ModelArts

用户指南（Lite Cluster）

文档版本 01

发布日期 2024-08-15

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。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

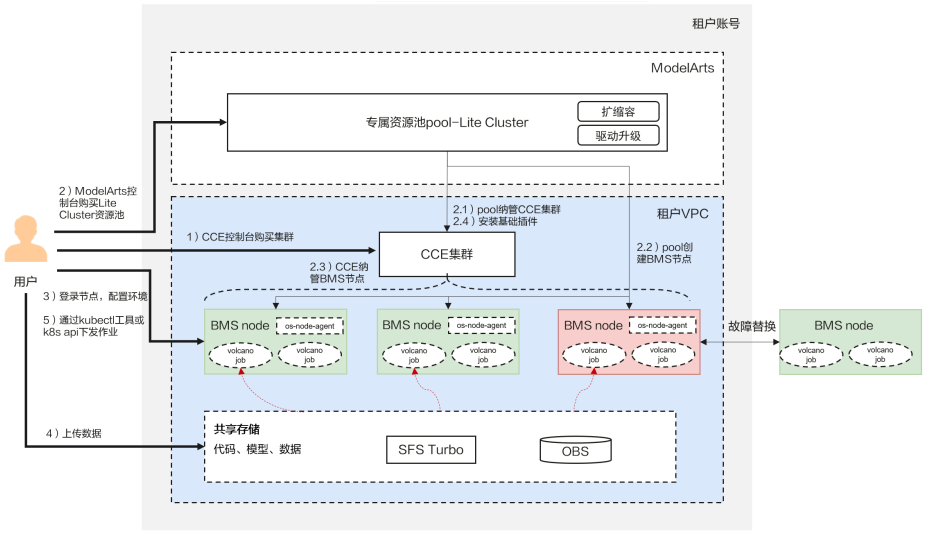
目 录

1 Lite Cluster 使用前必读

1.1 Lite Cluster 使用流程

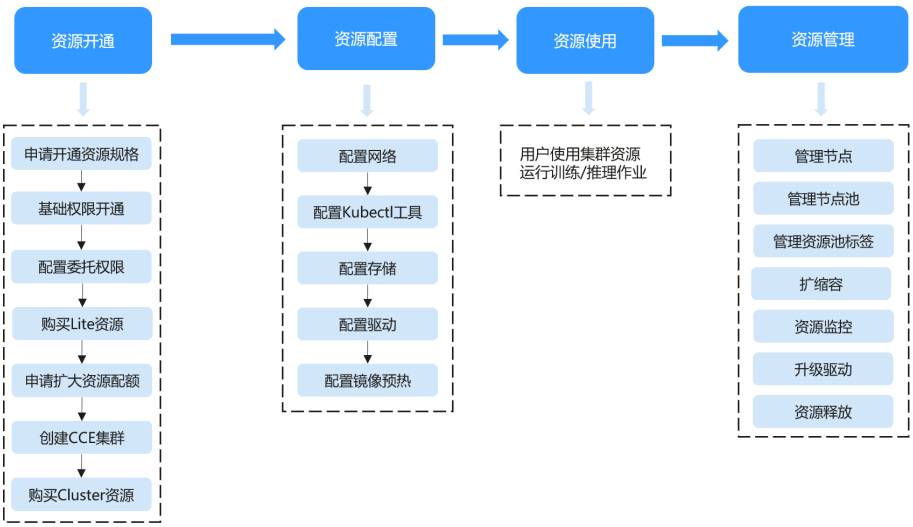
ModelArts Lite Cluster面向k8s资源型用户，提供托管式k8s集群，并预装主流AI开发插件以及自研的加速插件，以云原生方式直接向用户提供AI Native的资源、任务等能力，用户可以直接操作资源池中的节点和k8s集群。本文旨在帮助您了解Lite Cluster的基本使用流程，帮助您快速上手。

图 1-1 资源池架构图



如图所示为Lite Cluster架构图。Lite Cluster基于CCE服务实现对资源节点的管理，因此，用户首先需要购买一个CCE集群。在ModelArts控制台购买Lite Cluster集群时，ModelArts的资源池会先纳管这个CCE集群，然后根据用户设置的规格创建相应的计算节点（BMS/ECS）。随后，CCE会对这些节点进行纳管，并且ModelArts会在CCE集群中安装npuDriver、os-node-agent等插件。完成Cluster资源池的购买后，您即可对资源进行配置，并将数据上传至存储云服务中。当您需要使用集群资源时，可以使用kubectl工具或k8s API来下发作业。此外，ModelArts还提供了扩缩容、驱动升级等功能，方便您对集群资源进行管理。

图 1-2 使用流程



推荐您根据以下使用流程对Lite Cluster进行使用。

1. 资源开通：您需要开通资源后才可使用Lite Cluster，在开通资源前，请确保完成所有相关准备工作，包括申请开通所需的规格和进行权限配置。随后，在ModelArts控制台上购买Lite Cluster资源。请参考**Lite Cluster资源开通**。
2. 资源配置：完成资源购买后，需要对网络、存储、驱动进行相关配置。请参考**Lite Cluster资源配置**。
3. 资源使用：完成资源配置后，您可以使用集群资源运行训练和推理训练，具体案例可参考**Lite Cluster资源使用**。
4. 资源管理：Lite Cluster提供扩缩容、驱动升级等管理手段，您可在ModelArts控制台上对资源进行管理。请参考**Lite Cluster资源管理**。

表 1-1 相关名词解释

名词	含义
容器	容器技术起源于Linux，是一种内核虚拟化技术，提供轻量级的虚拟化，以便隔离进程和资源。尽管容器技术已经出现很久，却是随着Docker的出现而变得广为人知。Docker是第一个使容器能在不同机器之间移植的系统。它不仅简化了打包应用的流程，也简化了打包应用的库和依赖，甚至整个操作系统的文件系统能被打包成一个简单的可移植的包，这个包可以被用来在任何其他运行Docker的机器上使用。
Kubernetes	Kubernetes是一个开源的容器编排部署管理平台，用于管理云平台中多个主机上的容器化应用。Kubernetes的目标是让部署容器化的应用简单并且高效，Kubernetes提供了应用部署、规划、更新、维护的一种机制。使用Lite Cluster需要用户具备一定的Kubernetes知识背景，您可参考**Kubernetes基础知识**。
CCE	云容器引擎（Cloud Container Engine，简称CCE）是一个企业级的Kubernetes集群托管服务，支持容器化应用的全生命周期管理，为您提供高度可扩展的、高性能的云原生应用部署和管理方案。CCE官网文档可参考**云容器引擎**。

名词	含义
BMS	裸金属服务器（Bare Metal Server）是一款兼具虚拟机弹性和物理机性能的计算类服务，为您和您的企业提供专属的云上物理服务器，为核心数据库、关键应用系统、高性能计算、大数据等业务提供卓越的计算性能以及数据安全。
ECS	弹性云服务器（Elastic Cloud Server）是一种可随时自助获取、可弹性伸缩的云服务器，可帮助您打造可靠、安全、灵活、高效的应用环境，确保服务持久稳定运行，提升运维效率。
os-node-agent	ModelArts Lite k8s Cluster节点默认会安装os-node-agent插件，用于对节点进行管理，例如： ● 驱动升级：通过os-node-agent插件下载驱动文件并进行驱动版本升级、回退。 ● 故障检测：通过os-node-agent插件在系统内周期性巡检故障特征，及时发现节点故障。 ● 指标采集：通过os-node-agent插件采集GPU/NPU利用率指标等重要的观测数据，上报到租户侧AOM。 ● 节点运维：授权后，通过os-node-agent插件执行诊断脚本，进行故障定位定界。

1.2 Lite Cluster 高危操作一览表

当您在CCE、ECS或BMS服务控制台直接操作ModelArts Lite Lite Cluster资源时，可能会导致资源池部分功能异常。下表可帮助您定位异常出现的原因，风险操作包括但不限于以下内容。

高危操作风险等级说明：

- 高：对于可能直接导致业务失败、数据丢失、系统不能维护、系统资源耗尽的高危操作。
- 中：对于可能导致安全风险及可靠性降低的高危操作。
- 低：高、中风险等级外的其他高危操作。

表 1-2 操作及其对应风险

操作对象	操作名称	风险描述	风险等级	应对措施
集群	升级、修改、休眠集群、删除集群等。	可能影响ModelArts侧基本功能，包括但不限于资源池管理、节点管理、扩缩容、驱动升级等。	高	不可恢复。
节点	退订、移除、关机、污点管理、切换/重装操作系统等。	可能影响ModelArts侧基本功能，包括但不限于节点管理、扩缩容、驱动升级、带本地盘机型的本地盘数据丢失等。	高	不可恢复。

操作对象	操作名称	风险描述	风险等级	应对措施
	修改网络安全组	可能影响ModelArts侧基本功能，包括但不限于节点管理、扩缩容、驱动升级等。	中	改回原有内容。
网络	修改/删除集群关联网段。	影响ModelArts侧基本功能，包括但不限于节点管理、扩缩容、驱动升级等。	高	不可恢复。
插件	升级、卸载gpu-beta插件。	可能导致GPU驱动使用异常。	中	回退版本、重装插件。
	升级、卸载huawei-npu插件。	可能导致NPU驱动使用异常。	中	回退版本、重装插件。
	升级、卸载volcano插件。	可能导致作业调度异常。	中	回退版本、重装插件。
	卸载ICAgent插件。	可能导致日志、监控功能异常。	中	回退版本、重装插件。
helm	升级、回退、卸载os-node-agent。	导致驱动升级、故障检测、指标采集、节点运维功能异常。	高	联系华为云技术支持重装os-node-agent。
	升级、回退、卸载rdma-sriov-dev-plugin。	可能影响容器内使用RDMA网卡。	高	联系华为云技术支持重装rdma-sriov-dev-plugin。

1.3 不同机型的对应的软件配套版本

由于弹性集群资源池可选择弹性裸金属或弹性云服务器作为节点资源，不同机型的节点对应的操作系统、适用的CCE集群版本等不相同，为了便于您制作镜像、升级软件等操作，本文对不同机型对应的软件配套版本做了详细介绍。

裸金属服务器的对应的软件配套版本

表 1-3 裸金属服务器

类型	卡类型	RDMA网络协议	操作系统	适用范围、约束	依赖插件
NPU	ascend-snt9b	RoCE	• 操作系统：EulerOS 2.10 64bit（推荐） • 内核版本：4.19.90-vhulk2211.3.0.h1543.eulerosv2r10.aarch64 • 架构类型：aarch64	• 集群类型：CCE Standard • 集群版本：v1.23（v1.23.5-r0及以上版本）|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC • 集群转发模式：iptables|ipvs	• huawei-npu • npu-driver • volcano 插件版本匹配关系请见表1-5。
		RoCE	• 操作系统：Huawei Cloud EulerOS 2.0 64bit • 内核版本：5.10.0-60.18.0.50.r865_35.hce2.aarch64 • 架构类型：aarch64	• 集群类型：CCE Turbo • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：ENI • 集群转发模式：iptables|ipvs	

类型	卡类型	RDMA网络协议	操作系统	适用范围、约束	依赖插件
	ascend-snt9	RoCE	• 操作系统：EulerOS 2.8 64bit • 内核版本：4.19.36-vhulk1907.1.0.h619.eulerosv2r8.aarch64 • 架构类型：aarch64	• 集群类型：CCE Standard|Turbo • 集群版本：v1.23（v1.23.5-r0及以上版本）|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC|ENI • 集群转发模式：iptables|ipvs	
GPU	gp-ant8	RoCE	• 操作系统：EulerOS 2.10 64bit • 内核版本：4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC 分布式训练时仅支持容器隧道网络 • 集群转发模式：iptables|ipvs	• gpu-beta • gpu-driver • rdma-sriov-dev-plugin 插件版本匹配关系请见**表1-5**。

类型	卡类型	RDMA网络协议	操作系统	适用范围、约束	依赖插件
	gp-ant1	RoCE	• 操作系统：EulerOS 2.10 64bit • 4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC 分布式训练时仅支持容器隧道网络 • 集群转发模式：iptables|ipvs	
	gp-vnt1	RoCE|IB	• 操作系统：EulerOS 2.9 64bit（仅上海一p6|p6s规格使用） • 内核版本：147.5.1.6.h1099.eulerosv2r9.x86_64 • 架构类型：x86	• 集群类型：CCE Standard| • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC 分布式训练时仅支持容器隧道网络 • 集群转发模式：iptables|ipvs	
			• 操作系统：EulerOS 2.9 64bit（推荐） • 内核版本：4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64 • 架构类型：x86		

- RDMA：Remote Direct Memory Access（RDMA）是一种直接内存访问技术，将数据直接从一台计算机的内存传输到另一台计算机。
- RoCE：RDMA over Converged Ethernet（RoCE）是一种网络协议，允许应用通过以太网实现远程内存访问。
- IB：InfiniBand (IB) 是一种高性能计算机网络通信协议，专为高性能计算和数据中心互连设计。

弹性云服务器的对应的软件配套版本

表 1-4 弹性云服务器

类型	卡类型	操作系统	适用范围	依赖插件
NPU	ascend-snt3p-300i	• 操作系统：EulerOS 2.9 • 架构类型：x86	• 集群类型：CCE Standard、CCE Turbo • 集群版本：v1.23（v1.23.5-r0及以上版本）|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC|ENI • 集群转发模式：iptables|ipvs	• huawei-npu • npu-driver • volcano 插件版本匹配关系请见**表1-5**。
	ascend-snt3	• 操作系统：EulerOS 2.5 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC • 集群转发模式：iptables|ipvs	
		• 操作系统：EulerOS 2.8 • 架构类型：arm	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC 集群转发模式：iptables|ipvs	

类型	卡类型	操作系统	适用范围	依赖插件
GPU	gp-vnt1	• 操作系统：EulerOS 2.9 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC • 集群转发模式：iptables|ipvs	• gpu-beta • gpu-driver • rdma-sriov-dev-plugin 插件版本匹配关系请见**表1-5**。
	gp-ant03	• 操作系统：EulerOS 2.9 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC • 集群转发模式：iptables|ipvs	
	gp-ant1-pcie40	• 操作系统：EulerOS 2.9 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC • 集群转发模式：iptables|ipvs	
	gp-tnt004	• 操作系统：EulerOS 2.9 • 架构类型：x86	• 集群类型：CCE Standard • 集群版本：v1.23|v1.25（推荐） • 集群规模：50|200|1000|2000 • 集群网络模式：容器隧道网络|VPC • 集群转发模式：iptables|ipvs	

插件版本与 CCE 集群版本适配关系

表 1-5 插件版本与 CCE 集群版本适配关系

类别	插件名称	插件版本	适配CCE集群版本	适用范围、约束	插件功能描述
ccePlugin	gpu-beta	2.0.48（推荐）	v1.(23|25).*	GPU	支持在容器中使用GPU显卡的设备管理插件。
		1.2.15	v1.23.*		
	huawei-npu	2.1.5（推荐）	v1.(23|25).*	NPU	支持容器里使用huawei NPU设备的管理插件。
	volcano	1.11.9（推荐）	v1.(23|25).*	NPU	基于Kubernetes的批处理平台。
npuDriver	npu-driver	7.1.0.7.220-23.0.5（推荐） 7.1.0.5.220-23.0.3	无约束	NPU	用于升级、回滚npu驱动。
helm	rdma-sriov-dev-plugin	0.1.0	无约束	BMS、RDMA且非ascend-1980	用于支持容器里使用RDMA网卡。
	memarts	3.23.6-r002	无约束	无约束	近计算侧分布式缓存插件，用于存储加速。
	os-node-agent	6.5.0-20240529142433	无约束	无约束	OS插件，用于故障检测。
icAgent	icagent	default	CCE默认安装当前适配版本	无约束	CCE基础组件，用于日志和监控。
gpuDriver	gpu-driver	515.65.01（推荐） 510.47.03 470.182.03 470.57.02 gpu-driver与系统内核版本有关，请见**表1-6**。			用于升级、回滚gpu驱动，插件依赖gpu-beta版本。

系统内核与 gpu-driver 配套关系

表 1-6 系统内核与 gpu-driver 配套关系

镜像版本	系统内核版本	适配CCE	gpu-driver版本
EulerOS 2.10	4.18.0-147.5.2.15.h1109.eulerosv2r10.x86_64	v1.(23|25|27|28).* 容器隧道网络|VPC|ENI	470.57.02
	4.18.0-147.5.2.5.h805.eulerosv2r10.x86_64	v1.(23|25|27).* 容器隧道网络|VPC|ENI	470.57.02
EulerOS 2.9	4.18.0-147.5.1.6.h841.eulerosv2r9.x86_64	v1.(23|25|27|28).* 容器隧道网络|VPC	470.57.02
EulerOS 2.3	3.10.0-514.44.5.10.h193.x86_64	v1.(23|25).* 容器隧道网络|VPC	470.57.02
	3.10.0-514.44.5.10.h254.x86_64	v1.(23|25).* 容器隧道网络|VPC	470.57.02

2 Lite Cluster 资源开通

集群资源开通流程

开通集群资源过程中用户侧需要完成的任务流程如下图所示。

图 2-1 用户侧任务流程

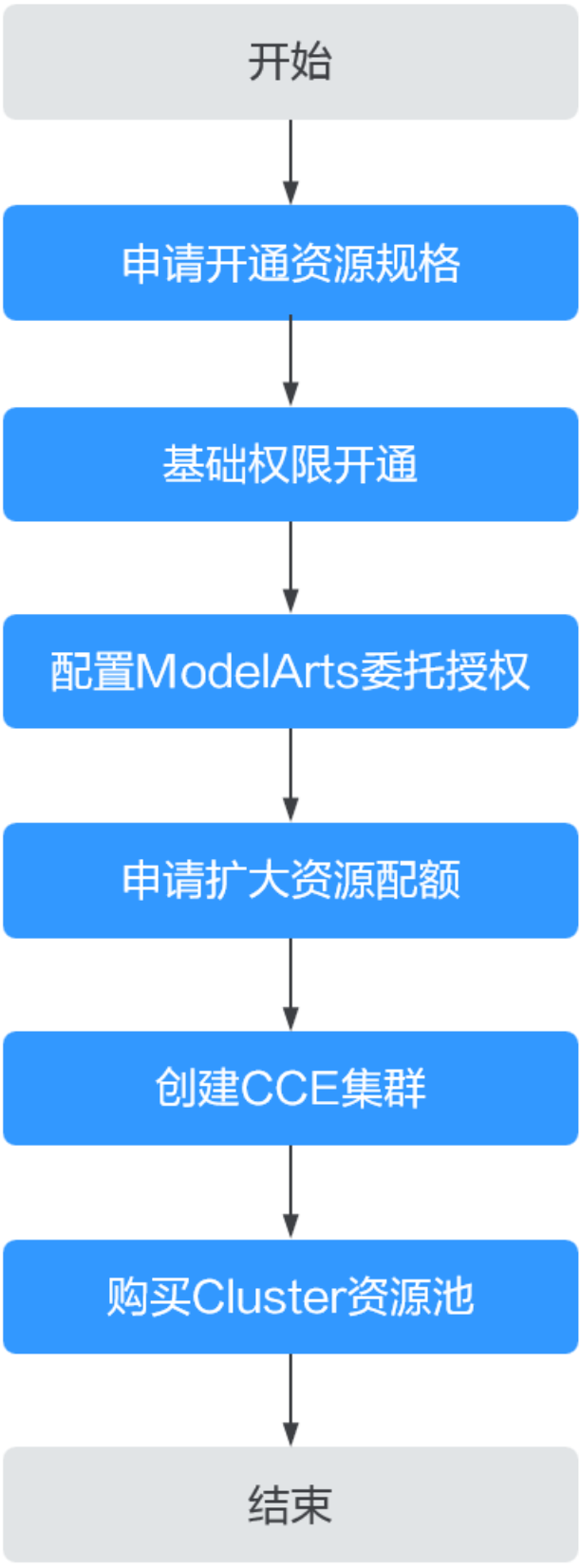


表 2-1 Cluster 资源开通流程

任务	说明
Step1 申请开通资源规格	当前部分规格为受限购买，需要提前联系客户经理申请开通资源规格，预计1~3个工作日内开通（若无客户经理可提交工单反馈）。
Step2 基础权限开通	为子用户开通使用资源池所需要的权限。
Step3 在ModelArts上创建委托授权	第一次使用ModelArts时需要创建委托授权，授权允许ModelArts代表用户去访问其他云服务。 如果之前已经创建过委托授权，需要更新委托相应的权限。
Step4 申请扩大资源配额	集群所需的ECS实例数、内存大小、CPU核数和EVS硬盘大小等资源会超出华为云默认提供的资源配额，因此需要申请扩大配额。 具体的配额方案请联系客户经理获取。 配额需大于要开通的资源，且在购买开通前完成配额提升，否则会导致资源开通失败。
Step5 购买CCE集群	购买Cluster资源池时，需要选择CCE集群，若您没有可用的CCE集群需要提前在CCE控制台购买。
Step6 购买Cluster资源	在ModelArts控制台上购买Cluster资源。

Step1 申请开通资源规格

当前部分规格为受限购买（如modelarts.bm.npu.arm.8snt9b3.d），需要提前联系客户经理申请开通资源规格，预计1~3个工作日内开通（若无客户经理可提交工单反馈）。

Step2 基础权限开通

基础权限开通需要登录管理员账号，为子用户账号开通使用资源池所需的基础权限。

步骤1 登录统一身份认证服务管理控制台。

步骤2 单击目录左侧“用户组”，然后在页面右上角单击“创建用户组”。

步骤3 填写“用户组名称”并单击“确定”。

步骤4 在操作列单击“用户组管理”，将需要配置权限的用户加入用户组中。

步骤5 单击用户组名称，进入用户组详情页。

步骤6 在权限管理页签下，单击“授权”。

图 2-2 “配置权限”

步骤7 在搜索栏输入“ModelArts FullAccess”，并勾选“ModelArts FullAccess”。

图 2-3 ModelArts FullAccess

以相同的方式，依次添加如下权限：

- ModelArts FullAccess
- CTS Administrator
- CCE Administrator
- BMS FullAccess
- IMS FullAccess
- DEW KeypairReadOnlyAccess
- VPC FullAccess
- ECS FullAccess
- SFS Turbo FullAccess
- OBS Administrator
- AOM FullAccess
- TMS FullAccess
- BSS Administrator

