

**人工智能实验报告**

题 目 基于Mindspore框架与

ModelArts 平台的MNIST

手写体识别实验

专 业 计算学部

学　　 号 120L022109

姓 名 李世轩

同 组 人 员

1. **背景简介/问题描述**

该实验包含了3部分内容：

基于Mindspore框架的模型本地训练及预测

基于Modelarts平台和Tensorflow框架的模型训练及部署

基于Modelarts平台和Mindspore框架的模型训练

1. **算法介绍**

本例子会实现一个简单的图片分类的功能，整体流程如下：

1、处理需要的数据集，这里使用了MNIST数据集。

1、 定义一个网络，这里我们使用LeNet网络。

2、 定义损失函数和优化器。

3、 加载数据集并进行训练，训练完成后，查看结果及保存模型文件。

4、 加载保存的模型，进行推理。

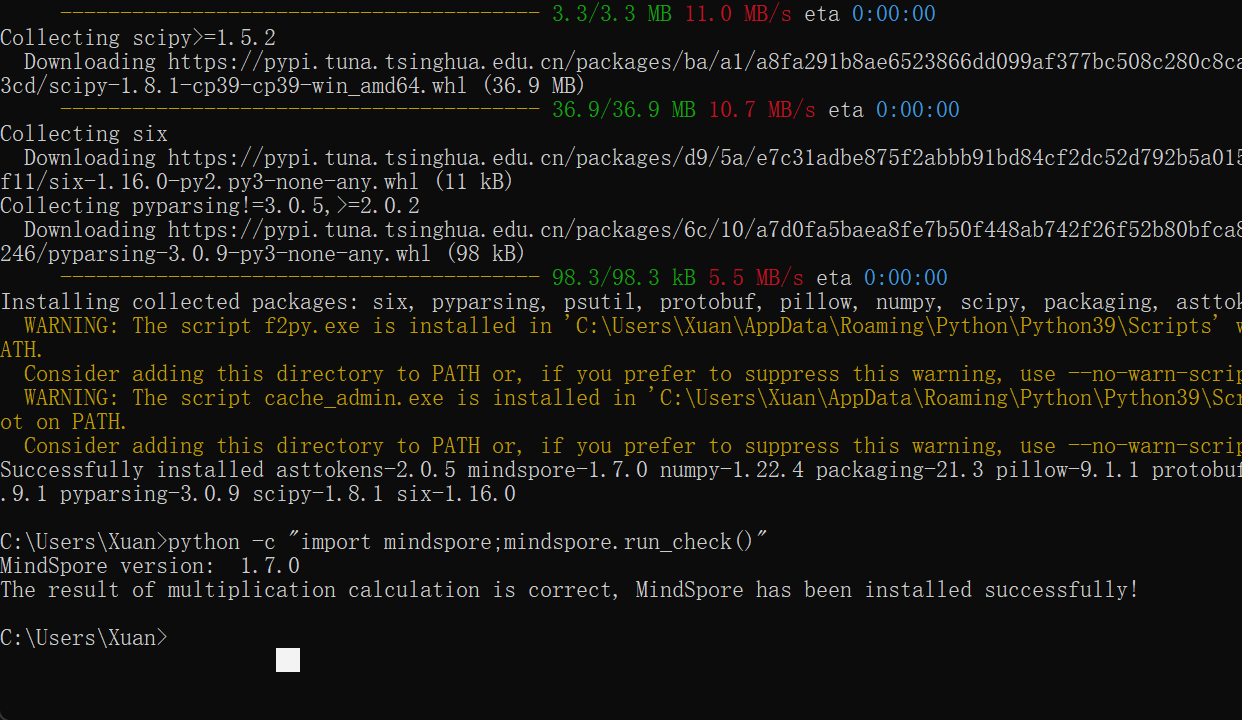
验证模型，加载测试数据集和训练后的模型，验证结果精度。

1. **算法实现**

## 基于Mindspore框架的模型本地训练及预测

实验准备

安装minespore



（图1.1-安装minespore）

1. 导入Python库&模块并配置运行信息

在使用前，导入需要的Python库。

目前使用到os库，为方便理解，其他需要的库，我们在具体使用到时再说明。

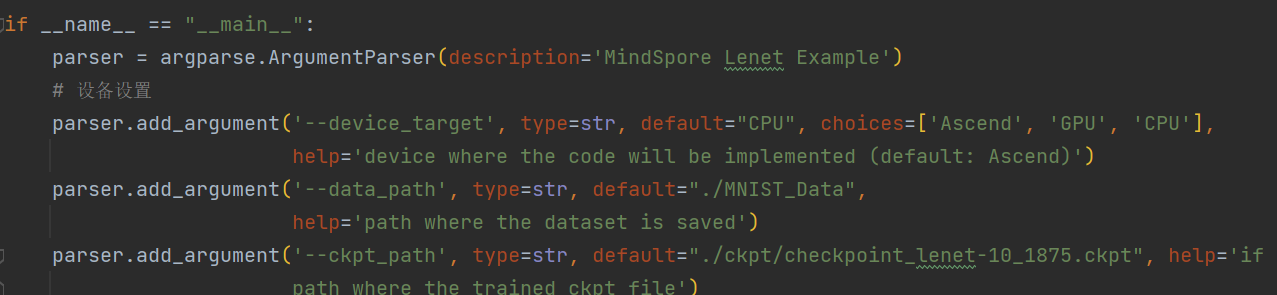
详细的MindSpore的模块说明，可以在MindSpore API页面中搜索查询。

可以通过context.set\_context来配置运行需要的信息，譬如运行模式、后端信息、硬件等信息。

导入context模块，配置运行需要的信息。

parser.add\_argument('--device\_target', type=str, default=" Ascend", choices=['Ascend', 'GPU', 'CPU'], help='device where the code will be implemented (default: Ascend)')

在样例中我们配置样例运行使用图模式。再train.py和eval.py根据实际情况配置硬件信息，譬如代码运行在Ascend AI处理器上，则device\_target选择Ascend，代码运行在CPU、GPU同理。详细参数说明，请参见context.set\_context接口说明。



（图1.2-配置运行信息）

1. 数据处理

#根据数据集存储地址，生成数据集，构建训练、验证函数

ds\_train = create\_dataset(os.path.join(args.data\_path, "train"), cfg.batch\_size)

ds\_eval = create\_dataset(os.path.join(args.data\_path, "test"), cfg.batch\_size,1)

数据集对于训练非常重要，好的数据集可以有效提高训练精度和效率。在加载数据集前，我们通常会对数据集进行一些处理。步骤二的以下内容里，mindspore 提供了更多数据处理的方法，供大家以后使用mindspore时参考：

/\*--------定义数据集及数据操作

我们定义一个函数create\_dataset来创建数据集。在这个函数中，我们定义好需要进行的数据增强和处理操作：

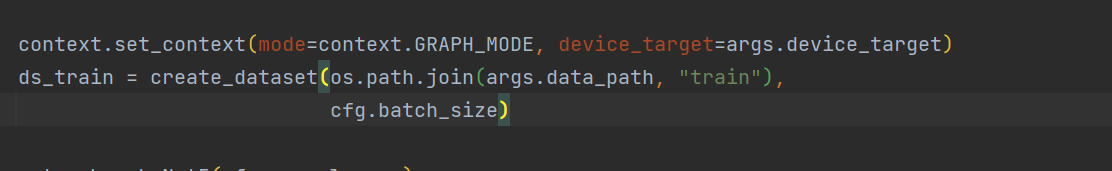
1. 定义数据集。

2. 定义进行数据增强和处理所需要的一些参数。

3. 根据参数，生成对应的数据增强操作。

4. 使用map映射函数，将数据操作应用到数据集。

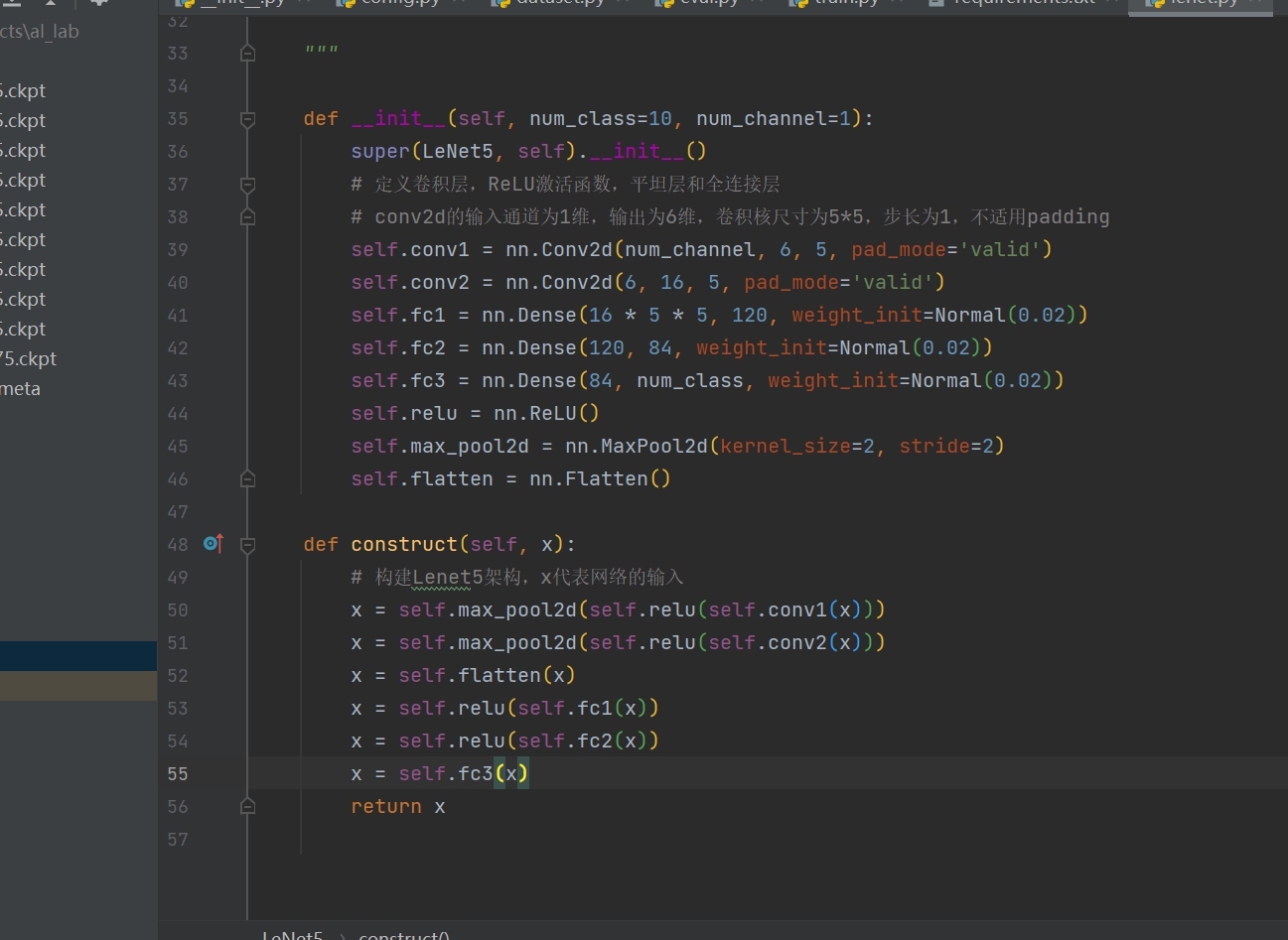
5. 对生成的数据集进行处理。



（图1.3-数据处理）

1. 定义网络

我们选择相对简单的LeNet网络。LeNet网络不包括输入层的情况下，共有7层：2个卷积层、2个下采样层（池化层）、3个全连接层。每层都包含不同数量的训练参数，如下图所示：



