智能数据湖 AI Datalake

快速入门

文档版本 01

发布日期 2026-04-14

华为技术有限公司

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

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

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

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

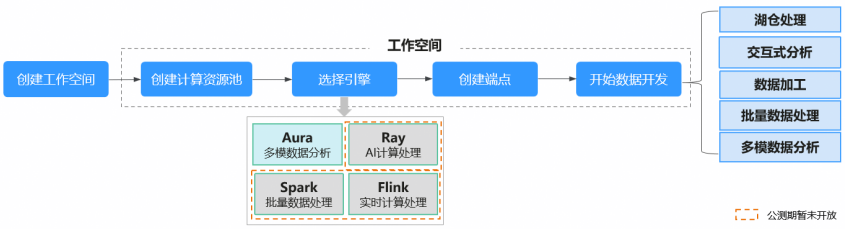


表 1-2 AI DataLake 使用流程

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**
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**

关键步骤	说明	详细操作链接
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。	**创建并配置端点**
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：

AI DataLake在服务公测期间，仅提供Aura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：快速使用多模数据引擎Aura分析智能驾驶数据**

  通过aura_frame 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。

2 快速上手使用 AI DataLake 查询数据

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Aura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“daata.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. 登录管理控制台。
2. 在页面左上角单击图标，选择“存储 > 对象存储服务”，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-job”。
- 其他参数根据实际情况选择。

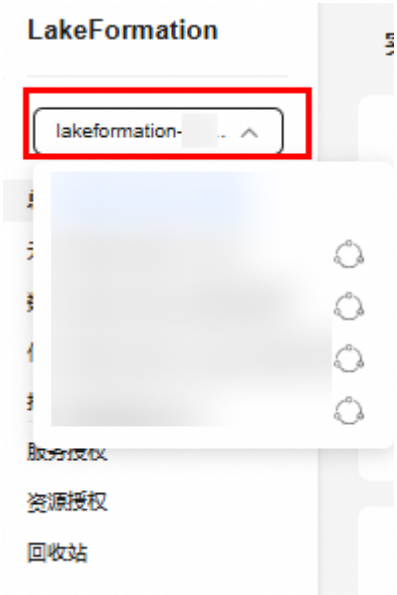
4. 在并行文件系统页面，单击已创建的文件系统名称，例如“aura-job”。
5. 在左侧导航栏选择“概览”，查看并记录“Endpoint(终端节点)”参数值。
6. 在左侧导航栏选择“文件”，单击“新建文件夹”，填写待创建的文件夹名称，单击“确定”。继续单击该新创建的文件夹名称，单击“新建文件夹”，可以创建其子文件夹。
7. 参考该步骤，依次创建用于存放CSV数据文件和结果文件、及LakeFormation元数据的路径，例如：
   - 数据文件
     - CSV数据文件存储路径：aura-job/input
     - 数据处理结果文件存储路径：aura-job/output
   - 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
8. 将本地数据文件（例如“daata.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“元数据 > Catalog”。
   b. 单击“创建Catalog”，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. 在左侧导航栏选择“元数据 > 数据库”。
   b. 在右上角“Catalog”后的下拉框中选择“catalog1”。
      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。
   c. 单击“创建数据库”，配置相关参数后，单击“提交”。
      - 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。
   d. 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	配置样例	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。

类型	参数	配置样例	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_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。
   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。

步骤四：创建计算资源池

1. 在AI DataLake管理控制台页面，切换页面右上角的工作空间为**步骤三：创建工作空间**新创建的空间。
2. 单击左侧导航栏的“新建”，在下拉框中选择“计算资源池”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 2-2 购买计算资源池参数说明

参数	配置样例	说明
计费模式	按需计费	选择“按需计费”。 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。

