

ModelArts

使用自定义镜像

文档版本 01
发布日期 2022-06-20



版权所有 © 华为技术有限公司 2022。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1 自定义镜像简介	1
2 Notebook 中使用自定义镜像	3
2.1 Notebook 使用自定义镜像介绍	3
2.2 将 Notebook 实例保存为自定义镜像	3
2.2.1 使用预置镜像创建 Notebook 实例	4
2.2.2 在 Notebook 中安装外部库	5
2.2.3 如何保存 Notebook 镜像环境	6
2.2.4 基于新的镜像创建 Notebook 实例	8
2.3 通过编写 Dockfile 构建自定义镜像	8
2.3.1 使用场景和构建流程说明	8
2.3.2 Step1 制作自定义镜像	9
2.3.3 Step2 调试新镜像	11
2.3.4 Step3 注册新镜像	11
2.3.5 Step4 创建开发环境并使用	12
2.4 参考文件	14
2.4.1 ma-cli 命令介绍	14
2.4.2 Dockerfile 文件信息	20
2.4.3 创建 ECS 并配置 VM 信息	21
3 使用自定义镜像训练模型（新版训练）	22
3.1 准备训练镜像	22
3.1.1 训练作业自定义镜像规范	22
3.1.2 基于预置镜像创建新镜像	23
3.1.2.1 使用预置镜像构建新的训练镜像	23
3.1.2.2 预置镜像详情（PyTorch）	25
3.1.2.3 预置镜像详情（TensorFlow）	26
3.1.2.4 预置镜像详情（Horovod）	28
3.1.2.5 预置镜像详情（MPI）	30
3.1.2.6 预置镜像详情（Ascend-Powered-Engine）	31
3.1.3 已有镜像如何适配迁移至 ModelArts	33
3.2 使用自定义镜像创建算法	34
3.3 使用自定义镜像创建训练作业（CPU/GPU）	40
3.4 使用自定义镜像创建训练作业（Ascend）	43

3.5 示例：从 0 到 1 制作自定义镜像并创建训练作业（ CPU/GPU ）	44
4 使用自定义镜像创建 AI 应用.....	51
4.1 创建 AI 应用的自定义镜像规范.....	51
4.2 使用自定义镜像创建的 AI 应用部署服务.....	52
5 FAQ.....	57
5.1 如何登录并上传镜像到 SWR.....	57

1 自定义镜像简介

ModelArts为用户提供了多种常见的预置镜像，但是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。

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。

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。您制作的自定义镜像需要上传至SWR服务。

图 1-1 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时时存在与OBS的数据交互，您需要使用的数据可以存储至OBS服务。

- 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，若没有授权，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](#)。

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。

2 Notebook 中使用自定义镜像

2.1 Notebook 使用自定义镜像介绍

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的进行依赖管理，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，这些镜像有以下特点：

1. 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，提供最优的软件、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优。
2. 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。
3. 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。

当预置镜像不能满足需求时，用户可以制作自定义镜像。

制作自定义镜像有两种方式：

- 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。
- 方式二，通过镜像构建的方式，基于ModelArts提供的基础镜像，在ECS服务器上自行编写Dockerfile构建镜像，并将镜像推送到SWR，作为自定义镜像使用。制作好的自定义镜像可以用来创建新的开发环境或者训练作业。

说明

用户制作的自定义镜像，必须基于ModelArts提供的基础镜像进行构建，采用其他来源的基础镜像制作的自定义镜像暂不支持在Notebook中使用。

2.2 将 Notebook 实例保存为自定义镜像

2.2.1 使用预置镜像创建 Notebook 实例

预置镜像使用场景

ModelArts开发环境给用户提供了预置镜像，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，比如用户开发基于MindSpore1.5，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。

预置镜像功能

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MindSpore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，分享案例到AI Gallery进行交流，停止实例（实例停止后CPU、Memory不再计费）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。
- ModelArts开发环境提供的预置镜像支持功能开发后，直接提到ModelArts训练作业中进行训练。

ModelArts提供的预置镜像是以ma-user启动的，用户进入实例后，工作目录默认是/home/ma-user。

```
sh-4.4$pwd
/home/ma-user
sh-4.4$
```


创建实例，持久化存储挂载放路径为/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，重新启动后是依然保留，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。

如何使用预置镜像

创建Notebook实例时选择预置镜像，创建成功，打开Notebook即可使用。

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”新版管理页面。
2. 单击“创建”，进入“创建Notebook”页面，选择公共镜像，选择资源、规格等参数，并提交。详细参数请参见[创建Notebook实例](#)。
3. Notebook实例状态为“运行中”时，即可以打开Notebook使用创建的镜像。

2.2.2 在 Notebook 中安装外部库

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 2-1 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	<code>conda env list</code> #显示所有的虚拟环境 <code>conda info -e</code> #显示所有的虚拟环境 <code>conda create -n myenv python=3.7</code> #创建一个名为myenv环境，指定Python版本是3.7 <code>conda activate myenv</code> #激活名为myenv的环境 <code>conda deactivate</code> #关闭当前环境 <code>conda remove -n myenv --all</code> #删除一个名为myenv的环境 <code>conda create -n newname --clone oldname</code> #克隆oldname环境为newname环境
package 管理	<code>conda list</code> #查看当前环境下已安装的package <code>conda list -n myenv</code> #指定myenv环境下安装的package <code>conda search numpy</code> #查找名为numpy的package的所有信息 <code>conda search numpy=1.12.0 --info</code> #查看版本为1.12.0的numpy的信息 <code>conda install numpy pandas</code> #安装numpy和pandas两个package，此命令可同时安装一个或多个包 <code>conda install numpy=1.12.0</code> #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 <code>conda install -n myenv numpy</code> #在myenv的环境中安装名字为numpy的package <code>conda install -c https://conda.anaconda.org/anaconda numpy</code> #使用源 https://conda.anaconda.org/anaconda 安装numpy <code>conda update numpy pandas</code> #更新numpy和pandas两个package，此命令可同时更新一个或多个包 <code>conda remove numpy pandas</code> #卸载numpy和pandas两个package，此命令可同时卸载一个或多个包 <code>conda update --all</code> #更新当前环境下所有的package
清理 conda	<code>conda clean -p</code> # 删除无用的包 <code>conda clean -t</code> # 删除压缩包 <code>conda clean -y --all</code> # 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[如何保存Notebook镜像环境](#)。

2.2.3 如何保存 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（pip包）不丢失，持久化存储的部分不会保存在最终产生的容器镜像中。VSCode远程开发场景下，在Server端安装的插件不丢失。

前提条件

Notebook实例状态必须为“运行中”才可以一键进行镜像保存。

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。

图 2-1 保存镜像



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“确认”保存镜像。

图 2-2 保存 Notebook 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。创建组织的详细操作请参见[创建组织](#)。

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（对于打开的JupyterLab界面和本地IDE 仍可操作）。

图 2-3 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。

2.2.4 基于新的镜像创建 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。

当前保存下来的镜像可以用于Notebook实例创建。详细操作请参见[创建Notebook实例](#)。

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 2-4 创建基于自定义镜像的 Notebook 实例

< 创建Notebook

* 名称: notebook-2bd6

描述: 0/256

* 自动停止: ☒ 1 小时 ☐ 2 小时 ☐ 4 小时 ☐ 6 小时 ☐ 自定义

* 镜像:

名称	版本
<input checked="" type="radio"/> jupyter-notebook-2.1	

2.3 通过编写 Dockfile 构建自定义镜像

2.3.1 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，这一点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PyTorch基础镜像，构建一个面向AI开发的新环境，并借助ma_cli(请参考[ma-cli介绍](#))调试和注册，确保可以成功创建新的开发环境。主要流程如下图所示：

图 2-5 构建与调测镜像流程



2.3.2 Step1 制作自定义镜像

这一节描述如何编写一个Dockerfile，并据此构建出一个新镜像。关于Dockerfile的具体编写方法，请参考[官网](#)。

步骤1 准备一台具有docker功能的机器，如果没有，建议申请一台弹性云服务器，并在准备好的机器上安装必要的软件。具体请参考[创建ECS并配置VM信息](#)。后续的构建、调试和注册操作都在此机器中进行。

步骤2 查询基础镜像

首先配置鉴权信息，账号、用户名、密码和局点。更多信息请查看[配置登录信息](#)。

```
ma-cli configure
```

执行后会提示用户输入信息，请注意输入的password信息会被隐藏。

图 2-6 配置鉴权信息

```
root@djy-ubuntu1804-cpu:~/dj# ma-cli configure
account: ***
username [***]:
password:
region(1:Beijing4, 2:Shanghai1, 3:Guangzhou): 1
save password(y/n): y
The information is saved in /root/.modelarts/config.json
```

然后使用ma-cli get-image [OPTIONS]命令查询当前账号可访问的所有镜像。该命令的更多信息请参考[查询镜像](#)。

```
ma-cli get-image -o wide
```