（图1.4-定义网络）

1. 定义损失函数及优化器

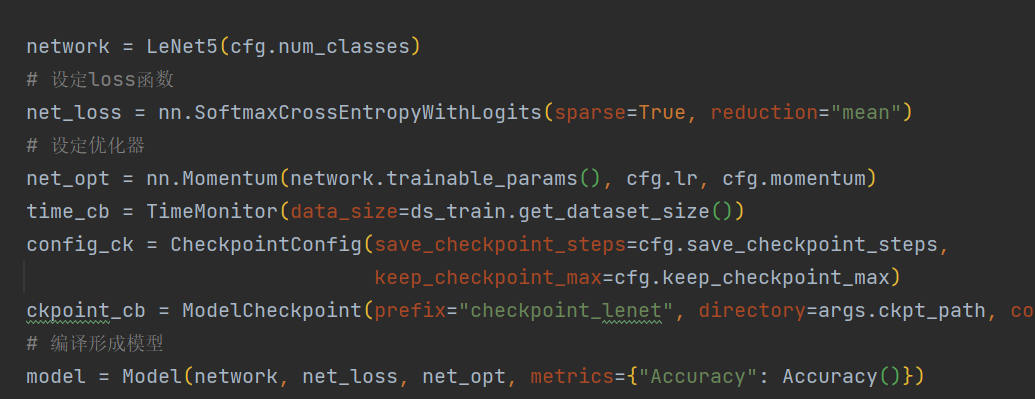
在进行定义之前，先简单介绍损失函数及优化器的概念。

损失函数：又叫目标函数，用于衡量预测值与实际值差异的程度。深度学习通过不停地迭代来缩小损失函数的值。定义一个好的损失函数，可以有效提高模型的性能。

优化器：用于最小化损失函数，从而在训练过程中改进模型。

定义了损失函数后，可以得到损失函数关于权重的梯度。梯度用于指示优化器优化权重的方向，以提高模型性能。

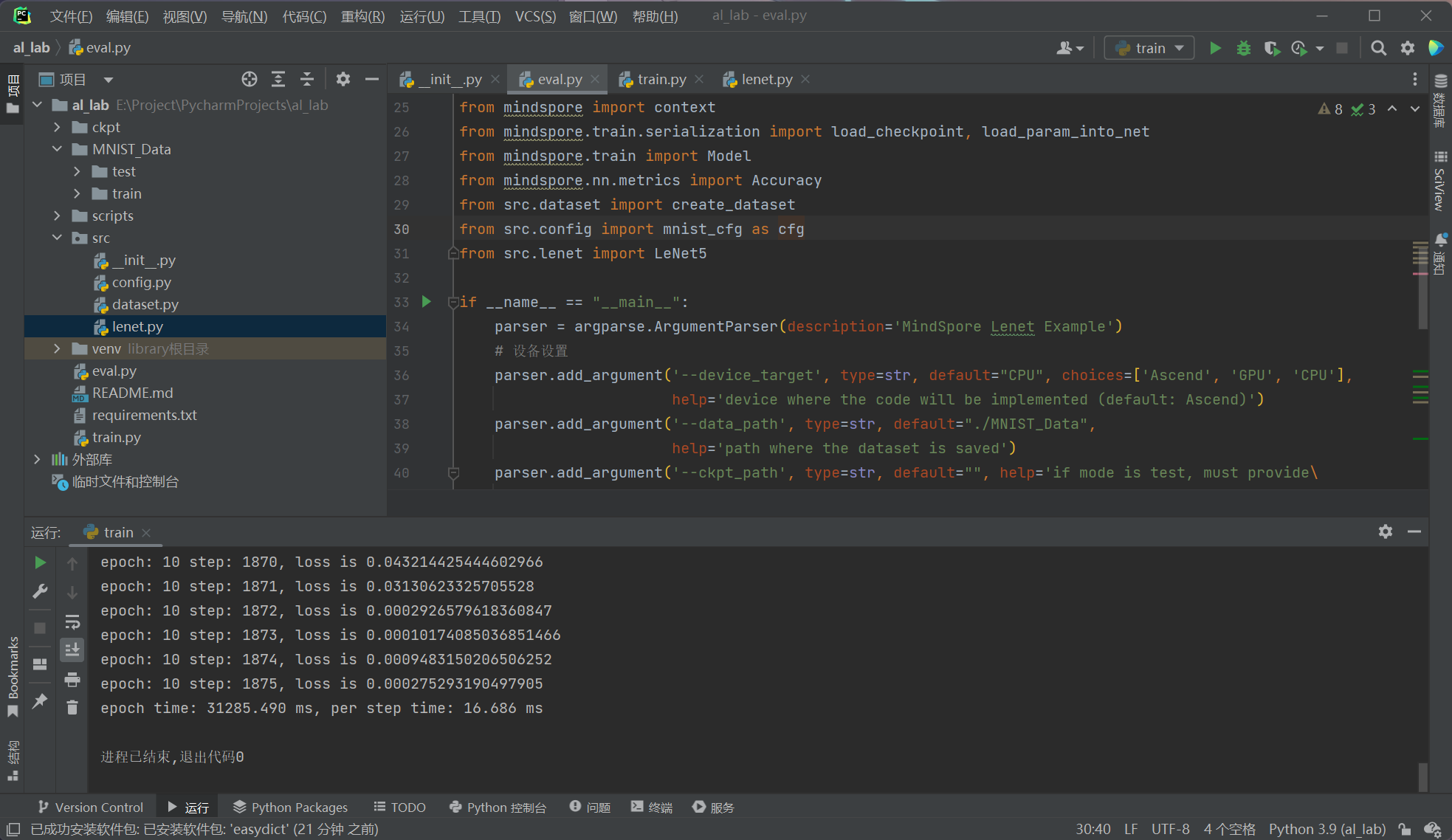
定义损失函数



