原因来源于用户代码。

您可以通过以下两种方式排查：

- 线上环境调试代码（仅适用于非分布式代码）
  a. 在开发环境（notebook）申请相同规格的开发环境实例。
  b. 在notebook调试用户代码，并找出问题的代码段。
  c. 通过关键代码段 + 退出码尝试去搜索引擎寻找解决办法。，
- 通过训练日志排查问题
  a. 通过日志判断出问题的代码范围。
  b. 修改代码，在问题代码段添加打印，输出更详细的日志信息。
  c. 再次运行作业，判断出问题的代码段。

15.4.7.21 训练作业进程被 kill

问题现象

用户进程被Kill表示用户进程因外部因素被Kill或者中断，表现为日志中断。

原因分析

- CPU软锁
  在解压大量文件可能会出现此情况并造成节点重启。可以适当在解压大量文件时，加入sleep。比如每解压1w个文件，就停止1s。
- 存储限制
  根据规格情况合理使用数据盘。
- CPU过载
  减少线程数。

排查办法

根据错误信息判断，报错原因来源于用户代码。

您可以通过以下两种方式排查：

- 线上环境调试代码（仅适用于非分布式代码）
  a. 在开发环境（notebook）申请相同规格的开发环境实例。
  b. 在notebook调试用户代码，并找出问题的代码段。
  c. 通过关键代码段 + 退出码尝试去搜索引擎寻找解决办法。，
- 通过训练日志排查问题
  a. 通过日志判断出问题的代码范围。
  b. 修改代码，在问题代码段添加打印，输出更详细的日志信息。
  c. 再次运行作业，判断出问题的代码段。

15.4.8 训练作业卡死

15.4.8.1 复制数据卡死

问题现象

调用mox.file.copy_parallel拷贝数据时卡死。

解决方案

- 拷贝文件和文件夹均可采用：
  ```
  import moxing as mox
  mox.file.set_auth(is_secure=False)
  ```
- 拷贝单个大文件5G以上时可采用：
  ```
  from moxing.framework.file import file_io
  ```
  查看当前moxing调用的接口版本：file_io._LARGE_FILE_METHOD，如果输出值为1则为V1版本，如果输出值为2，则为V2版本。

  V1版本修改：file_io._NUMBER_OF_PROCESSES=1

  V2版本修改：可以 file_io._LARGE_FILE_METHOD = 1，将模式设置成V1然后用V1的方式修改规避，也可以直接file_io._LARGE_FILE_TASK_NUM=1。
- 拷贝文件夹时可采用：
  ```
  mox.file.copy_parallel(threads=0,is_processing=False)
  ```

15.4.8.2 训练前卡死

作业为多节点训练，且还未开始训练时发生卡死，可以在代码中加入os.environ["NCCL_DEBUG"] = "INFO"，查看NCCL DEBUG信息。

问题现象 1

日志中还未出现NCCL DEBUG信息时已卡死。

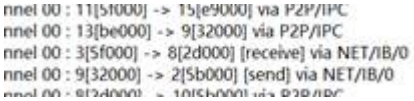
解决方案 1

检查代码，检查是否有参数中未传入“master_ip”和“rank”参数等问题。

问题现象 2

若发现有的节点含有GDR信息（分布式）。

而有的节点无GDR信息，导致卡死的原因可能为GDR。



解决方案 2

在程序开头设置os.environ["NCCL_NET_GDR_LEVEL"] = '0'或者寻找运维人员将机器添加GDR。

问题现象 3

NCCL信息中报出Got completion with error 12, opcode 1, len 32478, vendor err 129等通信信息时，说明当前网络不是很稳定。

解决方案 3

可加入3个环境变量。

- NCCL_IB_GID_INDEX=3： 使用RoCE v2协议，默认使用RoCE v1，但是v1在交换机上没有拥塞控制，可能丢包，而且后面的交换机不会支持v1，就无法启动。
- NCCL_IB_TC=128：数据包走交换机的队列4通道，这是RoCE协议标准。
- NCCL_IB_TIMEOUT=22：把超时时间设置长一点，正常情况下网络不稳定会有5s钟左右的间断，超过5秒就返回timeout了，改成22预计有二十秒左右，算法为4.096 μs * 2 ^ timeout。

15.4.8.3 训练中途卡死

问题现象 1

检测每个节点日志是否有报错信息，某个节点报错但作业未退出导致整个训练作业卡死。

解决方案 1

查看报错原因，解决报错。

问题现象 2

作业卡在sync-batch-norm中或者训练速度变慢。pytorch如果开了sync-batch-norm，多机会慢，因开了sync-batch-norm以后，每一个iter里面每个batch-norm层都要做同步，通信量很大，而且要所有节点同步。

```
from sync_batchnorm import SynchronizedBatchNorm1d, DataParallelWithCallback

sync_bn = SynchronizedBatchNorm1d(10, eps=1e-5, affine=False)
sync_bn = DataParallelWithCallback(sync_bn, device_ids=[0, 1])
```

解决方案 2

关掉sync-batch-norm，或者升pytorch版本，升级pytorch到1.10。

问题现象 3

作业卡在tensorboard中，如下图所示：

```
writer = SummaryWriter('./path/to/log')
```

解决方案 3

存储路径设为本地路径，如cache/tensorboard，不要使用OBS路径。

问题现象 4

使用pytorch中的dataloader读数据时，作业卡在读数据过程中，日志停在训练的过程中并不再更新日志。

解决方案 4

用dataloader读数据时，适当减小Numwork，如下图所示：

```
from torch.utils.data import DataLoader

train_loader = DataLoader(dataset=train_data, batch_size=train_bs, shuffle=True, num_worker=4)

valid_loader = DataLoader(dataset=valid_data, batch_size=valid_bs, num_worker=4)
```

15.4.8.4 训练最后一个 epoch 卡死

问题现象

通过日志查看数据切分是否对齐，若未对齐，容易导致部分进程完成训练退出，而部分训练进程因未收到其他进程反馈卡死，如下图同一时间有的进程在epoch48，而有的进程在epoch49。

```
loss exit lane:0.12314446270465851
step loss is 0.29470521211624146
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:2 Epoch:[48][20384/all] Data Time 0.000(0.000) Net Time 0.705(0.890) Loss 0.3403(0.3792)LR 0.00021887
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:1 Epoch:[48][20384/all] Data Time 0.000(0.000) Net Time 0.705(0.891) Loss 0.3028(0.3466) LR 0.00021887
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:4 Epoch:[49][20384/all] Data Time 0.000(0.147) Net Time 0.705(0.709) Loss 0.3364(0.3414)LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:3 Epoch:[49][20384/all] Data Time 0.000 (0.115) Net Time 0.706(0.814) Loss 0.3345(0.3418) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:0 Epoch:[49][20384/all] Data Time 0.000(0.006) Net Time 0.704(0.885) Loss 0.2947(0.3566) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:7 Epoch:[49][20384/all] Data Time 0.001 (0.000) Net Time 0.706 (0.891) Loss 0.3782(0.3614) LR 0.00021887
[2022-04-26 13:57:20,759][INFO][train_epoch]:Rank:5 Epoch:[48][20384/all] Data Time 0.000(0.000) Net Time 0.706(0.891) Loss 0.5471(0.3642) LR 0.00021887
[2022-04-26 13:57:20,763][INFO][train_epoch]:Rank:6 Epoch:[49][20384/all] Data Time 0.000(0.000) Net Time 0.704(0.891) Loss 0.2643(0.3390)LR 0.00021887
stage 1 loss 0.4600560665130615 mul_cls_loss loss:0.01245919056236744 mul_offset_loss 0.44759687781333923 origin stage2_loss 0.048592399805784225
stage 1 loss:0.4600560665130615 stage 2 loss:0.048592399805784225 loss exit lane:0.10233864188194275
```

解决方案

使用tensor的切分操作对齐数据。

15.4.9 训练作业运行失败

15.4.9.1 训练作业运行失败排查指导

问题现象

训练作业的“状态”出现“运行失败”的现象。

原因分析及处理方法

- 查看训练作业的“日志”，出现报错“MoxFileNotExistsException(resp, 'file or directory or bucket not found.')”。
  - 原因：Moxing在进行文件复制时，未找到train_data_obs目录。
  - 处理建议：修改train_data_obs目录为正确地址，重新启动训练作业。

  须知

  另外在Moxing下载OBS对象过程中，不要删除相应OBS目录下的对象，否则Moxing在下载到被删除的对象时会下载失败。

- 查看训练作业的“日志”，出现报错“NVIDIA A30 with CUDA capability sm_80 is not compatible with the current PyTorch installation.The current PyTorch install supports CUDA capabilities sm_37 sm_50 sm_60 sm_70'”。
  - 原因：训练作业使用的镜像CUDA版本只支持sm_37、sm_50、sm_60和sm_70的加速卡，不支持sm_80。
  - 处理建议：使用自定义镜像创建训练作业，并安装高版本的cuda以及对应的PyTorch版本。
- 查看训练作业的“日志”，出现报错“ERROR:root:label_map.pbtxt cannot be found. It will take a long time to open every annotation files to generate a tmp label_map.pbtxt.”。
  - 如果使用的是AI Gallery订阅的算法，建议先检查数据的标签是否有问题。
  - 如果使用的是物体检测类算法，建议检查数据的label框是否为非矩形。