查询结果如下图：

图 2-7 查询基础镜像方式一

	name	id	swr_path	arch	service	resource_category	visibility	description
1	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	e1a07296-22a8-4f85-8bcb-e936c8e54099	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PUBLIC	CPU and GPU general algorithm development and training, preconfigured with AI engine PyTorch1.4
2	tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04	e1a07296-22a8-4f85-8bcb-e936c8e54100	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-tensorflow-2.1-kernel-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PUBLIC	CPU and GPU general algorithm development and training, preconfigured with AI engine TensorFlow2.1
3	mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04	89de30ec-6871-4f22-84af-be37ef28335d	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-gpu-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU']	PUBLIC	GPU algorithm development and training, preconfigured with the AI engine MindSpore-CPU
4	mindspore1.2.0-openmpi2.1.1-ubuntu18.04	65f636a0-56cf-49df-b941-7d2a07ba0c8c	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-cpu-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['CPU']	PUBLIC	CPU algorithm development and training, preconfigured with the AI engine MindSpore-CPU
5	mlstudio-pyspark2.4.5-ubuntu18.04	0b308728-4681-118c-994f-081a76da7111	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mlstudio-cp37:3.3.3-release-v1-20220120	x86_64	['NOTEBOOK']	['CPU']	PUBLIC	CPU algorithm development and training, including the MLStudio tool for graphical ML algorithm development, and preconfigured PySpark 2.4.5
6	mlstudio-pyspark2.3.2-ubuntu18.04	0e5f9a41-c9c2-4d9a-a198-4e1b17a7782f	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mlstudio-cp36:3.3.1.9	x86_64	['NOTEBOOK']	['CPU']	PUBLIC	CPU algorithm development and training, including the MLStudio tool for graphical ML algorithm development, and preconfigured PySpark 2.3.2
7	pytorch1.4	47df4d7e-35c2-4415-8b5b-9de7baf4b301	swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch1.4:1.0.1	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PRIVATE	
8	modelbox1.2.1-tensorrt7.1.3-pytorch1.9.1-cuda10.2-cudnn8-euler2.9.6	e1a07296-22a8-4f85-8bcb-e936c8e54101	swr.cn-north-4.myhuaweicloud.com/atelier/modelarts-aiflow-gpu-x86:1.1.2.1-20220117093753-1161a24	x86_64	['SSH', 'MODELBOX']	['GPU']	PUBLIC	AI Inference application development, preconfigured ModelBox, AI engine PyTorch, TensorRT and TensorFlow, only SSH connection supported.
9	mindspore1.2.0-openmpi2.1.1-ubuntu18.04	d7f53555-9845-4deb-94cc-4833e1e02728	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-gpu-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU']	PUBLIC	GPU algorithm development and training, preconfigured with the AI engine MindSpore-CPU
10	cyip0.91.4-cbcp2.10-ortools9.0-cplex20.1.0-ubuntu18.04	b9933af6-3119-4845-a427-5e6e0327dafa	swr.cn-north-4.myhuaweicloud.com/r1-dev/or-cpu:1.0.8	x86_64	['NOTEBOOK', 'SSH']	['CPU']	PUBLIC	CPU operations research development, preconfigured with cyip, cbcp, ortools, cplex(community).

上述命令展示了镜像的所有信息，但是swr_path字段不方便复制，使用以下命令只展示name和swr_path。

```
ma-cli get-image
```

结果如下图：

图 2-8 查询基础镜像方式二

	name	swr_path
1	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114
2	tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-tensorflow-2.1-kernel-cp37:3.3.3-release-v1-20220114
3	mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-gpu-cp37:3.3.3-release-v1-20220114
4	mindspore1.2.0-openmpi2.1.1-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-cpu-cp37:3.3.3-release-v1-20220114
5	mlstudio-pyspark2.3.2-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mlstudio-cp36:3.3.1.9
6	mlstudio-pyspark2.4.5-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mlstudio-cp37:3.3.3-release-v1-20220120
7	pytorch1.4	swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch1.4:1.0.1
8	modelbox1.2.1-tensorrt7.1.3-pytorch1.9.1-cuda10.2-cudnn8-euler2.9.6	swr.cn-north-4.myhuaweicloud.com/atelier/modelarts-aiflow-gpu-x86:1.1.2.1-20220117093753-1161a24
9	mindstudio1.0.2-ascend910-cann5.0.2.1-ubuntu18.04-aarch64	swr.cn-north-4.myhuaweicloud.com/atelier/mindstudio-modelarts-image:3.0.2.5
10	cyip0.91.4-cbcp2.10-ortools9.0-cplex20.1.0-ubuntu18.04	swr.cn-north-4.myhuaweicloud.com/r1-dev/or-cpu:1.0.8

步骤3 制作新镜像

1. 拉取基础镜像

拉取[步骤3 查询基础镜像](#)中查询到的镜像或ModelArts提供的公共镜像（请参考[预置镜像](#)）。

```
docker pull swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114
```

也可以拉取其他的镜像，但如果在图2-7中，'visibility'字段是'PRIVATE'，则首先要执行SWR登录命令，再进行拉取。如何登录请参考[登录SWR](#)。

2. 编写Dockerfile

本例的Dockerfile将基于PyTorch基础镜像安装pytorch 1.8, ffmpeg 3和gcc 8，构建一个面向AI任务的镜像。

Dockerfile的具体内容可参考[Dockerfile示例](#)。关于如何编写Dockerfile，请参考[官网](#)。

3. 构建镜像

使用docker build命令从Dockerfile构建出一个新镜像。命令参数解释如下：

- “-t” 指定了新的镜像地址，建议使用完整的swr地址，因为后续的调试和注册需要使用。
- “-f ” 指定了Dockerfile的文件名，根据实际填写。
- 最后的 “.” 指定了构建的上下文是当前目录，根据实际填写。

```
docker build -t swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2 -f Dockerfile .
```

----结束

2.3.3 Step2 调试新镜像

新镜像调试操作

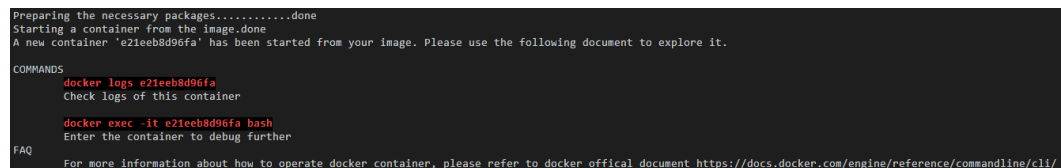
使用ma-cli debug-image [OPTIONS] SWR_PATH来调试镜像。了解ma-cli更多信息请参考[ma-cli介绍](#)。

```
ma-cli debug-image swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2 -s NOTEBOOK
```

该命令调试镜像swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2，-s参数指定该镜像提供Notebook服务。

命令执行后，首先会停止上一次调试相同镜像时已经启动的容器，再进行一些准备工作，最后运行目标镜像，启动一个新容器，并打印相关指令，帮助用户进一步调试新镜像，输出信息如下图所示。

图 2-9 调试镜像



该命令的详细信息请参考[调试镜像](#)。

2.3.4 Step3 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，进而在能够在ModelArts中使用该镜像。了解ma-cli更多信息请参考[ma-cli介绍](#)。

1. 将镜像推到SWR

推送前需要登录SWR，请参考[登录SWR](#)。登录后使用docker push命令进行推送，如下：

```
docker push swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2
```

完成后即可在SWR上看到该镜像。

图 2-10 将镜像推到 SWR



2. 注册镜像

有两种方式来注册镜像。

- 方式一：使用`ma-cli register-image [OPTIONS] SWR_PATH`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 register-image swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2
```

图 2-11 注册镜像

```
Successfully registered this image and image information is
{
  "arch": "x86_64",
  "create_at": 1643184173431,
  "dev_services": [
    "NOTEBOOK",
    "SSH"
  ],
  "id": "961550d2-84b5-4169-ba01-0c89629b7f87",
  "name": "pytorch_1_8",
  "namespace": "sdk-test",
  "origin": "CUSTOMIZE",
  "resource_categories": [
    "GPU",
    "CPU"
  ],
  "size": 9713885893,
  "status": "ACTIVE",
  "swr_path": "swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2",
  "tag": "v2",
  "type": "DEDICATED",
  "update_at": 1643184173431,
  "visibility": "PRIVATE",
  "workspace_id": "0"
}
```

- 方式二：在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。单击“注册镜像”，镜像源即为[1.将镜像推到SWR](#)中推送到SWR中的镜像，将完整的SWR地址拷贝到这里即可。

图 2-12 注册镜像

< 注册镜像

* 镜像源 查看可选镜像
示例: swr.<region>.myhuaweicloud.com/<namespace>/<repository>:<tag>

描述
0/256

* 架构 ☒ X86_64 ☐ ARM

* 类型 ☒ CPU ☐ GPU ☐ Ascend

2.3.5 Step4 创建开发环境并使用

创建开发环境

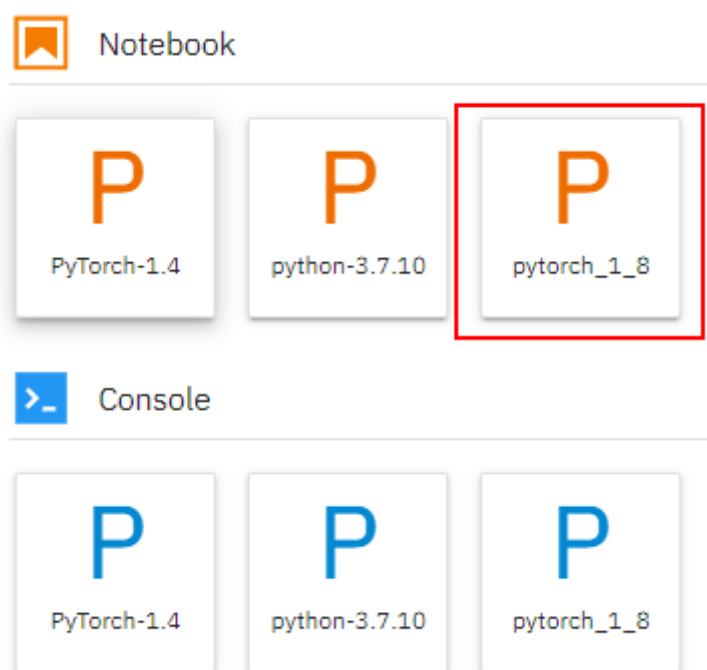
1. 镜像注册成功后，即可在ModelArts控制台的Notebook页面，创建开发环境时选择该自定义镜像。

图 2-13 创建开发环境



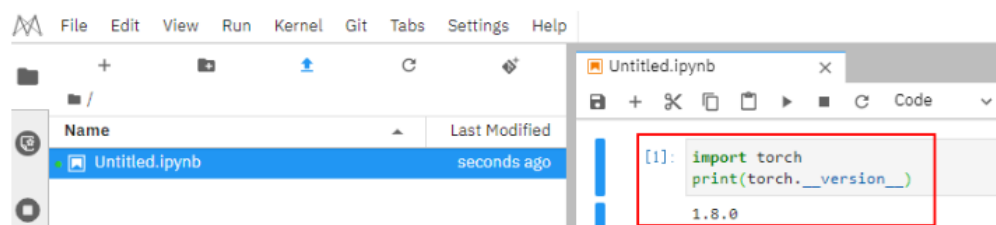
2. 打开开发环境，即可看到Dockerfile中创建的conda环境pytorch_1_8。

图 2-14 打开开发环境



3. 点击图中的pytorch_1_8，即可创建一个ipynb文件，导入torch，可以看到安装的pytorch 1.8已经能够使用。

图 2-15 创建一个 ipynb 文件



4. 再打开一个terminal，查看ffmpeg和gcc的版本，是Dockerfile中安装的版本。

图 2-16 查看 ffmpeg 和 gcc 的版本

