ModelArts

使用自定义镜像

文档版本 01

发布日期 2024-04-30

华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2024。 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

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云计算技术有限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，华为云计算技术有限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 镜像管理 1
2 使用预置镜像 3
2.1 Notebook 基础镜像介绍 3
2.1.1 Notebook 基础镜像功能 3
2.1.2 Notebook 基础镜像列表 4
2.1.3 Notebook 基础镜像 x86 PyTorch 5
2.1.4 Notebook 基础镜像 x86 Tensorflow 11
2.1.5 Notebook 基础镜像 x86 MindSpore 15
2.1.6 Notebook 基础镜像 x86 自定义专用镜像 22
2.2 训练基础镜像详情介绍 24
2.2.1 训练基础镜像列表 25
2.2.2 训练基础镜像详情（PyTorch） 25
2.2.3 训练基础镜像详情（TensorFlow） 26
2.2.4 训练基础镜像详情（Horovod） 27
2.2.5 训练基础镜像详情（MPI） 29
2.2.6 预置框架启动流程说明 29
2.2.6.1 PyTorch 29
2.2.6.2 Tensorflow 32
2.2.6.3 Horovod/MPI/MindSpore-GPU 35
2.3 推理基础镜像介绍 37
2.3.1 推理基础镜像列表 37
2.3.2 推理基础镜像详情 TensorFlow（CPU/GPU） 39
2.3.3 推理基础镜像详情 Pytorch（CPU/GPU） 43
2.3.4 推理基础镜像详情 MindSpore（CPU/GPU） 47
3 Notebook 中使用自定义镜像 53
3.1 在 ModelArts 中进行镜像注册 53
3.2 Notebook 制作自定义镜像方法 54
3.3 将 Notebook 实例保存为自定义镜像 55
3.3.1 保存 Notebook 镜像环境 55
3.3.2 基于自定义镜像创建 Notebook 实例 56
3.4 在 Notebook 中构建自定义镜像并使用 57
3.4.1 使用场景和构建流程说明 57

 版权所有 © 华为云计算技术有限公司

3.4.2 Step1 制作自定义镜像 57
3.4.3 Step2 注册新镜像 59
3.4.4 Step3 创建开发环境并使用 59
3.5 在 ECS 上构建自定义镜像并在 Notebook 中使用 60
3.5.1 使用场景和构建流程说明 60
3.5.2 Step1 准备 Docker 机器并配置环境信息 60
3.5.3 Step2 制作自定义镜像 61
3.5.4 Step3 注册新镜像 65
3.5.5 Step4 创建开发环境并使用 65
4 使用自定义镜像训练模型（模型训练） 67
4.1 训练管理中使用自定义镜像介绍 67
4.2 示例：从 0 到 1 制作自定义镜像并用于训练 69
4.2.1 示例：从 0 到 1 制作自定义镜像并用于训练（PyTorch+CPU/GPU） 69
4.2.2 示例：从 0 到 1 制作自定义镜像并用于训练（MPI+CPU/GPU） 75
4.2.3 示例：从 0 到 1 制作自定义镜像并用于训练（Horovod-PyTorch+GPU） 83
4.2.4 示例：从 0 到 1 制作自定义镜像并用于训练（MindSpore+GPU） 93
4.2.5 示例：从 0 到 1 制作自定义镜像并用于训练（Tensorflow+GPU） 102
4.3 准备训练镜像 108
4.3.1 训练作业自定义镜像规范 108
4.3.2 已有镜像如何适配迁移至 ModelArts 训练平台 109
4.3.3 使用基础镜像构建新的训练镜像 110
4.3.4 在容器镜像中安装 MLNX_OFED 110
4.4 使用自定义镜像创建算法 111
4.5 使用自定义镜像创建训练作业（CPU/GPU） 115
4.6 自定义镜像训练作业失败定位思路 119
5 使用自定义镜像创建 AI 应用（推理部署） 121
5.1 创建 AI 应用的自定义镜像规范 121
5.2 从 0-1 制作自定义镜像并创建 AI 应用 123
6 FAQ 127
6.1 如何登录并上传镜像到 SWR 127
6.2 如何给镜像设置环境变量 128
6.3 如何通过 docker 启动 Notebook 保存后的镜像 128
6.4 如何在 Notebook 开发环境中配置 Conda 源 129
6.5 自定义镜像软件版本匹配注意事项 130
7 修订记录 131

 版权所有 © 华为云计算技术有限公司

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。

版权所有 © 华为云计算技术有限公司

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。

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 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**。

版权所有 © 华为云计算技术有限公司

2 使用预置镜像

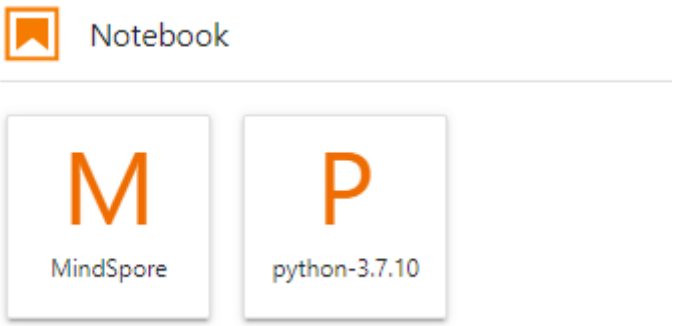
2.1 Notebook 基础镜像介绍

2.1.1 Notebook 基础镜像功能

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，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。

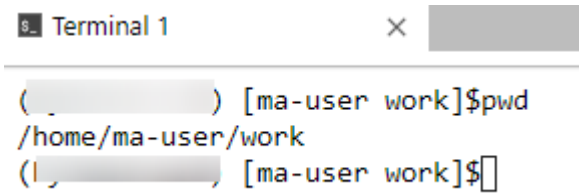
- Notebook：是一款Web应用，用户能够在界面编写代码，并且将代码、数学方程和可视化内容组合到一个文档中。
- JupyterLab插件：插件包括规格切换，停止实例（实例停止后CPU、Memory不再计费）等，提升用户体验。
- 支持SSH远程连接功能：通过SSH连接启动实例，在本地调试就可以操作实例，方便调试。

 版权所有 © 华为云计算技术有限公司

- 预置镜像支持功能开发：基于ModelArts预置镜像进行依赖安装配置后，保存为自定义镜像，能直接在ModelArts用于训练作业。

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/work”。

创建实例，持久化存储挂载存放路径为“/home/ma-user/work”目录，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。



使用预置镜像创建 Notebook 实例

创建Notebook实例时选择预置镜像，创建成功后，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”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。

2.1.2 Notebook 基础镜像列表

ModelArts开发环境提供Docker容器镜像，可作为预构建容器运行。预置镜像里面包含PyTorch，Tensorflow，MindSpore等常用AI引擎框架，镜像命名以AI引擎为主，并且每个镜像里面都预置了很多常用包，用户可以直接使用而无需重新安装。

表 2-1 X86 预置镜像列表

引擎类型	镜像名称
PyTorch	pytorch1.8-cuda10.2-cudnn7-ubuntu18.04
	pytorch1.10-cuda10.2-cudnn7-ubuntu18.04
	pytorch1.4-cuda10.1-cudnn7-ubuntu18.04
Tensorflow	tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04
	tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04
MindSpore	mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04
	mindspore1.7.0-py3.7-ubuntu18.04
	mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04
	mindspore1.2.0-openmpi2.1.1-ubuntu18.04

版权所有 © 华为云计算技术有限公司

引擎类型	镜像名称
无AI引擎（专用于自定义镜像的基础镜像）	conda3-cuda10.2-cudnn7-ubuntu18.04
	conda3-ubuntu18.04

2.1.3 Notebook 基础镜像 x86 PyTorch

PyTorch包含三种镜像：pytorch1.8-cuda10.2-cudnn7-ubuntu18.04，pytorch1.10-cuda10.2-cudnn7-ubuntu18.04，pytorch1.4-cuda10.1-cudnn7-ubuntu18.04

镜像一：pytorch1.8-cuda10.2-cudnn7-ubuntu18.04

表 2-2 pytorch1.8-cuda10.2-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Pytorch 1.8	是（cuda 10.2）	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			torch 1.8.0 torchvision 0.9.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 pillow 9.2.0 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 tensorboard 2.1.1	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像二：pytorch1.10-cuda10.2-cudnn7-ubuntu18.04

表 2-3 pytorch1.10-cuda10.2-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Pytorch 1.10	是（cuda 10.2）	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_10:pytorch_1.10.2-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20221008154718-2b3e39c	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			torch 1.10.2 torchvision 0.11.3 ipykernel 5.3.4 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 pillow 9.2.0 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像三：pytorch1.4-cuda10.1-cudnn7-ubuntu18.04

表 2-4 pytorch1.4-cuda10.1-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Pytorch 1.4	是（cuda 10.1）	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_4:pytorch_1.4-cuda_10.1-py37-ubuntu_18.04-x86_64-20220926104017-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			torch 1.4.0 torchvision 0.5.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 pillow 9.2.0 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 tensorboard 2.1.1	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx pandoc python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

2.1.4 Notebook 基础镜像 x86 Tensorflow

Tensorflow包含两种镜像：tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04，tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04

镜像一：tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04

表 2-5 tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Tensorflow 2.1	是（cuda 10.1）	swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220926144607-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			tensorflow 2.1.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 opencv-python 4.1.2.30 pandas 1.1.5 pillow 9.2.0 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 tensorboard 2.1.1	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像二：tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04

表 2-6 tensorflow1.13-cuda10.0-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Tensorflow 1.13-gpu	是（cuda 10.0）	swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_1_13:tensorflow_1.13-cuda_10.0-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			tensorflow-gpu 1.13.1 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.0.1.rc0.ffd1c0c8 numpy 1.17.0 opencv-python 4.1.2.30 pandas 1.1.5 pillow 6.2.0 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.2.2 scikit-learn 0.22.1 tornado 6.2	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

2.1.5 Notebook 基础镜像 x86 MindSpore

MindSpore包含四种镜像：mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04，mindspore1.7.0-py3.7-ubuntu18.04，mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04，mindspore1.2.0-openmpi2.1.1-ubuntu18.04

镜像一：mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04

表 2-7 mindspore1.7.0-cuda10.1-py3.7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Mindspore-gpu 1.7.0	是（cuda 10.1）	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220926104017-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			mindspore-gpu 1.7.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.17.0 pandas 1.1.5 pillow 9.1.1 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.1 mindinsight 1.7.0 mindvision 0.1.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像二：mindspore1.7.0-py3.7-ubuntu18.04

表 2-8 mindspore1.7.0-py3.7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Mindspore 1.7.0	无	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220926104017-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			mindspore 1.7.0 ipykernel 6.7.0 ipython 7.34.0 jupyter-client 7.3.4 ma-cau 1.1.6 ma-cau-adapter 1.1.3 ma-cli 1.2.2 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.17.0 pandas 1.1.5 pillow 9.1.1 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.1 mindinsight 1.7.0 mindvision 0.1.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像三：mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04

表 2-9 mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Mindspore-gpu 1.2.0	是（cuda 10.1）	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_2_0:mindspore_1.2.0-py_3.7-cuda_10.1-ubuntu_18.04-x86_64-20220926104106-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			mindspore-gpu 1.2.0 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 1.1.5 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 pandas 1.1.5 pillow 6.2.0 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 mindinsight 1.2.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libcudnn7 libcudnn7-dev libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像四：mindspore1.2.0-openmpi2.1.1-ubuntu18.04

表 2-10 mindspore1.2.0-openmpi2.1.1-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
Mindspore 1.2.0	无	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_2_0:mindspore_1.2.0-py_3.7-ubuntu_18.04-x86_64-20220926104106-041ba2e	PyPI 程序包	Ubuntu 软件包

版权所有 © 华为云计算技术有限公司

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
			mindspore 1.2.0 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 1.1.5 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.19.5 pandas 6.2.0 pillow 9.1.1 pip 22.1.2 psutil 5.8.0 PyYAML 5.1 scipy 1.5.2 scikit-learn 0.22.1 tornado 6.2 mindinsight 1.2.0	automake build-essential ca-certificates cmake cpp curl ffmpeg g++ gcc gfortran git git-lfs grep libjpeg-dev:amd64 libjpeg8-dev:amd64 openssh-client openssh-server nginx python3 rpm screen tar tmux unzip vim wget zip

2.1.6 Notebook 基础镜像 x86 自定义专用镜像

自定义镜像包含两种镜像：conda3-cuda10.2-cudnn7-ubuntu18.04，conda3-ubuntu18.04，该类镜像是无AI引擎以及相关的软件包，镜像较小，只有2~5G。用户

版权所有 © 华为云计算技术有限公司

使用此类镜像做基础镜像，安装自己需要的引擎版本和依赖包，可扩展性更高。并且这些镜像预置了一些开发环境启动所必要的配置，用户无需对此做任何适配，安装所需的软件包即可使用。

此类镜像为最基础的镜像，主要应对用户做自定义镜像时基础镜像太大的问题，所以镜像中未安装任何组件；如果需使用OBS SDK相关功能，推荐使用ModelArts SDK进行文件复制等操作，详细操作请参考**文件传输**。

镜像一：conda3-cuda10.2-cudnn7-ubuntu18.04

表 2-11 conda3-cuda10.2-cudnn7-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
无	是（cuda 10.2）	swr.{region_id}.myhuaweicloud.com/atelier/user_defined_base:cuda_10.2-ubuntu_18.04-x86_64-20221008154718-2b3e39c	PyPI 程序包	Ubuntu 软件包
			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 1.1.5 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.21.6 pandas 1.3.5 pillow 9.2.0 pip 20.3.3 psutil 5.9.1 PyYAML 6.0 scipy 1.7.3 tornado 6.2	automake build-essential ca-certificates cmake cpp curl g++ gcc gfortran grep libcudnn7 libcudnn7-dev nginx python3 rpm tar unzip vim wget zip

版权所有 © 华为云计算技术有限公司

镜像二：conda3-ubuntu18.04

表 2-12 conda3-ubuntu18.04 镜像介绍

AI引擎框架	是否使用GPU（CUDA版本）	URL	包含的依赖项	
无	否	swr.{region_id}.myhuaweicloud.com/atelier/user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c 例如： 华北-北京四 swr.cn-north-4.myhuaweicloud.com/atelier/user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c 华东-上海一 swr.cn-east-3.myhuaweicloud.com/atelier/user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c 华南-广州 swr.cn-south-1.myhuaweicloud.com/atelier/user_defined_base:ubuntu_18.04-x86_64-20221008154718-2b3e39c	PyPI 程序包	Ubuntu 软件包
			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 1.1.5 matplotlib 3.5.1 modelarts 1.4.11 moxing-framework 2.1.0.5d9c87c8 numpy 1.21.6 pandas 1.3.5 pillow 9.2.0 pip 20.3.3 psutil 5.9.1 PyYAML 6.0 scipy 1.7.3 tornado 6.2	automake build-essential ca-certificates cmake cpp curl g++ gcc gfortran grep nginx python3 rpm tar unzip vim wget zip

2.2 训练基础镜像详情介绍

版权所有 © 华为云计算技术有限公司

2.2.1 训练基础镜像列表

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。

训练基础镜像列表

ModelArts中预置的训练基础镜像如下表所示。

表 2-13 ModelArts 训练基础镜像列表

引擎类型	版本名称
PyTorch	pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64
TensorFlow	tensorflow_2.1.0-cuda_10.1-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_1804-x86_64

说明

不同区域支持的AI引擎有差异，请以实际环境为准。

2.2.2 训练基础镜像详情（PyTorch）

介绍预置的PyTorch镜像详情。

引擎版本：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
```

 版权所有 © 华为云计算技术有限公司

```
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
mmcv 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

2.2.3 训练基础镜像详情（TensorFlow）

介绍预置的TensorFlow镜像详情。

引擎版本：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
```

 版权所有 © 华为云计算技术有限公司

```
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
tifffile 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

2.2.4 训练基础镜像详情（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
```

 版权所有 © 华为云计算技术有限公司

```
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

 版权所有 © 华为云计算技术有限公司

2.2.5 训练基础镜像详情（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
tifffile 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

2.2.6 预置框架启动流程说明

2.2.6.1 PyTorch

ModelArts训练服务支持了多种AI引擎，并对不同的引擎提供了针对性适配，用户在使用这些引擎进行模型训练时，训练的启动命令也需要做相应适配，本文讲解了使用PyTorch引擎所需要做的适配。

 版权所有 © 华为云计算技术有限公司

PyTorch 框架启动原理

规格和节点个数

下面以选择“GPU: 8*GP-Vnt1 | CPU: 72核 | 内存：512GB”规格为例，介绍在单机和分布式场景下ModelArts规格资源的分配情况。

单机作业时（即选择的节点数为1），ModelArts只会在一个节点上启动一个训练容器，该训练容器独享节点规格的可使用资源。

分布式作业时（即选择的节点数大于1），worker的数量和创建作业时选择的节点数一致，每个worker将被分配到所选规格对应的计算资源。例如计算节点个数为“2”时，将启动2个worker，每个worker拥有“GPU: 8*GP-Vnt1 | CPU: 72核 | 内存：512GB”的计算资源。

