华为云 UCS

用户指南

文档版本 01

发布日期 2023-08-31

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

漏洞处理流程

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

目 录

1 UCS 集群

1.1 UCS 集群概述

UCS服务支持跨云、跨地域的集群统一接入、统一管理，支持接入如下几种集群类型：

- **华为云集群**：包括华为云CCE集群和CCE Turbo集群。
- **本地集群**：由UCS提供的、运行在您的数据中心基础设施之上的Kubernetes集群，如UCS on Bare Metal、UCS on VMware。
- **附着集群**：满足CNCF（Cloud Native Computing Foundation）标准的第三方Kubernetes集群，如AWS（EKS）、GCP（GKE）以及自建的Kubernetes集群。
- **多云集群**：由UCS提供的、运行在第三方云服务供应商（如AWS）基础设施之上的Kubernetes集群，如UCS on AWS、UCS on Azure。

⚠ 注意

如果您接入的集群中包含超大容量的节点，并且希望其不被统计在UCS控制台集群列表上的CPU和内存分配率指标中，那么需要为这个节点打上“type:virtual-kubelet”标签，以便您准确识别集群资源分配情况。为集群节点打标签请参照**为节点添加标签/污点**。

1.2 华为云集群

UCS支持一键注册华为云集群（CCE Standard集群、CCE Turbo集群），注册完成后即可实现集群的统一管理。

约束与限制

- 仅**华为云账号**或具备**UCS FullAccess**权限的用户可进行集群注册的操作。
- 若集群地域位于境外，应确保您的行为符合所适用的法律法规要求。
- 请确保注册的Kubernetes集群版本在1.19至1.28之间。

前提条件

已创建一个准备接入UCS的CCE Standard集群或CCE Turbo集群，并且集群状态为“运行中”。

操作步骤

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击华为云集群选项卡中的“注册集群”按钮。

步骤3 勾选需要注册的CCE集群，并选择一个舰队，单击“确定”。

注册集群时若不选择舰队，集群注册成功后将显示在“未加入舰队的集群”页签下，您可以随后再将其添加至舰队中，具体操作请参见**管理未加入舰队的集群**。

说明

不支持在注册集群阶段选择已开通集群联邦能力的舰队，如果一定要加入这个舰队，请在集群注册成功后，再添加到该舰队中。关于集群联邦的介绍，请参见**开通集群联邦**章节。

----结束

1.3 本地集群

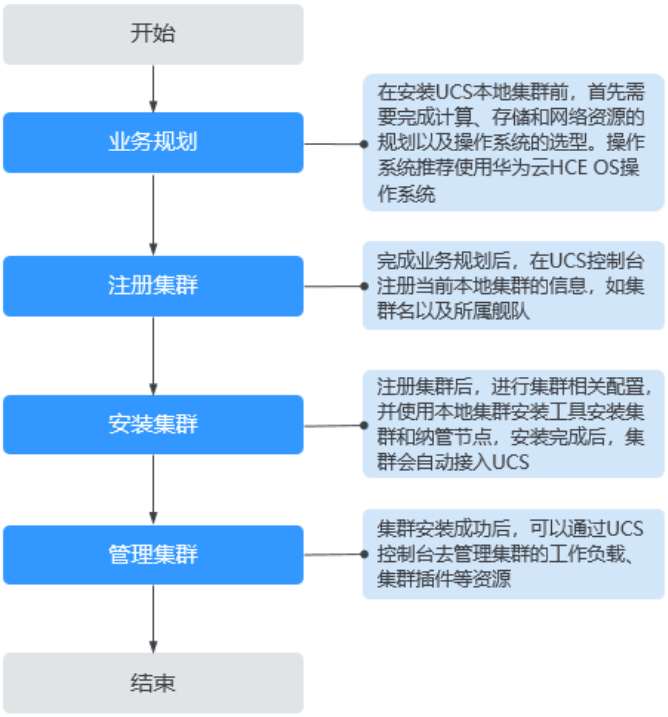
1.3.1 本地集群概述

本地集群是由UCS提供的、运行在您的数据中心基础设施之上的Kubernetes集群。您只需要准备好相关物理资源，安装Kubernetes软件以及接入UCS的过程完全交给华为云来处理。

本地集群兼容多种底层基础设施，支持部署在裸金属服务器和VMware等虚拟化Iaas上，支持容器网络与底层网络打通，支持利用CSI对接多种底层存储服务（如VMware Vsphere等），提供持久化存储能力。

本地集群管理流程如**图 本地集群管理流程**所示。

图 1-1 本地集群管理流程



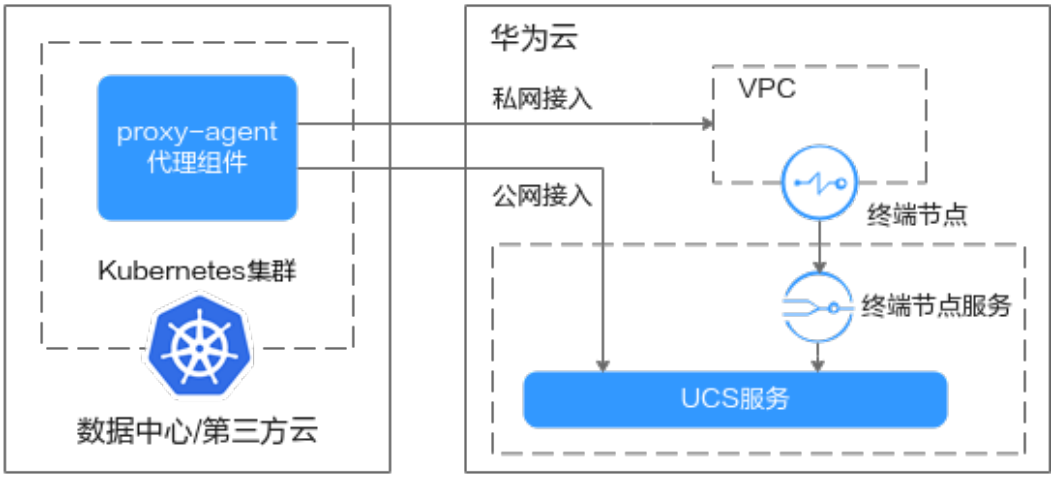
接入网络模式

UCS使用集群网络代理的连接方式，如图1-2所示。您无需在防火墙上启用任何入方向端口，仅通过集群代理程序的方式即可在出方向与UCS服务建立会话。

本地集群接入网络的方法有两种，具有不同的优点：

- 公网接入：具有弹性灵活、成本低、易接入的优点。
- 私网接入：可获得更加高速、低时延、稳定安全的体验。

图 1-2 集群接入原理



1.3.2 安装本地集群的业务规划

1.3.2.1 基础软件规划

本地集群节点的操作系统、内核版本等基础软件规划需要符合表1-1中的要求。

表 1-1 基础软件规划

系统架构	系统类型	网络模型	操作系统版本	内核版本限制
x86	Ubuntu 22.04	Cilium	检查命令：**cat /etc/lsb-release** DISTRIB_DESCRIPTION="Ubuntu 22.04.1 LTS"	检查命令：**uname -r** 5.10.0-46-generic及以上
	Redhat 8.6	Cilium	检查命令：**cat /etc/os-release** Red Hat Enterprise Linux release 8.6 (Ootpa)	检查命令：**uname -r** 4.18.0-372.9.1.el8.x86_64
	HCE OS 2.0	Cilium	检查命令：**cat /etc/os-release** Huawei Cloud EulerOS release 2.0 (West Lake)	检查命令：**uname -r** 5.10.0-60.18.0.50.r865_35.hce2.x86_64

说明

- Cilium是一种网络插件，支持BGP、eBPF等网络协议，详细了解Cilium请参见Cilium官方文档。
- HCE OS 2.0（Huawei Cloud EulerOS）是基于华为开源社区openEuler构建的Linux操作系统，提供云原生、高性能、安全稳定的执行环境来开发和运行应用程序，支持X86、ARM64等硬件架构。如需安装HCE OS 2.0，请提交工单，联系技术支持人员。

1.3.2.2 数据规划

防火墙规划

防火墙的规划需符合表1-2中要求。

表 1-2 防火墙规划

源设备	源IP	源端口	目的设备	目的IP	目的端口（侦听）	协议	端口说明	侦听端口是否可更改	认证方式	加密方式
ucsctl执行机	源设备所在节点IP	ALL	所有节点	目的设备所在节点IP	22	TCP	SSH	否	证书/用户名密码	TLSv1.2
所有节点	源设备所在节点IP	ALL	NTP server	目的设备所在节点IP	123	UDP	ntp	否	无	无
所有节点	源设备所在节点IP	ALL	DNS server	目的设备所在节点IP	53	UDP	dns	否	无	无
所有节点	源设备所在节点IP	ALL	自建APT源	目的设备所在节点IP	80/443	TCP	http	否	无	无
所有节点	源设备所在节点IP	ALL	集群负载均衡/VIP	目的设备所在节点IP	5443	TCP	kube-apiserver	否	https+证书	TLSv1.2
所有节点	源设备所在节点IP	1024-65535	所有节点	目的设备所在节点IP	1024-65535	ALL	无	否	无	无