```
[ma-user work]$ffmpeg -v
ffmpeg version 3.4.8-0ubuntu0.2 Copyright (c) 2000-2020 the FFmpeg developers
  built with gcc 7 (Ubuntu 7.5.0-3ubuntu1~18.04)
  configuration: --prefix=/usr --extra-version=0ubuntu0.2 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/
stripping --enable-avresample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --enable-libs2b --en
ibflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-lib
openmpt --enable-libopus --enable-libpulse --enable-librubberband --enable-libsrt --enable-libshine --enable-libsnap --enable-libsoxr
ble-libtheora --enable-libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libx
nable-libzbi --enable-omx --enable-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enabl
ibopencv --enable-libx264 --enable-shared
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter    6.107.100 / 6.107.100
  libavresample  3.  7.  0 / 3.  7.  0
  libswscale     4.  8.100 / 4.  8.100
  libswresample  2.  9.100 / 2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Missing argument for option '-v'
Error splitting the argument list: Invalid argument
[ma-user work]$gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/8/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 8.4.0-1ubuntu1~18.04' --with-bugurl=file:///usr/share/doc/gcc-8/README.Bu
d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-8 --program-prefix=x86_64-linux-gnu- --enable-share
/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale=gnu --enable-libstdc++-debug
lt-libstdc++-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --enable-plugin --enable-default-pie --with-syst
-enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch=32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-m
load-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64
Thread model: posix
gcc version 8.4.0 (Ubuntu 8.4.0-1ubuntu1~18.04)
```

2.4 参考文件

2.4.1 ma-cli 命令介绍

ma-cli是modelarts服务提供的一套命令行工具，它本身包含很多内容，这里介绍自定义镜像构建部分。ma-cli提供了5个命令，分别为：

- ma-cli configure，配置用户登录信息，用以操作ModelArts资源；
- ma-cli get-image，查询用户可访问的镜像，包括modelarts提供的公共镜像和用户自己的私有镜像；
- ma-cli debug-image，调试镜像，仿照ModelArts开发环境的方式运行用户的镜像，发现镜像中可能存在的问题；
- ma-cli register-image，注册镜像，将用户本地的镜像注册到modelarts中，注册后镜像可用于创建ModelArts的开发环境；
- ma-cli unregister-image，取消注册，将注册的镜像从modelarts中删除。

使用上述5个命令，用户可使用自己的镜像创建ModelArts的开发环境。

配置用户登录信息 ma-cli configure

用户填写IAM用户信息。关于IAM请参考[IAM基本概念](#)，要确认填写信息的正确性，请先使用IAM用户登录的方式登录。

```
ma-cli configure
```

执行后，会提示填写以下信息：

图 2-17 配置登录信息

```
root@djy-ubuntu1804-cpu:~/dj# ma-cli configure
account: ***
username [***]:
password:
region(1:Beijing4, 2:Shanghai1, 3:Guangzhou): 1
save password(y/n): y
The information is saved in /root/.modelarts/config.json
```

表 2-2 登录信息

选项	含义
account	IAM用户的账号。
username	IAM用户的用户名，不填则默认与account相同。
password	密码。请注意，password这一项的输入值会被隐藏。
region	局点信息，填入界面对应数值即可。
save password(y/n)	是否保存密码。 y: 加密后保存。 n: 不保存。

填写完成后，信息会被保存在\$HOME/.modelarts/config.json中，文件内容如下：

```
{
  "cn-north-4": {
    "auths": {
      "account": "****",
      "username": "****",
      "password": "****"
    }
  },
  "current region": "cn-north-4"
}
```

若save password(y/n)选择了n，则password不会被保存。

支持登录多个局点，但每个局点只支持一个账号，登录多局点时文件内容如下：

```
{
  "cn-south-1": {
    "auths": {
      "account": "****",
      "username": "****",
      "password": "****"
    }
  },
  "current region": "cn-north-4",
  "cn-north-4": {
    "auths": {
      "account": "****",
      "username": "****"
    }
  },
  "cn-east-3": {
```

```
"auths": {  
  "account": "****",  
  "username": "****",  
  "password": "****"  
}
```

请注意"current region"这一项，代表了最近一次登录的局点。其他命令默认会访问该局点。如果要访问别的局点，请在命令中使用-r 参数指定。

查询镜像 ma-cli get-image

查询用户拥有的镜像。支持两种查询方式：根据镜像类型查询和根据镜像名称查询。可使用ma-cli get-image -h获取帮助信息。

- 根据镜像类型查询
ma-cli get-image -t BUILD_IN -o wide

查询结果如下图：

图 2-18 根据镜像类型查询

	name	id	swr_path	arch	service	resource_category	visibility	description
1	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	e1a07296-22a8-4f05-8bc8-e936c8e54099	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PUBLIC	CPU and GPU general algorithm development and training, preconfigured with AI engine PyTorch1.4
2	tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04	e1a07296-22a8-4f05-8bc8-e936c8e54100	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-tensorflow-2.1-kernel-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PUBLIC	CPU and GPU general algorithm development and training, preconfigured with AI engine TensorFlow2.1
3	mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04	80de38ec-6871-4f22-8daf-bc7a7e72835d	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-gpu-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU']	PUBLIC	GPU algorithm development and training, preconfigured with the AI engine MindSpore-GPU
4	mindspore1.2.0-openeuler2.1-ubuntu18.04	65f636a8-56cf-40df-b941-7d2a07ba8c8c	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mindspore-kernel-cpu-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['CPU']	PUBLIC	CPU algorithm development and training, preconfigured with the AI engine MindSpore-CPU
5	mlstudio-pyspark2.4.5-ubuntu18.04	0b2d0728-4c01-11ec-994f-081a7d6a7111	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mlstudio/notebook2.0-mlstudio-cp37:3.3.3-release-v1-20220120	x86_64	['NOTEBOOK']	['CPU']	PUBLIC	CPU algorithm development and training, including the MLStudio tool for graphical ML algorithm development, and preconfigured PySpark 2.4.5
6	mlstudio-pyspark2.3.2-ubuntu18.04	0e5f9a41-c9c2-4d9a-a190-4e1b17a7782f	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-mlstudio-cp36:3.3.1.9	x86_64	['NOTEBOOK']	['CPU']	PUBLIC	CPU algorithm development and training, including the MLStudio tool for graphical ML algorithm development, and preconfigured PySpark 2.3.2
7	modelbox1.1.2.1-tensorrt7.1.3-pytorch1.9.1-cuda10.2-cudnn8-euler2.5.6	e1a07296-22a8-4f05-8bc8-e936c8e54101	swr.cn-north-4.myhuaweicloud.com/atelier/modelarts-ai-flow-gpu-x86:1.1.2.1-20220117093753-1161a24	x86_64	['SSH', 'MODELBOX']	['GPU']	PUBLIC	AI inference application development, preconfigured ModelBox, AI engine PyTorch, TensorRT and TensorFlow, only SSH connection supported.
8	notebook2.0-ml-kernel-cpu-cp36	e1a07296-22a8-4f05-8bc8-e936c8e54092	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-ml-kernel-gpu-cp36:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU']	PUBLIC	notebook2.0 gpu
9	cyip88.91.4-ccpy2.10-ortools9.0-cplex20.1.0-ubuntu18.04	b9931af0-3119-4045-a427-5e668327dafd	swr.cn-north-4.myhuaweicloud.com/r1-dev/or-cpu:1.0.8	x86_64	['NOTEBOOK', 'SSH']	['CPU']	PUBLIC	CPU operations research development, preconfigured with cyp, cbcp, ortools, cplex(community).
10	pytorch1.8-cuda10.2-cudnn7-ubuntu18.04	278e88d1-5b71-4766-8502-b3ba72e82469	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.8-kernel-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PUBLIC	CPU and GPU general algorithm development and training, preconfigured with AI engine PyTorch1.8

- 根据镜像名称查询
ma-cli get-image -n pytorch1.4-cuda10.1-cudnn7-ubuntu18.04 -o wide

查询结果如下图：

图 2-19 根据镜像名称查询

	name	id	swr_path	arch	service	resource_category	visibility	description
1	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	e1a07296-22a8-4f05-8bc8-e936c8e54099	swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114	x86_64	['NOTEBOOK', 'SSH']	['GPU', 'CPU']	PUBLIC	CPU and GPU general algorithm development and training, preconfigured with AI engine PyTorch1.4

命令支持的选项如下表：

表 2-3 查询镜像支持的选项

选项	含义
--type, -t	镜像类型，有3个可选值。 <ul style="list-style-type: none">• BUILD_IN：内置镜像，即ModelArts团队提供的公共镜像；• DEDICATED：用户注册的镜像；• ALL：所有类型的镜像。默认为ALL。
--name, -n	镜像名称。
--region, -r	指定要查询哪个region，默认由\$HOME/.modelarts/config.json的"current region"指定。 支持的值：1 (北京4)，2(上海1)，3(华南广州)。
--password, -p	密码。如果登录时未保存密码，则需要指定。
--output, -o	指定如何展示查询的结果，目前支持： <ul style="list-style-type: none">• default：使用表格展示两个字段，name和swr_path，默认为default。• wide：使用表格展示8个字段，如下表所示。

查询后打印的表格中各字段的含义如下：

表 2-4 返回值各字段的含义

名称	含义
name	镜像名称。
id	镜像id。
swr_path	镜像的SWR地址。
arch	镜像支持的架构，值为X86_64或AARCH64。
service	镜像支持的服务，值为NOTEBOOK，MODELBOX或SSH。
resource_category	镜像支持的计算资源，值为CPU、GPU或ASCEND。
visibility	镜像可见性，值为PUBLIC（所有用户可见）、PRIVATE（仅自己可见）。
description	镜像描述。

调试镜像 ma-cli debug-image

使用ma-cli debug-image [OPTIONS] SWR_PATH来调试镜像。SWR_PATH是一个完整的swr路径。可使用ma-cli debug-image -h获取帮助信息。