网络通信介绍

- 单机作业不涉及网络通信情况。
- 分布式作业的涉及网络通信则可以分为节点内网络通信和节点间网络通信。

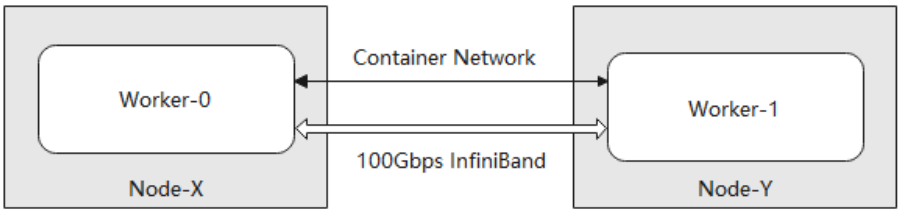
节点内网络

使用NVLink和共享内存通信。

节点间网络

当计算节点个数大于1时，将启动PyTorch引擎分布式训练模式。PyTorch引擎的分布式模式如下图所示，worker之间可通过容器网络和100Gbps的InfiniBand网卡或RoCE网卡通信，部分规格中使用RoCE网卡，将在规格中额外提示。其中，容器网络可以通过DNS域名通信，但网络性能一般，可用于点对点小规模通信需求；InfiniBand网络和RoCE网络为高性能网络，可用于集合通信等分布式训练的场景。

图 2-1 分布式模式



启动命令

训练服务使用作业镜像中默认的python解释器启动训练脚本（即“which python”命令指向的可执行文件），启动时的工作目录（即pwd命令或python中“os.getcwd()”返回的目录）为“/home/ma-user/user-job-dir/<代码目录最外层目录名>”。

- **单机单卡启动命令**

  ```
  python <启动文件相对路径> <作业运行参数>
  ```

  - 启动文件相对路径：启动文件相对“/home/ma-user/user-job-dir/<代码目录最外层目录名>”的路径 。
  - 作业运行参数：训练作业中配置的运行参数 。

版权所有 © 华为云计算技术有限公司

图 2-2 创建训练作业详情

例如控制台上设置如上图所示，则控制台后台执行命令如下：

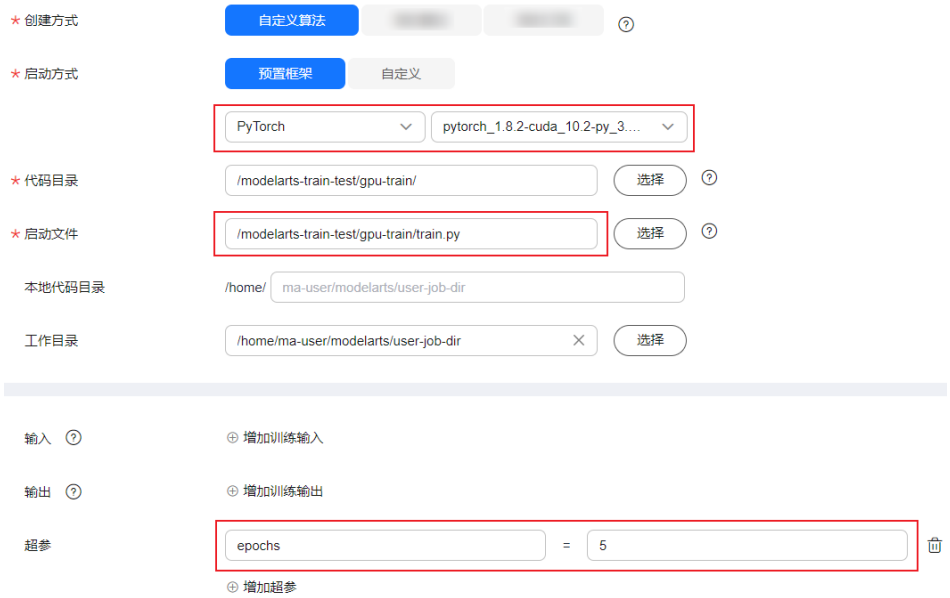
```
python /home/ma-user/modelarts/user-job-dir/gpu-train/train.py --epochs 5
```

- **单机多卡启动命令**

  ```
  python <启动文件相对路径> --init_method "tcp://${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}:${port}" <作业运行参数>
  ```

  - 启动文件相对路径：启动文件相对“/home/ma-user/user-job-dir/<代码目录最外层目录名>”的路径 。
  - ${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}：worker-0所在容器的域名，请参考**系统默认加载的环境变量说明**。
  - port: worker-0所在容器的默认通信端口。
  - 作业运行参数：训练作业中配置的运行参数。

图 2-3 创建训练作业详情



版权所有 © 华为云计算技术有限公司

例如控制台上设置如上图所示，则控制台后台执行命令如下：

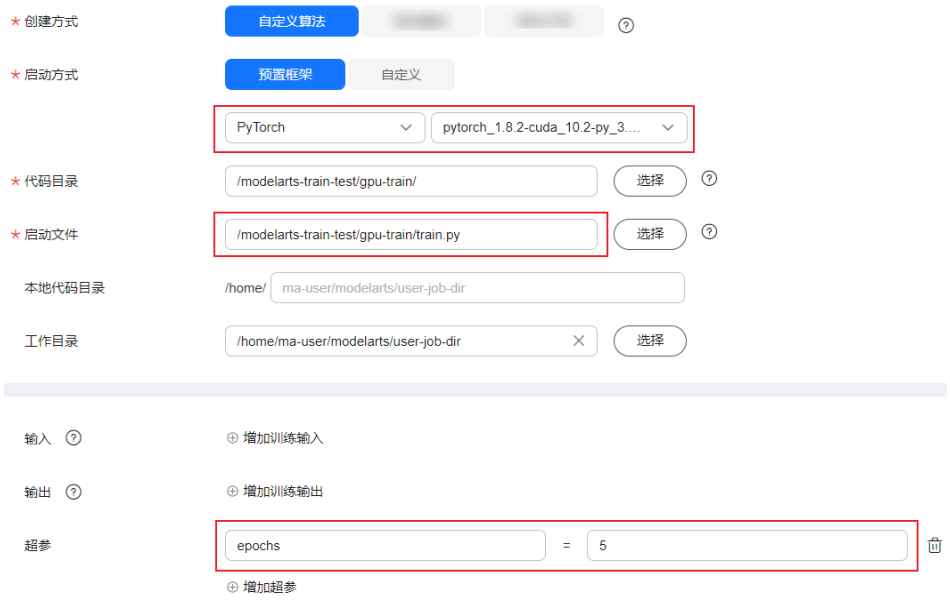
```
python /home/ma-user/modelarts/user-job-dir/gpu-train/train.py --init_method "tcp://${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}:${port}" --epochs 5
```

- **多机多卡启动命令**

```
python <启动文件相对路径> --init_method "tcp://${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}:${port}" --rank <rank_id> --world_size <node_num> <作业运行参数>
```

  - 启动文件相对路径：启动文件相对“/home/ma-user/user-job-dir/<代码目录最外层目录名>”的路径 。
  - ${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}：worker-0所在容器的域名，请参考**系统默认加载的环境变量说明**。
  - port: worker-0所在容器的默认通信端口。
  - rank：worker的序号。
  - node_num：worker的数量。
  - 作业运行参数：训练作业中配置的运行参数。

图 2-4 创建训练作业详情



例如控制台上设置如上图所示，则控制台后台执行命令如下：

```
python /home/ma-user/modelarts/user-job-dir/gpu-train/train.py --init_method "tcp://${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}:${port}" --rank "${rank_id}" --world_size "${node_num}" --epochs 5
```

2.2.6.2 Tensorflow

ModelArts训练服务支持了多种AI框架，并对不同的引擎提供了针对性适配，用户在使用这些框架进行模型训练时，训练的启动命令也需要做相应适配。本文介绍了Tensorflow框架启动原理、控制台上创建训练任务时后台对应的启动命令。

Tensorflow 框架启动原理

规格和节点个数

下面以选择“GPU: 8*GP-Vnt1 | CPU: 72核 | 内存：512GB”规格为例，介绍在单机和分布式场景下ModelArts规格资源的分配情况。

版权所有 © 华为云计算技术有限公司

单机作业时（即选择的节点数为1），ModelArts只会在一个节点上启动一个训练容器，该训练容器独享节点规格的可使用资源。

分布式作业时（即选择的节点数大于1），ModelArts会优先在相同节点上启动一个parameter server（以下简称ps）和一个worker，其中ps将分配一半的CPU和内存资源，即ps拥有“CPU: 36核 | 内存：256GB”的计算资源，worker拥有“GPU: 8*GP-Vnt1 | CPU: 36核 | 内存：256GB”的计算资源。

需要注意的是ps只会分配到CPU和内存资源，而worker除CPU和内存外，还可能分配到加速卡（纯CPU规格除外）。如本例中，每个worker将分配到八张GP Vnt1加速卡，如果ps和worker在相同节点上启动，则磁盘资源由ps和worker共享。

网络通信介绍

- 单机作业不涉及网络通信情况。
- 分布式作业的涉及网络通信则可以分为节点内网络通信和节点间网络通信。

节点内网络

节点内网络通信即同一个节点上的ps和woker间的网络通信，又可以分为两种情况：容器网络和主机网络。

- 在使用公共规格进行训练时，使用的是容器网络。
- 在使用专属池训练时，如果节点配置的是RoCE网卡，使用的是主机网络；如果节点配置的是Infiniband网卡，使用的是容器网络。

节点间网络

分布式作业存在节点间ps和worker的通信，当前ModelArts主要提供了Infiniband网卡或RoCE网卡，带宽高达100Gb/s。

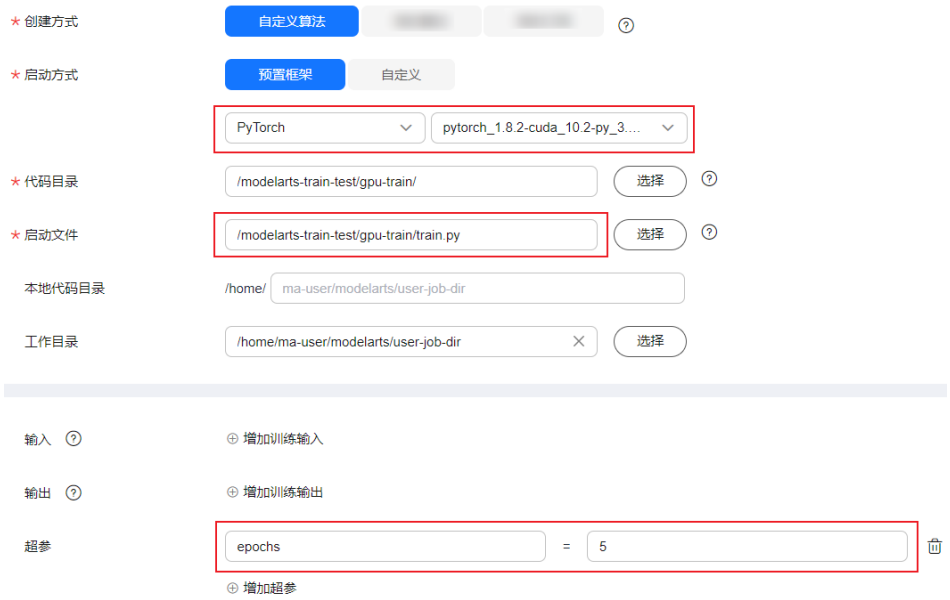
启动命令

训练服务使用作业镜像中默认的python解释器启动训练脚本（即“which python”命令指向的可执行文件），启动时的工作目录（即pwd命令或python中“os.getcwd()”返回的目录）为“/home/ma-user/user-job-dir/<代码目录最外层目录名>”。

- **单机启动命令**
  ```
  python <启动文件相对路径> <作业运行参数>
  ```
  - 启动文件相对路径：启动文件相对“/home/ma-user/user-job-dir/<代码目录最外层目录名>”的路径 。
  - 作业运行参数：训练作业中配置的运行参数 。

 版权所有 © 华为云计算技术有限公司

图 2-5 创建训练作业详情



例如控制台上设置如上图所示，则控制台后台执行命令如下：

```
python /home/ma-user/modelarts/user-job-dir/gpu-train/train.py --epochs 5
```

- **分布式启动命令**

  ```
  python --task_index ${VC_TASK_INDEX} --ps_hosts ${TF_PS_HOSTS} --worker_hosts ${TF_WORKER_HOSTS} --job_name ${MA_TASK_NAME} <启动文件相对路径> <作业运行参数>
  ```

  - VC_TASK_INDEX： task序号，如0/1/2。
  - TF_PS_HOSTS ： ps节点地址数组，如[xx-ps-0.xx:TCP_PORT,xx-ps-1.xx:TCP_PORT]，TCP_PORT是一个在5000~10000的随机端口。
  - TF_WORKER_HOSTS：worker节点地址数组，如[xx-worker-0.xx:TCP_PORT,xx-worker-1.xx:TCP_PORT]，TCP_PORT是一个在5000~10000的随机端口。
  - MA_TASK_NAME：任务名称，ps或worker。
  - 启动文件相对路径：启动文件相对“/home/ma-user/user-job-dir/<代码目录最外层目录名>”的路径。
  - 作业运行参数：训练作业中配置的运行参数。

版权所有 © 华为云计算技术有限公司

图 2-6 创建训练作业详情

例如控制台上设置如上图所示，则控制台后台执行命令如下：

```
python --task_index "$VC_TASK_INDEX" --ps_hosts "$TF_PS_HOSTS" --worker_hosts
"$TF_WORKER_HOSTS" --job_name "$MA_TASK_NAME"
 /home/ma-user/modelarts/user-job-dir/gpu-train/train.py --epochs 5
```

2.2.6.3 Horovod/MPI/MindSpore-GPU

ModelArts训练服务支持了多种AI引擎，并对不同的引擎提供了针对性适配，用户在使用这些引擎进行模型训练时，训练的算法代码也需要做相应适配，本文讲解了使用Horovod/MPI/MindSpore-GPU引擎所需要做的代码适配。

Horovod/MPI/MindSpore-GPU 框架启动原理

规格和节点个数

下面以选择“GPU: 8*GP-Vnt1 | CPU: 72核 | 内存：512GB”规格为例，介绍在单机和分布式场景下ModelArts规格资源的分配情况。

单机作业时（即选择的节点数为1），ModelArts只会在一个节点上启动一个训练容器，该训练容器独享节点规格的可使用资源。

分布式作业时（即选择的节点数大于1），worker的数量和创建作业时选择的节点数一致，每个worker将被分配到所选规格对应的计算资源。例如计算节点个数为“2”时，将启动2个worker，每个worker拥有“GPU: 8*GP-Vnt1 | CPU: 72核 | 内存：512GB”的计算资源。

网络通信介绍

- 单机作业不涉及网络通信情况。
- 分布式作业的涉及网络通信则可以分为节点内网络通信和节点间网络通信。

节点内网络

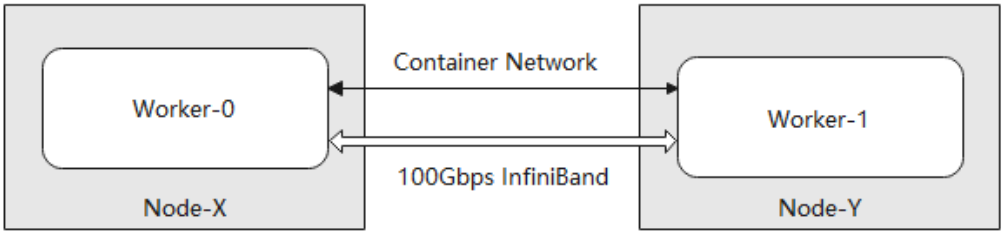
使用NVLink和共享内存通信。

节点间网络

版权所有 © 华为云计算技术有限公司

当计算节点个数大于1时，将启动PyTorch引擎分布式训练模式。PyTorch引擎的分布式模式如下图所示，worker之间可通过容器网络和100Gbps的InfiniBand网卡或RoCE网卡通信，部分规格中使用RoCE网卡，将在规格中额外提示。其中，容器网络可以通过DNS域名通信，但网络性能一般，可用于点对点小规模通信需求；InfiniBand网络和RoCE网络为高性能网络，可用于集合通信等分布式训练的场景。

图 2-7 分布式模式



启动命令

训练服务使用作业镜像中默认的python解释器启动训练脚本，即which python命令指向的可执行文件，启动时的工作目录（即pwd命令或python中os.getcwd()返回的目录）为/home/ma-user/user-job-dir/<代码目录最外层目录名>。

启动命令

```
mpirun \
-np ${OPENMPI_NP} \
-hostfile ${OPENMPI_HOST_FILE_PATH} \
-mca plm_rsh_args "-p ${SSHD_PORT}" \
-tune ${TUNE_ENV_FILE} \
${OPENMPI_BIND_ARGS} \
${OPENMPI_X_ARGS} \
${OPENMPI_MCA_ARGS} \
${OPENMPI_EXTRA_ARGS} \
python <启动文件相对路径> <作业运行参数>
```