源设备	源IP	源端口	目的设备	目的IP	目的端口（侦听）	协议	端口说明	侦听端口是否可更改	认证方式	加密方式
所有节点	源设备所在节点IP	ALL	所有节点	目的设备所在节点IP	8472	UDP	vxlan端口	否	无	无
需要访问ingress的节点	源设备所在节点IP	ALL	网络节点	目的设备所在节点IP	80/443/按需指定端口	TCP	http	否	https+证书	TLSv1.2
所有节点	源设备所在节点IP	ALL	3台master节点	目的设备所在节点IP	5444	TCP	kube-apiserver	否	https+证书	TLSv1.2
ucsctl执行机	源设备所在节点IP	ALL	华为云OBS服务	OBS终端节点所在IP	443	TCP	http	否	https+证书	TLSv1.2
3台master节点	源设备所在节点IP	ALL	华为云UCS服务	124.70.21.61 proxyurl.ucs.myhuaweicloud.com	30123	TCP	grpc	否	https+证书	TLSv1.2
3台master节点	源设备所在节点IP	ALL	华为云IAM服务	外部访问IAM服务的域名地址	443	TCP	http	否	https+证书	TLSv1.2

源设备	源IP	源端口	目的设备	目的IP	目的端口（侦听）	协议	端口说明	侦听端口是否可更改	认证方式	加密方式
所有节点	源设备所在节点IP	All	华为云SWR服务	SWR终端节点所在IP	443	TCP	http	否	https+证书	TLSv1.2
所有节点	源设备所在节点IP	ALL	Ubuntu官方源/国内代理源	按需配置	80/443	TCP	http	否	无	无
监控节点	源设备所在节点IP	ALL	华为云AOM	域名对应IP地址	443	TCP	http	否	https+证书	TLSv1.2
监控节点	源设备所在节点IP	ALL	华为云LTS	域名对应IP地址	443	TCP	http	否	https+证书	TLSv1.2

资源规格

UCS所安装的本地集群为HA版，适用于商用场景，以满足容灾高可用需求。商用版资源规格如下所述：

表 1-3 容器平台基础能力资源规格

节点类型	数量	CPU (Cores)	Mem (GiB)	Disk (G)-系统盘	Disk (G)-高性能盘	Disk (G)-数据盘	备注
集群管理节点	3	8	16	100	50	300	需要提供一个VIP用于高可用。

节点类型	数量	CPU (Cores)	Mem (GiB)	Disk (G)-系统盘	Disk (G)-高性能盘	Disk (G)-数据盘	备注
集群计算节点	按需	2	4	40	-	100	数量按需可扩展。

表 1-4 容器智能分析节点资源规格

节点类型	CPU (Cores)	Mem (GiB)
监控prometheus	Requests：1 Limits：4	Requests：2 Limits：12
事件log-agent	Requests：0.5 Limits：3	Requests：1.5 Limits：2.5

表 1-5 云原生服务中心计算节点资源规格

类型	数量	CPU (Cores)	Mem (GiB)	Disk (G)-系统盘	Disk (G)-高性能盘	Disk (G)-数据盘
operator-chef	1	Requests：0.5 Limits：2	Requests：0.5 Limits：2	不涉及	不涉及	10（日志）
helm-operator	1	Requests：0.3 Limits：1.5	Requests：0.3 Limits：1.5	不涉及	不涉及	10（日志）
ops-operator	1	Requests：0.3 Limits：1.5	Requests：0.3 Limits：1.5	不涉及	不涉及	10（日志）

外部依赖

表 1-6 外部依赖项

依赖项	功能解释
DNS服务器	DNS服务器需要能够解析OBS、SWR、IAM、DNS等服务的域名，这些服务的域名请参见**地区及终端节点**。 公网接入情况下，节点可自动识别默认DNS配置，需提前将DNS服务器上游设置为公网DNS，接下来无需再手动进行DNS服务器配置。 私网接入情况下，节点无法识别默认DNS配置，因此需提前配置VPCEP解析能力，详情请参考安装前准备。若您还未搭建DNS服务器，可参考**DNS**进行搭建。
apt源	确保有可用的apt源，因为在本地集群执行纳管节点时（纳管节点是指待添加到本地集群管理的服务器），部分安装组件如ntp等，需要从apt源中获取依赖包。
NTP服务器	可选，用于保证集群各节点时间同步，如果使用，推荐用户提供外置NTP服务器。

磁盘挂卷

表 1-7 磁盘挂卷

节点类型	磁盘挂载点	可用大小（GB）	用途
集群管理节点	/var/lib/containerd	50	存放containerd镜像目录
	/run/containerd	30	containerd运行时目录
	/var/paas/run	50	etcd数据目录（推荐使用ssd盘）
	/var/paas/sys/log	20	存放日志目录
	/mnt/paas	40	容器运行挂载目录
	/tmp	20	临时文件目录
集群计算节点	/var/lib/containerd	100	存放containerd镜像目录
	/run/containerd	50	containerd运行时目录
	/mnt/paas	50	容器运行挂载目录

负载均衡规划

本地集群的HA版本部署在多台管理面节点实现容灾高可用，需要提供统一地址，供集群计算节点和其他外部服务访问。本地集群支持VIP或对接外部负载均衡器两种方式，请根据实际需求选择，**二选一**即可。

- VIP规划

  要求用户设备的底层规划一个空闲IP为VIP，能够针对三台Master管理面节点规划一个VIP。该VIP会在安装过程由本地集群服务随机绑定到其中一台Master节点，并且当节点或者节点上服务异常时会自动切换到其他节点，以保证集群高可用。

  表 1-8 IP 规划

IP类型	地址	用途
VIP（虚IP）	10.10.11.10（示例）	此处仅为示例，请以实际情况规划此IP，用于实现高可用。

- ELB规划

  如果用户已有外部负载均衡器（ELB），本地集群可以对接外部负载均衡器实现集群高可用。具体配置如下：

  - TCP监听器：3个，端口分别为80、443、5443。
  - TCP后端服务器组：3个，分别对应三台Master节点的80、443、5444端口。

    监听器关联的TCP后端服务器组关系，请参见**表1-9**。

  表 1-9 监听器与后端服务器组

监听器（协议/端口）	后端服务器组名称	后端服务器组节点映射和端口		
TCP/80	ingress-http	master-01-IP:80	master-02-IP:80	master-03-IP:80
TCP/443	ingress-https	master-01-IP:443	master-02-IP:443	master-03-IP:443
TCP/5443	kube-apiserver	master-01-IP:5444	master-02-IP:5444	master-03-IP:5444

说明

- 不同ELB产品，外接ELB配置界面略有差异。请根据实际界面，配置上述对应关系。
- 安装本地集群前，请提前将**TCP监听器**与**TCP后端服务器组**的映射关系配置到外部ELB上，并确保外部ELB功能可用。
- 负载均衡器可以将来自所有Node节点（包括Master节点）上的kubelet等进程的访问流量分发到三台Master主机上，并支持自动检测并隔离不可用的进程，从而提高业务的服务能力和可用性。用户可以使用云厂商提供的负载均衡器或相关的LB硬件设备，还可以通过Keepalived和Haproxy的方式实现多个Master节点的高可用部署。
- 【**推荐配置**】ELB对上述监听端口**开启源地址透传**，并**关闭环路检查**能力。若无法单独关闭环路检查，则需关闭源地址透传。判断是否存在环路检查的方法如下：
  1. 在可被外网访问的服务器A上创建一个http服务，将默认监听端口80修改为88，并添加测试的index.html文件。
     ```
     yum install -y httpd
     sed -i 's/Listen 80/Listen 88/g' /etc/httpd/conf/httpd.conf
     echo "This is a test page" > /var/www/html/index.html
     systemctl start httpd
     ```
     在浏览器中访问**${服务器A_IP}:88**，页面显示“This is a test page”。
  2. ELB配置监听端口如30088端口，转发至服务器A的88端口，并开启源地址透传。
  3. 在服务器A上通过ELB内网ip访问该http服务：
     ```
     curl -v ${ELB_IP}:30088
     ```
     查看http响应状态码是否为200，若非200则说明存在环路检查。

用户规划

表 1-10 用户规划

用户	用户组	用户ID	用户组ID	密码	用途
root	root	0	0	-	UCS本地集群安装时使用的默认用户，也可以指定其他用户来安装本地集群。安装用户要求满足如下条件： ● 规划的所有集群管理节点密码一致。 ● 用户具有完全的root用户权限。 **说明** 安装完成后用户可以自行修改该用户密码或限制该用户root权限。
paas	paas	10000	10000	-	UCS本地集群服务进程的运行用户、用户组，在安装过程中创建。该用户和用户组对应名称为paas:paas，用户ID和用户组ID对应为10000:10000，因此安装前需要保证用户名、用户组名、用户ID和用户组ID不被占用。若发生冲突，需提前删除对应的用户或者用户组。

1.3.3 注册本地集群

本小节指导您将本地集群注册至UCS。

约束与限制

仅**华为云账号**且具备**UCS FullAccess**权限的用户可进行集群注册的操作。

前提条件

- 已在UCS控制台申请本地集群试用。
- UCS集群配额充足。
- 节点/tmp目录需要预留20GB空间。
- 根据**安装本地集群**确保待执行机检查项已满足。
- 准备一台执行机，要求与集群网络连通。