参数	配置样例	说明
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	本例购买1个昇腾AI加速型（B3）1卡	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“可使用”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，切换页面右上角的工作空间为**步骤三：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎管理 > 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	配置样例	参数说明
端点类型	Job端点	选择端点类型。 • Job 端点：用于执行定时调度任务，保障大规模数据稳定处理。
端点名称	aura_endpoint_test001	输入端点名称。 • 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。
资源使用模式	混合模式	选择资源使用模式。 • 预留模式：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。详细介绍请参见**资源使用模式：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。详细介绍请参见**资源使用模式：混合模式**。
选择资源池	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	配置样例	参数说明
CPU资源	8	配置对应CPU的保障配额及最大配额。 ● 静态余量：资源池中此规格资源尚未被“保障分配”的剩余容量。 ● 保障配额 (Min)：资源池分配给当前端点的最低可用资源，确保核心业务有资源可用。 – 预留模式： 该数值需与最大配额值相同，且需大于0。 – 混合模式 该数值需小于等于最大配额值，且需大于0。 ● 最大配额 (Max)：资源池分配给当前端点的资源上限，防止资源池资源被完全占用。 – 预留模式：参数取值范围为0~*静态余量值*，且需与保障配额值相同。 – 混合模式：参数取值可根据实际业务进行配置，即预估业务需使用的最大CPU配额。参数值需大于0。 CPU为通用计算处理器，适合数据处理、ETL、批处理等任务。
GPU资源	不涉及	配置对应GPU的保障配额及最大配额。 ● 静态余量：资源池中此规格资源尚未被“保障分配”的剩余容量。 ● 保障配额 (Min)：资源池分配给当前端点的最低可用资源，确保核心业务有资源可用。 – 预留模式： 该数值需与最大配额值相同，且需大于0。 – 混合模式 该数值需小于等于最大配额值，且需大于0。 ● 最大配额 (Max)：资源池分配给当前端点的资源上限，防止资源池资源被完全占用。 – 预留模式：参数取值范围为0~*静态余量值*，且需与保障配额值相同。 – 混合模式：参数取值可根据实际业务进行配置，即预估业务需使用的最大CPU配额。参数值需大于0。 GPU为图形处理器，具有强大的并行计算能力。

参数	配置样例	参数说明
NPU资源	24	配置对应NPU的保障配额及最大配额。 ● 静态余量：资源池中此规格资源尚未被“保障分配”的剩余容量。 ● 保障配额 (Min)：资源池分配给当前端点的最低可用资源，确保核心业务有资源可用。 – 预留模式： 该数值需与最大配额值相同，且需大于0。 – 混合模式 该数值需小于等于最大配额值，且需大于0。 ● 最大配额 (Max)：资源池分配给当前端点的资源上限，防止资源池资源被完全占用。 – 预留模式：参数取值范围为0~*静态余量值*，且需与保障配额值相同。 – 混合模式：参数取值可根据实际业务进行配置，即预估业务需使用的最大CPU配额。参数值需大于0。 NPU为神经网络处理器，擅长AI推理任务。
日志对接LTS	勾选“日志对接LTD”	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“已就绪”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”右侧的按钮进入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”页面，配置如**表2-4**所示的参数。

表 2-4 作业参数

参数	说明
作业名称	自定义作业的名称，只能包含英文字母、数字、中文、“-”、“_”、“.”，且长度为1～128个字符。
作业类型	选择作业的类型，选择“批处理作业”。 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	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	选择作业所属的目录，默认为根目录。
责任人	填写该作业的责任人。
作业优先级	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。

参数	说明
委托配置	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。
日志路径	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	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“数据集成”中“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，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	说明
责任人	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。

参数	说明
作业委托	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	自动匹配创建作业时配置的作业优先级，此处支持修改。
实例超时时间	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	配置实例超时是否忽略等待时间。 ● 如果勾选上，表示实例运行时等待时间不会被计入超时时间。 ● 如果未选上，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	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	选择“AI DataLake”。
镜像	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	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	说明
资源配置	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	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/daata.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	是否必选	说明
节点状态轮询时间（秒）	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	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	是否必选	说明
失败重试	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	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“引擎管理 > 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”，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 快速使用多模数据引擎 Aura 分析智能驾驶数据

操作场景

在智能驾驶数据处理场景中，车辆会采集大量的视频、图像数据，对于这些多模数据的处理，要经过预处理、向量化等环节，才能为构建智驾模型训练数据集提供数据输入，这种数据密集型、计算密集型的分析场景给计算资源的调度和数据存储的I/O能力提出了极高的要求。

在传统的数据分析中，开发人员通过配置串行流水线，先在CPU执行解码任务，然后将数据存盘，再调度NPU资源执行推理任务。这种方式存在以下问题：资源调度效率低、算力利用率低、数据读写效率低等问题，异构资源的管理增加了工程运维的复杂性。

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，让视频解码、向量化等预处理环节流水线式执行，让智能驾驶数据生产业务流实现了从串行到并行的架构升级，真正释放了异构算力的全部潜力。