- OPENMPI_NP：可控制mpirun启动的进程数，默认为“GPU 卡数 * 节点数”，不建议修改。
- OPENMPI_HOST_FILE_PATH：可控制hostfile参数，不建议修改。
- SSHD_PORT：可控制ssh登录端口，不建议修改。
- TUNE_ENV_FILE：将worker-0的如下env，广播到当前训练作业的其他worker节点。
  - MA_ 前缀的env
  - SHARED_ 前缀的env
  - S3_ 前缀的env
  - PATH的env
  - VC_WORKER_ 前缀的env
  - SCC前缀的env
  - CRED前缀的env

    ```
    env|grep -E '^MA_|^SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED'|grep -v '='> ${TUNE_ENV_FILE}
    ```
- OPENMPI_BIND_ARGS：可控制mprun cpu绑核行为，默认设置如下。

  ```
  OPENMPI_BIND_ARGS="-bind-to none -map-by slot"
  ```


版权所有 © 华为云计算技术有限公司

- OPENMPI_X_ARGS：可控制mpirun -x参数，默认设置如下。
  OPENMPI_X_ARGS="-x LD_LIBRARY_PATH -x HOROVOD_MPI_THREADS_DISABLE=1 -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=ib0,bond0,eth0 -x NCCL_SOCKET_FAMILY=AF_INET -x NCCL_IB_DISABLE=0"
- OPENMPI_MCA_ARGS：可控制mpirun -mca参数，默认设置如下。
  OPENMPI_MCA_ARGS="-mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true"
- OPENMPI_EXTRA_ARGS：控制可额外传递到mpirun的参数，默认设置为空。
- 启动文件相对路径：启动文件相对“/home/ma-user/user-job-dir/<代码目录最外层目录名>”的路径 。
- 作业运行参数：训练作业中配置的运行参数 。

图 2-8 创建训练作业详情

例如控制台上设置如上图所示，则控制台后台执行命令如下：

```
mpirun \
-np ${np} \
-hostfile ${OPENMPI_HOST_FILE_PATH} \
-mca plm_rsh_args "-p ${SSHD_PORT}" \
-tune ${TUNE_ENV_FILE} \
${OPENMPI_BIND_ARGS} \
${OPENMPI_X_ARGS} \
${OPENMPI_MCA_ARGS} \
${OPENMPI_EXTRA_ARGS} \
python /home/ma-user/user-job-dir/gpu-train/train.py --datasets=obs://modelarts-train-test/gpu-train/
data_url_0
```

说明

Horovod/MPI/MindSpore-GPU框架单机启动命令和分布式启动命令无区别。

2.3 推理基础镜像介绍

2.3.1 推理基础镜像列表

ModelArts的推理平台提供了一系列的基础镜像，用户可以基于这些基础镜像构建自定义镜像，用于部署推理服务。

版权所有 © 华为云计算技术有限公司

X86 架构（CPU/GPU）的推理基础镜像

表 2-14 TensorFlow

AI引擎版本	支持的运行环境	URI
2.1.0	CPU GPU(cuda10.1)	swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20221121111529-d65d817
1.15.5	CPU GPU(cuda11.4)	swr.{region_id}.myhuaweicloud.com/aip/tensorflow_1_15:tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64-20220524162601-50d6a18
2.6.0	CPU GPU(cuda11.2)	swr.{region_id}.myhuaweicloud.com/aip/tensorflow_2_6:tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

表 2-15 Pytorch

AI引擎版本	支持的运行环境	URI
1.8.0	CPU GPU(cuda10.2)	swr.{region_id}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20221118143845-d65d817
1.8.2	CPU GPU(cuda11.1)	swr.{region_id}.myhuaweicloud.com/aip/pytorch_1_8:pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220524162601-50d6a18

表 2-16 MindSpore

AI引擎版本	支持的运行环境	URI
1.7.0	CPU	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
1.7.0	GPU(cuda10.1)	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
1.7.0	GPU(cuda11.1)	swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76

版权所有 © 华为云计算技术有限公司

2.3.2 推理基础镜像详情 TensorFlow（CPU/GPU）

ModelArts提供了以下TensorFlow（CPU/GPU）推理基础镜像：

- **引擎版本一：tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64**
- **引擎版本二： tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64**
- **引擎版本三：tensorflow_2.6.0-cuda_11.2-py_3.7-ubuntu_18.04-x86_64**

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

- 镜像地址：swr.{region_id}.myhuaweicloud.com/atelier/tensorflow_2_1:tensorflow_2.1.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20221121111529-d65d817
- 镜像构建时间：20220713110657(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pip安装包列表：

```
Cython                    0.29.21
easydict                  1.9
Flask                     2.0.1
grpcio                    1.47.0
gunicorn                  20.1.0
h5py                      3.7.0
ipykernel                 6.7.0
Jinja2                    3.0.1
lxml                      4.9.1
matplotlib                3.5.1
moxing-framework          2.1.0.5d9c87c8
numpy                     1.19.5
opencv-python             4.1.2.30
pandas                    1.1.5
Pillow                    9.2.0
pip                       22.1.2
protobuf                  3.20.1
psutil                    5.8.0
PyYAML                    5.1
requests                  2.27.1
scikit-learn              0.22.1
scipy                     1.5.2
sklearn                   0.0
tensorboard               2.1.1
tensorboardX              2.0
tensorflow                2.1.0
tensorflow-estimator      2.1.0
wheel                     0.37.1
zipp                      3.8.0
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
cuda
curl
```


版权所有 © 华为云计算技术有限公司

```
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

引擎版本二： tensorflow_1.15.5-cuda_11.4-py_3.8-ubuntu_20.04-x86_64

- 镜像地址：swr.{region_id}.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

 版权所有 © 华为云计算技术有限公司

- 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pip安装包列表：

```
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
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
```

 版权所有 © 华为云计算技术有限公司

```
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

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_id}.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pip安装包列表：

```
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
gunicorn 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
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
```

 版权所有 © 华为云计算技术有限公司

```
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

2.3.3 推理基础镜像详情 Pytorch（CPU/GPU）

ModelArts提供了以下Pytorch（CPU/GPU）推理基础镜像：

- **引擎版本一：pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64**
- **引擎版本二：pytorch_1.8.2-cuda_11.1-py_3.7-ubuntu_18.04-x86_64**

版权所有 © 华为云计算技术有限公司

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_id}.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20221118143845-d65d817
- 镜像构建时间：20220713110657 (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pip安装包列表：

```
Cython                  0.27.3
easydict                1.9
Flask                   2.0.1
fonttools               4.34.4
gunicorn                20.1.0
ipykernel               6.7.0
Jinja2                  3.0.1
lxml                    4.9.1
matplotlib              3.5.1
mmcv                    1.2.7
moxing-framework            2.1.0.5d9c87c8
numpy                   1.19.5
opencv-python               4.1.2.30
pandas                  1.1.5
Pillow                  9.2.0
pip                     22.1.2
protobuf                3.20.1
psutil                  5.8.0
PyYAML                  5.1
requests                2.27.1
scikit-learn            0.22.1
scipy                   1.5.2
sklearn                 0.0
tensorboard             2.1.1
tensorboardX            2.0
torch                   1.8.0
torchtext               0.5.0
torchvision             0.9.0
tornado                 6.2
tqdm                    4.64.0
traitlets               5.3.0
typing_extensions           4.3.0
urllib3                 1.26.10
watchdog                2.0.0
wcwidth                 0.2.5
Werkzeug                2.1.2
wheel                   0.37.1
yapf                    0.32.0
zipp                    3.8.0
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
```

 版权所有 © 华为云计算技术有限公司

```
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

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_id}.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

 版权所有 © 华为云计算技术有限公司

- 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pip安装包列表：

```
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
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
```

 版权所有 © 华为云计算技术有限公司

```
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

2.3.4 推理基础镜像详情 MindSpore（CPU/GPU）

ModelArts提供了以下MindSpore（CPU/GPU）推理基础镜像：

- **引擎版本一：mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64**
- **引擎版本二：mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64**
- **引擎版本三：mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64**

引擎版本一：mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cpu-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- 镜像构建时间：20220702120711(yyyy-mm-dd-hh-mm-ss)
- 镜像系统版本：Ubuntu 18.04.4 LTS
- 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pip安装包列表：

```
cycler                        0.11.0
easydict                       1.9
Flask                         2.0.1
grpcio                        1.47.0
gunicorn                       20.1.0
ipykernel                     6.7.0
Jinja2                        3.0.1
lxml                          4.9.0
matplotlib                     3.5.1
mindinsight                    1.7.0
mindspore                      1.7.0
mindvision                    0.1.0
moxing-framework                2.1.0.5d9c87c8
numpy                         1.17.0
opencv-contrib-python-headless   4.6.0.66
opencv-python-headless          4.6.0.66
pandas                        1.1.5
```


版权所有 © 华为云计算技术有限公司

```
Pillow                  9.1.1
pip                    22.1.2
protobuf                3.20.1
psutil                 5.8.0
PyYAML                  5.1
requests                2.27.1
scikit-learn            0.22.1
scipy                  1.5.2
setuptools              62.6.0
sklearn                 0.0
tensorboardX            2.0
threadpoolctl           3.1.0
tomli                  2.0.1
tornado                 6.1
tqdm                    4.64.0
traitlets              5.3.0
treelib                1.6.1
urllib3                1.26.9
wheel                   0.37.1
zipp                   3.8.0
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
```

 版权所有 © 华为云计算技术有限公司

```
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

引擎版本二：mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_10.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- 镜像构建时间：20220702120711(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/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pip安装包列表：

```
cycler                        0.11.0
easydict                       1.9
Flask                         2.0.1
grpcio                        1.47.0
gunicorn                       20.1.0
ipykernel                      6.7.0
Jinja2                        3.0.1
lxml                          4.9.0
matplotlib                     3.5.1
mindinsight                    1.7.0
mindspore                      1.7.0
mindvision                     0.1.0
moxing-framework                2.1.0.5d9c87c8
numpy                         1.17.0
opencv-contrib-python-headless   4.6.0.66
opencv-python-headless          4.6.0.66
pandas                        1.1.5
Pillow                        9.1.1
pip                           22.1.2
protobuf                       3.20.1
psutil                        5.8.0
PyYAML                         5.1
requests                       2.27.1
scikit-learn                   0.22.1
scipy                         1.5.2
setuptools                     62.6.0
sklearn                       0.0
tensorboardX                    2.0
threadpoolctl                  3.1.0
tomli                         2.0.1
tornado                        6.1
tqdm                          4.64.0
traitlets                     5.3.0
```

 版权所有 © 华为云计算技术有限公司

```
treelib                 1.6.1
urllib3                 1.26.9
wheel                    0.37.1
zipp                    3.8.0
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

 版权所有 © 华为云计算技术有限公司

引擎版本三：mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64

- 镜像地址：swr.{region_id}.myhuaweicloud.com/atelier/mindspore_1_7_0:mindspore_1.7.0-cuda_11.1-py_3.7-ubuntu_18.04-x86_64-20220702120711-8590b76
- 镜像构建时间：20220702120711(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/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pip安装包列表：

```
cycler                        0.11.0
easydict                       1.9
Flask                         2.0.1
grpcio                        1.47.0
gunicorn                       20.1.0
ipykernel                      6.7.0
Jinja2                        3.0.1
lxml                          4.9.0
matplotlib                     3.5.1
mindinsight                    1.7.0
mindspore                      1.7.0
mindvision                     0.1.0
moxing-framework                2.1.0.5d9c87c8
numpy                         1.17.0
opencv-contrib-python-headless   4.6.0.66
opencv-python-headless          4.6.0.66
pandas                         1.1.5
Pillow                        9.1.1
pip                           22.1.2
protobuf                       3.20.1
psutil                        5.8.0
PyYAML                         5.1
requests                       2.27.1
scikit-learn                   0.22.1
scipy                         1.5.2
setuptools                     62.6.0
sklearn                       0.0
tensorboardX                    2.0
threadpoolctl                  3.1.0
tomli                         2.0.1
tornado                        6.1
tqdm                          4.64.0
traitlets                     5.3.0
treelib                       1.6.1
urllib3                       1.26.9
wheel                          0.37.1
zipp                          3.8.0
...
```

- 部分apt安装包列表：

```
apt
ca-certificates
cmake
cuda
curl
ethtool
fdisk
ffmpeg
g++
```

 版权所有 © 华为云计算技术有限公司

```
gcc
git
gpg
graphviz
libsm6
libxext6
libopencv-dev
libxrender-dev
libatlas3-base
libnuma-dev
libcap-dev
libssl-dev
liblz-dev
libbz2-dev
liblzma-dev
libboost-graph-dev
libsndfile1
libcurl4-openssl-dev
libopenblas-base
liblapack3
libopenblas-dev
libprotobuf-dev
libleveldb-dev
libsnappy-dev
libhdf5-serial-dev
liblapacke-dev
libgflags-dev
libgoogle-glog-dev
liblmdb-dev
libatlas-base-dev
librdmacm1
libcap2-bin
libpq-dev
mysql-common
net-tools
nginx
openslide-tools
openssh-client
openssh-server
openssh-sftp-server
openssl
protobuf-compiler
redis-server
redis-tools
rpm
tar
tofrodos
unzip
vim
wget
zip
zlib1g-dev
...
```

 版权所有 © 华为云计算技术有限公司

3 Notebook 中使用自定义镜像

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

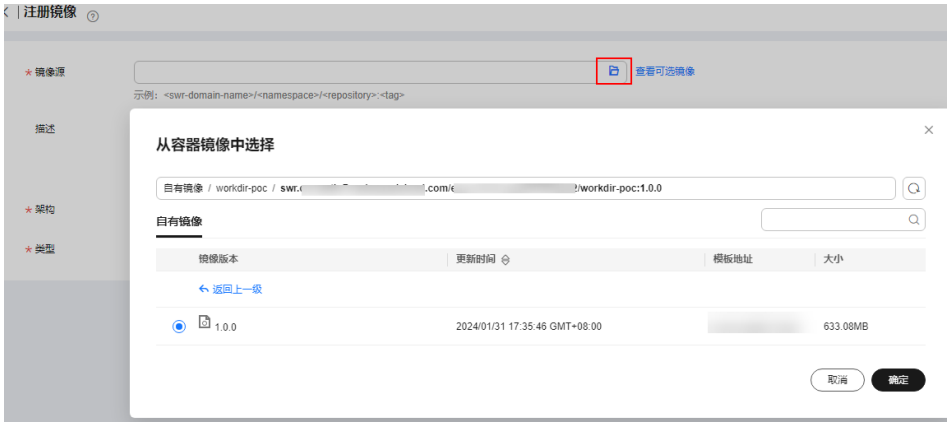
SWR镜像类型设置为“私有”时，同一账号下的子用户（IAM用户）可以注册使用。

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。

     图 3-1 选择镜像源

   - “架构”和“类型”：根据自定义镜像的实际框架选择。

版权所有 © 华为云计算技术有限公司

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 3-2 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 3-3 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 3-4 镜像同步

3.2 Notebook 制作自定义镜像方法

制作自定义镜像有以下方式：

版权所有 © 华为云计算技术有限公司

- 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提供的基础镜像以及镜像构建模板来编写Dockerfile，在Notebook中构建出完全适合自己的镜像。然后将镜像进行注册，用以创建新的开发环境，满足自己的业务需求。详细操作请参考**在Notebook中构建自定义镜像并使用**。
- 方式三：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**ModelArts要求**的新的自定义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ECS上构建自定义镜像并在Notebook中使用**。

3.3 将 Notebook 实例保存为自定义镜像

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。

版权所有 © 华为云计算技术有限公司

图 3-5 保存镜像

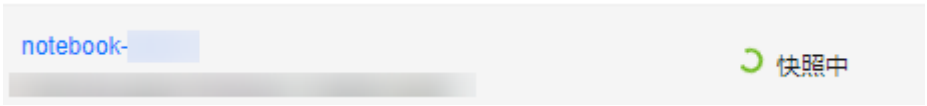
3. 在保存镜像对话框中，设置组织、镜像名称、镜像版本和描述信息。单击“确定”保存镜像。

   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 3-6 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。

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。

版权所有 © 华为云计算技术有限公司

图 3-7 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。

3.4 在 Notebook 中构建自定义镜像并使用

3.4.1 使用场景和构建流程说明

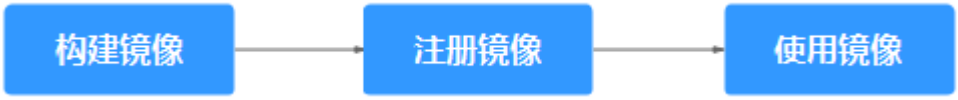
在预置镜像不满足客户使用诉求时，可以基于预置镜像自行构建容器镜像用于开发和训练。

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PyTorch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 3-8 构建镜像流程



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
```

```
(MindSpore) [ma-user work]$ma-cli configure --auth PWD -P yuan
account []: hws
username []:
password:
```


版权所有 © 华为云计算技术有限公司

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。

```
(PyTorch-1.8) [ma-user work]$env|grep -i CURRENT_IMAGE_NAME
CURRENT_IMAGE_NAME=swr.ap-southeast-1.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e
```