注册集群

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击本地集群选项卡中的“注册集群”按钮。

步骤3 参考**表1-11**填写待添加集群的基础信息，其中带“*”的参数为必填参数。

表 1-11 注册集群基础信息配置

参数	参数说明
集群名称*	输入集群的自定义名称，需以小写字母开头，由小写字母、数字、中划线（-）组成，且不能以中划线（-）结尾。
所属区域*	选择集群所在的区域。
集群标签	非必填项，以键值对的形式为集群添加标签，可以通过标签实现集群的分类。键值对可自定义，以字母或者数字开头和结尾，由字母、数字、连接符（-）、下划线（_）、点号（.）组成，且63个字符之内。
容器舰队	选择集群所属的舰队。 舰队用于权限精细化管理，一个集群只能加入一个舰队。若不选择舰队，集群注册成功后将显示在“未加入舰队的集群”页签下，后续还可以再添加至舰队中。 不支持在注册集群阶段选择已开通集群联邦能力的舰队，如果一定要加入这个舰队，请在集群注册成功后，再添加到该舰队中。关于集群联邦的介绍，请参见**开通集群联邦**章节。 如需新建舰队，请参见**管理容器舰队**。

步骤4 单击“确定”，集群注册成功后如**图1-3**所示，请在24小时内接入网络。您可选择集群的接入方式或单击右上角按钮查看详细的网络接入流程。

如您未在24小时内接入网络，将会导致集群注册失败，可单击右上角按钮重新注册集群。如果已经接入但数据未采集上来，请等待2分钟后刷新集群。

图 1-3 集群等待接入状态

----结束

1.3.4 安装本地集群

1.3.4.1 安装前检查

节点软件更新与升级要求

请关闭节点软件自动更新，请勿进行docker安装和containerd版本升级。禁用Ubuntu软件自动更新可参考**Ubuntu Enable Automatic Updates Unattended Upgrades**。

检查 OS 系统语言

安装本地集群前，请先检查OS系统语言，确保系统语言为英文。

检查节点 apt 源（Ubuntu）

须知

检查节点apt源操作适用于操作系统为Ubuntu的节点，若您的节点操作系统为HCE或Redhat，请参见**检查节点yum源（HCE、Redhat）**进行检查。

在本地集群执行纳管节点操作时（纳管节点是指待添加到本地集群管理的服务器），部分安装组件如ntpdate等，需要从apt源中获取依赖包。故纳管节点前，请确保节点上apt源是可用的，若不可用，请执行如下操作。

步骤1 以安装用户（默认为root）登录待安装的集群管理节点。

步骤2 编辑“/etc/apt/sources.list”。

具体信息以实际规划的apt源服务器地址为准。

步骤3 保存文件，执行如下命令。

sudo apt-get update

步骤4 （可选）若有多个管理节点，如HA版本，请分别登录到规划的管理节点执行上述操作。

----结束

检查节点 yum 源（HCE、Redhat）

在本地集群执行纳管节点操作时，部分安装组件如ntpdate等，需要从yum源中获取依赖包。故纳管节点前，请确保节点上yum源是可用的，若不可用，请执行如下操作。

步骤1 以安装用户（默认为root）登录待安装的集群管理节点。

步骤2 修改/etc/yum.repos.d/目录下的软件源配置文件。

具体信息以实际规划的yum源服务器地址为准。

步骤3 保存文件，执行如下命令。

sudo yum-get update

步骤4 （可选）若有多个管理节点，如HA版本，请分别登录到规划的管理节点执行上述操作。

----结束

主机最小化安装要求

- 不使用的软件包不允许存在系统中

  遵循最小化安装原则，只安装业务需要的软件包与服务组件。减少系统漏洞，降低系统遭受攻击风险。

- 用于生产环境的系统中不允许保留开发和编译工具

  系统中不允许存在如下开发工具和编译工具：

  ```
  'cpp' (/usr/bin/cpp)
  'gcc' (/usr/bin/gcc)
  'ld' (/usr/bin/ld)
  'lex' (/usr/bin/lex)
  'rpcgen' (/usr/bin/rpcgen)
  ```

  如果产品的生产环境，比如在部署或运行过程中需要python、lua等解释器，则可以保留解释器运行环境，否则不允许保留。

  ```
  'python' (/usr/bin/python)
  'lua' (/usr/bin/lua)
  ```

  同样情形适用于perl解释器，Suse系统的一些管理程序依赖perl解释器，在这种情形下，则可保留perl解释器，否则需要去除。

  ```
  perl (/usr/bin/perl)
  ```

- 操作系统中不允许安装显示安全策略的工具

  防止系统的安全信息泄露。依照业务需求，预安装的安全加固工具，限制其文件的所有者为root，并且仅root具有执行权限。

- 操作系统中不允许存在网络嗅探类的工具

  tcpdump、ethereal等嗅探工具不允许出现在系统上，防止被恶意使用。

- 在不需要Modem系统中不应默认安装Modem

  在不需要Modem系统中不应默认安装Modem，严格遵循系统最小化安装。

集群安装检查项

在安装本地集群前，您需要对节点进行一系列检查。

表格中的命令适用于HCE与Redhat操作系统，若您使用Ubuntu操作系统，请将命令中的“yum”修改为“apt”。

检查类型	检查名	检查内容	检查通过标准
集群检查	节点架构检查	所有安装的Master节点架构检查	所有安装节点架构必须一致
	节点主机名检查	所有安装的Master节点主机名检查	所有安装节点主机名必须不同
	节点时钟同步检查	所有安装的Master节点时钟同步状态检查	所有安装节点主机时间差异必须小于10秒
	VIP使用检查	VIP是否被其他节点占用	VIP必须处于空闲状态，检查依据22端口是否能被访问通
节点检查	节点语言检查	节点语言设置必须符合约束	节点语言设置符合en_US.UTF-8、en_GB.UTF-8任何一种
	节点操作系统检查	节点操作系统必须符合约束	节点操作系统为Ubuntu 22.04、Redhat 8.6、HCE 2.0任何一种
	系统命令检查	节点具备基础命令行工具	操作系统具备以下命令行工具：ifconfig、netstat、curl、systemctl、nohup、pidof、mount、uname、lsmod、swapoff、hwclock、ip、ntpdate（对接ntp场景具备）
	端口空闲检查	节点必装服务端口未被占用	操作系统以下端口未被占用： 4001、4002、4003、2380、2381、2382、4011、4012、4013、4005、4006、4007、5444、8080、10257、10259、4133、20100、9444、20102、9443、5443、4134、4194、10255、10248、10250、80、443、10256、10249、20101
	keepalived安装检查	keepalived未安装	执行**yum list --installed keepalived**列表无对应服务
	haproxy安装检查	haproxy未安装	执行**yum list --installed haproxy**列表无对应服务
	runit安装检查	runit未安装	执行**yum list --installed runit**列表无对应服务
	paas用户检查	节点paas用户处于可创建状态	节点paas用户不存在且id为10000的用户未被占用
	ntp服务检查	ntp服务处于可用状态	节点执行**ntpdate -u ${ntp_server}**可正常访问ntp

1.3.4.2 安装前准备（私网接入）

本小节指导您进行本地集群的安装前准备。在选择私网接入集群时，才需执行安装前准备。选择“公网接入”时，可直接执行安装及验证。

本地集群的安装前准备包括创建虚拟私有云并与线下IDC网络环境打通、创建终端节点并将其配置在VPC中的DNS服务器中。

部署网络环境

在UCS提供服务的区域中创建一个VPC，该VPC将用于后续安装终端节点，需保证该VPC与用户自有IDC网络环境打通。

VPC创建操作请参见**创建虚拟私有云和子网**，当前仅支持“亚太-新加坡”区域。

说明

该VPC子网网段不能与IDC中已使用的网络网段重叠，否则将无法接入集群。例如，IDC中已使用的VPC子网为192.168.1.0/24，那么华为云VPC中不能使用192.168.1.0/24这个子网。

将线下自有IDC的网络环境与华为云VPC打通，方法如下：

- 虚拟专用网络（VPN）方案：请参见**通过VPN连接云下数据中心与云上VPC**。
- **须知**

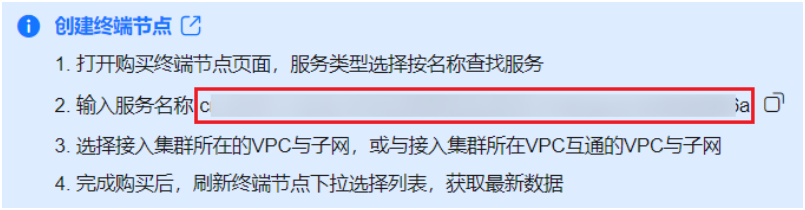
  云下、云上网络打通后，建议从本地数据中心服务器ping目标VPC下的华为云服务器私网IP，以验证网络是否成功连接。