步骤8 单击“下一步”，授权范围方案选择“所有资源”。

步骤9 单击“确认”，完成基础权限开通。

----结束

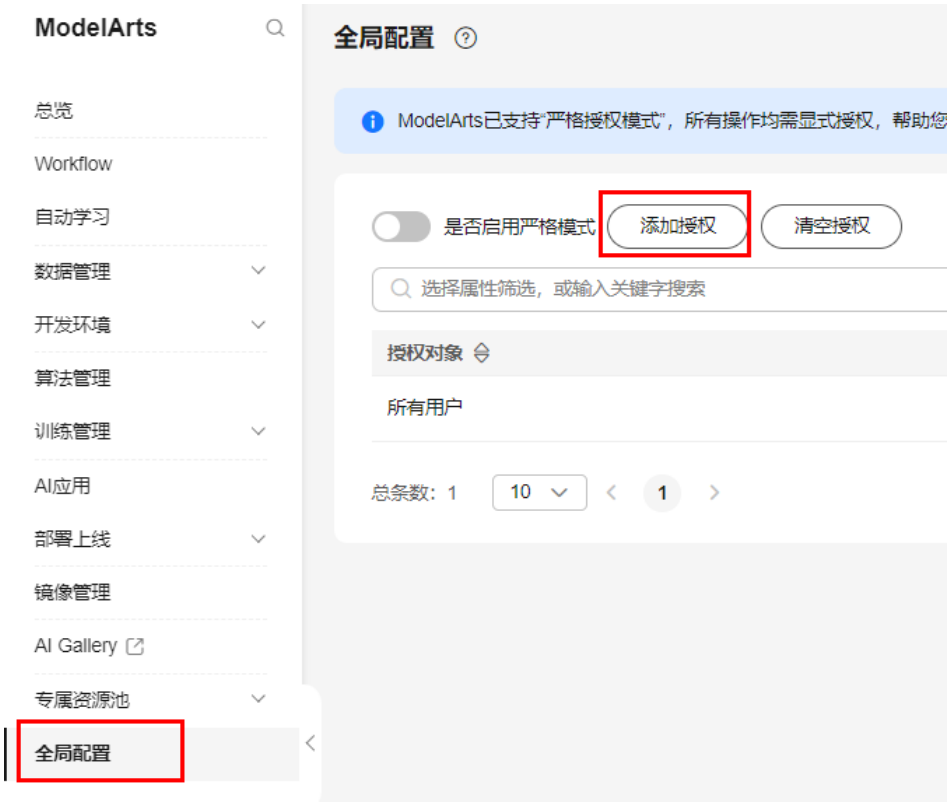
Step3 在 ModelArts 上创建委托授权

- 新建委托

  第一次使用ModelArts时需要创建委托授权，授权允许ModelArts代表用户去访问其他云服务。使用ModelArts Lite的资源池需要授权允许ModelArts代表用户访问云容器引擎服务CCE、裸金属服务BMS、镜像服务IMS和密钥管理服务DEW。

  进入到ModelArts控制台的“全局配置”页面，单击“添加授权”，根据提示进行操作。

图 2-4 添加授权



- 更新委托

  如果之前给ModelArts创过委托授权，此处可以更新授权。

  a. 进入到ModelArts控制台的“专属资源池”页面，查看是否存在授权缺失的提示。

  b. 如果有授权缺失，根据提示，单击“此处”更新委托。根据提示选择“追加至已有授权”，单击“确定”，系统会提示权限更新成功。

Step4 申请扩大资源配额

由于AI机型规格相对较大，资源池所需的ECS实例数、内存大小、CPU核数和EVS硬盘大小很可能会超出华为云默认提供的资源配额，因此需要申请扩大配额。请先联系客户经理确认资源配额提升具体方案，再参考本章节申请扩大配额。

步骤1 登录华为云管理控制台。

步骤2 在顶部导航栏单击“资源 > 我的配额”，进入服务配额页面。

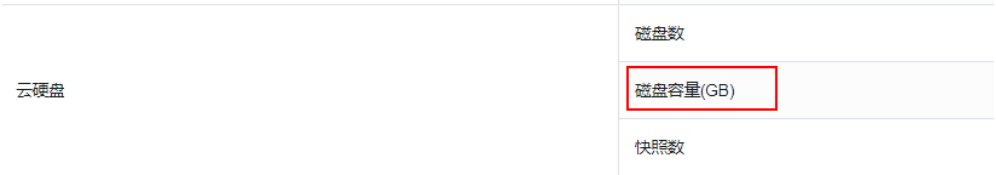
图 2-5 我的配额

步骤3 在服务配额页面，单击右上角的“申请扩大配额”，填写申请材料后提交工单。

申请扩大配额主要是申请弹性云服务器ECS实例数、核心数（CPU核数）、RAM容量（内存大小）和云硬盘EVS磁盘容量这4个资源配额。具体的配额数量请先联系客户经理获取。

图 2-6 ECS 资源类型

图 2-7 云硬盘资源类型



说明

配额需大于需要开通的资源，且在购买开通前完成提升，否则会导致资源开通失败。

----结束

Step5 购买 CCE 集群

购买Cluster资源池时，需要选择CCE集群，若您没有可用的CCE集群，可参考**购买Standard/Turbo集群**进行购买，集群配套版本请参考**不同机型的对应的软件配套版本**。

创建Cluster资源池时，请确保CCE集群为“运行中”状态。

说明

- 当前仅支持CCE集群1.23&1.25&1.28版本。
- 若您没有可用的CCE集群，可先创建CCE集群。CCE 1.28集群版本支持通过控制台、API方式创建，CCE 1.23和CCE 1.25版本支持通过API方式创建。不同版本的CCE集群创建方式请见**Kubernetes版本策略**。
- 若您已有CCE集群，但CCE集群版本低于1.23，则可参考**升级集群的流程和方法**，建议将集群升级至1.28版本。

Step6 购买 Cluster 资源

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“AI专属资源池 > 弹性集群 Cluster”，进入“弹性集群 Cluster”页面。
2. 在“弹性集群 Cluster”页签，单击“购买AI专属集群”，进入购买AI专属集群界面，参见下表填写参数。

表 2-2 专属资源池的参数说明

参数名称	子参数	说明
名称	-	专属资源池的名称。 只能以小写字母开头，由小写字母、数字、中划线（-）组成，不能以中划线结尾。
描述	-	专属资源池的简要说明。
使用场景	-	选择“ModelArts Lite”。
计费模式	-	选择“按需计费”或“包年/包月”模式。 ● 包年/包月 包年/包月是预付费模式，按订单的购买周期计费，适用于可预估资源使用周期的场景，价格比按需计费模式更优惠。 ● 按需计费 按需计费是后付费模式，按云服务器的实际使用时长计费，可以随时开通/删除云服务器。
CCE集群	-	在下拉列表中选择用户账户下已有的CCE集群。如果没有集群，单击右侧的“创建集群”，先去创建集群。集群配套版本请参考**不同机型的对应的软件配套版本**。 创建Cluster资源池时，请确保CCE集群为“运行中”状态。 **说明** ● 当前仅支持CCE集群1.23&1.25&1.28版本。 ● 若您没有可用的CCE集群，可先创建CCE集群。CCE 1.28集群版本支持通过控制台、API方式创建，CCE 1.23和CCE 1.25版本支持通过API方式创建。不同版本的CCE集群创建方式请见**Kubernetes版本策略**。 ● 若您已有CCE集群，但CCE集群版本低于1.23，则可参考**升级集群的流程和方法**，建议将集群升级至1.28版本。

参数名称	子参数	说明
自定义节点名称	-	开启后，可为节点名称添加前缀。 ● 添加前缀后，节点名称由前缀+随机数组成。 ● 输入长度范围为1到64个字符。 ● 前缀必须以小写字母开头，并由小写字母和数字组成，以“-”分隔。例如：node-com。
规格管理	-	支持添加多个规格。限制如下： ● 当选择多个相同规格时，可打开高级选项指定节点池名称，至多只有一个可不指定节点池名称。 ● 选择多个规格的CPU架构必须相同。例如都是X86，或者都是ARM。 ● 如果选择了多个GPU或NPU规格，由于不同规格的参数网络平面不互通，分布式训练时训练速度会受到影响。如果您要做分布式训练，建议您只选择一个GPU或NPU规格。 ● 一个资源池中，最多可添加10种规格。
	规格类型	请根据界面提示选择需要使用的规格。平台分配的资源规格包含了一定的系统损耗，实际可用的资源量小于规格标称的资源。实际可用的资源量可在专属资源池创建成功后，在详情页的“节点”页签中查看。
	可用区	您可以根据实际情况选择“随机分配”或“指定AZ”。可用区是在同一区域下，电力、网络隔离的物理区域。可用区之间内网互通，不同可用区之间物理隔离。 ● 随机分配：系统自动分配可用区。 ● 指定AZ：指定资源池节点在哪个可用区域。考虑系统容灾时，推荐指定节点在同一个可用区。可设置可用区的节点数。
	节点数量	选择专属资源池的节点数，选择的节点数越多，计算性能越强。 当“可用区”选择“指定AZ”时，节点数量会根据可用区的数据自动计算，此处无须再次设置。 **说明** 单次创建时，节点数建议不大于30，否则可能触发限流导致创建失败。 部分局点的部分规格支持整柜购买，此时节点数量会显示为“数量*整柜”，购买的节点总数为两者的乘积。整柜购买可实现不同任务间的物理隔离，避免通信冲突，在任务规模增大的同时保证计算性能线性度不下降。整柜下的节点生命周期需保持一致，需要一起创建、一起删除。 **图 2-8** 整柜购买 

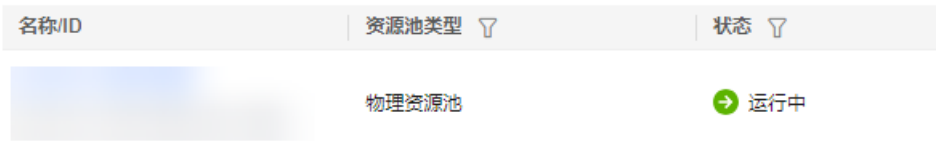
参数名称	子参数	说明
	高级选项	开启后，可设置以下参数： ● 容器引擎空间大小（dockersize）。若没开启，默认容器引擎空间大小为50GiB。默认值与最小值为50GiB，不同规格的最大值不同，数值有效范围请参考界面提示。 ● 容器引擎：容器引擎是Kubernetes最重要的组件之一，负责管理镜像和容器的生命周期。Kubelet通过Container Runtime Interface (CRI) 与容器引擎交互，以管理镜像和容器。 您可以在创建时资源池时选择容器引擎，也可在资源池创建完成后，在扩缩容界面修改。其中Containerd调用链更短，组件更少，更稳定，占用节点资源更少，Containerd和Docker差异对比请见**容器引擎**。 若CCE集群版本低于1.23，仅支持选择Docker作为容器引擎。若CCE集群版本大于等于1.27，仅支持选择Containerd作为容器引擎。其余CCE集群版本，支持选择Containerd或Docker作为容器引擎。 ● 节点池名称：新建节点池的名称，可自定义，若未指定则默认使用“规格-default”作为节点池名称。当多个节点池选择相同规格时，至多只有一个可不指定节点池名称。 ● 虚拟私有云：默认为CCE集群所在VPC网络，不可修改。 ● 节点子网：选择同一VPC网络下的子网作为节点子网，新创建的节点将会使用该子网资源。 ● 关联安全组：用于指定节点池创建出来的节点使用的安全组。最多选择4个安全组。节点安全组需要放通一些端口以保障节点通信。若不关联安全组将会使用集群中默认的节点安全组规则。 ● 资源标签：通过为资源添加标签，可以对资源进行自定义标记，实现资源的分类。 ● K8S标签：设置附加到Kubernetes对象（比如Pod）上的键值对。最多可以添加20条标签。使用该标签可区分不同节点，可结合工作负载的亲和能力实现容器Pod调度到指定节点的功能。 ● 污点：默认为空。支持给节点加污点来设置反亲和性，每个节点最多配置20条污点。 ● 安装后执行脚本：请输入脚本命令，命令中不能包含中文字符，需传入Base64转码后的脚本，转码后的字符数不能超过2048。脚本将在Kubernetes软件安装后执行，不影响Kubernetes软件安装。 **说明** ● 暂不支持资源池中的存量节点池修改名称。 ● 请不要在安装后执行脚本中使用reboot命令立即重启，如果需要重启，可以使用“shutdown -r 1”命令延迟1分钟重启。

参数名称	子参数	说明
自定义驱动	-	默认关闭。部分GPU和Ascend规格资源池允许自定义安装驱动。集群中默认会安装驱动，无需用户操作。只有需要指定驱动版本时，需要开启。建议在购买Cluster资源时，确定需要的驱动版本并选择对应驱动。
GPU驱动/Ascend驱动	-	打开“自定义驱动”开关，显示此参数，选择GPU/Ascend驱动。若规格类型为GPU则显示“GPU驱动”，若规格类型为Ascend则显示“Ascend驱动”。 gpu-driver配套版本请参考**不同机型的对应的软件配套版本**。
购买时长	-	选择购买时长。只有选择“包年/包月”计费模式时才需填写。
登录方式	-	集群登录方式，可以设置密码登录，也可以设置密钥对登录。 ● 密码登录：默认用户名为root，用户自己设置密码。 ● 密钥对（KeyPair）登录：可以选择已有的密钥对，或者单击右侧的“创建密钥对”，先去创建一个密钥对。
高级选项	-	选中“现在配置”，可配置标签信息。 ModelArts支持对接标签管理服务TMS，在ModelArts中创建资源消耗性任务（例如：创建Notebook、训练作业、推理在线服务）时，可以为这些任务配置标签，通过标签实现资源的多维分组管理。 标签详细用法请参见**ModelArts如何通过标签实现资源分组管理**。

单击“下一步”确认规格。规格确认无误后，单击“提交”，即可创建专属资源池。

- 当资源池创建成功后，资源池的状态会变成“运行中”，当“节点个数”中的“可用”和“总数”值大于0时，资源池才能下发任务。

  图 2-9 查看资源池

- 可以将鼠标放在“创建中”字样上，查看当前创建过程详情。若点击查看详情，可跳转到“操作记录”中。

图 2-10 创建中状态

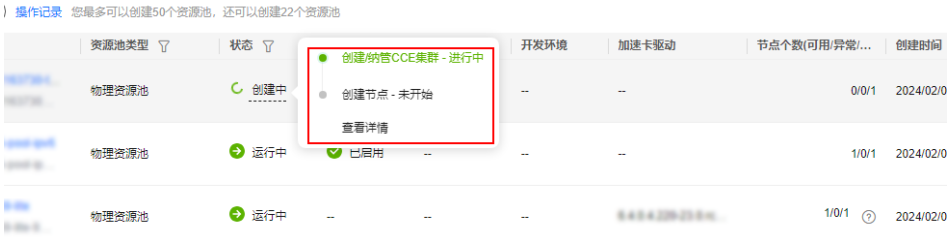


图 2-11 查看详情

- 可以在资源池列表左上角“操作记录”中查看资源池的任务记录。

图 2-12 操作记录

图 2-13 查看资源池状态

当资源池创建成功后，资源池的状态会变成“运行中”。单击集群资源名称，进入资源详情页。确认购买的规格是否正确。

图 2-14 查看资源详情

3 Lite Cluster 资源配置

3.1 Lite Cluster 资源配置流程

本章节介绍Lite Cluster环境配置详细流程，适用于加速卡环境配置。

前提条件

- 已完成集群资源购买和开通，具体请参见**Lite Cluster资源开通**。
- 集群的配置使用需要用户具备一定的知识背景，包括但不限于**Kubernetes基础知识**、网络知识、存储和镜像知识。

配置流程

图 3-1 Lite Cluster 资源配置流程图

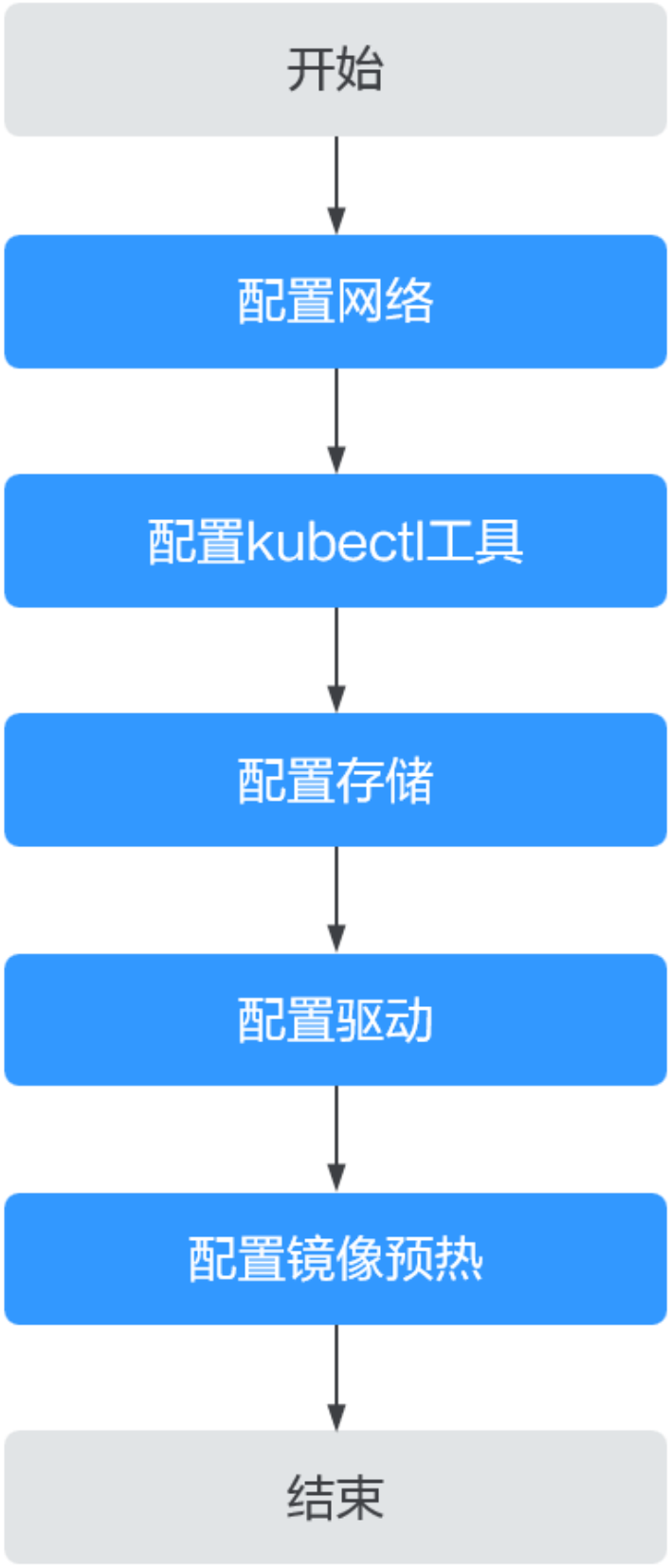


表 3-1 Cluster 资源配置流程

配置顺序	配置任务	场景说明
1	配置Lite Cluster网络	购买资源池后，需要弹性公网IP并进行网络配置，配置网络后可通过公网访问集群资源。
2	配置kubectl工具	kubectl是Kubernetes集群的命令行工具，配置kubectl后，您可通过kubectl命令操作Kubernetes集群。
3	配置Lite Cluster存储	如果没有挂载任何外部存储，此时可用存储空间根据dockerBaseSize的配置来决定，可访问的存储空间比较小，因此建议通过挂载外部存储空间解决存储空间受限问题。容器中挂载存储有多种方式，不同的场景下推荐的存储方式不一样，您可根据业务实际情进行选择。
4	（可选）配置驱动	当专属资源池中的节点含有GPU/Ascend资源时，为确保GPU/Ascend资源能够正常使用，需要配置好对应的驱动。如果在购买资源池时，没配置自定义驱动，默认驱动不满足业务要求，可通过本章节将驱动升级到指定版本。
5	（可选）配置镜像预热	Lite Cluster资源池支持镜像预热功能，镜像预热可实现将镜像提前在资源池节点上拉取好，在推理及大规模分布式训练时有效缩短镜像拉取时间。

快速配置 Lite Cluster 资源案例

下文提供一个快速配置的案例，配置完成后您可登录到节点查看加速卡信息并完成一个训练任务。在运行此案例前，您需要购买资源，购买资源的步骤请参考Lite Cluster资源开通。

