智能数据湖 AI DataLake

快速入门

文档版本 01

发布日期 2026-07-14

华为云计算技术有限公司

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 首次使用 AI DataLake

欢迎使用AI DataLake服务，本文为您介绍使用AI DataLake前需要完成的准备工作，包括从账号准备，到首次体验AI DataLake数据开发的全流程。

第一项：准备账号与权限

在开始使用AI DataLake前，您需完成华为云账号注册、实名认证及相关服务授权，这是保障服务正常使用的基础步骤。

表 1-1 AI DataLake 准备工作规划

阶段	准备工作	是否必选	说明
账号与权限	注册华为云账号	必选	在使用AI DataLake前，需完成华为云账号注册、实名认证及相关权限授权，这是保障服务正常使用的基础步骤。 • **注册华为账号并开通华为云** • **进行实名认证**
	开通AI DataLake服务并授权使用云服务资源	必选	您需要开通AI DataLake服务，才可以在AI DataLake服务中执行创建工作空间，开发作业等操作。 主账号（管理员）首次进入AI DataLake控制台时，会自动弹出“AI DataLake云资源访问授权”页面，单击“立即授权”，授权允许AI DataLake服务代表用户访问其他云服务。 **开通AI DataLake服务并授权使用云服务资源**

阶段	准备工作	是否必选	说明
	创建IAM用户并授权使用AI DataLake	可选	• 如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用AI DataLake服务的其他功能。 • 如果您是企业用户，需要对您所拥有的AI DataLake资源进行精细的权限管理，您可以在IAM控制台创建用户组，添加IAM用户，创建自定义策略并关联至用户组。 **通过IAM授予使用AI DataLake的权限**

第二项：了解 AI DataLake 服务使用流程

AI DataLake包含多个功能模块，覆盖工作空间创建、数据连接、创建资源池、选择引擎、创建端点、开发作业等多个功能模块。了解AI DataLake各个功能模块配合关系与服务使用流程，可以帮助您快速开始上手使用AI DataLake。

只需简单的几步操作即可开始数据开发：创建工作空间、创建计算资源池、选择计算引擎、配置端点、开发作业。

了解更多AI DataLake产品功能请参考**产品功能**。AI DataLake**服务公测**期间开放的功能清单请参考**公测期间开放功能说明**。

图 1-1 AI DataLake 使用流程图

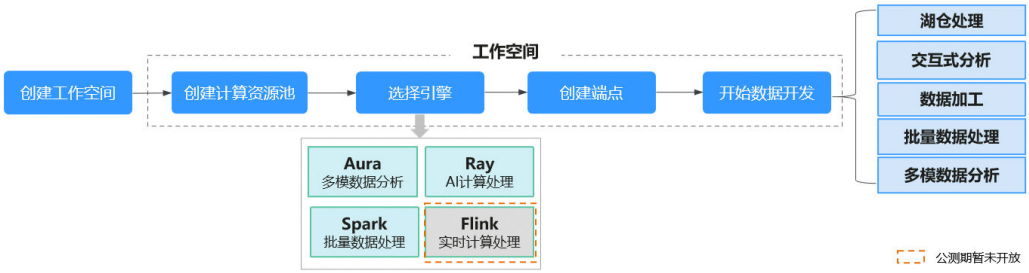


表 1-2 AI DataLake 使用流程

关键步骤	说明	详细操作链接
创建工作空间	首次使用AI DataLake，首先需要创建工作空间。工作空间是逻辑隔离的运行环境，您可以为不同项目或团队创建独立的工作空间，实现资源与权限的隔离。 在创建工作空间时，绑定LakeFormation实例，将业务数据源接入AI DataLake，建立统一的数据访问通道。	**了解工作空间**

关键步骤	说明	详细操作链接
创建计算资源池	作业需要计算资源才能运行，因此需要通过创建计算资源池来为作业分配所需的计算资源。AI DataLake的资源池功能提供了CPU、GPU、NPU计算资源的统一管理与分配能力。	了解计算资源池
选择计算引擎	引擎是计算处理的核心组件，负责执行数据处理与分析任务。不同的业务场景需要选择合适的引擎以获得最佳性能与成本效益。 AI DataLake提供**多模数据引擎Aura**、**AI计算引擎Ray**、**批处理引擎Spark**和**流处理引擎Flink**四大核心计算引擎，聚焦多模数据处理、异构算力混合调度，开放湖仓处理，构建新一代多模湖仓架构，促进Data+AI协同创新。	了解AI DataLake计算引擎
创建并配置端点	端点提供了访问AI DataLake服务的入口，通过端点可以连接计算引擎与计算资源，进行数据开发与查询。 同时配置端点使用资源的最小保障配额（确保业务连续性）和最大配额（防止资源耗尽），有效控制端点资源弹性范围。	• 创建Aura引擎端点 • 创建Ray引擎端点 • 创建Spark引擎端点
作业开发	作业是数据处理与分析任务的执行单元，通过编写代码或配置逻辑，对数据进行转换、分析或机器学习训练。	作业开发

第三项：进阶指导，从体验到实践

完成首次体验后，可根据自身需求深入学习AI DataLake的核心功能，逐步开展实战项目：

说明

AI DataLake在服务公测期间提供Aura引擎，Ray引擎和Spark引擎，更多引擎和功能持续开放中，敬请期待。

- **零基础进阶：快速上手使用AI DataLake查询数据**

  入门示例中介绍了在AI DataLake创建工作空间、计算资源、配置端点，然后在DataArts Studio作业开发页面提交一个单任务Shell或Pipeline类型的AI DataLake作业至Aura端点运行的操作流程。

- **开发者进阶：**
  - **基于Aura DataFrame的多模数据处理**

    本示例通过Aura DataFrame SDK接入Aura端点，在完成数据处理算子的注册后，调用算子依次完成图片解码、图片转Embedding，并查看数据处理结果。

- **基于RayCluster Data的数据处理**

  本示例以通过创建计算资源池、配置Ray类型的端点、连接数据、使用API提交作业为例，演示通过AI DataLake分析数据的操作指导。

- **基于RayJob Data的数据处理**

  本示例以通过创建计算资源池、创建RayJob类型的端点、连接数据、使用API提交作业为例，演示通过AI DataLake分析数据的操作指导。

- **基于PySpark的数据处理**

  本示例以通过创建Spark引擎端点、连接数据、使用API提交作业为例，演示通过AI DataLake分析数据的操作指导。

2 基于用户自定义镜像的多模数据处理

操作场景

AI DataLake是华为云提供的全托管的湖仓一体大数据分析服务，提供端到端的数据管理与分析能力，支持多种引擎的作业开发，满足多样化的大数据处理需求。

本文以通过创建计算资源池、配置Aura类型的端点、连接数据、在DataArts Studio提交作业为例，演示通过AI DataLake和DataArts Studio分析数据的操作指导。

操作流程

开始使用如下样例前，请务必按**准备工作**指导完成必要操作。

1. **规划并创建OBS并行文件系统**：创建一个用于存储LakeFormation Catalog和数据库的OBS并行文件系统。
2. **规划并创建LakeFormation实例、Catalog、数据库**：创建一个LakeFormation实例，并在该实例中创建Catalog及数据库。
3. **创建工作空间**：创建一个工作空间并绑定LakeFormation实例。
4. **创建计算资源池**：创建运行作业的计算资源。
5. **创建运行作业的Aura端点**：创建运行作业的AuraJob端点，并关联资源空间以获得运行作业的资源。
6. **使用DataArts Studio运行作业**：在DataArts Studio作业开发页面提交一个单任务Shell或Pipeline类型的AI DataLake作业至Aura端点运行，本入门以提交单任务Shell的AI DataLake作业为例进行演示。

准备工作

- 已注册账号并实名认证，且账号不能处于欠费或冻结状态。
- 已开通AI DataLake服务并授权使用云服务资源。
- 已开通LakeFormation、OBS权限并进行了委托确认。
- 已开通DataArts Studio服务，并创建了DataArts Studio实例及工作空间。
- 已构建镜像并上传至SWR中。构建镜像时需同时加入处理数据的Python脚本文件，例如“data_processer.py”，用于读取OBS上的CSV文件，并对每行数字求和，将结果写入新文件并保存到OBS中，内容为：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```



```
import csv
import io
from obs import ObsClient, PutObjectHeader
import os


class ObsCsvProcessor:
    def __init__(self, ak: str, sk: str, endpoint: str, bucket_name: str):
        """
        初始化OBS客户端

        :param ak: Access Key ID
        :param sk: Secret Access Key
        :param endpoint: OBS终端节点
        :param bucket_name: 存储桶名称
        """
        self.obs_client = ObsClient(access_key_id=ak, secret_access_key=sk, server=endpoint)
        self.bucket_name = bucket_name


    def read_csv_from_obs(self, object_key: str) -> list:
        """
        从OBS读取CSV文件

        :param object_key: 对象键，即文件在OBS中的路径
        :return: 二维列表，每行是一个列表
        """
        resp = self.obs_client.getObject(bucketName=self.bucket_name, objectKey=object_key,
loadStreamInMemory=True)
        if resp.status < 300:
            content = resp.body.buffer.decode('utf-8')
            # 使用csv模块解析
            reader = csv.reader(io.StringIO(content))
            data = []
            for row in reader:
                # 将每行转换为int类型
                int_row = [int(x) for x in row]
                data.append(int_row)
            return data
        else:
            raise Exception(f"读取OBS文件失败: {resp.errorCode}, {resp.errorMessage}")


    def write_csv_to_obs(self, object_key: str, data: list):
        """
        将数据写入OBS的CSV文件

        :param object_key: 对象键，即文件在OBS中的路径
        :param data: 二维列表
        """
        # 将数据转换为CSV格式
        output = io.StringIO()
        writer = csv.writer(output)
        for row in data:
            writer.writerow(row)
        content = output.getvalue()

        # 上传到OBS
        # 1. Define the headers and set the content type correctly
        headers = PutObjectHeader()
        headers.contentType = 'text/csv'

        # 2. Call the method with the correct argument names
        resp = self.obs_client.putContent(
            bucketName=self.bucket_name,
            objectKey=object_key,
            content=content,
            headers=headers  # Pass the header object here
```



```
        )
        if resp.status >= 300:
            raise Exception(f"写入OBS文件失败: {resp.errorCode}, {resp.errorMessage}")

    def process_csv(self, input_key: str, output_key: str):
        """
        处理CSV文件：对每行求和

        :param input_key: 输入文件的对象键
        :param output_key: 输出文件的对象键
        """
        # 读取CSV
        data = self.read_csv_from_obs(input_key)
        print(f"读取到 {len(data)} 行数据")

        # 对每行求和
        row_sums = []
        for row in data:
            row_sum = sum(row)
            row_sums.append(row_sum)
            print(f"行 {row} 的和为: {row_sum}")

        # 将结果写入新文件（每行一个求和结果）
        self.write_csv_to_obs(output_key, [[s] for s in row_sums])
        print(f"结果已写入: {output_key}")


if __name__ == "__main__":
    # 配置参数
    AK = os.environ.get("AK")
    SK = os.environ.get("SK")
    ENDPOINT = os.environ.get("OBS_ENDPOINT", "obs.xxx.xxx.xxx.com")  # 根据实际区域选择
    BUCKET_NAME = os.environ.get("BUCKET_NAME")

    INPUT_FILE = os.environ.get("INPUT_FILE")      # 输入文件路径 input/data.csv
    OUTPUT_FILE = os.environ.get("OUTPUT_FILE")   # 输出文件路径 output/sum_result.csv

    # 创建处理器并执行
    processor = ObsCsvProcessor(AK, SK, ENDPOINT, BUCKET_NAME)
    processor.process_csv(INPUT_FILE, OUTPUT_FILE)
```

- 已在本地准备数据文件，例如名称为“data.csv”，内容如下：

```
1,2,3
4,5,6
7,8,9
```

- 已获取提交作业用户的AK、SK信息。
  a. 在管理控制台页面，鼠标移动至右上方的用户名，在下拉列表中选择“我的凭证”。
  b. 单击“访问密钥”页签，单击“新增访问密钥”，输入验证码或密码。单击“确定”，生成并下载访问密钥。在.csv文件中获取用户的AK、SK信息。