购买终端节点（VPCEP）

步骤1 登录UCS控制台，单击待接入集群栏的“点击接入”进入集群接入界面，单击“私网接入”。

步骤2 查看“创建终端节点”中的服务名称，单击，记录服务名称。

图 1-4 创建终端节点



步骤3 登录VPC终端节点控制台，单击“购买终端节点”，创建连接不同服务的终端节点。

步骤4 选择终端节点的区域，单击“按名称查找服务”，输入**步骤2**中所记录的服务名称，并单击“验证”，创建UCS的终端节点。

图 1-5 按名称查找服务

步骤5 创建DNS、SWR、OBS的终端节点。

步骤6 选择**部署网络环境**中创建的虚拟私有云以及对应的子网。

步骤7 “节点IP”选择“自动分配”或“手动分配”均可。

步骤8 单击“立即购买”，规格确认无误后，单击“提交”。

步骤9 将创建的终端节点配置到所使用的DNS服务器中。单击创建出的VPCEP节点名称，记录“节点IP”，以便在IDC的DNS Server中增加华为云的DNS转发器。

----结束

配置 DNS 服务器

步骤1 **配置DNS转发**：在DNS服务器配置相应的DNS转发规则，将解析华为云内网域名的请求转发到DNS终端节点。以常见的DNS软件Bind为例：例如/etc/named.conf内，增加DNS转发器的配置，forwarders为DNS终端节点IP地址。

下示代码中xx.xx.xx.xx是DNS的终端节点IP。

```
options {
        forward only;
        forwarders{ xx.xx.xx.xx;};
};
```

步骤2 **增加静态DNS配置解析**：增加DNS静态配置，SWR与CIE实例地址。以北京四为例，如使用dnsmasq为例，在/etc/dnsmasq.conf中添加以下两个静态解析：

第一个静态解析如下，下示代码中xx.xx.xx.xx是SWR的终端节点IP。其中region应替换为服务应用区域的URL。

```
address=/swr.region.myhuaweicloud.com/xx.xx.xx.xx
```

第二个静态解析如下，下示代码中xx.xx.xx.xx是域名对应的IP地址，在开启集群监控后生成。其中region应替换为服务应用区域的URL。

```
address=/cia-{当前选择接入的VPCID前八位}{当前选择接入的VPC子网ID前八位}.region.myhuaweicloud.com/xx.xx.xx.xx
```

示例：address=/cia-9992be3cf3eace24.cn-north-4.myhuaweicloud.com/
172.16.0.81

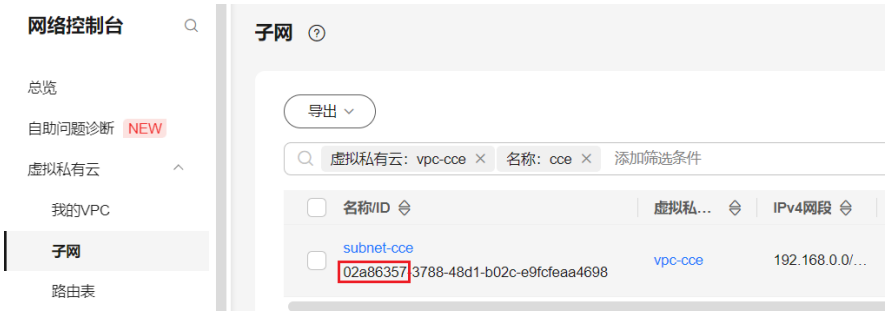
步骤3 生成域名。

SWR：address=/swr.cn-north-4.myhuaweicloud.com/{SWR VPC-EP}

CIA：域名的获取：如当前选择接入的VPC和子网如下（**如下截图vpc-cce仅是示例，实际VPC以UCS服务所在的VPC为准**）

图 1-6 VPC 的 ID 前 8 位

图 1-7 子网的 ID 前 8 位



最终域名拼接后是：cia-e52a5d7e02a86357.cn-north-4.myhuaweicloud.com

----结束

1.3.4.3 安装及验证

在UCS控制台成功添加集群后，集群状态将会显示为“等待安装并接入”，此时UCS并没有为集群安装Kubernetes软件以及打通与集群的网络连接，因此需要在集群中配置网络代理来接入网络并完成集群安装。

须知

请在添加集群后的24小时内接入网络，您可单击右上角按钮查看详细的网络接入流程。如您未在24小时内接入网络，将会导致集群接入失败，可单击右上角按钮重新接入集群。如果已经接入但状态未更新，请等待2分钟后刷新集群。

上传配置文件

步骤1 登录UCS控制台，单击待接入集群栏的“点击接入”进入集群接入界面，可选择“公网接入”和“私网接入”。

步骤2 选择接入方式并下载代理配置文件。

若选择“公网接入”，直接在界面单击“下载文件”，下载本地集群代理配置文件“agent-[集群名称].yaml”。

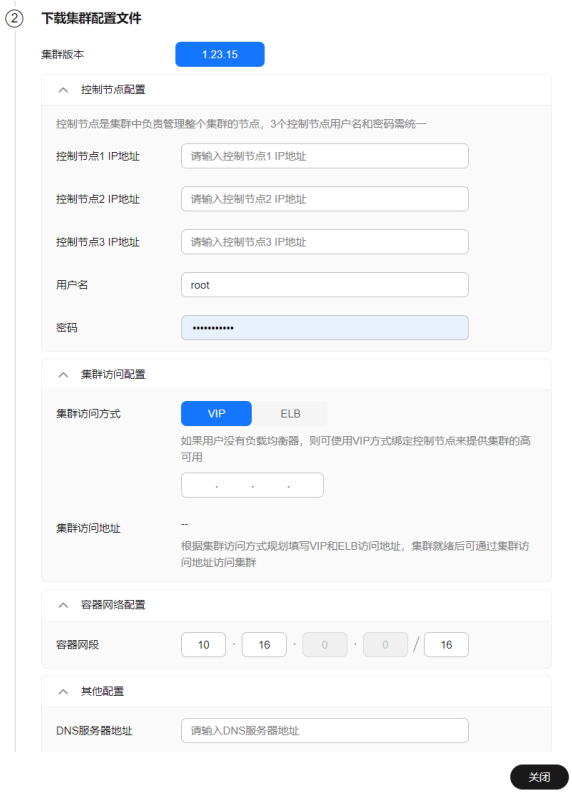
若选择“私网接入”，需要先选择项目，再选择**安装前准备（私网接入）**中创建的终端节点，然后单击“下载文件”，下载本地集群代理配置文件“agent-[集群名称].yaml”。

说明

集群代理配置文件存在私有秘钥信息，每个集群仅能下载一次，请您妥善保管。

步骤3 输入集群安装所需参数并下载集群配置文件“cluster-[集群名称].yaml”。

图 1-8 下载集群配置文件



步骤4 通过远程传输工具，使用root用户将下载的“agent-[集群名称].yaml”和“cluster-{集群名称}.yaml”文件上传到执行机的“/root/”目录下。

说明

- 若您需要使用L4或L7负载均衡能力，则需要将集群网络类型配置为BGP，具体操作请参见**Cilium**。
- 执行机如果出现SSH连接超时，请参考**虚拟机SSH连接超时处理方法**处理。

----结束

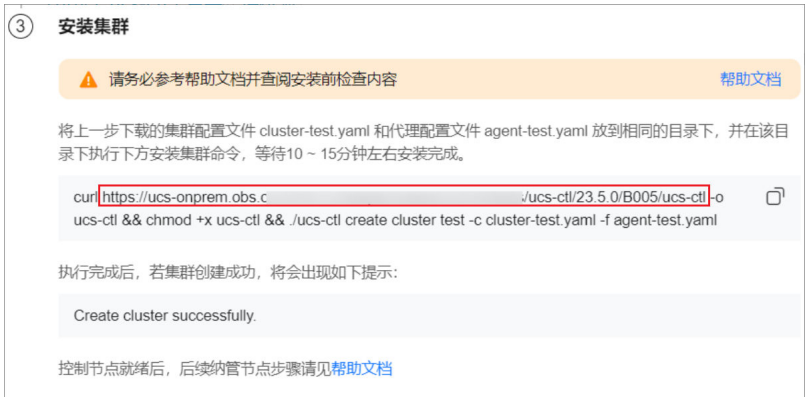
（可选）ucs-ctl 工具完整性校验

ucs-ctl是管理UCS本地集群的命令行工具，在安装本地集群并使用ucs-ctl工具前，为防止您执行被篡改的ucs-ctl工具，请先进行工具的完整性校验。ucs-ctl的详细介绍请参见**使用ucs-ctl命令行工具管理本地集群**。

本地集群支持使用sha256校验文件来验证 ucsctl 文件的完整性。

步骤1 单击“安装集群”，复制如图1-9所示位置的地址并记录，即ucs-ctl工具的安装地址。

图 1-9 ucs-ctl 下载地址



步骤2 将如下命令中的下载地址替换为步骤1中所记录的地址，然后执行命令，以下载 sha256 校验文件。

```
curl 下载地址.sha256 -o ucs-ctl.sha256 #
```

步骤3 将该校验文件移动至 ucs-ctl 同目录下，执行如下命令，以校验工具的完整性。

```
sha256sum -c <(grep ucs-ctl ucs-ctl.sha256)
```