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.ap-southeast-1.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e

2. 加载镜像构建模板。

   执行**ma-cli image get-template**命令查询镜像模板。

```
(MindSpore) [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
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
```

   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_current_notebook_apt_packages镜像构建模板。

```
ma-cli image add-template upgrade_current_notebook_apt_packages
```

```
(PyTorch-1.8) [ma-user work]$ma-cli image add-template upgrade_current_notebook_apt_packages
[ OK ] Successfully add configuration template [ upgrade_current_notebook_apt_packages ] under folder [ /home/ma-user/work/.ma/upgrade_current_notebook_apt_packages ]
```

3. 修改Dockerfile。

   本例的Dockerfile将基于PyTorch基础镜像pytorch1.8-cuda10.2-cudnn7-ubuntu18.04进行改造，加载镜像模板upgrade_current_notebook_apt_packages，升级gcc和g++。

   加载镜像模板后，Dockerfile文件自动加载，在“.ma/upgrade_current_notebook_apt_packages”路径下，内容参考如下，根据实际需求修改：

```
FROM swr.ap-southeast-1.myhuaweicloud.com/atelier/pytorch_1_8:pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64-20220926104358-041ba2e

# Set proxy to download internet resources
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

# Config apt source which can accelerate the apt package download speed. Also install ffmpeg and gcc-8 in root mode
RUN cp -f /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 update && \
    apt -y install ffmpeg && \
    apt install -y --no-install-recommends gcc-8 g++-8 && apt-get autoremove -y && \
    update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 80 --slave /usr/bin/g++ g++ /usr/bin/g++-8

# ModelArts requires ma-user as the default user to start image
USER ma-user
```

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_current_notebook_apt_packages/Dockerfile -swr notebook-test/
my_image:0.0.1 -P XXX
```

其中“.ma/upgrade_current_notebook_apt_package/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

3.4.3 Step2 注册新镜像

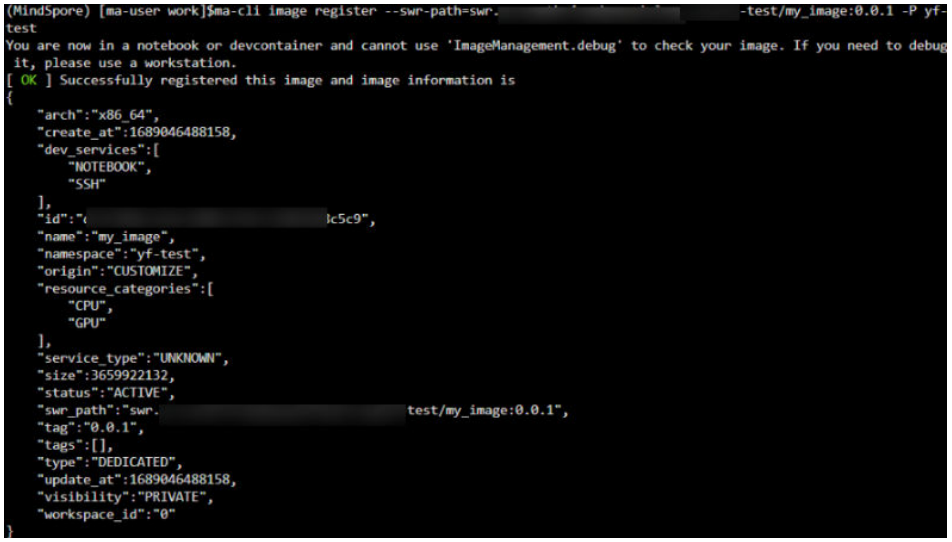
调试完成后，将新镜像注册到ModelArts镜像管理服务中，进而能够在ModelArts中使用该镜像。

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.ap-southeast-1.myhuaweicloud.com/notebook-test/
  my_image:0.0.1 -P XXX
  ```

  图 3-9 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。

3.4.4 Step3 创建开发环境并使用

创建开发环境

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。

版权所有 © 华为云计算技术有限公司

3.5 在 ECS 上构建自定义镜像并在 Notebook 中使用

3.5.1 使用场景和构建流程说明

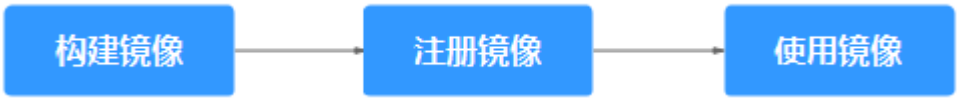
用户在使用ModelArts开发环境时，经常需要对开发环境进行一些改造，如安装、升级或卸载一些包。但是某些包的安装升级需要root权限，运行中的Notebook实例中无root权限，所以在Notebook实例中安装需要root权限的软件，这一点在预置的开发环境镜像中是无法实现的。

此时，用户可以使用ModelArts提供的基础镜像或第三方的镜像来编写Dockerfile，构建出完全适合自己的镜像。然后将镜像进行注册，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ubuntu镜像，安装pytorch 1.8, ffmpeg 3和gcc 8，构建一个面向AI开发的新环境。

主要流程如下图所示：

图 3-10 构建与调测镜像流程



3.5.2 Step1 准备 Docker 机器并配置环境信息

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购买弹性公网IP，并在准备好的机器上安装必要的软件。

ModelArts提供了ubuntu系统的脚本，方便安装docker。

说明

本地Linux机器的操作等同ECS服务器上的操作，请参考本案例。

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；系统盘设置为100GiB。具体操作请参考**购买并登录弹性云服务器**。

版权所有 © 华为云计算技术有限公司

图 3-11 选择镜像和磁盘

- 购买弹性公网IP并绑定到弹性云服务器。具体操作请参考**配置网络**。

配置 VM 环境

1. 在docker机器中，使用如下命令下载安装脚本。

```
wget https://cnnorth4-modelarts-sdk.obs.cn-north-4.myhuaweicloud.com/modelarts/custom-image-build/install_on_ubuntu1804.sh
```

说明

当前仅支持ubuntu系统的脚本。

2. 在docker机器中并执行如下命令，即可完成环境配置。

```
bash install_on_ubuntu1804.sh
```

图 3-12 配置成功

```
source /etc/profile
```

安装脚本依次执行了如下任务：

a. 安装docker。

b. 如果挂载了GPU，则会安装nvidia-docker2，用以将GPU挂载到docker容器中。

3.5.3 Step2 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在Notebook创建实例并使用。关于Dockerfile的具体编写方法，请参考**官网**。

前提条件

已参考**Step1 准备Docker机器并配置环境信息**完成docker机器准备。

查询基础镜像（第三方镜像可跳过此步骤）

ModelArts提供的公共镜像，请参考**Notebook基础镜像列表**，根据预置镜像的引擎类型在对应的章节查看镜像URL。

版权所有 © 华为云计算技术有限公司

制作新镜像

1. 连接容器镜像服务。
   a. 登录容器镜像服务控制台。
   b. 选择左侧导航栏的“总览”，单击页面右上角的“登录指令”，在弹出的页面中单击复制登录指令。

   图 3-13 获取登录指令

   说明

   - 此处生成的登录指令有效期为24小时，若需要长期有效的登录指令，请参见**获取长期有效登录指令**。获取了长期有效的登录指令后，在有效期内的临时登录指令仍然可以使用。
   - 登录指令末尾的域名为镜像仓库地址，请记录该地址，后面会使用到。

   c. 在安装容器引擎的机器中执行上一步复制的登录指令。

   登录成功会显示“Login Succeeded”。

2. 拉取基础镜像或第三方镜像（此处以第三方镜像举例）。
   ```
   docker pull swr.ap-southeast-1.myhuaweicloud.com/notebook-xxx/ubuntu:18.04 #组织名和镜像替换成自己的
   ```

3. 编写Dockerfile。

   vim一个Dockerfile，如果使用的基础镜像是ModelArts提供的公共镜像，Dockerfile的具体内容可参考**Dockerfile文件（基础镜像为ModelArts提供）**。

   如果使用的基础镜像是第三方镜像（非ModelArts提供的公共镜像），Dockerfile文件中需要添加uid为1000的用户ma-user和gid为100的用户组ma-group，具体可参考**Dockerfile文件（基础镜像为非ModelArts提供）**。

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

4. 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，包括{局点信息}/{组织名称}/{镜像名称}:{版本名称}，请根据实际填写。建议使用完整的swr地址，因为后续的调试和注册需要使用。

版权所有 © 华为云计算技术有限公司

- “-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.ap-southeast-1.myhuaweicloud.com/notebook-xxx/pytorch_1_8:v1 -f Dockerfile .
```

图 3-14 构建成功

Dockerfile 文件（基础镜像为 ModelArts 提供）

vim一个Dockerfile文件。基础镜像为ModelArts提供的镜像时，Dockerfile文件的具体内容如下：

```
FROM swr.ap-southeast-1.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/r\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: 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
```

Dockerfile 文件（基础镜像为非 ModelArts 提供）

如果使用的镜像是第三方镜像，Dockerfile文件中需要添加uid为1000的用户ma-user和gid为100的用户组ma-group。如果基础镜像中uid 1000或者gid 100已经被其他用户和用户组占用，需要将其对应的用户和用户组删除。**如下Dockerfile文件已添加指定的用户和用户组，您直接使用即可。**

说明

用户需要设置uid为1000的用户ma-user和gid为100的用户组ma-group，并使/home/ma-user目录权限为750。

vim一个Dockerfile文件，添加第三方镜像（即非ModelArts提供的官方镜像）为基础镜像，如以ubuntu18.04为例。Dockerfile文件的具体内容如下：

```
# Replace it with the actual image version.
FROM ubuntu:18.04
# Set the user ma-user whose UID is 1000 and the user group ma-group whose GID is 100
USER root
RUN default_user=$(getent passwd 1000 | awk -F ':' '{print $1}') || echo "uid: 1000 does not exist" && \
    default_group=$(getent group 100 | awk -F ':' '{print $1}') || echo "gid: 100 does not exist" && \
    if [ ! -z ${default_user} ] && [ ${default_user} != "ma-user" ]; then \
        userdel -r ${default_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 && \
# Grant the read, write, and execute permissions on the target directory to the user ma-user.
chmod -R 750 /home/ma-user

#Configure the APT source and install the ZIP and Wget tools (required for installing conda).
RUN mv /etc/apt/sources.list /etc/apt/sources.list.bak && \
    echo "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 multivers e" > /etc/apt/sources.list && \
apt-get update && \
apt-get install -y zip wget

#Modifying the system Configuration of the image (required for creating the Conda environment)
RUN rm /bin/sh && ln -s /bin/bash /bin/sh

#Switch to user ma-user , download miniconda from the Tsinghua repository, and install miniconda in /
home/ma-user.
USER ma-user
RUN cd /home/ma-user/ && \
    wget --no-check-certificate https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-4.6.14-
Linux-x86_64.sh && \
    bash Miniconda3-4.6.14-Linux-x86_64.sh -b -p /home/ma-user/anaconda3 && \
    rm -rf Miniconda3-4.6.14-Linux-x86_64.sh

#Configure the conda and pip sources
RUN mkdir -p /home/ma-user/.pip && \
    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/r
\n  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2" > /home/ma-user/.condarc && \
    echo -e "[global]\nindex-url = https://pypi.tuna.tsinghua.edu.cn/simple\n[install]\ntrusted-host = https://
pypi.tuna.tsinghua.edu.cn" > /home/ma-user/.pip/pip.conf

#Create the conda environment and install the Python third-party package. The ipykernel package is
mandatory for starting a kernel.
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==1.8.1 torchvision==0.9.1 && \
    pip install ipykernel==6.7.0 && \
    conda init bash && \
    conda deactivate

#Install FFmpeg and GCC
USER root
```

 版权所有 © 华为云计算技术有限公司

```
RUN apt-get -y install ffmpeg && \
    apt -y install gcc-8 g++-8
```

3.5.4 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，进而能够在ModelArts中使用该镜像。

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.ap-southeast-1.myhuaweicloud.com/notebook-xxx/pytorch_1_8:v1
   ```

   完成后即可在SWR上看到该镜像。

   图 3-15 将镜像推到 SWR

2. 注册镜像

   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”，镜像源即为**1.将镜像推到SWR**中推送到SWR中的镜像。

      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。

   说明

   注册镜像时，“架构”和“类型”需要和镜像源保持一致，否则在使用此自定义镜像创建Notebook时会创建失败。

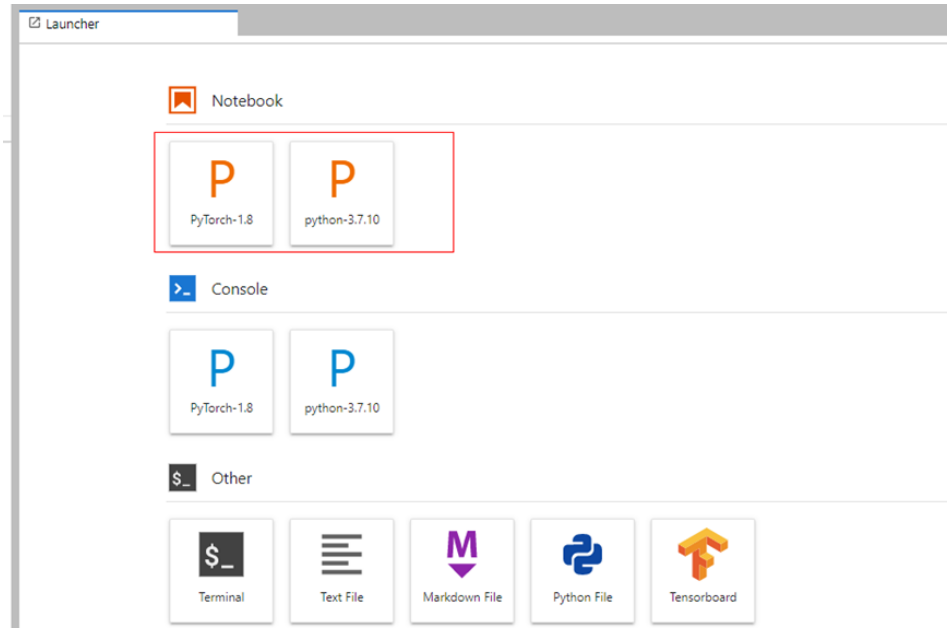
3.5.5 Step4 创建开发环境并使用

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自定义镜像，选中**Step3 注册新镜像**中注册的镜像。
2. Notebook创建成功后，在ModelArts Notebook列表页，单击“打开”，启动该开发环境，启动之后Notebook Launcher界面展示如下：

版权所有 © 华为云计算技术有限公司

图 3-16 打开开发环境



3. 打开一个Terminal，查看conda env环境。conda更多知识可以通过**conda官网**了解。

   开发环境中展示的每个kenrel本质是安装在/home/ma-user/anaconda3/下面的conda env环境。conda env环境可通过命令/home/ma-user/anaconda3/bin/conda env list查看。

图 3-17 查看 conda env 环境

```
(PyTorch-1.8) [ma-user work]$/home/ma-user/anaconda3/bin/conda env list
# 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
```


版权所有 © 华为云计算技术有限公司

4 使用自定义镜像训练模型（模型训练）

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

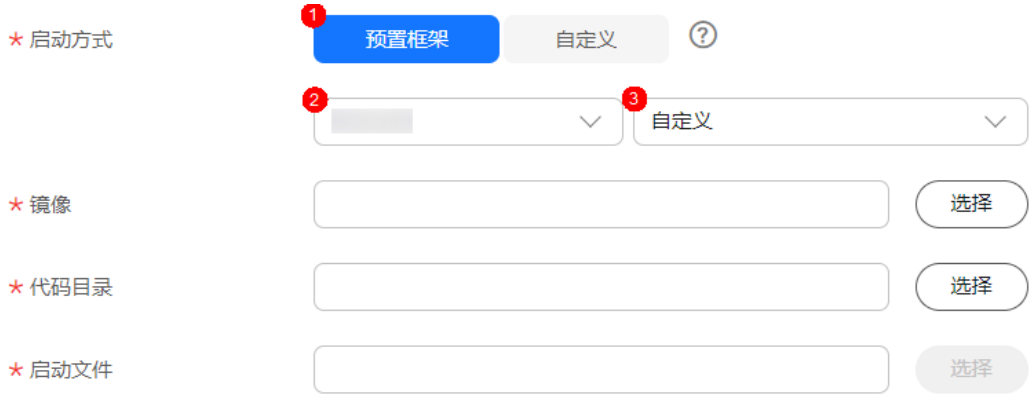
  当使用完全自定义镜像创建训练作业时，“启动命令”必须在“/home/ma-user”目录下执行，否则训练作业可能会运行异常。