    说明

    物体检测类算法仅支持矩形label框。
- 查看训练作业的“日志”，出现报错“RuntimeError: The server socket has failed to listen on any local network address. The server socket has failed to bind to [::]:29500 (errno: 98 - Address already in use). The server socket has failed to bind to 0.0.0.0:29500 (errno: 98 - Address already in use).”。
  - 原因：训练作业的端口号有冲突。
  - 处理建议：更改代码中的端口号，重启训练作业。
- 查看训练作业的“日志”，出现报错“WARNING: root: Retry=7, Wait=0.4, Times tamp=1697620658.6282516”。
  - 原因：Moxing版本太低。
  - 处理建议：联系技术支持将Moxing版本升级至2.1.6及以上版本。

15.4.9.2 训练作业运行失败，出现 NCCL 报错

问题现象

训练作业的状态“运行失败”，查看训练作业的“日志”，存在NCCL的报错，例如“NCCL timeout”、“RuntimeError: NCCL communicator was aborted on rank 7”、“NCCL WARN Bootstrap : no socket interface found”或“NCCL INFO Call to connect returned Connection refused, retrying”。

原因分析

NCCL是一个提供GPU间通信原语的库，实现集合通信和点对点发送/接收原语。当训练作业出现NCCL的报错时，可以通过调整NCCL的环境变量尝试解决问题。

处理步骤

1. 进入状态“运行失败”的训练作业详情页，单击“日志”页签，查看NCCL报错。
   - 如果出现报错“NCCL timeout”或者“RuntimeError: NCCL communicator was aborted on rank 7”，则表示InfiniBand Verbs超时。单击右侧“重建”，重新创建训练作业，设置环境变量“NCCL_IB_TIMEOUT=22”，提交训练作业后等待作业完成。
   - 如果出现报错“NCCL WARN Bootstrap : no socket interface found”或“NCCL INFO Call to connect returned Connection refused, retrying”，则表示NCCL无法找到通信网卡或者是无法正常访问IP地址。需要排查训练代码中是否有设置NCCL_SOCKET_IFNAME环境变量，该环境变量由系统自动注入，训练代码中无需设置。训练代码去除NCCL_SOCKET_IFNAME环境变量设置逻辑后，单击右侧“重建”，重新创建训练作业，提交训练作业后等待作业完成。
2. 等待训练作业是否变成“已完成”状态。
   - 是，故障处理完成。
   - 否，则联系技术支持排查节点状态。

建议与总结

- 环境变量NCCL_SOCKET_IFNAME用于指定通信的网卡名称。“NCCL_SOCKET_IFNAME=eth0”表示仅使用eth0网卡通信。该环境变量由系统自动注入，由于通信网卡名称不固定，因此训练代码不应默认设置该环境变量。
- 环境变量NCCL_IB_TIMEOUT用于控制InfiniBand Verbs超时。NCCL使用的默认值为18，取值范围是1~22。

15.4.9.3 使用自定义镜像创建的训练作业一直处于运行中

问题现象

使用自定义镜像创建训练作业，训练作业的“状态”一直处于“运行中”。

原因分析及处理办法

日志打印如下内容，表示自定义镜像的CPU架构与资源池节点的CPU架构不一致。

```
standard_init_linux.go:215: exec user process caused "exec format error"
libcontainer: container start initialization failed: standard_init_linux.go:215: exec user process caused "exec format error"
```

常见场景为使用自定义镜像创建作业时选择的资源类型和规格错误。例如，自定义镜像是ARM CPU架构，应选用NPU规格的资源，却使用了X86 CPU/X86 GPU规格的资源。

15.4.9.4 训练作业的监控内存指标持续升高直至作业失败

问题现象

训练作业的“状态”为“运行失败”。

原因分析

训练作业的监控内存指标持续升高导致最后训练作业失败。

处理步骤

1. 查询训练作业的日志和监控信息，是否存在明确的OOM报错信息。
   - 是，训练作业的日志里存在OOM报错，执行**2**。
   - 否，训练作业的日志里没有OOM报错，但是存在监控指标异常，执行**3**。
2. 排查训练代码是否存在不断占用资源的代码，使得资源未被合理使用。
   - 是，优化代码，等待作业运行正常。
   - 否，提高训练作业使用的资源规格或者联系技术支持。
3. 重启训练作业，使用CloudShell登录训练容器监控内存指标，确认是否有突发性的内存增加现象。
   - 是，排查内存突发增加的时间点附近的训练作业日志，优化对应的代码逻辑，减少内存申请。
   - 否，提高训练作业使用的资源规格或者联系技术支持。

15.4.10 专属资源池创建训练作业

15.4.10.1 创建训练作业界面无云存储名称和挂载路径排查思路

问题现象

创建训练作业界面没有云存储名称和挂载路径这两个选项。

原因分析

用户的专属资源池没有进行网络打通，或者用户没有创建过SFS。

处理方法

在专属资源池列表中，单击资源池“ID/名称”，进入详情页。单击右上角“配置NAS VPC”，检查是否开启了NAS VPC。详情页面的“NAS VPC名称”和“NAS 子网ID”如果为空则证明没有开启，单击右上角配置NAS VPC即可。

如果单击开启后报错，可能是由于对应的VPC已经创建了对等连接，删除对等连接即可。

15.4.10.2 创建训练作业时出现“实例挂卷失败”的事件

问题现象

训练作业的状态一直在“创建中”，查看训练作业的“事件”，有异常信息“实例挂卷失败”，详情为“Unable to mount volumes for pod xxx ... list of unmounted volumes=[nfs-x]”。

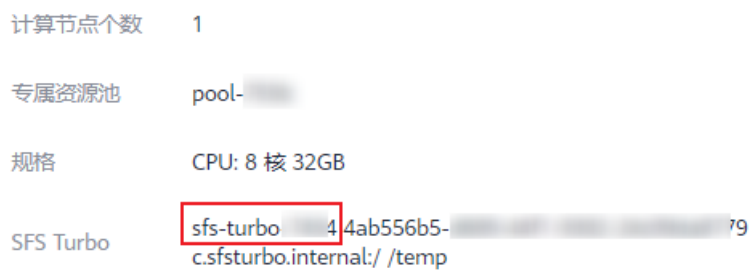
原因分析

用户账号下的SFS Turbo所在的VPC网络需要与专属资源池所在的网络打通，运行于该专属资源池的训练作业才能正常挂载SFS。因此，当训练作业挂载SFS失败时，可能是网络不通导致的。

处理步骤

1. 进入训练作业详情页，在左侧获取SFS Turbo的名称。

   图 15-38 获取 SFS Turbo 的名称

2. 登录弹性文件服务SFS控制台，在SFS Turbo列表找到训练作业挂载的SFS Turbo，单击名称进入详情页。获取VPC信息、安全组信息和endpoint信息。
   - VPC信息：SFS Turbo详情页的“虚拟私有云”。
   - 安全组信息：SFS Turbo详情页的“安全组”。
   - endpoint信息：SFS Turbo详情页的“共享路径”，去除“:/”即为sfs-turbo-endpoint。例如共享路径为“4ab556b5-d689-44f1-9302-24c09daxxxxc.sfsturbo.internal:/”，则sfs-turbo-endpoint为“4ab556b5-d689-44f1-9302-24c09daxxxxc.sfsturbo.internal”。
3. 查看SFS Turbo的VPC网段是否满足如下2个条件。

   条件一：SFS Turbo网段不能与192.168.20.0/24重叠，否则会和专属资源池的网段发生冲突，因为专属资源池的默认网段为192.168.20.0/24。专属资源池实际使用的网段可以在资源池的详情页面查看“网络”获取。

   条件二：SFS Turbo网段不能与172网段重叠，否则会和容器网络发生冲突，因为容器网络使用的是172网段。
   - 如果不满足条件，则修改SFS Turbo的VPC网段，推荐网段为10.X.X.X。
   - 如果满足条件，则继续下一步。

4. 查看SFS Turbo的VPC网段的安全组是否被限制了。

   在所选专属资源池中新建一个未挂载的SFS Turbo的训练作业，当训练作业处于“运行中”时，通过Cloud Shell功能登录训练作业worker-0实例，使用**curl {sfs-turbo-endpoint}:{port}**命令检查port是否正常打开，SFS Turbo所需要入方向的端口号为111、445、2049、2051、2052、20048。

   - 是，则修改安全组的配置。
   - 否，则继续下一步。

5. 确认SFS Turbo是否存在异常。

   新建一个和SFS Turbo在同一个网段的ECS，用ECS去挂载SFS Turbo，如果挂载失败，则表示SFS Turbo异常。

   a. 是，联系SFS服务的技术支持处理。
   b. 否，联系ModelArts的技术支持处理。

15.4.11 训练作业性能问题

15.4.11.1 训练作业性能降低

问题现象

使用ModelArts平台训练算法训练耗时增加。

原因分析

可能存在如下原因：

1. 平台上的代码经过修改优化、训练参数有过变更。
2. 训练的GPU硬件工作出现异常。

处理方法

1. 请您对作业代码进行排查分析，确认是否对训练代码和参数进行过修改。
2. 检查资源分配情况（cpu/mem/gpu/snt9/infiniband）是否符合预期。
3. 通过CloudShell登录到Linux工作页面，检查GPU工作情况：
   - 通过输入“nvidia-smi”命令，查看GPU工作是否异常。
   - 通过输入“nvidia-smi -q -d TEMPERATURE”命令， 查看TEMP参数是否存在异常， 如果温度过高，会导致训练性能下降。

15.5 推理部署

15.5.1 AI 应用管理

15.5.1.1 创建 AI 应用失败，如何定位和处理问题?