步骤4 执行命令后输出为 “OK”，则校验通过；若输出为“FAILED”，则校验错误，请提交工单，联系技术支持人员解决。

----结束

安装本地集群

步骤1 单击“安装集群”，复制安装命令，并在“/root”目录（或其他可用目录）下执行命令。

图 1-10 安装本地集群

步骤2 前往UCS控制台刷新集群状态，集群处于“运行中”。

步骤3 单击本地集群名称，进入集群控制台页面，对集群节点、工作负载等资源进行操作，操作正常无报错，即本地集群接入成功。

----结束

1.3.5 管理本地集群

1.3.5.1 本地集群 KubeConfig 文件

获取本地集群 KubeConfig 文件

KubeConfig是Kubernetes集群中组织有关集群、用户、命名空间和身份认证机制信息的配置文件，Kubectl使用KubeConfig来获取集群的信息并与API server进行通信。

获取本地集群的KubeConfig需要使用ucs-ctl工具，获取步骤如下:

步骤1 使用ucs-ctl获取集群名称。

```
$ ./ucs-ctl get cluster
```

```
[root@local-cluster-0001 ~]# ./ucs-ctl get cluster
+---------------+---------+---------------+--------------+--------------+--------------+
| CLUSTER NAME  | USE ELB |    VIP/ELB    |   MASTER-1   |   MASTER-2   |   MASTER-3   |
+---------------+---------+---------------+--------------+--------------+--------------+
| test-redhat86 | false   | 192.168.0.165 | 192.168.0.68 | 192.168.0.225 | 192.168.0.145 |
+---------------+---------+---------------+--------------+--------------+--------------+
```

步骤2 使用ucs-ctl导出指定集群的KubeConfig。

```
$ ./ucs-ctl get kubeconfig -c test-redhat86 -o kubeconfig
```

说明

可以使用ucs-ctl get kubeconfig -h查看获取KubeConfig所使用到的参数。

- -c, --cluster：指定待导出KubeConfig的集群名。
- -e, --eip：指定API server的eip。
- -o, --output：指定KubeConfig导出文件名。

----结束

使用本地集群 KubeConfig 文件

拿到ucs-ctl生成的KubeConfig之后，还需要使KubeConfig在节点上生效，步骤如下：

步骤1 复制KubeConfig到节点上。

```
$ scp /local/path/to/kubeconfig user@remote:/remote/path/to/kubeconfig
```

步骤2 如果当前节点添加过EnableSecretEncrypt环境变量，需要先取消。

```
$ unset EnableSecretEncrypt
```

步骤3 使KubeConfig生效，可选方法：

- 方法一：复制到默认位置。
  ```
  $ mv /remote/path/to/kubeconfig $HOME/.kube/config
  ```
- 方法二：环境变量指定。
  ```
  $ export KUBECONFIG=/remote/path/to/kubeconfig
  ```
- 方法三：命令行参数指定。
  ```
  $ kubectl --kubeconfig=/remote/path/to/kubeconfig
  ```

----结束

上述操作之后，就可以访问本地集群的API server了。关于KubeConfig更详细的用法可参考：**使用kubeconfig文件组织集群访问Kubernetes**。

1.3.5.2 本地集群配置文件

本地集群配置文件为一个Cluster.yaml文件，是在UCS 控制台中自动生成，用于初始化本地集群的master节点。表1-12是该文件内各个字段的说明。

表 1-12 命令

配置项	配置命令
# ssh登录master节点用户	USERNAME: root
# ssh登录master节点密码	PASSWORD:
# 集群master1节点IP地址	MASTER-1:
# 集群master2节点IP地址	MASTER-2:
# 集群master3节点IP地址	MASTER-3:
# 是否使用ELB# 是否使用ELB	ACCESS_EXTERNAL_LOAD_BALANCE: false
# 可用ELB地址	EXTERNAL_LOAD_BALANCE_IP:
# 集群VIP地址	VIRTUAL_IP:
# 容器网络服务	NETWORK_PROVIDER: cilium
# 容器网段	CILIUM_IPV4POOL_CIDR: 172.16.0.0/16
# cilium bgp 开关	CILIUM_BGP_ENABLE: false
# cilium bgp 邻居地址	CILIUM_BGP_PEER_ADDRESS: 127.0.0.1
# cilium bgp AS编号	CILIUM_BGP_PEER_ASN: 65010
# cilium 负载均衡网段	LOAD_BALANCER_CIDR:
# cilium 容器网络模式	CILIUM_NETWORK_MODE: overlay
# 时区	TIMEZONE: Asia/Shanghai
# 是否对管理节点打污点	TAINT_MANAGE: yes
# 是否使用NTP	INSTALL_NTP: true
# 外接ntp服务地址	NTP_SERVER_IP:
# 代理转发模式	PROXY_MODE: ebpf
# 外接dns服务地址	DNS_SERVER_IP:
# 集群外部访问地址	CUSTOM_IP:
# 集群安装包下载地址	PACKAGE_PATH:
# 集群镜像包下载地址	IMAGES_PACKAGE_PATH:
# IAM 租户ID	IAM_DOMAIN_ID:
# IAM 云服务地址	IAM_ENDPOINT:

1.3.5.3 管理本地集群节点

本小节介绍如何通过ucs-ctl工具管理本地集群节点。

说明

ucs-ctl是管理UCS本地集群的命令行工具，ucs-ctl的详细介绍请参见**使用ucs-ctl命令行工具管理本地集群**。

纳管节点

步骤1 在执行机上使用**./ucs-ctl config generator -t node -o node.csv**命令生成纳管节点时使用的配置文件。

步骤2 将所需节点的参数写入配置文件，使用英文逗号分隔，如下所示。参数描述如**表1-13**。

表 1-13 配置文件参数描述

参数	描述
Node IP	节点IP
User	SSH连接用户
Password	SSH连接密码

示例：

```
Node IP,User,Password
123.45.6.789,root,********
123.45.6.890,root,********
```

步骤3 在执行机上执行以下命令**./ucs-ctl create node -c [集群名称] -m node.csv**，完成节点的纳管。

----结束

注意

node.csv文件中存在秘钥信息，请妥善保管。

卸载节点

- 方法一：
  在执行机上执行
  ```
  ./ucs-ctl delete node  -c [集群名称] -n [node ip1],[node ip2],…
  ```
  使用-n指定具体IP，使用英文逗号分隔。
- 方法二：

在执行机执行

```
./ucs-ctl delete node  -c [集群名称] -m node.csv
```

使用-m指定纳管时使用的节点文件，可以一次性卸载所有节点。

说明

如果命令执行失败，请参考**如何手动清理本地集群节点?** 处理。

1.3.5.4 管理本地集群网络

1.3.5.4.1 Cilium 概述

为什么需要 Cilium

Cilium是一种高性能、高可靠性的容器网络解决方案，它通过eBPF技术在Linux内核层面实现网络和安全通信。它支持多种传输层协议，例如TCP、UDP和HTTP，并提供了多种安全特性，例如应用层访问控制和服务网格支持。Cilium还支持Kubernetes网络策略，并提供了全局网络和服务发现功能，能够帮助管理员更好地管理和部署云原生应用和服务。

Cilium的eBPF 技术通过在Linux内核层面实时监控网络流量，实现了高效的安全数据包交换。该技术在网络功能虚拟化、容器网络和边缘计算等场景中都有广泛应用，能够帮助企业提升网络性能和安全性，为云原生应用提供更好的基础设施支持。

基本功能

- 为容器提供网络互通：Cilium通过为每个容器分配一个独特的IP地址来实现容器间的网络互通，同时支持多种网络协议。
- 具备网络安全检测能力：Cilium支持通过集成第三方的网络安全检测服务，如Snort等，来进行网络流量分析和检测。
- 自动进行容器安全策略管理：Cilium通过基于Kubernetes 的自定义资源定义（CRD）机制，为每个容器自动创建安全策略，保障容器的安全。
- 实现容器级别的负载均衡：Cilium支持实现容器级别的负载均衡，允许通过多种负载均衡算法来分配网络请求流量。
- 提供服务发现功能：Cilium支持借助基于Kubernetes的服务探测机制，自动发现容器内的服务，并将其注册到Kubernetes API中，以便于其他容器访问。

约束与限制

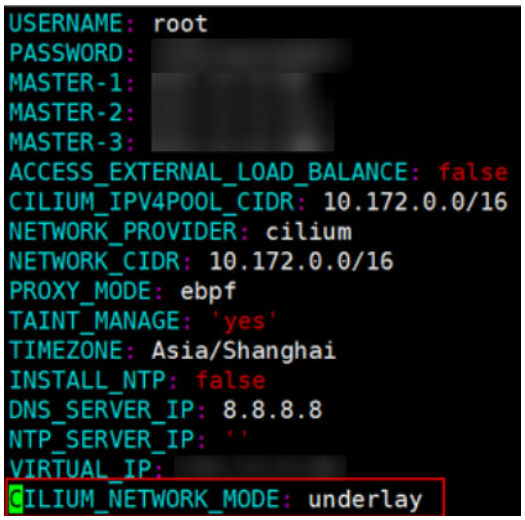
本章节仅适用于新安装的本地集群的增量功能，不支持存量本地集群升级该功能。

Cilium underlay 能力

在本地集群配置文件“cluster-[集群名称].yaml”中添加以下配置：

```
CILIUM_NETWORK_MODE: underlay
```

例图：



模式优势：

- 此模式下，Cilium会将所有未发送到其他容器的数据包委托给Linux内核的路由子系统。这意味着数据包将被路由直接转发，就好像本地进程发出数据包一样，减少了数据包的封装和转换。因此，在大流量场景下，该模式存在一定性能优势。
- 该模式下自动配置ipv4-native-routing-cidr，Cilium会在Linux内核中自动启用IP转发。