```
ma-cli debug-image <swr_path> -s NOTEBOOK
```

表 2-5 镜像调试选项说明

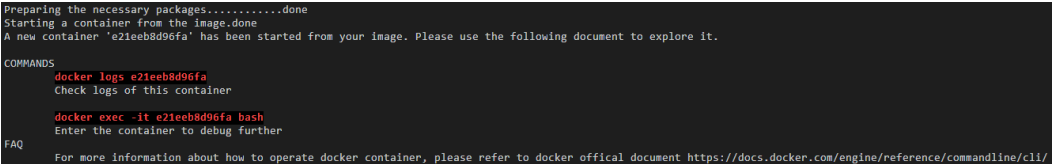
选项	含义
--service, -s	该镜像提供的服务，目前支持两种类型：NOTEBOOK和MODELBOX。 该参数可以输入多个值，如-s NOTEBOOK -s MODELBOX。默认为NOTEBOOK。
--arch, -a	镜像支持的架构，可选值为X86_64和AARCH64。默认为X86_64。
--region, -r	指定要调试哪个region的镜像，默认由\$HOME/.modelarts/config.json的"current region"指定。 支持的值：1（北京4），2（上海1），3（华南广州）。
--gpus, -g	为启动的容器挂载gpu。

命令执行后，用户的镜像会被运行作为一个容器，运行流程和ModelArts上实际的流程接近，所以调试通过后，该镜像在ModelArts服务中也能够正常使用。

命令执行过程为：首先停止上次调试时已经启动的容器，然后进行准备工作，最后运行该镜像。

容器启动成功后，会打印常用命令，帮助用户进一步调试自定义镜像，如下图所示。

图 2-20 容器启动成功



注册镜像 ma-cli register-image

使用命令ma-cli register-image [OPTIONS] SWR_PATH将镜像注册到ModelArts服务中。可使用ma-cli register-image -h获取帮助信息。

```
ma-cli register-image <swr_path> -a X86_64 -d "This image contains pytorch_1_8 and ffmpeg 3" -s NOTEBOOK -rs CPU -r 2
```

表 2-6

选项	含义
--arch, -a	镜像支持的架构，可选值为X86_64和AARCH64。默认为X86_64。
--description, -d	镜像描述，最长512个字符。默认为空字符串。

选项	含义
--service, -s	镜像支持的服务，可选值为NOTEBOOK和MODELBOX。 可以输入多个值，如-s NOTEBOOK -s MODELBOX。默认为NOTEBOOK。
--resource_category, -rs	镜像支持的计算资源，可填写多个，支持以下几种选择： <ul style="list-style-type: none">• -rs CPU• -rs CPU -rs GPU• -rs CPU -rs ASCEND 默认为-rs CPU -rs GPU
--visibility, -v	镜像注册后的可见性，可选值为PUBLIC（所有用户可见）、PRIVATE（仅自己可见）。默认为PRIVATE。
--workspace_id, -wi	工作空间的id，默认为0。
--password, -p	密码。如果登录时未保存密码，则需要指定。
--region, -r	指定要将镜像注册到哪个region，默认由\$HOME/.modelarts/config.json的"current region"指定。 支持的值：1（北京4），2（上海1），3（华南广州）。

注册成功后，会返回一些参数信息，样例如下：

图 2-21 注册成功

```
Successfully registered this image and image information is
{
  "arch": "x86_64",
  "create_at": 1643184173431,
  "dev_services": [
    "NOTEBOOK",
    "SSH"
  ],
  "id": "961550d2-84b5-4169-ba01-0c89629b7f87",
  "name": "pytorch_1_8",
  "namespace": "sdk-test",
  "origin": "CUSTOMIZE",
  "resource_categories": [
    "GPU",
    "CPU"
  ],
  "size": 9713885893,
  "status": "ACTIVE",
  "swr_path": "swr.cn-north-4.myhuaweicloud.com/sdk-test/pytorch_1_8:v2",
  "tag": "v2",
  "type": "DEDICATED",
  "update_at": 1643184173431,
  "visibility": "PRIVATE",
  "workspace_id": "0"
}
```

取消注册 ma-cli unregister-image

使用命令ma-cli unregister-image [OPTIONS] IMAGE_ID来取消注册。取消后在ModelArts的镜像管理页面查询不到该镜像。可使用ma-cli unregister-image -h获取帮助信息。

```
ma-cli unregister-image <IMAGE ID> --delete_swr_image
```

表 2-7 取消注册选项说明

选项	含义
--delete_swr_image, -d	标志位，在取消注册的同时是否在SWR中也删除该镜像。
--password, -p	密码。如果登录时未保存密码，则需要指定。
--region, -r	指定要取消哪个region的镜像。默认由\$HOME/.modelarts/config.json的"current region"指定。 支持的值：1 (北京4)，2(上海1)，3(华南广州)。

2.4.2 Dockerfile 文件信息

Dockerfile的具体内容如下：

```
FROM swr.cn-north-4.myhuaweicloud.com/atelier/notebook2.0-pytorch-1.4-kernel-cp37:3.3.3-release-v1-20220114

USER root
# section1: config apt source
RUN mv /etc/apt/sources.list /etc/apt/sources.list.bak && \
    echo -e "deb http://repo.huaweicloud.com/ubuntu/ bionic main restricted\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-updates main restricted\ndeb http://repo.huaweicloud.com/ubuntu/ bionic universe\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-updates universe\ndeb http://repo.huaweicloud.com/ubuntu/ bionic multiverse\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-updates multiverse\ndeb http://repo.huaweicloud.com/ubuntu/ bionic-backports main restricted universe multiverse\ndeb http://repo.huaweicloud.com/ubuntu bionic-security main restricted\ndeb http://repo.huaweicloud.com/ubuntu bionic-security universe\ndeb http://repo.huaweicloud.com/ubuntu bionic-security multiverse" > /etc/apt/sources.list && \
    apt-get update
# section2: install ffmpeg and gcc
RUN apt-get -y install ffmpeg && \
    apt -y install gcc-8 g++-8 && \
    update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 80 --slave /usr/bin/g++ g++ /usr/bin/g++-8 && \
    rm $HOME/.pip/pip.conf
USER ma-user
# section3: configure conda source and pip source
RUN echo -e "channels:\n - defaults\nshow_channel_urls: true\ndefault_channels:\n - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main\n - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2\ncustom_channels:\n conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n menpo: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n pytorch-lts: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud\n simpleitk: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud" > $HOME/.condarc && \
    echo -e "[global]\nindex-url = https://pypi.tuna.tsinghua.edu.cn/simple\n[install]\ntrusted-host = https://pypi.tuna.tsinghua.edu.cn" > $HOME/.pip/pip.conf
# section4: create a conda environment(only support python=3.7) and install pytorch1.8
RUN source /home/ma-user/anaconda3/bin/activate && \
```



```
conda create -y --name pytorch_1_8 python=3.7 && \  
conda activate pytorch_1_8 && \  
pip install torch==1.8.0 torchvision==0.9.0 torchaudio==0.8.0 && \  
conda deactivate
```

2.4.3 创建 ECS 并配置 VM 信息

创建 ECS 服务器

登录ECS控制台，购买弹性云服务器，镜像选择公共镜像，推荐使用ubuntu18.04的镜像。

图 2-22 选择公共镜像



配置 VM 环境

1. 在ECS中，使用如下命令下载安装脚本。
`wget https://cn-north-4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/modelarts/custom-image-build/install_on_ubuntu1804.sh`
2. 在ECS中并执行如下命令，即可完成环境配置。
`bash install_on_ubuntu1804.sh`

安装脚本依次执行了如下任务：

- a. 安装docker。
- b. 如果ECS挂载了GPU，则会安装nvidia-docker2，用以将GPU挂载到docker容器中。
- c. 安装ModelArts工具，包括modelarts-sdk和ma-cli。

3 使用自定义镜像训练模型（新版训练）

3.1 准备训练镜像

3.1.1 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。

说明

新版训练管理和旧版训练管理均支持使用自定义镜像创建训练作业。本手册介绍的是新版训练功能，旧版训练即将下线，推荐用户使用新版训练。

规范要求

- 自定义镜像大小不能超过30GB，建议不要超过15GB。镜像大小过大，会直接影响训练作业启动时间。
- 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，上述命令回显示如下

```
"Architecture": "arm64"
```

- 规格中带有 ARM 字样的显示，为 ARM CPU 架构。ARM CPU 架构规格示意如下

★ 规格

Ascend: 1*Ascend 910(32GB) | ARM: 24 核 96GB 3200GB

- 规格中未带有 ARM 字样的显示，为 X86 CPU 架构。X86 CPU 架构规格示意如下

★ 规格

GPU: 1*NVIDIA-V100(32GB) | CPU: 8 核 64GB 3200GB

- 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。

训练自定义镜像的制作流程

- 使用弹性云服务器或者本地主机搭建Docker环境。
- 基于上述规范要求和实际需要编写Dockerfile文件，构建自定义镜像。
如何高效编写Dockerfile指导可参考[SWR服务最佳实践](#)。
- 当完成自定义镜像制作后，请将镜像上传到自己的SWR中。
具体请参考[如何登录并上传镜像到SWR](#)。

3.1.2 基于预置镜像创建新镜像

3.1.2.1 使用预置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预置镜像，镜像里已经安装好运行任务所需软件。当预置镜像里的软件无法满足您的程序运行需求时，您可以基于预置镜像进行自定义镜像的制作来减少工作量。

预置镜像列表

ModelArts平台中预置的训练镜像如下表所示。

表 3-1 预置镜像列表

引擎类型	版本名称
PyTorch	pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
	pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64
Tensorflow	tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64
	tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64
	tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64
Horovod	horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64
	horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
MPI	mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64
Ascend-Powered-Engine	tensorflow_1.15.0-cann_5.0.4-py_3.7-euler_2.8.3-aarch64
	mindspore_1.6.1-cann_5.0.4-py_3.7-euler_2.8.3-aarch64

引擎类型	版本名称
	pytorch_1.5.0-cann_5.0.4-py_3.7-euler_2.8.3-aarch64

预置镜像支持的Region有：

- cn-north-1
- cn-north-2
- cn-north-4
- cn-north-9
- cn-east-3
- cn-south-1
- la-south-2
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3

基于预置镜像构建新镜像的操作步骤

您可以参考如下步骤基于预置镜像来构建新镜像。

1. 安装Docker。

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
```