问题定位和处理

创建AI应用失败有两种场景：创建AI应用时直接报错或者是调用API报错和创建AI应用任务下发成功，但最终AI应用创建失败。

1. 创建AI应用时直接报错或者是调用API报错。一般都是输入参数不合法导致的。您可以根据提示信息进行排查修改即可。
2. 创建AI应用任务下发成功，但最终AI应用创建失败。需要从以下几个方面进行排查：
   - 在AI应用详情页面，查看“事件”页签中的事件信息。根据事件信息分析AI应用失败原因，进行处理。
   - 如果AI应用状态为“构建失败”，可以在AI应用详情页面，查看“事件”页签中的“查看构建日志”。构建日志中有对应的构建镜像失败的详细原因，根据构建失败的原因进行排查处理。

   图 15-39 查看构建日志

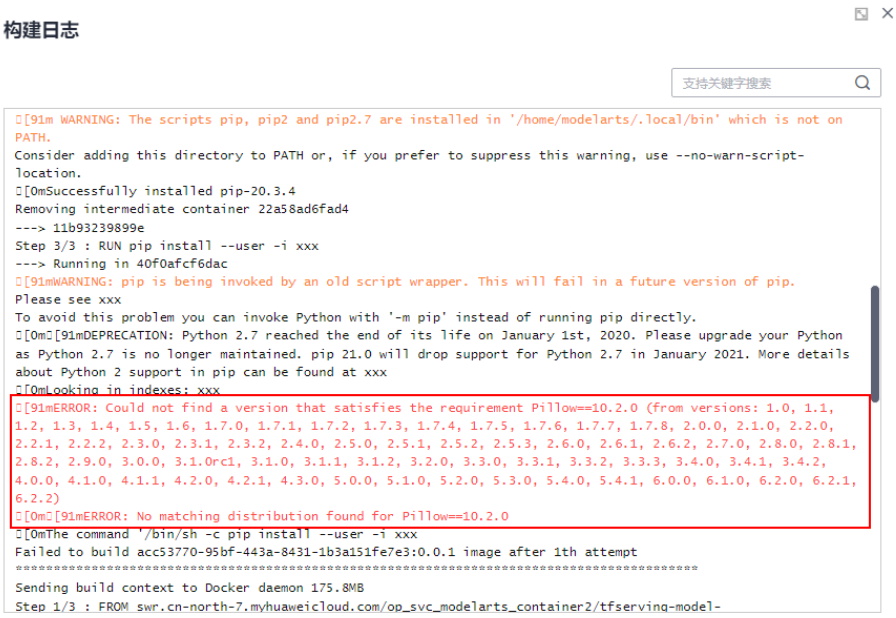
常见问题

1. 模型文件目录下不能出现dockerfile文件；
   “查看构建日志”中显示“Not only a Dockerfile in your OBS path, please make sure, The dockerfile list”，表示dockerfile文件目录有问题，模型文件目录下不能出现dockerfile文件，需要去掉模型文件目录下存在dockerfile文件。

图 15-40 构建日志：dockerfile 文件目录有问题

2. pip软件包版本不匹配，需要修改为日志中打印的存在的版本。

图 15-41 pip 版本不匹配



3. 构建日志中出现报错：“exec /usr/bin/sh: exec format error”。

   这种报错一般是因为所用镜像系统引擎和构建镜像的系统引擎不一致引起的，例如使用的是x86的镜像却标记的是arm的系统架构。

   可以通过查看AI应用详情看到配置的系统运行架构。

15.5.1.2 用户创建 AI 应用时构建镜像或导入文件失败

问题现象

- 用户创建AI应用时，构建镜像失败，失败日志中提示下载obs文件失败（Get object size from OBS failed！）。

图 15-42 下载 obs 文件失败

```
#================================ download_obs_file ================================
Successfully to download file cnnorth7-infer-model1/2d3d3591-1aaa-49e8-af22-483d0a6b31bc/Dockerfile from OBS
Successfully to download file cnnorth7-infer-model1/2d3d3591-1aaa-49e8-af22-483d0a6b31bc/model/config.json from
OBS
Get object size from OBS failed! errorCode: None, errorMsg: None
Download directory from OBS failed! Exception: (None, None)
Download file from OBS failed! Exception: ('Get object size from OBS failed! ', OBSException(None, None))
Download directory from OBS failed! Exception: ('Download file from OBS failed! ', Exception('Get object size
from OBS failed! ', OBSException(None, None)))
Retry for the 1 th times
```

- 用户创建AI应用时，事件提示：复制模型文件失败，请检查OBS权限是否正常（Failed to copy model file due to obs exception. Please Check your obs access right.）或用户%s没有OBS的obs:object:PutObjectAcl权限（User %s does not have obs:object:PutObjectAcl permission.）。

图 15-43 复制模型文件失败

原因分析

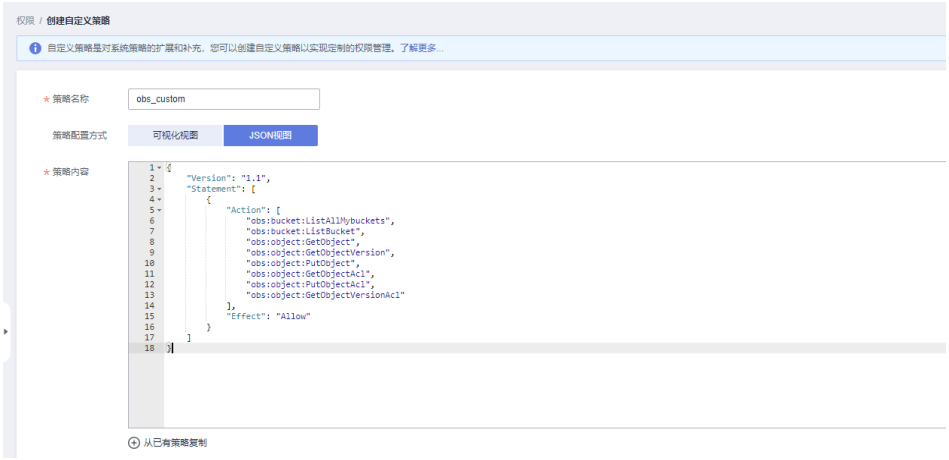
由于ModelArts的使用权限依赖OBS服务的授权，需要为用户授予OBS的系统权限。子用户的IAM权限是由其主用户设置的，如果主用户没有赋予OBS的putObjectAcl权限即会导致创建AI应用构建失败。

处理方法

1. 登录“统一身份认证服务”控制台，左侧菜单选择“权限管理 > 权限”，单击右上角“创建自定义策略”，创建自定义策略权限。

图 15-44 统一身份认证服务添加权限

图 15-45 创建自定义策略



权限内容如下：

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Action": [
            "obs:bucket:ListAllMybuckets",
            "obs:bucket:ListBucket",
            "obs:object:GetObject",
            "obs:object:GetObjectVersion",
            "obs:object:PutObject",
            "obs:object:GetObjectAcl",
            "obs:object:PutObjectAcl",
            "obs:object:GetObjectVersionAcl"
         ],
         "Effect": "Allow"
      }
   ]
}
```

2. 在子用户所属用户组中添加该自定义策略权限。

图 15-46 子用户添加权限

15.5.1.3 创建 AI 应用时，OBS 文件目录对应镜像里面的目录结构是什么样的?

问题现象

创建AI应用时，元模型来源指定的OBS目录下存放了自定义的文件和文件夹，都会拷贝到镜像中去。拷贝进去的路径是什么，怎么读取对应的文件或者文件夹里面的内容?

原因分析

通过OBS导入AI应用时，ModelArts会将指定的OBS目录下的所有文件和文件夹拷贝到镜像中的指定路径下，镜像内路径可以通过self.model_path获取。

处理方法

获取镜像内的路径方法见**模型推理代码编写说明**。

15.5.1.4 通过 OBS 导入 AI 应用时，如何编写打印日志代码才能在 ModelArts 日志查询界面看到日志

问题现象

用户通过OBS导入AI应用时，选择使用基础镜像，用户自己编写了部分推理代码实现自己的推理逻辑，出现故障后希望通过故障日志排查定位故障原因，但是通过logger打印日志无法在“在线服务”的日志中查看到部分内容。

原因分析

推理服务的日志如果需要显示出来，需要代码中将日志打印到Console控制台。当前推理基础镜像使用的python的logging模块，采用的是默认的日志级别Warnning，即当前只有warning级别的日志可以默认查询出来。如果想要指定INFO等级的日志能够查询出来，需要在代码中指定logging的输出日志等级为INFO级别。

处理方法

在推理代码所在的py文件中，指定日志输出到Console的默认级别为INFO级别，确保将对应级别的日志打印出来。参考代码如下:

```
import logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

15.5.1.5 通过 OBS 创建 AI 应用时，构建日志中提示 pip 下载包失败

问题现象

通过OBS创建AI应用构建失败，查看构建日志，提示pip下载包失败。如下载numpy 1.16版本失败。

原因分析

一般下载包失败时，可能有如下几个原因：

1. pip源中不存在该包，当前默认pip源为pypi.org中的包，请在pypi.org中查看是否有对应版本的包并查看包安装限制。
2. 下载的包与对应基础镜像架构不匹配，如arm系统下载了x86的包，python2版本的pip下载了python3的包。具体基础镜像运行环境请参见**推理基础镜像列表**。
3. 安装pip包有先后依赖关系。