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。

 版权所有 © 华为云计算技术有限公司

图 4-1 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 4-2 完全使用自定义镜像创建算法

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**。

版权所有 © 华为云计算技术有限公司

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
```

4.2 示例：从 0 到 1 制作自定义镜像并用于训练

4.2.1 示例：从 0 到 1 制作自定义镜像并用于训练（PyTorch+CPU/GPU）

本章节介绍如何从0到1制作镜像，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PyTorch，训练使用的资源是CPU或GPU。

说明

本实践教程仅适用于新版训练作业。

版权所有 © 华为云计算技术有限公司

场景描述

本示例使用Linux x86_64架构的主机，操作系统ubuntu-18.04，通过编写Dockerfile文件制作自定义镜像。

目标：构建安装如下软件的容器镜像，并在ModelArts平台上使用CPU/GPU规格资源运行训练任务。

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- pytorch-1.8.1

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. **前提条件**
2. **Step1 创建OBS桶和文件夹**
3. **Step2 准备训练脚本并上传至OBS**
4. **Step3 准备镜像主机**
5. **Step4 制作自定义镜像**
6. **Step5 上传镜像至SWR服务**
7. **Step6 在ModelArts上创建训练作业**

前提条件

已注册华为账号并开通华为云，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。

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。需要创建的文件夹列表如**表4-1**所示，示例中的桶名称“test-modelarts”和文件夹名称均为举例，请替换为用户自定义的名称。

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 4-1 OBS 桶文件夹列表

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

版权所有 © 华为云计算技术有限公司

Step2 准备训练脚本并上传至 OBS

准备本案例所需的训练脚本“pytorch-verification.py”文件，并上传至OBS桶的“obs://test-modelarts/pytorch/demo-code/”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)
```

Step3 准备镜像主机

准备一台Linux x86_64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Linux弹性云服务器**。“CPU架构”选择“x86计算”，“镜像”选择“公共镜像”，推荐使用Ubuntu18.04的镜像。

Step4 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-11.1
- python-3.7.13
- pytorch-1.8.1

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 。

1. 安装Docker。

   以Linux x86_64架构的操作系统为例，获取Docker安装包。您可以执行以下指令安装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。

2. 执行如下命令确认Docker Engine版本。

   ```
   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。

3. 准备名为context的文件夹。

   ```
   mkdir -p context
   ```

4. 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。

 版权所有 © 华为云计算技术有限公司

```
[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。

5. 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+cu111-cp37-cp37m-linux_x86_64.whl
   - torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
   - torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl

   说明

   “+”符号的URL编码为“%2B”，在上述网站中搜索目标文件名时，需要将原文件名中的“+”符号替换为“%2B”。

   例如“torch-1.8.1%2Bcu111-cp37-cp37m-linux_x86_64.whl”。

6. 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）。

7. 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+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

8. 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:11.1.1-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

# 复制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+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchvision-0.9.1+cu111-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
```

 版权所有 © 华为云计算技术有限公司

```
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+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchvision-0.9.1+cu111-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:11.1.1-runtime-ubuntu18.04

# 安装 vim和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 中复制 /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**。

9. 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+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

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-cuda11.1。

```
docker build . -t pytorch:1.8.1-cuda11.1
```

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-cuda11.1
```

Step5 上传镜像至 SWR 服务

1. 登录容器镜像服务控制台，选择区域，要和ModelArts区域保持一致，否则无法选择到镜像。
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。
3. 单击右上角“登录指令”，获取登录访问指令，本文选择复制临时登录指令。
4. 以root用户登录本地环境，输入复制的SWR临时登录指令。
5. 上传镜像至容器镜像服务镜像仓库。

 版权所有 © 华为云计算技术有限公司

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 pytorch:1.8.1-cuda11.1 swr.{region-id}.{domain}/deep-learning/pytorch:1.8.1-cuda11.1
```

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-id}.{domain}/deep-learning/pytorch:1.8.1-cuda11.1
```

6. 完成镜像上传后，在容器镜像服务控制台的“我的镜像”页面可查看已上传的自定义镜像。

Step6 在 ModelArts 上创建训练作业

1. 登录ModelArts管理控制台，检查当前账号是否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。
2. 在左侧导航栏中选择“训练管理 > 训练作业”，默认进入“训练作业”列表。
3. 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：**Step5 上传镜像至SWR服务**中创建的镜像。
   - 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/pytorch/demo-code/”，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。
   - 启动命令：“/home/ma-user/miniconda3/bin/python ${MA_JOB_DIR}/demo-code/pytorch-verification.py”，此处的“demo-code”为用户自定义的OBS存放代码路径的最后一级目录，可以根据实际修改。
   - 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/pytorch/log/”
4. 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”。
5. 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，日志信息如下所示。

 版权所有 © 华为云计算技术有限公司

图 4-3 GPU 规格运行日志信息

```
tensor([[-0.4181,  0.8150, -0.2581],
        [-0.6062,  0.5347,  0.1890],
        [ 0.5751,  1.2730, -0.3907],
        [ 0.4812, -0.4064, -0.2753],
        [ 1.0377, -1.1248,  1.2977]])
tensor([[-0.7440, -0.8577, -0.2340],
        [ 0.9569,  0.5516, -1.3350],
        [-1.2878, -0.2791,  0.3486],
        [-1.0997,  0.7627, -0.3188],
        [-1.0865, -1.2626, -0.5900]], device='cuda:0')

```

图 4-4 CPU 规格运行日志信息

```
tensor([[ 0.8945, -0.6946,  0.3807],
        [ 0.6665,  0.3133,  0.8285],
        [-0.5353, -0.1730, -0.5419],
        [ 0.4870,  0.5183,  0.2505],
        [ 0.2679, -0.4996,  0.7919]])
tensor([[ 0.9692,  0.4652,  0.5659],
        [ 2.2032,  1.4157, -0.1755],
        [-0.6296,  0.5466,  0.6994],
        [ 0.2353, -0.0089, -1.9546],
        [ 0.9319,  1.1781, -0.4587]])

```

4.2.2 示例：从 0 到 1 制作自定义镜像并用于训练（MPI+CPU/GPU）

本章节介绍如何从0到1制作镜像，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MPI，训练使用的资源是CPU或GPU。

说明

本实践教程仅适用于新版训练作业。

场景描述

本示例使用Linux x86_64架构的主机，操作系统ubuntu-18.04，通过编写Dockerfile文件制作自定义镜像。

目标：构建安装如下软件的容器镜像，并在ModelArts平台上使用CPU/GPU规格资源运行训练任务。

版权所有 © 华为云计算技术有限公司

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- openmpi-3.0.0

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. **前提条件**
2. **Step1 创建OBS桶和文件夹**
3. **Step2 准备脚本文件并上传至OBS中**
4. **Step3 准备镜像主机**
5. **Step4 制作自定义镜像**
6. **Step5 上传镜像至SWR服务**
7. **Step6 在ModelArts上创建训练作业**

前提条件

已注册华为账号并开通华为云，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。

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。需要创建的文件夹列表如**表4-2**所示，示例中的桶名称“test-modelarts” 和文件夹名称均为举例，请替换为用户自定义的名称。

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 4-2 OBS 桶文件夹列表

文件夹名称	用途
“obs://test-modelarts/mpi/demo-code/”	用于存储MPI启动脚本与训练脚本文件。
“obs://test-modelarts/mpi/log/”	用于存储训练日志文件。

Step2 准备脚本文件并上传至 OBS 中

准备本案例所需的MPI启动脚本run_mpi.sh文件和训练脚本mpi-verification.py文件，并上传至OBS桶的“obs://test-modelarts/mpi/demo-code/”文件夹下。

- MPI启动脚本run_mpi.sh文件内容如下：

```
#!/bin/bash
MY_HOME=/home/ma-user
```

 版权所有 © 华为云计算技术有限公司

```
MY_SSHD_PORT=${MY_SSHD_PORT:-"38888"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: $
{MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: ${MY_MPI_SLOTS}\n"

env | grep -E '^MA_|^SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED' | grep -v '=$' > $
{MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<$MA_NUM_HOSTS; i++))
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .([0-9.]+). .*/\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
            fi
            sleep 1
        done

        echo "[run_mpi] the sshd of ip ${ip} is up"

        echo "${ip} slots=$MY_MPI_SLOTS" >> ${MY_HOME}/hostfile
    done

    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ $MY_TASK_INDEX -eq 0 ]; then

    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca plm_rsh_args \"-
```

 版权所有 © 华为云计算技术有限公司

```
p ${MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... $@"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG -x NCCL_SOCKET_IFNAME -x NCCL_IB_HCA -x NCCL_IB_TIMEOUT -x
NCCL_IB_GID_INDEX -x NCCL_IB_TC \
        -x HOROVOD_MPI_THREADS_DISABLE=1 \
        -x PATH -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"

        sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
        printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

        mpirun \
        --hostfile ${MY_HOME}/hostfile \
        --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -x PATH -x LD_LIBRARY_PATH \
        pkill sleep \
        > /dev/null 2>&1
    fi

    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
    echo "[run_mpi] the training log is in worker-0"
    sleep 365d
    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi

exit $RET_CODE
```

说明

“run_mpi.sh”脚本需要以LF作为换行符。使用CRLF作为换行符会导致训练作业运行失败，日志中会打印“$'\r': command not found”的错误信息。

- 训练脚本mpi-verification.py文件内容如下：

```
import os
import socket

if __name__ == '__main__':
    print(socket.gethostname())

    # https://www.open-mpi.org/faq/?category=running#mpi-environmental-variables
    print('OMPI_COMM_WORLD_SIZE: ' + os.environ['OMPI_COMM_WORLD_SIZE'])
    print('OMPI_COMM_WORLD_RANK: ' + os.environ['OMPI_COMM_WORLD_RANK'])
    print('OMPI_COMM_WORLD_LOCAL_RANK: ' + os.environ['OMPI_COMM_WORLD_LOCAL_RANK'])
```

 版权所有 © 华为云计算技术有限公司

Step3 准备镜像主机

准备一台Linux x86_64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Linux弹性云服务器**。“CPU架构”选择“x86计算”，“镜像”选择“公共镜像”，推荐使用Ubuntu18.04的镜像。

Step4 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-11.1
- python-3.7.13
- openmpi-3.0.0

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。

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。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。

2. 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。

3. 准备名为context的文件夹。

   ```
   mkdir -p context
   ```

4. 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）。

5. 下载openmpi 3.0.0安装文件。

   使用地址https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz，下载 horovod v0.22.1已经编译好的openmpi 3.0.0文件。

6. 将上述Miniconda3安装文件、openmpi 3.0.0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
   └── openmpi-3.0.0-bin.tar.gz
   ```

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
   ```


版权所有 © 华为云计算技术有限公司

```
#
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage-builds
# require Docker Engine >= 17.05
#
# builder stage
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04 AS builder

# 基础容器镜像的默认用户已经是 root
# USER root

# 复制 Miniconda3 (python 3.7.13) 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

# 安装 Miniconda3 到基础容器镜像的 /home/ma-user/miniconda3 目录中
# https://conda.io/projects/conda/en/latest/user-guide/install/linux.html#installing-on-linux
RUN bash /tmp/Miniconda3-py37_4.12.0-Linux-x86_64.sh -b -p /home/ma-user/miniconda3

# 构建最终容器镜像
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

# 安装 vim / curl / net-tools / ssh 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 && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping \
    openssh-client openssh-server && \
    ssh -V && \
    mkdir -p /run/sshd && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# 安装 horovod v0.22.1 已经编译好的 openmpi 3.0.0 文件
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# 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 中复制 /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

# 配置 sshd，使得 ssh 可以免密登录
RUN MA_HOME=/home/ma-user && \
    # setup sshd dir
    mkdir -p ${MA_HOME}/etc && \
    ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N '' -t rsa  && \
    mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run  && \
    # setup sshd config (listen at {{MY_SSHD_PORT}} port)
    echo "Port {{MY_SSHD_PORT}}\n\
HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
```

 版权所有 © 华为云计算技术有限公司

```
PidFile ${MA_HOME}/var/run/sshd.pid\n\
StrictModes no\n\
UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
    # generate ssh key
    ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P '' && \
    cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
    # disable ssh host key checking for all hosts
    echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config
```

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
└── openmpi-3.0.0-bin.tar.gz
```

9. 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mpi:3.0.0-cuda11.1。

```
docker build . -t mpi:3.0.0-cuda11.1
```

构建过程结束时出现如下构建日志说明镜像构建成功。

```
naming to docker.io/library/mpi:3.0.0-cuda11.1
```

Step5 上传镜像至 SWR 服务

1. 登录容器镜像服务控制台，选择区域，要和ModelArts区域保持一致，否则无法选择到镜像。
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。
3. 单击右上角“登录指令”，获取登录访问指令，本文选择复制临时登录指令。
4. 以root用户登录本地环境，输入复制的SWR临时登录指令。
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 mpi:3.0.0-cuda11.1 swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1
   ```

   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.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1
   ```

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/mpi:3.0.0-cuda11.1”即为此自定义镜像的“SWR_URL”。

Step6 在 ModelArts 上创建训练作业

1. 登录ModelArts管理控制台，检查当前账号是否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。
2. 在ModelArts管理控制台，左侧导航栏中选择“训练管理 > 训练作业”，默认进入“训练作业”列表。
3. 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”

 版权所有 © 华为云计算技术有限公司

- 镜像地址：“swr.cn-north-4.myhuaweicloud.com/deep-learning/mpi:3.0.0-cuda11.1”
- 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/mpi/demo-code/”
- 启动命令：bash ${MA_JOB_DIR}/demo-code/run_mpi.sh python ${MA_JOB_DIR}/demo-code/mpi-verification.py
- 环境变量：添加“MY_SSHD_PORT = 38888”
- 资源池：选择公共资源池
- 类型：选择GPU规格
- 计算节点个数：选择“1”或“2”
- 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/mpi/log/”

4. 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”。
5. 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4-5所示。

图 4-5 1 个计算节点 GPU 规格 worker-0 运行日志信息

```
MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 0
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888           0.0.0.0:*               LISTEN      60/sshd
tcp6       0      0 :::38888                :::*                    LISTEN      60/sshd
172.16.0.122 slots=1
modelarts-job-8cf8a682-21cb-4d73-9bb3-789cecdc458b-worker-0
OMPI_COMM_WORLD_SIZE: 1
OMPI_COMM_WORLD_RANK: 0
OMPI_COMM_WORLD_LOCAL_RANK: 0
```

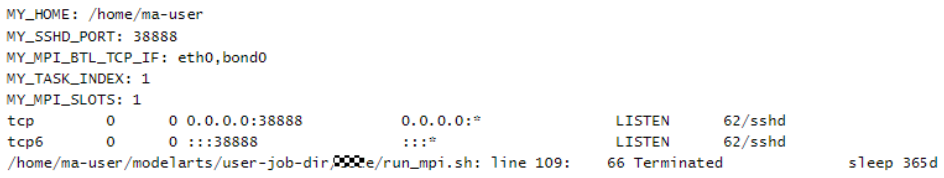
计算节点个数选择为2，训练作业也可以运行。日志信息如图4-6和图4-7所示。

图 4-6 2 个计算节点 worker-0 运行日志信息

```
MY_HOME: /home/ma-user
MY_SSHD_PORT: 38888
MY_MPI_BTL_TCP_IF: eth0,bond0
MY_TASK_INDEX: 0
MY_MPI_SLOTS: 1
tcp        0      0 0.0.0.0:38888           0.0.0.0:*               LISTEN      61/sshd
tcp6       0      0 :::38888                :::*                    LISTEN      61/sshd
172.16.0.39 slots=1
172.16.0.123 slots=1
Warning: Permanently added '[172.16.0.123]:38888' (RSA) to the list of known hosts.
modelarts-job-31732752-6857-4e33-96ff-7a28afae26fb-worker-0
OMPI_COMM_WORLD_SIZE: 2
OMPI_COMM_WORLD_RANK: 0
OMPI_COMM_WORLD_LOCAL_RANK: 0
modelarts-job-31732752-6857-4e33-96ff-7a28afae26fb-worker-1
OMPI_COMM_WORLD_SIZE: 2
OMPI_COMM_WORLD_RANK: 1
OMPI_COMM_WORLD_LOCAL_RANK: 0
```


版权所有 © 华为云计算技术有限公司

图 4-7 2 个计算节点 worker-1 运行日志信息



4.2.3 示例：从 0 到 1 制作自定义镜像并用于训练（Horovod-PyTorch+GPU）

本章节介绍如何从0到1制作镜像，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Horovod 0.22.1 + PyTorch 1.8.1，训练使用的资源是GPU。

说明

本实践教程仅适用于新版训练作业。

场景描述

本示例使用Linux x86_64架构的主机，操作系统ubuntu-18.04，通过编写Dockerfile文件制作自定义镜像。

目标：构建安装如下软件的容器镜像，并在ModelArts平台上使用CPU/GPU规格资源运行训练任务。

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- pytorch-1.8.1
- horovod-0.22.1

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. **前提条件**
2. **Step1 创建OBS桶和文件夹**
3. **Step2 准备训练脚本并上传至OBS**
4. **Step3 准备镜像主机**
5. **Step4 制作自定义镜像**
6. **Step5 上传镜像至SWR服务**
7. **Step6 在ModelArts上创建训练作业**

前提条件

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。

版权所有 © 华为云计算技术有限公司

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。需要创建的文件夹列表如**表4-3**所示，示例中的桶名称“test-modelarts”和文件夹名称均为举例，请替换为用户自定义的名称。