如果**docker images**命令可以执行成功，表示Docker已安装，此步骤可跳过。

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，并放置在 context 文件夹内。

以华为开源镜像站提供的 pip 源为例，在华为开源镜像站<https://mirrors.huaweicloud.com/home>中，搜索 pypi，获取pip.conf 文件内容。pip.conf 文件内容如下。

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

4. 参考如下 Dockerfile 文件内容来基于预置镜像构建新镜像。将编写好的 Dockerfile 文件放置在 context 文件夹内。

```
FROM {预置镜像地址}

# 使用华为开源镜像站提供的 pypi 配置
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
```

3.1.2.2 预置镜像详情（PyTorch）

介绍预置的PyTorch镜像详情。

引擎版本一：pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/pytorch_1_8:train-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-roma-20220309171256-40adcc1
- 镜像构建时间：20220309171256 (yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS
- cuda：10.2.89
- cudnn：7.6.5.32
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/PyTorch-1.8/bin/python, python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages
- 部分三方包版本信息列表：

```
Cython 0.27.3
dask 2022.2.0
easydict 1.9
enum34 1.1.10
torch 1.8.0
Flask 1.1.1
grpcio 1.44.0
gunicorn 20.1.0
idna 3.3
torchtext 0.5.0
imageio 2.16.0
imgaug 0.4.0
lxml 4.8.0
matplotlib 3.5.1
torchvision 0.9.0
mmdcv 1.2.7
numba 0.47.0
numpy 1.21.5
opencv-python 4.1.2.30
toml 0.10.2
pandas 1.1.5
Pillow 9.0.1
pip 21.2.2
protobuf 3.19.4
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 6.0
requests 2.27.1
scikit-image 0.19.2
...
```
- 历史版本：无

引擎版本二：pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/pytorch_1_8:pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18
- 镜像构建时间：20220524162601(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS
- cuda：11.1.1
- cudnn：8.0.5.39
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/PyTorch-1.8.2/bin/python，python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/PyTorch-1.8.2/lib/python3.7/site-packages
- 部分三方包版本信息列表：

Cython 0.27.3
mmcv 1.2.7
easydict 1.9
tensorboardX 2.0
torch 1.8.2+cu111
Flask 2.0.1
pandas 1.1.5
gunicorn 20.1.0
PyYAML 5.1
torchaudio 0.8.2
Pillow 9.0.1
psutil 5.8.0
lxml 4.8.0
matplotlib 3.5.1
torchvision 0.9.2+cu111
pip 21.2.2
protobuf 3.20.1
numpy 1.17.0
opencv-python 4.1.2.30
scikit-learn 0.22.1
...
- 历史版本：无

3.1.2.3 预置镜像详情（TensorFlow）

介绍预置的TensorFlow镜像详情。

引擎版本一：tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/tensorflow_1_15:tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64-20220524162601-50d6a18
- 镜像构建时间：20220524162601(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 20.04.4 LTS
- cuda：11.4.3
- cudnn：8.2.4.15
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/TensorFlow-1.15.5/bin/python，python 3.8.13
- 三方包安装路径：/home/ma-user/anaconda3/envs/TensorFlow-1.15.5/lib/python3.8/site-packages

- 部分三方包版本信息列表：

```
Cython 0.29.21
psutil 5.9.0
matplotlib 3.5.1
protobuf 3.20.1
tensorflow 1.15.5+nv
Flask 2.0.1
grpcio 1.46.1
gunicorn 20.1.0
Pillow 9.0.1
tensorboard 1.15.0
PyYAML 6.0
pip 22.0.4
lxml 4.7.1
numpy 1.18.5
tensorflow-estimator 1.15.1
...
```

- 历史版本：无

引擎版本二：tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/tensorflow_2_1:train-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20210912152543-1e0838d
- 镜像构建时间：20210912152543(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS
- cuda：10.1.243
- cudnn：7.6.5.32
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/TensorFlow-2.1/bin/python， python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/TensorFlow-2.1/lib/python3.7/site-packages
- 部分三方包版本信息列表：

```
Cython 0.29.21
dask 2021.9.0
easydict 1.9
enum34 1.1.10
tensorflow 2.1.0
Flask 1.1.1
grpcio 1.40.0
gunicorn 20.1.0
idna 3.2
tensorflow-estimator 2.1.0
imageio 2.9.0
imgaug 0.4.0
lxml 4.6.3
matplotlib 3.4.3
termcolor 1.1.0
scikit-image 0.18.3
numba 0.47.0
numpy 1.17.0
opencv-python 4.1.2.30
tiff file 2021.8.30
pandas 1.1.5
Pillow 6.2.0
pip 21.0.1
protobuf 3.17.3
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 5.1
```

```
requests 2.26.0
```

```
...
```

- 历史版本：无

引擎版本三：tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/tensorflow_2.6:tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18
- 镜像构建时间：20220524162601(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS
- cuda：11.2.0
- cudnn：8.1.1.33
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/TensorFlow-2.6.0/bin/python，python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/TensorFlow-2.6.0/lib/python3.7/site-packages
- 部分三方包版本信息列表：

```
Cython 0.29.21
requests 2.27.1
easydict 1.9
tensorboardX 2.0
tensorflow 2.6.0
Flask 2.0.1
grpcio 1.46.1
unicorn 20.1.0
idna 3.3
tensorflow-estimator 2.9.0
pandas 1.1.5
Pillow 9.0.1
lxml 4.8.0
matplotlib 3.5.1
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 5.1
numpy 1.17.0
opencv-python 4.1.2.30
protobuf 3.20.1
pip 21.2.2
...
```
- 历史版本：无

3.1.2.4 预置镜像详情（Horovod）

介绍预置的Horovod镜像详情。

引擎版本一：horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/horovod_tensorflow:train-horovod_0.20.0-tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20210912152543-1e0838d
- 镜像构建时间：20210912152543(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS

- cuda: 10.1.243
- cudnn: 7.6.5.32
- Python解释器路径及版本: /home/ma-user/anaconda3/envs/horovod_0.20.0-tensorflow_2.1.0/bin/python, python 3.7.10
- 三方包安装路径: /home/ma-user/anaconda3/envs/horovod_0.20.0-tensorflow_2.1.0/lib/python3.7/site-packages
- 部分三方包版本信息列表:

```
Cython 0.29.21
dask 2021.9.0
easydict 1.9
enum34 1.1.10
horovod 0.20.0
Flask 1.1.1
grpcio 1.40.0
gunicorn 20.1.0
idna 3.2
tensorboard 2.1.1
imageio 2.9.0
imgaug 0.4.0
lxml 4.6.3
matplotlib 3.4.3
tensorflow-gpu 2.1.0
tensorboardX 2.0
numba 0.47.0
numpy 1.17.0
opencv-python 4.1.2.30
toml 0.10.2
pandas 1.1.5
Pillow 6.2.0
pip 21.0.1
protobuf 3.17.3
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 5.1
requests 2.26.0
scikit-image 0.18.3
...
```
- 历史版本: 无

引擎版本二: horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64

- 镜像地址: swr.{region}.myhuaweicloud.com/aip/horovod_pytorch:train-horovod_0.22.1-pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20210912152543-1e0838d
- 镜像构建时间: 20210912152543(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本: Ubuntu 18.04.4 LTS
- cuda: 11.1.1
- cudnn: 8.0.5.39
- Python解释器路径及版本: /home/ma-user/anaconda3/envs/horovod-0.22.1-pytorch-1.8.0/bin/python, python 3.7.10
- 三方包安装路径: /home/ma-user/anaconda3/envs/horovod-0.22.1-pytorch-1.8.0/lib/python3.7/site-packages
- 部分三方包版本信息列表:

```
Cython 0.27.3
dask 2021.9.0
```

```
easydict 1.9
enum34 1.1.10
horovod 0.22.1
Flask 1.1.1
grpcio 1.40.0
gunicorn 20.1.0
idna 3.2
mmcv 1.2.7
imageio 2.9.0
imgaug 0.4.0
lxml 4.6.3
matplotlib 3.4.3
torch 1.8.0
tensorboardX 2.0
numba 0.47.0
numpy 1.17.0
opencv-python 4.1.2.30
torchtext 0.5.0
pandas 1.1.5
Pillow 6.2.0
pip 21.0.1
protobuf 3.17.3
scikit-learn 0.22.1
psutil 5.8.0
PyYAML 5.1
requests 2.26.0
scikit-image 0.18.3
torchvision 0.9.0
...
```

- 历史版本：无

3.1.2.5 预置镜像详情（MPI）

介绍预置的mindspore_1.3.0镜像详情。

引擎版本：mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_1804-x86_64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/mindspore_1_3_0:train-mindspore_1.3.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-roma-20211104202338-f258e59
- 镜像构建时间：20211104202338(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS
- cuda：10.1.243
- cudnn：7.6.5.32
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/MindSpore-1.3.0-gpu/bin/python，python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/MindSpore-1.3.0-gpu/lib/python3.7/site-packages
- 部分三方包版本信息列表：

```
requests 2.26.0
dask 2021.9.0
easydict 1.9
enum34 1.1.10
mindspore-gpu 1.3.0
Flask 1.1.1
grpcio 1.41.1
gunicorn 20.1.0
idna 3.3
PyYAML 5.1
imageio 2.10.1
```

```
imgaug 0.4.0  
lxml 4.6.4  
matplotlib 3.4.2  
psutil 5.8.0  
scikit-image 0.18.3  
numba 0.47.0  
numpy 1.17.0  
opencv-python 4.5.2.54  
tiffle 2021.11.2  
pandas 1.1.5  
Pillow 8.4.0  
pip 21.0.1  
protobuf 3.17.3  
scikit-learn 0.22.1  
...
```