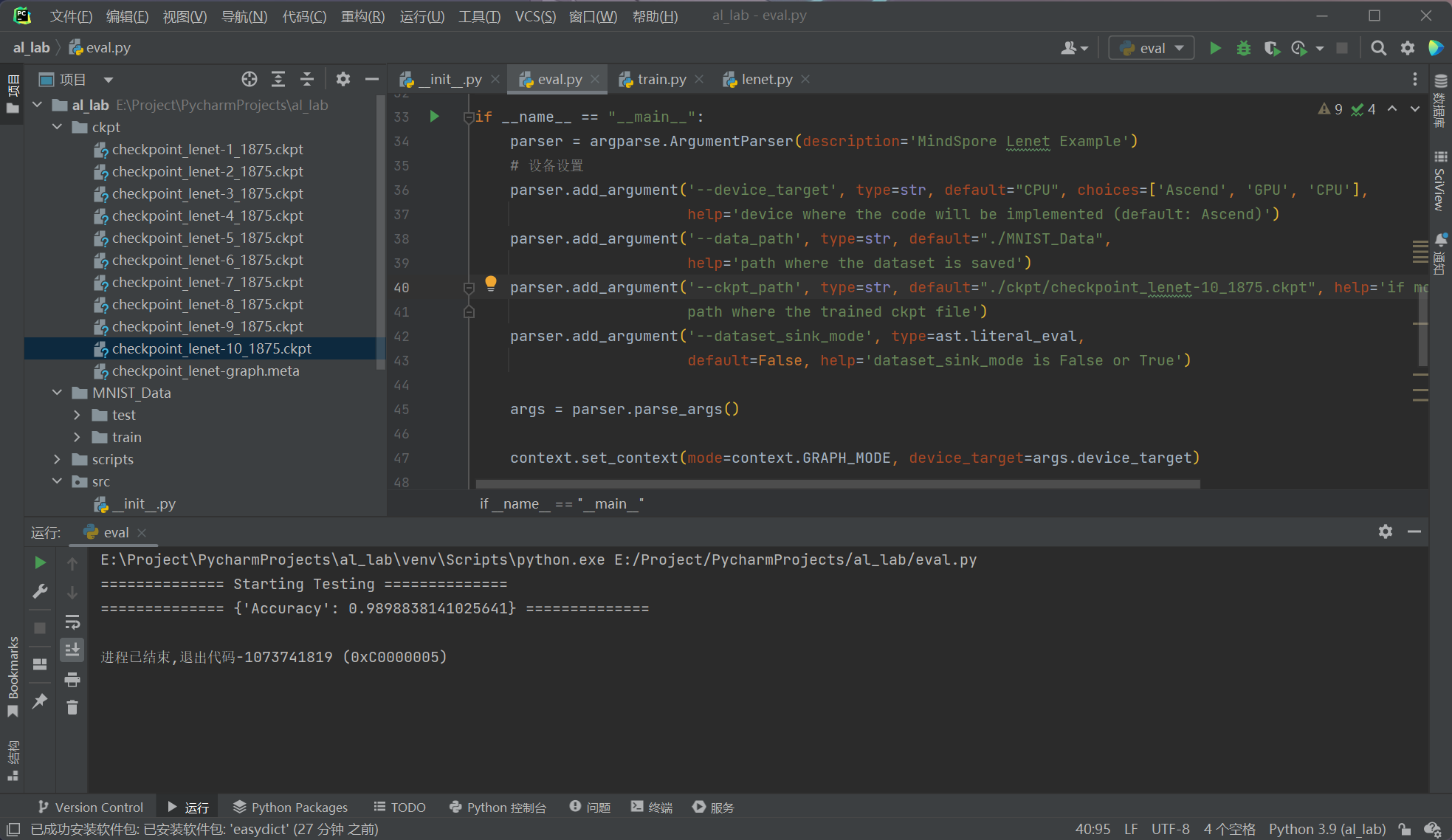
（图1.5-定义损失函数和优化器）

1. 开始训练及验证过程

利用train.py和eval.py完成模型的训练及验证过程。



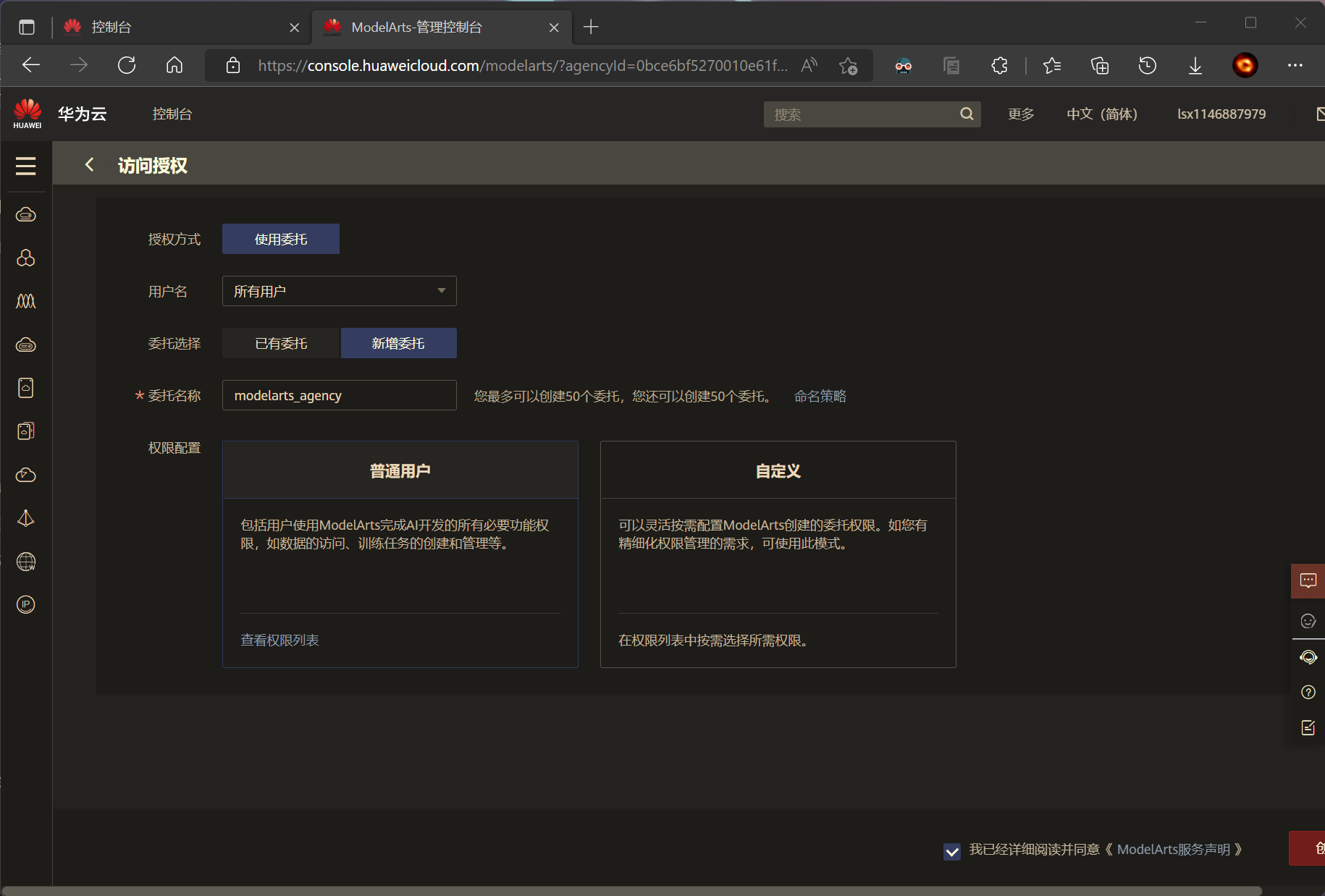
（图1.6-训练过程）



（图1.7-验证过程）

1. **基于Modelarts平台和Tensorflow框架的模型训练及部署**

配置访问权限

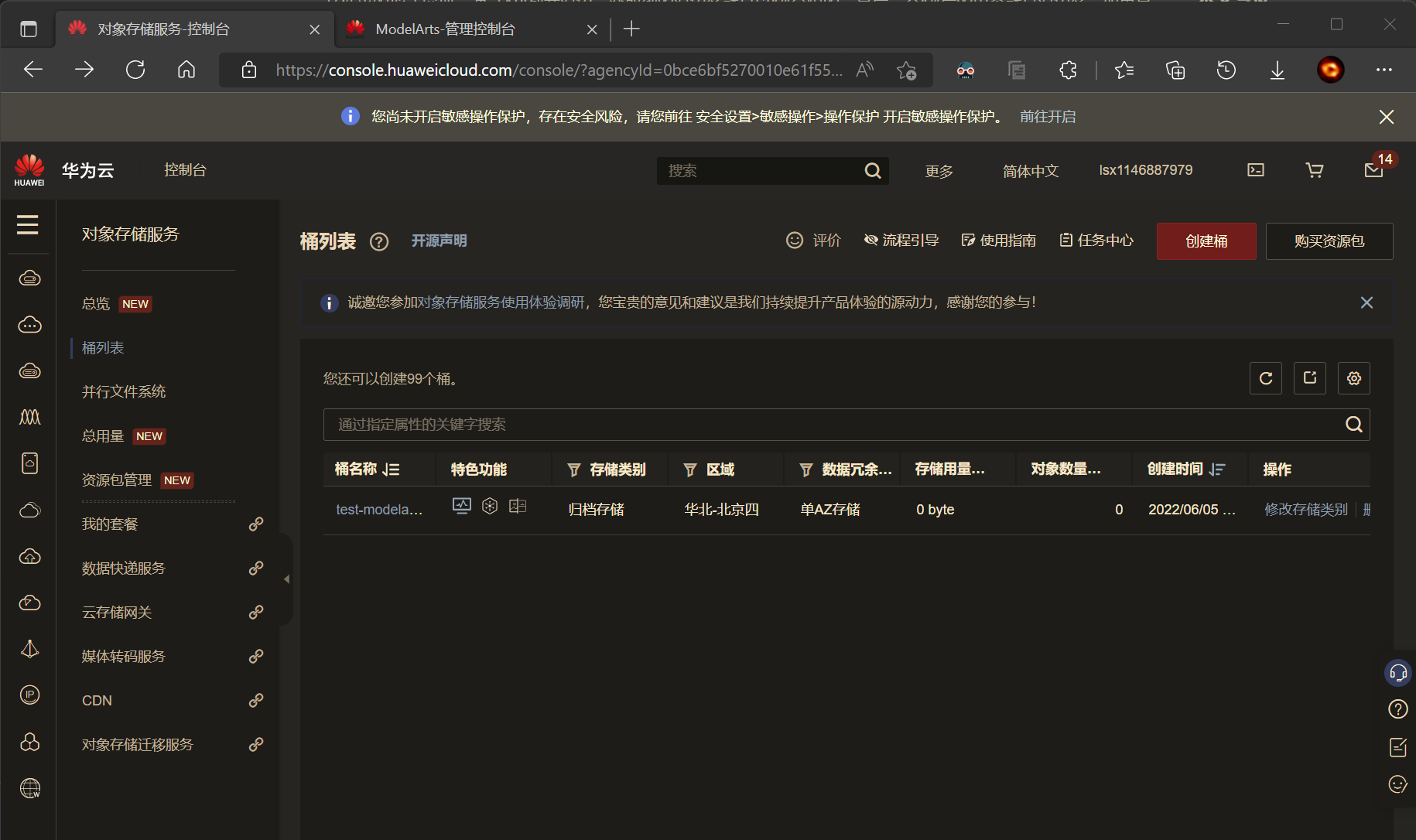


（图2.1-配置访问权限）

创建桶

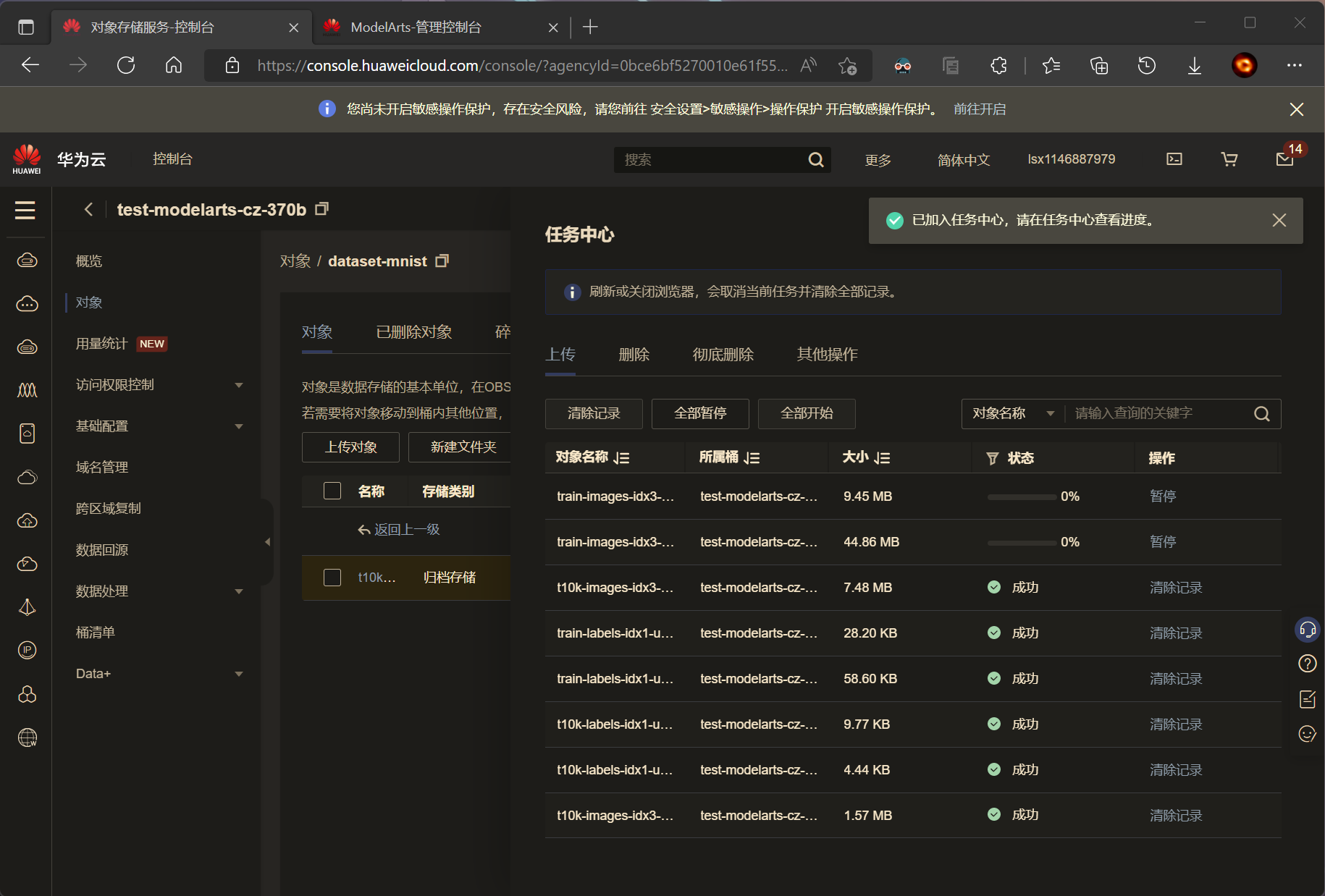