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 4-3 OBS 桶文件夹列表

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

Step2 准备训练脚本并上传至 OBS

准备本案例所需的训练脚本pytorch_synthetic_benchmark.py和run_mpi.sh文件，并上传至OBS桶的“obs://test-modelarts/horovod/demo-code/”文件夹下。

pytorch_synthetic_benchmark.py文件内容如下：

```
import argparse
import torch.backends.cudnn as cudnn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data.distributed
from torchvision import models
import horovod.torch as hvd
import timeit
import numpy as np

# Benchmark settings
parser = argparse.ArgumentParser(description='PyTorch Synthetic Benchmark',
                                 formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('--fp16-allreduce', action='store_true', default=False,
                    help='use fp16 compression during allreduce')

parser.add_argument('--model', type=str, default='resnet50',
                    help='model to benchmark')
parser.add_argument('--batch-size', type=int, default=32,
                    help='input batch size')

parser.add_argument('--num-warmup-batches', type=int, default=10,
                    help='number of warm-up batches that don\'t count towards benchmark')
parser.add_argument('--num-batches-per-iter', type=int, default=10,
                    help='number of batches per benchmark iteration')
parser.add_argument('--num-iters', type=int, default=10,
                    help='number of benchmark iterations')

parser.add_argument('--no-cuda', action='store_true', default=False,
                    help='disables CUDA training')

parser.add_argument('--use-adasum', action='store_true', default=False,
                    help='use adasum algorithm to do reduction')

args = parser.parse_args()
```


版权所有 © 华为云计算技术有限公司

```
args.cuda = not args.no_cuda and torch.cuda.is_available()

hvd.init()

if args.cuda:
    # Horovod: pin GPU to local rank.
    torch.cuda.set_device(hvd.local_rank())

cudnn.benchmark = True

# Set up standard model.
model = getattr(models, args.model)()

# By default, Adasum doesn't need scaling up learning rate.
lr_scaler = hvd.size() if not args.use_adasum else 1

if args.cuda:
    # Move model to GPU.
    model.cuda()
    # If using GPU Adasum allreduce, scale learning rate by local_size.
    if args.use_adasum and hvd.nccl_built():
        lr_scaler = hvd.local_size()

optimizer = optim.SGD(model.parameters(), lr=0.01 * lr_scaler)

# Horovod: (optional) compression algorithm.
compression = hvd.Compression.fp16 if args.fp16_allreduce else hvd.Compression.none

# Horovod: wrap optimizer with DistributedOptimizer.
optimizer = hvd.DistributedOptimizer(optimizer,
                                     named_parameters=model.named_parameters(),
                                     compression=compression,
                                     op=hvd.Adasum if args.use_adasum else hvd.Average)

# Horovod: broadcast parameters & optimizer state.
hvd.broadcast_parameters(model.state_dict(), root_rank=0)
hvd.broadcast_optimizer_state(optimizer, root_rank=0)

# Set up fixed fake data
data = torch.randn(args.batch_size, 3, 224, 224)
target = torch.LongTensor(args.batch_size).random_() % 1000
if args.cuda:
    data, target = data.cuda(), target.cuda()


def benchmark_step():
    optimizer.zero_grad()
    output = model(data)
    loss = F.cross_entropy(output, target)
    loss.backward()
    optimizer.step()


def log(s, nl=True):
    if hvd.rank() != 0:
        return
    print(s, end='\n' if nl else '')


log('Model: %s' % args.model)
log('Batch size: %d' % args.batch_size)
device = 'GPU' if args.cuda else 'CPU'
log('Number of %ss: %d' % (device, hvd.size()))

# Warm-up
log('Running warmup...')
timeit.timeit(benchmark_step, number=args.num_warmup_batches)

# Benchmark
```

 版权所有 © 华为云计算技术有限公司

```
log('Running benchmark...')
img_secs = []
for x in range(args.num_iters):
    time = timeit.timeit(benchmark_step, number=args.num_batches_per_iter)
    img_sec = args.batch_size * args.num_batches_per_iter / time
    log('Iter #%d: %.1f img/sec per %s' % (x, img_sec, device))
    img_secs.append(img_sec)

# Results
img_sec_mean = np.mean(img_secs)
img_sec_conf = 1.96 * np.std(img_secs)
log('Img/sec per %s: %.1f +-%.1f' % (device, img_sec_mean, img_sec_conf))
log('Total img/sec on %d %s(s): %.1f +-%.1f' %
    (hvd.size(), device, hvd.size() * img_sec_mean, hvd.size() * img_sec_conf))
```

run_mpi.sh文件内容如下：

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"36666"}

MY_MPI_BTL_TCP_IF=${MY_MPI_BTL_TCP_IF:-"eth0,bond0"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: $
{MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: ${MY_MPI_SLOTS}\n"

env | grep -E '^MA_|SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED' | grep -v '=$' > $
{MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}

if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<$MA_NUM_HOSTS; i++))
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .([0-9.]+). .*/\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
```

 版权所有 © 华为云计算技术有限公司

```
        fi
        sleep 1
    done

    echo "[run_mpi] the sshd of ip ${ip} is up"

    echo "${ip} slots=$MY_MPI_SLOTS" >> ${MY_HOME}/hostfile
  done

  printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ $MY_TASK_INDEX -eq 0 ]; then

  echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

  np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

  echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca plm_rsh_args \"-p $
{MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... $@"

  # execute mpirun at worker-0
  # mpirun
  mpirun \
    -np ${np} \
    -hostfile ${MY_HOME}/hostfile \
    -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
    -tune ${MY_MPI_TUNE_FILE} \
    -bind-to none -map-by slot \
    -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=${MY_MPI_BTL_TCP_IF} -x
NCCL_SOCKET_FAMILY=AF_INET \
    -x HOROVOD_MPI_THREADS_DISABLE=1 \
    -x LD_LIBRARY_PATH \
    -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
    "$@"

  RET_CODE=$?

  if [ $RET_CODE -ne 0 ]; then
    echo "[run_mpi] exec command failed, exited with $RET_CODE"
  else
    echo "[run_mpi] exec command successfully, exited with $RET_CODE"
  fi

  # stop 1...N worker by killing the sleep proc
  sed -i '1d' ${MY_HOME}/hostfile
  if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
    echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"

    sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

    mpirun \
    --hostfile ${MY_HOME}/hostfile \
    --mca btl_tcp_if_include ${MY_MPI_BTL_TCP_IF} \
    --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
    -x PATH -x LD_LIBRARY_PATH \
    pkill sleep \
    > /dev/null 2>&1
  fi

  echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
  echo "[run_mpi] the training log is in worker-0"
  sleep 365d
  echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi
```

 版权所有 © 华为云计算技术有限公司

```
exit $RET_CODE
```

Step3 准备镜像主机

准备一台Linux x86_64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Linux弹性云服务器**。“CPU架构”选择“x86计算”，“镜像”选择“公共镜像”，推荐使用Ubuntu18.04的镜像。

Step4 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-11.1
- python-3.7.13
- mlnx ofed-5.4
- pytorch-1.8.1
- horovod-0.22.1

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。

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。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。

2. 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。

3. 准备名为context的文件夹。

   ```
   mkdir -p context
   ```

4. 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。

5. 下载horovod源码文件。

   进入网站https://pypi.org/project/horovod/0.22.1/#files，下载horovod-0.22.1.tar.gz文件。

版权所有 © 华为云计算技术有限公司

6. 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+cu111-cp37-cp37m-linux_x86_64.whl
   - torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
   - torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl

   说明

   “+”符号的URL编码为“%2B”；在上述网站中搜索目标文件名时，需要将原文件名中的“+”符号替换为“%2B”。

   例如“torch-1.8.1%2Bcu111-cp37-cp37m-linux_x86_64.whl”。

7. 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）。

8. 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:11.1.1-devel-ubuntu18.04 AS builder

# 安装 cmake 工具（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 && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y build-essential cmake g++-7 && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# 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

# 复制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+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl /tmp
COPY torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl /tmp
COPY openmpi-3.0.0-bin.tar.gz /tmp
COPY horovod-0.22.1.tar.gz /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

# 安装 horovod v0.22.1 已经编译好的 openmpi 3.0.0 文件
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
RUN cd /usr/local && \
```

 版权所有 © 华为云计算技术有限公司

```
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# 编译 Horovod with PyTorch 所需的环境变量
ENV HOROVOD_NCCL_INCLUDE=/usr/include \
    HOROVOD_NCCL_LIB=/usr/lib/x86_64-linux-gnu \
    HOROVOD_MPICXX_SHOW="/usr/local/openmpi/bin/mpicxx -show" \
    HOROVOD_GPU_OPERATIONS=NCCL \
    HOROVOD_WITH_PYTORCH=1

# 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+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl \
    /tmp/torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl

# 使用 Miniconda3 默认 python 环境 (即 /home/ma-user/miniconda3/bin/pip) 编译安装
horovod-0.22.1.tar.gz
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/horovod-0.22.1.tar.gz

# 构建最终容器镜像
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp

# 安装 vim / curl / net-tools / mlnx ofed / ssh 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 && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping libfile-find-rule-perl-perl \
    openssh-client openssh-server && \
    ssh -V && \
    mkdir -p /run/sshd && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-route-3-dev pciutils libnuma1
libpci3 m4 libelf1 debhelper automake graphviz bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-
dev flex chrpath libltdl-dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-space-only --basic --
without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# 安装 horovod v0.22.1 已经编译好的 openmpi 3.0.0 文件
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# 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 中复制 /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
```

 版权所有 © 华为云计算技术有限公司

```
# 设置容器镜像默认用户与工作目录
USER ma-user
WORKDIR /home/ma-user

# 配置 sshd，使得 ssh 可以免密登录
RUN MA_HOME=/home/ma-user && \
    # setup sshd dir
    mkdir -p ${MA_HOME}/etc && \
    ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N '' -t rsa  && \
    mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run  && \
    # setup sshd config (listen at {{MY_SSHD_PORT}} port)
    echo "Port {{MY_SSHD_PORT}}\n\
HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
PidFile ${MA_HOME}/var/run/sshd.pid\n\
StrictModes no\n\
UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
    # generate ssh key
    ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P '' && \
    cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
    # disable ssh host key checking for all hosts
    echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config

# 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=/home/ma-user/miniconda3/bin:$PATH \
    PYTHONUNBUFFERED=1
```

关于Dockerfile文件编写的更多指导内容参见**Docker 官方文档**。

9. 下载MLNX_OFED安装包。

   进入**地址**，单击“Download”，在“Current Versions”和“Archive Versions”中选择合适的安装包下载。本文示例中选择“Archive Versions”，“Version”选择“5.4-3.5.8.0-LTS”，“OS Distribution”选择“Ubuntu”，“OS Distribution Version”选择“Ubuntu 18.04”，“Architecture”选择“x86_64”，下载MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz。

10. 下载openmpi-3.0.0-bin.tar.gz。

    使用地址https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz，下载openmpi-3.0.0-bin.tar.gz文件。

11. 将上述pip源文件、torch*.whl文件、Miniconda3安装文件等放置在context文件夹内，context文件夹内容如下。

```
context
├── Dockerfile
├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── horovod-0.22.1.tar.gz
├── openmpi-3.0.0-bin.tar.gz
├── pip.conf
├── torch-1.8.1+cu111-cp37-cp37m-linux_x86_64.whl
├── torchaudio-0.8.1-cp37-cp37m-linux_x86_64.whl
└── torchvision-0.9.1+cu111-cp37-cp37m-linux_x86_64.whl
```

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horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1。

```
docker build . -t horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
```

    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 horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
```

Step5 上传镜像至 SWR 服务

1. 登录容器镜像服务控制台，选择区域，要和ModelArts区域保持一致，否则无法选择到镜像。

 版权所有 © 华为云计算技术有限公司

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。
3. 单击右上角“登录指令”，获取登录访问指令，本文选择复制临时登录指令。
4. 以root用户登录本地环境，输入复制的SWR临时登录指令。
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 horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1 swr.{region-id}.{domain}/deep-learning/horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
   ```
   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-id}.{domain}/deep-learning/horovod-pytorch:0.22.1-1.8.1-ofed-cuda11.1
   ```
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。

Step6 在 ModelArts 上创建训练作业

1. 登录ModelArts管理控制台，检查当前账号是否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。
2. 在左侧导航栏中选择“训练管理 > 训练作业”，默认进入“训练作业”列表。
3. 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：**Step5 上传镜像至SWR服务**中创建的镜像。
   - 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/pytorch/demo-code/”，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。
   - 启动命令：“bash ${MA_JOB_DIR}/demo-code/run_mpi.sh python ${MA_JOB_DIR}/demo-code/pytorch_synthetic_benchmark.py”，此处的“demo-code”为用户自定义的OBS存放代码路径的最后一级目录，可以根据实际修改。
   - 环境变量：单击“增加环境变量”，增加环境变量：MY_SSHD_PORT=38888。
   - 资源池：选择公共资源池。
   - 资源类型：选择GPU规格。
   - 计算节点个数：1个或者2个。
   - 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/pytorch/log/”
4. 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”。
5. 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。

 版权所有 © 华为云计算技术有限公司

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，日志信息如下所示。

图 4-8 GPU 规格运行日志信息（1 个计算节点）

```
58 Iter #0: 342.4 img/sec per GPU
59 Iter #1: 342.7 img/sec per GPU
60 Iter #2: 342.8 img/sec per GPU
61 Iter #3: 342.5 img/sec per GPU
62 Iter #4: 342.9 img/sec per GPU
63 Iter #5: 342.8 img/sec per GPU
64 Iter #6: 342.9 img/sec per GPU
65 Iter #7: 343.0 img/sec per GPU
66 Iter #8: 342.6 img/sec per GPU
67 Iter #9: 342.7 img/sec per GPU
68 Img/sec per GPU: 342.7 +-0.3
69 Total img/sec on 1 GPU(s): 342.7 +-0.3
```

图 4-9 GPU 规格运行日志信息（2 个计算节点）

```
84 Iter #0: 115.1 img/sec per GPU
85 Iter #1: 123.5 img/sec per GPU
86 Iter #2: 115.7 img/sec per GPU
87 Iter #3: 117.4 img/sec per GPU
88 Iter #4: 120.6 img/sec per GPU
89 Iter #5: 126.9 img/sec per GPU
90 Iter #6: 119.4 img/sec per GPU
91 Iter #7: 118.4 img/sec per GPU
92 Iter #8: 122.0 img/sec per GPU
93 Iter #9: 118.2 img/sec per GPU
94 Img/sec per GPU: 119.7 +-6.8
95 Total img/sec on 2 GPU(s): 239.4 +-13.5
```

4.2.4 示例：从 0 到 1 制作自定义镜像并用于训练（MindSpore +GPU）

本章节介绍如何从0到1制作镜像，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GPU。

说明

本实践教程仅适用于新版训练作业。

版权所有 © 华为云计算技术有限公司

场景描述

本示例使用Linux x86_64架构的主机，操作系统ubuntu-18.04，通过编写Dockerfile文件制作自定义镜像。

目标：构建安装如下软件的容器镜像，并在ModelArts平台上使用GPU规格资源运行训练任务。

- ubuntu-18.04
- cuda-11.1
- python-3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

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：

- **前提条件**
- **Step1 创建OBS桶和文件夹**
- **Step2 创建数据集并上传至OBS**
- **Step3 准备训练脚本并上传至OBS**
- **Step4 准备镜像主机**
- **Step5 制作自定义镜像**
- **Step6 上传镜像至SWR服务**
- **Step7 在ModelArts上创建训练作业**

前提条件

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。

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。需要创建的文件夹列表如**表4-4**所示，示例中的桶名称“test-modelarts” 和文件夹名称均为举例，请替换为用户自定义的名称。

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 4-4 OBS 桶文件夹列表

文件夹名称	用途
“obs://test-modelarts/mindspore-gpu/resnet/”	用于存储训练脚本文件。

版权所有 © 华为云计算技术有限公司

文件夹名称	用途
“obs://test-modelarts/mindspore-gpu/cifar-10-batches-bin/”	用于存储数据集文件。
“obs://test-modelarts/mindspore-gpu/output/”	用于存储训练输出文件。
“obs://test-modelarts/mindspore-gpu/log/”	用于存储训练日志文件。

Step2 创建数据集并上传至 OBS

进入网站http://www.cs.toronto.edu/~kriz/cifar.html，下载“CIFAR-10 binary version (suitable for C programs)”，解压后将数据上传至OBS桶的“obs://test-modelarts/mindspore-gpu/cifar-10-batches-bin/”文件夹下。

Step3 准备训练脚本并上传至 OBS

准备本案例所需的resnet文件和脚本run_mpi.sh，并上传至OBS桶的“obs://test-modelarts/mindspore-gpu/resnet/”文件夹下。

resnet文件下载地址：**https://gitee.com/mindspore/models/tree/r1.8/official/cv/resnet**

run_mpi.sh文件内容如下：