- 历史版本：无

3.1.2.6 预置镜像详情（Ascend-Powered-Engine）

引擎版本一：tensorflow_1.15.0-cann_5.0.4-py_3.7-euler_2.8.3-aarch64

- 镜像地址：
swr.{region}.myhuaweicloud.com/aip/
tensorflow_1_15_ascend:tensorflow_1.15-cann_5.0.4-py_3.7-euler_2.8.3-aarch64-d910-roma-20220318164813-b3feb87
- 镜像构建时间：20220318164813(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：euleros 2.8.3
- cann：5.0.4
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/
TensorFlow-1.15.0/bin/python， python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/TensorFlow-1.15.0/lib/
python3.7/site-packages

- 部分三方包版本信息列表：

```
Cython 0.29.14  
dask 2.18.1  
hccl 0.1.0  
tensorflow 1.15.0  
tensorboard 1.15.0  
Flask 1.1.1  
grpcio 1.26.0  
gunicorn 20.0.4  
idna 2.10  
PyYAML 5.3.1  
imageio 2.9.0  
imgaug 0.2.6  
lxml 4.4.2  
matplotlib 3.1.2  
psutil 5.7.0  
scikit-image 0.17.2  
numba 0.47.0  
numpy 1.17.5  
opencv-python 4.2.0.34  
requests 2.27.1  
pandas 0.24.2  
Pillow 7.0.0  
pip 21.3.1  
protobuf 3.11.3  
scikit-learn 0.20.0  
...
```

- 历史版本：无

引擎版本二：mindspore_1.6.1-cann_5.0.4-py_3.7-euler_2.8.3-aarch64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/
mindspore_1_6_1:mindspore_1.6.1-cann_5.0.4-py_3.7-euler_2.8.3-aarch64-d910-roma-20220318164813-b3feb87
- 镜像构建时间：20220318164813(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：euleros 2.8.3
- cann：5.0.4
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/MindSpore/bin/python， python 3.7.10
- 三方包安装路径：/home/ma-user/anaconda3/envs/MindSpore/lib/python3.7/site-packages
- 部分三方包版本信息列表：
Cython 0.29.14
dask 2.18.1
hccl 0.1.0
mmcv 0.2.14
npu-bridge 1.15.0
Flask 2.0.1
grpcio 1.44.0
gunicorn 20.0.4
idna 2.10
PyYAML 5.3.1
imageio 2.9.0
imgaug 0.2.6
lxml 4.4.2
matplotlib 3.2.1
psutil 5.7.0
scikit-image 0.17.2
numba 0.47.0
numpy 1.17.5
opencv-python 4.2.0.34
requests 2.23.0
pandas 0.24.2
Pillow 7.0.0
pip 21.3.1
protobuf 3.19.4
scikit-learn 0.20.0
npu-device 0.1
mindinsight 1.6.1
mindspore-ascend 1.6.1
...
- 历史版本：无

引擎版本三：pytorch_1.5.0-cann_5.0.4-py_3.7-euler_2.8.3-aarch64

- 镜像地址：swr.{region}.myhuaweicloud.com/aip/
pytorch_1_5_ascend:pytorch_1.5.0-cann_5.0.4-py_3.7-euler_2.8.3-aarch64-d910-roma-20220318164813-b3feb87
- 镜像构建时间：20220318164813(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：euleros 2.8.3
- cann：5.0.4
- Python解释器路径及版本：/home/ma-user/anaconda3/envs/PyTorch-1.5.0/bin/python， python 3.7.10

- 三方包安装路径：/home/ma-user/anaconda3/envs/PyTorch-1.5.0/lib/python3.7/site-packages

- 部分三方包版本信息列表：

```
Cython 0.29.14
dask 2.18.1
hccl 0.1.0
mmlcv 0.2.14
npu-bridge 1.15.0
Flask 1.1.1
grpcio 1.26.0
gunicorn 20.0.4
idna 2.10
PyYAML 5.3.1
imageio 2.9.0
imgaug 0.2.6
lxml 4.4.2
matplotlib 3.1.2
psutil 5.7.0
scikit-image 0.17.2
numba 0.47.0
numpy 1.17.5
opencv-python 4.2.0.34
requests 2.23.0
pandas 0.24.2
Pillow 7.0.0
pip 21.3.1
protobuf 3.11.3
scikit-learn 0.20.0
npu-device 0.1
torch 1.5.0+ascend.post4.20220114
torchvision 0.6.0
...
```

- 历史版本：无

3.1.3 已有镜像如何适配迁移至 ModelArts

已有镜像迁移至新版训练管理需要关注如下步骤。

- 为镜像增加新版训练管理的默认用户组 ma-group, gid = 100。假若已有 gid = 100 用户组，则该步骤跳过。
- 为镜像增加新版训练管理的默认用户 ma-user, uid = 1000。假若已有 uid = 1000 用户，则该步骤跳过。
- 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

RUN groupadd ma-group -g 100 && \
    useradd -d /home/ma-user -m -u 1000 -g 100 -s /bin/bash ma-user

# [important] avoid missing training log
ENV PYTHONUNBUFFERED=1

USER ma-user
WORKDIR /home/ma-user
```

注意到新版训练管理默认以 uid = 1000 的用户来运行自定义镜像。因此建议在使用已有镜像用于新版训练管理之前，首先在本地（构建镜像的机器）完成镜像文件权限适配的调试。调试方法如下：

执行如下命令，指定以 uid = 1000 的用户来运行镜像。

```
docker run -u 1000 -ti {自定义镜像} bash
```

在容器（镜像的运行时）中执行相关启动命令，例如

```
python train.py
```

查看是否有 Permission denied 错误，以检查文件权限是否满足要求。

对出现 Permission denied 错误的文件，例如 train.py，执行如下命令修改文件属主，使得文件归属于 uid = 1000 的用户。