（图2.2-创建桶）



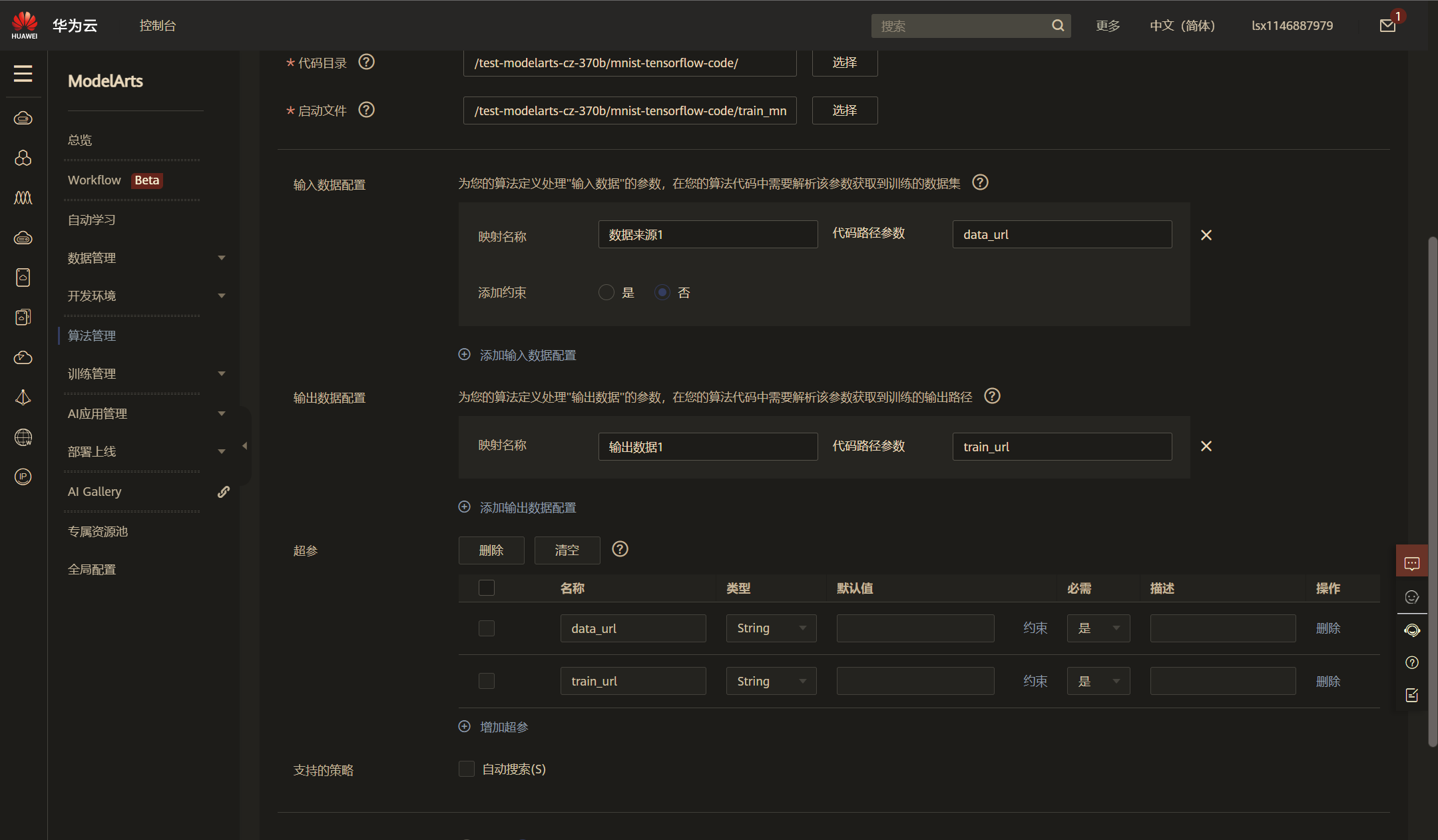
（图2.3-创建好的桶）

上传文件



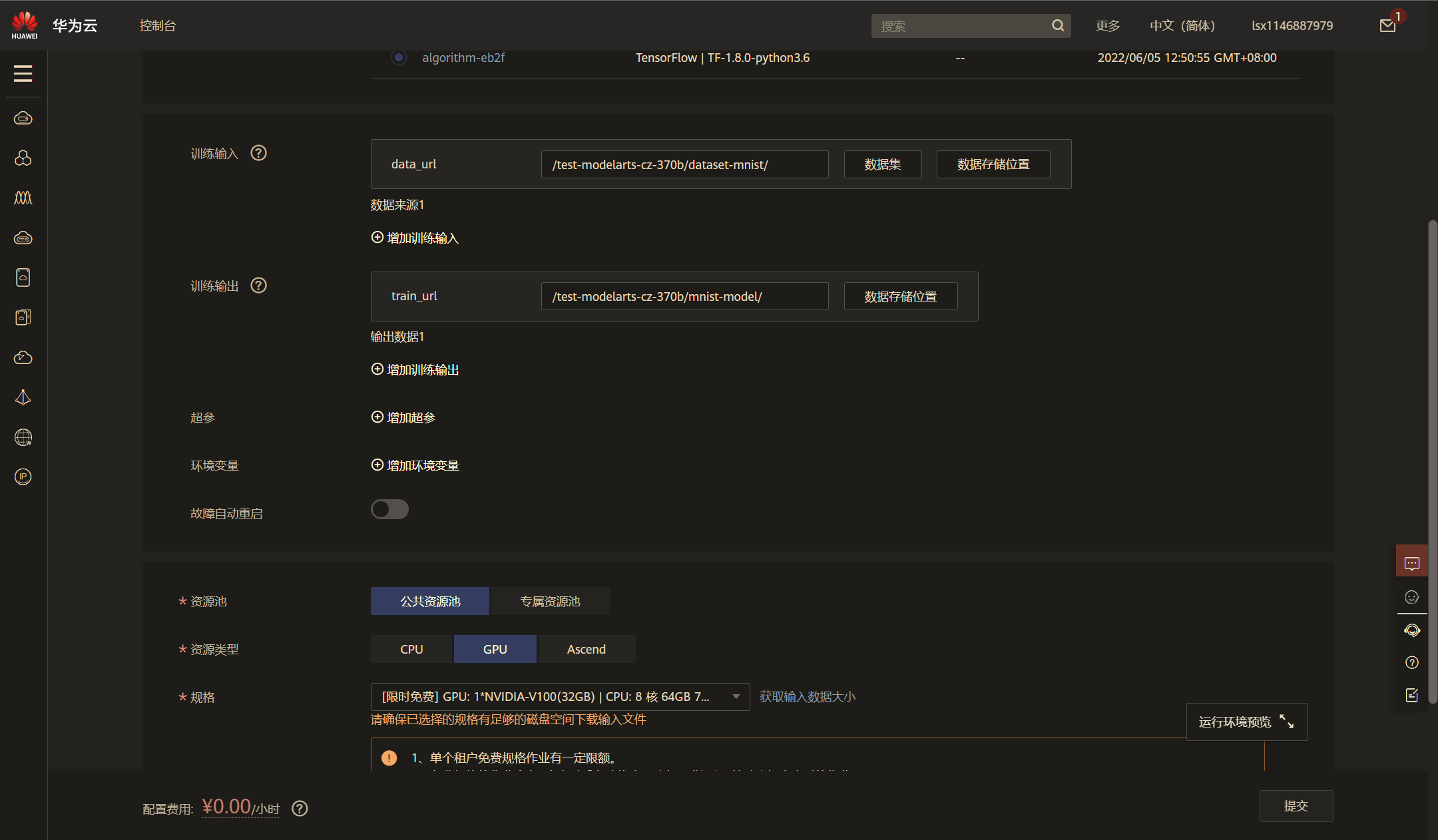
（图2.4-上传需要的文件）

创建算法



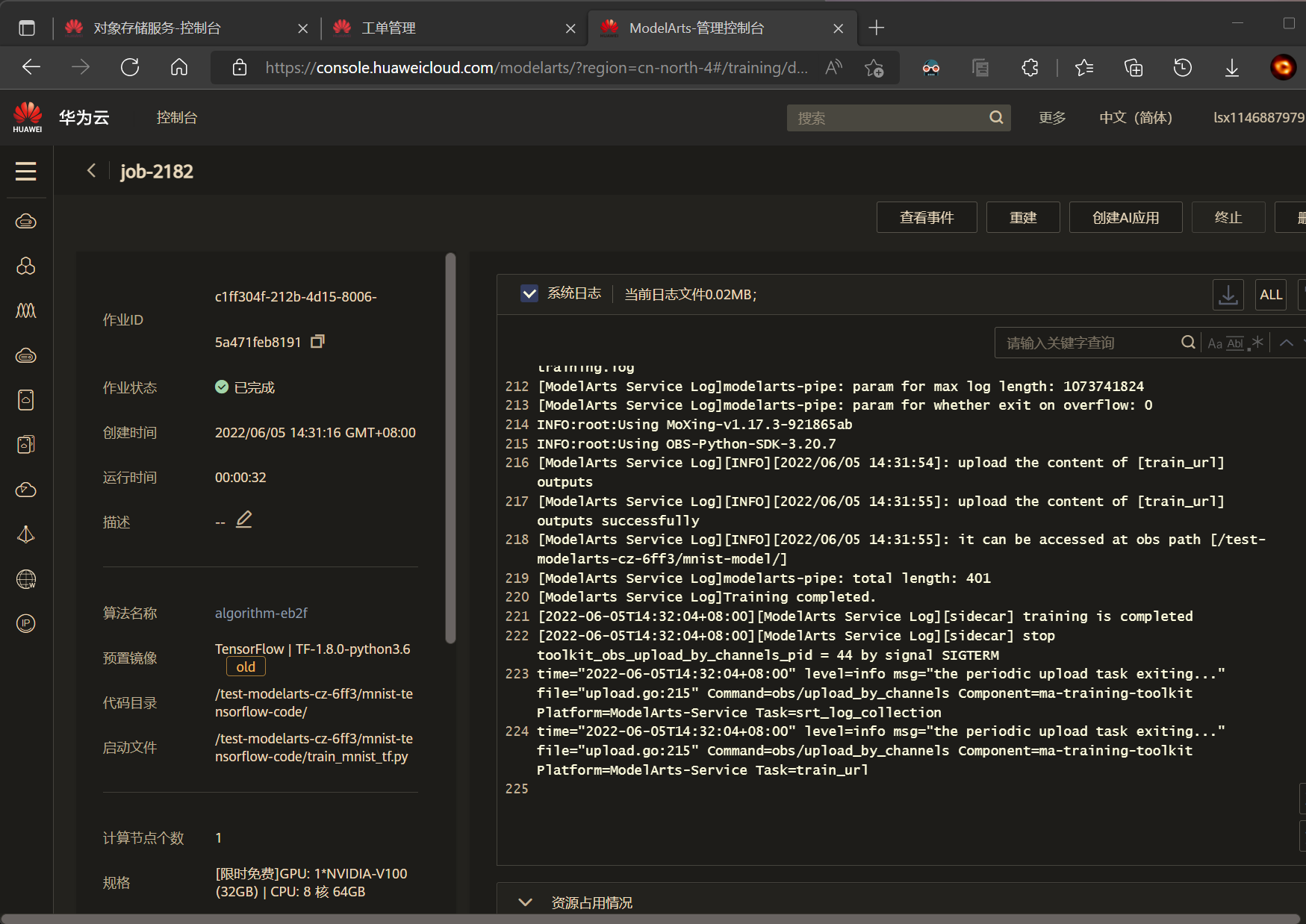
（图2.5-创建算法）

训练模型



（图2.6-训练模型）

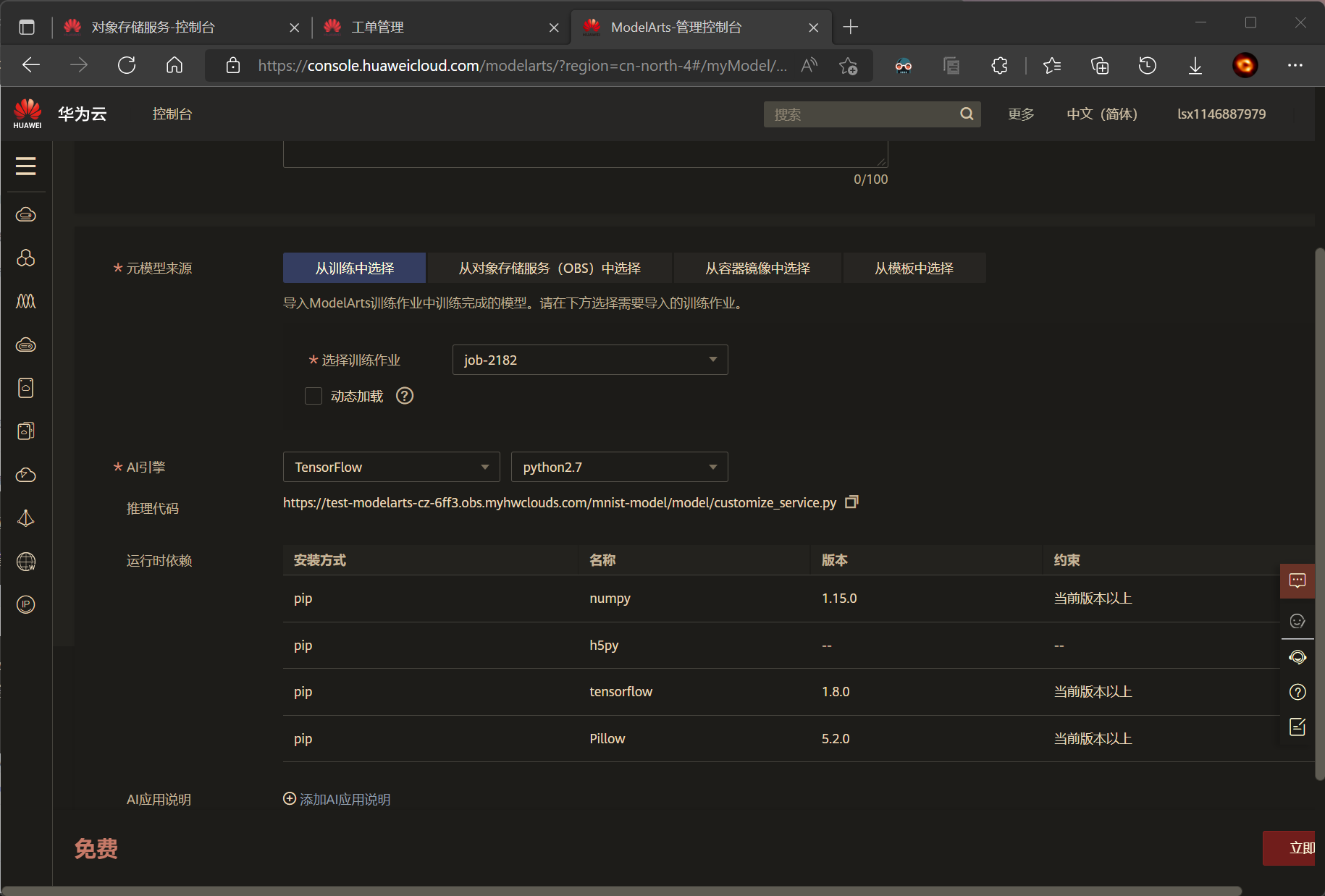
训练完成



（图2.7-训练作业完成日志）

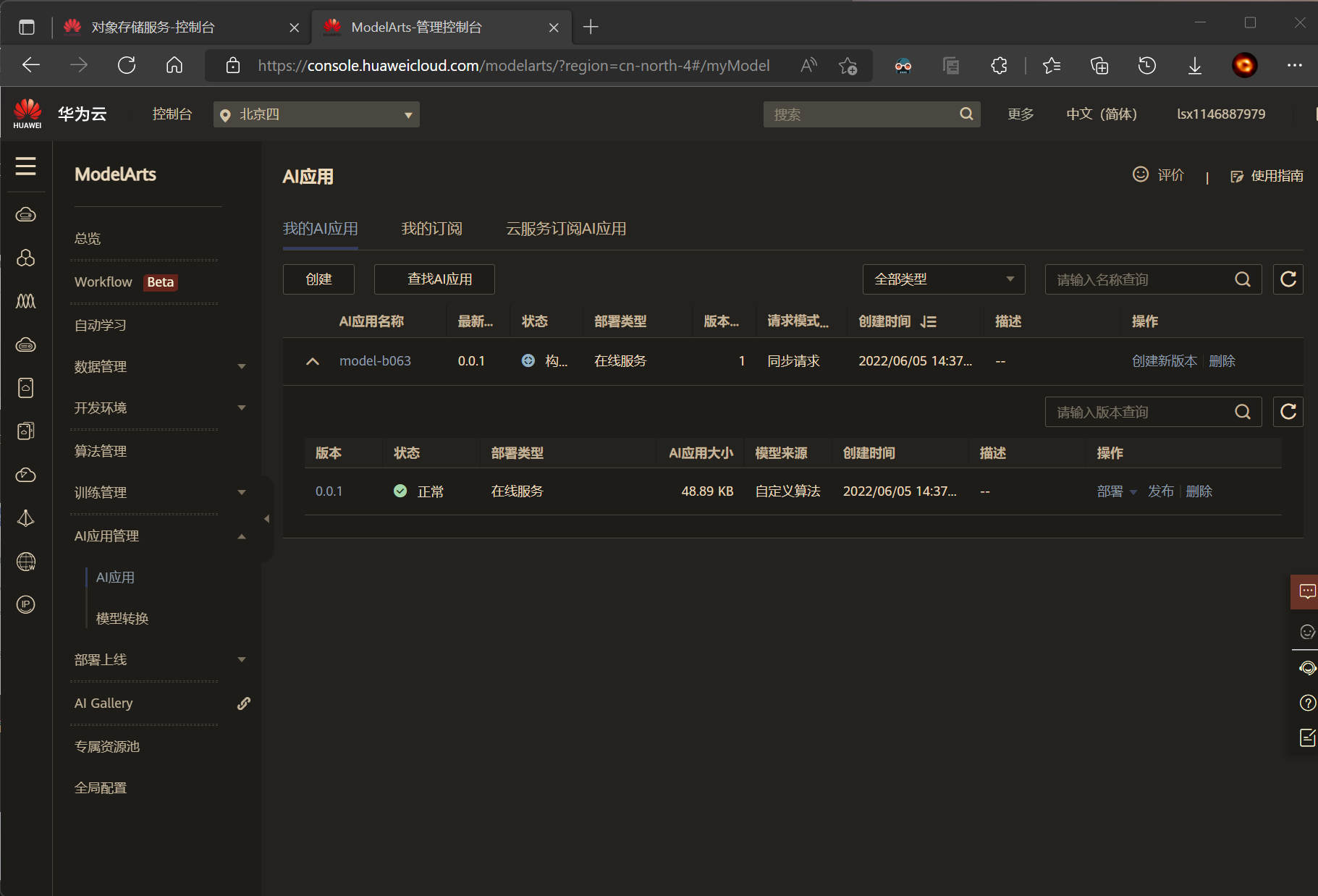