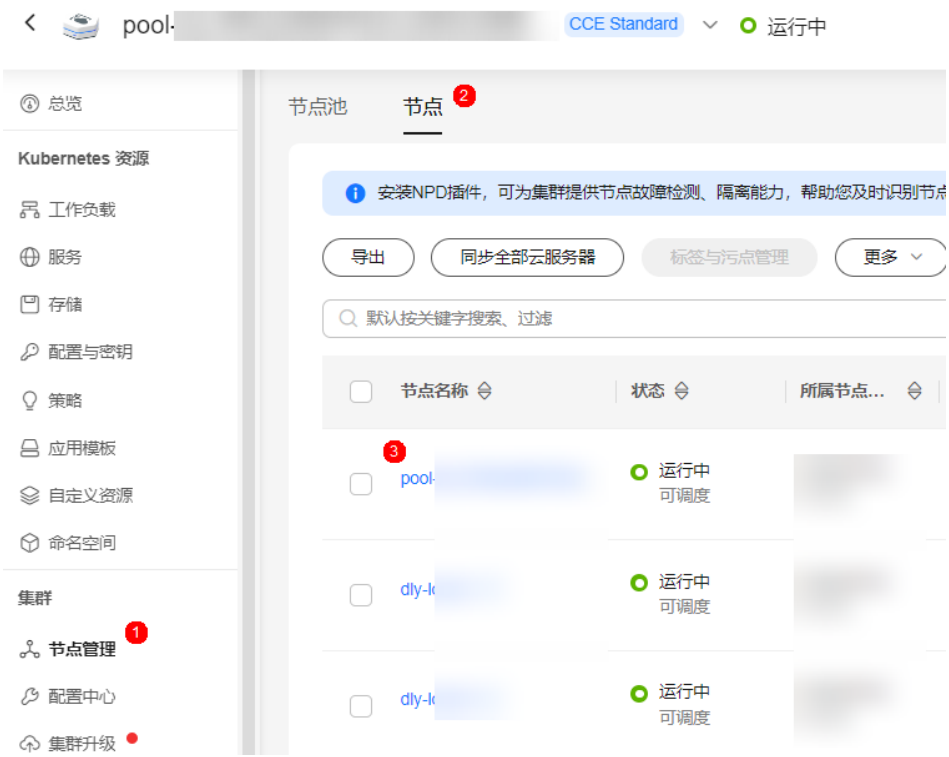
步骤1 登录节点。

（推荐）方式1：通过绑定公网ip的方式

客户可以为需要登录的节点绑定公网ip，然后可以通过Xshell、MobaXterm等bash工具登录节点。

1. 使用华为云账号登录CCE管理控制台。
2. 在CCE集群详情页面，单击“节点管理”页签，在“节点”页签中单击需要登录的节点名称，跳转至弹性云服务器页面。

图 3-2 节点管理



3. 绑定弹性公网IP。

   若已有未绑定的弹性公网IP，直接选择即可。如果没有可用的弹性公网IP，需要先购买弹性公网IP。

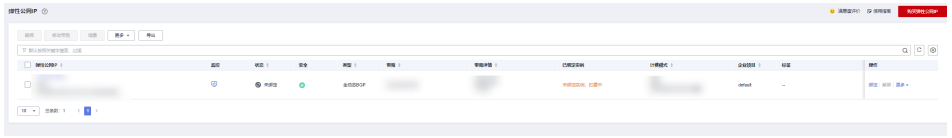
图 3-3 弹性公网 IP

单击“购买弹性公网IP”，进入购买页。

图 3-4 绑定弹性公网 IP

图 3-5 购买弹性公网 IP

图 3-6 未绑定的弹性公网 IP



完成购买后，返回弹性云服务器页面，刷新列表。

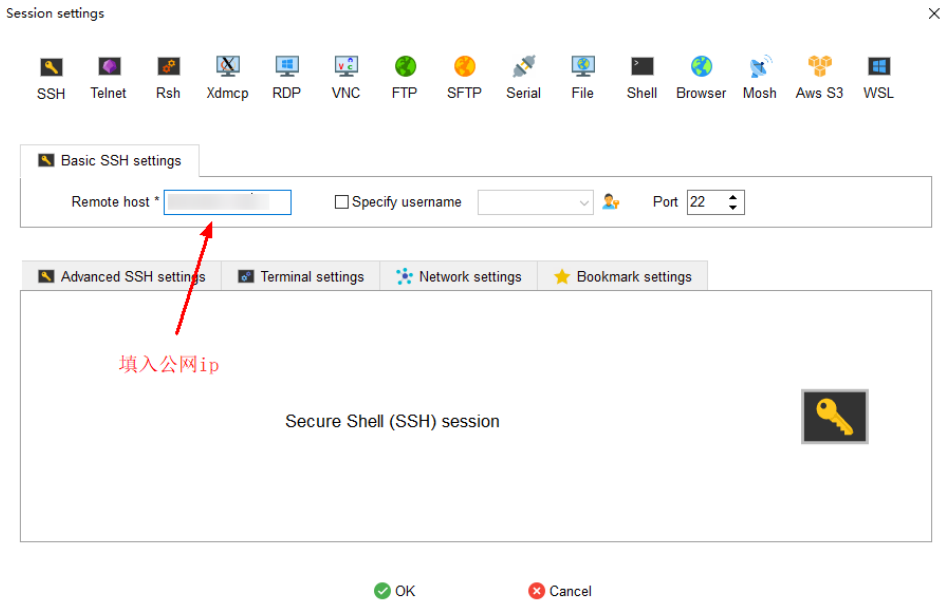
图 3-7 刷新列表

选择刚才创建的弹性公网IP，单击“确定”。

图 3-8 绑定弹性公网 IP

4. 绑定完成后，通过MobaXterm、Xshell登录。以MobaXterm为例，填入弹性公网IP，登录节点。

图 3-9 登录节点



方式2：通过华为云自带的远程登录功能

1. 使用华为云账号登录**CCE管理控制台**。
2. 在CCE集群详情页面，单击“节点管理”页签，在“节点”页签中单击需要登录的节点名称，跳转至弹性云服务器页面。

图 3-10 节点管理

3. 单击“远程登录”，在弹出的窗口中，单击“CloudShell登录”。

图 3-11 远程登录

4. 在CloudShell中设置密码等参数后，单击“连接”即可登录节点，CloudShell介绍可参见**远程登录Linux弹性云服务器（CloudShell方式）**。

步骤2 配置kubectl工具。

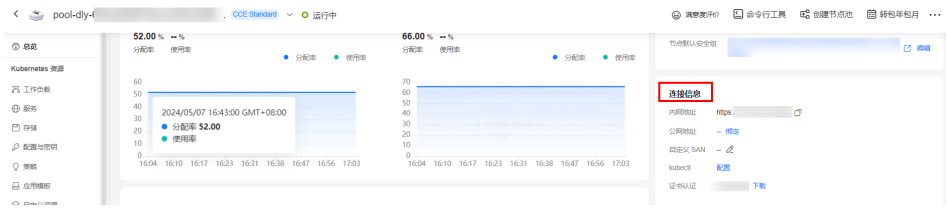
登录ModelArts管理控制台，在左侧菜单栏中选择“AI专属资源池 > 弹性集群 Cluster”，进入“弹性集群 Cluster”页面。

点击创建的专属资源池，进入专属资源池详情页面，点击对应的CCE集群，进入CCE集群详情页面。

图 3-12 专属资源池详情

在CCE集群详情页面中，在“集群信息”找到“连接信息”。

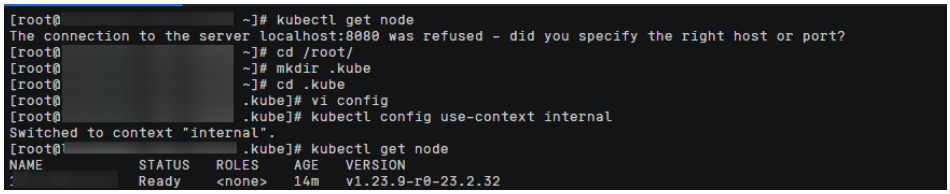
图 3-13 链接信息



使用kubectl工具。

- 若通过内网使用kubectl工具，需要将kubectl工具安装在和集群在相同vpc下的某一台机器上。点击连接信息下kubectl后的“配置”按钮，根据界面提示使用kubectl工具。

  图 3-14 通过内网使用 kubectl 工具

- 通过公网使用kubectl工具，可以将kubectl安装在任一台可以访问公网的机器。首先需要绑定公网地址，点击公网地址后的“绑定”按钮。

  图 3-15 绑定公网地址

选择公网IP后单击“确定”，完成公网IP绑定。如果没有可选的公网IP，单击“创建弹性IP”跳至弹性公网IP页面进行创建。

绑定完成后，点击连接信息下kubectl后的“配置”按钮，根据界面提示使用kubectl工具。

步骤3 docker run方式启动任务。

Snt9B集群在纳管到CCE集群后，都会自动安装Docker，下文仅做测试验证，可以不需要通过创建deployment或者volcano job的方式，直接启动容器进行测试。训练测试用例使用NLP的bert模型，详细代码和指导可参考**Bert**。

1. 拉取镜像。本测试镜像为bert_pretrain_mindspore:v1，已经把测试数据和代码打进镜像中。

```
docker pull swr.cn-southwest-2.myhuaweicloud.com/os-public-repo/bert_pretrain_mindspore:v1
docker tag swr.cn-southwest-2.myhuaweicloud.com/os-public-repo/bert_pretrain_mindspore:v1
bert_pretrain_mindspore:v1
```

2. 启动容器。

```
docker run -tid --privileged=true \
-u 0 \
-v /dev/shm:/dev/shm \
--device=/dev/davinci0 \
--device=/dev/davinci1 \
--device=/dev/davinci2 \
--device=/dev/davinci3 \
--device=/dev/davinci4 \
--device=/dev/davinci5 \
--device=/dev/davinci6 \
--device=/dev/davinci7 \
--device=/dev/davinci_manager \
--device=/dev/devmm_svm \
--device=/dev/hisi_hdc \
-v /usr/local/Ascend/driver:/usr/local/Ascend/driver  \
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi \
-v /etc/hccn.conf:/etc/hccn.conf \
bert_pretrain_mindspore:v1 \
bash
```

参数含义：

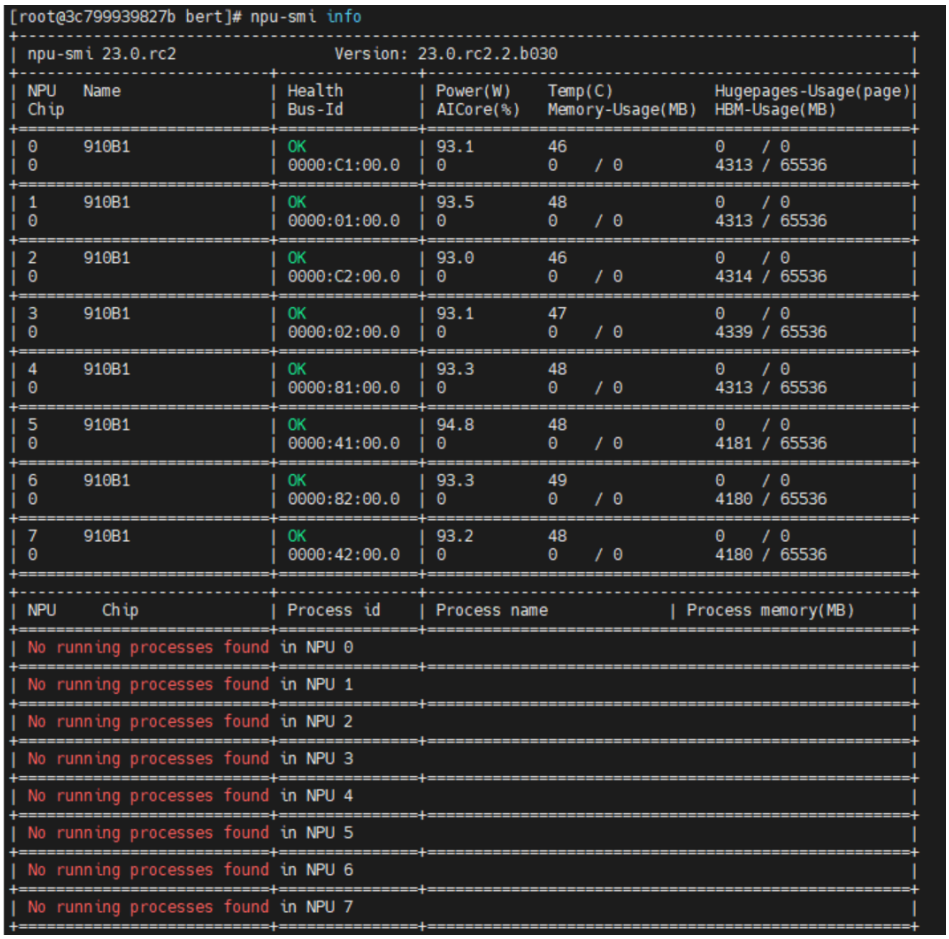
- --privileged=true //特权容器，允许访问连接到主机的所有设备
- -u 0 //root用户
- -v /dev/shm:/dev/shm //防止shm太小训练任务失败
- --device=/dev/davinci0 //npu卡设备
- --device=/dev/davinci1 //npu卡设备
- --device=/dev/davinci2 //npu卡设备
- --device=/dev/davinci3 //npu卡设备
- --device=/dev/davinci4 //npu卡设备
- --device=/dev/davinci5 //npu卡设备
- --device=/dev/davinci6 //npu卡设备
- --device=/dev/davinci7 //npu卡设备
- --device=/dev/davinci_manager //davinci相关的设备管理的设备
- --device=/dev/devmm_svm //管理设备
- --device=/dev/hisi_hdc //管理设备
- -v /usr/local/Ascend/driver:/usr/local/Ascend/driver //npu卡驱动挂载
- -v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi //npu-smi工具挂载

– -v /etc/hccn.conf:/etc/hccn.conf //hccn.conf配置挂载

3. 进入容器，并查看卡信息。

```
docker exec -it xxxxxxx bash    //进入容器，xxxxxxx替换为容器id
npu-smi info    //查看卡信息
```

图 3-16 查看卡信息



4. 执行下述命令启动训练任务。

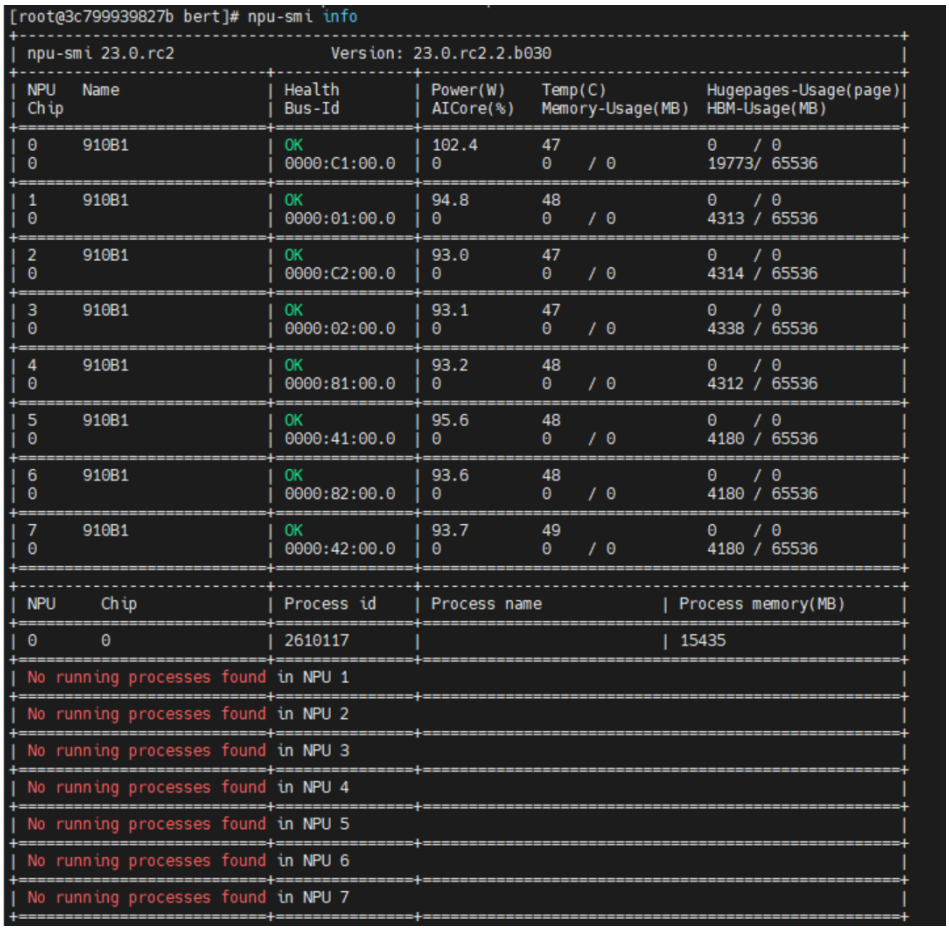
```
cd /home/ma-user/modelarts/user-job-dir/code/bert/
export MS_ENABLE_GE=1
export MS_GE_TRAIN=1
bash scripts/run_standalone_pretrain_ascend.sh 0 1 /home/ma-user/modelarts/user-job-dir/data/cn-news-128-1f-mind/
```

图 3-17 训练进程

查看卡占用情况，如图所示，此时0号卡被占用，说明进程正常启动。

```
npu-smi info    //查看卡信息
```

图 3-18 查看卡信息

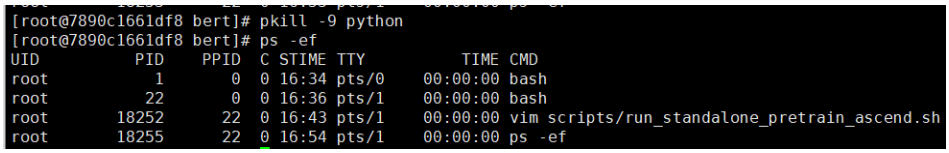


```
[root@3c799939827b bert]# npu-smi info
+------------------------------------------------------------------------------------------------+
| npu-smi 23.0.rc2                 Version: 23.0.rc2.2.b030                                      |
+---------------------------+---------------+----------------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                      | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+====================================================+
| 0     910B1               | OK            | 102.4       47                0    / 0             |
| 0                         | 0000:C1:00.0  | 0           0    / 0          19773/ 65536         |
+===========================+===============+====================================================+
| 1     910B1               | OK            | 94.8        48                0    / 0             |
| 0                         | 0000:01:00.0  | 0           0    / 0          4313 / 65536         |
+===========================+===============+====================================================+
| 2     910B1               | OK            | 93.0        47                0    / 0             |
| 0                         | 0000:C2:00.0  | 0           0    / 0          4314 / 65536         |
+===========================+===============+====================================================+
| 3     910B1               | OK            | 93.1        47                0    / 0             |
| 0                         | 0000:02:00.0  | 0           0    / 0          4338 / 65536         |
+===========================+===============+====================================================+
| 4     910B1               | OK            | 93.2        48                0    / 0             |
| 0                         | 0000:81:00.0  | 0           0    / 0          4312 / 65536         |
+===========================+===============+====================================================+
| 5     910B1               | OK            | 95.6        48                0    / 0             |
| 0                         | 0000:41:00.0  | 0           0    / 0          4180 / 65536         |
+===========================+===============+====================================================+
| 6     910B1               | OK            | 93.6        48                0    / 0             |
| 0                         | 0000:82:00.0  | 0           0    / 0          4180 / 65536         |
+===========================+===============+====================================================+
| 7     910B1               | OK            | 93.7        49                0    / 0             |
| 0                         | 0000:42:00.0  | 0           0    / 0          4180 / 65536         |
+===========================+===============+====================================================+
+---------------------------+---------------+----------------------------------------------------+
| NPU     Chip              | Process id    | Process name             | Process memory(MB)      |
+===========================+===============+====================================================+
| 0       0                 | 2610117       |                          | 15435                   |
+===========================+===============+====================================================+
| No running processes found in NPU 1                                                            |
+===========================+===============+====================================================+
| No running processes found in NPU 2                                                            |
+===========================+===============+====================================================+
| No running processes found in NPU 3                                                            |
+===========================+===============+====================================================+
| No running processes found in NPU 4                                                            |
+===========================+===============+====================================================+
| No running processes found in NPU 5                                                            |
+===========================+===============+====================================================+
| No running processes found in NPU 6                                                            |
+===========================+===============+====================================================+
| No running processes found in NPU 7                                                            |
+===========================+===============+====================================================+
```

训练任务大概会运行两小时左右，训练完成后自动停止。若想停止训练任务，可执行下述命令关闭进程，查询进程后显示已无运行中python进程。

```
pkill -9 python
ps -ef
```

图 3-19 关闭训练进程



```
[root@7890c1661df8 bert]# pkill -9 python
[root@7890c1661df8 bert]# ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 16:34 pts/0    00:00:00 bash
root          22       0  0 16:36 pts/1    00:00:00 bash
root       18252      22  0 16:43 pts/1    00:00:00 vim scripts/run_standalone_pretrain_ascend.sh
root       18255      22  0 16:54 pts/1    00:00:00 ps -ef
```

----结束

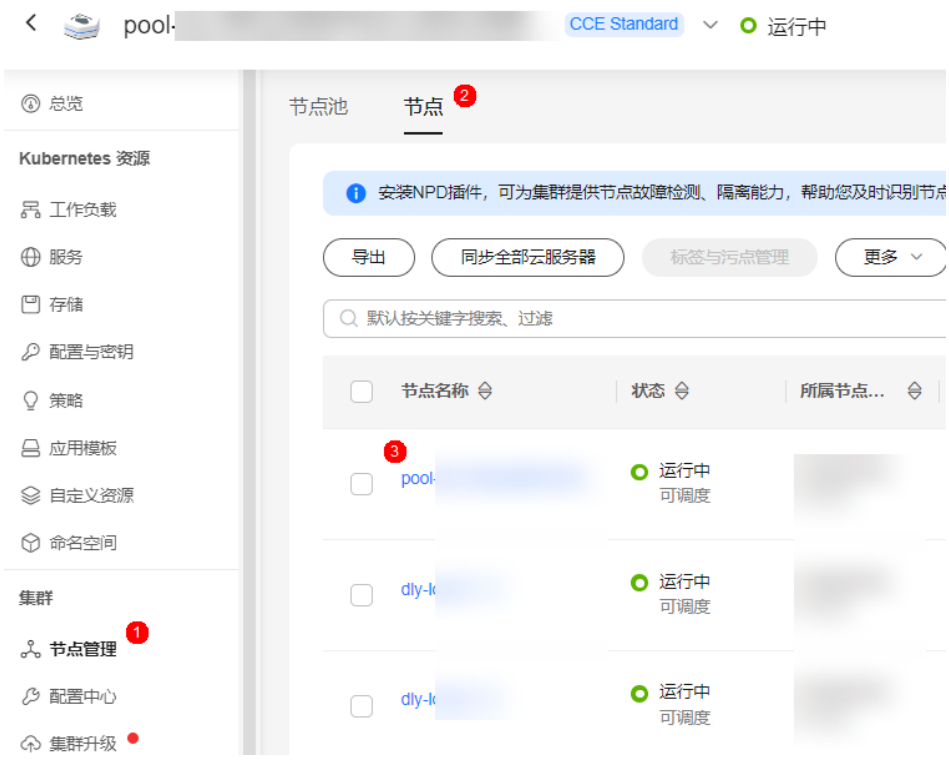
3.2 配置 Lite Cluster 网络

本章节介绍如何申请弹性公网IP并绑定到弹性云服务器。通过本文档，您可以实现弹性云服务器访问公网的目的。

步骤1 使用华为云账号登录**CCE管理控制台**。

步骤2 找到购买Cluster资源时选择的CCE集群，单击名称进入CCE集群详情页面，单击“节点管理”页签，在“节点”页签中单击需要登录的节点名称，跳转至弹性云服务器页面。

图 3-20 节点管理



步骤3 绑定弹性公网IP。

若已有未绑定的弹性公网IP，直接选择即可。如果没有可用的弹性公网IP，需要先购买弹性公网IP，具体操作请参见**申请弹性公网IP**。

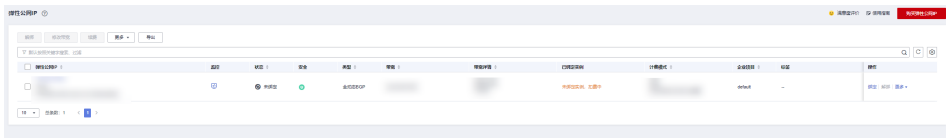
图 3-21 弹性公网 IP

单击“购买弹性公网IP”，进入购买页。

图 3-22 绑定弹性公网 IP

图 3-23 购买弹性公网 IP

图 3-24 未绑定的弹性公网 IP



完成购买后，返回弹性云服务器页面，刷新列表。

图 3-25 刷新列表

选择刚才创建的弹性公网IP，单击“确定”。

图 3-26 绑定弹性公网 IP

步骤4 通过SSH方式远程访问集群资源包括2种方式，密码方式或密钥方式，二选一即可。

- 通过SSH密钥方式登录云服务器，具体操作请参见**SSH密钥登录方式**。
- 通过SSH密码方式登录云服务器，具体操作请参见**SSH密码登录方式**。

----结束

3.3 配置 kubectl 工具

kubectl是Kubernetes集群的命令行工具，配置kubectl后，您可通过kubectl命令操作Kubernetes集群。本文介绍如何配置kubectl工具，操作步骤如下。