```
#!/bin/bash
MY_HOME=/home/ma-user

MY_SSHD_PORT=${MY_SSHD_PORT:-"36666"}

MY_MPI_BTL_TCP_IF=${MY_MPI_BTL_TCP_IF:-"eth0,bond0"}

MY_TASK_INDEX=${MA_TASK_INDEX:-${VC_TASK_INDEX:-${VK_TASK_INDEX}}}

MY_MPI_SLOTS=${MY_MPI_SLOTS:-"${MA_NUM_GPUS}"}

MY_MPI_TUNE_FILE="${MY_HOME}/env_for_user_process"

if [ -z ${MY_MPI_SLOTS} ]; then
    echo "[run_mpi] MY_MPI_SLOTS is empty, set it be 1"
    MY_MPI_SLOTS="1"
fi

printf "MY_HOME: ${MY_HOME}\nMY_SSHD_PORT: ${MY_SSHD_PORT}\nMY_MPI_BTL_TCP_IF: $
{MY_MPI_BTL_TCP_IF}\nMY_TASK_INDEX: ${MY_TASK_INDEX}\nMY_MPI_SLOTS: ${MY_MPI_SLOTS}\n"

env | grep -E '^MA_|^SHARED_|^S3_|^PATH|^VC_WORKER_|^SCC|^CRED' | grep -v '=$' > $
{MY_MPI_TUNE_FILE}
# add -x to each line
sed -i 's/^/-x /' ${MY_MPI_TUNE_FILE}

sed -i "s|{{MY_SSHD_PORT}}|${MY_SSHD_PORT}|g" ${MY_HOME}/etc/ssh/sshd_config

# start sshd service
bash -c "$(which sshd) -f ${MY_HOME}/etc/ssh/sshd_config"

# confirm the sshd is up
netstat -anp | grep LIS | grep ${MY_SSHD_PORT}
```


版权所有 © 华为云计算技术有限公司

```
if [ $MY_TASK_INDEX -eq 0 ]; then
    # generate the hostfile of mpi
    for ((i=0; i<$MA_NUM_HOSTS; i++))
    do
        eval hostname=${MA_VJ_NAME}-${MA_TASK_NAME}-${i}.${MA_VJ_NAME}
        echo "[run_mpi] hostname: ${hostname}"

        ip=""
        while [ -z "$ip" ]; do
            ip=$(ping -c 1 ${hostname} | grep "PING" | sed -E 's/PING .* .([0-9.]+). .*/\1/g')
            sleep 1
        done
        echo "[run_mpi] resolved ip: ${ip}"

        # test the sshd is up
        while :
        do
            if [ cat < /dev/null >/dev/tcp/${ip}/${MY_SSHD_PORT} ]; then
                break
            fi
            sleep 1
        done

        echo "[run_mpi] the sshd of ip ${ip} is up"

        echo "${ip} slots=$MY_MPI_SLOTS" >> ${MY_HOME}/hostfile
    done

    printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"
fi

RET_CODE=0

if [ $MY_TASK_INDEX -eq 0 ]; then

    echo "[run_mpi] start exec command time: "$(date +"%Y-%m-%d-%H:%M:%S")

    np=$(( ${MA_NUM_HOSTS} * ${MY_MPI_SLOTS} ))

    echo "[run_mpi] command: mpirun -np ${np} -hostfile ${MY_HOME}/hostfile -mca plm_rsh_args \"-p $
{MY_SSHD_PORT}\" -tune ${MY_MPI_TUNE_FILE} ... $@"

    # execute mpirun at worker-0
    # mpirun
    mpirun \
        -np ${np} \
        -hostfile ${MY_HOME}/hostfile \
        -mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -tune ${MY_MPI_TUNE_FILE} \
        -bind-to none -map-by slot \
        -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=${MY_MPI_BTL_TCP_IF} -x
NCCL_SOCKET_FAMILY=AF_INET \
        -x HOROVOD_MPI_THREADS_DISABLE=1 \
        -x LD_LIBRARY_PATH \
        -mca pml ob1 -mca btl ^openib -mca plm_rsh_no_tree_spawn true \
        "$@"

    RET_CODE=$?

    if [ $RET_CODE -ne 0 ]; then
        echo "[run_mpi] exec command failed, exited with $RET_CODE"
    else
        echo "[run_mpi] exec command successfully, exited with $RET_CODE"
    fi

    # stop 1...N worker by killing the sleep proc
    sed -i '1d' ${MY_HOME}/hostfile
    if [ `cat ${MY_HOME}/hostfile | wc -l` -ne 0 ]; then
        echo "[run_mpi] stop 1 to (N - 1) worker by killing the sleep proc"
```

 版权所有 © 华为云计算技术有限公司

```
        sed -i 's/${MY_MPI_SLOTS}/1/g' ${MY_HOME}/hostfile
        printf "[run_mpi] hostfile:\n`cat ${MY_HOME}/hostfile`\n"

        mpirun \
        --hostfile ${MY_HOME}/hostfile \
        --mca btl_tcp_if_include ${MY_MPI_BTL_TCP_IF} \
        --mca plm_rsh_args "-p ${MY_SSHD_PORT}" \
        -x PATH -x LD_LIBRARY_PATH \
        pkill sleep \
        > /dev/null 2>&1
    fi

    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
else
    echo "[run_mpi] the training log is in worker-0"
    sleep 365d
    echo "[run_mpi] exit time: "$(date +"%Y-%m-%d-%H:%M:%S")
fi

exit $RET_CODE
```

“obs://test-modelarts/mindspore-gpu/resnet/”文件夹下包含resnet文件和run_mpi.sh。

Step4 准备镜像主机

准备一台Linux x86_64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Linux弹性云服务器**。“CPU架构”选择“x86计算”，“镜像”选择“公共镜像”，推荐使用Ubuntu18.04的镜像。

Step5 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-11.1
- python-3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

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。

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。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。

2. 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
   ```

 版权所有 © 华为云计算技术有限公司

说明

推荐使用大于等于该版本的Docker Engine来制作自定义镜像。

3. 准备名为context的文件夹。

```
mkdir -p context
```

4. 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。

5. 下载mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl文件。

使用网站**https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.8.1/MindSpore/gpu/x86_64/cuda-11.1/mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl**，mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl文件。

6. 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）。

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:11.1.1-devel-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

# 复制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 mindspore_gpu-1.8.1-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 mindspore whl
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl \
    easydict PyYAML

# 构建最终容器镜像
FROM nvidia/cuda:11.1.1-cudnn8-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp
```

 版权所有 © 华为云计算技术有限公司

```
# 安装 vim / curl / net-tools / mlnx ofed / ssh 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 && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping libfile-find-rule-perl-perl \
    openssh-client openssh-server && \
    ssh -V && \
    mkdir -p /run/sshd && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-route-3-dev pciutils libnuma1
libpci3 m4 libelf1 debhelper automake graphviz bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-
dev flex chrpath libltdl-dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-space-only --basic --
without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# 安装 horovod v0.22.1 已经编译好的 openmpi 3.0.0 文件
# https://github.com/horovod/horovod/blob/v0.22.1/docker/horovod/Dockerfile
# https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz
COPY openmpi-3.0.0-bin.tar.gz /tmp
RUN cd /usr/local && \
    tar -zxf /tmp/openmpi-3.0.0-bin.tar.gz && \
    ldconfig && \
    mpirun --version

# 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 中复制 /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

# 设置容器镜像默认用户与工作目录
USER ma-user
WORKDIR /home/ma-user

# 配置 sshd，使得 ssh 可以免密登录
RUN MA_HOME=/home/ma-user && \
    # setup sshd dir
    mkdir -p ${MA_HOME}/etc && \
    ssh-keygen -f ${MA_HOME}/etc/ssh_host_rsa_key -N '' -t rsa  && \
    mkdir -p ${MA_HOME}/etc/ssh ${MA_HOME}/var/run  && \
    # setup sshd config (listen at {{MY_SSHD_PORT}} port)
    echo "Port {{MY_SSHD_PORT}}\n\
HostKey ${MA_HOME}/etc/ssh_host_rsa_key\n\
AuthorizedKeysFile ${MA_HOME}/.ssh/authorized_keys\n\
PidFile ${MA_HOME}/var/run/sshd.pid\n\
StrictModes no\n\
UsePAM no" > ${MA_HOME}/etc/ssh/sshd_config && \
    # generate ssh key
    ssh-keygen -t rsa -f ${MA_HOME}/.ssh/id_rsa -P '' && \
    cat ${MA_HOME}/.ssh/id_rsa.pub >> ${MA_HOME}/.ssh/authorized_keys && \
    # disable ssh host key checking for all hosts
    echo "Host *\n\
  StrictHostKeyChecking no" > ${MA_HOME}/.ssh/config

# 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=/home/ma-user/miniconda3/bin:$PATH \
```

 版权所有 © 华为云计算技术有限公司

```
LD_LIBRARY_PATH=/usr/local/cuda/lib64:/usr/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH \
PYTHONUNBUFFERED=1
```

关于Dockerfile文件编写的更多指导内容参见**Docker 官方文档**。

8. 下载MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz。

   进入地址https://network.nvidia.com/products/infiniband-drivers/linux/mlnx_ofed/，单击“Download”，“Version”选择“5.4-3.5.8.0-LTS”，“OSDistributionVersion”选择“Ubuntu 18.04”，“Architecture”选择“x86_64”，下载MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz。

9. 下载openmpi-3.0.0-bin.tar.gz。

   使用地址https://github.com/horovod/horovod/files/1596799/openmpi-3.0.0-bin.tar.gz，下载openmpi-3.0.0-bin.tar.gz文件。

10. 将上述Dockerfile文件、Miniconda3安装文件等放置在context文件夹内，context文件夹内容如下。

```
context
├── Dockerfile
├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── mindspore_gpu-1.8.1-cp37-cp37m-linux_x86_64.whl
├── openmpi-3.0.0-bin.tar.gz
└── pip.conf
```

11. 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mindspore:1.8.1-ofed-cuda11.1。

```
docker build . -t mindspore:1.8.1-ofed-cuda11.1
```

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:1.8.1-ofed-cuda11.1
```

Step6 上传镜像至 SWR 服务

1. 登录容器镜像服务控制台，选择区域，要和ModelArts区域保持一致，否则无法选择到镜像。
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。
3. 单击右上角“登录指令”，获取登录访问指令，本文选择复制临时登录指令。
4. 以root用户登录本地环境，输入复制的SWR临时登录指令。
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:1.8.1-ofed-cuda11.1 swr.{region-id}.{domain}/deep-learning/mindspore:1.8.1-ofed-cuda11.1
```

   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-id}.{domain}/deep-learning/mindspore:1.8.1-ofed-cuda11.1
```

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。

Step7 在 ModelArts 上创建训练作业

1. 登录ModelArts管理控制台，检查当前账号是否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。
2. 在左侧导航栏中选择“训练管理 > 训练作业”，默认进入“训练作业”列表。

 版权所有 © 华为云计算技术有限公司

3. 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：**Step6 上传镜像至SWR服务**中创建的镜像。。
   - 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/mindspore-gpu/resnet/”，训练代码会被自动下载至训练容器的“${MA_JOB_DIR}/resnet”目录中，“resnet”为OBS存放代码路径的最后一级目录，可以根据实际修改。
   - 启动命令：“bash ${MA_JOB_DIR}/resnet/run_mpi.sh python ${MA_JOB_DIR}/resnet/train.py”，此处的“resnet”为用户自定义的OBS存放代码路径的最后一级目录，可以根据实际修改。
   - 训练输入：单击“增加训练输入”，参数名称设置为“data_path”，选择OBS中存放数据集的目录，例如“obs://test-modelarts/mindspore-gpu/cifar-10-batches-bin/”，获取方式设置为“超参”。
   - 训练输出：单击“增加训练输出”，参数名称设置为“output_path”，选择OBS存放训练输出的路径，例如：“obs://test-modelarts/mindspore-gpu/output/”，获取方式设置为“超参”，预下载至本地目录设置为“不下载”。
   - 超参：单击“增加超参”，增加如下四个超参：
     - run_distribute=True
     - device_num=1（根据规格中的GPU卡数设置）
     - device_target=GPU
     - epoch_size=2
   - 环境变量：单击“增加环境变量”，增加环境变量：MY_SSHD_PORT=38888。
   - 资源池：选择公共资源池。
   - 资源类型：选择GPU规格。
   - 计算节点个数：1个或者2个。
   - 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/mindspore-gpu/log/”。
4. 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”。
5. 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，日志信息如下所示。

图 4-10 GPU 规格运行日志信息（1 个计算节点）

```
epoch: 1 step: 1875, loss is 1.4800076
Train epoch time: 369422.027 ms, per step time: 197.025 ms
epoch: 2 step: 1875, loss is 1.0306032
Train epoch time: 66996.087 ms, per step time: 35.731 ms
```

 版权所有 © 华为云计算技术有限公司

图 4-11 GPU 规格运行日志信息（2 个计算节点）

```
187 epoch: 1 step: 937, loss is 1.8482083
188 epoch: 1 step: 937, loss is 1.342748
189 Train epoch time: 488492.170 ms, per step time: 521.336 ms
190 Train epoch time: 488490.528 ms, per step time: 521.335 ms
191 epoch: 2 step: 937, loss is 0.9150252
192 Train epoch time: 180200.654 ms, per step time: 192.317 ms
193 epoch: 2 step: 937, loss is 0.9933052
194 Train epoch time: 180199.969 ms, per step time: 192.316 ms
195 172.16.0.45 slots=1
```

4.2.5 示例：从 0 到 1 制作自定义镜像并用于训练（Tensorflow +GPU）

本章节介绍如何从0到1制作镜像，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Tensorflow，训练使用的资源是GPU。

说明

本实践教程仅适用于新版训练作业。

场景描述

本示例使用Linux x86_64架构的主机，操作系统ubuntu-18.04，通过编写Dockerfile文件制作自定义镜像。

目标：构建安装如下软件的容器镜像，并在ModelArts平台上使用GPU规格资源运行训练任务。

- ubuntu-18.04
- cuda-11.2
- python-3.7.13
- mlnx ofed-5.4
- tensorflow gpu-2.10.0

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. **前提条件**
2. **Step1 创建OBS桶和文件夹**
3. **Step2 创建数据集并上传至OBS**
4. **Step3 准备训练脚本并上传至OBS**
5. **Step4 准备镜像主机**
6. **Step5 制作自定义镜像**
7. **Step6 上传镜像至SWR服务**
8. **Step7 在ModelArts上创建训练作业**

版权所有 © 华为云计算技术有限公司

前提条件

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。

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。需要创建的文件夹列表如**表4-5**所示，示例中的桶名称“test-modelarts” 和文件夹名称均为举例，请替换为用户自定义的名称。

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 4-5 OBS 桶文件夹列表

文件夹名称	用途
“obs://test-modelarts/tensorflow/code/”	用于存储训练脚本文件。
“obs://test-modelarts/tensorflow/data/”	用于存储数据集文件。
“obs://test-modelarts/tensorflow/log/”	用于存储训练日志文件。

Step2 创建数据集并上传至 OBS

使用网站**https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz**，下载“mnist.npz”文件并上传至OBS桶的“obs://test-modelarts/tensorflow/data/”文件夹下。

Step3 准备训练脚本并上传至 OBS

准备本案例所需的训练脚本mnist.py，并上传至OBS桶的“obs://test-modelarts/tensorflow/code/”文件夹下。

mnist.py文件内容如下：

```
import argparse
import tensorflow as tf

parser = argparse.ArgumentParser(description='TensorFlow quick start')
parser.add_argument('--data_url', type=str, default="./Data", help='path where the dataset 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)
```


版权所有 © 华为云计算技术有限公司

```
])

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)
```

Step4 准备镜像主机

准备一台Linux x86_64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Linux弹性云服务器**。“CPU架构”选择“x86计算”，“镜像”选择“公共镜像”，推荐使用Ubuntu18.04的镜像。

Step5 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-11.1
- python-3.7.13
- mlnx ofed-5.4
- mindspore gpu-1.8.1

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。

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。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。

2. 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。

3. 准备名为context的文件夹。

   ```
   mkdir -p context
   ```

4. 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
   ```

 版权所有 © 华为云计算技术有限公司

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。

5. 下载tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl文件。

   使用网站https://pypi.org/project/tensorflow-gpu/2.10.0/#files，下载tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl文件。

6. 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）。

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:11.2.2-cudnn8-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

# 复制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 tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_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 tensorflow whl
RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir \
    /tmp/tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl

RUN cd /tmp && \
    /home/ma-user/miniconda3/bin/pip install --no-cache-dir keras==2.10.0

# 构建最终容器镜像
FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu18.04

COPY MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz /tmp

# 安装 vim / curl / net-tools / mlnx ofed（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 && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install -y vim curl net-tools iputils-ping && \
    # mlnx ofed
    apt-get install -y python libfuse2 dpatch libnl-3-dev autoconf libnl-route-3-dev pciutils libnuma1
libpci3 m4 libelf1 debhelper automake graphviz bison lsof kmod libusb-1.0-0 swig libmnl0 autotools-
```

 版权所有 © 华为云计算技术有限公司

```
dev flex chrpath libltdl-dev && \
    cd /tmp && \
    tar -xvf MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz && \
    MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64/mlnxofedinstall --user-space-only --basic --
without-fw-update -q && \
    cd - && \
    rm -rf /tmp/* && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf

# 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 中复制 /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

# 设置容器镜像默认用户与工作目录
USER ma-user
WORKDIR /home/ma-user

# 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=/home/ma-user/miniconda3/bin:$PATH \
    LD_LIBRARY_PATH=/usr/local/cuda/lib64:/usr/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH \
    PYTHONUNBUFFERED=1
```