部署AI应用

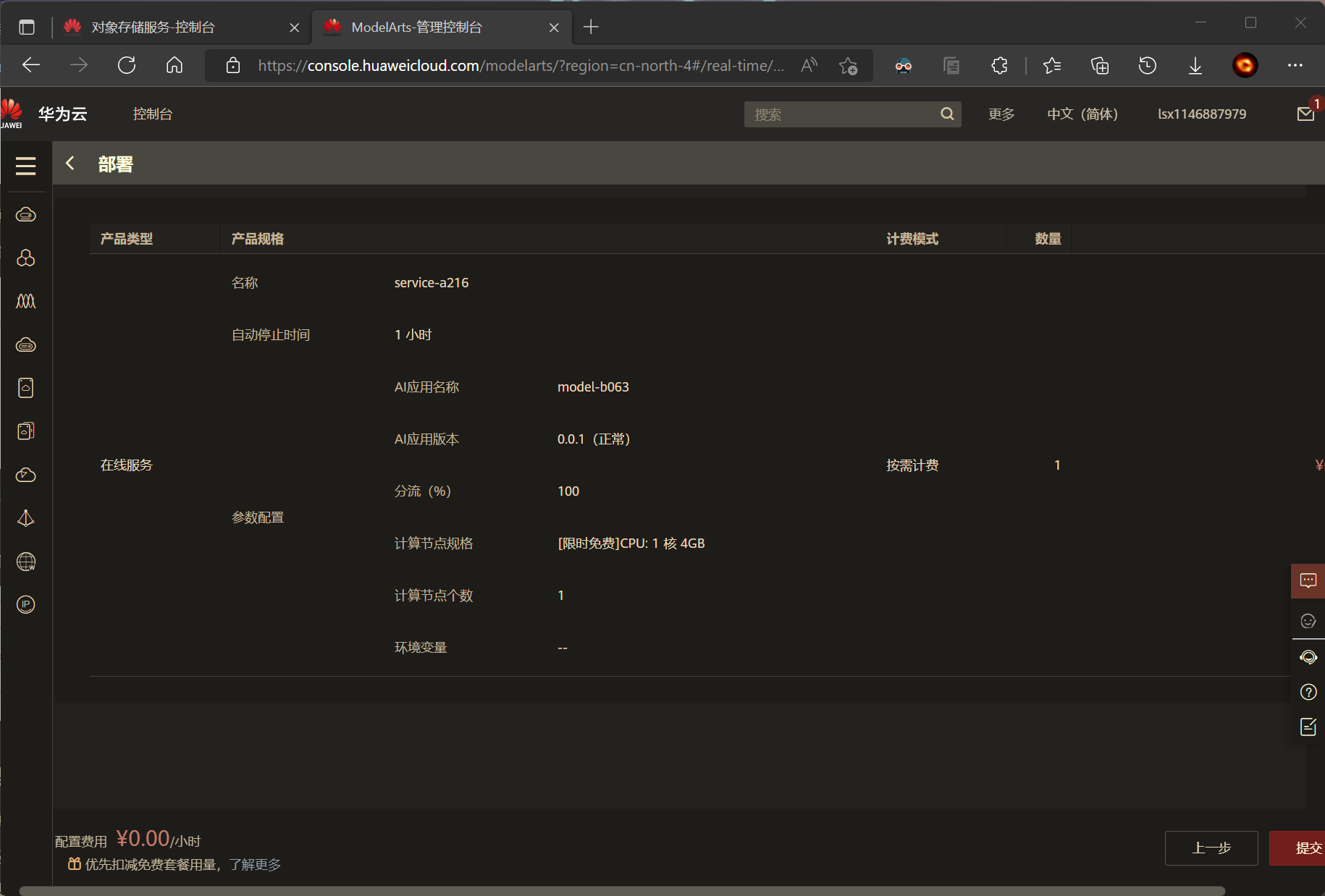
创建AI应用



（图2.8-创建AI应用）

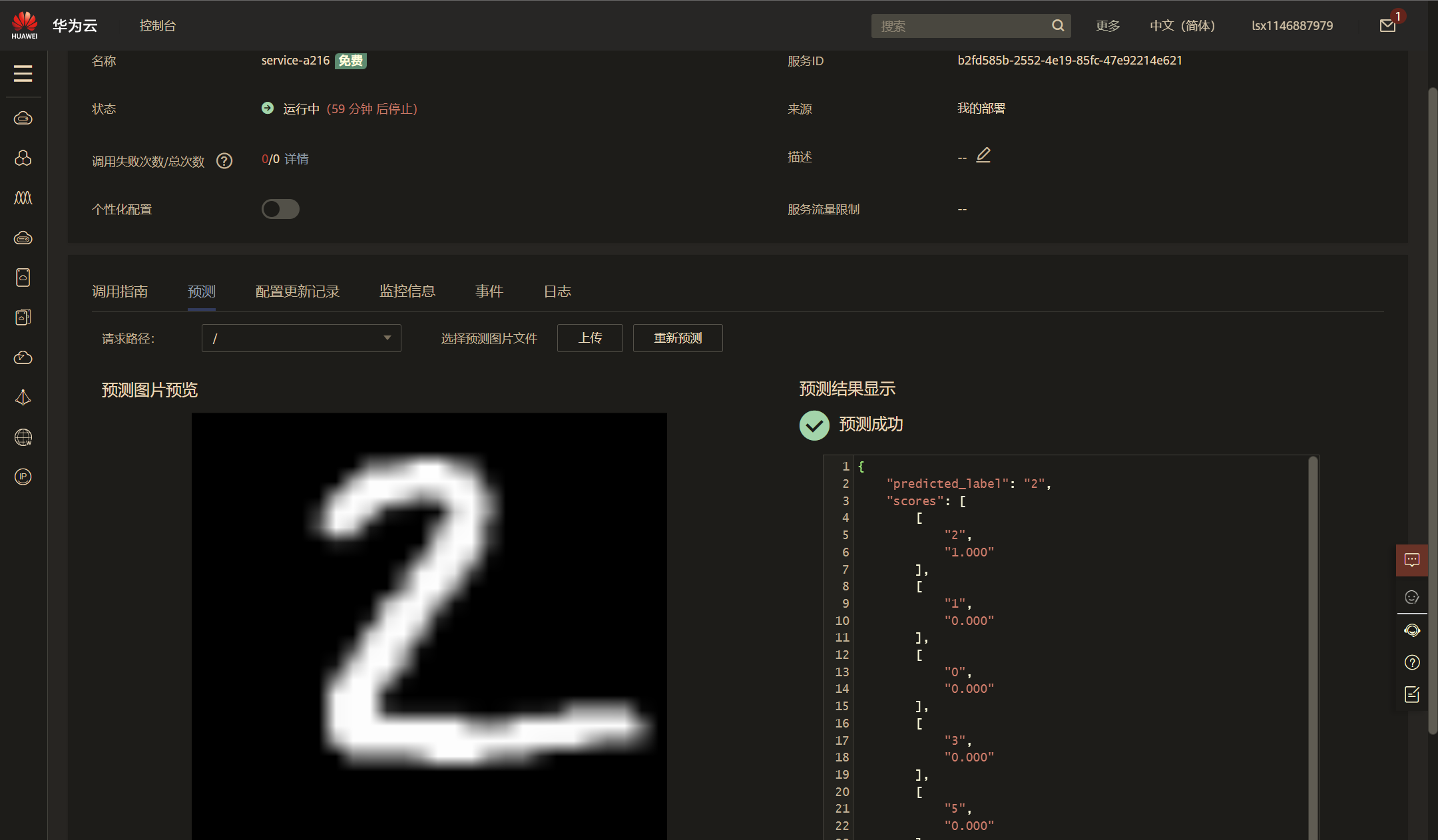
部署在线服务





（图2.9部署在线服务-）

预测结果



（图2.9-预测结果）

1. **基于Modelarts平台和Mindspore框架的模型训练及部署**

数据上传步骤省略

创建算法train，对应本地的train.py执行过程

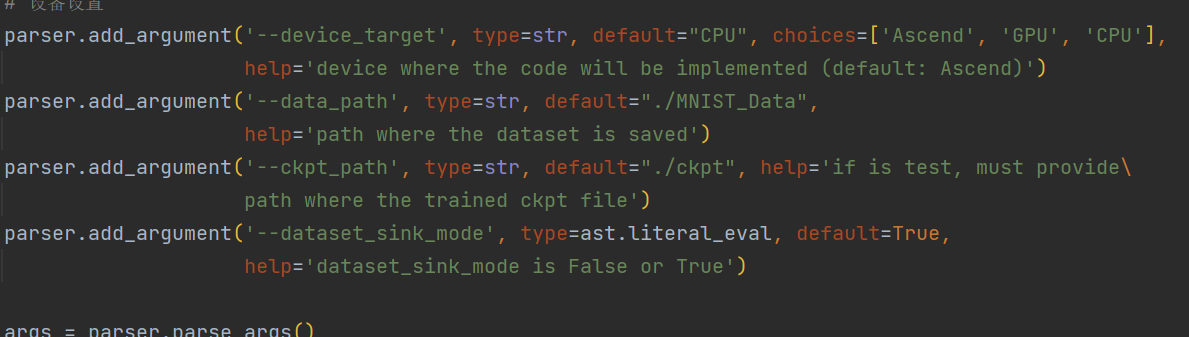
将启动文件设为上传的train.py