步骤1 进入专属资源池。

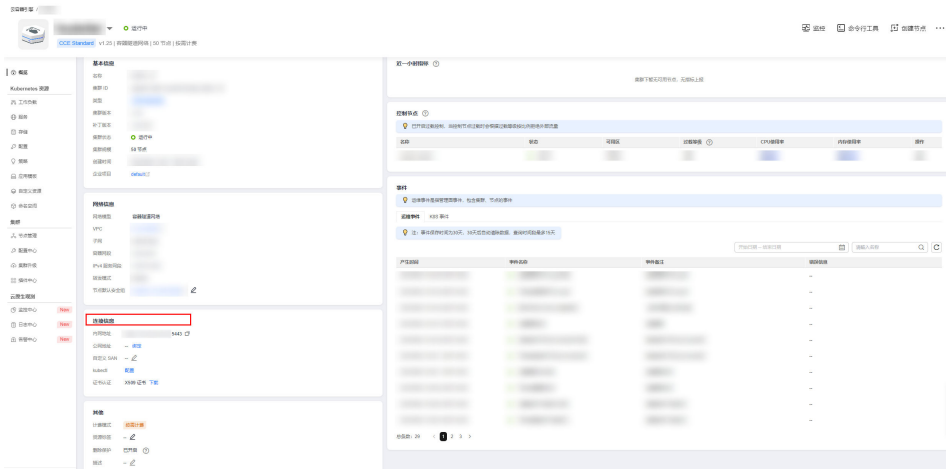
图 3-27 专属资源池页签

步骤2 单击创建的专属资源池，进入专属资源池详情页面。

图 3-28 专属资源池详情

步骤3 单击对应的CCE集群，进入CCE集群详情页面，在“集群信息”找到“连接信息”。

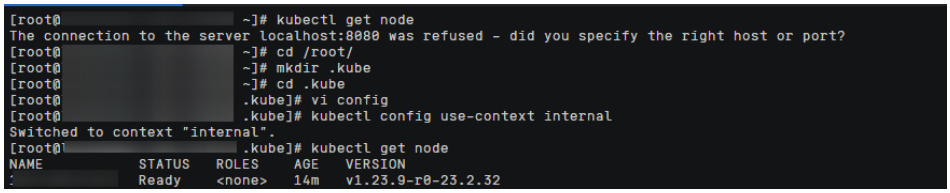
图 3-29 链接信息



步骤4 使用kubectl工具。

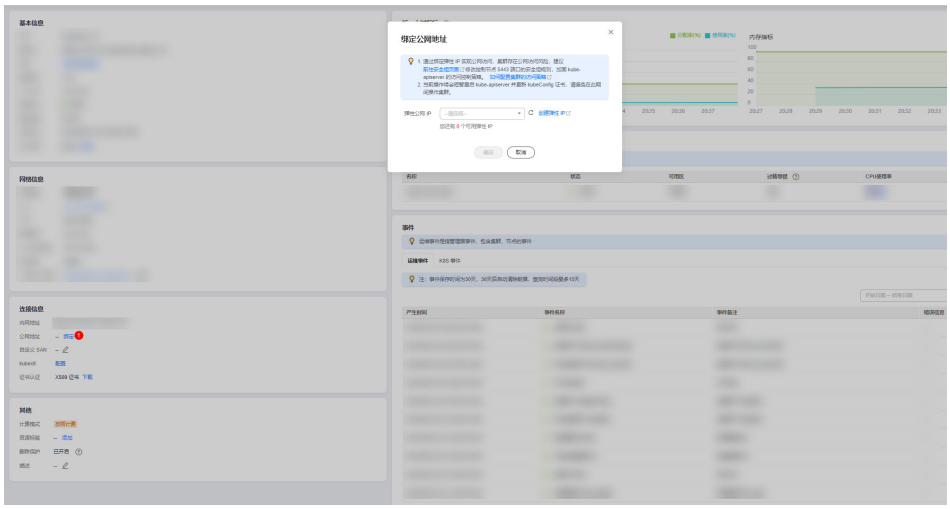
- 若通过内网使用kubectl工具，需要将kubectl工具安装在和集群在相同vpc下的某一台机器上。单击kubectl后的“配置”按钮。按照界面提示步骤操作即可。

  图 3-30 通过内网使用 kubectl 工具

- 通过公网使用kubectl工具，可以将kubectl安装在任一台可以访问公网的机器。首先需要绑定公网地址，单击公网地址后的“绑定”按钮。

图 3-31 绑定公网地址

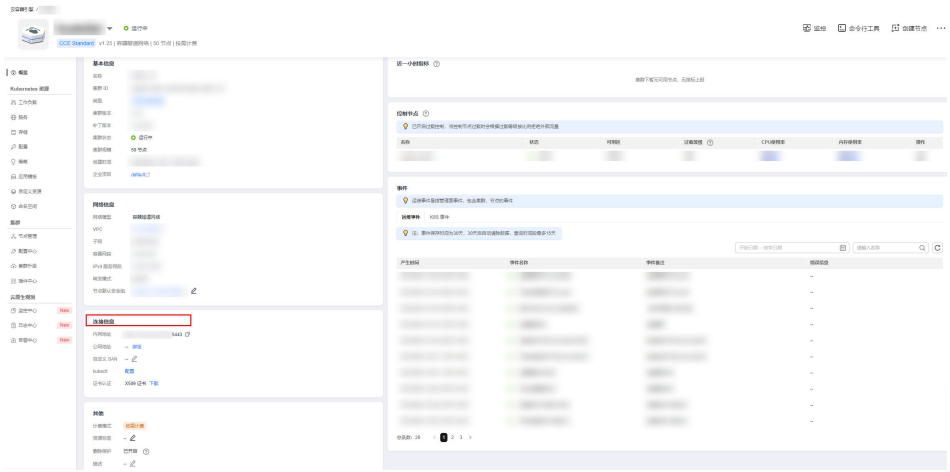


选择已有的公网IP，或者跳至创建，创建新的弹性公网IP。

完成公网地址绑定后，在“集群信息”找到“连接信息”，单击kubectl后的“配置”按钮。

按照界面提示步骤操作即可。

图 3-32 配置 kubectl



步骤5 验证。

在安装了kubectl工具的机器上执行如下命令，显示集群节点即为成功。

```
kubectl get node
```

----结束

3.4 配置 Lite Cluster 存储

如果没有挂载任何外部存储，此时可用存储空间根据dockerBaseSize的配置来决定，可访问的存储空间比较小，因此建议通过挂载外部存储空间解决存储空间受限问题。

容器中挂载存储有多种方式，不同的场景下推荐的存储方式不一样，详情如**表3-2**所示。容器存储的基础知识了解请参见**存储基础知识**，有助您理解本章节内容。您可查

看**数据盘空间分配说明**，了解节点数据盘空间分配的情况，以便您根据业务实际情况配置数据盘大小。

表 3-2 容器挂载存储的方式及差异

容器挂载存储的方式	使用场景	特点	挂载操作参考
EmptyDir	适用于训练缓存场景。	Kubernetes的临时存储卷，临时卷会遵从Pod的生命周期，与Pod一起创建和删除。	**使用临时存储路径**
HostPath	适用于以下场景： 1. 容器工作负载程序生成的日志文件需要永久保存。 2. 需要访问宿主机上Docker引擎内部数据结构的容器工作负载。	节点存储。多个容器可能会共享这一个存储，会存在写冲突的问题。 Pod删除后，存储不会清理。	**使用主机路径**
OBS	适用于训练数据集的存储。	对象存储。常用OBS SDK进行样本数据下载。存储量大，但是离节点比较远，直接训练速度会比较慢，通常会先将数据拉取到本地cache，然后再进行训练任务。	• **静态挂载** • **动态挂载**
SFS Turbo	适用于海量小文件业务场景。	• 提供posix协议的文件系统； • 需要和资源池在同一个VPC下或VPC互通； • 价格较高。	• **静态挂载** • 动态挂载：不支持
SFS	适用于多读多写场景的持久化存储。	适用大容量扩展以及成本敏感型的业务场景，包括媒体处理、内容管理、大数据分析和分析工作负载程序等。 SFS容量型文件系统不适合海量小文件业务。	• **静态挂载** • **动态挂载**
EVS	适用于Notebook场景，开发过程的数据持久化。	每个云盘只能在单个节点挂载。 存储大小根据云硬盘的大小而定。	• **静态挂载** • **动态挂载**

3.5 （可选）配置驱动

当专属资源池中的节点含有GPU/Ascend资源时，为确保GPU/Ascend资源能够正常使用，需要配置好对应的驱动。

Cluster支持两种配置驱动的方式：

- **方式一：购买资源池时通过自定义驱动参数进行配置**
- **方式二：通过驱动升级功能对已有的资源池驱动版本进行升级**

方式一：购买资源池时通过自定义驱动参数进行配置

在购买资源池页面，部分GPU和Ascend规格资源池允许自定义安装驱动。开启自定义驱动开关并选择需要的驱动版本即可。

图 3-33 自定义驱动

方式二：通过驱动升级功能对已有的资源池驱动版本进行升级

如果在购买资源池时，没配置自定义驱动，默认驱动不满足业务要求，可通过驱动升级功能将驱动升级到指定版本。驱动升级功能介绍可参考**升级Lite Cluster资源池驱动**。

图 3-34 驱动升级

3.6 （可选）配置镜像预热

Lite Cluster资源池支持镜像预热功能，镜像预热可实现将镜像提前在资源池节点上拉取好，在推理及大规模分布式训练时有效缩短镜像拉取时间。本文将介绍如何配置镜像预热功能。

操作步骤

步骤1 单击资源池名称，进入资源池详情。

步骤2 单击左侧“配置管理”。

图 3-35 配置管理

步骤3 在镜像预热中单击编辑图标，填写镜像预热信息。

表 3-3 镜像预热参数

参数名称	说明
镜像来源	可选择“预置”或“自定义”的镜像。 ● 预置：可选择SWR服务上自有的或他人共享的镜像。 ● 自定义：可直接填写镜像地址。
添加镜像密钥	若本租户不具有预热的镜像的权限（即非公开/非本租户私有/非他人共享的镜像），此时需要添加镜像密钥。在开启镜像密钥开关后，选择命名空间及对应密钥。创建密钥方法可参考**创建密钥**，密钥类型须为kubernetes.io/dockerconfigjson类型。 若需添加多个密钥，可以单击“+”新增密钥数。
添加镜像预热配置	若需添加多个镜像，可单击此按键。

图 3-36 预置镜像预热

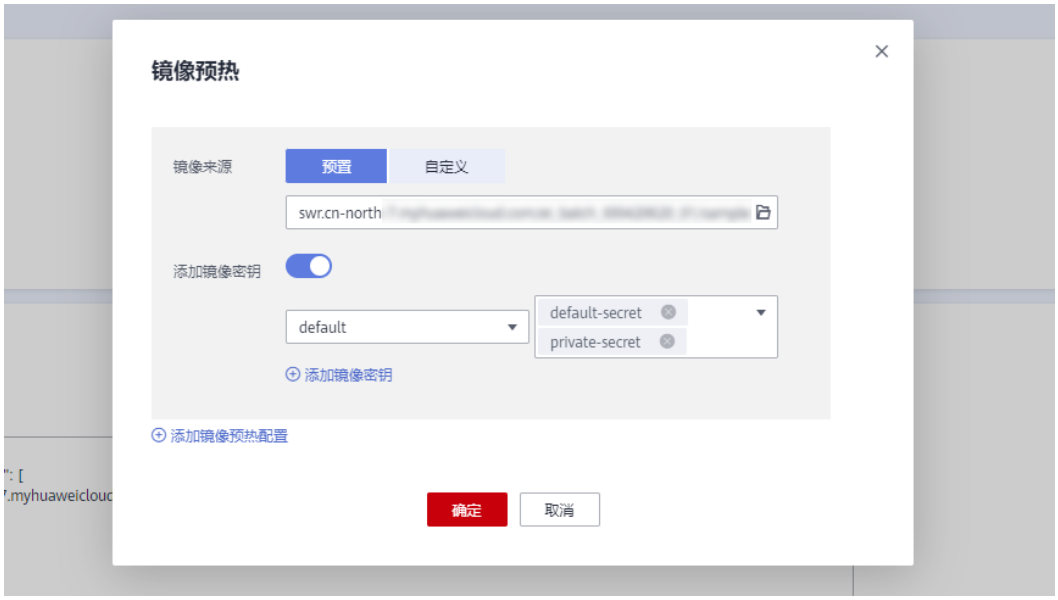


图 3-37 预置镜像选择

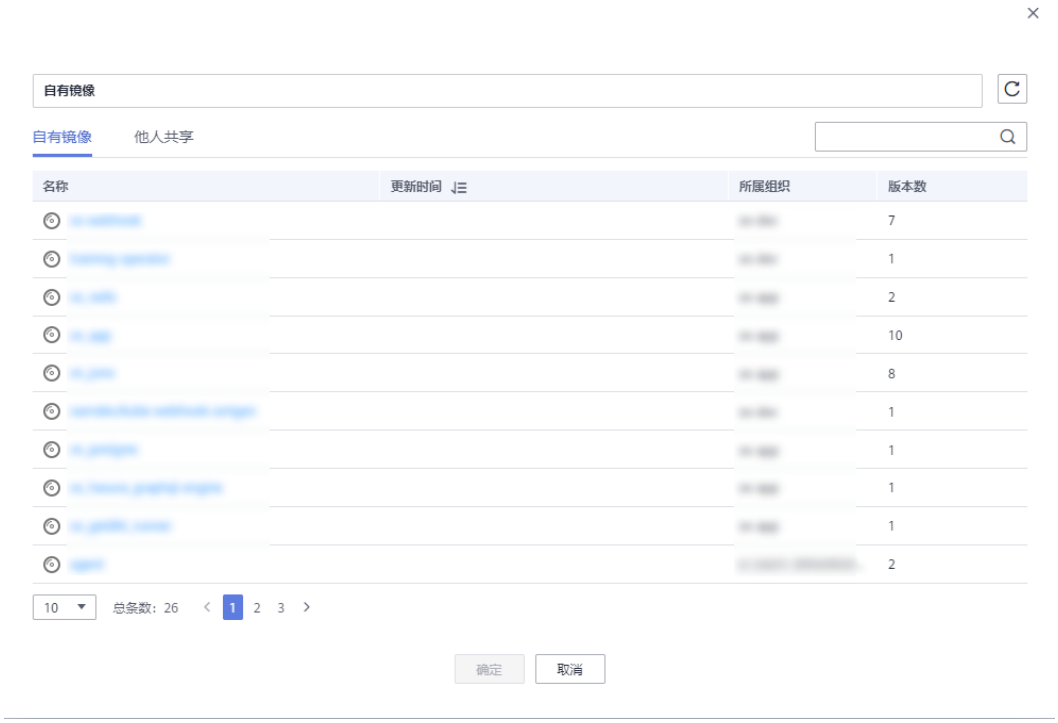


图 3-38 自定义镜像预热

创建密钥所需的仓库地址、用户名、密码、可以参考对应租户的SWR登录指令。

图 3-39 创建密钥

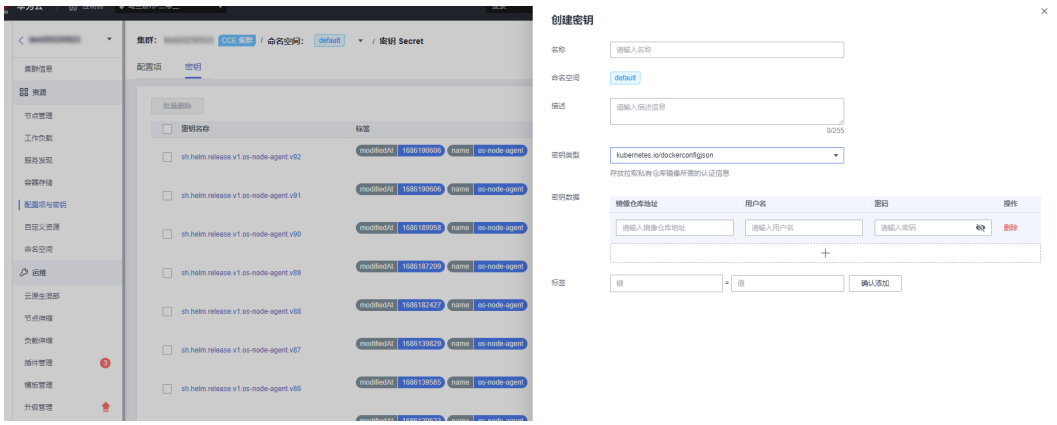
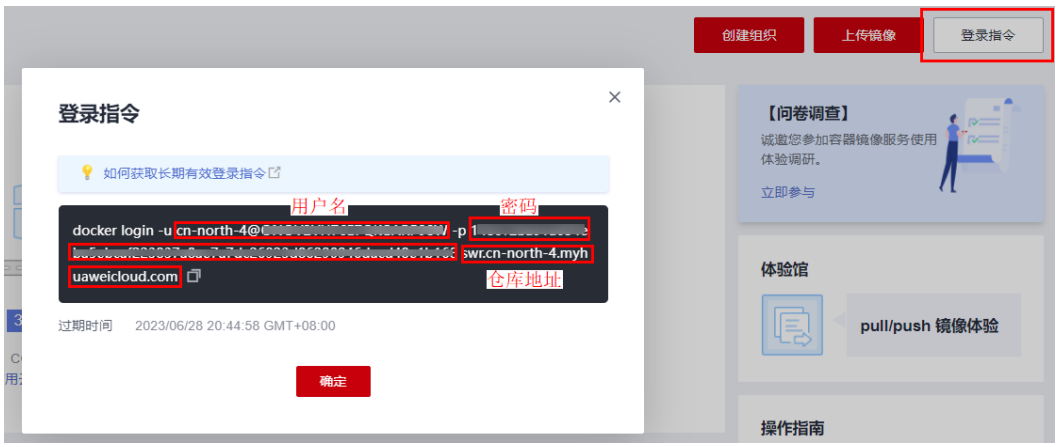


图 3-40 登录指令

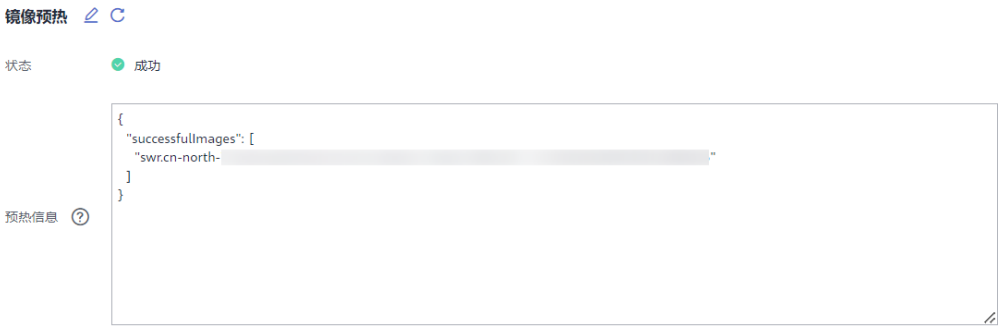


说明

上图中为临时登录指令，若需长期有效登录指令，可单击图中的“如何获取长期有效指令”链接获取指导。

步骤4 单击“确认”后，在预热信息框中可以看到已成功预热的镜像信息。

图 3-41 镜像预热成功



说明

若镜像预热失败，请检查镜像地址以及密钥是否正确。

----结束

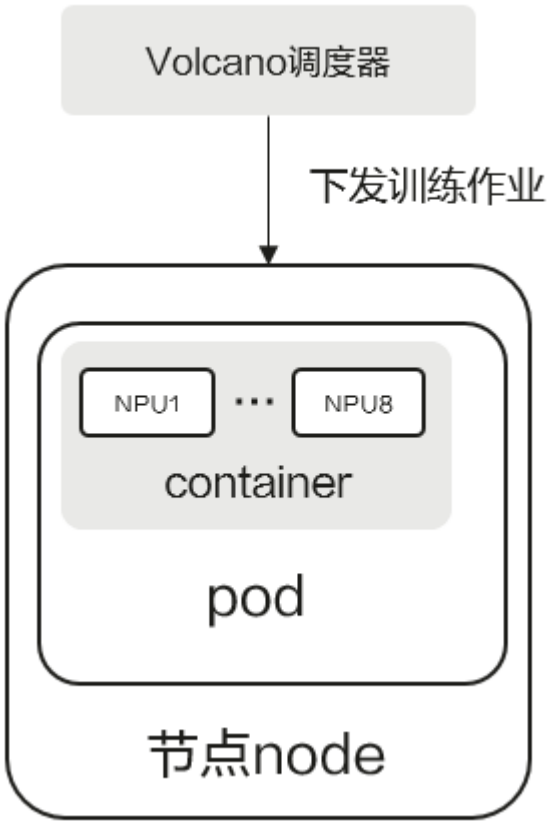
4 Lite Cluster 资源使用

4.1 在 Lite Cluster 资源池上使用 Snt9B 完成分布式训练任务

场景描述

本案例介绍如何在Snt9B上进行分布式训练任务，其中Cluster资源池已经默认安装volcano调度器，训练任务默认使用volcano job形式下发lite池集群。训练测试用例使用NLP的bert模型，详细代码和指导可参考**Bert**。

图 4-1 任务示意图



操作步骤

步骤1 拉取镜像。本测试镜像为bert_pretrain_mindspore:v1，已经把测试数据和代码打进镜像中。

```
docker pull swr.cn-southwest-2.myhuaweicloud.com/os-public-repo/bert_pretrain_mindspore:v1
docker tag swr.cn-southwest-2.myhuaweicloud.com/os-public-repo/bert_pretrain_mindspore:v1
bert_pretrain_mindspore:v1
```

步骤2 在主机上新建config.yaml文件。

config.yaml文件用于配置pod，本示例中使用sleep命令启动pod，便于进入pod调试。您也可以修改command为对应的任务启动命令（如“python train.py”），任务会在启动容器后执行。

config.yaml内容如下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap1980-yourvcjobname     # 前缀使用“configmap1980-”不变，后接vcjob的名字
  namespace: default                    # 命名空间自选，需要和下边的vcjob处在同一命名空间
  labels:
    ring-controller.cce: ascend-1980   # 保持不动
data:                    #data内容保持不动，初始化完成，会被volcano插件自动修改
  jobstart_hccl.json: |
    {
        "status":"initializing"
    }
---
apiVersion: batch.volcano.sh/v1alpha1   # The value cannot be changed. The volcano API must be used.
```