步骤一：规划并创建 OBS 并行文件系统

AI DataLake Aura通过OBS服务实现数据存储，需要先在OBS控制台进行桶及文件夹创建，并导入样例数据。

1. 登录**华为云管理控制台**。
2. 在页面左上角单击  ，选择“存储 > 对象存储服务 OBS”，进入对象存储服务页面。
3. 选择“并行文件系统 > 创建并行文件系统”，进入创建页面，配置相关参数后单击“立即创建”。

- 文件系统名称：根据界面要求设置并行文件系统名称，例如“aura-job”。
- 其他参数根据实际情况选择。

4. 在“并行文件系统”页面，单击已创建的文件系统名称，例如“aura-job”。
5. 在左侧导航栏选择“概览”，查看并记录“Endpoint(终端节点)”参数值。
6. 在左侧导航栏选择“文件”，单击“新建文件夹”，填写待创建的文件夹名称，单击“确定”。继续单击该新创建的文件夹名称，单击“新建文件夹”，可以创建其子文件夹。

   依次创建用于存放CSV数据文件和结果文件、及LakeFormation元数据的路径，例如：

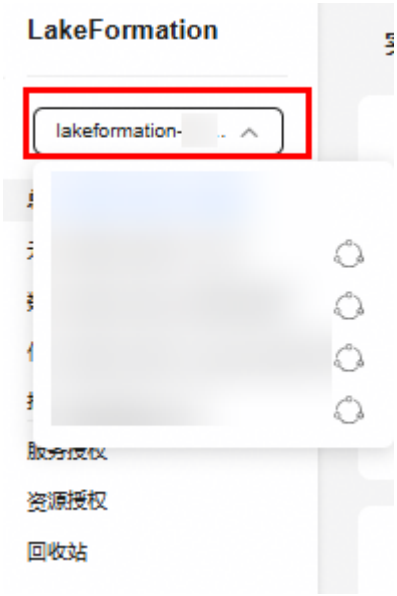
   - 数据文件
     - CSV数据文件存储路径：aura-job/input
     - 数据处理结果文件存储路径：aura-job/output
   - 元数据存储路径
     - Catalog存储路径：aura-job/catalog1
     - 数据库存储路径：aura-job/catalog1/database1
7. 将本地数据文件（例如“data.csv”）上传至“input”路径下。

步骤二：规划并创建 LakeFormation 实例、Catalog、数据库

AI DataLake Aura通过LakeFormation服务管理数据源，需要在LakeFormation购买实例，并配置该实例的Catalog、数据库信息。

1. 登录**华为云管理控制台**。
2. 在页面左上角单击  ，选择“大数据 > 湖仓构建LakeFormation”，进入LakeFormation页面。
3. 在“总览”页面右上角单击“购买实例”，配置相关参数购买LakeFormation实例。
4. 在页面左上角选择切换到该实例。

图 2-1 切换目标实例



5. 创建Catalog。
   a. 在左侧导航栏选择“数据目录”。
   b. 单击“创建”，配置以下参数后，单击“提交”。
      - Catalog名称：输入Catalog名称，例如“catalog1”。
      - 选择位置：单击“ + ”，选择存储位置，例如选择“obs://aura-serverless/catalog1”，单击“确定”。
      - Catalog类型：选择“DEFAULT”。
      - 其他参数保持默认。
   c. 创建完成后，即可在“Catalog”页面查看相关信息。
6. 创建数据库。
   a. 单击待创建数据库所属Catalog名称，在“数据库”页签查看当前Catalog中包含的数据库。
      如果当前已包含名称为“default”的数据库，则跳过数据库的创建操作。
   b. 在“数据库”页签单击“创建”，配置相关参数。
      - 库名称：输入数据库名称，例如“database1”。
      - 所属Catalog：选择“catalog1”。
      - 选择位置：单击“ + ”，选择位置，例如选择“obs://aura-serverless/catalog1/database1”，单击“确定”。
      - 其他参数保持默认。
   c. 单击“提交”，创建完成后，即可在“数据库”页面查看详细信息。

步骤三：创建工作空间

1. 登录**AI DataLake管理控制台**。

2. 在左侧导航栏中，单击工作空间区域。
3. 在下拉列表中选择“创建工作空间”。
4. 配置工作空间的相关参数：

表 2-1 创建工作空间参数说明

类型	参数	示例	说明
基础信息	工作空间名称	workspace_auratest	工作空间的具体名称。 • 名称只能包含字母、中文、数字、中划线、下划线。 • 输入长度为4~32个字符。 工作空间名称不区分大小写，系统会自动转换为小写。
	描述	智能驾驶数据分析	对工作空间的简要描述。
	企业项目	default	如果所建工作空间属于企业项目，可选择对应的企业项目。 企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 关于如何设置企业项目请参考《**企业管理用户指南**》。 **说明** 只有开通了企业管理服务的用户才显示该参数。
多模数据管理	LakeFormation实例	lakeformation_for_auratest	选择当前工作空间关联的LakeFormation实例，即**3**创建的LakeFormation实例。 每个工作空间需关联1个LakeFormation实例，空间创建后已关联的实例不支持修改。

5. 确认所有配置信息无误后，单击“立即创建”，完成工作空间的创建。
   工作空间创建成功后，您可以在工作空间管理的列表中查看新创建的工作空间。

步骤四：创建计算资源池

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤三：创建工作空间**中所创建的空间。
2. 单击左侧导航栏的“资源管理”，选择“计算资源池”，在页面右上角单击“购买”，进入购买计算资源池页面。
3. 在“购买计算资源池”界面，填写具体参数。

表 2-2 购买计算资源池参数说明

参数	示例	说明
计费模式	按需计费	• 包年/包月：预付费模式，按订单的购买周期计费。在购买周期内资源独享，空闲（无作业运行）时不会释放，使用体验更佳，价格比按需计费模式更优惠。 适用于可预估资源使用周期的场景，例如已完成开发进入生产阶段的项目，推荐使用包年包月计费模式。 • 按需计费：即后付费模式，按实际使用量计费，在购买周期内资源独享，空闲时资源不被释放。
资源池名称	resource_pool_for_auratest	计算资源池的具体名称。 • 名称只能包含数字、小写英文字母和中划线，且只能以字母开头、以字母或数字结尾。 • 输入长度不能超过63个字符。
CPU资源	本例配置8个通用计算标准型实例。	CPU是通用型计算资源，适用于各种类型的计算任务。 • CPU资源的特点：通用性强，适合各种类型的计算任务。相比于GPU和NPU的成本更低。擅长处理顺序执行的任务。 • CPU资源的适用场景：ETL 数据抽取、转换、加载；轻量计算场景，例如小规模数据处理、脚本运行，日志分析场景，例如日志采集、解析、统计分析。 • CPU资源的具体规格请参考**产品规格**。
GPU资源	本例不购买GPU实例	GPU是图形处理器适用于图形渲染和大规模并行计算场景，适合深度学习训练和科学计算。GPU拥有成百上千个计算核心，可以同时处理大量简单计算任务。 • GPU资源的特点：支持并行计算，适用于批处理作业场景，数千个核心同时计算，处理效率高。天然适合深度学习、神经网络、AI计算场景。 • GPU资源的适用场景：深度学习，例如神经网络训练、模型调优场景；图形处理场景，例如图像识别、目标检测；视频处理场景，例如视频分析、转码、视频渲染场景，等其他AI科学计算场景。 • GPU资源的具体规格请参考**产品规格**。

参数	示例	说明
NPU资源	本例不购买GPU实例	NPU适用于AI计算场景。NPU资源采用架构优化和指令集，专门加速AI推理任务，具有高能效比的特点。 ● NPU资源的特点：AI计算场景专用，具备高性能、低延迟、推理成本更优的特点。 ● NPU资源的适用场景：AI计算场景、图像识别场景，推荐系统等AI计算设计场景。 ● NPU资源的具体规格请参考**产品规格**。
AI DataLake网络	自定义	选择AI DataLake资源池所属网络，该网络基于虚拟私有云（VPC）进行封装。如果不存在可使用的网络，也可单击“创建网络”进行创建。 **一个工作空间仅支持创建一个网络。**

4. 参数填写完成后，单击“立即购买”，在界面上确认当前配置是否正确。
5. 单击“提交”完成创建，等待资源池状态变成“可用”表示当前资源池创建成功。

步骤五：创建运行作业的 Aura 端点

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤三：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎端点 > 多模数据引擎Aura”进入Aura引擎端点列表页面。
3. 单击页面右上角的“创建端点”，配置以下参数并单击“立即创建”。

表 2-3 创建 Aura 引擎端点

参数	示例	参数说明
端点类型	AuraJob	选择端点类型为“AuraJob”。 AuraJob用于执行定时调度任务，保障大规模数据稳定处理。
端点版本	v1.0	选择端点版本，选择“v1.0”。 ● v1.0：自定义镜像算子，智算、通算混合资源池统一调度，支持Job级弹性。 ● v2.0：细粒度Actor级算子编排调度，数据处理Pipeline执行，性能和效率极大提升。
端点名称	aura-endpoint-test001	输入端点名称，是端点的唯一标识符，不可与已存在的端点重名，且创建后不支持修改。 ● 名称只能包含小写字母、数字、中划线，且只能以字母开头，以字母或数字结尾。 ● 输入长度不能超过63个字符。

参数	示例	参数说明
端点显示名称	智能驾驶团队专用Aura端点-01	输入端点显示名称，创建后可修改，用于用户界面展示，方便用户识别和记忆。 ● 名称只能包含中文、英文、数字、中划线。 ● 输入长度不能超过63个字符。
资源使用模式	混合模式	选择资源使用模式。 ● 预留资源：预留资源通过预留专属计算资源，确保业务所需的计算资源的稳定性，同时获得最优的单价成本。更多信息，请参见**预留模式**。 ● 混合模式：混合模式结合了预留资源和弹性资源，优先消耗预留资源，高峰期预留资源不足时调度弹性资源组合使用。预留资源保证了业务基线的稳定性，弹性资源的自动调度又具备应对突发负载的弹性能力。更多信息，请参见**混合模式**。
选择计算资源池	resource_pool_for_auratest	在下拉框中选择**步骤四：创建计算资源池**已创建的资源池。
规格配置	● 资源规格：aidatalake.cpu.x86.general.1c4g ● 数量(Min-Max)：8-8	选择对应的资源规格并配置资源数量。 ● 资源规格：在下拉框中选择需要配置的资源规格，支持选择CPU、GPU、NPU资源。 ● 数量 (Min-Max)：配置资源池分配给当前端点的最低可用资源和资源上限。 – Min值默认为0，且置灰不可修改。 – Max值需大于0，但不能超过所选计算资源池的资源上限。 可单击“添加规格”添加多个规格配置，也可单击“删除”删除配置，至少需保留一个规格配置。
存储配置	勾选“存储配置”	是否开启存储配置，勾选后还需配置“选择临时存储资源”和“挂载目录”。 开启存储配置后即可使用配置的存储资源存储业务数据，并能实现作业之间的数据共享，否则只能使用CCE容器自带的磁盘存储数据，无法实现数据共享。
选择临时存储资源	test-aura-job	选择用于缓存Aura引擎的运行时程序的存储资源，端点创建后不支持修改。 如果没有可用的存储资源可单击“购买临时存储”进行创建。
挂载目录	/home/service/works	配置Aura Job端点运行作业Pod中挂载存储文件所在的路径。

参数	示例	参数说明
日志对接LTS	勾选“日志对接LTS”	是否对接LTS。对接LTS后日志会投递到云日志服务（Log Tank Service，简称LTS）进行管理。可以使用LTS对云服务日志进行关键词搜索、运营数据统计分析、运行状况监控告警等多种操作。 勾选该参数后，还需配置“日志组”和“日志流”参数。
日志组	container_aurajob_test	在下拉框中选择日志组，如果下拉框中没有可选的日志组，可以单击“创建日志组”进行创建。 日志组（LogGroup）是云日志服务进行日志管理的基本单位，用于对日志流进行分类，一个日志组下面可以创建多个日志流。日志组本身不存储任何日志数据，仅方便管理日志流，每个账号下可以创建100个日志组。
日志流	container_job_log	在下拉框中选择日志流，如果下拉框中没有可选的日志流，可以单击“创建日志流”进行创建。 云日志服务是以日志流（LogStream）作为日志管理维度。日志采集后，以日志流为单位，将不同类型的日志分类存储在不同的日志流上，方便对日志进一步分类管理。