增加数据输入配置data\_path,对应于代码中的

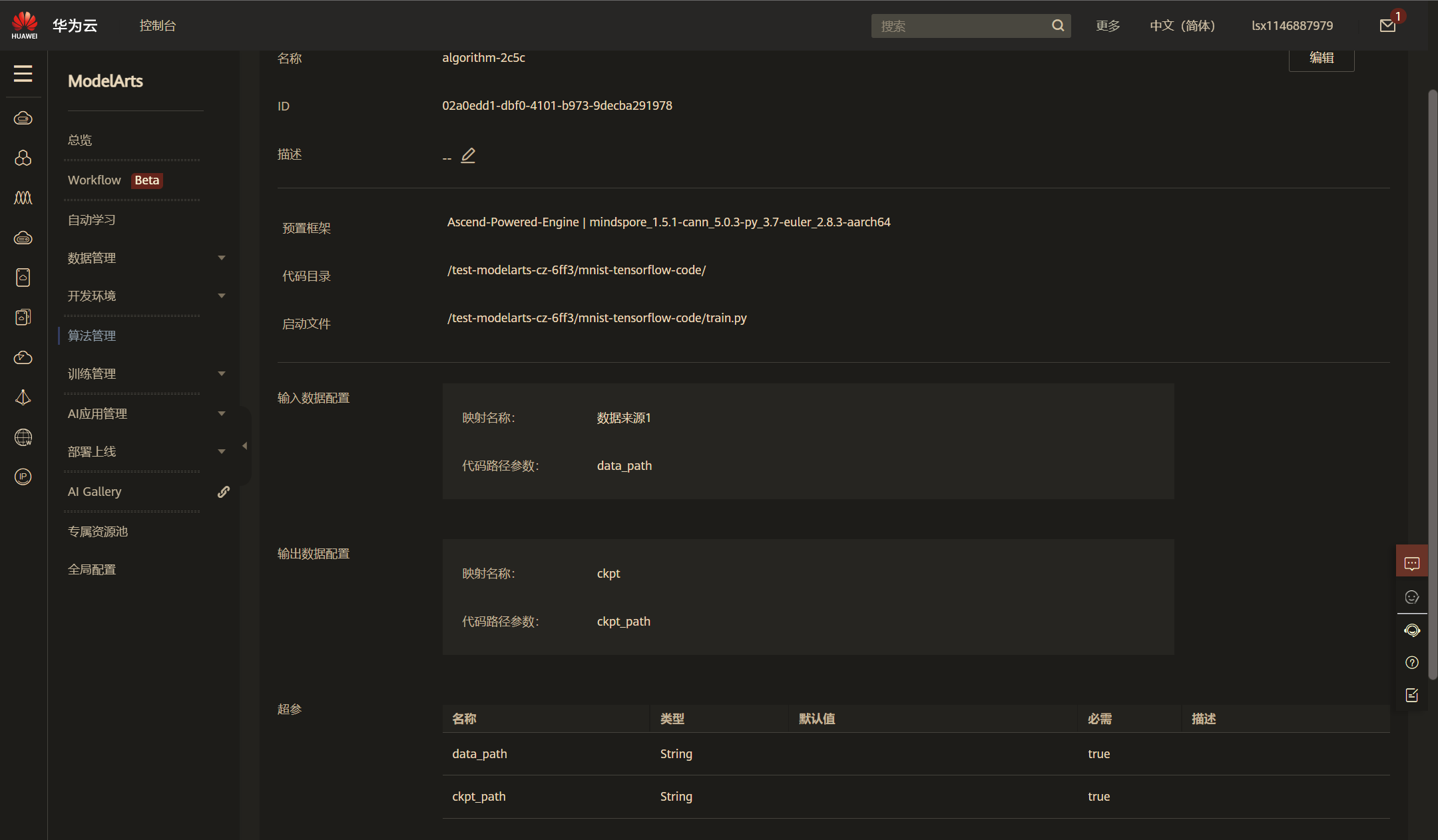
parser.add\_argument('--data\_path', type=str, default="./MNIST\_Data",help='path where the dataset is saved')

增加ckpt\_path对应代码

parser.add\_argument('--ckpt\_path', type=str, default="./ckpt", help='if is test, must provide path where the trained ckpt file')