```
kind: Job                           # Only the job type is supported at present.
metadata:
  name: yourvcjobname               # job名字，需要和configmap中名字保持联系
  namespace: default                # 和configmap保持一致
  labels:
    ring-controller.cce: ascend-1980   # 保持不动
    fault-scheduling: "force"
spec:
  minAvailable: 1                   # The value of minAvailable is 1 in a single-node scenario and N in an N-node distributed scenario.
  schedulerName: volcano            # 保持不动，Use the Volcano scheduler to schedule jobs.
  policies:
    - event: PodEvicted
      action: RestartJob
  plugins:
    configmap1980:
    - --rank-table-version=v2  # 保持不动，生成v2版本ranktablefile
    env: []
    svc:
    - --publish-not-ready-addresses=true
  maxRetry: 3
  queue: default
  tasks:
  - name: "yourvcjobname-1"
    replicas: 1                     # The value of replicas is 1 in a single-node scenario and N in an N-node scenario. The number of NPUs in the requests field is 8 in an N-node scenario.
    template:
      metadata:
        labels:
          app: mindspore
          ring-controller.cce: ascend-1980  # 保持不动，The value must be the same as the label in ConfigMap and cannot be changed.
      spec:
        affinity:
          podAntiAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
              - labelSelector:
                  matchExpressions:
                    - key: volcano.sh/job-name
                      operator: In
                      values:
                        - yourvcjobname
                topologyKey: kubernetes.io/hostname
        containers:
        - image: bert_pretrain_mindspore:v1             # 镜像地址，Training framework image, which can be modified.
          imagePullPolicy: IfNotPresent
          name: mindspore
          env:
          - name: name                          # The value must be the same as that of Jobname.
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: ip                            # IP address of the physical node, which is used to identify the node where the pod is running
            valueFrom:
              fieldRef:
                fieldPath: status.hostIP
          - name: framework
            value: "MindSpore"
          command:
          - "sleep"
          - "1000000000000000000"
          resources:
            requests:
              huawei.com/ascend-1980: "1"               # 需求卡数，key保持不变。Number of required NPUs. The maximum value is 16. You can add lines below to configure resources such as memory and CPU.
            limits:
              huawei.com/ascend-1980: "1"               # 限制卡数，key保持不变。The value must be consistent
```



```
with that in requests.
        volumeMounts:
        - name: ascend-driver              #驱动挂载，保持不动
          mountPath: /usr/local/Ascend/driver
        - name: ascend-add-ons             #驱动挂载，保持不动
          mountPath: /usr/local/Ascend/add-ons
        - name: localtime
          mountPath: /etc/localtime
        - name: hccn                       #驱动hccn配置，保持不动
          mountPath: /etc/hccn.conf
        - name: npu-smi                    #npu-smi
          mountPath: /usr/local/bin/npu-smi
      nodeSelector:
        accelerator/huawei-npu: ascend-1980
      volumes:
      - name: ascend-driver
        hostPath:
          path: /usr/local/Ascend/driver
      - name: ascend-add-ons
        hostPath:
          path: /usr/local/Ascend/add-ons
      - name: localtime
        hostPath:
          path: /etc/localtime             # Configure the Docker time.
      - name: hccn
        hostPath:
          path: /etc/hccn.conf
      - name: npu-smi
        hostPath:
          path: /usr/local/bin/npu-smi
      restartPolicy: OnFailure
```

步骤3 根据config.yaml创建pod。

```
kubectl apply -f config.yaml
```

步骤4 检查pod启动情况，执行下述命令。如果显示“1/1 running”状态代表启动成功。

```
kubectl get pod -A
```

步骤5 进入容器，{pod_name}替换为您的pod名字（get pod中显示的名字），{namespace}替换为您的命名空间（默认为default）。

```
kubectl exec -it {pod_name} bash -n {namespace}
```

步骤6 查看卡信息，执行以下命令。

```
npu-smi info
```

kubernetes会根据config.yaml文件中配置的卡数分配资源给pod，如下图所示由于配置了1卡因此在容器中只会显示1卡，说明配置生效。

图 4-2 查看卡信息

```
[root@louleilei-louleilei-1-0 ma-user]# npu-smi info
+------------------------------------------------------------------------------------------------+
| npu-smi 23.0.rc2                 Version: 23.0.rc2.2.b030                                      |
+---------------------------+---------------+----------------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                      | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+====================================================+
| 0     910B1               | OK            | 93.1        48                0    / 0             |
| 0                         | 0000:C1:00.0  | 0           0    / 0          4313 / 65536         |
+===========================+===============+====================================================+
+---------------------------+---------------+----------------------------------------------------+
| NPU     Chip              | Process id    | Process name             | Process memory(MB)      |
+===========================+===============+====================================================+
| No running processes found in NPU 0                                                            |
+===========================+===============+====================================================+
```

步骤7 修改pod的卡数。由于本案例中为分布式训练，因此所需卡数修改为8卡。

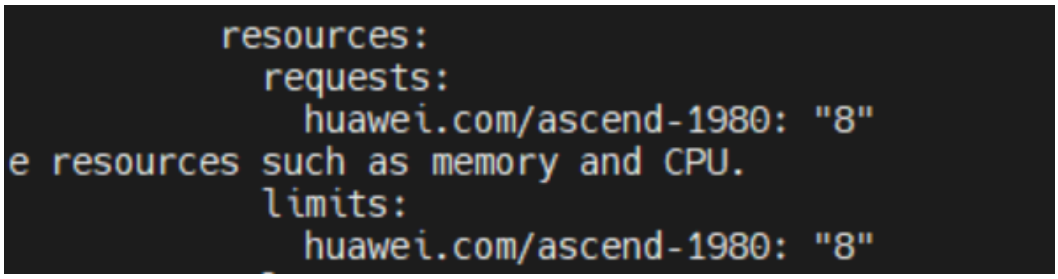
删除已创建的pod。

```
kubectl delete -f config.yaml
```


将config.yaml文件中“limit”和“request”改为8。

```
vi config.yaml
```

图 4-3 修改卡数



重新创建pod。

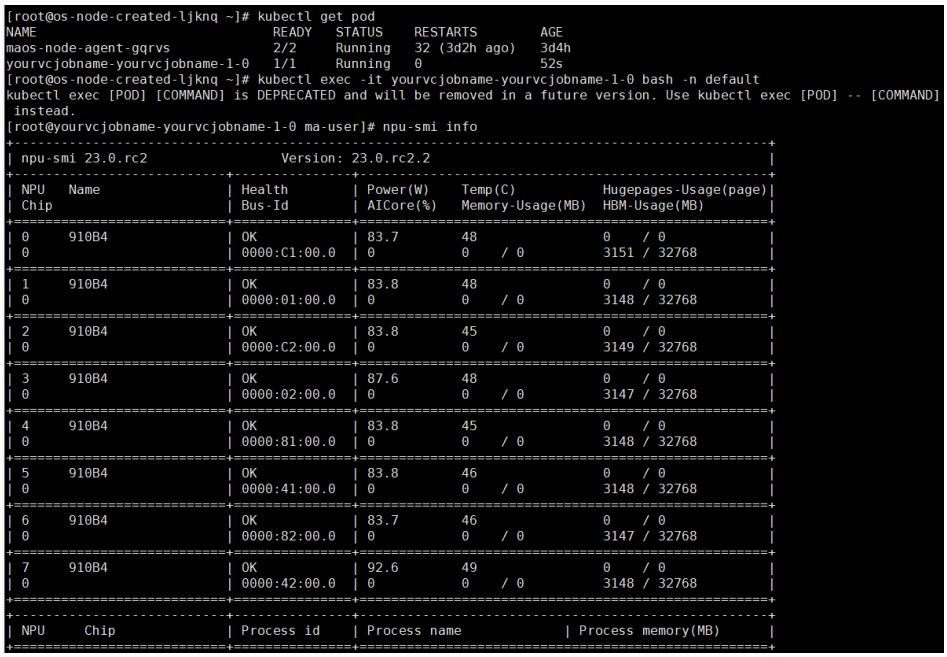
```
kubectl apply -f config.yaml
```

进入容器并查看卡信息，{pod_name}替换为您的pod名字，{namespace}替换为您的命名空间（默认为default）。

```
kubectl exec -it {pod_name} bash -n {namespace}
npu-smi info
```

如图所示为8卡，pod配置成功。

图 4-4 查看卡信息

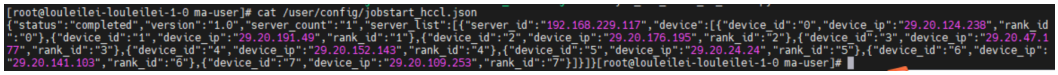


步骤8 查看卡间通信配置文件，执行以下命令。

```
cat /user/config/jobstart_hccl.json
```

多卡训练时，需要依赖“rank_table_file”做卡间通信的配置文件，该文件自动生成，pod启动之后文件地址。为“/user/config/jobstart_hccl.json”，“/user/config/jobstart_hccl.json”配置文件生成需要一段时间，业务进程需要等待“/user/config/jobstart_hccl.json”中“status”字段为“completed”状态，才能生成卡间通信信息。如下图所示。

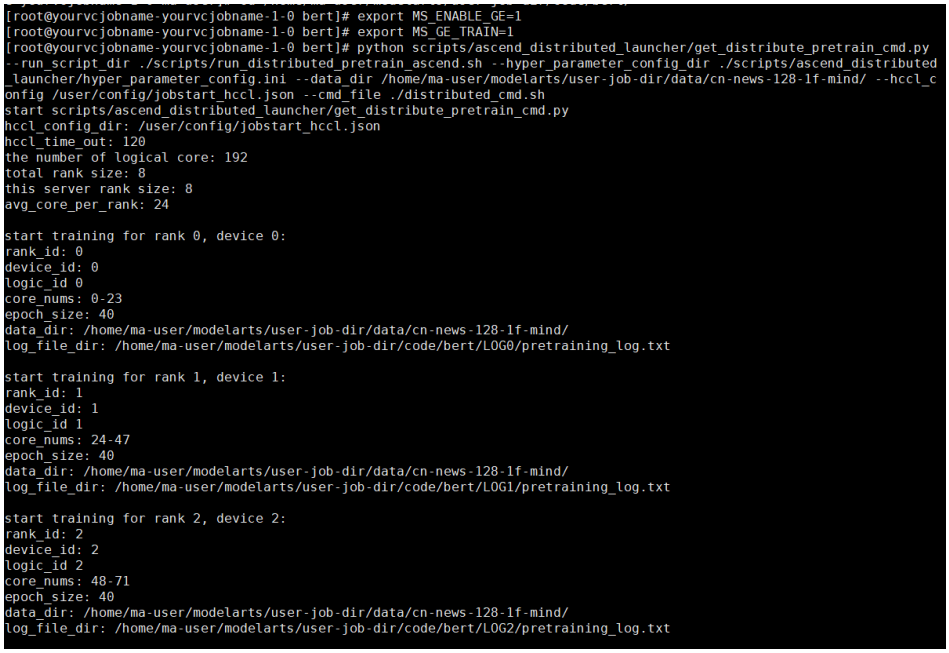
图 4-5 卡间通信配置文件



步骤9 启动训练任务。

```
cd /home/ma-user/modelarts/user-job-dir/code/bert/
export MS_ENABLE_GE=1
export MS_GE_TRAIN=1
python scripts/ascend_distributed_launcher/get_distribute_pretrain_cmd.py --run_script_dir ./scripts/
run_distributed_pretrain_ascend.sh --hyper_parameter_config_dir ./scripts/ascend_distributed_launcher/
hyper_parameter_config.ini --data_dir /home/ma-user/modelarts/user-job-dir/data/cn-news-128-1f-mind/ --
hccl_config /user/config/jobstart_hccl.json --cmd_file ./distributed_cmd.sh
bash scripts/run_distributed_pretrain_ascend.sh /home/ma-user/modelarts/user-job-dir/data/cn-
news-128-1f-mind/ /user/config/jobstart_hccl.json
```

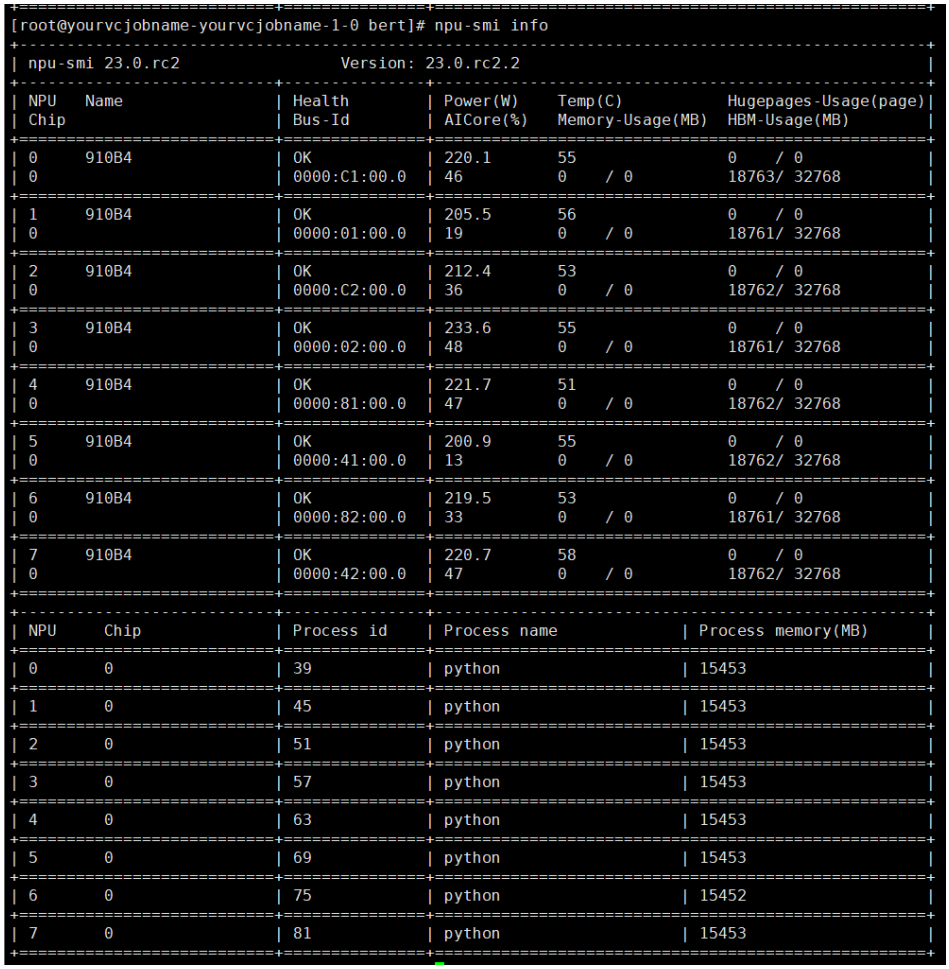
图 4-6 启动训练任务



训练任务加载需要一定时间，在等待若干分钟后，可以执行下述命令查看卡信息。如下图可见，8张卡均被占用，说明训练任务在进行中

```
npu-smi info
```

图 4-7 查看卡信息

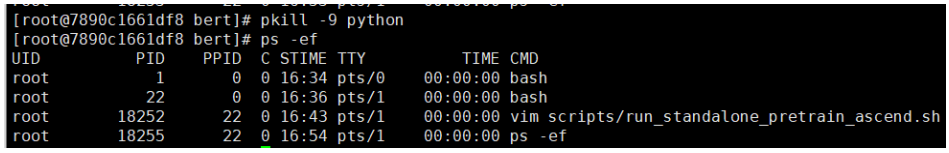


```
[root@yourvcjobname-yourvcjobname-1-0 bert]# npu-smi info
+------------------------------------------------------------------------------------------------+
| npu-smi 23.0.rc2                 Version: 23.0.rc2.2                                           |
+---------------------------+---------------+----------------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                      | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+====================================================+
| 0     910B4               | OK            | 220.1       55                0    / 0             |
| 0                         | 0000:C1:00.0  | 46          0    / 0          18763/ 32768         |
+===========================+===============+====================================================+
| 1     910B4               | OK            | 205.5       56                0    / 0             |
| 0                         | 0000:01:00.0  | 19          0    / 0          18761/ 32768         |
+===========================+===============+====================================================+
| 2     910B4               | OK            | 212.4       53                0    / 0             |
| 0                         | 0000:C2:00.0  | 36          0    / 0          18762/ 32768         |
+===========================+===============+====================================================+
| 3     910B4               | OK            | 233.6       55                0    / 0             |
| 0                         | 0000:02:00.0  | 48          0    / 0          18761/ 32768         |
+===========================+===============+====================================================+
| 4     910B4               | OK            | 221.7       51                0    / 0             |
| 0                         | 0000:81:00.0  | 47          0    / 0          18762/ 32768         |
+===========================+===============+====================================================+
| 5     910B4               | OK            | 200.9       55                0    / 0             |
| 0                         | 0000:41:00.0  | 13          0    / 0          18762/ 32768         |
+===========================+===============+====================================================+
| 6     910B4               | OK            | 219.5       53                0    / 0             |
| 0                         | 0000:82:00.0  | 33          0    / 0          18761/ 32768         |
+===========================+===============+====================================================+
| 7     910B4               | OK            | 220.7       58                0    / 0             |
| 0                         | 0000:42:00.0  | 47          0    / 0          18762/ 32768         |
+===========================+===============+====================================================+
+---------------------------+---------------+----------------------------------------------------+
| NPU     Chip              | Process id    | Process name             | Process memory(MB)      |
+===========================+===============+==========================+=========================+
| 0       0                 | 39            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
| 1       0                 | 45            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
| 2       0                 | 51            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
| 3       0                 | 57            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
| 4       0                 | 63            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
| 5       0                 | 69            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
| 6       0                 | 75            | python                   | 15452                   |
+===========================+===============+==========================+=========================+
| 7       0                 | 81            | python                   | 15453                   |
+===========================+===============+==========================+=========================+
```

若想停止训练任务，可执行下述命令关闭进程，查询进程后显示已无运行中python进程。

```
pkill -9 python
ps -ef
```

图 4-8 关闭训练进程



```
[root@7890c1661df8 bert]# pkill -9 python
[root@7890c1661df8 bert]# ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 16:34 pts/0    00:00:00 bash
root          22       0  0 16:36 pts/1    00:00:00 bash
root       18252      22  0 16:43 pts/1    00:00:00 vim scripts/run_standalone_pretrain_ascend.sh
root       18255      22  0 16:54 pts/1    00:00:00 ps -ef
```

说明

limit/request配置cpu和内存大小，已知单节点Snt9B机器为：8张Snt9B卡+192u1536g，请合理规划，避免cpu和内存限制过小引起任务无法正常运行。

----结束

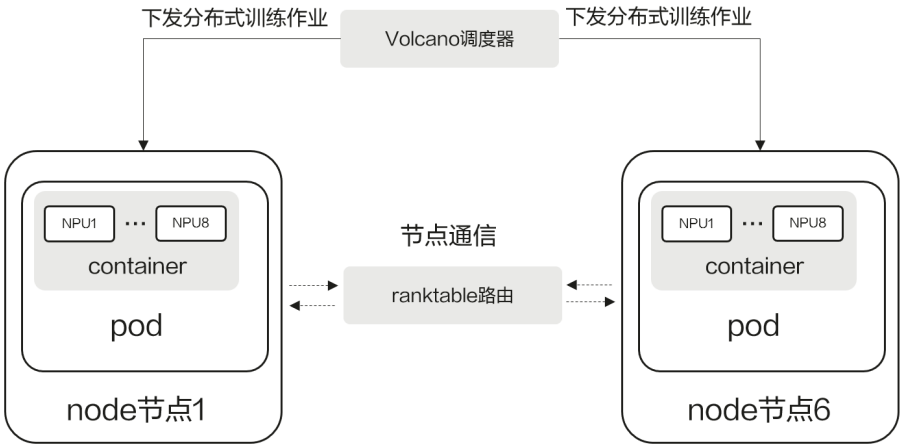
4.2 在 Lite Cluster 资源池上使用 ranktable 路由规划完成 Pytorch NPU 分布式训练

场景描述

ranktable路由规划是一种用于分布式并行训练中的通信优化能力，在使用NPU的场景下，支持对节点之间的通信路径根据交换机实际topo做网络路由亲和规划，进而提升节点之间的通信速度。

本案例介绍如何在ModelArts Lite场景下使用ranktable路由规划完成Pytorch NPU分布式训练任务，训练任务默认使用Volcano job形式下发到Lite资源池集群。

图 4-9 任务示意图



约束与限制

- 该功能只支持贵阳一区域，如果要在其他区域使用请联系技术支持。
- ModelArts Lite资源池对应的CCE集群需要安装1.10.12及以上版本的华为云版Volcano插件。Volcano调度器的安装升级请参见**Volcano调度器**。仅华为云版Volcano插件支持开启路由加速特性。
- 训练使用的Python版本是3.7或3.9，否则无法实现ranktable路由加速。
- 训练作业的任务节点数要大于或等于3，否则会跳过ranktable路由加速。建议在大模型场景（512卡及以上）使用ranktable路由加速。
- 脚本执行目录不能是共享目录，否则ranktable路由加速会失败。
- 路由加速的原理是改变rank编号，所以代码中对rank的使用要统一，如果rank的使用不一致会导致训练异常。

操作步骤

步骤1 开启ModelArts Lite资源池对应的CCE集群的cabinet插件。

1. 在ModelArts Lite专属资源池列表，单击资源池名称，进入专属资源池详情页面。
2. 在基本信息页面单击CCE集群，跳转到CCE集群详情页面。

3. 在左侧导航栏选择“插件中心”，搜索“Volcano调度器”。
4. 单击“编辑”，查看高级配置的“plugins”参数下是否有“{"name":"cabinet"}”，如图4-10所示。

图 4-10 Volcano 调度器的高级配置

   - 是，则执行步骤2。
   - 否，则在高级配置的“plugins”参数下添加“{"name":"cabinet"}”，单击下方的“安装”使Volcano调度器更新配置，完成滚动重启。

步骤2 修改torch_npu训练启动脚本。

须知

脚本要使用torch.distributed.launch/run命令启动，不能使用mp.spawn命令启动，否则无法实现ranktable路由加速。

在使用Pytorch训练时，需要将“RANK_AFTER_ACC”环境变量赋值给“NODE_RANK”，使得ranktable路由规划生效。训练启动脚本（xxxx_train.sh）示例如下。其中“MASTER_ADDR”和“NODE_RANK”必须保持该赋值。

```
#!/bin/bash

# MASTER_ADDR
MASTER_ADDR="${MA_VJ_NAME}-${MA_TASK_NAME}-${MA_MASTER_INDEX}.${MA_VJ_NAME}"
NODE_RANK="$RANK_AFTER_ACC"
NNODES="$MA_NUM_HOSTS"
NGPUS_PER_NODE="$MA_NUM_GPUS"
# self-define, it can be changed to >=10000 port
MASTER_PORT="39888"