4. 端点创建后，可在列表中查看相关信息，端点状态变为“运行中”后即可提交作业到该端点中运行。

步骤六：使用 DataArts Studio 运行作业

1. 在AI DataLake管理控制台页面，单击导航栏下方“DataArts Studio”进入DataArts Studio作业开发界面。
2. 选择左侧导航栏的“配置管理 > 配置”，单击“敏感参数”，在“敏感参数配置”页面单击“新增敏感参数”，配置以下参数并单击“确定”：
   - 参数名：输入自定义的参数名称，例如“SK”。
   - 参数值：已获取的用户SK信息。
   - 配置类型：选择“加密”。
3. 在左侧导航栏选择“数据开发 > 作业开发”，在作业目录中，右键单击目录名称，选择“新建作业”。
4. 在弹出的“新建作业”对话框中，配置如表2-4所示的参数。

表 2-4 作业参数

参数	示例	说明
作业名称	aidatalake_aura_job_test	自定义作业的名称，只能包含英文字母、数字、中文、中划线、下划线及小数点，且长度为1～128个字符。

参数	示例	说明
作业类型	批处理	选择作业的类型，选择“批处理”。 批处理作业：按调度计划定期处理批量数据，主要用于实时性要求低的场景。批作业是由一个或多个节点组成的流水线，以流水线作为一个整体被调度。被调度触发后，任务执行一段时间必须结束，即任务不能无限时间持续运行。 批处理作业可以配置作业级别的调度任务，即以作业为一整体进行调度，具体请参见**配置作业调度任务（批处理作业）**。
模式	单任务SHELL	选择作业模式。 • Pipeline：即传统的流水线式作业，作业通过画布编辑，可以拖入一个或多个节点组成作业，各节点依次被流水线式地执行。 **说明** 在企业模式下，实时处理作业类型不支持Pipeline模式，仅支持单任务模式。 • 单任务：单任务作业可以认为是有且只有一个节点的批处理作业，整个作业即为一个脚本节点。单任务作业相比于先新建脚本再在作业中以节点引用脚本的开发方式，单任务作业可以直接在SQL编辑器中调测脚本并进行调度配置。 如果模式选择“单任务”，则作业类型仅支持选择“SHELL”。
选择目录	/作业/aura_job_test/	选择作业所属的目录，默认为根目录。
责任人	保持默认	填写该作业的责任人。
作业优先级	高	选择作业的优先级，提供高、中、低三个等级。 **说明** 作业优先级是作业的一个标签属性，不影响作业的实际调度执行的先后顺序。
委托配置	保持默认	配置委托后，作业执行过程中，以委托的身份与其他服务交互。 如果该工作空间已配置过委托，参见**配置公共委托**，则新建的作业默认使用该工作空间级委托。您也可参见**配置作业委托**，修改为作业级委托。 **说明** 作业级委托优先于工作空间级委托。

参数	示例	说明
日志路径	保持默认	选择作业日志的OBS存储路径。日志默认存储在以dlf-log-{Projectid}命名的桶中。 **说明** • 若您想自定义存储路径，请参见**（可选）修改作业日志存储路径**选择您已在OBS服务侧创建的桶。 • 请确保您已具备该参数所指定的OBS路径的读、写权限，否则系统将无法正常写日志或显示日志。
企业项目	default	单击右侧下拉框，选择当前用户已创建的企业项目。
作业描述	-	作业的描述信息。

5. 单击“确定”，创建作业。
6. 进入作业开发页面。
   - 单任务SHELL作业

     在作业目录中，双击新创建的作业名称，进入作业开发页面。
   - Pipeline作业

     拖拽“计算&分析”区域中的“Shell”至右侧画布中，进入作业开发页面。
7. 在SQL编辑器右侧，单击“基本信息”，可以配置作业的相关参数：
   - 单任务SHELL作业：可以配置作业的基本信息、属性和高级信息等。
   - Pipeline作业：可以配置作业的基本信息。注意：作业的属性和高级信息需双击Shell图标进行配置。

   基本信息如**表2-5**所示，属性如**表2-6**所示，高级信息如**表2-7**所示。

表 2-5 作业基本信息

参数	示例	说明
责任人	保持默认	自动匹配创建作业时配置的作业责任人，此处支持修改。
执行用户	-	当“作业调度身份是否可配置”设置为“是”，该参数可见。设置“作业调度身份是否可配置”请参考**配置调度身份**。 执行作业的用户。如果输入了执行用户，则作业以执行用户身份执行；如果没有输入执行用户，则以提交作业启动的用户身份执行。
作业委托	保持默认	当“作业调度身份是否可配置”设置为“是”，该参数可见。设置“作业调度身份是否可配置”请参考**配置调度身份**。 配置委托后，作业执行过程中，以委托的身份与其他服务交互。
作业优先级	高	自动匹配创建作业时配置的作业优先级，此处支持修改。

参数	示例	说明
实例超时时间	0	配置作业实例的超时时间，设置为0或不配置时，该配置项不生效。如果您为作业设置了异常通知，当作业实例执行时间超过超时时间，将触发异常通知，发送消息给用户，作业不会中断，继续运行。
实例超时是否忽略等待时间	不忽略	配置实例超时是否忽略等待时间。 ● 选择“忽略”，表示实例运行时等待时间不会被计入超时时间。 ● 选择“不忽略”，表示实例运行时等待时间会被计入超时时间。
自定义字段	-	配置自定义字段的参数名称和参数值。
作业标签	-	配置作业的标签，用以分类管理作业。 单击“新增”，可给作业重新添加一个标签。
作业描述	-	作业的描述信息。

表 2-6 SQL 作业属性信息

属性	示例	说明
节点名称	-	仅Pipeline作业支持配置，表示SHELL节点名称。节点名称只能包含英文字母、数字、中文字符、中划线、下划线、/、<>和点号，且长度小于等于128个字符。 默认情况下，节点名称会和作业名称保持同步。
Shell或脚本	-	仅Pipeline作业支持配置，选择作业运行的方式，包括： ● Shell语句：需在“Shell语句”框中输入具体的Shell命令。 ● Shell脚本：需在“Shell脚本”脚本中添加已准备好的Shell脚本，也可以单击 新建。
运行环境	AI DataLake	选择“AI DataLake”。
镜像	自定义	选择镜像名称以及版本。 服务公测期间，需要您参考SWR的镜像操作指导自行准备镜像上传SWR。
端点	aura_endpoint_test001	运行作业的Aura端点名称，单击 进入端点列表页面，选择**步骤五：创建运行作业的Aura端点**新创建的端点名称。

属性	示例	说明
资源配置	自定义	运行资源的配置信息。根据实际需求配置资源类型、规格、用量。
容器环境变量	自定义	容器运行时的环境变量配置，容器的环境变量要支持可传入空间变量/常量。 支持单个添加和批量添加。 批量添加时，通过文本模式，可以批量添加多个资源配置，最多添加100条。 运行本入门的示例需新增以下环境变量： ● AK 键为“AK”，值为已获取的用户AK信息。 ● SK 键为“SK”，值为“@priv{SK}”。 “@priv{SK}”中的“SK”即为2新增的敏感参数名称。 ● 并行文件系统名称 键为“BUCKET_NAME”，值为存放数据文件的并行文件系统名称，例如“aura-job”。 ● 数据文件所在的OBS路径 键为“INPUT_FILE”，值为待处理的数据文件所在的OBS路径，例如“input/data.csv”。 ● 存放数据处理结果文件的OBS路径 键为“OUTPUT_FILE”，值为存放数据处理结果文件的OBS路径，例如“output/sum_result.csv”。 ● OBS并行文件系统的终端节点信息 键为“OBS_ENDPOINT”，值为OBS并行文件系统的终端节点信息，即5查看到的Endpoint值。

表 2-7 高级参数

参数	示例	是否必选	说明
节点状态轮询时间（秒）	20	是	设置轮询时间（1~60秒），每隔x秒查询一次节点是否执行完成。 节点运行过程中，根据设置的节点状态轮询时间查询节点是否执行完成。
节点执行的最长时间	6分钟	是	设置节点执行的超时时间，如果节点配置了重试，在超时时间内未执行完成，该节点将会再次重试。

参数	示例	是否必选	说明
失败重试	否	是	节点执行失败后，是否重新执行节点。 ● 是：重新执行节点，请配置以下参数。 – 超时重试 – 最大重试次数 – 重试间隔时间（秒） ● 否：默认值，不重新执行节点。 **说明** ● 如果作业节点配置了重试，并且配置了超时时间，该节点执行超时后，系统支持再重试。 ● 当节点运行超时导致的失败不会重试时，您可前往“默认项设置”修改此策略。 ● 当“失败重试”配置为“是”才显示“超时重试”。
当前节点失败后，后续节点处理策略	终止当前作业执行计划	是	当前节点执行失败后，后续节点的处理策略： ● 终止当前作业执行计划：终止当前作业运行，当前作业实例状态显示为“失败”。如果是周期调度作业，后续周期调度会正常运行。 ● 忽略失败，作业结果设为成功：忽略当前节点失败，当前作业实例状态显示为“忽略是失败”。如果是周期调度作业，后续周期调度会正常运行。

8. 如果运行单任务SHELL作业，需在SQL编辑器中输入SQL语句，支持输入多条SQL语句。例如：

```
python3 /opt/cloud/data_processer.py
```

说明

- SQL语句之间以“;”分隔。如果其它地方使用“;”，请通过“\”进行转义。例如：

```
select 1;
select * from a where b="dsfa\;";  --example 1\;example 2.
```

- SQL脚本内容大小不能超过1MB。
- 使用SQL语句获取的系统日期和通过数据库工具获取的系统日期是不一样，查询结果存到数据库是以YYYY-MM-DD格式，而页面显示查询结果是经过转换后的格式。
- 请勿输入敏感信息，例如ak、sk、password、authorization等信息。

9. 单击画布上方的“提交”提交作业，再单击运行按钮，运行作业。

说明

用户可以查看该作业的运行日志，单击“查看日志”可以进入查看日志界面查看日志的详细信息记录，在日志信息中搜索“jobId”即可获取该作业的ID。

运行完成后，单击画布上方的保存按钮，保存作业的配置信息。

保存后，在右侧的版本里面，会自动生成一个保存版本，支持版本回滚。保存版本时，一分钟内多次保存只记录一次版本。对于中间数据比较重要时，可以通过“新增版本”按钮手动增加保存版本。

10. （可选）如果运行的为调度作业，可返回数据开发首页，在左侧导航栏选择“运维调度 > 作业监控”，在列表中单击提交的作业名称，即可查看作业状态，“运行成功”即表示作业执行成功。

    单击“查看日志”可以进入查看日志界面查看日志的详细信息记录，搜索“jobId”即可获取作业的ID。

11. 返回AI DataLake管理控制台，切换工作空间，并在左侧导航栏选择“引擎端点 > 多模数据引擎Aura”，单击运行作业的Aura端点名称，选择“作业运行历史”，可通过作业ID查看作业运行相关信息。
12. 查看数据结果文件内容。

    a. 在页面左上角单击  图标，选择“存储 > 对象存储服务 OBS”，进入对象存储服务页面。

    b. 选择“并行文件系统”进入并行文件系统列表页面，单击存放结果文件的文件系统名称，例如“aura-job”。

    c. 单击存放结果文件的目录，例如“output”，获取结果文件“sum_result.csv”，并下载至本地查看文件内容，例如内容为：

    ```
    6
    15
    24
    ```

3 基于 Aura DataFrame 的多模数据处理

操作场景

多模数据引擎Aura是专为分析多模态类型数据而设计，支持数据的**边读边算**能力，避免了传统方式的频繁存盘操作，使得视频解码、向量化等预处理环节能够流水线式执行。提供自定义UDF功能，满足多模态场景定制化数据处理的需求。

本章节操作以表格数据处理为示例，介绍在AI DataLake使用多模数据引擎Aura和DataArts Notebook交互式地完成数据的处理与分析的操作。

在作业开发过程中使用到了DataArts Notebook镜像内置的Aura DataFrame SDK，可直接使用魔法命令创建连接。

准备工作

- 已注册账号并实名认证，且账号不能处于欠费或冻结状态。
- 已开通AI DataLake服务并授权使用云服务资源。
- 已开通LakeFormation、OBS权限并进行了委托确认。
- 已开通DataArts Studio服务，并创建了DataArts Studio实例及工作空间。
- 已购买Notebook资源组并创建Notebook实例。