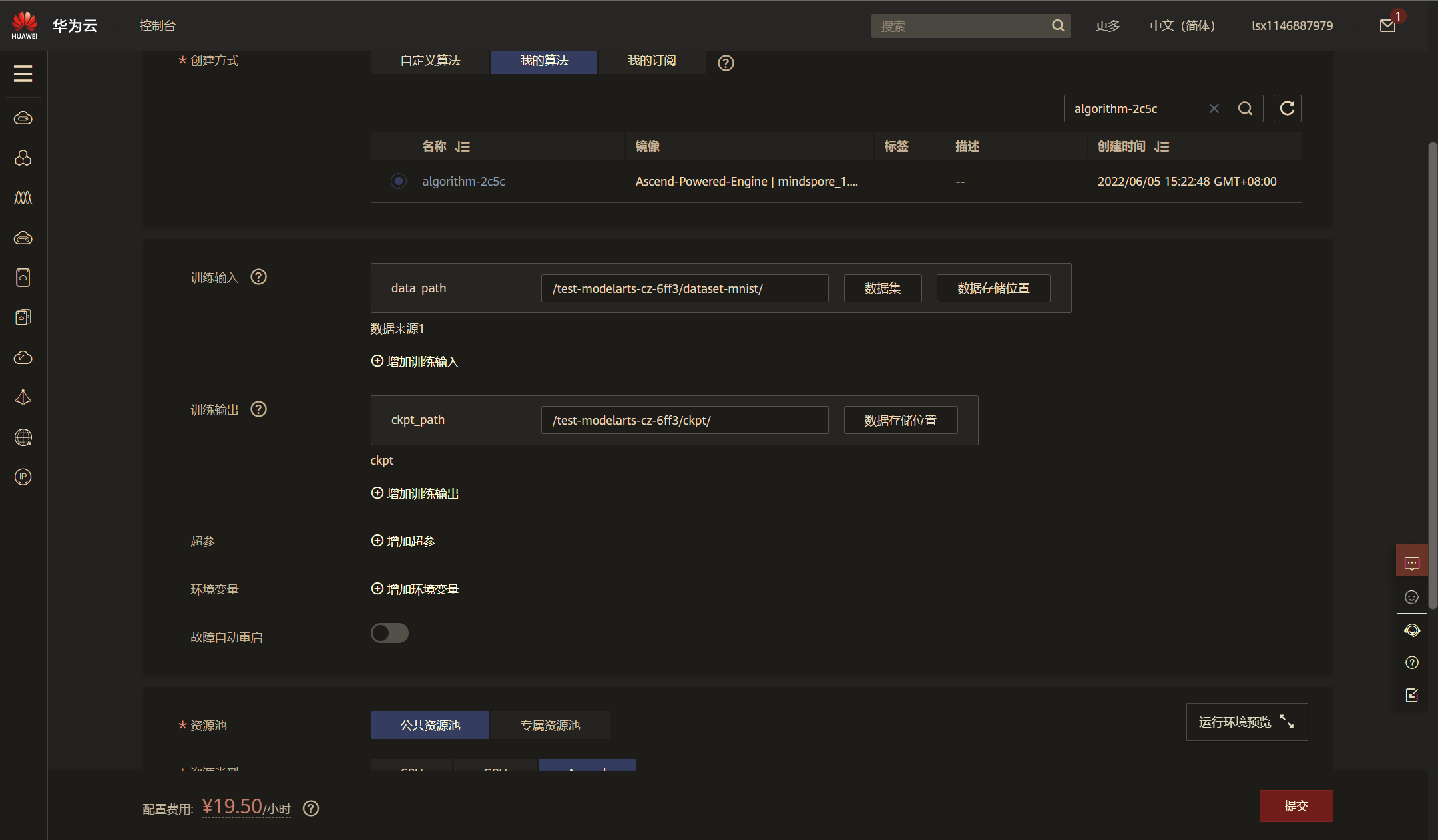
（图3.1-对应代码）



（图3.2-创建算法）

创建训练作业

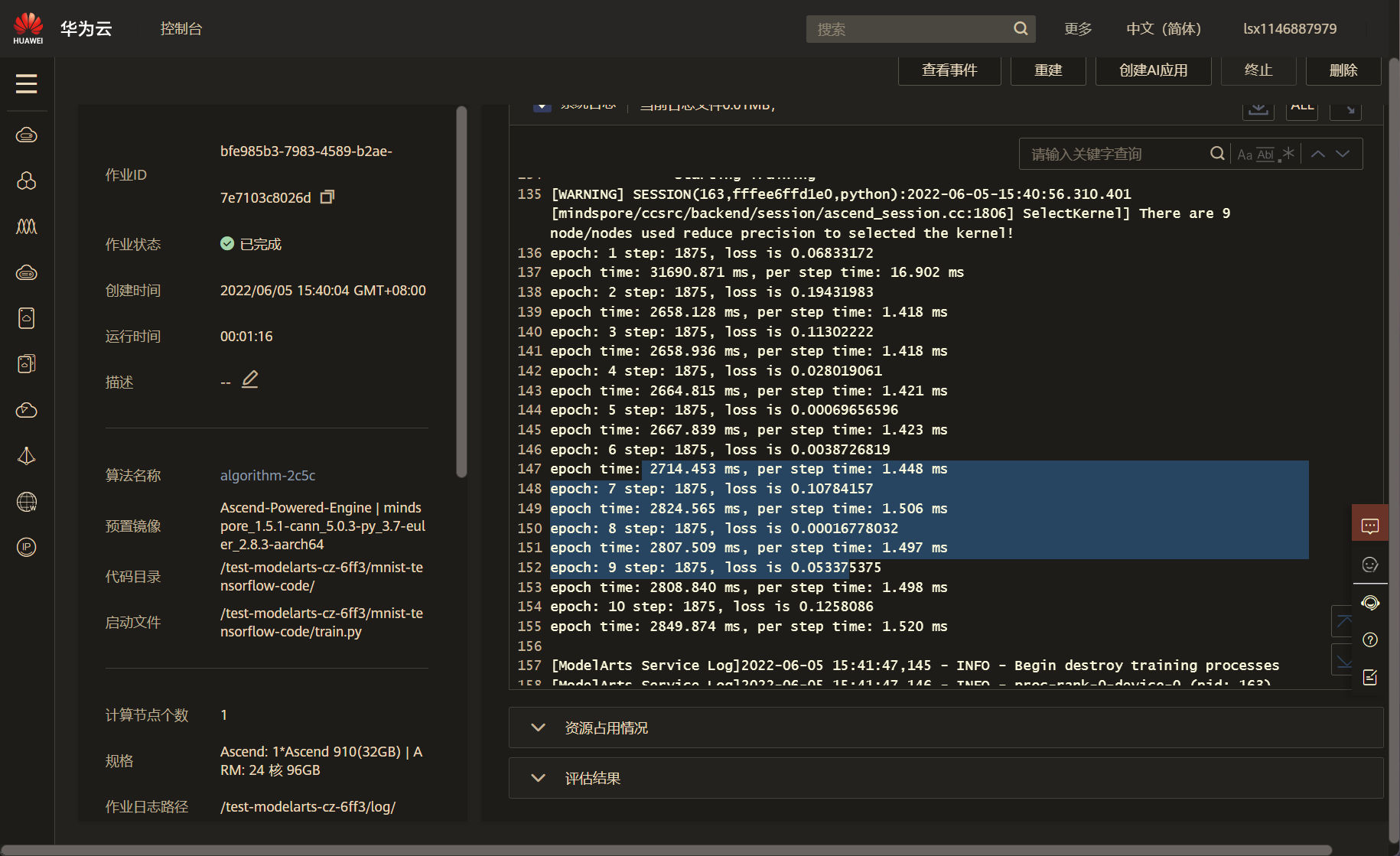
将data\_path和ckpt\_path指向对应的文件位置



（图3.3-创建训练作业）

提交训练作业，等待运行完成，查看结果

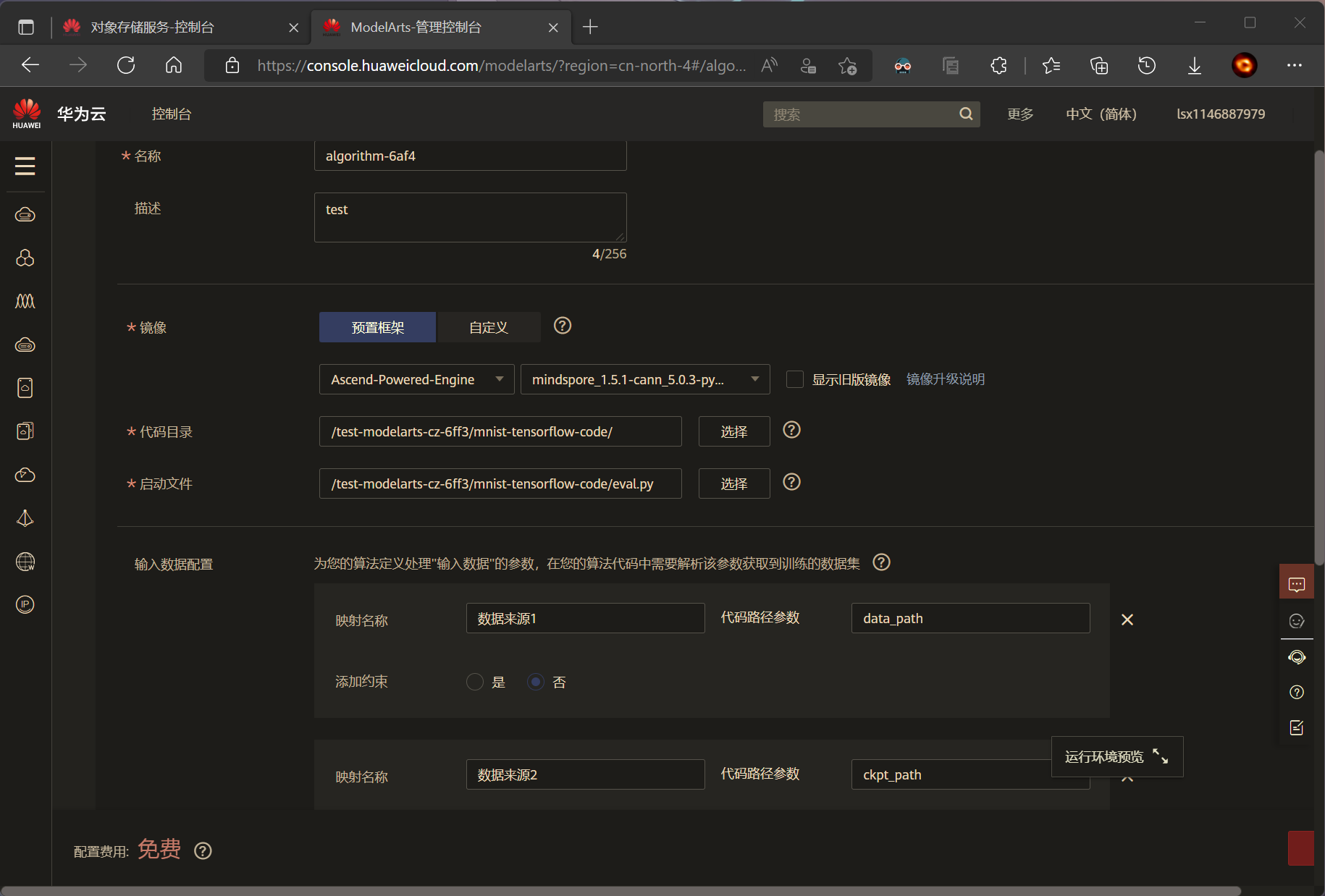
在日志中可以看到执行过程的输出信息



（图3.4-训练作业完成日志）

创建算法eval，对应本地的eval.py执行过程

将启动文件设为上传的eval.py



（图3.5创建算法-）

增加数据输入配置data\_path,对应于代码中的

parser.add\_argument('--data\_path', type=str, default="./MNIST\_Data",help='path where the dataset is saved')

增加ckpt\_path对应代码