# replace ${MA_JOB_DIR}/code/torch_ddp.py to the actutal training script
```

```
PYTHON_SCRIPT=${MA_JOB_DIR}/code/torch_ddp.py
PYTHON_ARGS=""

# set hccl timeout time in seconds
export HCCL_CONNECT_TIMEOUT=1800

# replace ${ANACONDA_DIR}/envs/${ENV_NAME}/bin/python to the actual python
CMD="${ANACONDA_DIR}/envs/${ENV_NAME}/bin/python -m torch.distributed.launch \
    --nnodes=$NNODES \
    --node_rank=$NODE_RANK \
    --nproc_per_node=$NGPUS_PER_NODE \
    --master_addr $MASTER_ADDR \
    --master_port=$MASTER_PORT \
    $PYTHON_SCRIPT \
    $PYTHON_ARGS
"
echo $CMD
$CMD
```

步骤3 在主机上新建“config.yaml”文件。

“config.yaml”文件用于配置pod，代码示例如下。代码中的“xxxx_train.sh”即为**步骤2**修改的训练启动脚本。

```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: yourvcjobname                # job名字，根据实际场景修改
  namespace: default           # 命名空间，根据实际场景修改
  labels:
    ring-controller.cce: ascend-1980   # 保持不动
    fault-scheduling: "force"
spec:
  minAvailable: 6                    # 节点数，根据实际场景修改，对应分布式训练使用的节点数
  schedulerName: volcano               # 保持不动
  policies:
    - event: PodEvicted
      action: RestartJob
  plugins:
    configmap1980:
      - --rank-table-version=v2          # 保持不动，生成v2版本ranktablefile
    env: []
    svc:
      - --publish-not-ready-addresses=true   # 保持不动，pod间互相通信使用及生成一些必要环境变量
  maxRetry: 1
  queue: default
  tasks:
    - name: "worker"  # 保持不动
      replicas: 6                        # 任务数，对于pytorch而言就是节点数，与minAvailable一致即可
      template:
        metadata:
          annotations:
            cabinet: "cabinet"  # 保持不动，开启tor-topo下发的开关
          labels:
            app: pytorch-npu   # 标签，根据实际场景修改
            ring-controller.cce: ascend-1980  # 保持不动
        spec:
          affinity:
            podAntiAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
                - labelSelector:
                    matchExpressions:
                      - key: volcano.sh/job-name
                        operator: In
                        values:
                          - yourvcjobname   # job名字，根据实际场景修改
                  topologyKey: kubernetes.io/hostname
          containers:
            - image:  swr.xxxxxx.com/xxxx/custom_pytorch_npu:v1          # 镜像地址，根据实际场景修改
              imagePullPolicy: IfNotPresent
```

```
            name: pytorch-npu          # 容器名称，根据实际场景修改
            env:
              - name: OPEN_SCRIPT_ADDRESS     # 开放脚本地址，其中region-id根据实际region修改，例如cn-
southwest-2
                value: "https://mtest-bucket.obs.{region-id}.myhuaweicloud.com/acc/rank"
              - name: NAME
                valueFrom:
                  fieldRef:
                    fieldPath: metadata.name
              - name: MA_CURRENT_HOST_IP                    # 保持不动，表示运行时当前pod所在节点的ip
                valueFrom:
                  fieldRef:
                    fieldPath: status.hostIP
              - name: MA_NUM_GPUS     # 每个pod使用的NPU卡数，根据实际场景修改
                value: "8"
              - name: MA_NUM_HOSTS   # 参与分布式训练的节点数，与minAvailable一致即可
                value: "6"
              - name: MA_VJ_NAME # volcano job名称
                valueFrom:
                  fieldRef:
                    fieldPath: metadata.annotations['volcano.sh/job-name']
              - name: MA_TASK_NAME #任务pod名称
                valueFrom:
                  fieldRef:
                    fieldPath: metadata.annotations['volcano.sh/task-spec']
            command:
              - /bin/bash
              - -c
              - "wget ${OPEN_SCRIPT_ADDRESS}/bootstrap.sh -q && bash bootstrap.sh; export
RANK_AFTER_ACC=${VC_TASK_INDEX}; rank_acc=$(cat /tmp/RANK_AFTER_ACC 2>/dev/null); [ -n \"$
{rank_acc}\" ] && export RANK_AFTER_ACC=${rank_acc};export MA_MASTER_INDEX=$(cat /tmp/
MASTER_INDEX 2>/dev/null || echo 0); bash xxxx_train.sh" #  xxxx_train.sh换成实际训练脚本路径
            resources:
              requests:
                huawei.com/ascend-1980: "8"               # 每个节点的需求卡数，key保持不变。与
MA_NUM_GPUS一致
              limits:
                huawei.com/ascend-1980: "8"               # 每个节点的限制卡数，key保持不变。与
MA_NUM_GPUS一致
            volumeMounts:
              - name: ascend-driver             #驱动挂载，保持不动
                mountPath: /usr/local/Ascend/driver
              - name: ascend-add-ons            #驱动挂载，保持不动
                mountPath: /usr/local/Ascend/add-ons
              - name: localtime
                mountPath: /etc/localtime
              - name: hccn                       # 驱动hccn配置，保持不动
                mountPath: /etc/hccn.conf
              - name: npu-smi
                mountPath: /usr/local/bin/npu-smi
          nodeSelector:
            accelerator/huawei-npu: ascend-1980
          volumes:
            - name: ascend-driver
              hostPath:
                path: /usr/local/Ascend/driver
            - name: ascend-add-ons
              hostPath:
                path: /usr/local/Ascend/add-ons
            - name: localtime
              hostPath:
                path: /etc/localtime
            - name: hccn
              hostPath:
                path: /etc/hccn.conf
            - name: npu-smi
              hostPath:
                path: /usr/local/bin/npu-smi
          restartPolicy: OnFailure
```

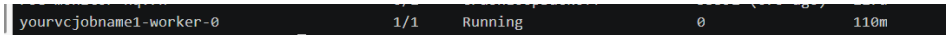

步骤4 执行如下命令，根据“config.yaml”创建并启动pod。容器启动后会自动执行训练作业。

```
kubectl apply -f config.yaml
```

步骤5 执行如下命令，检查pod启动情况。如果显示“1/1 running”状态代表启动成功。

```
kubectl get pod
```

图 4-11 启动成功的回显

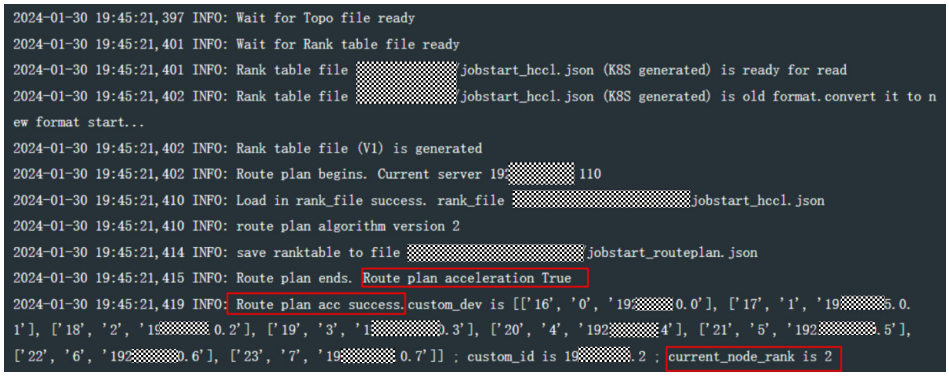


步骤6 执行如下命令，查看日志。日志显示如图所示表示成功执行动态路由。

```
kubectl logs {pod-name}
```

其中{pod-name}替换为实际pod名称，可以在**步骤5**的回显信息中获取。

图 4-12 成功执行动态路由的回显



说明

- 只有任务节点大于等于3的训练任务才能成功执行动态路由。
- 如果执行失败可以参考**故障排除：ranktable路由优化执行失败**处理。

----结束

故障排除：ranktable 路由优化执行失败

故障现象

容器日志有error信息。

可能原因

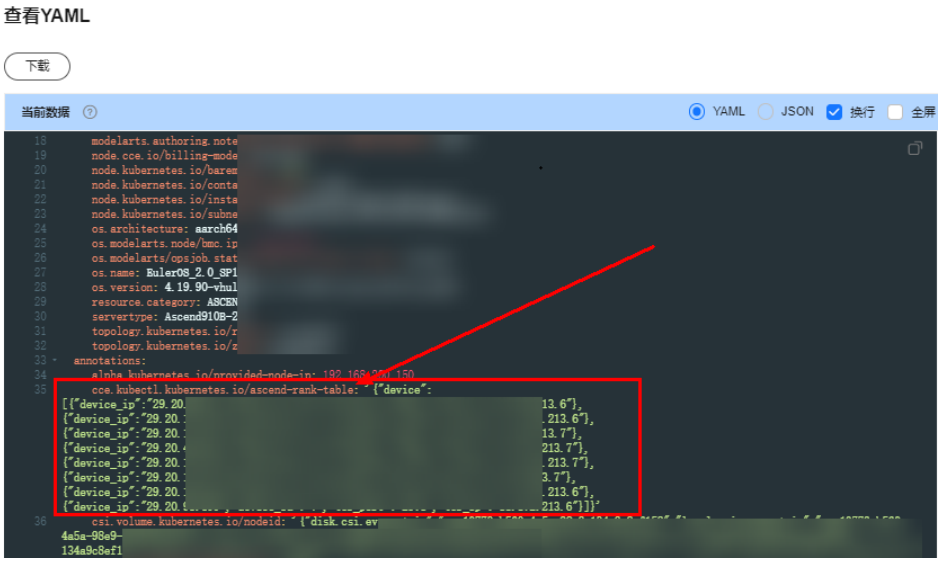
集群节点没有下发topo文件和ranktable文件。

操作步骤

1. 在ModelArts Lite专属资源池列表，单击资源池名称，进入专属资源池详情页面。
2. 在基本信息页面单击CCE集群，跳转到CCE集群详情页面。
3. 在CCE集群详情页，选择左侧导航栏的“节点管理”，选择“节点”页签。
4. 在节点列表，单击操作列的“更多 > 查看YAML”查看节点配置信息。
5. 查看节点的yaml文件里“cce.kubectl.kubernetes.io/ascend-rank-table”字段是否有值。

如图所示，表示有值，节点已开启topo文件和ranktable文件的下发。否则，联系技术支持处理。

图 4-13 查看节点的 yaml 文件

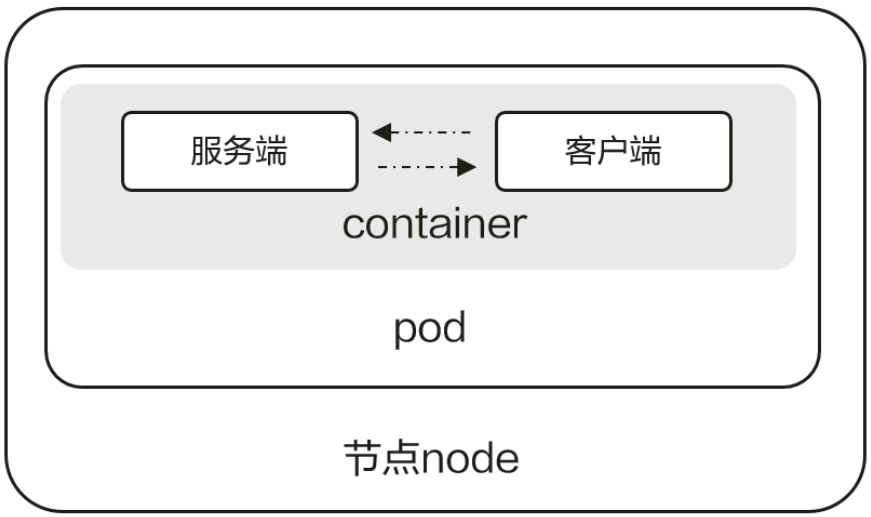


4.3 在 Lite Cluster 资源池上使用 Snt9B 完成推理任务

场景描述

本案例介绍如何在Snt9B环境中利用Deployment机制部署在线推理服务。首先创建一个Pod以承载服务，随后登录至该Pod容器内部署在线服务，并最终通过新开一个终端作为客户端来访问并测试该在线服务的功能。

图 4-14 任务示意图



操作步骤

步骤1 拉取镜像。本测试镜像为bert_pretrain_mindspore:v1，已经把测试数据和代码打进镜像中。

```
docker pull swr.cn-southwest-2.myhuaweicloud.com/os-public-repo/bert_pretrain_mindspore:v1
docker tag swr.cn-southwest-2.myhuaweicloud.com/os-public-repo/bert_pretrain_mindspore:v1
bert_pretrain_mindspore:v1
```

步骤2 在主机上新建config.yaml文件。

config.yaml文件用于配置pod，本示例中使用sleep命令启动pod，便于进入pod调试。您也可以修改command为对应的任务启动命令（如“python inference.py”），任务会在启动容器后执行。

config.yaml内容如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: yourapp
  labels:
     app: infers
spec:
  replicas: 1
  selector:
    matchLabels:
      app: infers
  template:
    metadata:
      labels:
        app: infers
    spec:
      schedulerName: volcano
      nodeSelector:
        accelerator/huawei-npu: ascend-1980
      containers:
      - image: bert_pretrain_mindspore:v1                # Inference image name
        imagePullPolicy: IfNotPresent
        name: mindspore
        command:
        - "sleep"
        - "1000000000000000000"
        resources:
          requests:
            huawei.com/ascend-1980: "1"            # 需求卡数，key保持不变。Number of required NPUs. The
maximum value is 16. You can add lines below to configure resources such as memory and CPU.
          limits:
            huawei.com/ascend-1980: "1"            # 限制卡数，key保持不变。The value must be consistent
with that in requests.
        volumeMounts:
        - name: ascend-driver              #驱动挂载，保持不动
          mountPath: /usr/local/Ascend/driver
        - name: ascend-add-ons           #驱动挂载，保持不动
          mountPath: /usr/local/Ascend/add-ons
        - name: hccn                        #驱动hccn配置，保持不动
          mountPath: /etc/hccn.conf
        - name: npu-smi                           #npu-smi
          mountPath: /usr/local/bin/npu-smi
        - name: localtime                    #The container time must be the same as the host time.
          mountPath: /etc/localtime
      volumes:
      - name: ascend-driver
        hostPath:
          path: /usr/local/Ascend/driver
      - name: ascend-add-ons
        hostPath:
          path: /usr/local/Ascend/add-ons
      - name: hccn
```



```
      hostPath:
        path: /etc/hccn.conf
    - name: npu-smi
      hostPath:
        path: /usr/local/bin/npu-smi
    - name: localtime
      hostPath:
        path: /etc/localtime
```

步骤3 根据config.yaml创建pod。

```
kubectl apply -f config.yaml
```

步骤4 检查pod启动情况，执行下述命令。如果显示“1/1 running”状态代表启动成功。

```
kubectl get pod -A
```

步骤5 进入容器，{pod_name}替换为您的pod名字（get pod中显示的名字），{namespace}替换为您的命名空间（默认为default）。

```
kubectl exec -it {pod_name} bash -n {namespace}
```

步骤6 激活conda模式。

```
su - ma-user  //切换用户身份
conda activate MindSpore //激活 MindSpore环境
```

步骤7 创建测试代码test.py。

```
from flask import Flask, request
import json
app = Flask(__name__)

@app.route('/greet', methods=['POST'])
def say_hello_func():
    print("----------- in hello func ----------")
    data = json.loads(request.get_data(as_text=True))
    print(data)
    username = data['name']
    rsp_msg = 'Hello, {}!'.format(username)
    return json.dumps({"response":rsp_msg}, indent=4)

@app.route('/goodbye', methods=['GET'])
def say_goodbye_func():
    print("----------- in goodbye func ----------")
    return '\nGoodbye!\n'


@app.route('/', methods=['POST'])
def default_func():
    print("----------- in default func ----------")
    data = json.loads(request.get_data(as_text=True))
    return '\n called default func !\n {} \n'.format(str(data))

# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

执行代码，执行后如下图所示，会部署一个在线服务，该容器即为服务端。

```
python test.py
```

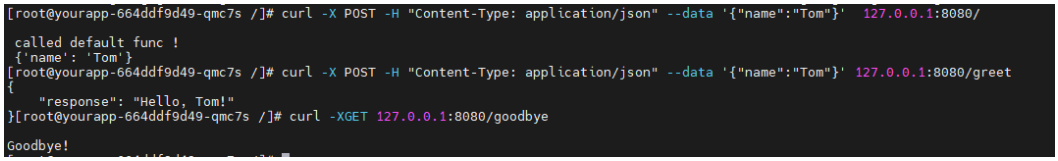
图 4-15 部署在线服务

```
(MindSpore) [root@yourapp-664ddf9d49-qmc7s /]# python a.py
 * Serving Flask app 'a' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.16.0.45:8080
Press CTRL+C to quit
```

步骤8 在XShell中新开一个终端，参考步骤5~7进入容器，该容器为客户端。执行以下命令验证自定义镜像的三个API接口功能。当显示如图所示时，即可调用服务成功。