操作流程

本节介绍Aura Job的基本开发流程。技术实现流程：

1. **环境准备**
   a. 在AI DataLake管理控制台创建工作空间，用于提供独立的作业运行环境。
   b. 在AI DataLake管理控制台创建计算资源池，为作业的运行提供计算资源。
   c. 在AI DataLake管理控制台创建端点，配置Aura引擎与计算资源池的关联关系。
2. **绑定计算资源和数据开发**
   a. 在Notebook页面或DataArts Studio管理中心绑定Aura计算资源。
   b. 在Notebook页面为当前Notebook指定已绑定的Aura计算资源。
   c. 在Notebook交互式开发页面使用**%aura_frame**魔法命令建立与多模态数据引擎Aura的连接。

3. **算子注册**

   注册数据处理算子，构建数据处理流水线：

   表格处理算子：对表格数据进行自定义处理。

4. **执行作业后查看结果**

   a. 执行算子操作，处理数据。

   b. 在Notebook交互式开发界面查看执行结果。

步骤一：规划并创建 OBS 并行文件系统

AI DataLake Aura通过OBS服务实现数据存储，需要先在OBS控制台进行桶及文件夹创建，并导入样例数据。

1. 登录**华为云管理控制台**。
2. 在页面左上角单击  ，选择“存储 > 对象存储服务 OBS”，进入对象存储服务页面。
3. 以并行文件系统为例：

   选择“并行文件系统 > 创建并行文件系统”，进入创建页面，配置相关参数后单击“立即创建”。

   - 文件系统名称：根据界面要求设置并行文件系统名称，例如“aura-serverless”。
   - 其他参数根据实际情况选择。

4. 在并行文件系统页面，单击已创建的文件系统名称，例如“aura-serverless”。
5. 在左侧导航栏选择“文件”，单击“新建文件夹”，填写待创建的文件夹名称，单击“确定”。继续单击该新创建的文件夹名称，单击“新建文件夹”，可以创建其子文件夹。

   创建用于存放元数据的路径，例如：

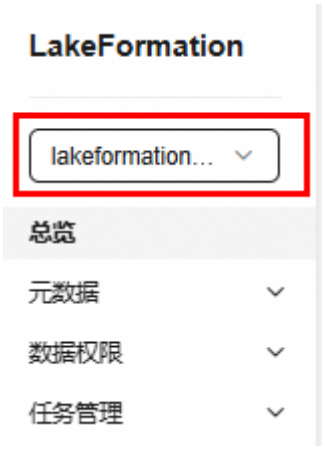
   - Catalog存储路径：aura-serverless/catalog1
   - 数据库存储路径：aura-serverless/catalog1/database1

步骤二：规划并创建 LakeFormation 实例、Catalog、数据库

AI DataLake Aura通过LakeFormation服务管理数据源，需要在LakeFormation购买实例，并配置该实例的Catalog、数据库信息。

1. 登录**华为云管理控制台**。
2. 在页面左上角单击  ，选择“大数据 > 湖仓构建 LakeFormation”，进入LakeFormation页面。
3. 在“总览”页面右上角单击“购买实例”，配置相关参数购买LakeFormation实例。
4. 在页面左上角选择切换到该实例。

图 3-1 切换目标实例



5. 创建Catalog。
   a. 在左侧导航栏选择“数据目录”。
   b. 在Catalog页面单击“创建”，配置以下参数后，单击“提交”。

参数	示例	参数说明
Catalog名称	lakeformation_for_auratest	填写待创建Catalog名称。 只能包含字母、数字和下划线，长度为1~256个字符。
Catalog类型	DEFAULT	选择Catalog类型： • DEFAULT：即默认数据目录，用于管理存储在OBS中的数据资产。 • CLICKHOUSE：CLICKHOUSE Catalog是用于连接外部ClickHouse数据库的外部目录类型。
选择位置	obs://lakeformation-test/catalog1	Catalog数据存储在OBS桶中的位置。可选参数。 根据实际需要选择“并行文件系统”或“对象存储桶”，并选择位置后，单击“确定”。 • 如果配置该参数，则所选位置只能以“obs://”开头，且必须包含一个存储对象，例如选择“obs://lakeformation-test/catalog1”。如果没有合适的OBS桶，可以单击“前往OBS创建”进行创建。 • 该路径不能与其他LakeFormation实例元数据存储路径重复，以免造成数据冲突。 • 建议选择未被其他Catalog选中的文件夹。

参数	示例	参数说明
描述	按需配置	所创建Catalog的描述信息。 长度为0~4000字节，1个中文字符对应3个字节。

c. 创建完成后，即可在“Catalog”页面查看相关信息。

6. 创建数据库。

a. 单击待创建数据库所属Catalog名称，在“数据库”页签查看当前Catalog中包含的数据库。

如果当前已包含名称为“default”的数据库，则跳过数据库的创建操作。

b. 在“数据库”页签单击“创建”，配置相关参数后，单击“提交”。

⚠ 注意

在正式使用新Catalog前，请先手动创建default数据库并指定合法OBS路径，可避免后续系统自动创建带来的隐藏风险。

参数	示例	参数说明
库名称	default	填写待创建数据库名称。 只能包含中文、字母、数字、下划线、中划线，长度为1~128个字符。
所属Catalog	lakeformation_for_auratest	待创建数据库所属Catalog。 本例选择步骤5创建的Catalog。
选择位置	obs://lakeformation-test/catalog1/default	数据库信息存储在OBS桶中的位置。 根据实际需要选择“并行文件系统”或“对象存储桶”，并选择位置后，单击“确定”。 • 所选位置只能以“obs://”开头，且必须包含一个存储对象，例如选择“obs://lakeformation-test/catalog1/default”。如果没有合适的OBS桶，可以单击“前往OBS创建”进行创建。 • 该路径必须与所属的Catalog存储路径（即创建Catalog时配置的“选择位置”参数）不同。 • 该路径不能与其他LakeFormation实例元数据存储路径重复，以免造成数据冲突。 • 如果所属Catalog配置了“数据库存储位置”参数，则此处该参数必须选择为所属Catalog“选择位置”的子路径、或“数据库存储位置”的子路径。

参数	示例	参数说明
描述	按需配置	所创建数据库的描述信息。 长度为0~4000字节，1个中文字符对应3个字节。

c. 创建完成后，即可在“数据库”页面查看详细信息。

步骤三：创建工作空间

1. 登录**AI DataLake管理控制台**。
2. 在左侧导航栏中，单击工作空间区域。
3. 在下拉列表中，选择“创建工作空间” 。
4. 配置工作空间的相关参数：

表 3-1 创建工作空间参数说明

类型	参数	示例	说明
基础信息	工作空间名称	workspace_auratest	工作空间的具体名称。 • 名称只能包含字母、中文、数字、中划线、下划线。 • 输入长度为4~32个字符。 工作空间名称不区分大小写，系统会自动转换为小写。
	描述	智能驾驶数据分析	对工作空间的简要描述。
	企业项目	default	如果所建工作空间属于企业项目，可选择对应的企业项目。企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 关于如何设置企业项目请参考《**企业管理用户指南**》。 **说明** 只有开通了企业管理服务的用户才显示该参数。
多模数据管理	LakeFormation实例	lakeformation_for_auratest	选择当前工作空间关联的LakeFormation实例，即**步骤二：规划并创建LakeFormation实例、Catalog、数据库**创建的LakeFormation实例。 每个工作空间需关联1个LakeFormation实例，空间创建后已关联的实例不支持修改。

5. 确认所有配置信息无误。
   单击“立即创建”，完成工作空间创建。

工作空间创建成功后，您可以在工作空间管理的列表中查看新创建的工作空间。

步骤四：创建计算资源池

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤三：创建工作空间**中所创建的空间。
2. 单击左侧导航栏的“资源管理”，选择“计算资源池”，单击右上方“购买”，进入购买计算资源池页面。
3. 在“购买计算资源池”界面，填写具体参数，参数填写参考**表2-2**。

表 3-2 购买计算资源池参数说明

参数	示例	说明
计费模式	按需计费	• **包年/包月：**预付费模式，按订单的购买周期计费。在购买周期内资源独享，空闲（无作业运行）时不会释放，使用体验更佳，价格比按需计费模式更优惠。 适用于可预估资源使用周期的场景，例如已完成开发进入生产阶段的项目，推荐使用包年包月计费模式。 • **按需计费**：即后付费模式，按实际使用量计费，在购买周期内资源独享，空闲时资源不被释放。
资源池名称	resource_pool_for_auratest	计算资源池的具体名称。 • 名称只能包含数字、小写英文字母和中划线，且只能以字母开头、以字母或数字结尾。 • 输入长度不能超过63个字符。
CPU 资源	本例配置8个通用计算标准型实例。	勾选CPU资源后，选择资源规格并配置购买的实例数量。 CPU是通用型计算资源，适用于各种类型的计算任务。 • CPU资源的特点：通用性强，适合各种类型的计算任务。相比于GPU和NPU的成本更低。擅长处理顺序执行的任务。 • CPU资源的适用场景：ETL数据抽取、转换、加载；轻量计算场景，例如小规模数据处理、脚本运行，日志分析场景，例如日志采集、解析、统计分析。 • CPU资源的具体规格请参考**产品规格**

参数	示例	说明
GPU 资源	本例不购买GPU实例	勾选GPU资源后，选择资源规格并配置购买的实例数量。 GPU是图形处理器适用于图形渲染和大规模并行计算场景，适合深度学习训练和科学计算。GPU拥有成百上千个计算核心，可以同时处理大量简单计算任务。 ● GPU资源的特点：支持并行计算，适用于批处理作业场景，数千个核心同时计算，处理效率高。天然适合深度学习、神经网络、AI计算场景。 ● GPU资源的适用场景：深度学习，例如神经网络训练、模型调优场景；图形处理场景，例如图像识别、目标检测；视频处理场景，例如视频分析、转码、视频渲染场景，等其他AI科学计算场景。 ● GPU资源的具体规格请参考**产品规格**
NPU 资源	本例不购买NPU资源	勾选NPU资源后，选择资源规格并配置购买的实例数量。 NPU适用于AI计算场景。NPU资源采用架构优化和指令集，专门加速AI推理任务，具有高能效比的特点。 ● NPU资源的特点：AI计算场景专用，具备高性能、低延迟、推理成本更优的特点。 ● NPU资源的适用场景：AI计算场景、图像识别场景，推荐系统等AI计算设计场景。 ● NPU资源的具体规格请参考**产品规格**
AI DataLake 网络	自定义	选择AI DataLake资源池所属网络，该网络基于虚拟私有云（VPC）进行封装。如果不存在可使用的网络，也可单击“创建网络”进行创建。 **一个工作空间仅支持创建一个网络。**

4. 参数填写完成后，单击“立即购买”，在界面上确认当前配置是否正确。
5. 单击“提交”完成创建，等待资源池状态变成“可用”表示当前资源池创建成功。

步骤五：创建运行作业的 Aura 端点

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤三：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎端点 > 多模数据引擎Aura”进入Aura引擎端点列表页面。
3. 单击页面右上角的“创建端点”，配置以下参数并单击“立即创建”。

表 3-3 创建 Aura 引擎端点

参数	示例	参数说明
端点类型	AuraJob	选择端点类型。 AuraJob：用于执行定时调度任务，保障大规模数据稳定处理。
端点版本	v2.0	选择端点版本，基于Aura DataFrame的多模数据处理需选择“v2.0”。 ● v1.0：自定义镜像算子，智算、通算混合资源池统一调度，支持Job级弹性。 ● v2.0（推荐）：细粒度Actor级算子编排调度，数据处理Pipeline执行，性能和效率极大提升。
端点名称	aura-endpoint-test001	输入端点名称，是端点的唯一标识符，不可与已存在的端点重名，且创建后不支持修改。 ● 名称只能包含小写字母、数字、中划线，且只能以字母开头，以字母或数字结尾。 ● 输入长度不能超过63个字符。
端点显示名称	智能驾驶团队专用Aura端点-02	输入端点显示名称，创建后可修改，用于用户界面展示，方便用户识别和记忆。 ● 名称只能包含中文、英文、数字、中划线。 ● 输入长度不能超过63个字符。
资源使用模式	混合模式	选择资源使用模式。 ● 预留资源：独享性能，成本最优。预留资源，单价最低，确保业务基线稳定。 适用于负载稳定业务场景。 ● 混合模式：基线保障，自动扩容。优先消耗预留资源，高峰期自动触发弹性补位。 适用于有规律波动的业务
选择计算资源池	resource_pool_for_auratest	在下拉框中选择**步骤四：创建计算资源池**已创建的资源池。