```
chown 1000 train.py
```

将适配后的镜像上传至SWR（参考[如何登录并上传镜像到SWR](#)），参考[使用自定义镜像创建训练作业（CPU/GPU）](#)章节在ModelArts上使用。

3.2 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 Gallery。
- 入口2：在ModelArts控制台“训练管理 > 训练作业New > 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

图 3-1 使用自定义镜像创建算法

★ 镜像

预置镜像

自定义

?

选择

代码目录

选择

★ 启动命令

?

0/512

表 3-2 创建算法参数说明

参数	说明
镜像	必选，选择“自定义”。 如果已经在“算法管理”中创建完成基于自定义镜像的算法，此处也可以在“我的算法”中直接选择创建好的算法。

参数	说明
镜像地址	必选。容器镜像地址。单击右边的“选择”，可以从SWR服务选择用户的容器镜像，前提是要先上传到SWR服务中。 用户也可以手动输入SWR上的镜像地址（<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/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}/code”目录中，code为OBS路径的最后一级目录。
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。 <ul style="list-style-type: none">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}/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}/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。

配置训练输入输出数据

训练过程中，自定义算法需要从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。

创建自定义算法时，用户需要将算法代码中定义的输入输出参数进行配置。

- 输入数据配置

表 3-3 输入数据配置

参数	参数说明
映射名称	输入参数的说明，用户可以自定义描述。默认为“数据来源”。
代码路径参数	如果您的算法代码中使用argparse解析data_url为输入数据参数，则在创建的算法需要配置输入数据代码参数为data_url。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。

参数	参数说明
是否添加约束	<p>用户可以根据实际情况限制数据来源的方式，可以设置为“数据存储位置”或者“ModelArts数据集”。</p> <p>如果用户选择数据来源为ModelArts数据集，还可以约束以下三种：</p> <ul style="list-style-type: none">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。
添加输入数据配置	用户可以根据实际算法确定多个输入数据来源。

图 3-2 输入数据配置

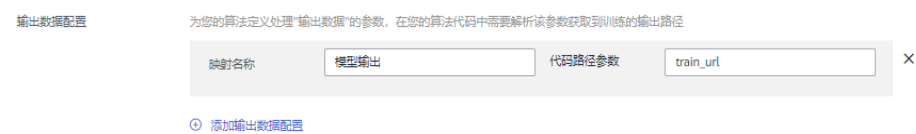


- 输出数据配置

表 3-4 输出数据配置

参数	参数说明
映射名称	输出参数的说明，用户可以自定义描述。默认为“输出数据”。
代码路径参数	<p>如果您的算法代码中使用argparse解析train_url为训练输出参数，则在创建的算法需要配置输出数据参数为train_url。根据实际代码中的训练输出参数定义此处的名称。</p> <p>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。</p>
添加输出数据配置	用户可以根据实际算法确定多个输出数据路径。

图 3-3 输出数据配置



定义超参

使用常用框架创建算法时，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。请注意超参输入的内容应该只包含大小写字母、中文、数字、空格、中划线、下划线、逗号和句号。

图 3-4 添加超参



2. 编辑超参。超参参数说明参见表3-5。

表 3-5 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。
描述	填入超参的描述说明。 超参描述支持大小写字母、中文、数字、空格、中划线、下划线、逗号和句号。
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	可选是或者否。如果您选择否，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。如果您选择是，则不支持删除操作。

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：可选“CPU”、“GPU”和“Ascend”，支持多选。
- 多卡训练：可选“支持”和“不支持”。
- 分布式训练：可选“支持”和“不支持”。

图 3-5 算法训练约束

添加训练约束 ☒ 是 ☐ 否

★ 资源类型	<input type="text"/>
★ 多卡训练	<input type="text"/>
★ 分布式训练	<input type="text"/>

运行环境预览

创建训练作业时，可以打开创建页面右下方的运行环境预览窗口

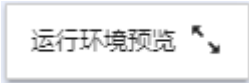
，辅助您了解代码目录、启动文件、输入输出等数据配置在训练容器中的路径。

图 3-6 运行环境预览



后续操作

创建算法完成后，在“算法管理”页面，等待算法就绪。当新创建的算法状态变更为“就绪”时，即可执行其他操作。

可以使用算法快速创建训练作业、构建模型，详细操作请参见[使用自定义镜像创建训练作业（CPU/GPU）](#)。

也可以将新创建的算法发布到AIGallery，分享给其他用户使用。

3.3 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 New”，进入“训练作业”列表。
2. 单击“创建训练作业”，进入“训练作业”页面，该页面填写训练作业相关参数。关键参数解释见[表3-6](#)。

表 3-6 作业参数说明

参数名称	说明
创建方式	必选，选择“自定义算法”。 如果已经在“算法管理”中创建完成基于自定义镜像的算法，此处也可以在“我的算法”中直接选择创建好的算法。
镜像来源	必选，选择“自定义”。
镜像地址	必选。容器镜像地址。单击右边的“选择”，可以从SWR服务选择用户的容器镜像，前提是要先上传到SWR服务中。 用户也可以手动输入SWR上的镜像地址（<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OBS路径“obs://obs-bucket/training-test/code”为例，训练代码会被自动下载至训练容器的“\${MA_JOB_DIR}/code”目录中，code为OBS路径的最后一级目录。
启动命令	必选，镜像的启动命令。在代码目录下载完成后，运行命令会被自动执行。 <ul style="list-style-type: none">• 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}/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}/code/main.sh。

参数名称	说明
训练输入- 参数名称	<p>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：car_data_url、dog_data_url、cat_data_url。</p> <p>例如您的训练代码中使用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。</p> <pre>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()</pre>
训练输入- 数据集/数 据存储位置	<p>选择“数据集”或“数据存储位置”作为训练输入。数据存储位置请选择OBS数据存储位置作为训练输入。</p> <p>训练启动时，会下载数据至训练容器中。</p> <p>以OBS 路径obs://obs-bucket/training-test/data为例，数据会被自动下载至训练容器的“\${MA_MOUNT_PATH}/inputs/\${data_url}_N”目录中，N取值为训练输入参数个数减去1。</p> <p>例如：</p> <ul style="list-style-type: none"> 只有一条训练输入参数data_url时，数据会被自动下载至训练容器的“\${MA_MOUNT_PATH}/inputs/data_url_0/”路径。 如果训练输入参数设置有多条，训练输入参数名称分别为car_data_url、dog_data_url、cat_data_url，则训练数据会被下载到容器的目录分别为“\${MA_MOUNT_PATH}/inputs/car_data_url_0/”、“\${MA_MOUNT_PATH}/inputs/dog_data_url_1/”、“\${MA_MOUNT_PATH}/inputs/cat_data_url_2/”。
训练输出- 参数名称	<p>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。</p>
训练输出- 数据存储位 置	<p>请选择OBS目录作为训练输出。为避免出现错误，建议选择一个空目录用作“训练输出”。</p> <p>训练容器“\${MA_MOUNT_PATH}/outputs/\${train_url}_N/”中的训练结果文件会自动上传到OBS的“obs://obs-bucket/training-test/output”目录下。N取值为训练输出参数个数减去1。</p> <p>例如：</p> <ul style="list-style-type: none"> 只有一条训练输出参数train_url时，训练容器的路径为“\${MA_MOUNT_PATH}/inputs/data_url_0/”。 如果训练输出参数设置有多条，例如，训练输出参数名称分别为car_train_url、dog_train_url、cat_train_url，训练输出数据的容器目录分别为“\${MA_MOUNT_PATH}/inputs/car_train_url_0/”、“\${MA_MOUNT_PATH}/inputs/dog_train_url_1/”、“\${MA_MOUNT_PATH}/inputs/cat_train_url_2/”。

参数名称	说明
超参	可选，超参用于训练调优。
环境变量	容器启动后，系统会加载一些默认的环境变量和用户在此处自行设置的“环境变量”。 系统默认提供的环境变量如表3-7所示，请了解。
故障自动重启	可选。打开开关后，可以设置训练故障重启次数。


表 3-7 系统默认加载的环境变量说明

环境变量	说明
MA_JOB_DIR	代码目录的父目录
MA_MOUNT_PATH	训练输入和训练输出目录的父目录
VC_TASK_INDEX	当前容器索引，容器从0开始编号。单机训练的时候，该字段无意义。在多机作业中，用户可以根据这个值来确定当前容器运行的算法逻辑。
MA_NUM_HOSTS	计算节点个数。系统自动从资源参数的“计算节点个数”中读取。
`\${MA_VJ_NAME}-\${MA_TASK_NAME}-N.\${MA_VJ_NAME}`	表示不同节点的通信域名，例如0号节点的通信域名为`\${MA_VJ_NAME}-\${MA_TASK_NAME}-0.\${MA_VJ_NAME}`。 N表示计算节点个数，例如：计算节点个数为4时，此环境变量分别为： `\${MA_VJ_NAME}-\${MA_TASK_NAME}-0.\${MA_VJ_NAME}`、 `\${MA_VJ_NAME}-\${MA_TASK_NAME}-1.\${MA_VJ_NAME}`、 `\${MA_VJ_NAME}-\${MA_TASK_NAME}-2.\${MA_VJ_NAME}`、 `\${MA_VJ_NAME}-\${MA_TASK_NAME}-3.\${MA_VJ_NAME}`。

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。

表 3-8 资源参数说明

参数名称	说明
资源池	选择公共资源池或专属资源池。
资源类型	根据实际镜像中定义的资源类型选择。

参数名称	说明
规格	<p>针对不同的资源类型，选择资源规格。这里选的规格必须和镜像中定义的训练规格保持一致。</p> <p>例如：训练代码中设置为GPU，这里选为CPU时，可能会导致训练失败。</p> <p>训练时，ModelArts会挂载高速固态硬盘（NVME SSD）至“/cache”目录，用户可以使用此目录来储存临时文件。</p> <p>不同的资源类型的数据盘容量是不同的，为避免训练过程中出现内存不足的情况，请先查看资源规格的硬盘容量大小。</p> 
计算节点个数	选择计算节点的个数。默认值为“1”。
永久保存日志	<p>可选，选项开启时，需要设置作业日志路径，请选择一个空的OBS目录用于存储作业运行中产生的日志文件。例如：obs://obs-bucket/training-test/output/log。</p> <p>说明 请注意选择的OBS目录有读写权限。</p>

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](#)。

3.4 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3-9](#)。

- 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": "jobxxxxxxx-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"
}
```

- RANK_TABLE_FILE 环境变量

表 3-9 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。

3.5 示例：从 0 到 1 制作自定义镜像并创建训练作业（CPU/GPU）

本章节提供了在ModelArts平台使用自定义镜像创建训练作业的样例，帮助您快速熟悉平台的使用方法。

说明

本实践教程仅适用于新版训练作业。

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. [准备工作](#)
2. [制作自定义镜像](#)
3. [上传镜像至SWR服务](#)

4. 创建训练作业

准备工作

- 已注册华为云账号，且在使用ModelArts前检查账号状态，账号不能处于欠费或冻结状态。
- 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](#)。针对之前使用访问密钥授权的用户，建议清空授权，然后使用委托进行授权。
- 已在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3-10](#)所示的文件夹。
创建OBS桶和文件夹的操作指导请参见[创建桶](#)和[新建文件夹](#)。
请确保您使用的OBS与ModelArts在同一区域。

表 3-10 文件夹列表

文件夹名称	用途
“obs://test-modelarts/deep-learning/pytorch/code/”	用于存储训练脚本文件。
“obs://test-modelarts/deep-learning/pytorch/log/”	用于存储训练日志文件。

- 训练脚本 pytorch-verification.py 文件，请上传至对应文件夹下。

pytorch-verification.py 文件内容如下

```
import torch
import torch.nn as nn

x = torch.randn(5, 3)
print(x)

available_dev = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
y = torch.randn(5, 3).to(available_dev)
print(y)
```

制作自定义镜像

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
- cuda-10.2
- python-3.7.13
- pytorch-1.8.1

本示例使用Dockerfile文件定制自定义镜像。以linux x86_x64架构的主机为例，您可以购买相同规格的ECS或者应用本地已有的主机进行自定义镜像的制作。

1. 安装Docker。

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
```

如果**docker images**命令可以执行成功，表示Docker已安装，此步骤可跳过。

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。本示例使用华为开源镜像站提供的 pip 源，其 pip.conf 文件内容如下。

```
[global]
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
timeout = 120
```

说明

在华为开源镜像站<https://mirrors.huaweicloud.com/home>中，搜索 pypi，也可以查看 pip.conf 文件内容。

4. 下载 torch*.whl 文件。

在网站 https://download.pytorch.org/whl/torch_stable.html 搜索并下载如下 whl文件。

- torch-1.8.1+cu102-cp37-cp37m-linux_x86_64.whl
- torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
- torchvision-0.9.1+cu102-cp37-cp37m-linux_x86_64.whl

5. 下载 Miniconda3 安装文件。

使用地址https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh, 下载 Miniconda3 py37 4.12.0 安装文件（对应 python 3.7.13）。

6. 将上述pip源文件、torch*.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
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
├── torch-1.8.1+cu102-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu102-cp37-cp37m-linux_x86_64.whl
```

7. 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
```

```
# 基础容器镜像, https://github.com/NVIDIA/nvidia-docker/wiki/CUDA
```

```
#
```

```
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
```

```
# require Docker Engine >= 17.05
```

```
#
```

```
# builder stage
```

```
FROM nvidia/cuda:10.2-runtime-ubuntu18.04 AS builder
```

```
# 基础容器镜像的默认用户已经是 root
```

```
# USER root
```

```
# 使用华为开源镜像站提供的 pypi 配置
```

```
RUN mkdir -p /root/.pip/
```

```
COPY pip.conf /root/.pip/pip.conf
```

```
# 拷贝待安装文件到基础容器镜像中的 /tmp 目录
```

```
COPY Miniconda3-py37_4.12.0-Linux-x86_64.sh /tmp
```

```
COPY torch-1.8.1+cu102-cp37-cp37m-linux_x86_64.whl /tmp
```

```
COPY torchvision-0.9.1+cu102-cp37-cp37m-linux_x86_64.whl /tmp
```

```
COPY torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl /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_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/miniconda3
```

```
# 使用 Miniconda3 默认 python 环境 (即 /home/ma-user/miniconda3/bin/pip) 安装 torch*.whl
RUN cd /tmp && \
  /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
  /tmp/torch-1.8.1+cu102-cp37-cp37m-linux_x86_64.whl \
  /tmp/torchvision-0.9.1+cu102-cp37-cp37m-linux_x86_64.whl \
  /tmp/torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl

# 构建最终容器镜像
FROM nvidia/cuda:10.2-runtime-ubuntu18.04

# 安装 vim / curl 工具（依然使用华为开源镜像站）
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
  sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
  sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
  apt-get update && \
  apt-get install -y vim curl && \
  apt-get clean && \
  mv /etc/apt/sources.list.bak /etc/apt/sources.list

# 增加 ma-user 用户 (uid = 1000, gid = 100)
# 注意到基础容器镜像已存在 gid = 100 的组，因此 ma-user 用户可直接使用
RUN useradd -m -d /home/ma-user -s /bin/bash -g 100 -u 1000 ma-user

# 从上述 builder stage 中拷贝 /home/ma-user/miniconda3 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

# 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 PATH=$PATH:/home/ma-user/miniconda3/bin \
  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](#)。

8. 确认已创建完成Dockerfile文件。此时 context 文件夹内容如下。

```
context
├── Dockerfile
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
├── torch-1.8.1+cu102-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu102-cp37-cp37m-linux_x86_64.whl
```

9. 确认 Docker Engine 版本。执行如下命令。

```
docker version | grep -A 1 Engine
```

以 Docker Engine 18.09.0 版本为例，上述命令回显示意如下。

```
Engine:
Version:      18.09.0
```

10. 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 pytorch:1.8.1-cuda10.2。

```
# 要求 Docker Engine 版本 >= 17.05
docker build . -t pytorch:1.8.1-cuda10.2
```

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 pytorch:1.8.1-cuda10.2
```

上传镜像至 SWR 服务

1. 登录容器镜像服务控制台，选择区域。

图 3-7 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。您可以自定义组织名称，本示例使用“deep-learning”。

图 3-8 创建组织

创建组织

1.组织名称，全局唯一。

2.当前租户最多可创建5个组织。

3.建议一个组织对应一个公司、部门或个人，以便集中高效地管理镜像资源。

示例：

以公司、部门作为组织:cloud-hangzhou、cloud-develop

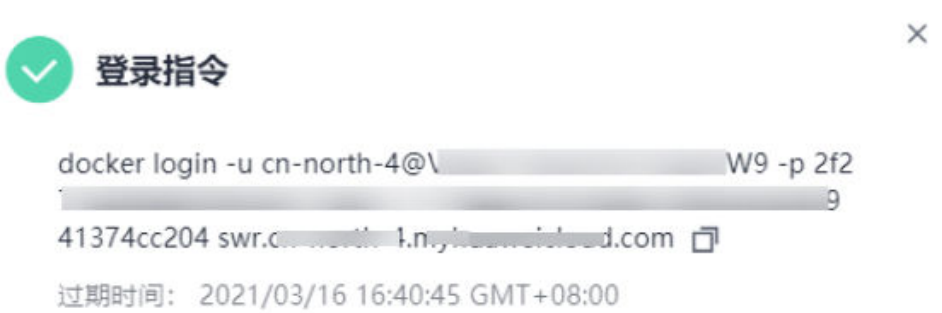
以个人作为组织:john

组织名称

deep-learning

3. 单击右上角“登录指令”，获取登录访问指令。

图 3-9 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。
sudo docker tag pytorch:1.8.1-cuda10.2 swr.cn-north-4.myhuaweicloud.com/deep-learning/pytorch:1.8.1-cuda10.2
- b. 使用docker push命令上传镜像。
sudo docker push swr.cn-north-4.myhuaweicloud.com/deep-learning/pytorch:1.8.1-cuda10.2
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。

“swr.cn-north-4.myhuaweicloud.com/deep-learning/pytorch:1.8.1-cuda10.2”
即为此自定义镜像的“SWR_URL”。

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“下一步”。
 - 创建方式：选择“自定义算法”
 - 镜像来源：选择“自定义”
 - 镜像地址：“swr.cn-north-4.myhuaweicloud.com/deep-learning/pytorch:1.8.1-cuda10.2”
 - 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/deep-learning/pytorch/code/”
 - 启动命令：“/home/ma-user/miniconda3/bin/python \${MA_JOB_DIR}/code/pytorch-verification.py”
 - 资源池：选择公共资源池
 - 类型：选择GPU或者CPU规格。
 - 永久保存日志：打开
 - 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/deep-learning/pytorch/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，训练时间将耗时几十分钟到几小时不等。训练作业执行成功后，日志信息如下所示。

图 3-10 GPU 规格运行日志信息



图 3-11 CPU 规格运行日志信息



```
1 tensor([[ 0.8945, -0.6946,  0.3807],
2          [ 0.6665,  0.3133,  0.8285],
3          [-0.5353, -0.1730, -0.5419],
4          [ 0.4870,  0.5183,  0.2505],
5          [ 0.2679, -0.4996,  0.7919]])
6 tensor([[ 0.9692,  0.4652,  0.5659],
7          [ 2.2032,  1.4157, -0.1755],
8          [-0.6296,  0.5466,  0.6994],
9          [ 0.2353, -0.0089, -1.9546],
10         [ 0.9319,  1.1781, -0.4587]])
11
```

4 使用自定义镜像创建 AI 应用

4.1 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10GB。
- **镜像对外接口**

镜像的对外服务接口需要为8080，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。

- 请求示例
`curl -X POST \ http://{listen_ip}:8080/ \ -F images=@seven.jpg`

- 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，那么必需提供健康检查的接口供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，必需配置健康检查接口。

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 4-1 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**

如果需要在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**

如果需要在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
```

4.2 使用自定义镜像创建的 AI 应用部署服务

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，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模型管理。
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，您可以购买相同规格的ECS或者应用本地已有的主机进行自定义镜像的制作。

1. 安装Docker，可参考[Docker官方文档](#)。可参考以下方式安装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
# 配置华为云的源，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.huaweicloud.com@g" /etc/apt/sources.list && \
    sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.com@g" /etc/apt/sources.list && \
    apt-get update && \
    apt-get install -y python3 python3-pip && \
    pip3 install --trusted-host https://repo.huaweicloud.com -i https://repo.huaweicloud.com/repository/pypi/simple Flask

# 拷贝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)
```

说明

modelarts 平台会将请求转发至自定义镜像起的服务的8080端口，故容器内的服务监听的端口必须是8080，如 test_app.py文件所示。

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 image”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 4-1 启动自定义镜像

```
:/opt/file# docker run -it -p 8080:8080 test:v1
* Serving Flask app "test_app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

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 4-2 校验接口

```
root@: ~# curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/
called default func !
{'name': 'Tom'}
root@: ~# curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/greet
{"response": "Hello, Tom!"}
root@: ~# curl -X GET 127.0.0.1:8080/goodbye
Goodbye!
```

3. 上传自定义镜像至SWR服务。上传镜像的详细操作可参考[SWR用户指南](#)。
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。

图 4-3 上传镜像列表

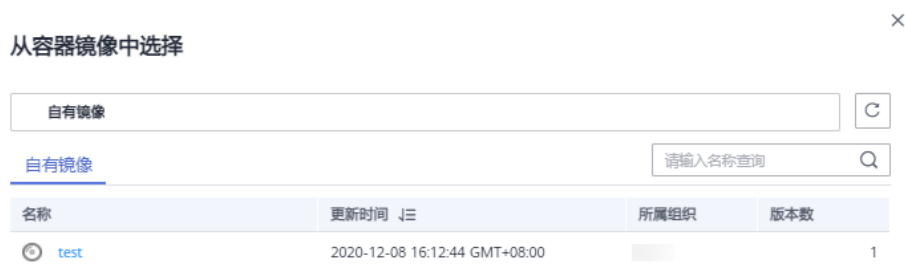


将自定义镜像创建为 AI 应用

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 4-4 选择已制作好的自有镜像



- 容器调用接口：选填，用于指定模型启动的协议和端口号。
- 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[模型配置文件说明](#)。

本样例的配置文件的如下所示：

```
[{
  "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 4-5 调用指南

名称

service-a5b4

服务ID

05744951-2960-4c94

状态

运行中 (57 分钟后停止)

来源

我的部署

调用失败次数/总次数

?

0/0 详情

描述

--

同步数据

同步数据至数据集

个性化配置

数据采集

?

难例筛选

?

服务流量限制

--

调用指南

预测

配置更新记录

难例筛选

监控信息

事件

日志

API接口地址

https://31ac244b430940caa533a44-

说明:

AI应用:

model-

参数配置

>

POST

/

>

POST

/greet

>

GET

/goodbye

3. 您可以通过“预测”页签访问在线服务。

图 4-6 访问在线服务

调用指南

预测

配置更新记录

难例筛选

监控信息

事件

日志

请求路径:

/

请求类型:

application/json

预测代码

返回结果

1

{ "hello": "my name is modelarts" }

1

called default func !

2

{ 'hello': 'my name is modelarts' }

3

4

预测

5 FAQ

5.1 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。

图 5-1 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。您可以自定义组织名称，本示例使用“deep-learning”。

图 5-2 创建组织

创建组织

1. 组织名称，全局唯一。
 2. 当前租户最多可创建5个组织。
 3. 建议一个组织对应一个公司、部门或个人，以便集中高效地管理镜像资源。
- 示例:
- 以公司、部门作为组织: cloud-hangzhou、cloud-develop
 - 以个人作为组织: john

组织名称

deep-learning

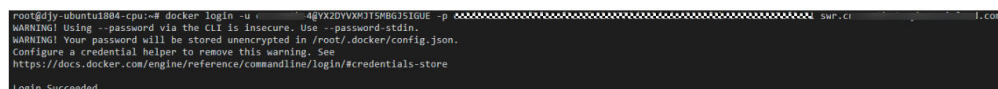
- 单击右上角“登录指令”，获取登录访问指令。

图 5-3 登录指令



- 以root用户登录ECS环境，输入登录指令。

图 5-4 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。

- 登录SWR后，使用docker tag命令给上传镜像打标签。
`sudo docker tag tf-1.13.2:latest swr.xxx.com/deep-learning/tf-1.13.2:latest`
- 使用docker push命令上传镜像。
`sudo docker push swr.xxx.com/deep-learning/tf-1.13.2:latest`

图 5-5 上传镜像



- 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。

图 5-6 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”。