```
curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}'  127.0.0.1:8080/
curl -X POST -H "Content-Type: application/json" --data '{"name":"Tom"}' 127.0.0.1:8080/greet
curl -X GET 127.0.0.1:8080/goodbye
```

图 4-16 访问在线服务



说明

limit/request配置cpu和内存大小，已知单节点Snt9B机器为：8张Snt9B卡+192u1536g，请合理规划，避免cpu和内存限制过小引起任务无法正常运行。

----结束

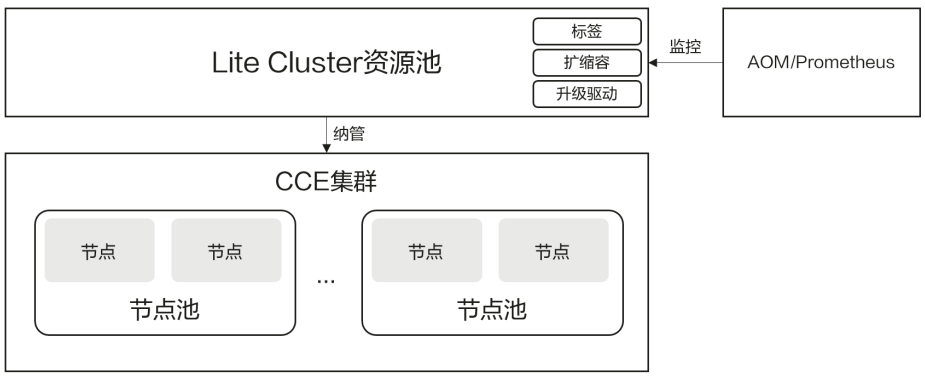
5 Lite Cluster 资源管理

5.1 Lite Cluster 资源管理介绍

在ModelArts控制台，您可以对已创建的资源进行管理。通过单击资源池名称，可以进入到资源池详情页，您可以在详情页进行下述操作。

- **管理Lite Cluster节点**：节点是容器集群组成的基本元素，您可以对资源池内单节点进行替换、删除、重置等操作。
- **管理Lite Cluster节点池**：为帮助您更好地管理Kubernetes集群内的节点，ModelArts支持通过节点池来管理节点。节点池是集群中具有相同配置的一组节点，一个节点池包含一个节点或多个节点，您可以创建、更新和删除节点池。
- **管理Lite Cluster资源池标签**：ModelArts支持为资源池添加标签，用来标识云资源，方便您快速搜索到资源池。
- **扩缩容Lite Cluster资源池**：当Cluster资源池创建完成，使用一段时间后，由于用户AI开发业务的变化，对于资源池资源量的需求可能会产生变化，面对这种场景，ModelArts提供了扩缩容功能，用户可以根据自己的需求动态调整。
- **升级Lite Cluster资源池驱动**：当资源池中的节点含有GPU/Ascend资源时，用户基于自己的业务，可能会有自定义GPU/Ascend驱动的需求，ModelArts面向此类客户提供了自助升级专属资源池GPU/Ascend驱动的能力。
- **监控Lite Cluster资源**：ModelArts支持使用AOM和Prometheus对资源进行监控，方便您了解当前的资源使用情况。
- **释放Lite Cluster资源**：针对不再使用的Lite Cluster资源，您可以释放资源。

图 5-1 Lite Cluster 资源管理介绍



5.2 管理 Lite Cluster 节点

节点是容器集群组成的基本元素，在资源池详情页，单击“节点管理”页签，进行替换、删除、重置等操作。

- 删除/退订/释放节点：
  - 若是“按需计费”的资源池，您可单击操作列的“删除”，即可实现对单个节点的资源释放。

    若想批量删除节点，勾选待删除节点名称前的复选框，然后单击名称上方的“删除”，即可实现对多个节点的资源释放。
  - 若是“包年/包月”且资源未到期的资源池，您可单击操作列的“退订”，即可实现对单个节点的资源释放。
  - 若是“包年/包月”且资源到期的资源池（处于宽限期），您可单击操作列的“释放”，即可实现对单个节点的资源释放。

  部分“包年/包月”节点会出现“删除”按钮，原因是该节点为存量节点，单击“删除”即可实现节点的资源释放。

  说明

  - 删除/退订/释放节点可能导致该节点上运行的作业失败，请保证该节点无任务运行时再进行操作。
  - 当资源池中存在异常节点时，可通过删除/退订/释放操作，将资源池中指定的异常节点移除，再通过扩容专属资源池获得和之前相同的总节点个数。
  - 仅有一个节点时，无法进行删除/退订/释放操作。
- 替换节点：

  “节点管理”页签中提供对单个节点替换的功能。可单击操作列的“替换”，即可实现对单个节点的替换。替换节点操作不会收取费用。

  单击“操作记录”可查看当前资源池替换节点的操作记录。“运行中”表示节点在替换中。替换成功后，节点列表中会显示新的节点名称。

  替换最长时间为24小时，超时后仍然未找到合适的资源，状态会变为“失败”。

  可将鼠标悬浮在图标上，查看具体失败原因。

  说明

  - 每天累计替换的次数不超过资源池节点总数的20%，同时替换的节点数不超过资源池节点总数的5%。
  - 替换节点时需确保有空闲节点资源，否则替换可能失败。
  - 当操作记录里有节点处于重置中时，该资源池无法进行替换节点操作。
- 重置节点

  “节点管理”页签中提供节点重置的功能。单击操作列的“重置”，可实现对单个节点的重置。勾选多个节点的复选框，单击操作记录旁的“重置”按钮，可实现对多个节点的重置。

  如图5-2，下发重置节点任务时需要填写以下参数：

表 5-1 重置参数说明

参数名称	说明
操作系统	选择下拉框中支持的操作系统。
配置方式	选择重置节点的配置方式。 • 按节点比例：重置任务包含多个节点时，同时被重置节点的最高比例。 • 按节点数量：重置任务包含多个节点时，同时被重置节点的最大个数。

图 5-2 重置节点

单击“操作记录”可查看当前资源池重置节点的操作记录（如图5-3）。重置中节点状态为“重置中”，重置成功后，节点状态变为“可用”（如图5-4）。重置节点操作不会收取费用。

说明

- 重置节点将影响相关业务的运行，重置时本地盘会被清空、节点上的k8s标签会被清除，请谨慎操作。
- 节点状态为“可用”的节点才能进行重置。
- 同一时间单个节点只能处于一个重置任务中，无法对同一个节点同时下发多个重置任务。
- 当操作记录里有节点处于替换中时，该资源池无法进行重置节点操作。
- 当资源池处于驱动升级状态时，该资源池无法进行重置节点操作。
- GPU和NPU规格，重置节点完成后，节点可能会出现驱动升级的现象，请耐心等待。

图 5-3 操作记录

图 5-4 查看资源池节点

- 授权运维

  华为云技术支持在故障定位和性能诊断时，部分运维操作需要用户授权才可进行。您可在资源池详情页的节点页签下，找到对应节点，在操作列单击“更多 > 授权”，在弹出的提示框中单击“确认”即可完成授权。

图 5-5 授权

说明

正常情况下，该授权按钮为置灰状态。当华为云技术支持发起运维申请后，按钮会变为可点状态。

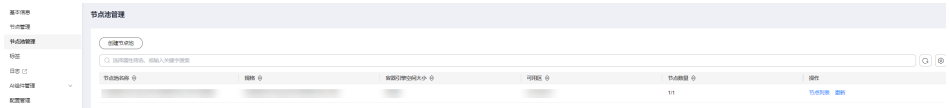
在完成运维操作后，华为云技术支持会主动关闭已获得授权，无需您额外操作。

5.3 管理 Lite Cluster 节点池

为帮助您更好地管理Kubernetes集群内的节点，ModelArts支持通过节点池来管理节点。一个节点池包含一个节点或多个节点，能通过节点池批量配置一组节点。关于更多节点池的介绍，可以查看**节点池概述**。

在资源池详情页，单击“节点池管理”页签，您可以创建、更新和删除节点池。

图 5-6 节点池管理



- 创建节点池

  当您需要更多节点池时，可单击“创建节点池”新增节点池，相关参数请参见**Lite Cluster资源开通**。

- 查看节点列表

  当您想查看某一节点池下的节点相关信息，可单击操作列的“节点列表”，可查询节点的名称、规格及可用区。

- 更新节点池

  当您想更新节点池配置时，可单击操作列的“更新”，相关参数介绍请参见**Step6 购买Cluster资源**。

  需注意，更新节点池配置时，配置仅对新增的节点生效，其中仅节点池K8S标签及污点支持对存量节点同步改动（勾选对应的复选框）。

  图 5-7 更新节点池

- 删除节点池

  当有多个节点池时，支持删除节点池，此时在操作列会显示“删除”按钮，单击“删除”后输入“DELETE”并单击“确定”即可。

  每个资源池至少需要有一个节点池，当只有一个节点池时不支持删除。

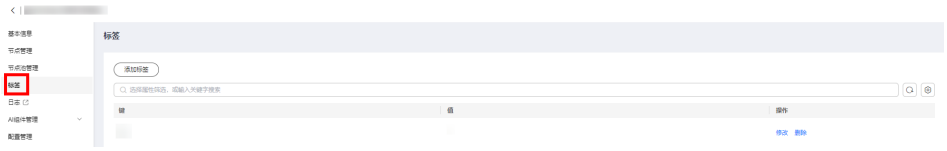
5.4 管理 Lite Cluster 资源池标签

通过给资源池添加标签，可以标识云资源，便于快速搜索资源池。

1. 在ModelArts管理控制台的左侧导航栏中选择“专属资源池 > 弹性集群”。
2. 在资源池列表中，单击资源池名称进入资源池详情页面。
3. 在资源池详情页面，单击“标签”页签查看标签信息。

   支持添加、修改、删除标签。标签详细用法请参见**ModelArts如何通过标签实现资源分组管理**。

图 5-8 标签



说明

最多支持添加20个标签。

5.5 扩缩容 Lite Cluster 资源池

场景介绍

当专属资源池创建完成，使用一段时间后，由于用户AI开发业务的变化，对于资源池资源量的需求可能会产生变化，面对这种场景，ModelArts专属资源池提供了扩缩容功能，用户可以根据自己的需求动态调整。

说明

缩容操作可能影响到正在运行的业务，建议用户在业务空窗期进行缩容，或进入资源池详情页面，在指定空闲的节点上进行删除来实现缩容。

约束限制

- 只支持对状态为“运行中”的专属资源池进行扩缩容。
- 专属资源池不能缩容到0。

扩缩容专属资源池

资源池扩缩容有以下类型，分别为：

- 对已有规格增减节点数量
- 修改容器引擎空间大小

1. 登录ModelArts管理控制台，在左侧菜单栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”页签，查看资源池列表。
2. 增减节点数量

   单击某个资源池操作列的“扩缩容”对资源池进行扩缩容（对于为包周期资源池，此按钮为“扩容”，若需要缩容，请进入到包周期资源池详情页对节点进行退订操作）。

   在“专属资源池扩缩容”页面，设置“资源配置 > 可用区”，可用区可选择随机分配和指定AZ。设置完成后，单击“提交”，在弹出的确认框中单击“确定”完成修改。

   - 选择随机分配时，可通过增减“目标总节点数”实现扩缩容，请用户根据本身业务诉求进行调整。增加目标节点数量即表示扩容，减少目标节点数量即表示缩容。扩缩容完成后，节点的可用区分布由系统后台随机选择。
   - 选择指定AZ时，可指定扩缩容完成后节点的可用区分布。

图 5-9 资源配置（单节点方式）

若购买资源池时，节点数量采用整柜方式购买（部分规格支持），则在扩缩容时为整柜方式扩缩容，目标节点总数等于“数量*整柜”。“整柜”参数为创建资源池时选择，扩缩容时不可修改。用户通过增减“数量”来改变“目标总节点数”。

图 5-10 资源配置（整柜方式）

说明

用户增加节点数量时，可以通过指定节点计费模式，为资源池新创建的节点设置不同于资源池的计费模式，例如用户可以在包周期的资源池中创建按需的节点。若用户不指定该参数，创建的节点计费模式和资源池保持一致。

3. 新增节点池

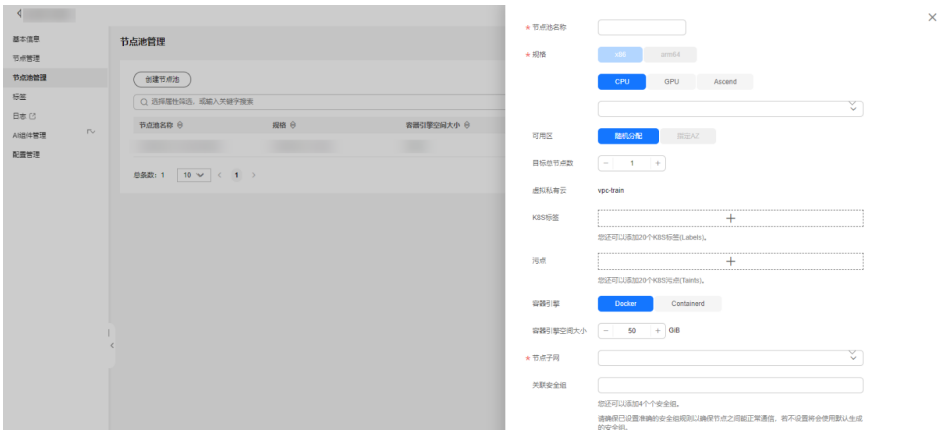
   若您需要更多的节点池，您可以通过以下操作创建新的节点池。

   方式一：在资源池详情页，单击“节点池管理”页签，单击创建节点池。

   方式二：在资源池列表页，单击某个资源池操作列下的“更多 > 新增节点池”，跳转至“节点池管理”页签，修改容器引擎空间大小。

   创建节点池相关参数请参见Lite Cluster资源开通。

图 5-11 新增节点池（单节点方式）



新增节点池时，部分规格支持整柜购买，若选中此类规格，可以通过下拉框选择整柜方式或单节点方式新增节点池。若选择整柜方式，目标总节点数等于“数量*整柜”，购买的节点总数为两者的乘积。

图 5-12 新增节点池（整柜方式）

说明

用户新增节点池时，可以通过指定节点的计费模式，为资源池新创建的节点设置不同于资源池的计费模式，例如用户可以在包周期的资源池中创建按需的节点。若用户不指定该参数，创建的节点计费模式和资源池保持一致。

4. 修改容器引擎空间大小

   若您需要更大的容器引擎空间，您可以通过以下操作调整容器引擎空间大小。

   - 对于新建的资源，支持在新建时指定容器引擎空间大小。
     - 方式一：支持新建资源池时指定容器引擎空间大小，请参见**Lite Cluster资源开通**中“规格管理”参数下“高级选项”。
     - 方式二：单击某个资源池名称，进入资源池详情，单击“节点池管理”页签，单击“创建节点池”，填写“容器引擎空间大小”后，单击“确认”。
     - 方式三：单击某个资源池操作列下的“更多 > 新增节点池”，跳转至“节点池管理”页签，修改容器引擎空间大小。（仅包周期支持新增节点池）
   - 对于存量的资源，支持修改容器引擎空间大小。
     - 方式一：单击某个资源池名称，进入资源池详情，单击“节点池管理”页签，单击对应节点池操作列的“更新”，填写“容器引擎空间大小”后，单击“确认”（新建节点的容器引擎空间大小会自动默认为修改后的值）。
     - 方式二：单击某个资源池操作列的“扩缩容”，修改容器引擎空间大小（仅作用在新建节点上）。

     须知

     存量节点不支持修改容器引擎空间大小，仅作用在新建节点上，且会导致资源池内该规格下节点的dockerBaseSize不一致，可能会使得部分任务在不同节点的运行情况不一致。

图 5-13 修改容器引擎空间大小（节点池管理页签界面）

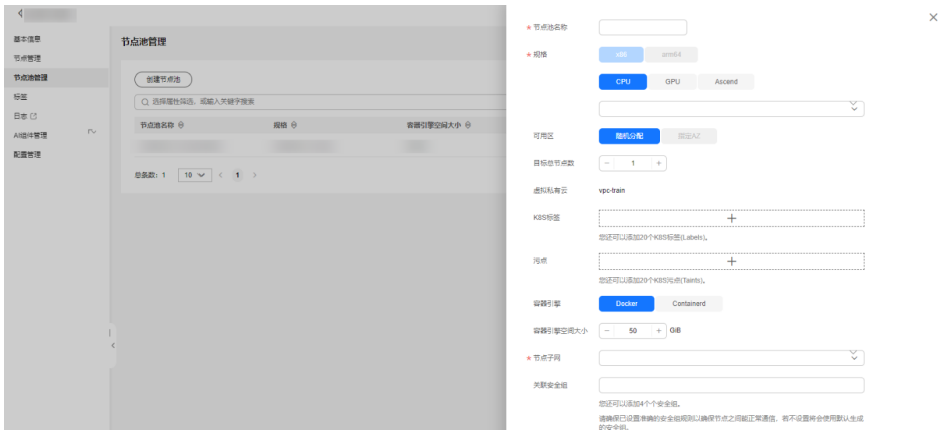


图 5-14 修改容器引擎空间大小（扩缩容界面）

您还可以在以上界面修改容器引擎类型。容器引擎是Kubernetes最重要的组件之一，负责管理镜像和容器的生命周期。Kubelet通过Container Runtime Interface (CRI) 与容器引擎交互，以管理镜像和容器。其中Containerd调用链更短，组件更少，更稳定，占用节点资源更少，Containerd和Docker差异对比请见**容器引擎**。

若CCE集群版本低于1.23，仅支持选择Docker作为容器引擎。若CCE集群版本大于等于1.27，仅支持选择Containerd作为容器引擎。其余CCE集群版本，支持选择Containerd或Docker作为容器引擎。

5.6 升级 Lite Cluster 资源池驱动

场景介绍

当专属资源池中的节点含有GPU/Ascend资源时，用户基于自己的业务，可能会有自定义GPU/Ascend驱动的需求，ModelArts面向此类客户提供了自助升级专属资源池GPU/Ascend驱动的能力。

驱动升级有两种升级方式：安全升级、强制升级。

说明

- 安全升级：不影响正在运行的业务，开始升级后会先将节点进行隔离（不能再下发新的作业），待节点上的存量作业运行完成后再进行升级，因需要等待作业完成，故升级周期可能比较长。
- 强制升级：忽略资源池中正在运行的作业，直接进行驱动升级，可能会导致运行中作业失败，需谨慎选择。

约束限制

专属资源池状态处于运行中，且专属池中的节点需要含有GPU/Ascend资源。

驱动升级操作

1. 登录ModelArts管理控制台，在左侧导航栏中选择“专属资源池 > 弹性集群”，默认进入“资源池”页面。
2. 在资源池列表中，选择需要进行驱动升级的资源池“操作 > 驱动升级”。
3. 在“驱动升级”弹窗中，会显示当前专属资源池的驱动类型、节点数量、当前版本、目标版本和升级方式。
   - 目标版本：在目标版本下拉框中，选择一个目标驱动版本。
   - 升级方式：选择“升级方式”，可选择安全升级或强制升级。
   - 开启滚动：点击开启后，支持滚动升级的方式进行驱动升级。当前支持“按节点比例”和“按节点数量”两种滚动方式。
     - 按节点比例：每批次驱动升级的节点数量为“节点比例*资源池节点总数”。
     - 按节点数量：每批次驱动升级的节点数量为设置的节点数量。

     对于不同的升级方式，滚动升级选择节点的策略会不同：
     - 如果升级方式为安全升级，则根据滚动节点数量选择无业务的节点，隔离节点并滚动升级。
     - 如果升级方式为强制升级，则根据滚动节点数量随机选择节点，隔离节点并滚动升级。

     说明

     - 无业务节点定义：在资源池详情“节点管理”页签下，如果GPU/Ascend的可用数等于总数，则为无业务节点。

       图 5-15 查看无业务节点

     - 滚动驱动升级时，驱动异常的节点对升级无影响，会和驱动正常的节点一起升级。

图 5-16 驱动升级

4. 选择完成后，单击“确定”开始驱动升级。

5.7 监控 Lite Cluster 资源

5.7.1 使用 AOM 看 Lite Cluster 监控指标

监控已有指标

Modelarts会定期收集资源池中各节点的关键资源（GPU、NPU、CPU、Memory等）的使用情况并上报到AOM，用户可直接在AOM上查看默认配置好的基础指标，详细步骤如下：

1. 登录控制台，搜索AOM，进入“应用运维管理 AOM”控制台。
2. 单击“监控 > 指标浏览”，进入“指标浏览”“页面”，单击“添加指标查询”。

3. 添加指标查询信息。

- 添加方式：选择“按指标维度添加”。
- 指标名称：在右侧下拉框中选择“全量指标”，然后选择想要查询的指标，参考**表5-2**、**表5-3**
- 指标维度：填写过滤该指标的标签，请参考**表4的Label名字栏**。样例如下：

4. 单击确定，即可出现指标信息。

表 5-2 容器级别的指标

分类	名称	指标	指标含义	单位	取值范围
CPU	CPU使用率	ma_container_cpu_util	该指标用于统计测量对象的CPU使用率。	百分比（Percent）	0～100%
	CPU内核占用量	ma_container_cpu_used_core	该指标用于统计测量对象已经使用的CPU核个数	核（Core）	≥0
	CPU内核总量	ma_container_cpu_limit_core	该指标用于统计测量对象申请的CPU核总量。	核（Core）	≥1
内存	内存总量	ma_container_memory_capacity_megabytes	该指标用于统计测量对象申请的物理内存总量。	兆字节（Megabytes）	≥0

分类	名称	指标	指标含义	单位	取值范围
	物理内存使用率	ma_container_memory_util	该指标用于统计测量对象已使用内存占申请物理内存总量的百分比。	百分比（Percent）	0～100%
	物理内存使用量	ma_container_memory_used_megabytes	该指标用于统计测量对象实际已经使用的物理内存(对应container_memory_working_set_bytes当前内存工作集（working set）使用量。（工作区内存使用量=活跃的匿名与和缓存，以及file-baked页<=container_memory_usage_bytes）)	兆字节（Megabytes）	≥0
存储	磁盘读取速率	ma_container_disk_read_kilobytes	该指标用于统计每秒从磁盘读出的数据量。	千字节/秒（Kilobytes/Second）	≥0
	磁盘写入速率	ma_container_disk_write_kilobytes	该指标用于统计每秒写入磁盘的数据量。	千字节/秒（Kilobytes/Second）	≥0
GPU显存	显存容量	ma_container_gpu_mem_total_megabytes	该指标用于统计训练任务的显存容量。	兆字节（Megabytes）	>0
	显存使用率	ma_container_gpu_mem_util	该指标用于统计测量对象已使用的显存占显存容量的百分比。	百分比（Percent）	0～100%

分类	名称	指标	指标含义	单位	取值范围
	显存使用量	ma_container_gpu_mem_used_megabytes	该指标用于统计测量对象已使用的显存。	兆字节（Megabytes）	≥0
GPU	GPU使用率	ma_container_gpu_util	该指标用于统计测量对象的GPU使用率。	百分比（Percent）	0～100%
	GPU内存带宽利用率	ma_container_gpu_mem_copy_util	表示内存带宽利用率。以英伟达GP Vnt1为例，其最大内存带宽为900 GB/sec，如果当前的内存带宽为450 GB/sec，则内存带宽利用率为50%。	百分比（Percent）	0～100%
	GPU编码器利用率	ma_container_gpu_enc_util	表示编码器利用率	百分比（Percent）	%
	GPU解码器利用率	ma_container_gpu_dec_util	表示解码器利用率	百分比（Percent）	%
	GPU温度	DCGM_FI_DEV_GPU_TEMP	表示GPU温度。	摄氏度（℃）	自然数
	GPU功率	DCGM_FI_DEV_POWER_USAGE	表示GPU功率。	瓦特（W）	>0
	显存温度	DCGM_FI_DEV_MEMORY_TEMP	表示显存温度。	摄氏度（℃）	自然数
网络IO	下行速率	ma_container_network_receive_bytes	该指标用于统计测试对象的入方向网络流速。	字节/秒（Bytes/Second）	≥0
	接收包速率	ma_container_network_receive_packets	每秒网卡接收的数据包个数。	个/秒（Packets/Second）	≥0

分类	名称	指标	指标含义	单位	取值范围
	下行错包率	ma_container_network_receive_error_packets	每秒网卡接收的错误包个数。	个/秒（Packets/Second）	≥0
	上行速率	ma_container_network_transmit_bytes	该指标用于统计测试对象的出方向网络流速。	字节/秒（Bytes/Second）	≥0
	上行错包率	ma_container_network_transmit_error_packets	每秒网卡发送的错误包个数。	个/秒（Packets/Second）	≥0
	发送包速率	ma_container_network_transmit_packets	每秒网卡发送的数据包个数。	个/秒（Packets/Second）	≥0
NPU	NPU使用率	ma_container_npu_util	该指标用于统计测量对象的NPU使用率。（即将废止，替代指标为ma_container_npu_ai_core_util）。	百分比（Percent）	0～100%
	NPU显存使用率	ma_container_npu_memory_util	该指标用于统计测量对象已使用的NPU显存占NPU存储容量的百分比。（即将废止，snt3系列替代指标为ma_container_npu_ddr_memory_util，snt9系列替代指标为ma_container_npu_hbm_util）。	百分比（Percent）	0～100%

分类	名称	指标	指标含义	单位	取值范围
	NPU显存使用量	ma_container_npu_memory_used_megabytes	该指标用于统计测量对象已使用的NPU显存。（即将废止，snt3系列替代指标为ma_container_npu_ddr_memory_usage_bytes，snt9系列替代指标为ma_container_npu_hbm_usage_bytes）。	≥0	兆字节（Megabytes）
	NPU显存容量	ma_container_npu_memory_total_megabytes	该指标用于统计测量对象的NPU显存容量。（即将废止，snt3系列替代指标为ma_container_npu_ddr_memory_bytes，snt9系列替代指标为ma_container_npu_hbm_bytes）。	>0	兆字节（Megabytes）
	AI处理器错误码	ma_container_npu_ai_core_error_code	昇腾系列AI处理器错误码	-	-
	AI处理器健康状态	ma_container_npu_ai_core_health_status	昇腾系列AI处理器健康状态	-	• 1：健康 • 0：不健康

分类	名称	指标	指标含义	单位	取值范围
	AI处理器功耗	ma_container_npu_ai_core_power_usage_watts	昇腾系列AI处理器功耗（snt9和snt3为处理器功耗，snt3P为板卡功耗）	瓦特（W）	>0
	AI处理器温度	ma_container_npu_ai_core_temperature_celsius	昇腾系列AI处理器温度	摄氏度（℃）	自然数
	AI处理器AI CORE利用率	ma_container_npu_ai_core_util	昇腾系列AI处理器AI Core利用率	百分比（Percent）	0～100%
	AI处理器AI CORE时钟频率	ma_container_npu_ai_core_frequency_hertz	昇腾系列AI处理器AI Core时钟频率	赫兹（Hz）	>0
	AI处理器电压	ma_container_npu_ai_core_voltage_volts	昇腾系列AI处理器电压	伏特（V）	自然数
	AI处理器DDR内存总量	ma_container_npu_ddr_memory_bytes	昇腾系列AI处理器DDR内存总量	字节（Byte）	>0
	AI处理器DDR内存使用量	ma_container_npu_ddr_memory_usage_bytes	昇腾系列AI处理器DDR内存使用量	字节（Byte）	>0
	AI处理器DDR内存利用率	ma_container_npu_ddr_memory_util	昇腾系列AI处理器DDR内存利用率	百分比（Percent）	0～100%
	AI处理器HBM内存总量	ma_container_npu_hbm_bytes	昇腾系列AI处理器HBM总内存（昇腾snt9 AI处理器专属）	字节（Byte）	>0
	AI处理器HBM内存使用量	ma_container_npu_hbm_usage_bytes	昇腾系列AI处理器HBM内存使用量（昇腾snt9 AI处理器专属）	字节（Byte）	>0

分类	名称	指标	指标含义	单位	取值范围
	AI处理器HBM内存利用率	ma_container_npu_hbm_util	昇腾系列AI处理器HBM内存利用率（昇腾snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存带宽利用率	ma_container_npu_hbm_bandwidth_util	昇腾系列AI处理器HBM内存带宽利用率（昇腾snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存时钟频率	ma_container_npu_hbm_frequency_hertz	昇腾系列AI处理器HBM内存时钟频率（昇腾snt9 AI处理器专属）	赫兹（Hz）	>0
	AI处理器HBM内存温度	ma_container_npu_hbm_temperature_celsius	昇腾系列AI处理器HBM内存温度（昇腾snt9 AI处理器专属）	摄氏度（℃）	自然数
	AI处理器AI CPU利用率	ma_container_npu_ai_cpu_util	昇腾系列AI处理器AI CPU利用率	百分比（Percent）	0～100%
	AI处理器控制CPU利用率	ma_container_npu_ctrl_cpu_util	昇腾系列AI处理器控制CPU利用率	百分比（Percent）	0～100%

表 5-3 节点指标

分类	名称	指标	指标含义	单位	取值范围
CPU	CPU内核总量	ma_node_cpu_limit_core	该指标用于统计测量对象申请的CPU核总量。	核（Core）	≥1

分类	名称	指标	指标含义	单位	取值范围
	CPU内核占用	ma_node_cpu_used_core	该指标用于统计测量对象已经使用的CPU核数。	核（Core）	≥0
	CPU使用率	ma_node_cpu_util	该指标用于统计测量对象的CPU使用率。	百分比（Percent）	0～100%
	CPU IO等待时间	ma_node_cpu_iowait_counter	从系统启动开始累计到当前时刻，硬盘IO等待时间	jiffies	≥0
内存	物理内存使用率	ma_node_memory_util	该指标用于统计测量对象已使用内存占申请物理内存总量的百分比。	百分比（Percent）	0～100%
	物理内存容量	ma_node_memory_total_megabytes	该指标用于统计测量申请的物理内存总量。	兆字节（Megabytes）	≥0
网络IO	下行Bps	ma_node_network_receive_rate_bytes_seconds	该指标用于统计测试对象的入方向网络流速。	字节/秒（Bytes/Second）	≥0
	上行Bps	ma_node_network_transmit_rate_bytes_seconds	该指标用于统计测试对象的出方向网络流速。	字节/秒（Bytes/Second）	≥0
存储	磁盘读取速率	ma_node_disk_read_rate_kilobytes_seconds	该指标用于统计每秒从磁盘读出的数据量。只考虑被容器使用的数据盘。	千字节/秒（Kilobytes/Second）	≥0

分类	名称	指标	指标含义	单位	取值范围
	磁盘写入速率	ma_node_disk_write_rate_kilobytes_seconds	该指标用于统计每秒写入磁盘的数据量。只考虑被容器使用的数据盘。	千字节/秒（Kilobytes/Second）	≥0
	cache空间的总量	ma_node_cache_space_capacity_megabytes	该指标用于统计k8s空间的总容量。	兆字节（Megabytes）	≥0
	cache空间的使用量	ma_node_cache_space_used_capacity_megabytes	该指标用于统计k8s空间的使用量。	兆字节（Megabytes）	≥0
	容器空间的总量	ma_node_container_space_capacity_megabytes	该指标用于统计容器空间的总容量。	兆字节（Megabytes）	≥0
	容器空间的使用量	ma_node_container_space_used_capacity_megabytes	该指标用于统计容器空间的使用量。	兆字节（Megabytes）	≥0
GPU	GPU使用率	ma_node_gpu_util	该指标用于统计测量对象的GPU使用率。	百分比（Percent）	0～100%
	显存容量	ma_node_gpu_mem_total_megabytes	该指标用于统计测量对象的显存容量。	兆字节（Megabytes）	>0
	显存使用率	ma_node_gpu_mem_util	该指标用于统计测量对象已使用的显存占显存容量的百分比。	百分比（Percent）	0～100%
	显存使用量	ma_node_gpu_mem_used_megabytes	该指标用于统计测量对象已使用的显存。	兆字节（Megabytes）	≥0

分类	名称	指标	指标含义	单位	取值范围
	共享GPU任务运行数据	node_gpu_share_job_count	针对一个GPU卡，当前运行的共享资源使用的任务数量。	个	≥0
	GPU温度	DCGM_FI_DEV_GPU_TEMP	表示GPU温度。	摄氏度（℃）	自然数
	GPU功率	DCGM_FI_DEV_POWER_USAGE	表示GPU功率。	瓦特（W）	>0
	显存温度	DCGM_FI_DEV_MEMORY_TEMP	表示显存温度。	摄氏度（℃）	自然数
NPU	NPU使用率	ma_node_npu_util	该指标用于统计测量对象的NPU使用率。（即将废止，替代指标为ma_node_npu_ai_core_util）。	百分比（Percent）	0～100%
	NPU显存使用率	ma_node_npu_memory_util	该指标用于统计测量对象已使用的NPU显存占NPU存储容量的百分比。（即将废止，snt3系列替代指标为ma_node_npu_ddr_memory_util，snt9系列替代指标为ma_node_npu_hbm_util）。	百分比（Percent）	0～100%

分类	名称	指标	指标含义	单位	取值范围
	NPU显存使用量	ma_node_npu_memory_used_megabytes	该指标用于统计测量对象已使用的NPU显存。（即将废止，snt3系列替代指标为ma_node_npu_ddr_memory_usage_bytes，snt9系列替代指标为ma_node_npu_hbm_usage_bytes）	兆字节（Megabytes）	≥0
	NPU显存容量	ma_node_npu_memory_total_megabytes	该指标用于统计测量对象的NPU显存容量。（即将废止，snt3系列替代指标为ma_node_npu_ddr_memory_bytes，snt9系列替代指标为ma_node_npu_hbm_bytes）。	兆字节（Megabytes）	>0
	AI处理器错误码	ma_node_npu_ai_core_error_code	昇腾系列AI处理器错误码	-	-
	AI处理器健康状态	ma_node_npu_ai_core_health_status	昇腾系列AI处理器健康状态	-	• 1：健康 • 0：不健康
	AI处理器功耗	ma_node_npu_ai_core_power_usage_watts	昇腾系列AI处理器功耗（snt9和snt3为处理器功耗，snt3P为板卡功耗	瓦特（W）	>0

分类	名称	指标	指标含义	单位	取值范围
	AI处理器温度	ma_node_npu_ai_core_temperature_celsius	昇腾系列AI处理器温度	摄氏度（℃）	自然数
	AI处理器风扇转速	ma_node_npu_fan_speed_rpm	昇腾系列AI处理器的风扇转速	转/每分（RPM）	自然数
	AI处理器AI CORE利用率	ma_node_npu_ai_core_util	昇腾系列AI处理器AI Core利用率	百分比（Percent）	0～100%
	AI处理器AI CORE时钟频率	ma_node_npu_ai_core_frequency_hertz	昇腾系列AI处理器AI Core时钟频率	赫兹（Hz）	>0
	AI处理器电压	ma_node_npu_ai_core_voltage_volts	昇腾系列AI处理器电压	伏特（V）	自然数
	AI处理器DDR内存总量	ma_node_npu_ddr_memory_bytes	昇腾系列AI处理器DDR内存总量	字节（Byte）	>0
	AI处理器DDR内存使用量	ma_node_npu_ddr_memory_usage_bytes	昇腾系列AI处理器DDR内存使用量	字节（Byte）	>0
	AI处理器DDR内存利用率	ma_node_npu_ddr_memory_util	昇腾系列AI处理器DDR内存利用率	百分比（Percent）	0～100%
	AI处理器HBM内存总量	ma_node_npu_hbm_bytes	昇腾系列AI处理器HBM总内存（昇腾snt9 AI处理器专属）	字节（Byte）	>0
	AI处理器HBM内存使用量	ma_node_npu_hbm_usage_bytes	昇腾系列AI处理器HBM内存使用量（昇腾snt9 AI处理器专属）	字节（Byte）	>0

分类	名称	指标	指标含义	单位	取值范围
	AI处理器HBM内存利用率	ma_node_npu_hbm_util	昇腾系列AI处理器HBM内存利用率（昇腾snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存带宽利用率	ma_node_npu_hbm_bandwidth_util	昇腾系列AI处理器HBM内存带宽利用率（昇腾snt9 AI处理器专属）	百分比（Percent）	0～100%
	AI处理器HBM内存时钟频率	ma_node_npu_hbm_frequency_hertz	昇腾系列AI处理器HBM内存时钟频率（昇腾snt9 AI处理器专属）	赫兹（Hz）	>0
	AI处理器HBM内存温度	ma_node_npu_hbm_temperature_celsius	昇腾系列AI处理器HBM内存温度（昇腾snt9 AI处理器专属）	摄氏度（℃）	自然数
	AI处理器AI CPU利用率	ma_node_npu_ai_cpu_util	昇腾系列AI处理器AI CPU利用率	百分比（Percent）	0～100%
	AI处理器控制CPU利用率	ma_node_npu_ctrl_cpu_util	昇腾系列AI处理器控制CPU利用率	百分比（Percent）	0～100%
infiniband或RoCE网络	网卡接收数据总量	ma_node_infiniband_port_received_data_bytes_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), received on all VLs from the port.	(counting in double words, 32 bits	≥0

分类	名称	指标	指标含义	单位	取值范围
	网卡发送数据总量	ma_node_infiniband_port_transmitted_data_bytes_total	The total number of data octets, divided by 4, (counting in double words, 32 bits), transmitted on all VLs from the port.	(counting in double words, 32 bits	≥0

分类	名称	指标	指标含义	单位	取值范围
NFS挂载状态	NFS检索文件属性操作拥塞时间	ma_node_mountstats_getattr_backlog_wait	Getattr is an NFS operation that retrieves the attributes of a file or directory, such as size, permissions, owner, etc. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS检索文件属性操作往返时间	ma_node_mountstats_getattr_rtt	Getattr is an NFS operation that retrieves the attributes of a file or directory, such as size, permissions, owner, etc. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS检查文件权限操作拥塞时间	ma_node_mountstats_access_backlog_wait	Access is an NFS operation that checks the access permissions of a file or directory for a given user. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS检查文件权限操作往返时间	ma_node_mountstats_access_rtt	Access is an NFS operation that checks the access permissions of a file or directory for a given user. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS解析文件句柄操作拥塞时间	ma_node_mountstats_lookup_backlog_wait	Lookup is an NFS operation that resolves a file name in a directory to a file handle. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS解析文件句柄操作往返时间	ma_node_mountstats_lookup_rtt	Lookup is an NFS operation that resolves a file name in a directory to a file handle. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS读文件操作拥塞时间	ma_node_mountstats_read_backlog_wait	Read is an NFS operation that reads data from a file. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performanc e and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS读文件操作往返时间	ma_node_mountstats_read_rtt	Read is an NFS operation that reads data from a file. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS写文件操作拥塞时间	ma_node_mountstats_write_backlog_wait	Write is an NFS operation that writes data to a file. Backlog wait is the time that the NFS requests have to wait in the backlog queue before being sent to the NFS server. It indicates the congestion on the NFS client side. A high backlog wait can cause poor NFS performance and slow system response times.	ms	≥0

分类	名称	指标	指标含义	单位	取值范围
	NFS写文件操作往返时间	ma_node_mountstats_write_rtt	Write is an NFS operation that writes data to a file. RTT stands for Round Trip Time and it is the time from when the kernel RPC client sends the RPC request to the time it receives the reply34. RTT includes network transit time and server execution time. RTT is a good measurement for NFS latency. A high RTT can indicate network or server issues.	ms	≥0

表 5-4 Label 名字栏

指标对象	Label名字	Label描述
容器级别指标	pod_name	容器所属pod的名字。
	pod_id	容器所属pod的ID。
	node_ip	容器所属的节点IP值。
	container_id	容器ID。

指标对象	Label名字	Label描述
	cluster_id	集群ID。
	cluster_name	集群名称。
	container_name	容器名称。
	namespace	是用户创建的POD所在的命名空间。
	app_kind	取自首个ownerReferences的kind字段。
	app_id	取自首个ownerReferences的uid字段。
	app_name	取自首个ownerReferences的name字段。
	npu_id	昇腾卡的ID信息，比如davinci0（即将废止）。
	device_id	昇腾系列AI处理器的Physical ID。
	device_type	昇腾系列AI处理器类型。
	pool_id	物理专属池对应的资源池id。
	pool_name	物理专属池对应的资源池name。
	gpu_uuid	容器使用的GPU的UUID。
	gpu_index	容器使用的GPU的索引。
	gpu_type	容器使用的GPU的型号。
node级别指标	cluster_id	该node所属CCE集群的ID。
	node_ip	节点的IP。
	host_name	节点的主机名。
	pool_id	物理专属池对应的资源池ID。
	project_id	物理专属池的用户的project id。
	npu_id	昇腾卡的ID信息，比如davinci0（即将废止）。
	device_id	昇腾系列AI处理器的Physical ID。
	device_type	昇腾系列AI处理器类型。
	gpu_uuid	节点上GPU的UUID。
	gpu_index	节点上GPU的索引。
	gpu_type	节点上GPU的型号。
	device_name	infiniband或RoCE网络网卡的设备名称。
	port	IB网卡的端口号。
	physical_state	IB网卡每个端口的状态。

指标对象	Label名字	Label描述
	firmware_version	IB网卡的固件版本。
	filesystem	NFS挂载的文件系统。
	mount_point	NFS的挂载点。
Diagnos	cluster_id	GPU所在节点所属的CCE集群ID。
	node_ip	GPU所在节点的IP。
	pool_id	物理专属池对应的资源池ID。
	project_id	物理专属池的用户的project id。
	gpu_uuid	GPU的UUID。
	gpu_index	节点上GPU的索引。
	gpu_type	节点上GPU的型号。
	device_name	infiniband或RoCE网络网卡的设备名称。
	port	IB网卡的端口号。
	physical_state	IB网卡每个端口的状态。
	firmware_version	IB网卡的固件版本。

监控自定义指标

用户有一些自定义的指标数据需要保存到AOM，ModelArts提供了命令方式将用户的自定义指标上报保存到AOM。

约束与限制

- ModelArts以10秒/次的频率调用自定义配置中提供的命令或http接口获取指标数据。
- 自定义配置中提供的命令或http接口返回的指标数据文本不能大于8KB。

命令方式采集自定义指标数据

用于创建自定义指标采集POD的YAML文件示例如下。