parser.add\_argument('--ckpt\_path', type=str, default="./ckpt", help='if is test, must provide path where the trained ckpt file')

创建训练作业

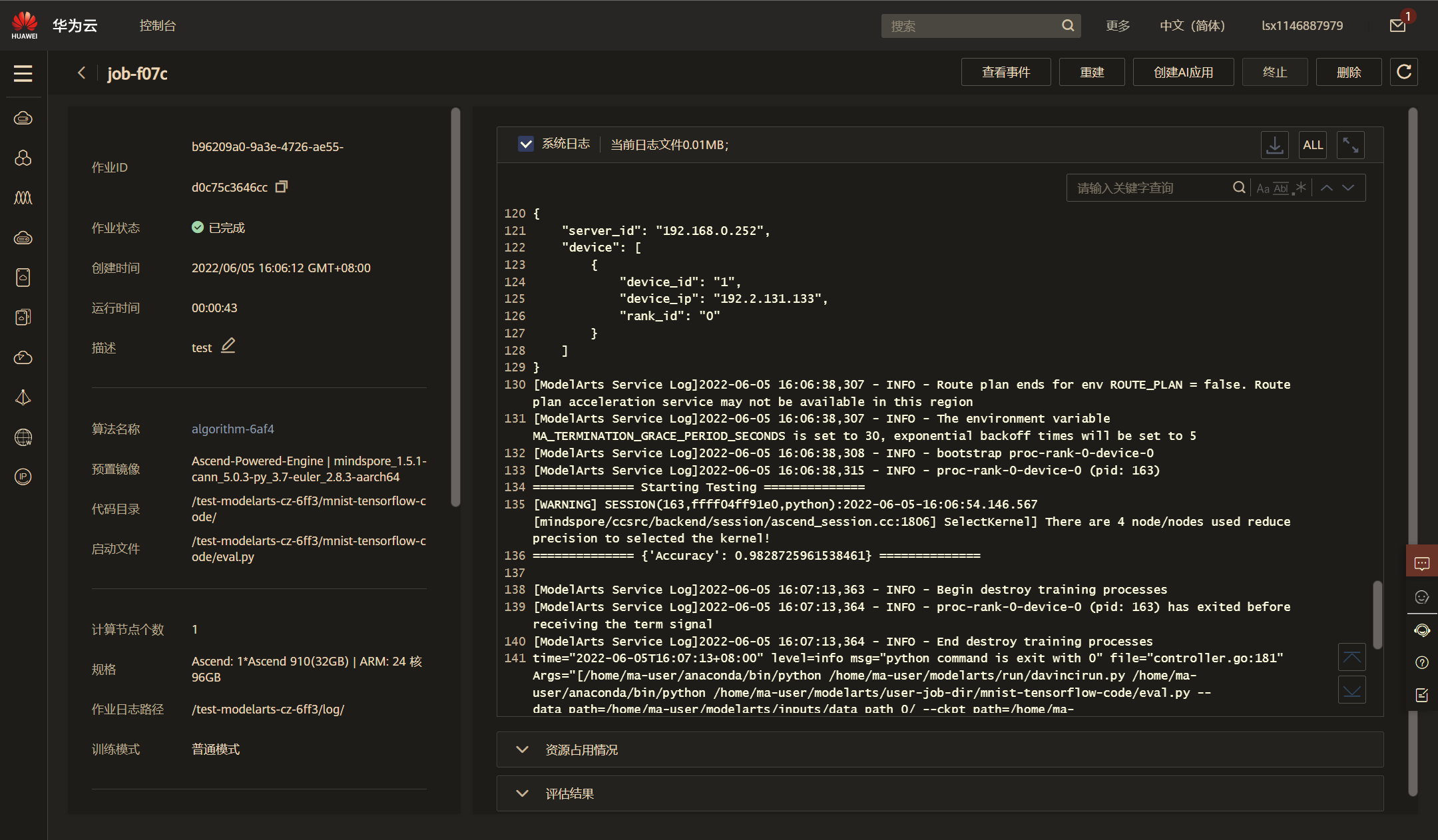
将data\_path和ckpt\_path指向对应的文件位置



（图3.6-创建训练作业）

提交训练作业，等待运行完成，查看结果

在日志中可以看到执行过程的输出信息



（图3.7-训练作业完成日志）

1. **讨论及结论**

本实验是利用Minespore框架在本地及ModelArts 平台实现了一个手写体识别实验。该实验演示了如何利用Minespore来完成手写体识别的过程，并同时完成在本地和华为云ModelArts平台上完成图像识别任务。在实验过程中通过MindSpore来构建图像识别模型，然后将模型部署到ModelArts上提供在线预测服务。

**参考文献**

[1] 华为官方网页实例（<https://support.huaweicloud.com/bestpractice-modelarts/modelarts_10_0080.html>）

[2] 人工智能实验指导