参数	示例	参数说明
工作组配置	-	配置Aura端点的工作组资源。 ● 资源规格：在下拉框中选择需要配置的资源规格，支持选择CPU、GPU、NPU资源。 ● 每Worker卡数：配置Worker卡数。分配了GPU/NPU资源的每个Worker的GPU/NPU卡数。只有GPU和NPU场景可以配置卡数，CPU场景不支持配置。且GPU/NPU场景不支持调整Worker CPU和内存。 ● 每Worker CPU及内存：配置Worker CPU和内存。分配了CPU资源的工作组，不支持自定义GPU/NPU卡数。 ● Worker 数量 (Min-Max)：每个工作组的Worker最大值和最小值。 – 预留资源：Worker最小值需与最大值相同，且需大于0。 – 混合模式：Worker最小值需小于等于最大值，且需大于0。 可单击“添加工作组”添加多个工作组配置，也可单击“删除”删除工作组，至少需保留一个工作组。
选择 OBS 桶	obs-aura	选择用于存储作业运行过程中的结果缓存和日志文件的OBS桶，端点创建后不支持修改。 如果没有可用的OBS桶可单击“新建OBS”进行创建。
日志对接LTS	勾选“日志对接LTS”	是否对接LTS。对接LTS后日志会投递到云日志服务（Log Tank Service，简称LTS）进行管理。可以使用LTS对云服务日志进行关键词搜索、运营数据统计分析、运行状况监控告警等多种操作。 勾选该参数后，还需配置“日志组”和“日志流”参数。
日志组	container_aurajob_test	在下拉框中选择日志组，如果下拉框中没有可选的日志组，可以单击“创建日志组”进行创建。 日志组（LogGroup）是云日志服务进行日志管理的基本单位，用于对日志流进行分类，一个日志组下面可以创建多个日志流。日志组本身不存储任何日志数据，仅方便管理日志流，每个账号下可以创建100个日志组。

参数	示例	参数说明
日志流	container_job_log	在下拉框中选择日志流，如果下拉框中没有可选的日志流，可以单击“创建日志流”进行创建。 云日志服务是以日志流（LogStream）作为日志管理维度。日志采集后，以日志流为单位，将不同类型的日志分类存储在不同的日志流上，方便对日志进一步分类管理。

4. 端点创建后，可在列表中查看相关信息，端点状态变为“运行中”后即可提交作业到该端点中运行。

步骤六：连接多模态数据引擎 Aura 的端点

1. 在AI DataLake管理控制台页面，单击导航栏下方“DataArts Studio”右侧的按钮进入DataArts Studio作业开发界面。
2. 在“数据开发 > 作业开发”页面的右下角，单击“运行环境为空”，选择“新建环境”，配置以下参数并单击“创建”创建Notebook：
   - 运行环境名称：输入自定义名称。
   - 所属资源组：选择已创建的Notebook资源组。
   - 运行资源规格：选择所需的运行资源规格。
   - 镜像选择：选择Notebook镜像。
3. 在“Notebook环境目录”区域右键单击“我的文件”选择“新建 Notebook”，配置Notebook名称并选择存放的目录，单击“确定”。
4. 在Notebook中的单元格执行以下魔法命令连接运行作业的Aura端点：

   ```
   conn = %aura_frame --lf_catalog_name xxx --obs_bucket_name xxx --obs_directory_base xxx/test --lf_instance_id xxx --default_database xxx
   ```

   其中：
   - lf_catalog_name：连接的LakeFormation中的Catalog名称。
   - --obs_bucket_name：用于存储UDF的OBS并行文件系统名称。
   - --obs_directory_base：用于存储UDF的OBS路径。
   - --lf_instance_id：连接的LakeFormation中的实例ID。
   - --default_database：连接的LakeFormation中的默认数据库。

   单击左上角的运行按钮运行命令连接Aura端点，回显以下信息，即表示连接Aura端点成功：

   ```
   [INFO] [Aura Frame] 2026-04-29 09:21:18 Setting guc options: 'SET timezone = UTC; SET obs_result_format = 2;SET current_schema = xxx;'
   [INFO] [Aura Frame] 2026-04-29 09:21:19 Created session 019dd6d3-7fe1-743d-8017-ce7974125079 successfully.
   ```

步骤七：导入 Python 和 Aura DataFrame SDK 依赖包

在Notebook中的新增单元格，执行以下命令导入Python和Aura DataFrame SDK依赖包：

```
import sys
import requests
import pandas as pd
```

```
import aura_frame as aura
from aura_frame.multimodal.types import image, embedding
from aura_frame.ibis.backends.sql.datatype import advance_dtype
```

单击左上角的运行按钮运行命令。

步骤八：创建数据表

1. 在Notebook中的新增单元格，执行以下命令创建数据表：

```
df = pd.DataFrame({
    'a': [1],
    'b': [3],
    'c': [1]
})
ds = conn.from_pandas(df)
ds.show()
```

2. 单击左上角的运行按钮运行命令，命令运行后，结果示例如下则表示创建表成功：

```
generated sql statment: CREATE TEMP TABLE aura_pandas_dataset_vp2r6lk7fna7zaldkcfsk3pajm (
  "a" BIGINT,"b" BIGINT,"c" BIGINT
)
STORE AS PANDAS
[INFO] [Aura Frame] 2026-04-29 09:26:52 Loading dataset
pg_temp.aura_pandas_dataset_vp2r6lk7fna7zaldkcfsk3pajm
[INFO] [Aura Frame] 2026-04-29 09:26:55 read pandas
aura_pandas_dataset_vp2r6lk7fna7zaldkcfsk3pajm
[INFO] [Aura Frame] 2026-04-29 09:26:55 SQL statement for executing:
SELECT *
FROM "pg_temp"."aura_pandas_dataset_vp2r6lk7fna7zaldkcfsk3pajm" AS "t0"
LIMIT 10
[INFO] [Aura Frame] 2026-04-29 09:26:58 Running SQL statement, id is 019dd6d8-
a5a2-747c-901b-1fb08d25f379
   a  b  c
0  1  3  1
```

步骤九：定义和注册 UDF

1. 在Notebook中的新增单元格，执行以下命令定义和注册UDF：

```
target_database = xxx

import ibis.expr.datatypes as dt
import aura_frame as aura

func_signature = aura.Signature(
    parameters=[
        aura.Parameter(name="start", annotation=int),
        aura.Parameter(name="endnum", annotation=int),
        aura.Parameter(name="step", annotation=int)
    ],
    return_annotation=dt.Struct({"col1": int}),
)

@aura.udf.python(database=target_database, signature=func_signature)
def py_generate_series(start, endnum, step):
    current = start
    while current <= endnum:
        yield {'col1': current}
        current += step

 # 如果未注册UDF，则无需执行删除操作，需注释掉delete_function操作
conn.delete_function("py_generate_series", database=target_database)

fn = conn.create_table_function(
    py_generate_series,
    name="py_generate_series",
    database=target_database,
```



```
    signature=func_signature
)
```

“target_database”表示LakeFormation中用于管理注册的UDF的信息的数据库名称。

2. 单击左上角的运行按钮运行命令，命令运行后，结果示例如下则表示定义和注册UDF成功：

```
[INFO] [Aura Frame] 2026-04-29 09:26:58 Start to register function xxx.py_generate_series
[INFO] [Aura Frame] 2026-04-29 09:26:58 Function main entry code has been serialized by
cloudpickle.
[INFO] [Aura Frame] 2026-04-29 09:26:58 The Python package dependencies have been resolved.
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types(4.0 KB) to archive zipfile as aura_frame/
multimodal/types
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types/__init__.py(0 bytes) to archive zipfile as
aura_frame/multimodal/types/__init__.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types/audio.py(8.93 KB) to archive zipfile as
aura_frame/multimodal/types/audio.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types/embedding.py(798 bytes) to archive zipfile
as aura_frame/multimodal/types/embedding.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types/image.py(5.83 KB) to archive zipfile as
aura_frame/multimodal/types/image.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types/vectorized.py(508 bytes) to archive zipfile as
aura_frame/multimodal/types/vectorized.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/types/video.py(10.83 KB) to archive zipfile as
aura_frame/multimodal/types/video.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Appending /opt/cloud/3rdComponent/python3.11/lib/
python3.11/site-packages/aura_frame/multimodal/utils.py(1.84 KB) to archive zipfile as aura_frame/
multimodal/utils.py
[INFO] [Aura Frame] 2026-04-29 09:26:58 Function py_generate_series's archive zipfile has been
uploaded to OBS.
[INFO] [Aura Frame] 2026-04-29 09:26:58 SQL statement for registering function is
CREATE
 FUNCTION
xxx.py_generate_series(start BIGINT, endnum BIGINT, step BIGINT)
RETURNS SETOF STRUCT<"col1":BIGINT>
LANGUAGE PYTHON
RUNTIME_VERSION = '3.11'

STRICT
VOLATILE
IMPORTS = ('https://xxx.com/xxx/xxx/py_generate_series/py_generate_series_20260429092658.zip')

HANDLER = 'py_generate_series'
AS $$
# Comment for UDF 'py_generate_series'



# === Function Metadata ===
# UDTFMeta(
#     _runtime_version='3.11',
#     _runtime_env=RuntimeEnv({}),
#     _registered=False,
#     _input_type=<InputType.PYTHON: 4>,
#     _replace=False,
#     _temporary=False,
#     _if_not_exists=False,
#     _strict=True,
#     _volatility=<VolatilityType.VOLATILE: 3>,
#     _imports=('/opt/cloud/3rdComponent/python3.11/lib/python3.11/site-packages/aura_frame/
multimodal/types', '/opt/cloud/3rdComponent/python3.11/lib/python3.11/site-packages/aura_frame/
multimodal/utils.py'),
#     _packages=(),
```

```
#     _register_type=None,
#     _comment=None,
#     _work_group_name=None,
#     _database='xxx',
#     _func_name='py_generate_series',
#     _signature=<Signature (start: int, endnum: int, step: int) -> struct<col1: int64>>,
#     _udf=<function py_generate_series at 0x7fc5be4d5c60>,
#     _module='__main__',
# )


# === Function Source Code ===
# @aura.udf.python(database=target_database, signature=func_signature)
# def py_generate_series(start, endnum, step):
#     current = start
#     while current <= endnum:
#         yield {'col1': current}
#         current += step

import cloudpickle
py_generate_series = cloudpickle.loads(bytes.fromhex("xxx"))

$$
COMMENT '{"description": null, "pretty_name": "py_generate_series"}'
[INFO] [Aura Frame] 2026-04-29 09:27:01 User-defined function xxx.py_generate_series registered
successfully.
```

步骤十：使用 UDF 向数据表中写入数据

1. 在Notebook中的新增单元格，执行以下命令使用UDF向数据表中写入数据：

```
fn = conn.create_table_function(
    py_generate_series,
    name="py_generate_series",
    database=target_database,
    signature=func_signature
)
ds = ds.flat_map(
    fn=fn,
    on=[ds.a, ds.b, ds.c],
    as_col='new_col'
)
ds.show()
```

2. 单击左上角的运行按钮运行命令，命令运行后，结果示例如下则表示向数据表中写入数据成功：

```
[INFO] [Aura Frame] 2026-04-29 09:27:01 SQL statement for executing:
SELECT "t2"."a",
       "t2"."b",
       "t2"."c",
       "t2"."new_col"
FROM
  (SELECT "t1"."a",
          "t1"."b",
          "t1"."c",
          "t1"."udtf_tmp_col", ("t1"."udtf_tmp_col")."col1" AS "new_col"
   FROM
     (SELECT "t0"."a",
             "t0"."b",
             "t0"."c",
             xxx.PY_GENERATE_SERIES("t0"."a", "t0"."b", "t0"."c") WITH ARGUMENTS(cpu = 0.5, memory
= 1024, arch = 'x86') AS "udtf_tmp_col"
      FROM "pg_temp"."aura_pandas_dataset_vp2r6lk7fna7zaldkcfsk3pajm" AS "t0") AS "t1") AS "t2"
LIMIT 10
[INFO] [Aura Frame] 2026-04-29 09:27:05 Running SQL statement, id is 019dd6d8-be56-71e9-9196-
d7f7e04a4c8f
   a  b  c  new_col
0  1  3  1        1
1  1  3  1        2
2  1  3  1        3
```