模式依赖：

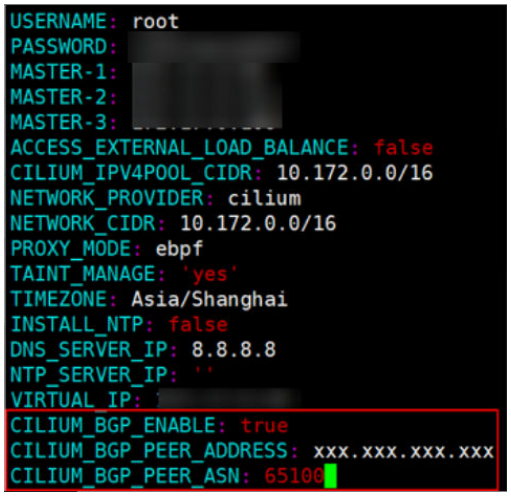
在underlay模式下，运行Cilium的主机的网络必须能够使用分配给Pod或其他工作负载的地址转发IP流量，需要关闭节点的源目的地址检查，并且节点安全组按需放通容器网段的端口及协议。

Cilium 对接主机 BGP

在页面提供的本地集群配置文件“cluster-[集群名称].yaml”中添加以下配置：

```
CILIUM_BGP_ENABLE: true
CILIUM_BGP_PEER_ADDRESS: 需要对接的交换机地址。
CILIUM_BGP_PEER_ASN: 需要对接的交换机BGP AS number（64512-65535）。
LOAD_BALANCER_CIDR：需要广播出去的负载均衡网段，可提供给metalLB等开源插件使用。
```

例图：



将需要对外暴露容器网络的节点IP，作为neighbor address（交换机邻居地址）配置在主机所在的BGP网络中，容器的ASN默认为65010。

Cilium的BGP能力是以节点粒度对外宣告节点容器路由，实现集群外节点服务直接访问集群内Pod。

1.3.5.4.2 使用 L4 负载均衡-MetalLB

Kubernetes不为裸机集群提供网络负载均衡器（LoadBalancer服务）的实现。 裸机集群只能使用NodePort和externalIPs服务来将用户流量引入他们的集群，Metallb旨在通过提供负载均衡器以便裸机集群上的外部服务可以更好工作。关于MetalLB的详细信息请参见**社区官方项目**和**MetalLB官网**。

本章节，将从本地集群的视角出发，提供Metallb的创建及使用指导。

约束与限制

该文档仅适用于为UCS本地集群安装Metallb。

前提条件

按照网络管理Cilium介绍章节，已完成开启集群的BGP功能，配置LOAD_BALANCER_CIDR，并将其广播给底层网络。

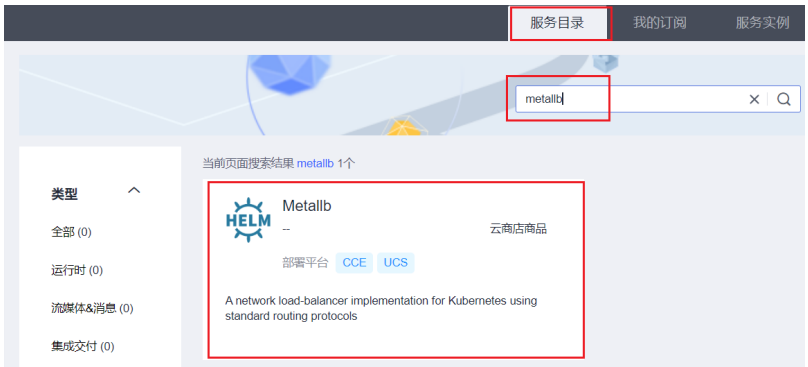
安装 Metallb

步骤1 登录UCS控制台。

步骤2 左侧导航栏内选择“云原生服务中心”。

步骤3 在“服务目录中”中搜索Metallb开源插件，单击进入插件详情。

图 1-11 搜索 Metallb



步骤4 订阅Metallb后，单击创建实例，选择需要使用Metallb的集群。按照页面引导完成安装。

----结束

功能验证

步骤1 进入UCS内的集群控制台。

- 如果是未加入舰队集群，直接单击集群名即可进入集群控制台。

- 如果是已加入容器舰队的集群，先进入对应容器舰队控制台，选择左侧 “集群管理”内的“容器集群”，再进入对应集群控制台。

步骤2 左侧导航栏内选择“工作负载”，单击镜像创建。

步骤3 选择可用镜像创建负载，并添加一个LoadBalancer类型的服务，具体操作请参见**步骤5 工作负载服务配置**。

步骤4 单击服务名称，复制负载均衡IP，于集群外节点上访问，访问成功。

图 1-12 访问负载均衡 IP

```
root@ucs-onpremise-node-0001:~# curl -o /dev/null -s -w "%{http_code}\n" 192.133.0.1:80
200
root@ucs-onpremise-node-0001:~#
```

----结束

1.3.5.4.3 使用 L7 负载均衡 Ingress-nginx

Ingress-nginx控制器用于存储nginx配置，实现统一路由转发管理。关于Ingress-nginx的详细信息请参见**Ingress-Nginx Controller**和**社区官方项目**。

本小节将指导您为本地集群安装与使用Ingress-nginx。

约束与限制

该小节指导仅适用于UCS本地集群安装Ingress-nginx。

安装 Ingress-nginx

步骤1 登录UCS控制台。

步骤2 左侧导航栏内选择“云原生服务中心”，在“服务目录中”中搜索Ingress-nginx开源插件，单击进入插件详情。

图 1-13 搜索 Ingress-nginx

步骤3 订阅Ingress-nginx后，单击“创建实例”，选择需要使用Ingress-nginx的集群。

- 如果是集群已安装Metallb，可以使用Metallb的负载均衡能力，将ingress-nginx服务暴露到集群外，直接根据页面引导进行安装。
- 如果集群未规划安装Metallb，则只能通过NodePort形式暴露ingress-nginx能力。

说明

需要将.values.controller.service.type从LoadBalancer修改为NodePort后进行安装，如图1-14所示。

图 1-14 参数修改

```
    udp: {}
  ports:
    http: 80
    https: 443
  targetPorts:
    http: http
    https: https
  type: NodePort
```

----结束

功能验证

步骤1 登录UCS控制台。

- 如果是未加入舰队集群，直接单击集群名即可进入集群控制台。
- 如果是已加入容器舰队的集群，先进入对应容器舰队控制台，选择左侧 “集群管理”内的“容器集群”，再进入对应集群控制台。

步骤2 左侧导航栏内选择“工作负载”，单击镜像创建。

步骤3 选择可用镜像创建负载，并在“服务配置”中单击，添加一个ClusterIP类型的Service，具体操作请参见步骤5工作负载服务配置。

步骤4 左侧导航栏内选择“服务与路由”，单击“路由”，单击“创建路由”，选择刚刚所创建的ClusterIP类型的Service。路由配置相关操作请参见路由。

步骤5 访问ingress服务，确认转发规则配置成功。

- 如果是使用LoadBalancer暴露的ingress服务，选择LoadBalancer的ingress服务进行集群外访问：

- 如果是使用NodePort暴露的ingress服务，选择任意节点+ingress svc端口进行集群外访问：

----结束

1.3.5.5 升级本地集群

集群升级能力用来完善本地集群的集群生命周期管理能力，目前本地集群的升级方式为用户手动进入集群内，使用命令行工具进行升级，在 UCS 集群管理控制台提供集群升级提示以及升级指引。

约束与限制

- 本地集群升级要求先升级master节点和组件，再升级node节点。

- 集群列表页面的升级提示依赖master节点状态，需要一次完成升级，只有node未升级不会提示在集群列表中。
- 升级版本不可选，默认升到当前集群版本可升级至的最新版本。
- 升级master节点时，集群控制台中的集群状态可能会出现短暂不可用状态，用户升级完成之后会重新接入集群。

升级操作

步骤1 登录UCS控制台，选择“容器舰队”或“未加入舰队的集群”内一个正在运行的低版本集群，单击右下方“升级集群”。

步骤2 下载更新工具，请使用一台能连接集群的节点作为执行机，先使用如下命令下载新版本的集群管理工具：

```
curl https://ucs-onprem.obs.XXXX.huawei.com/toolkits/ucs-ctl/ucs-ctl -o ucs-ctl && chmod +x ucs-ctl
```

步骤3 升级master节点，这里可以使用-y命令来跳过所有选择项，其他可配置的flag请参照**master节点与组件升级命令说明**：

```
./ucs-ctl upgrade cluster [cluster name]
```

注意

集群名称需要和创建本地集群时指定的名称一致，如果不确定名称可进集群内，使用命令查看：

```
./ucs-ctl get cluster
```

步骤4 node节点升级，node节点升级可以选择两种升级方式：

- 全量升级，全量升级指的是将集群内剩余节点全部升级，命令如下：
  ```
  ./ucs-ctl upgrade node -a -c [cluster name]
  ```
- 分批次部分升级，为了防止升级过程出现业务中断的情况，用户也可以选择分批次部分升级节点，此时需要手动选择节点。
  ```
  ./ucs-ctl upgrade node -n [node ip] -c [cluster name]
  ```