本节操作以智能驾驶场景的数据处理为示例，介绍在AI DataLake使用多模数据引擎Aura完成多模数据处理与分析的操作步骤。

在作业开发过程中使用了aura_frame SDK，请您联系客户支持提前获取aura_frame SDK安装包。

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。

方案架构

AI DataLake搭载Aura多模数据分析引擎，实现跨异构资源(One Pool)、不落盘(One Flow)的数据处理流水线。

- AI DataLake将CPU、GPU、NPU等异构资源进行统一池化抽象，实现资源的弹性调度与高效利用。通过统一资源池，将CPU、GPU、NPU异构资源进行统一管理。能够根据业务负载动态调整资源分配，实现资源的高效利用。
- AI DataLake通过LakeFormation统一管理元数据，实现数据处理过程中全程不落盘，大幅提升处理效率。

  解码算子在CPU节点执行，向量化处理算子在NPU节点执行。Aura引擎自动通过LakeFormation完全避免中间数据落盘。这种设计消除了传统数据处理中磁盘I/O带来的性能瓶颈，显著降低数据处理延迟。

图 3-1 Aura 多模数据分析-智能驾驶数据预处理解决方案架构图

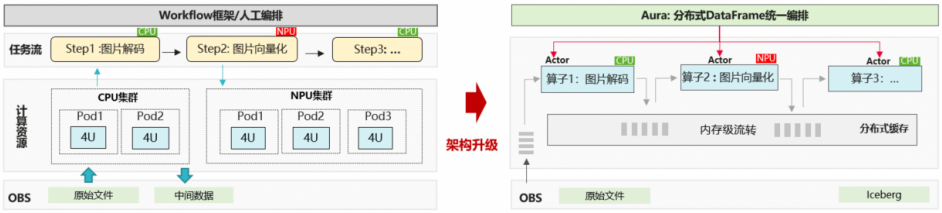
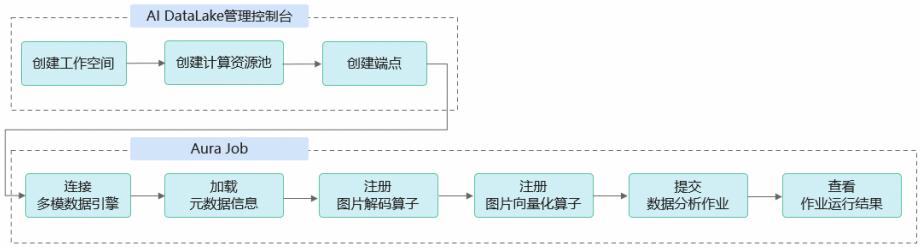


表 3-1 与传统方案的对比

特性	传统方案	Aura方案
跨架构协同	串行且割裂：先在CPU跑解码任务，然后将数据存盘，再去NPU跑推理任务。	业务逻辑如同在一个单机进程内，资源调度无感知，而物理资源自动跨机流转。
数据流转	数据需重复落盘存储：中间图片数据必须写入对象存储，读写I/O效率低，存储资源成本增加。	数据不落盘，依赖纯内存/网络流转，数据在内存中解码后直接流向NPU节点。
资源利用率	计算资源闲置，利用效率低，任务串行，部分资源需要等待数据I/O。	计算资源利用效率高：流水线并行（Pipeline Parallelism）设计，异构算力不再闲置。

操作流程

图 3-2 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. **数据接入**
   a. 获取Aura端点信息，建立与多模态数据引擎Aura的链接。
   b. 通过Lazy Mode读取元数据表。
3. **算子注册**
   注册数据处理算子，构建数据处理流水线：
   - 图片解码算子：支持多种格式的视频/图像解码。
   - 图片向量化算子：将图像转换为高维特征向量。
4. **执行作业就查看结果**
   a. 执行多模态数据处理流水线。
   b. 查看处理后的多模态数据，输出高质量的特征向量，为智驾模型训练提供数据输入。

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. 登录管理控制台。
2. 在页面左上角单击图标，选择“存储 > 对象存储服务”，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。
6. 参考该步骤，依次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

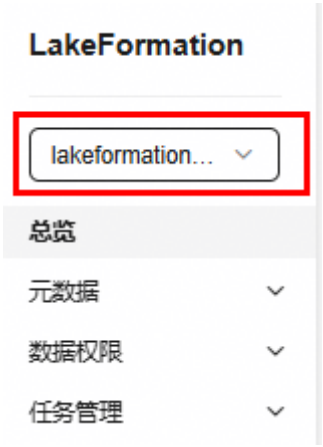
步骤二：规划并创建 Lakeformation 实例、Catalog、数据库

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. 登录管理控制台。

2. 在页面左上角单击图标，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 3-3 切换目标实例



5. 创建Catalog。
   a. 在左侧导航栏选择“元数据 > Catalog”。
   b. 单击“创建Catalog”，配置以下参数后，单击“提交”。

参数	配置样例	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。

参数	配置样例	参数说明
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。
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. 在左侧导航栏选择“元数据 > 数据库”。
   b. 在右上角“Catalog”后的下拉框中选择步骤5创建的Catalog。
   c. 单击“创建数据库”，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。