处理方法

1. 到pypi.org上查询依赖的待安装包是否存在，如果不存在则建议使用whl包进行安装（将待安装的whl包放到模型所在的OBS目录下）。
2. 查看待安装包的安装限制和前置依赖等，排查是否满足相关要求。
3. 如果包有依赖关系，请参考**导入模型时，模型配置文件中的安装包依赖参数如何编写？**配置包的先后依赖关系。

15.5.1.6 通过自定义镜像创建 AI 应用失败

问题现象

通过用户自定义镜像创建AI应用失败。

原因分析

可能原因如下：

- 导入AI应用使用的镜像地址不合法或实际镜像不存在
- 用户给ModelArts的委托中没有SWR相关操作权限
- 用户为子账号，没有主账号SWR的权限
- 使用的是非自己账号的镜像
- 使用的镜像为公开镜像

处理方法

1. 到SWR检查下对应的镜像是否存在，对应镜像的镜像地址是否和实际地址一致，大小写，拼写等是否一致。
2. 检查用户给ModelArts的委托中是否有SWR的权限，可以在全局配置中查看对应用户的授权内容，查看授权详情。如果没有对应权限，需要到统一身份认证服务给对应委托中加上对应权限。

图 15-47 全局配置

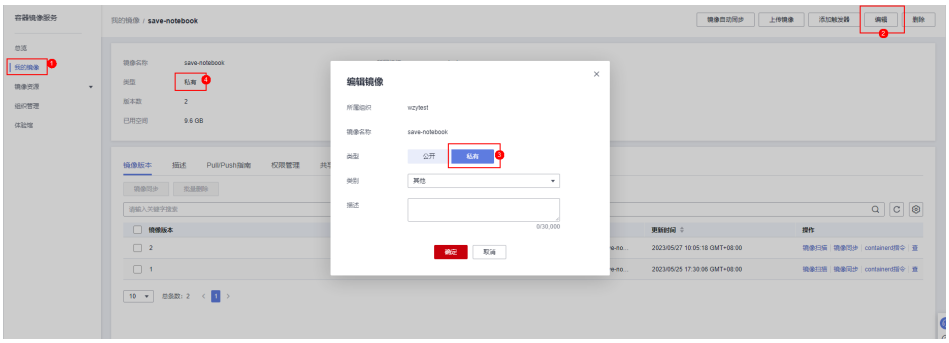
图 15-48 查看权限详情和去 IAM 修改委托权限

图 15-49 给委托添加授权

3. 将镜像设置成私有镜像

   登录容器镜像服务（SWR），左侧导航栏选择“我的镜像”，查看镜像详情，单击右上角“编辑”按钮，把镜像类型修改为“私有”。

图 15-50 修改镜像类型为私有



15.5.1.7 导入 AI 应用后部署服务，提示磁盘不足

问题现象

用户在导入AI应用后，部署服务时，提示磁盘空间不足：“No space left on device”。

原因分析

ModelArts部署使用的是容器化部署，容器运行时有空间大小限制，当用户的模型文件或者其他自定义文件，系统文件超过Docker size大小时，会提示镜像内空间不足。

处理方法

公共资源池容器Docker size的大小最大支持10G，专属资源池Docker size的大小最大支持30G。

如果使用的是OBS导入或者训练导入，则包含基础镜像、模型文件、代码、数据文件和下载安装软件包的大小总和。

如果使用的是自定义镜像导入，则包含解压后镜像和镜像下载文件的大小总和。

15.5.1.8 创建 AI 应用成功后，部署服务报错，如何排查代码问题

问题现象

创建AI应用成功后，部署服务失败，如何定位是模型代码编写有问题。

原因分析

用户自定义镜像或者通过基础镜像导入的AI应用时，用户自己编写了很多自定义的业务逻辑，这些逻辑有问题将会导致服务部署或者预测失败，需要能够排查出哪里有问题。

处理方法

1. 服务部署失败后，进入服务详情界面，查看服务部署日志，明确服务部署失败原因（用户代码输出需要使用标准输入输出函数，否则输出的内容不会呈现到前端页面日志）。根据日志中提示的报错信息找到对应的代码进行定位。

15.5.1.9 自定义镜像导入配置运行时依赖无效

问题现象

通过API接口选择自定义镜像导入创建AI应用，配置了运行时依赖，没有正常安装pip依赖包。

原因分析

自定义镜像导入不支持配置运行时依赖，系统不会自动安装所需要的pip依赖包。

处理方法

重新构建镜像。

在构建镜像的dockerfile文件中安装pip依赖包，例如安装Flask依赖包。

15.5.1.10 通过 API 接口查询 AI 应用详情，model_name 返回值出现乱码

问题现象

通过API接口查询AI应用详情，model_name返回值出现乱码。例如model_name为query_vec_recall_model，但是api接口返回结果是query_vec_recall_model_b。

图 15-51 model_name 乱码

原因分析

当模型名称包含下划线时，下划线涉及转义处理。

处理方法

需要在请求中增加exact_match参数，且参数值设置为true，确保model_name返回值正确。

15.5.1.11 导入 AI 应用提示模型或镜像大小超过限制

问题现象

在导入AI应用时，提示模型或镜像大小超过限制。

原因分析

如果使用的是OBS导入或者训练导入，则是基础镜像、模型文件、代码、数据文件和下载安装软件包的大小总和超过了限制。

如果使用的是自定义镜像导入，则是解压后镜像和镜像下载文件的大小总和超过了限制。

处理方法

精简模型或镜像后，重新导入。

15.5.1.12 导入 AI 应用提示单个模型文件超过 5G 限制

问题现象

在导入AI应用时，提示单个模型文件大小超过5G限制。

原因分析

在不使用动态加载的情况下，系统对单个模型文件的限制大小为5G，超过时无法进行导入。

处理方法

- 精简模型文件后，重新导入。
- 使用动态加载功能进行导入。

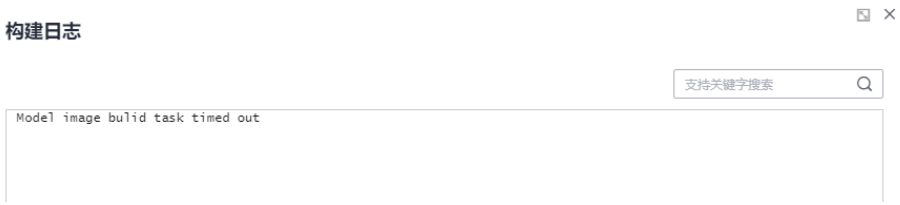
图 15-52 使用动态加载

15.5.1.13 创建 AI 应用失败，提示模型镜像构建任务超时，没有构建日志

问题现象

创建AI应用失败，构建日志提示超时“Model image build task timed out”，没有详细构建日志。

图 15-53 模型镜像构建任务超时



原因分析

imagePacker构建镜像有超时时间限制，默认值为30min（各区域可能存在差异）。当模型镜像构建时间太长，构建日志最后未能完成构建任务，构建超时中断，即会出现“Model image build task timed out”提示，不显示详细的构建日志。

处理方法

- 预先准备需要编译下载的依赖包，减少依赖包下载和编译的时间。可通过线下wheel包方式安装运行环境依赖。线下wheel包安装，需确保wheel包与模型文件放在同一目录。
- 优化模型代码，提高构建模型镜像的编译效率。

15.5.2 服务部署

15.5.2.1 自定义镜像模型部署为在线服务时出现异常

问题现象

在部署在线服务时，部署失败。进入在线服务详情页面，“事件”页签，提示“failed to pull image, retry later”，同时在“日志”页签中，无任何信息。

解决方法

出现此问题现象，通常是因为您部署的模型过大导致的。解决方法如下：

- 精简模型，重新导入模型和部署上线。
- 购买专属资源池，在部署上线为在线服务时，使用专属资源池进行部署。

15.5.2.2 部署的在线服务状态为告警

问题现象

在部署在线服务时，状态显示为“告警”。

解决方法

使用状态为告警的服务进行预测，可能存在预测失败的风险，请从以下4个角度进行排查，并重新部署。

1. 后台预测请求过多。

   如果您使用API接口进行预测，请检查是否预测请求过多。大量的预测请求会导致部署的在线服务进入告警状态。

2. 业务内存不正常。

   请检查推理代码是否存在内存溢出或者内存泄漏的问题。

3. 模型运行异常。

   请检查您的模型是否能正常运行。例如模型依赖的资源是否故障，需要排查推理日志。

4. 实例pod数量异常 。

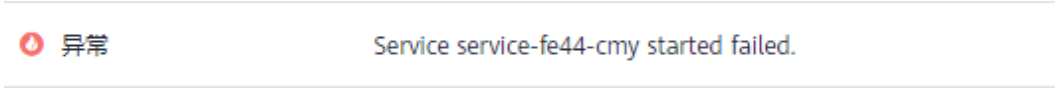
   如果您曾经找过运维人员删除过异常的实例pod，事件中可能会出现告警“服务异常，不正常的实例数为XXX”。在出现这种告警后，服务会自动拉起新的正常实例，从而恢复到正常运行状态。请您耐心等待。