步骤十一：关闭数据库连接

在使用数据库连接完成相应的数据操作后，需要关闭数据库连接。

```
conn.close()
```

4 基于 RayCluster Data 的数据处理

操作场景

AI DataLake是华为云提供的全托管的湖仓一体大数据分析服务，提供端到端的数据管理与分析能力，支持多种引擎的作业开发，满足多样化的大数据处理需求。

本文以通过创建计算资源池、配置RayCluster类型的端点、连接数据、使用API提交作业为例，演示通过AI DataLake分析数据的操作指导。

操作流程

开始使用如下样例前，请务必按**准备工作**指导完成必要操作。

1. **规划并创建OBS桶**：创建一个用于存储作业脚本的OBS桶。
2. **创建工作空间**：创建一个工作空间。
3. **创建计算资源池**：创建运行作业的计算资源。
4. **创建运行作业的Ray端点**：创建运行作业的Ray端点，并关联资源空间以获得运行作业的资源。
5. **使用API提交作业**：使用API封装rest请求提交AI DataLake作业至Ray端点运行，本入门以提交一个简单文本处理的AI DataLake作业为例进行演示。

准备工作

- 已注册账号并实名认证，且账号不能处于欠费或冻结状态。
- 已开通AI DataLake服务并授权使用云服务资源。
- 已开通OBS权限并进行了委托确认。
- 已构建镜像并上传至SWR中，并将镜像注册至AI DataLake。
- 已经获取IAM Token，详细操作请参见**获取IAM用户Token**。

步骤一：规划并创建 OBS 桶

AI DataLake Ray通过OBS服务实现作业提交与数据存储，需要先在OBS控制台进行桶及文件夹创建，并导入样例数据或作业脚本。

1. 登录**华为云管理控制台**。

2. 在页面左上角单击  ，选择“存储 > 对象存储服务 OBS”，进入对象存储服务页面。
3. 选择“桶列表 > 创建桶”，进入创建桶页面，配置相关参数后单击“立即创建”。
   - 桶名称：根据界面要求设置桶名称。
   - 其他参数根据实际情况选择。
4. 在桶列表页面，单击已创建的桶名称。
5. 将作业脚本压缩为.zip文件，然后在“对象”页签单击“上传对象”，上传压缩包至所创建的OBS桶中。

   例如文本处理脚本“ray-job-code.py”，内容示例如下：

```
import ray
import random
import time
import sys

# 连接Ray集群
ray.init()

@ray.remote
def compute_points_in_circle(num_points: int) -> int:
    inside = 0
    for _ in range(num_points):
        x, y = random.random(), random.random()
        if x*x + y*y <= 1:
            inside += 1
    return inside

def estimate_pi(total_points: int, num_workers: int = 4) -> float:
    start = time.time()
    points_per_worker = total_points // num_workers
    # 分布式执行任务
    tasks = [compute_points_in_circle.remote(points_per_worker) for _ in range(num_workers)]
    total_inside = sum(ray.get(tasks))
    pi = 4.0 * total_inside / total_points
    print(f"估算的圆周率: {pi:.6f}, 耗时: {time.time() - start:.2f}秒")
    return pi

if __name__ == "__main__":
    estimate_pi(total_points=int(sys.argv[1]), num_workers=1)
    ray.shutdown()
```

步骤二：创建工作空间

1. 登录**AI DataLake管理控制台**。
2. 在左侧导航栏中，单击工作空间区域。
3. 在下拉列表中选择“创建工作空间”。
4. 配置工作空间的相关参数。

表 4-1 工作空间的基本信息

类型	参数	示例	说明
基础信息	工作空间名称	workspace_raytest	工作空间的具体名称，同一个账号下的工作空间不可重名。 ● 名称只能包含字母、数字、中划线、下划线。 ● 名称字符长度为4~32位。
	描述	智能驾驶数据分析	对工作空间的简要描述，最长不超过255个字符。
	企业项目	default	如果所建工作空间属于企业项目，可选择对应的企业项目。 企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 关于如何设置企业项目请参考《**企业管理用户指南**》。 **说明** 只有开通了企业管理服务的用户才显示该参数。
多模数据管理	LakeFormation实例	lakeformation_for_raytest	选择当前工作空间关联的LakeFormation实例，每个工作空间需关联1个LakeFormation实例，空间创建后已关联的实例不支持修改。 ● 选择已创建的LakeFormation实例，用于统一管理元数据和数据访问控制。 ● 如无可用实例，请先创建LakeFormation实例，具体操作请参考**创建LakeFormation实例**。

5. 确认所有配置信息无误，单击“立即创建”，完成工作空间创建。
   工作空间创建成功后，您可以在工作空间管理的列表中查看新创建的工作空间。

步骤三：创建计算资源池

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤二：创建工作空间**新创建的空间。
2. 单击左侧导航栏的“资源管理”，选择“计算资源池”，单击右上方“购买”，进入购买计算资源池页面。
3. 在“购买计算资源池”界面，填写具体参数。

表 4-2 购买计算资源池参数说明

参数	示例	说明
计费模式	按需计费	• **包年/包月**：预付费模式，按订单的购买周期计费。在购买周期内资源独享，空闲（无作业运行）时不会释放，使用体验更佳，价格比按需计费模式更优惠。 适用于可预估资源使用周期的场景，例如已完成开发进入生产阶段的项目，推荐使用包年包月计费模式。 • **按需计费**：即后付费模式，按实际使用量计费，在购买周期内资源独享，空闲时资源不被释放。
资源池名称	resource-pool-for-raytest	计算资源池的具体名称。 • 名称只能包含数字、小写英文字母和中划线，且只能以字母开头、以字母或数字结尾。 • 输入长度不能超过63个字符。
CPU资源	本例配置8个通用计算标准型实例。	CPU是通用型计算资源，适用于各种类型的计算任务。 • CPU资源的特点：通用性强，适合各种类型的计算任务。相比于GPU和NPU的成本更低。擅长处理顺序执行的任务。 • CPU资源的适用场景：ETL数据抽取、转换、加载；轻量计算场景，例如小规模数据处理、脚本运行，日志分析场景，例如日志采集、解析、统计分析。 • CPU资源的具体规格请参考**产品规格**
GPU资源	本例不购买GPU实例	GPU是图形处理器适用于图形渲染和大规模并行计算场景，适合深度学习训练和科学计算。GPU拥有成百上千个计算核心，可以同时处理大量简单计算任务。 • GPU资源的特点：支持并行计算，适用于批处理作业场景，数千个核心同时计算，处理效率高。天然适合深度学习、神经网络、AI计算场景。 • GPU资源的适用场景：深度学习，例如神经网络训练、模型调优场景；图形处理场景，例如图像识别、目标检测；视频处理场景，例如视频分析、转码、视频渲染场景，等其他AI科学计算场景。 • GPU资源的具体规格请参考**产品规格**

参数	示例	说明
NPU资源	本例不购买NPU实例	NPU适用于AI计算场景。NPU资源采用架构优化和指令集，专门加速AI推理任务，具有高能效比的特点。 • NPU资源的特点：AI计算场景专用，具备高性能、低延迟、推理成本更优的特点。 • NPU资源的适用场景：AI计算场景、图像识别场景，推荐系统等AI计算设计场景。 • NPU资源的具体规格请参考**产品规格**
AI DataLake网络	自定义	选择AI DataLake资源池所属网络，该网络基于虚拟私有云（VPC）进行封装。如果不存在可使用的网络，也可单击“创建网络”进行创建。 **一个工作空间仅支持创建一个网络。**

4. 参数填写完成后，单击“立即购买”，在界面上确认当前配置是否正确。
5. 单击“提交”完成创建。等待资源池状态变成“可用”表示当前资源池创建成功。

步骤四：创建运行作业的 Ray 端点

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤二：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎端点 > AI 计算引擎 Ray”进入Ray端点列表页面。
3. 单击页面右上角的“创建端点”，配置以下参数并单击“立即创建”。

表 4-3 创建 Ray 引擎端点

参数	示例	参数说明
端点类型	RayCluster	选择端点类型。 RayCluster：常驻模式，可多次提交Ray作业。
端点名称	Ray-Test	输入端点名称，是端点的唯一标识符，不可与已存在的端点重名，且创建后不支持修改。 名称只能包含小写字母、数字、中划线，且只能以字母开头，以字母或数字结尾。 输入长度不能超过63个字符。
端点显示名称	开发团队专用Ray端点-01	输入端点显示名称，创建后可修改，用于用户界面展示，方便用户识别和记忆。 • 名称只能包含中文、英文、数字、中划线。 • 输入长度不能超过63个字符。

参数	示例	参数说明
资源使用模式	预留资源	选择资源使用模式。 预留资源：预留模式通过预留专属计算资源，确保业务所需的计算资源的稳定性，同时获得最优的单价成本。
选择计算资源池	resource_pool_for_raytest	在下拉框中选择**步骤三：创建计算资源池**已创建的资源池。
集群配置	• Head:64vCPU,128GiB • Workergroups:64vCPU,128GiB	配置对应Ray集群的Head节点规格、Workergroup规格，可创建多个Workergroup，根据业务需求选择。 规格选择列表中可以看到所有的规格，选择的规格可根据创建的Ray资源向下兼容，例如创建了一个aidatalake.cpu.x86.general.1c2g的资源，那么在选择head规格的时候可以选择aidatalake.cpu.x86.general.1c2g，之后配置vCPU与内存资源。
镜像配置	自定义镜像“Ray”	选择集群镜像版本。 预置镜像：预置镜像为AI DataLake提供的基础镜像版本。 自定义镜像：自定义镜像为用户自行注册的SWR镜像。
存储配置	• 选择临时存储：sfs-turbo-01 • 存储路径：/ • 挂载路径：/home/service/works	基于共享存储（SFS Turbo），通过挂载路径规范容器内访问路径，并配合存储路径为Head与Worker节点划定独立的物理子目录，从而在实现跨节点数据互通的同时，解决多节点数据写入冲突的问题。

4. 端点创建后，可在列表中查看相关信息，端点状态变为“运行中”后即可提交作业到该端点中运行。

步骤五：使用 API 提交作业

本示例通过requests库展示一个简单示例，例如“request.py”，模拟通过OpenAPI从创建Ray Job到提交Ray Job的完整流程，执行脚本“request.py”即可提交一个文本处理AI DataLake Job至指定Ray集群。

请求脚本示例如下：

```
import requests
import json
import urllib3
import time
```

```
# 禁用 SSL 警告
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

# 公共配置
TOKEN = "XXX"  # IAM的token
WORKSPACE_ID = "3fa4b5a6-9208-4254-b823-119aa29c4016"  # 工作空间的ID
HEADERS = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}
ENDPOINT = "https://fabric.cn-southwest-2.myhuaweicloud.com" # 实际使用时这个ENDPOINT为对应region endpoint的值，此处为贵阳一地址
# 提交运行 URL
URL_SUBMIT_RUN = f"https://{ENDPOINT}/v2/workspaces/{WORKSPACE_ID}/ray-jobs"

def submit_run():
    payload = {
        "name": "Ray-Test",
        "description": "description",
        "endpoint_name": "Ray-Test",
        "config": {
            "entrypoint": "python3 -m estimate_pi 1000000",
            "runtime_env": {
                "working_dir": "obs://ray-shared-csc/ray-job-code.zip", #需替换为作业脚本存储的obs路径，且必须为.zip文件
                "py_modules": [
                    "string"
                ],
                "pip": [
                    "numpy=1.16.1"  # 可选值，填写值为涉及pip依赖库，若不涉及可不填
                ],
                "env_vars": {
                    "additionalProp1": "value1",  # 可选值，填写值为环境变量，若不涉及可不填
                    "additionalProp2": "value2",
                    "additionalProp3": "value3"
                }
            }
        }
    }

    try:
        response = requests.post(
            URL_SUBMIT_RUN, headers=HEADERS, json=payload, verify=False
        )
        if response.status_code in [200, 201, 202]:
            print(
                f"提交运行成功！响应码：{response.status_code}。响应内容：{response.text}"
            )
        else:
            print(f"提交运行失败 (Code: {response.status_code})")
            print(response.text)
    except Exception as e:
        print(f"请求异常: {e}")

if __name__ == "__main__":
        submit_run()
```

提交作业后，您可以查看作业，操作如下：

1. 在AI DataLake管理控制台页面的左侧导航栏，选择“作业开发 > 作业运行历史”，进入作业运行历史页面。
2. 单击“AI计算引擎Ray”页签，查看作业信息。