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。

参数	配置样例	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。

参数	配置样例	参数说明
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。
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。

d. 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 3-2 创建工作空间参数说明

类型	参数	配置样例	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。

类型	参数	配置样例	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_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”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。

步骤四：创建计算资源池

1. 在AI DataLake管理控制台页面，切换页面右上角的工作空间为**步骤三：创建工作空间**新创建的空间。
2. 单击左侧导航栏的“新建”，在下拉框中选择“计算资源池”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 3-3 购买计算资源池参数说明

参数	配置样例	说明
计费模式	按需计费	选择“按需计费”。 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。

参数	配置样例	说明
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**
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	本例购买1个昇腾AI加速型（B3）1卡	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“可使用”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，切换页面右上角的工作空间为**步骤三：创建工作空间**新创建的空间。
2. 在左侧导航栏选择“引擎管理 > 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 3-4 创建 Aura 引擎端点

参数	配置样例	参数说明
端点类型	Job端点	选择端点类型。 • SQL 端点：提供交互式查询，满足BI报表与实时分析需求。 • Job 端点：用于执行定时调度任务，保障大规模数据稳定处理。
端点名称	aura_endpoint_test001	输入端点名称。
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。
CPU资源	8	配置对应CPU的保障配额及最大配额。 CPU为通用计算处理器，适合数据处理、ETL、批处理等任务。
GPU资源	不涉及	配置对应GPU的保障配额及最大配额。 GPU为神经网络处理器，擅长AI推理任务。
NPU资源	24	配置对应NPU的保障配额及最大配额。 NPU为图形处理器，具有强大的并行计算能力。

参数	配置样例	参数说明
日志对接LTS	勾选“日志对接LTD”	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“已就绪”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. 获取**步骤五：创建运行作业的Aura端点**中创建的端点的基本信息。
2. 使用aura_frame配置与端点的连接：

```
from fabric_data.multimodal import ai_lake
import logging
import os
access_key = os.environ.get("access_key")
secret_key = os.environ.get("secret_key")
target_database = "multimodal_lake"
# 建立连接
conn = ai_lake.connect(
fabric_endpoint="100.85.xxx.xxx:xxxxx",
fabric_endpoint_id="8a708bbf-f862-4c53-9622-xxxxxxxxx",
fabric_workspace_id="ca319048-b07c-498c-97df-xxxxxxxxxx",
lf_catalog_name="fabricsql_default",
lf_instance_id="0102c9cf-0759-4478-b42d-xxxxxxxxxx",
access_key=access_key,
secret_key=secret_key,
default_database=target_database,
use_single_cn_mode=True,
logging_level=logging.WARNING
)
conn
```

连接成功的回显信息：

```
<fabric_data.multimodal.ai_lake.FabricConnection at 0x2717948e990>
```

步骤七：加载原始数据（元信息表）

通过 Lazy Mode读取元数据表，加载Schema。

输入数据：包含 camera_type (摄像头类型) 和 image_uri (图片存储路径) 的表。

```
metadata_table_name = "image_object_table"
ds = conn.load_dataset(metadata_table_name).limit(3)
ds.execute(
```

返回结果：

```
camera_type image_uri
0 FRONT_FISHEYE obs://aura-test/cherry-data/A_1_417.3375.dat\r
1 FRONT_LEFT obs://aura-test/cherry-data/B_30_506.1382.dat\r
2 FRONT_TELE obs://aura-test/cherry-data/D_40_536.7318.dat\r
```

步骤八：注册数据处理算子

当前案例中涉及的算子如下，需要您先在本地完成算子的开发，然后再按照本节的操作注册算子。

- DownloadAndDecodeImage：下载原始数据并进行图片解码
- ClipExtractEmbedding：使用CLIP模型进行图片向量化处理

1. 设置UDF归档存储位置(仅需一次)

```
conn.set_function_staging_workspace(
    obs_directory_base="data_ops_dev/user-defined-functions",
    obs_bucket_name="xxx",
    obs_server="obs.xxx.xxx.com",
    access_key=access_key,
    secret_key=secret_key
)
```