15.5.2.3 服务启动失败

问题现象

当服务事件中出现如下事件时，表示容器启动失败。

图 15-54 服务启动失败



原因分析

服务启动失败的原因比较多样，可能有如下几种情况：

- **AI应用本身问题，无法启动**
- **镜像中配置的端口错误**
- **健康检查配置有问题**
- **模型推理代码customize_service.py编写有问题**
- **镜像拉取失败**
- **资源不足，服务调度失败**

AI 应用本身问题，无法启动

如果创建AI应用使用的镜像本身有问题，需要在创建AI应用之前，参考**从0-1制作自定义镜像并创建AI应用**，确保镜像可以正常启动，并可以在本地curl通返回预期内容。

镜像中配置的端口错误

AI应用可以正常启动，但是因为镜像中启用的端口非8080，或者镜像启用的端口与创建AI应用时配置的端口不一致，导致部署服务时register-agent无法与AI应用通信，超过一定时间后（最长20分钟）认为AI应用启动失败。

需要检查两个地方：自定义镜像中的代码开放的端口和创建AI应用界面上配置的端口。确认两处端口保持一致。AI应用创建界面如果不填端口信息，则ModelArts会默认监听8080端口，即镜像代码中启用的端口必须是8080。

图 15-55 自定义镜像中的代码开放的端口

```
# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

图 15-56 创建 AI 应用界面上配置的端口

健康检查配置有问题

镜像如果配置了健康检查，服务启动失败，从以下两个方面进行排查：

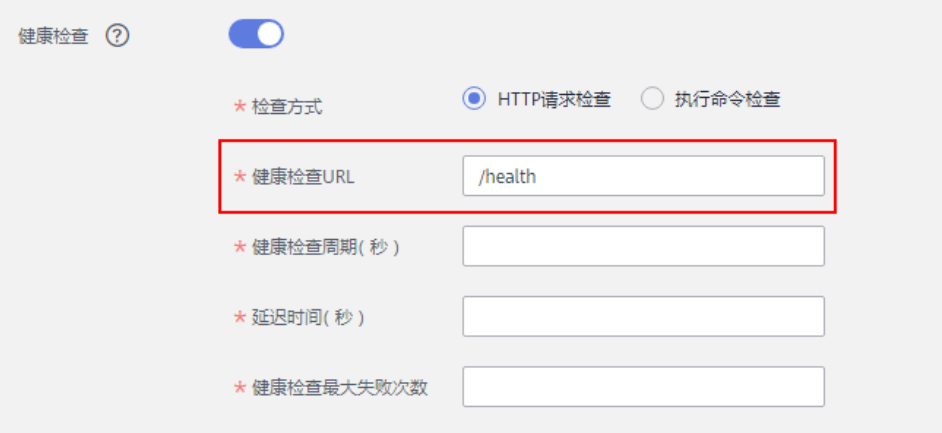
- 健康检查端口是否可以正常工作

  自定义镜像中配置了健康检查，需要在测试镜像时，同步测试健康检查接口是否可以正常工作，具体参考**从0-1制作自定义镜像并创建AI应用**中的本地验证镜像方法。

- 创建AI应用界面上配置的健康检查地址与实际配置的是否一致

  如果使用的是ModelArts提供的基础镜像创建AI应用，健康检查URL默认必须为/health。

图 15-57 设置健康检查 URL



模型推理代码 customize_service.py 编写有问题

如果模型推理代码customize_service.py编写有误，可以通过查看服务运行日志，定位具体原因进行修复。

拉取镜像失败

服务启动失败，提示拉取镜像失败，请参考**服务部署、启动、升级和修改时，拉取镜像失败如何处理?**

资源不足，服务调度失败

服务启动失败，提示资源不足，服务调度失败，请参考**服务部署、启动、升级和修改时，资源不足如何处理?**

内存不足

服务启动失败，提示内存不足，请参考**内存不足如何处理?**

15.5.2.4 服务部署、启动、升级和修改时，拉取镜像失败如何处理?

原因分析

节点磁盘不足，镜像大小过大

解决方法

1. 首先考虑优化镜像，减小节点磁盘的占用。
2. 优化镜像无法解决问题，请联系系统管理员处理。

15.5.2.5 服务部署、启动、升级和修改时，镜像不断重启如何处理?

原因分析

容器镜像代码错误

解决方法

根据容器日志进行排查，修复代码，重新创建AI应用，部署服务。

15.5.2.6 服务部署、启动、升级和修改时，容器健康检查失败如何处理?

原因分析

容器提供的健康检查接口调用失败。容器健康检查接口调用失败，原因可能有两种：

- 镜像健康检查配置问题
- AI应用健康检查配置问题

解决方法

根据容器日志进行排查，查看健康检查接口失败的具体原因。

- 镜像健康检查配置问题，需修复代码后重新制作镜像创建AI应用后部署服务。了解镜像健康接口配置请参考**模型配置文件编写说明**中health参数说明。
- AI应用健康检查配置问题，需重新创建AI应用或者创建AI应用新版本，配置正确的健康检查，使用新的AI应用或版本重新部署服务。了解AI应用健康检查请参考**制作模型镜像并导入**中的“健康检查”参数说明。

15.5.2.7 服务部署、启动、升级和修改时，资源不足如何处理?

问题现象

启动服务失败，报错：资源不足，服务调度失败。（Schedule failed due to insufficient resources. Retry later.或ModelArts.3976：No resources are available for the selected specification.）

图 15-58 资源不足，服务调度失败

原因分析

- 实例配置的规格过大，CPU或者内存剩余资源不足；（"insufficient CPU" / "insufficient memory"）
- 模型需要的磁盘空间大，磁盘空间不足；（"x node(s) had taint {node.kubernetes.io/disk-pressure: }" / "No space"）

解决方法

在遇到资源不足的情况时，ModelArts会进行三次重试，在服务重试期间，如果有资源释放出来，则服务可以正常部署成功。

如果三次重试后依然没有足够的资源，则本次服务部署失败。参考以下方式解决：

- 如果是在公共资源池部署服务，可等待其他用户释放资源后，再进行服务部署。
- 如果是在专属资源池部署服务，在满足模型需求的前提下，尝试选用更小的容器规格或自定义规格，进行服务部署；
- 如果当前资源池的资源确实不够，也可以考虑将资源池扩容后再进行服务部署。公共资源池扩容，请联系系统管理员。专属资源池扩容，可参考**扩缩容资源池**。
- 如果磁盘空间不够，可以尝试重试，使实例调度到其他节点。如果单实例仍磁盘空间不足，请联系系统管理员，更换合适的规格。

说明

如果是大模型导入的AI应用部署服务，请确保专属资源池磁盘空间大于1T（1000GB）。

15.5.2.8 模型使用 CV2 包部署在线服务报错

问题现象

使用CV2包部署在线服务报错

原因分析

使用OBS导入元模型，会用到服务侧的标准镜像，标准镜像里面没有CV2依赖的so的内容。所以ModelArts不支持从对象存储服务（OBS）导入CV2模型包。

处理方法

需要您把CV2包制作为自定义镜像，上传至容器镜像服务（SWR），选择从容器镜像中导入元模型，部署在线服务。如何制作自定义镜像请参考**从0-1制作自定义镜像并创建AI应用**。

15.5.2.9 服务状态一直处于“部署中”

问题现象

服务状态一直处于“部署中”，查看AI应用日志未发现服务有明显错误。

原因分析

一般情况都是AI应用的端口配置有问题。建议您首先检查创建AI应用的端口是否正确。

处理方法

AI应用的端口没有配置，默认为8080，如您在自定义镜像配置文件中修改了端口号，需要在部署AI应用时，配置对应的端口号，使新的AI应用重新部署服务。

如何修改默认端口号，请参考**使用自定义镜像创建在线服务，如何修改默认端口**。

15.5.2.10 服务启动后，状态断断续续处于“告警中”

问题现象

预测流量不大但频繁出现以下报错

- Backend service internal error. Backend service read timed out
- Send the request from gateway to the service failed due to connection refused, please confirm your service is connectable
- Send the request from gateway to the service failed due to connection timeout, please confirm your service is able to process the new request

原因分析

该报错是因为发送预测请求后，服务出现停止后又启动的情况。

处理方法

需要您检查服务使用的镜像，确定服务停止的原因，修复问题。重新创建AI应用部署服务。

15.5.2.11 服务部署失败，报错 No Module named XXX

问题现象

服务部署失败，报错：No Module named XXX

原因分析

No Module named XXX，表示模型中没有导入对应依赖模块。

处理方法

依赖模块没有导入，需要您在模型推理代码中导入缺失依赖模块。

例如您的AI应用是Pytorch框架，部署为在线服务时出现告警：ModuleNotFoundError: No module named ‘model_service.tfserving_model_service’，则需要您在推理代码customize_service.py里使用from model_service.pytorch_model_service import PTServingBaseService。示例代码：