```
apiVersion: v1
kind: Pod
metadata:
  name: my-task
  annotations:
    ei.huaweicloud.com/metrics: '{"customMetrics":[{"containerName":"my-task","exec":{"command":["cat","/
metrics/task.prom"]}}]}' # ModelArts从哪个容器以及使用哪个命令获取指标数据，请根据实际情况替换
containerName参数和command参数
spec:
  containers:
  - name: my-task
    image: my-task-image:latest # 替换为实际使用的镜像
```


备注：业务负载和自定义指标采集可以共用一个容器，也可以由SideCar容器采集指标数据，然后将自定义指标采集容器指定到SideCar容器，这样可以不占用业务负载容器的资源。

自定义指标数据格式

自定义指标数据的格式必须是符合open metrics规范的文本，即每个指标的格式应为：

```
<指标名称>{<标签名称>=<标签值>,...} <采样值> [毫秒时戳]
```

举例如下（#开头为注释，非必需）：

```
# HELP http_requests_total The total number of HTTP requests.
# TYPE http_requests_total gauge
html_http_requests_total{method="post",code="200"} 1656 1686660980680
html_http_requests_total{method="post",code="400"} 2 1686660980681
```

图 5-17 自定义指标数据结果

5.7.2 使用 Prometheus 查看 Lite Cluster 监控指标

背景信息

Prometheus是一款开源监控工具，ModelArts支持Exporter功能，方便用户使用Prometheus等第三方监控系统获取ModelArts采集到的指标数据。

使用说明

- 该功能为白名单功能，如需要使用，请联系提交工单开通此功能。
- 开通此功能后，兼容Prometheus指标格式的第三方组件可通过API http://<节点IP>:<端口号>/metrics获取ModelArts采集到的指标数据。
- 开通前需要确认使用的端口号，端口号可选取10120~10139范围内的任一端口号，请确认选取的端口号在各个节点上都没有被其他应用占用。

Kubernetes 下 Prometheus 对接 ModelArts

1. 使用kubectl连接集群，详细操作请参考**通过kubectl连接集群**。
2. 配置Kubernetes的访问授权。

   使用任意文本编辑器创建prometheus-rbac-setup.yml，YAML文件内容如下：

   说明

   该YAML用于定义Prometheus要用到的角色（ClusterRole），为该角色赋予相应的访问权限。同时创建Prometheus所使用的账号（ServiceAccount），将账号与角色进行绑定（ClusterRoleBinding）。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
```

```
  resources:
  - pods
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
  namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus
  namespace: default
```

3. 执行如下命令创建RBAC对应的各个资源。

```
$ kubectl create -f prometheus-rbac-setup.yml
clusterrole "prometheus" created
serviceaccount "prometheus" created
clusterrolebinding "prometheus" created
```

4. 使用任意文本编辑器创建prometheus-config.yml，内容如下。该YAML用于管理Prometheus的配置，部署Prometheus时通过文件系统挂载的方式，容器可以使用这些配置。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
data:
  prometheus.yml: |
    global:
      scrape_interval: 10s
    scrape_configs:
    - job_name: 'modelarts'
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      kubernetes_sd_configs:
      - role: pod
      relabel_configs:
        - source_labels: [__meta_kubernetes_pod_name]    # 指定从maos-node-agent-字符串开头的POD
收集指标数据
          action: keep
          regex: ^maos-node-agent-.+
        - source_labels: [__address__]    # 指定获取指标数据的地址和端口号为__address__:9390，
__address__为POD的IP地址，也是节点IP地址
          action: replace
          regex: '(.*)'
          target_label: __address__
          replacement: "${1}:10120"
```

5. 执行如下命令创建ConfigMap资源。

```
$ kubectl create -f prometheus-config.yml
configmap "prometheus-config" created
```

6. 使用任意文本编辑器创建prometheus-deployment.yml，内容如下。

说明

该YAML用于部署Prometheus。将上面创建的账号（ServiceAccount）权限赋予了Prometheus，同时将上面创建的ConfigMap资源以文件系统的方式挂载到了prometheus容器的“/etc/prometheus”目录，并且通过--config.file=/etc/prometheus/prometheus.yml参数指定了“/bin/prometheus”使用该配置文件。

```
apiVersion: v1
kind: "Service"
metadata:
  name: prometheus
  labels:
    name: prometheus
spec:
  ports:
  - name: prometheus
    protocol: TCP
    port: 9090
    targetPort: 9090
  selector:
    app: prometheus
  type: NodePort
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    name: prometheus
  name: prometheus
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: prometheus
    spec:
      hostNetwork: true
      serviceAccountName: prometheus
      serviceAccount: prometheus
      containers:
      - name: prometheus
        image: prom/prometheus:latest
        imagePullPolicy: IfNotPresent
        command:
        - "/bin/prometheus"
        args:
        - "--config.file=/etc/prometheus/prometheus.yml"
        ports:
        - containerPort: 9090
          protocol: TCP
        volumeMounts:
        - mountPath: "/etc/prometheus"
          name: prometheus-config
      volumes:
      - name: prometheus-config
        configMap:
          name: prometheus-config
```

7. 执行如下命令创建Prometheus实例，并查看创建情况：

```
$ kubectl create -f prometheus-deployment.yml
service "prometheus" created
deployment "prometheus" created

$ kubectl get pods
NAME                          READY     STATUS        RESTARTS   AGE
prometheus-55f655696d-wjqcl       1/1       Running       0          5s

$ kubectl get svc
NAME           TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
```



```
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP          131d
prometheus      NodePort    10.101.255.236   <none>        9090:32584/TCP   42s
```

查看 Prometheus 采集的指标数据

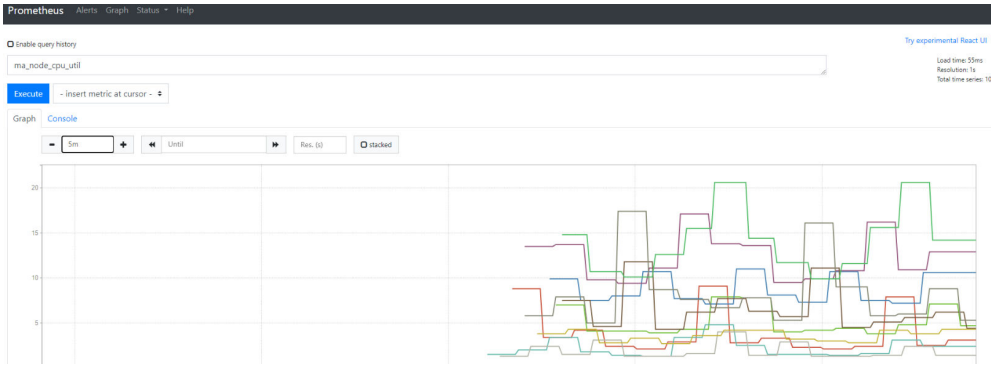
1. 在CCE页面为Prometheus所在节点绑定弹性公网IP，并打开节点的安全组配置，添加入方向规则，允许外部访问9090端口。

   说明

   如果使用Grafana对接Prometheus制作报表，可以将Grafana部署在集群内，这里不需要对Prometheus绑定公网IP和配置安全组，只需要对Grafana绑定公网IP和配置安全组即可。

2. 在浏览器地址栏输入http://<弹性公网IP>:9090，即可打开Prometheus监控浏览页面。单击Graph菜单，在输入框输入任意一个指标名称即可看到Prometheus收集到的指标数据：

5.8 释放 Lite Cluster 资源

针对不再使用的Lite Cluster资源，可以释放资源，停止计费相关介绍请见**停止计费**。

说明

专属资源池资源释放后不可恢复，请谨慎操作。

删除按需计费的 Lite Cluster 资源

步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“AI专属资源池 > 弹性集群 Cluster”，进入“弹性集群 Cluster”列表页面。

步骤3 在弹性集群列表中，单击操作列的“更多 > 删除”。

步骤4 在弹出的确认对话框中，输入“DELETE”，单击“确定”，删除资源池。

----结束

退订包年/包月的 Lite Cluster 资源

步骤1 登录ModelArts管理控制台。

步骤2 在左侧导航栏中，选择“AI专属资源池 > 弹性集群 Cluster”，进入“弹性集群 Cluster”列表页面。

步骤3 在弹性集群列表中，单击操作列的“更多 > 退订”，跳转至“退订资源”页面。

步骤4 根据界面提示，确认需要退订的资源，并选择退订原因。

步骤5 确认退订信息无误后，勾选“资源退订后……”提示信息。

步骤6 单击“退订”，再次根据界面信息确认要退订的资源。

步骤7 再次单击“退订”，完成包年/包月资源的退订操作。

----结束