说明

特殊情况：若本地集群目前只有master节点，无node节点，此时仅提供master节点的升级命令。

其他可配置的flag请参照**node节点升级命令说明**。

----结束

master 节点与组件升级命令说明

本地集群用户可以通过最新版本的本地集群命令行工具ucs-ctl来进行集群升级，对于管理组件和管理节点的升级，命令如下：

```
./ucs-ctl upgrade cluster [cluster_name] [flags]
```

可以配置的flag如下：

- -a：节点的全量升级，默认情况下upgrade cluster只会升级管理面节点和服务组件，加上-a后则表示全量升级，包括升级所有的业务节点。
- -y：默认同意所有请求。
- -patch：升级补丁包。

- -R：回滚选项。

```
root@ucs-onpremise-master-0001:~# ./ucs-ctl upgrade cluster -h
This command upgrade cluster of the UCS On Premises.

Usage:
  ucs-ctl upgrade cluster [flags]

Examples:
  ucs-ctl upgrade cluster cluster_name

Flags:
  -a, --all          upgrade all nodes
  -y, --assumeyes    answer yes for all questions
  -h, --help         help for cluster
      --patch        upgrade only the patch packages
  -r, --retry        retry: true/false
  -R, --rollback     rollback back to before the upgrade
```

node 节点升级命令说明

对于常规业务节点的升级，命令如下:

```
./ucs-ctl upgrade node [flags] -c [cluster_name]
```

其中必须指定集群名称，即加上`-c [cluster_name]`这个flag。

可以配置的flag如下：

- -a：节点的全量升级。
- -y：默认同意所有请求。
- -c：指定集群名称。
- -r：回滚选项。
- -n：指定节点IP。
- -f：指定节点配置文件。

```
root@ucs-onpremise-master-0001:~# ./ucs-ctl upgrade node -h
This command upgrade the nodes of UCS On Premises.

Usage:
  ucs-ctl upgrade node [flags]

Examples:
  ucs-ctl upgrade node node_name

Flags:
  -a, --allupgrade           full nodes upgrade: true/false
  -y, --assumeyes            answer yes for all questions
  -c, --clustername string   cluster name
  -f, --filename string      filename of node ip list
  -h, --help                 help for node
  -n, --nodeips strings      node ip list
  -r, --retry                retry: true/false
```

1.3.5.6 注销本地集群

在控制台注销本地集群

⚠ 注意

仅在控制台进行注销操作，不会删除集群。

步骤1 进入 UCS 界面，左侧导航栏选择“容器舰队”。

步骤2 找到待注销的本地集群：

- 已加入容器舰队的本地集群，先进入对应的容器舰队控制台，然后再左侧导航栏选择容器集群。
- 未加入容器舰队的本地集群，单击容器舰队列表最上面的“未加入舰队的集群”即可。

步骤3 单击本地集群右上角的注销按钮，会弹出注销确认框。

步骤4 确认待注销的集群名称等信息，并勾选“我已阅读并知晓上述信息”，单击“确定”即可在控制台注销本地集群。

----结束

本地资源清理

⚠ 注意

该操作可能导致与该集群有绑定关系的资源（比如指定调度集群的负载等）无法正常使用，请谨慎操作，避免对运行中的业务造成影响。

步骤1 在控制台注销并没有真正删除本地集群，您需要手动进入集群内完成删除过程。

步骤2 复制注销之后控制台返回的卸载命令。

步骤3 进入本地集群中的节点内，执行卸载命令。

```
./ucs-ctl delete cluster cluster_name
```

说明

cluster_name请替换为集群名称。

----结束

1.3.5.7 使用 ucs-ctl 命令行工具管理本地集群

ucs-ctl是管理UCS本地集群的命令行工具，它仅适用于UCS本地集群。

在使用ucs-ctl工具前，为防止您执行被篡改的ucs-ctl工具，请先进行工具的完整性校验，具体操作请参考**安装及验证**章节的ucs-ctl工具完整性校验操作步骤。

表 1-14 常用命令

命令	解释
config generator	提供集群、节点的创建模板
create	创建集群或者添加节点
delete	删除集群或者移除节点
get	获取本地集群信息
help	帮助信息

命令	解释
version	ucs-ctl版本信息

参数说明

ucs-ctl config generator

```
Flags:
        -o 导出文件路径及名称
        -t 导出模板类型，集群/节点
```

示例：

```
./ucs-ctl config generator -t clustername
```

ucs-ctl create

- 创建集群(ucs-ctl create cluster)

Object:

Clustername 集群名称

```
Flags:
        -f, --agent string       集群连线配置文件
        -c, --config string      集群配置文件
        -h, --help               帮助信息
        -r, --retry              安装重试
```

示例：

```
./ucs-ctl create cluster clustername -c cluster.yaml -f agent.yaml
```

- 新增节点(ucs-ctl create node)

```
Flags:
        -c, --cluster string     操作集群名称
        -h, --help               帮助信息
        -m, --machine string     待纳管节点信息
        -r, --retry              纳管重试
```

示例：

```
./ucs-ctl create node -c cluster_name -m machine.csv
```

ucs-ctl delete

- 删除集群(ucs-ctl delete cluster)

```
Flags:
        -y, --default-yes        确认输入默认yes
        -h, --help               帮助信息
```

示例：

```
./ucs-ctl delete cluster clustername
```

- 删除节点(ucs-ctl delete node)

```
Flags:
        -y, --assumeyes          确认输入默认yes
        -c, --cluster string     待删除节点集群名称
        -h, --help               帮助信息
        -m, --machine string     待删除节点信息
        -n, --node-ip string     待删除节点ip
```

示例：

```
./ucs-ctl delete node -c clustername -m machine.csv
```

ucs-ctl get

- 获取本地集群信息(ucs-ctl get cluster)

示例：

```
./ucs-ctl get cluster
```

- 获取kubeconfig信息(ucs-ctl get kubeconfig)

```
Flags:
        -c, --cluster string    集群名称
        -e, --eip string        使用eip为api接入点
        -h, --help              帮助信息
        -o, --output string     导出路径文件
```

示例：

```
./ucs-ctl get kubeconfig -c clustername -o kubeconfig
```

1.3.5.8 GPU 虚拟化

1.3.5.8.1 GPU 虚拟化概述

UCS On Premises GPU采用xGPU虚拟化技术，能够动态对GPU设备显存与算力进行划分，单个GPU卡最多虚拟化成20个GPU虚拟设备。相对于静态分配来说，虚拟化的方案更加灵活，最大程度保证业务稳定的前提下，可以完全由用户定义使用的GPU数量，提高GPU利用率。

GPU 虚拟化的优势

UCS On Premises提供的GPU虚拟化功能优势如下：

- **灵活：**精细配置GPU算力占比及显存大小，算力分配粒度为5%GPU，显存分配粒度达MB级别。
- **隔离：**支持显存和算力的严格隔离，支持单显存隔离，算力与显存同时隔离两类场景。
- **兼容：**业务无需重新编译，无需进行CUDA库替换，对业务无感。

1.3.5.8.2 准备 GPU 虚拟化资源

本文介绍如何在使用GPU虚拟化能力前所需要的基础软件、硬件规划与准备工作。

基础规划

配置	支持版本
集群版本	v1.25.15-r7及以上
操作系统	Huawei Cloud EulerOS 2.0
GPU类型	T4、V100
GPU驱动版本	470.57.02、470.103.01、470.141.03、510.39.01、510.47.03

配置	支持版本
容器运行时	containerd
插件	集群中需要同时安装以下插件： • volcano插件：1.10.1及以上版本 • gpu-device-plugin插件：2.0.0及以上版本

步骤一：纳管并标记 GPU 节点

说明

如果您的集群中已有符合基础规划的GPU节点，您可以跳过此步骤。

步骤1 在集群中纳管支持GPU虚拟化的节点，具体操作步骤请参见纳管节点。

步骤2 纳管成功后，给对应支持GPU虚拟化节点打上“accelerator: nvidia-{显卡型号}”标签，具体操作步骤请参见为节点添加标签/污点。

图 1-15 为虚拟化节点打标签

----结束

步骤二：安装插件

说明

如果您的集群中已安装符合基础规划的插件，您可以跳过此步骤。

更改驱动版本后，需要重启节点才能生效。

步骤1 登录UCS控制台，单击集群名称进入集群，在左侧导航栏中选择“插件管理”，查看“已安装插件”中是否存在volcano插件与gpu-device-plugin插件。