关于Dockerfile文件编写的更多指导内容参见**Docker 官方文档**。

8. 下载MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz。

   进入**地址**，单击“Download”，“Version”选择“5.4-3.5.8.0-LTS”，“OSDistributionVersion”选择“Ubuntu 18.04”，“Architecture”选择“x86_64”，下载MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz。

9. 将上述Dockerfile文件、 Miniconda3 安装文件等放置在context文件夹内，context文件夹内容如下。

```
context
├── Dockerfile
├── MLNX_OFED_LINUX-5.4-3.5.8.0-ubuntu18.04-x86_64.tgz
├── Miniconda3-py37_4.12.0-Linux-x86_64.sh
├── pip.conf
└── tensorflow_gpu-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```

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tensorflow:2.10.0-ofed-cuda11.2。

```
docker build . -t tensorflow:2.10.0-ofed-cuda11.2
```

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 tensorflow:2.10.0-ofed-cuda11.2
```

Step6 上传镜像至 SWR 服务

1. 登录容器镜像服务控制台，选择区域，要和ModelArts区域保持一致，否则无法选择到镜像。
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。
3. 单击右上角“登录指令”，获取登录访问指令，本文选择复制临时登录指令。
4. 以root用户登录本地环境，输入复制的SWR临时登录指令。
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。

 版权所有 © 华为云计算技术有限公司

```
#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 tensorflow:2.10.0-ofed-cuda11.2 swr.{region-id}.{domain}/deep-learning/
tensorflow:2.10.0-ofed-cuda11.2
```

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-id}.{domain}/deep-learning/tensorflow:2.10.0-ofed-cuda11.2
```

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。

Step7 在 ModelArts 上创建训练作业

1. 登录ModelArts管理控制台，检查当前账号是否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。
2. 在左侧导航栏中选择“训练管理 > 训练作业”，默认进入“训练作业”列表。
3. 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：**Step5 制作自定义镜像**中创建的镜像。。
   - 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/tensorflow/code/”，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存放代码路径的最后一级目录，可以根据实际修改。
   - 启动命令：“python ${MA_JOB_DIR}/code/mnist.py”，此处的“code”为用户自定义的OBS存放代码路径的最后一级目录，可以根据实际修改。
   - 训练输入：单击“增加训练输入”，参数名称设置为“data_path”，选择OBS中存放“mnist.npz”的目录，例如“obs://test-modelarts/tensorflow/data/mnist.npz”，获取方式设置为“超参”。
   - 资源池：选择公共资源池。
   - 资源类型：选择GPU规格。
   - 计算节点个数：1个。
   - 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/mindspore-gpu/log/”。
4. 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”。
5. 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，日志信息如下所示。

 版权所有 © 华为云计算技术有限公司

图 4-12 GPU 规格运行日志信息

```
    0.9767
323 1503/1875 [=======================>......] - ETA: 0s - loss: 0.0741 - accuracy:
    0.9769
324 1533/1875 [=======================>......] - ETA: 0s - loss: 0.0743 - accuracy:
    0.9769
325 1564/1875 [========================>.....] - ETA: 0s - loss: 0.0746 - accuracy:
    0.9768
326 1595/1875 [========================>.....] - ETA: 0s - loss: 0.0741 - accuracy:
    0.9770
327 1624/1875 [========================>.....] - ETA: 0s - loss: 0.0742 - accuracy:
    0.9770
328 1654/1875 [=========================>....] - ETA: 0s - loss: 0.0745 - accuracy:
    0.9770
329 1685/1875 [=========================>....] - ETA: 0s - loss: 0.0747 - accuracy:
    0.9768
330 1716/1875 [=========================>...] - ETA: 0s - loss: 0.0752 - accuracy:
    0.9767
331 1747/1875 [==========================>...] - ETA: 0s - loss: 0.0755 - accuracy:
    0.9767
332 1778/1875 [==========================>..] - ETA: 0s - loss: 0.0753 - accuracy:
    0.9767
333 1809/1875 [==========================>..] - ETA: 0s - loss: 0.0751 - accuracy:
    0.9768
334 1841/1875 [==========================>.] - ETA: 0s - loss: 0.0753 - accuracy:
    0.9767
335 1872/1875 [==========================>.] - ETA: 0s - loss: 0.0753 - accuracy:
    0.9767
336 1875/1875 [==============================] - 3s 2ms/step - loss: 0.0752 - accuracy: 0.9767
```

4.3 准备训练镜像

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。

版权所有 © 华为云计算技术有限公司

- 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。

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**）。

版权所有 © 华为云计算技术有限公司

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.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。训练基础镜像地址请参见**训练基础镜像列表**。

   ```
   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。

4.3.4 在容器镜像中安装 MLNX_OFED

场景描述

ModelArts GPU服务器上配置了Mellanox Technologies网卡，支持RDMA（Remote Direct Memory Access）。因此可以在容器镜像中安装MLNX_OFED，使得**NCCL**可以启用该网卡，提高跨节点通信效率。

NCCL启用该网卡后，跨节点通信采用的方法为NET/IB。未启用该网卡时，跨节点通信采用的方法为NET/Socket。NET/IB在时延与带宽方面都要优于NET/Socket。

版权所有 © 华为云计算技术有限公司

表 4-6 ModelArts GPU 服务器 Mellanox Technologies 网卡和 MLNX_OFED 安装情况

服务器GPU型号	Mellanox Technologies网卡	服务器安装的MLNX_OFED版本	推荐容器镜像安装的MLNX_OFED版本
Vnt1	ConnectX-5	4.3-1.0.1.0/4.5-1.0.1.0	4.9-6.0.6.0-LTS
Ant8/Ant1	ConnectX-6 Dx	5.5-1.0.3.2	5.8-2.0.3.0-LTS

安装 MLNX_OFED

以Ubuntu18.04的容器镜像为例，安装MLNX_OFED 4.9-6.0.6.0-LTS的Dockerfile示例如下。

说明

Dockerfile中涉及文件下载，构建容器镜像的主机要求能够连通公网。

```
FROM nvidia/cuda:11.1.1-runtime-ubuntu18.04

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 && \
    echo > /etc/apt/apt.conf.d/00skip-verify-peer.conf "Acquire { https::Verify-Peer false }" && \
    apt-get update && \
    apt-get install --no-install-recommends -y lsb-core curl && \
    curl -k -o /tmp/MLNX_OFED_LINUX-4.9-6.0.6.0-ubuntu18.04-x86_64.tgz https://content.mellanox.com/
ofed/MLNX_OFED-4.9-6.0.6.0/MLNX_OFED_LINUX-4.9-6.0.6.0-ubuntu18.04-x86_64.tgz && \
    cd /tmp && \
    tar xzf MLNX_OFED_LINUX-4.9-6.0.6.0-ubuntu18.04-x86_64.tgz && \
    cd MLNX_OFED_LINUX-4.9-6.0.6.0-ubuntu18.04-x86_64 && \
    ./mlnxofedinstall --user-space-only --without-fw-update --without-neohost-backend --force && \
    rm /tmp/MLNX_OFED_LINUX-4.9-6.0.6.0-ubuntu18.04-x86_64.tgz && \
    rm -rf /tmp/MLNX_OFED_LINUX-4.9-6.0.6.0-ubuntu18.04-x86_64 && \
    apt-get clean && \
    mv /etc/apt/sources.list.bak /etc/apt/sources.list && \
    rm /etc/apt/apt.conf.d/00skip-verify-peer.conf
```

构建容器镜像命令示例如下：

```
docker build -f Dockerfile . -t nvidia/cuda:mlnx-ofed-4.9-11.1.1-runtime-ubuntu18.04
```

构建完成后，执行如下命令可以查询容器镜像中的MLNX_OFED版本。

```
docker run -ti --rm nvidia/cuda:mlnx-ofed-4.9-11.1.1-runtime-ubuntu18.04 ofed_info | head -n 1
```

命令回显示例如下：

```
MLNX_OFED_LINUX-4.9-6.0.6.0 (OFED-4.9-6.0.6):
```

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：

版权所有 © 华为云计算技术有限公司

- 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 > 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 4-7 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。

版权所有 © 华为云计算技术有限公司

- 输入配置

表 4-8 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 4-9 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。

版权所有 © 华为云计算技术有限公司

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。

   图 4-13 添加超参

2. 编辑超参

   超参的参数说明参见表4-10。

   表 4-10 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、中文、数字、空格、中划线、下划线、中英文逗号和中英文句号。

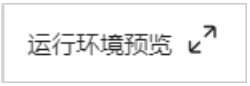
添加训练约束

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。

版权所有 © 华为云计算技术有限公司

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）**。

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 4-11 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”。

版权所有 © 华为云计算技术有限公司

参数名称	说明
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。
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。

版权所有 © 华为云计算技术有限公司

参数名称	说明
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 4-12 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。

版权所有 © 华为云计算技术有限公司

参数名称	子参数	说明

	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。
	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。

版权所有 © 华为云计算技术有限公司

参数名称	子参数	说明
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**。

4.6 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**使用ModelArts的基础镜像构建新的训练镜像**。
   - 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**训练作业日志中提示“No such file or directory”**。
   - 提示找不到包等错误，请参见**训练作业日志中提示“No module named .*”**。
   - 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。

版权所有 © 华为云计算技术有限公司

- 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**基础镜像**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**训练故障案例**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**训练作业自定义镜像规范**制作镜像。文档中同时提供了端到端的示例供用户参考。

 版权所有 © 华为云计算技术有限公司

5 使用自定义镜像创建 AI 应用（推理部署）

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50GB。
- 对于同步请求模式的AI应用，如果预测请求时延超过60s，会造成请求失败，甚至会有服务业务中断的风险，预测请求时延超过60s时，建议制作异步请求模式的镜像。
- **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 5-1 listen_ip 获取示例

    ```
    root@[redacted]:/# cat /etc/hosts
    127.0.0.1       localhost
    ::1     localhost ip6-localhost ip6-loopback
    fe00::0 ip6-localnet
    ff00::0 ip6-mcastprefix
    ff02::1 ip6-allnodes
    ff02::2 ip6-allrouters
    169.254.30.2    d6211431d0e3
    ```

  - 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。

 版权所有 © 华为云计算技术有限公司

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 5-1 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
```

 版权所有 © 华为云计算技术有限公司

```
    wait $gunicorn_pid          # 等待gunicorn进程完全终止
  fi
}

trap handle_sigterm TERM
```

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。

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。

图 5-2 创建 ECS 服务器-选择 X86 架构的公共镜像

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

版权所有 © 华为云计算技术有限公司

```
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

# 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。

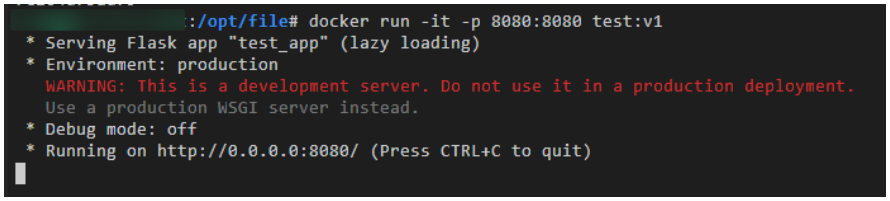
本地验证镜像并上传镜像至 SWR 服务

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
```

 版权所有 © 华为云计算技术有限公司

图 5-3 启动自定义镜像

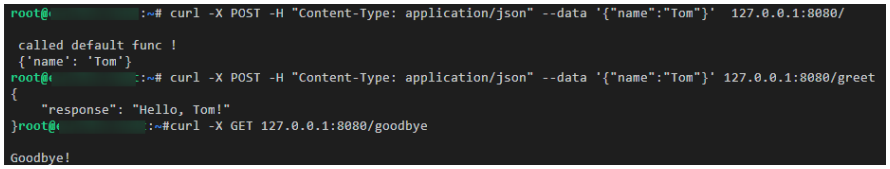


2. 另开一个终端，执行以下命令验证自定义镜像的三个API接口功能。

```
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 5-4 校验接口



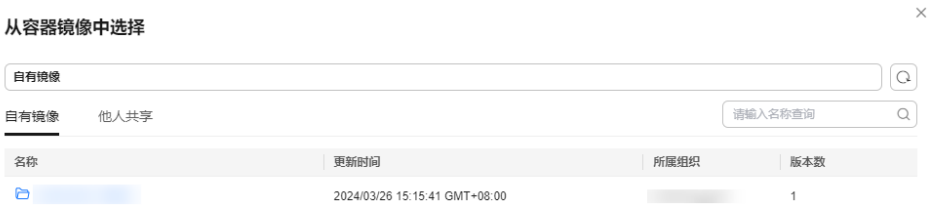
3. 上传自定义镜像至SWR服务。上传镜像的详细操作可参考**如何登录并上传镜像到SWR**。
4. 完成自定义镜像上传后，您可以在“容器镜像服务>我的镜像>自有镜像”列表中看到已上传镜像。

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 5-5 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": {
```

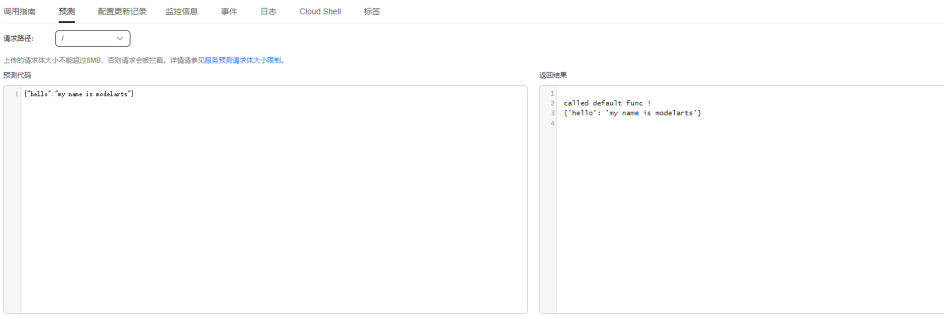

版权所有 © 华为云计算技术有限公司

```
            "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。
3. 您可以通过“预测”页签访问在线服务。

图 5-6 访问在线服务



 版权所有 © 华为云计算技术有限公司

6 FAQ

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 6-1 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
   ```

 版权所有 © 华为云计算技术有限公司

图 6-2 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”。

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**。

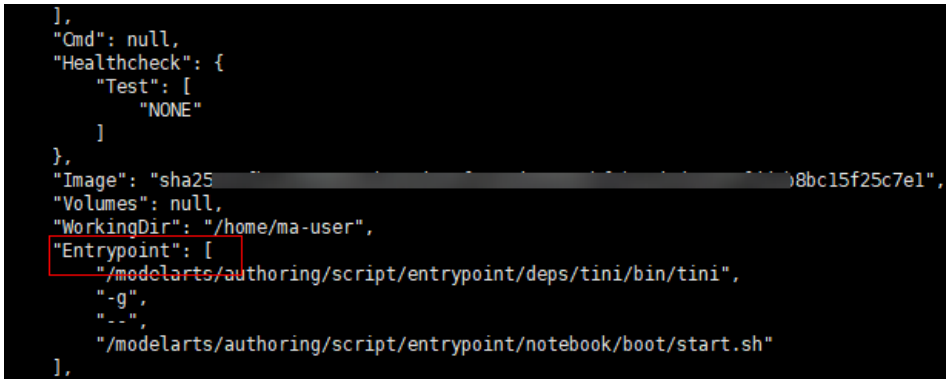
6.3 如何通过 docker 启动 Notebook 保存后的镜像

Notebook保存后的镜像有Entrypoint参数，如**图6-3**。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6-4**。

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：

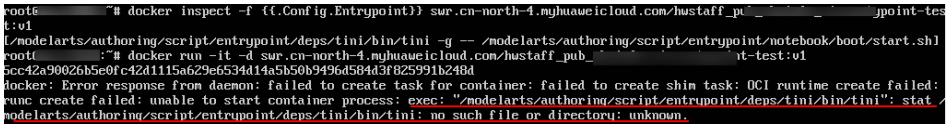
```
docker run -it -d --entrypoint /bin/bash image:tag
```

图 6-3 Entrypoint 参数



 版权所有 © 华为云计算技术有限公司

图 6-4 启动镜像报错



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 6-1 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
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

 版权所有 © 华为云计算技术有限公司

命令说明	命令
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**。

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。

 版权所有 © 华为云计算技术有限公司

7 修订记录

发布时间	修改内容
2023-10-01	新增内容如下： ● 新增章节**预置框架启动流程说明**。
2023-09-07	新增内容如下： ● 新增章节**自定义镜像软件版本匹配注意事项** 修改内容如下： ● “使用自定义镜像”手册更名为“镜像管理”。 ● 预置镜像结构调整，开发环境，训练，推理基础镜像整合为**使用预置镜像**。 ● 镜像保存故障处理FAQ移到**“故障排除 > 开发环境 > 镜像保存故障”**章节。

版权所有 © 华为云计算技术有限公司