2. 注册UDF：以图片Embedding模型离线推理为例。

```
from fabric_data.multimodal.types import image, embedding
from onnx_model import OnnxModel
import numpy as np
class ClipExtractEmbedding:
    def __init__(self):
        self.img_size = 224
        self.model = OnnxModel(providers=['CPUExecutionProvider'])
        self.model.init_onnx_session('img.onnx')
    def __call__(self, img: image.Image) -> embedding.EmbeddingVector:
        image_pil = img.pil_image.convert('RGB')
        image_pil = image_pil.resize((224, 224))
        image_array = np.array(image_pil, dtype=np.float32)
        image_array = image_array.transpose((2, 0, 1))
        image_array = image_array[np.newaxis, :]
        emb = self.model.calc(image_array)
        vector = emb.tolist()[0]
        dims = len(vector)
        return embedding.EmbeddingVector({"vector": vector,"dims": dims})
conn.delete_function('ClipExtractEmbedding', if_it_exists=True)
conn.create_scalar_function(
    ClipExtractEmbedding,
    imports=['img.onnx', "onnx_model.py"],
    packages=["pillow", "onnx==1.17.0", "onnxruntime", "numpy==1.25.2", "protobu
    database=target_database,
    comment="Extract image embedding using CLIP model",
```

3. 执行list_functions查看算子的注册结果。

```
conn.list_functions(database=target_database)
```

表 3-5 注册的算子

序号	udf name	signature	description
0	ClipExtractEmbedding	(img: Image) ->EmbeddingVector	Extract imageembedding usingCLIP model
1	CosineSimilarity	(lhs: EmbeddingVector, rhs:EmbeddingVector) -> doubleprecision	Compute cosinesimilarity between2 embeddingvectors
2	DownloadAndDecodeImage	(text) -> Image	Download raw dataand decode image
3	DownloadAudioFromUrl	(text) -> Audio	None
4	PrivateImageEmbedding	(text) -> EmbeddingVector	Apply embeddingmodel on image
5	SpeakerWordsUDAF	(bigint, text) -> Image	None
6	SpeechRecognitionUDF	(audios: Audio) -> text	None
7	VideoContentAnalyzer	(vio: Video) ->struct	None

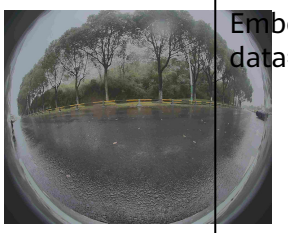
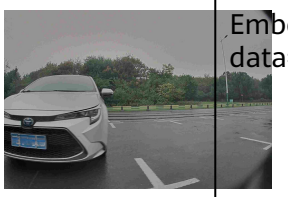
步骤九：提交数据分析作业

```
metadata_table_name = "image_object_table"
ds = conn.load_dataset(metadata_table_name, database=target_database)
# 1. 图片解码算子依赖用户提供的动态链接库，必须在x86机器上运行
ds = ds.map(
fn= '{}.DownloadAndDecodeImage'.format(target_database),
on=['image_uri'],
as_col='image',
num_dpus=0.5,
concurrency=6
)
# 2. 图片转Embedding算子需要使用NPU机器上运行
ds = ds.map(
fn='{}.ClipExtractEmbedding'.format(target_database),
on=['image'],
as_col='embedding',
num_apus=1,
apu_model='NPU/Ascend910B3/1'
)
In [7]:
高：流水线并行（Pipeline Parallelism）
设计，打满异构算力。
# 3. 将处理后的多模态数据存储至Iceberg表中
iceberg_table_name="iceberg_mm_tbl"
```

```
conn.delete_table(iceberg_table_name, if_it_exists=True)
ds.write_iceberg(
target_name=iceberg_table_name,
database=target_database,
create_if_not_exists=True
)
```

步骤十：查看处理后的多模态数据

```
ds = conn.load_dataset(iceberg_table_name).limit(3)
ds.show()
```

camera_type	image_uri	image	embedding
FRONT_FISHEYE	obs://aura-test/cherry-data/A_1_417.3375.dat\r		Embedding(shape=(512,), data=[0.391,-0.83,-0.105,...])
FRONT_LEFT	obs://aura-test/cherry-data/B_30_506.1382.dat\r		Embedding(shape=(512,), data=[0.243,-0.87,-0.191,...])
FRONT_TELE	obs://aura-test/cherry-data/D_40_536.7318.dat\r		Embedding(shape=(512,), data=[0.409,-0.802,-0.2,...])

步骤十一：断开连接并清理计算资源

```
conn.close()
```