```
import log
from model_service.pytorch_model_service import PTServingBaseService
```

15.5.2.12 批量服务输入/输出 obs 目录不存在或者权限不足

问题现象

1. 输入输出目录不存在，报如下错误
   ```
   "error_code": "ModelArts.3551",
   "error_msg": "OBS path xxxx does not exist."
   ```
2. 当访问目录权限不足时，报如下错误
   ```
   "error_code": "ModelArts.3567",
   "error_msg": "OBS error occurs because Access Denied."
   ```

原因分析

ModelArts.3551：数据输入或者输出的obs目录不存在

ModelArts.3567：使用的数据输入或者输出obs目录存在，但是当前账号无权限访问

处理方法

ModelArts.3551：到obs检查输入数据目录是否存在，如果不存在，请按照实际需要创建obs目录；如果检查发现目录存在，但依然报同样的错，可以提工单申请技术支持

ModelArts.3567：用户只能访问自己账号下的obs目录，ModelArts在读取其他用户obs下的数据时，需要用户委托权限，没有创建委托，就没有权限使用其他用户obs中的数据。

登录ModelArts控制台，管理控制台，在左侧导航栏中选择“全局配置”，单击“查看权限”，检查是否配置了obs的委托权限。

图 15-59 查看权限

如果检查后已经存在委托，但是仍然无法访问，可以提工单寻求技术支持。

15.5.2.13 内存不足如何处理?

问题现象

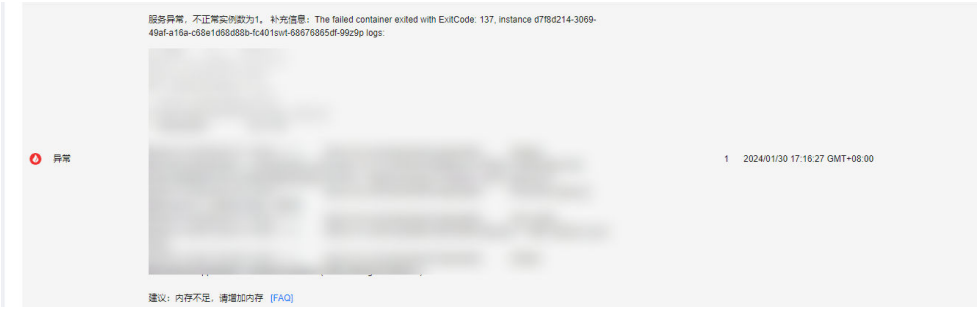
- 在部署或升级在线服务时，如果部署或升级失败，并且在事件中出现如下类似提示。

  图 15-60 内存不足提示样例 1

- 运行中服务出现告警时，在事件中出现建议：内存不足，请增加内存。

图 15-61 内存不足提示样例 2



原因分析

- 部署或升级时出现该提示，可能原因是选择的计算节点规格内存太小，无法满足应用部署，请增大内存规格。
- 运行中服务告警中出现该提示，可能代码有问题导致内存溢出或者业务使用量太大导致内存需求增多。

处理方法

- 在部署或升级在线服务时，选择更大内存规格的计算节点。

  图 15-62 选择计算节点规格

- 运行中服务出现告警时，需要分析是您的代码是否出现漏洞导致内存溢出、是否因为业务使用量太大需要更多的内存。如果因业务原因需要更多内存，请升级在线服务选择更大内存规格的计算节点。

15.5.3 服务预测

15.5.3.1 服务预测失败

问题现象

在线服务部署完成且服务已经处于“运行中”的状态，向服务发起推理请求，预测失败。

原因分析及处理方法

服务预测需要经过客户端、外部网络、APIG、Dispatch、模型服务多个环节。每个环节出现都会导致服务预测失败。

图 15-63 推理服务流程图

1. 出现APIG.XXXX类型的报错，表示请求在APIG（API网关）出现问题而被拦截。
   常见问题请参见**服务预测失败，报错APIG.XXXX**。
   其他被APIG（API网关）拦截的场景：
   - **Method Not Allowed**
   - **请求超时返回Timeout**
2. 出现Modelart.XXXX类型的报错，表示请求在Dispatcher出现问题而被拦截。
   常见报错：
   - **在线服务预测报错ModelArts.4302**
   - **在线服务预测报错ModelArts.4206**
   - **在线服务预测报错ModelArts.4503**
3. 当使用推理的镜像并且出现MR.XXXX类型的错误时，表示已进入模型服务，一般是模型推理代码编写有问题。
   请根据构建日志报错信息，定位服务预测失败原因，修改模型推理代码后，重新导入模型进行预测。
   经典案例：**在线服务预测报错MR.0105**
4. 出现其他情况，优先检查客户端和外部网络是否有问题。
5. 以上方法均未解决问题，请联系系统管理员。

15.5.3.2 服务预测失败，报错 APIG.XXXX

请求在APIG（API网关）出现问题被拦截，报错APIG.XXXX。

常见报错：

- **APIG.0101 预测地址错误**
- **APIG.0201 请求体内容过大**
- **APIG.0301 鉴权失败**

查看更多的APIG（API网关）错误码含义及处理方案可参考API错误码。

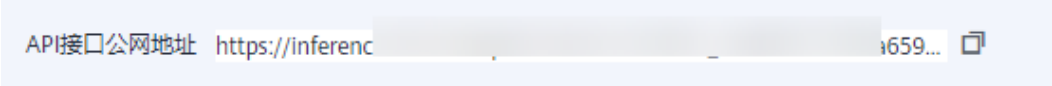
APIG.0101 预测地址错误

当预测的地址有问题时，APIG（API网关）将拦截请求，报错“APIG.0101”：“The API does not exist or has not been published in the environment”，请到在线服务详情界面，“调用指南”页签中获取正确的API接口地址。

说明

如果您在配置文件url中有定义路径，需要在API调用body体中调用路径后拼接自定义路径，例如：您定义url为“/predictions/poetry”，那么在API调用时路径为“{API接口地址}/predictions/poetry”。

图 15-64 获取 API 接口地址



APIG.0201 请求体内容过大

请求体内容过大时，APIG（API网关）会拦截请求，报错“APIG.0201”：“Request entity too large”。请减少预测请求内容后重试。

当使用API调用地址预测时，请求体的大小限制是12MB，超过12MB时，请求会被拦截。

使用ModelArts console的预测页签进行的预测，由于console的网络链路的不同，要求请求体的大小不超过8MB。

图 15-65 请求报错 APIG.0201

APIG.0301 鉴权失败

通过API进行服务预测，或者使用Token进行APP认证，需要获取正确的Token鉴权，当Token不合法时，APIG（API网关）拦截请求，报错“APIG.0301”：“Incorrect IAM authentication information： decrypt token fail”。请获取正确的token填入X-Auth-Token，进行预测。

获取某一区域的Token，需使用此区域的Endpoint，并在获取用户Token的URI部分找到resource-path（**/v3/auth/tokens**），拼接起来如下所示。