5 基于 RayJob Data 的数据处理

操作场景

本文以通过创建计算资源池、创建RayJob类型的端点、连接数据、使用API提交作业为例，演示通过AI DataLake分析数据的操作指导。

操作流程

开始使用如下样例前，请务必按**准备工作**指导完成必要操作。

1. **规划并创建OBS桶**：创建一个用于存储作业脚本的OBS桶。
2. **创建工作空间**：创建一个工作空间并关联LakeFormation实例。
3. **创建计算资源池**：创建运行作业的计算资源。
4. **创建运行作业的RayJob端点**：创建运行作业的RayJob端点，并关联计算资源池（按需弹性模式除外）以获得运行作业的资源。
5. **使用API提交作业**：使用API提交作业至RayJob端点运行，本示例以提交一个简单文本处理的AI DataLake作业为例进行演示。

准备工作

- 已注册账号并实名认证，且账号不能处于欠费或冻结状态。
- 已开通AI DataLake服务并授权使用云服务资源。
- 已开通OBS权限并进行了委托确认。
- 已经获取IAM Token，详细操作请参见**获取IAM用户Token**。

步骤一：规划并创建 OBS 桶

AI DataLake Ray通过OBS服务实现作业提交与数据存储，需要先在OBS控制台进行桶及文件夹创建，并导入样例数据或作业脚本。

1. 登录**华为云管理控制台**。
2. 在页面左上角单击  ，选择“存储 > 对象存储服务 OBS”，进入对象存储服务页面。
3. 选择“桶列表 > 创建桶”，进入创建桶页面，配置相关参数后单击“立即创建”。

- 桶名称：根据界面要求设置桶名称。
- 其他参数根据实际情况选择。

4. 在桶列表页面，单击已创建的桶名称。
5. 将作业脚本压缩为.zip文件，然后在“对象”页签单击“上传对象”，上传压缩包至所创建的OBS桶中。

   例如文本处理脚本“ray-job-code.py”，内容示例如下：

```
import ray
import random
import time
import sys

# 连接Ray集群
ray.init()

@ray.remote
def compute_points_in_circle(num_points: int) -> int:
    inside = 0
    for _ in range(num_points):
        x, y = random.random(), random.random()
        if x*x + y*y <= 1:
            inside += 1
    return inside

def estimate_pi(total_points: int, num_workers: int = 4) -> float:
    start = time.time()
    points_per_worker = total_points // num_workers
    # 分布式执行任务
    tasks = [compute_points_in_circle.remote(points_per_worker) for _ in range(num_workers)]
    total_inside = sum(ray.get(tasks))
    pi = 4.0 * total_inside / total_points
    print(f"估算的圆周率: {pi:.6f}, 耗时: {time.time() - start:.2f}秒")
    return pi

if __name__ == "__main__":
    estimate_pi(total_points=int(sys.argv[1]), num_workers=1)
    ray.shutdown()
```

步骤二：创建工作空间

1. 登录**AI DataLake管理控制台**。
2. 在左侧导航栏中，单击工作空间区域。
3. 在下拉列表中选择“创建工作空间”。
4. 配置工作空间的相关参数。

表 5-1 工作空间的基本信息

类型	参数	示例	说明
基础信息	工作空间名称	workspace_raytest	工作空间的具体名称，同一个账号下的工作空间不可重名。 ● 名称只能包含字母、数字、中划线、下划线。 ● 名称字符长度为4~32位。
	描述	智能驾驶数据分析	对工作空间的简要描述，最长不超过255个字符。

类型	参数	示例	说明
	企业项目	default	如果所建工作空间属于企业项目，可选择对应的企业项目。 企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 关于如何设置企业项目请参考《企业管理用户指南》。 **说明** 只有开通了企业管理服务的用户才显示该参数。
多模数据管理	LakeFormation实例	lakeformation_for_raytest	选择当前工作空间关联的LakeFormation实例，每个工作空间需关联1个LakeFormation实例，空间创建后已关联的实例不支持修改。 • 选择已创建的LakeFormation实例，用于统一管理元数据和数据访问控制。 • 如无可用实例，请先创建LakeFormation实例，具体操作请参考**创建LakeFormation实例**。

5. 确认所有配置信息无误，单击“立即创建”，完成工作空间创建。
   工作空间创建成功后，您可以在工作空间管理的列表中查看新创建的工作空间。

步骤三：创建计算资源池

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤二：创建工作空间**新创建的空间。
2. 单击左侧导航栏的“资源管理”，选择“计算资源池”，单击右上方“购买”，进入购买计算资源池页面。
3. 在“购买计算资源池”界面，填写具体参数。

表 5-2 购买计算资源池参数说明

参数	示例	说明
计费模式	按需计费	• **包年/包月**：预付费模式，按订单的购买周期计费。在购买周期内资源独享，空闲（无作业运行）时不会释放，使用体验更佳，价格比按需计费模式更优惠。 适用于可预估资源使用周期的场景，例如已完成开发进入生产阶段的项目，推荐使用包年包月计费模式。 • **按需计费**：即后付费模式，按实际使用量计费，在购买周期内资源独享，空闲时资源不被释放。

参数	示例	说明
资源池名称	resource-pool-for-raytest	计算资源池的具体名称。 ● 名称只能包含数字、小写英文字母和中划线，且只能以字母开头、以字母或数字结尾。 ● 输入长度不能超过63个字符。
CPU资源	本例配置8个通用计算标准型实例。	CPU是通用型计算资源，适用于各种类型的计算任务。 ● CPU资源的特点：通用性强，适合各种类型的计算任务。相比于GPU和NPU的成本更低。擅长处理顺序执行的任务。 ● CPU资源的适用场景：ETL数据抽取、转换、加载；轻量计算场景，例如小规模数据处理、脚本运行，日志分析场景，例如日志采集、解析、统计分析。 ● CPU资源的具体规格请参考**产品规格**
GPU资源	本例不购买GPU实例	GPU是图形处理器适用于图形渲染和大规模并行计算场景，适合深度学习训练和科学计算。GPU拥有成百上千个计算核心，可以同时处理大量简单计算任务。 ● GPU资源的特点：支持并行计算，适用于批处理作业场景，数千个核心同时计算，处理效率高。天然适合深度学习、神经网络、AI计算场景。 ● GPU资源的适用场景：深度学习，例如神经网络训练、模型调优场景；图形处理场景，例如图像识别、目标检测；视频处理场景，例如视频分析、转码、视频渲染场景，等其他AI科学计算场景。 ● GPU资源的具体规格请参考**产品规格**
NPU资源	本例不购买NPU实例	NPU适用于AI计算场景。NPU资源采用架构优化和指令集，专门加速AI推理任务，具有高能效比的特点。 ● NPU资源的特点：AI计算场景专用，具备高性能、低延迟、推理成本更优的特点。 ● NPU资源的适用场景：AI计算场景、图像识别场景，推荐系统等AI计算设计场景。 ● NPU资源的具体规格请参考**产品规格**
AI DataLake网络	自定义	选择AI DataLake资源池所属网络，该网络基于虚拟私有云（VPC）进行封装。如果不存在可使用的网络，也可单击“创建网络”进行创建。 **一个工作空间仅支持创建一个网络。**

4. 参数填写完成后，单击“立即购买”，在界面上确认当前配置是否正确。

5. 单击“提交”完成创建。等待资源池状态变成“可用”表示当前资源池创建成功。

步骤四：创建运行作业的 RayJob 端点

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤二：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎端点 > AI 计算引擎 Ray”进入Ray端点列表页面。
3. 单击页面右上角的“创建端点”，配置以下参数并单击“立即创建”。
   a. 基础信息配置。

表 5-3 基础信息配置说明

参数	示例	参数说明
端点类型	RayJob	选择端点类型为“RayJob”。 RayJob端点提供按需启动、用完即销毁的“任务执行单元”，任务完成后自动释放资源，节省资源成本，但每次提交任务需要启动集群，存在一定的启动延迟。适用于一次性运行、自动清理资源的场景。
端点名称	ray-job-owin	输入端点名称，是端点的唯一标识符，不可与已存在的端点重名，且创建后不支持修改。 ● 名称只能包含小写英文字母、数字、中划线，且只能以英文字母开头，以英文字母或数字结尾。 ● 输入长度不能超过63个字符。
端点显示名称	ray-job-owin	输入端点显示名称，创建后可修改，用于用户界面展示，方便用户识别和记忆。 ● 名称只能包含中文、英文、数字、中划线。 ● 输入长度不能超过63个字符。

   b. 资源配置。

表 5-4 资源配置说明

参数	示例	参数说明
资源使用模式	预留资源	选择资源使用模式。 ● 预留资源：使用预留资源池，资源独享。适合负载稳定、持续运行的业务场景，如生产项目、关键任务。 ● 混合模式：基线保障，自动扩容。优先消耗预留资源，高峰期自动触发弹性补位。适用于有规律波动的业务。 ● 按需弹性：按任务实际运行时长扣费，无任务不产生费用。适合短期或偶发性的业务需求，如开发测试或临时任务。
选择计算资源池	resource-pool-for-raytest	在下拉框中选择已创建的资源池。如果下拉框中没有可选的资源池，可以单击“购买计算资源”进行购买。具体操作，请参见**购买预留资源池**。

c. 规格配置。

表 5-5 规格配置说明

参数	参数说明
规格配置	选择对应的资源规格并配置资源数量。 ● 资源规格：在下拉框中选择需要配置的资源规格，支持选择CPU、GPU、NPU资源。 ● 资源配额 (Min-Max)：配置资源池分配给当前端点的最低可用资源和资源上限，Min值默认置灰，不能配置。Min值需小于等于Max值，且需大于0。 您可以单击“添加规格”添加多个规格配置，最多支持添加10个，也可以单击“删除”删除配置，至少需保留一个规格配置。 **说明** ● 仅资源使用模式选择“预留资源”和“混合模式”时，配置此参数。 ● 规格配置后，即可在页面右侧的资源配额区域查看端点关联资源池后配置的资源规格及Min和Max资源数量。

d. 委托配置。

表 5-6 委托配置说明

参数	示例	参数说明
委托配置	ai_datalake_admin_trust	委托给该端点的权限，会在运行时自动换取凭证并刷新至Ray集群，您可以在Ray集群中使用该委托凭证调用对应服务。 通过下拉菜单选择已有的委托。创建后不支持修改，若未配置，可单击“新建委托”进行创建。具体操作请参考**创建自定义委托**。

4. 端点创建后，可在列表中查看相关信息，当端点状态变为“运行中”，即可提交作业到该端点中运行。

步骤五：使用 API 提交作业

本示例以通过Postman等工具，调用运行作业API，提交一个简单文本处理的RayJob作业为例进行演示。

1. 查看镜像版本。
   a. 在AI DataLake管理控制台页面的左侧导航栏，选择“资产管理 > 镜像”。
   b. 在“预置镜像”或“自定义镜像”页签下，单击镜像名称。
   c. 查看预置镜像或者自定义镜像的版本。
      - 查看预置镜像版本：单击“版本页签”，在“镜像版本”列，查看预置镜像的版本。
      - 查看自定义镜像版本：在“镜像版本”列，查看自定义镜像的版本。
        如果没有自定义镜像，请手动注册。具体操作，请参见**注册自定义镜像**。
2. 使用Postman等工具，调用运行作业API提交作业。
   - 调用API的具体操作，请参见**如何调用API**。
   - 运行作业API的参数说明，请参见**运行作业**。
   - image_id的值，请通过调用**查看自定义镜像列表**接口查询获取。

   使用API提交作业请求示例如下：

```
POST https://{endpoint}/v2/workspaces/{workspace_id}/ray-jobs
{
    "name": "rayjob0615-4",     #作业名称
    "description": "description",     #作业描述信息
    "config": {
        "runtime_env":{
            "working_dir":"obs://ray-shared-csc/ray-job-code.zip",#需替换为作业脚本存储的obs路径，且必须为.zip文件
        },
        "entrypoint": "python3 -m estimate_pi 1000000",
        "image": {
            "image_id": "dd877a59-5cbd-44c5-b8a7-dc464cca****",     #需替换为实际镜像ID
            "image_version": "01ae8ee6-af8a-48a8-895b-8d4a70bf****"    #需替换为实际镜像版本
        },
        "resource_config": {
            "head_resource_spec": {
                "cpu": "1020",
                "memory": "2048",
                "worker_gpu_amount": 0,
                "worker_npu_amount": 0
```

```
        },
        "worker_resource_spec": [
          {
            "cpu": "1000",
            "memory": "2049",
            "worker_gpu_amount": 0,
            "worker_npu_amount": 0,
            "worker_replicas": 1,
            "enable_autoscaling": true,
            "worker_min_replicas": 1,
            "worker_max_replicas": 1
          }
        ]
      },
      "rayjob_strategy": {
        "queued_timeout": 100000,
        "running_timeout": 100000
      }
    },
    "endpoint_name": "ray-job-owin"    #需替换为实际端点名称
}
```