```
https://{iam-endpoint}/v3/auth/tokens
```

15.5.3.3 在线服务预测报错 ModelArts.4206

问题现象

在线服务部署完成且服务已经处于“运行中”的状态，向服务发起推理请求，报错“ModelArts.4206”。

原因分析

ModelArts.4206表示该API的请求流量超过了设定值。为了保证服务的平稳运行，ModelArts对单个API的推理请求流量做了限制，同时为了保证推理服务可以稳定运行在合理区间，ModelArts将限流值设定在一个较高区间。

处理办法

降低API的流量，如果确有超高并发的需求，请提工单处理。

15.5.3.4 在线服务预测报错 ModelArts.4302

问题现象

在线服务部署完成且服务已经处于“运行中”的状态后，向运行的服务发起推理请求，报错ModelArts.4302。

原因分析及处理方法

服务预测报错ModelArts.4302有多种场景，以下主要介绍两种场景：

1. "error_msg": "Gateway forwarding error. Failed to invoke backend service due to connection refused. "

   出现该报错有两种情况：

   - 流量超过了模型的处理能力。可以考虑降低流量或者增加模型实例数量。
   - 镜像自身有问题。需要单独运行镜像确保镜像本身能正确提供服务。

2. "error_msg"： "Due to self protection, the backend service is disconncted, please wait moment."

   出现该错误，是因为模型报错太多。当模型报错太多时，会触发dispatcher的熔断机制，导致预测失败。建议您检查模型返回结果，处理模型报错问题，可尝试通过调整请求参数、降低请求流量等方式，提高模型调用的成功率。

15.5.3.5 在线服务预测报错 ModelArts.4503

问题现象

在线服务部署完成且服务已经处于“运行中”的状态后，向运行的服务发起推理请求，报错ModelArts.4503。

原因分析及处理方法

服务预测报错ModelArts.4503有多种场景，常见场景如下：

1. 通信出错

   请求报错：{"error_code":"ModelArts.4503","error_msg":"Failed to respond due to backend service not found or failed to respond"}

   基于高性能考虑，ModelArts会复用同模型服务的连接。根据tcp协议，连接的断开可以由该连接的client端发起，也可以由server端发起。断开连接需要经过四次握手，所以可能会存在作为服务端的模型服务侧发起断开连接，但是该连接正在被作为客户端的ModelArts使用，从而导致通信出错，返回此错误信息。

   如果您使用的是自定义镜像导入的模型，请增大自定义镜像中所使用的web server的keep-alive的参数值，尽量避免由服务端发起关闭连接。如您使用的**Gunicorn**来作为web server，可以通过**Gunicorn**命令的--keep-alive参数来设置该值。其他方式导入的模型，服务内部已做处理。

2. 协议错误

   请求报错：{"error_code":"ModelArts.4503", "error_msg":"Failed to find backend service because SSL error in the backend service, please check the service is https"}

   部署在线服务使用的模型是从容器镜像中导入时，容器调用接口协议填写错误，会导致此错误信息。

   出于安全考虑，ModelArts提供的推理请求都是https请求，从容器镜像中选择导入模型时，ModelArts允许使用的镜像提供https或http服务，但必须在“容器调用接口”中明确指定该镜像使用的是https或http服务。如下图所示：

图 15-66 容器调用接口

如果您在“容器调用接口”中选择的结果跟您镜像实际提供的结果不匹配，例如您在这里选择的是https，但镜像里面实际提供的是http，就会遇到上述错误。反之，如果您选择的是http，但镜像里面实际提供的是https，也会遇到类似错误。

您可以创建一个新的AI应用版本，选择正确的协议（http或者https），重新部署在线服务或更新已有在线服务。

3. 请求预测时间过长

   报错：{"error_code": "ModelArts.4503", "error_msg": "Backend service respond timeout, please confirm your service is able to process the request without timeout. "}及报错：{"error_code": "ModelArts.4503", "error_msg": "Failed to find backend service because response timed out, please confirm your service is able to process the request without timeout. "}

   因APIG（API网关）限制，平台每次请求预测的时间不超过40秒。数据从平台发送到服务，服务预测推理，再将结果返回的时间不超过限制，可以成功返回预测结果。当服务预测的时间过长或者频繁预测导致服务接收不过来请求，即会出现该报错。

   可以通过以下方式解决问题：

   - 服务预测请求内容过大时，会因数据处理慢导致请求超时，优化预测代码，缩短预测时间。
   - 推理速度与模型复杂度强相关，优化模型，缩短预测时间。
   - 扩容实例数或者选择性能更好的“计算节点规格”，例如使用GPU资源代替CPU资源，提升服务处理能力。

4. 服务出错

   报错：{"error_code": "ModelArts.4503","error_msg": "Backend service respond timeout, please confirm your service is able to process the request without timeout. "}

   服务日志输出：

   ```
   [2022-10-24 11:37:31 +0000] [897] [INFO] Booting worker with pid: 897
   [2022-10-24 11:41:47 +0000] [1997] [INFO] Booting worker with pid: 1997
   [2022-10-24 11:41:22 +0000] [1897] [INFO] Booting worker with pid: 1897
   [2022-10-24 11:37:54 +0000] [997] [INFO] Booting worker with pid: 997
   ```

   服务异常进程反复重启导致预测请求无法发送到服务实例。

   可以通过以下方式解决问题：

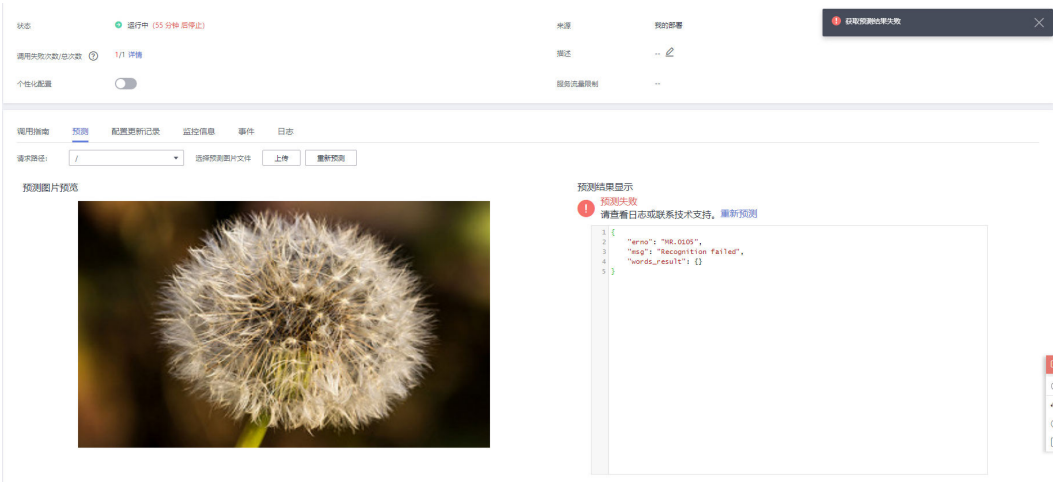
   - 缩小预测请求数量看是否问题还复现，如果不复现是因为负载过大导致服务进程退出，需要扩容实例数量或者提升规格。
   - 推理代码本身存在错误，请排查推理代码解决。

15.5.3.6 在线服务预测报错 MR.0105

问题现象

部署为在线服务，服务处于运行中状态，预测时报错：{ "erno": "MR.0105", "msg": "Recognition failed","words_result": {}}。

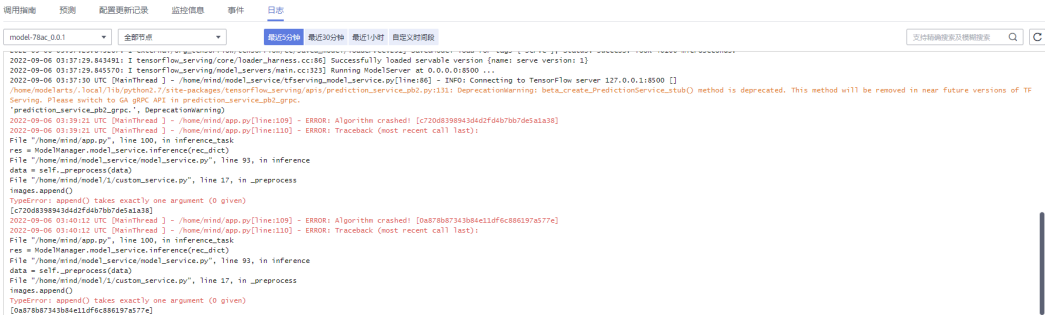
图 15-67 预测报错



原因分析

请在“在线服务”详情页面的日志页签中查看对应的报错日志，分析报错原因。

图 15-68 报错日志



从上图报错日志判断，预测失败是模型推理代码编写有问题。

解决方法

根据日志报错提示，append方法中缺少必填参数，修改模型推理代码文件“customize_service.py”中的代码，给append方法中传入合理的参数。

15.5.3.7 在线服务预测报错 ModelArts.2803

问题现象

服务预测报错：Method Not Allowed

原因分析

服务预测默认注册的API需要使用POST方法调用。如您使用了GET方法，APIG（API网关）将会拦截请求。

处理方法

使用POST方法调用。

15.5.3.8 请求超时返回 Timeout

问题现象

服务预测请求超时，报错{"error_code": "ModelArts.4205","error_msg":"Connection time out."}

原因分析

请求超时，大概率是APIG（API网关）拦截问题。需排查APIG（API网关）和模型。

处理方法

1. 优先排查APIG（API网关）是否是通的，可以在本地使用curl命令排查，命令行：**curl -kv {预测地址}**。如返回Timeout则需排查本地防火墙，代理和网络配置。
2. 检查模型是否启动成功或者模型处理单个消息的时长。因APIG（API网关）的限制，模型单次预测的时间不能超过40S，超过后系统会默认返回Timeout错误。

15.5.3.9 自定义镜像导入模型部署上线调用 API 报错

部署上线调用API报错，排查项如下：

1. 确认配置文件模型的接口定义中有没有POST方法。
2. 确认配置文件里url是否有定义路径。例如：“/predictions/poetry”（默认为“/”）。
3. 确认API调用中body体中的调用路径是否拼接自定义路径。如：“{API接口地址}/predictions/poetry”。

15.6 MoXing

15.6.1 使用 MoXing 复制数据报错

问题现象

1. 调用**moxing.file.copy_parallel()**将文件从开发环境的OBS桶中复制到其他OBS桶里，但是桶内没有出现目标文件。
2. 使用MoXing复制数据不成功，出现报错。如：
   - ModelArts开发环境使用MoXing复制OBS数据报错：keyError: 'request-id'
   - ModelArts使用MoXing复制报错：No files to copy
   - socket.gaierror: [Errno -2] Name or service not known
   - ERROR:root:Failed to call:

     func=<bound method ObsClient.getObject of <obs.client.ObsClient object at 0x7fd705939710>>

     args=('bucket', 'data/TFRecord/HY_all_inside/no_adjust_light_3/09_06_6x128x128_0000000212.tfrecord')
3. 使用MoXing复制数据报错，提示超时。如：
   - 报错：TimeoutError: [Errno 110] Connection timed out
   - WARNING:root:Retry=9,Wait=0.1, Timestamp = 1567152567.5327423

原因分析

当使用MoXing复制数据不成功，可能原因如下：

- 源文件不存在。
- OBS路径不正确或者是两个OBS路径不在同一个区域。
- 训练作业空间不足。

处理方法

按照报错提示，需要排查以下几个问题：

1. 检查**moxing.file.copy_parallel()**的第一个参数中是否有文件，否则会出现报错：No files to copy
   - 文件存在，请执行**2**。
   - 文件不存在，请忽略该报错继续执行后续操作。
2. 检查复制的OBS的路径是否与开发环境或训练作业在同一个区域。

   进入ModelArts管理控制台，查看其所在区域。然后再进入OBS管理控制台，查看您使用的OBS桶所在的区域。查看是否在同一区域。
   - 是，请执行**3**。
   - 否，请在ModelArts同一区域的OBS中新建桶和文件夹，并将所需的数据上传至此OBS桶中。
3. 检查OBS的路径是否正确，是否写为了“obs://xxx”。可使用如下方式判断OBS路径是否存在。

   mox.file.exists('obs://bucket_name/sub_dir_0/sub_dir_1')
   - 路径存在，请执行**4**。
   - 路径不存在，请在更换为一个可用的OBS路径。

4. 检查使用的资源是否为CPU，CPU的“/cache”与代码目录共用10G，可能是空间不足导致，可在代码中使用如下命令查看磁盘大小。

   os.system('df -hT')

   - 磁盘空间满足，请执行**5**。
   - 磁盘空间不足，请您使用GPU资源。

5. 如果是在Notebook使用MoXing复制数据不成功，可以在Terminal界面中使用**df -hT**命令查看空间大小，排查是否因空间不足导致，可在创建Notebook时使用EVS挂载。

如果代码写作正确，仍然无法解决该问题，请提交工单，由专业工程师为您分析并解决问题。

15.6.2 如何关闭 Mox 的 warmup

问题现象

训练作业mox的Tensorflow版本在运行的时候，会先执行“50steps” 4次，然后才会开始正式运行。

warmup即先用一个小的学习率训练几个epoch（warmup），由于网络的参数是随机初始化的，如果一开始就采用较大的学习率会出现数值不稳定的问题，这是使用warmup的原因。等到训练过程基本稳定之后就可以使用原先设定的初始学习率进行训练。

原因分析

Tensorflow分布式有多种执行模式，mox会通过4次执行50 step记录执行时间，选择执行时间最少的模型。

处理方法

创建训练作业时，在“运行参数”中增加参数“variable_update=parameter_server”来关闭Mox的warmup。

15.6.3 Pytorch Mox 日志反复输出

问题现象

ModelArts训练作业算法来源选用常用框架的Pytorch引擎，在训练作业运行时Pytorch Mox日志会每个epoch都打印Mox版本，具体日志如下：