3. 查看作业。
   a. 在AI DataLake管理控制台页面的左侧导航栏，选择“作业开发 > 作业运行历史”，进入作业运行历史页面。
   b. 单击“AI计算引擎Ray”页签，查看作业信息。

6 基于 PySpark 的数据处理

操作场景

AI DataLake是华为云提供的全托管的湖仓一体大数据分析服务，提供端到端的数据管理与分析能力，支持多种引擎的作业开发，满足多样化的大数据处理需求。

本文以通过创建Spark引擎端点、连接数据、使用API提交作业为例，演示通过AI DataLake分析数据的操作指导。

操作流程

开始使用如下样例前，请务必按**准备工作**指导完成必要操作。

1. **规划并创建OBS桶**：创建一个用于存储作业脚本的OBS桶。
2. **创建工作空间**：创建一个工作空间并关联LakeFormation实例。
3. **创建运行作业的Spark Job端点**：创建运行作业的Spark Job端点。
4. **使用API提交作业**：使用API封装rest请求提交AI DataLake作业至Spark Job端点运行。

准备工作

- 已注册账号并实名认证，且账号不能处于欠费或冻结状态。
- 已开通AI DataLake服务并授权使用云服务资源。
- 已开通LakeFormation服务并创建实例。
- 已开通OBS权限并进行了委托确认。
- 已经获取IAM Token，详细操作请参见**获取IAM用户Token**。

步骤一：规划并创建 OBS 桶

AI DataLake Spark通过OBS服务实现作业提交与数据存储，需要先在OBS控制台进行桶及文件夹创建，并导入样例数据或作业脚本。

1. 登录**华为云管理控制台**。
2. 在页面左上角单击  ，选择“存储 > 对象存储服务 OBS”，进入对象存储服务页面。

3. 选择“桶列表 > 创建桶”，进入创建桶页面，配置相关参数后单击“立即创建”。
   - 桶名称：根据界面要求设置桶名称。
   - 其他参数根据实际情况选择。
4. 在桶列表页面，单击已创建的桶名称。
5. 在“对象”页签，单击“上传对象”，将作业脚本上传至所创建的OBS桶中。

   例如Spark RDD数据转换脚本“simple-rdd-job.py”，完整脚本示例如下：

```
from __future__ import print_function
from pyspark.sql import SparkSession
if __name__ == "__main__":
    spark = SparkSession \
        .builder \
        .appName("SimpleSpark") \
        .getOrCreate()
    # 简单的 RDD 操作
    rdd = spark.sparkContext.parallelize([1, 2, 3, 4, 5], 3)
    data = rdd.map(lambda x: x + 2).collect()
    print("Result:", data)
    spark.stop()
```

步骤二：创建工作空间

1. 登录**AI DataLake管理控制台**。
2. 在左侧导航栏中，单击工作空间区域。
3. 在下拉列表中，单击“创建工作空间”。
4. 在“创建工作空间”页面，配置工作空间的相关参数。

表 6-1 工作空间的基本信息

类型	参数	是否必填	说明
基础信息	工作空间名称	是	工作空间的具体名称，同一个账号下的工作空间不可重名。 • 名称只能包含字母、数字、中划线、下划线。 • 名称字符长度为4~32位。 工作空间名称不区分大小写，系统会自动转换为小写。
	描述	否	对工作空间的简要描述，最长不超过255个字符。
	企业项目	否	如果所建工作空间属于企业项目，可选择对应的企业项目。 企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 关于如何设置企业项目请参考《**企业管理用户指南**》。 **说明** 只有开通了企业管理服务的用户才显示该参数。

类型	参数	是否必填	说明
多模数据管理	LakeFormation实例	是	选择当前工作空间关联的LakeFormation实例，每个工作空间需关联1个LakeFormation实例，空间创建后已关联的实例不支持修改。 • 选择已创建的LakeFormation实例，用于统一管理元数据和数据访问控制。 • 如无可用实例，请先创建LakeFormation实例，具体操作请参考**创建LakeFormation实例**。

5. 确认所有配置信息无误。
   单击“立即创建”按钮，完成工作空间创建。
   工作空间创建成功后，您可以在工作空间管理的列表中查看新创建的工作空间。

步骤三：创建运行作业的 Spark Job 端点

1. 在AI DataLake管理控制台页面，切换页面左上角的工作空间为**步骤二：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎端点 > 批处理引擎Spark”进入Spark端点列表页面。
3. 单击页面右上角的“创建端点”，根据界面提示配置参数。
   a. 配置端点的基础信息。

表 6-2 端点的基础信息配置

参数	示例	参数说明
端点类型	SparkJob	选择端点类型。 • SparkSQL：适用于数据查询和分析的Spark SQL场景，通过编写SQL语句快速完成数据的筛选、聚合、计算等操作，满足数据分析师需要快速探索数据的业务需求。 • SparkJob：适用于复杂数据处理的Spark Job场景，适合执行大规模数据转换、机器学习模型训练、ETL作业等复杂的数据处理任务。
端点名称	spark-job-01	端点名称是系统内部的唯一标识，在API调用、CLI命令中引用该端点。 **说明** 端点名称创建后不可修改。 命名规则： • 名称只能包含小写字母、数字、中划线，且只能以字母开头，以字母或数字结尾。 • 输入长度不能超过63个字符。

参数	示例	参数说明
端点显示名称	数据分析团队专用SparkJob端点-01	端点显示名称创建后可修改。 端点显示名称用于用户界面展示，方便用户识别和记忆。 命名规则： ● 名称只能输入中文、英文、数字、中划线。 ● 输入长度不能超过63个字符
添加描述	数据分析团队专用	填写描述信息，用于详细说明端点的用途、用途背景等信息，帮助其他用户理解该端点的创建目的和使用场景。

b. 配置端点的“资源配置”类信息。

表 6-3 端点的资源配置

参数	示例	参数说明
资源使用模式	按需弹性	当前Spark端点仅支持选择“按需弹性”的资源模式。 “按需弹性”的资源模式使用的是弹性资源，无需提前购买，按实际使用量计费，无业务运行不产生费用。 适用于短期或偶发性的业务需求，如开发测试或临时任务。
端点CPU使用最大值	216	选择“按需弹性”的资源模式时需要配置端点CPU使用最大值，即设置弹性资源的使用上限，防止突发流量导致资源无限扩展，耗尽集群算力。 同时系统根据设置的CPU使用最大值可以进行合理的资源调度和分配。

c. 配置端点的“调度配置”类信息。

表 6-4 调度配置信息

参数	示例	参数说明
端点最大并发作业数量	5	设置端点同时执行的作业任务数，通过资源分流避免单点过载，确保多用户并发访问时的响应稳定性。

参数	示例	参数说明
作业调度策略	随机（RANDOM）	• 随机（RANDOM）：根据随机选择的方式选择作业进行处理。每次调度时，随机选择一个作业执行。 • 先进先出（FIFO）：指作业按照到达的顺序排队等待处理。最早到达的作业最先被处理，然后是第二个到达的作业，依此类推。

d. 配置Spark参数。

以“key=value”的形式设置提交Spark作业的属性。

- **键（Key）**：输入Spark作业所需的配置参数名称，例如spark.executor.memory等。
- **值（Value）**：为对应的键设置参数值，例如：4294967296。内存的默认单位为byte。

e. 配置端点的“存储配置”信息。

表 6-5 存储配置信息

参数	示例	参数说明
选择OBS桶	aidatalake-obs	通过下拉菜单选择已有的OBS桶。用于存储作业日志和临时数据等信息。 创建后不支持修改，若未配置，可单击“新建OBS桶”进行创建。
委托名	aidatalake-agency-log	配置用于读写作业event log、日志和临时数据等信息所使用的委托（即权限委托，如IAM委托）。 通过下拉菜单选择已有的委托。创建后不支持修改，若未配置，可单击“新建委托”进行创建。具体操作请参考**创建自定义委托**。

f. 配置端点的“作业委托配置”信息。

参数	示例	参数说明
委托名	aidatalake-agency-job	委托给该端点的权限，会在运行时自动换取凭证并刷新至Spark集群。用户可在Spark集群中使用该委托凭证调用对应服务。具体操作请参考**创建自定义委托**。

4. 完成上述信息的配置后，单击“立即创建”，系统按照配置要求开始创建端点。

端点创建完成后，可在列表中查看相关信息，端点状态变为“运行中”后，即可提交作业到该端点中运行。

步骤四：使用 API 提交作业

执行脚本“request.py”，即可提交一个AI DataLake Spark Job至指定Spark Job端点。本示例通过requests库展示一个简单示例，例如“request.py”，模拟通过OpenAPI从创建Spark Job到提交Spark Job的完整流程，请求脚本示例如下：

```
# -*- coding: utf-8 -*-
import requests
import urllib3
# 禁用 SSL 警告
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
# ==================== 配置区域 ====================
# 替换为实际值
TOKEN = "XXX"  # IAM的token
WORKSPACE_ID = "3fa4b5a6-9208-4254-b823-119aa29c4016"  # 工作空间的ID
X_CLIENT_TOKEN = "68d2002f-866c-4c4d-84d2-747137b12e9d" # 服务事务ID，用于追踪请求，可自定义
# 请求头
HEADERS = {
    "X-Client-Token": X_CLIENT_TOKEN,
    "X-auth-token": TOKEN,
    "Content-Type": "application/json"
}
ENDPOINT = "fabric.cn-southwest-2.myhuaweicloud.com" # 实际使用时这个ENDPOINT为对应region endpoint的值，此处为贵阳一地址

# API URL
URL_SUBMIT = f"{ENDPOINT}/v2/workspaces/{WORKSPACE_ID}/spark-jobs"

# 作业配置
JOB_CONFIG = {
    "execution_mode": "batch",
    "name": "sparkpython_airan", # 作业名称
    "endpoint_name": "spark_endpoint", # spark job endpoint名称
    "catalog_name": "spark_catalog",  # catalog名称
    "description": "spark_test", # 作业描述
    "job_agency": "aidatalake-agency-job",  # 作业代理名称
    "spark_version": "4.0.0",  # spark版本号
    "job_config": {
        "job_type": "spark_python_job",
        "spark_py_parameter": {
            "main_python_file": "obs://bucket/dir1/",  # 即作业脚本存储的obs路径
            "main_args": []
        }
    },
    "resource_config": {
        "executor_number": 3,
        "driver_resource_spec": {
            "cpu": 2,
            "memory": "4GB",
            "disk": 10
        },
        "executor_resource_spec": {
            "cpu": 2,
            "memory": "4GB",
            "disk": 10
        }
    },
    "spark_config": {
        "spark.executor.memoryOverhead": "1g",
        "spark.driver.memoryOverhead": "512m",
        "spark.dynamicAllocation.enabled": "true",
        "spark.dynamicAllocation.maxExecutors": "3",
        "spark.shuffle.service.enabled": "false",
        "spark.dynamicAllocation.shuffleTracking.enabled": "true",
        "spark.default.parallelism": "4000",
        "spark.hadoop.fs.obs.security.provider": "com.huawei.spark.provider.JobAgencyCredentialProvider"
    },
    "restore_strategy": {
        "max_retry": 0,
```

```
        "retry_delay": 30,
        "launching_timeout": 10800,
        "running_timeout": -1
    },
    "labels": [
        {
            "key": "environment0",
            "value": "dev0"
        }
    ]
}
def submit_spark_job(config=None):
    """提交 Spark Job"""
    payload = config or JOB_CONFIG
    try:
        response = requests.post(
            URL_SUBMIT, headers=HEADERS, json=payload, verify=False
        )
        if response.status_code in [200, 201, 202]:
            result = response.json()
            job_id = result.get("id", "unknown")
            print(f"作业提交成功！")
            print(f"作业ID: {job_id}")
            print(f"响应: {response.text}")
            return job_id
        else:
            print(f"作业提交失败 (Code: {response.status_code})")
            print(response.text)
            return None
    except Exception as e:
        print(f"请求异常: {e}")
        return None


if __name__ == "__main__":
    submit_spark_job()
```