```
INFO:root:Using MoXing-v1.13.0-de803ac9
INFO:root:Using OBS-Python-SDK-3.1.2
INFO:root:Using MoXing-v1.13.0-de803ac9
INFO:root:Using OBS-Python-SDK-3.1.2
```

原因分析

Pytorch通过spawn模式创建了多个进程，每个进程会调用多进程方式使用Mox下载数据。此时子进程会不断销毁重建，Mox也就会不断的被导入，导致打印很多Mox的版本信息。

处理方法

为避免训练作业Pytorch Mox日志反复输出的问题，需要您在“启动文件”中添加如下代码，当“MOX_SILENT_MODE = “1””时，可在日志中屏蔽mox的版本信息：

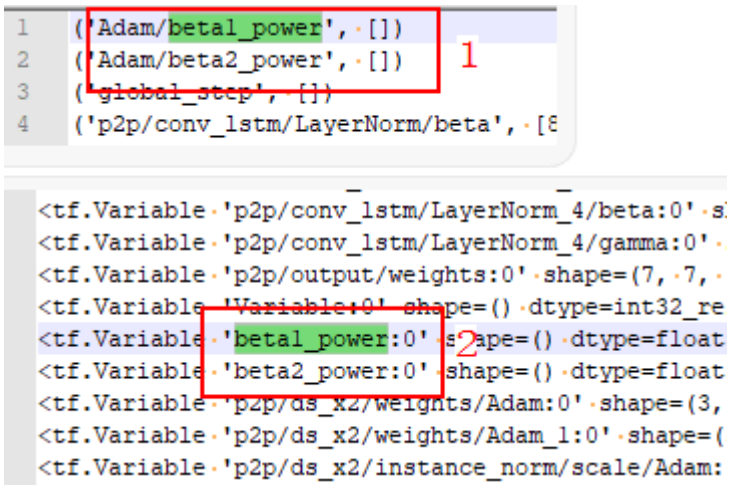
```
import os
os.environ["MOX_SILENT_MODE"] = "1"
```

15.6.4 moxing.tensorflow 是否包含整个 TensorFlow，如何对生成的 checkpoint 进行本地 Fine Tune?

问题现象

使用MoXing训练模型，“global_step”放在Adam名称范围下，而非MoXing代码中没有Adam名称范围，如图15-69所示。其中1为使用MoXing代码，2代表非MoXing代码。

图 15-69 代码示例



处理方法

Fine Tune就是用别人训练好的模型，加上自己的数据，来训练新的模型。相当于使用别人的模型的前几层，来提取浅层特征，然后在最后再落入我们自己的分类中。

由于一般新训练模型准确率都会从很低的值开始慢慢上升，但是Fine Tune能够让我们在比较少的迭代次数之后得到一个比较好的效果。Fine Tune的好处在于不用完全重新训练模型，从而提高效率，在数据量不是很大的情况下，Fine Tune会是一个比较好的选择。

moxing.tensorflow包含所有的接口，对TensorFlow做了优化，里面的实际接口还是TensorFlow的原生接口。

当非MoXing代码中没有Adam名称范围时，需要修改非MoXing代码，在其中增加如下内容：

```
with tf.variable_scope("Adam"):
```

在增加代码时不建议使用自定义“global_step”，推荐使用**tf.train.get_or_create_global_step()**。

15.6.5 训练作业使用 MoXing 拷贝数据较慢，重复打印日志

问题现象

- ModelArts训练作业使用MoXing拷贝数据较慢。
- 重复打印日志“INFO:root:Listing OBS”。

图 15-70 重复打印日志

```
INFO:root:Listing OBS: 77000
INFO:root:Listing OBS: 78000
INFO:root:Listing OBS: 79000
INFO:root:Listing OBS: 80000
INFO:root:Listing OBS: 81000
INFO:root:Listing OBS: 82000
INFO:root:Listing OBS: 83000
INFO:root:Listing OBS: 84000
INFO:root:Listing OBS: 85000
INFO:root:Listing OBS: 86000
INFO:root:Listing OBS: 87000
INFO:root:Listing OBS: 88000
INFO:root:Listing OBS: 89000
```

原因分析

1. 拷贝数据慢的可能原因如下：
   - 直接从OBS上读数据会造成读数据变成训练的瓶颈，导致迭代缓慢。
   - 由于环境或网络问题，读OBS时遇到读取数据失败情况，从而导致整个作业失败。
2. 重复打印日志，该日志表示正在读取远端存在的文件，当文件列表读取完成以后，开始下载数据。如果文件比较多，那么该过程会消耗较长时间。

处理方法

在创建训练作业时，数据可以保存到OBS上。不建议使用TensorFlow、MXNet、PyTorch的OBS接口直接从OBS上读取数据。

- 如果文件较小，可以将OBS上的数据保存成“.tar”包。训练开始时从OBS上下载到“/cache”目录，解压以后使用。
- 如果文件较大，可以保存成多个“.tar”包，在入口脚本中调用多进程进行并行解压数据。不建议把散文件保存到OBS上，这样会导致下载数据很慢。
- 在训练作业中，使用如下代码进行“.tar”包解压：

```
import moxing as mox
import os
mox.file.copy_parallel("obs://donotdel-modelarts-test/AI/data/PyTorch-1.0.1/tiny-imagenet-200.tar", '/cache/tiny-imagenet-200.tar')
os.system('cd /cache; tar -xvf tiny-imagenet-200.tar > /dev/null 2>&1')
```

15.6.6 MoXing 如何访问文件夹并使用 get_size 读取文件夹大小?

问题现象

- 使用MoXing无法访问文件夹。
- 使用MoXing的“get_size”读取文件夹大小，显示为0。

原因分析

使用MoXing访问文件夹，需添加参数：“recursive=True”，默认为False。

处理方法

获取一个OBS文件夹的大小：

```
import moxing as mox
mox.file.get_size('obs://bucket_name/sub_dir_0/sub_dir_1', recursive=True)
```

获取一个OBS文件的大小：

```
import moxing as mox
mox.file.get_size('obs://bucket_name/obs_file.txt')
```

15.7 API/SDK

15.7.1 安装 ModelArts SDK 报错“ERROR: Could not install packages due to an OSError”

问题现象

安装ModelArts SDK报错，完整报错信息“ERROR: Could not install packages due to an OSError: [WinError 2] The system cannot find the file specified: 'c:\python39\Scripts\ephemeral-port-reserve.exe' -> 'c:\python39\Scripts\ephemeral-port-reserve.exe.deleteme ”。

原因分析

用户使用权限问题导致。

处理方法

用户电脑切换到管理员角色，键盘快捷键（Windows+R模式）并输入cmd，进入黑色窗口，执行如下命令：

```
python -m pip install --upgrade pip
```

15.7.2 ModelArts SDK 下载文件目标路径设置为文件名，部署服务时报错

问题现象

ModelArts SDK在OBS下载文件时，目标路径设置为文件名，在本地IDE运行不报错，部署为在线服务时报错。

代码如下：

```
session.obs.download_file（ obs_path, local_path ）
```

报错信息如下：

```
2022-07-06 16:22:36 CST [ThreadPoolEx] - /home/work/predict/model/customize_service.py[line:184] - WARNING: 4 try: IsADirectoryError(21, 'Is a directory'). update products failed!
```

原因分析

用户代码中设置的目标路径（ local_path ）有误。

处理方法

需要将local_path路径设置为文件夹且后缀必须以“/”结尾。

15.7.3 调用 API 创建训练作业，训练作业异常

问题现象

调用API接口创建训练作业（ 专属资源池为CPU规格 ），训练作业状态由“创建中”转变为“异常”，训练作业详情界面“规格信息”为“--”。

原因分析

调用接口传入了CPU规格的专属资源池不支持的参数。

处理步骤

检查API请求的请求体中是否有“flavor_id”参数，CPU规格的专属资源池不支持使用“flavor_id”参数。

16 修订记录

发布日期	修订记录
2024-04-30	第一次正式发布。