步骤2 若未安装volcano插件，请安装该插件，具体操作请参见volcano。

若未安装gpu-device-plugin插件，请安装该插件，具体操作请参见gpu-device-plugin。

----结束

1.3.5.8.3 创建 GPU 虚拟化应用

本文介绍如何使用GPU虚拟化能力实现算力和显存隔离，高效利用GPU设备资源。

前提条件

- 已完成**GPU虚拟化资源准备**。
- 如果您需要通过命令行创建，需要使用kubectl连接到集群，详情请参见通过kubectl连接集群。

约束与限制

- init容器不支持进行GPU虚拟化。
- 对于单张GPU卡：
  - 最多虚拟化为20个GPU虚拟设备。
  - 最多调度20个使用隔离能力的Pod。
  - 仅支持调度相同隔离模式（GPU虚拟化支持显存隔离、显存与算力隔离两种隔离模式。）的工作负载。
- 对于同一工作负载中的不同容器：
  - 仅支持配置单一显卡型号，不支持混合配置两种及以上GPU显卡型号。
  - 仅支持配置一致GPU使用模式，不支持混合配置虚拟化和非虚拟化模式。
- 使用GPU虚拟化后，该GPU节点不再支持调度使用**共享GPU资源**的工作负载。

通过控制台创建 GPU 虚拟化应用

步骤1 登录UCS On Premises集群控制台。

步骤2 单击集群名称进入集群，在左侧选择“工作负载”，在右上角单击“创建负载”。

步骤3 配置工作负载信息。在“容器配置>基本信息”中设置GPU配额：

显存：显存值单位为Mi，需为正整数，若配置的显存超过单张GPU卡的显存，将会出现无法调度状况。

算力：算力值单位为%，需为5的倍数，且最大不超过100。

图 1-16 配置工作负载信息

步骤4 配置其余信息，完成后单击“创建”。

步骤5 工作负载创建成功后，您可以尝试验证GPU虚拟化的隔离能力。

- 登录容器查看容器被分配显存总量

```
kubectl exec -it gpu-app -- nvidia-smi
```

预期输出：

```
Wed Apr 12 07:54:59 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4792MiB /  5000MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

预期输出表明，该容器被分配显存总量为**5000 MiB**，实际使用了**4792MiB**。

- 查看所在节点的GPU显存隔离情况（在节点上执行）。

```
export PATH=$PATH:/usr/local/nvidia/bin;nvidia-smi
```

预期输出：

```
Wed Apr 12 09:31:10 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4837MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    760445      C   python                          4835MiB |
+-----------------------------------------------------------------------------+
```

预期输出表明，GPU节点上的显存总量为**16160 MiB**，其中示例Pod使用了**4837MiB**。

----结束

通过 kubectl 命令行创建 GPU 虚拟化应用

步骤1 登录集群master节点，使用kubectl连接集群。

步骤2 创建使用GPU虚拟化的应用。创建gpu-app.yaml文件，内容如下：

说明

当前支持隔离显存或同时隔离显存与算力，暂不支持设置为仅隔离算力，即不支持单独设置volcano.sh/gpu-core.percentage。

- 仅隔离显存：

```
apiVersion: apps/v1
 kind: Deployment
 metadata:
   name: gpu-app
   labels:
```



```
    app: gpu-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      containers:
      - name: container-1
        image: <your_image_address>     # 请替换为您的镜像地址
        resources:
          limits:
            volcano.sh/gpu-mem: 5000    # 该Pod分配的显存大小
      imagePullSecrets:
        - name: default-secret
```

- 同时隔离显存与算力：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-app
  labels:
    app: gpu-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpu-app
  template:
    metadata:
      labels:
        app: gpu-app
    spec:
      containers:
      - name: container-1
        image: <your_image_address>     # 请替换为您的镜像地址
        resources:
          limits:
            volcano.sh/gpu-mem: 5000    # 该Pod分配的显存大小
            volcano.sh/gpu-core.percentage: 25    # 该Pod分配的算力大小
      imagePullSecrets:
        - name: default-secret
```

表 1-15 关键参数说明

参数	是否必选	描述
volcano.sh/gpu-mem	否	显存值单位为MiB，需为正整数，若配置的显存超过单张GPU卡的显存，将会出现无法调度状况。
volcano.sh/gpu-core.percentage	否	算力值单位为%，需为5的倍数，且最大不超过100。

步骤3 执行以下命令，创建应用。

```
kubectl apply -f gpu-app.yaml
```

步骤4 验证GPU虚拟化的隔离能力。

- 登录容器查看容器被分配显存总量。

```
kubectl exec -it gpu-app -- nvidia-smi
```


预期输出：

```
Wed Apr 12 07:54:59 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4792MiB /  5000MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

预期输出表明，该容器被分配显存总量为**5000 MiB**，实际使用了**4792MiB**。

- 查看所在节点的GPU显存隔离情况（在节点上执行）。

```
/usr/local/nvidia/bin/nvidia-smi
```

预期输出：

```
Wed Apr 12 09:31:10 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.141.03   Driver Version: 470.141.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   27C    P0    37W / 300W |   4837MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    760445      C   python                           4835MiB |
+-----------------------------------------------------------------------------+
```

预期输出表明，GPU节点上的显存总量为**16160 MiB**，其中示例Pod使用了**4837MiB**。

----结束

1.3.5.8.4 监控 GPU 虚拟化资源

本章介绍如何在UCS控制台界面查看GPU虚拟化资源的全局监控指标。

前提条件

- 完成GPU虚拟化资源准备。
- 当前本地集群内存在节点开启GPU虚拟化能力。
- 当前本地集群开启了监控能力。

GPU 虚拟化监控

步骤1 登录UCS控制台，在左侧导航栏选择“容器智能分析”。

步骤2 选择对应的集群并开启监控，详细操作请参照**集群开启监控**。

步骤3 单击集群名称，进入“容器洞察”总览页面。

步骤4 选择“仪表盘”，在“集群视图”旁单击“切换视图”，切换为“XGPU视图”。

图 1-17 仪表盘

步骤5 查看xGPU视图。

----结束

1.3.5.9 备份与恢复

背景

UCS本地集群安装完成后，为保证集群高可用，防止在发生集群故障时数据丢失，UCS支持对于本地集群上的3个master节点上的证书文件、加解密物料、etcd数据等信息的备份，以保障UCS本地集群故障后的数据恢复。

约束与限制

无论是单master还是多master故障，节点IP须保持不变。

集群备份

本地备份

1. 创建备份文件压缩包存放的目录。
2. 执行备份命令：

```
./ucs-ctl backup 集群名 --path 备份路径 --type local
```

示例如下：

```
./ucs-ctl backup gpu-test --path /home/ggz/gpu-test --type local
```

执行成功后，会在指定的备份路径下产生名为：集群名-backup-时间戳.tar.gz这样一个备份压缩文件。

备份压缩文件存在ha.yaml、etcd-snapshot目录、crt目录，etcd-snapshot包含etcd备份数据，crt包含证书与加解密材料。

远端备份

1. 在远端sftp主机创建备份文件压缩包存放的目录。
2. 执行备份命令：

```
./ucs-ctl backup 集群名 --path 备份路径 --type sftp --ip 远端主机ip --user 远端主机用户名
```

示例如下：

```
./ucs-ctl backup gpu-test --path /home/ggz/gpu-test --type sftp --ip 100.95.142.93 --user root
```

须知

- 首次进行远端备份需要输入sftp密码，请在please input sftp password提示后输入远端sftp服务器密码。
- 备份命令中的备份路径必须真实有效，否则执行命令后在远端生成的备份文件可能存在错误。

执行成功后，远端主机的指定备份路径下会产生名为“集群名-backup-时间戳.tar.gz”的备份压缩文件。该备份压缩文件包含ha.yaml、etcd-snapshot目录、crt目录，etcd-snapshot包含etcd备份数据，crt包含证书与加解密材料。

周期备份

执行crontab -e，通过编写crontab表达式使得备份命令周期执行。

示例如下，代表每天16：40执行集群周期本地备份任务。

```
40 16 * * * /root/cluster/ucs-ctl backup cluster-redhat --path /root/cluster/backup --type local
```

周期远端备份时，在首次输入完远端sftp服务器密码后，在crontab表达式中无需指明密码。

为防止备份文件数量一直膨胀，需在备份机器上同样通过crontab周期执行备份文件老化脚本，备份文件老化参考如下：

```
#!/bin/bash
backup_dir=${1}                                  # 备份文件保存路径
keep_days=${2}                                   # 保存天数
if [ ! -d ""$backup_dir"" ]; then                   # 检查日志目录是否存在
  echo "备份文件路径不存在！"
  exit 1
fi
find "$backup_dir" -type f -mtime +$keep_days -exec rm {} \;       # 删除旧日志
echo "过期备份文件已删除！"
```

数据恢复

etcd数据恢复

1. 准备etcd备份数据文件。

   将备份数据压缩文件{clustername}-backup-{timestamp}.tar.gz发送到需要恢复的集群的各个master/etcd节点上。
2. 关闭etcd服务

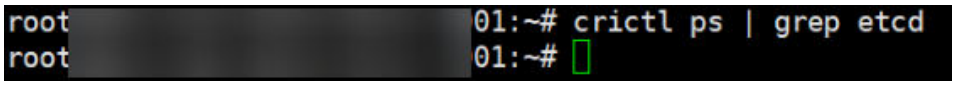
   在节点上执行：

```
mv /var/paas/kubernetes/manifests/etcd*.manifest /var/paas/kubernetes/
```

等待服务停止。

```
crictl ps | grep etcd
```

若查询不到etcd容器，说明服务已经停止。



3. 备份节点的etcd数据（可选）。

```
mv /var/paas/run/etcd/data /var/paas/run/etcd/data-bak
mv /var/paas/run/etcd-event/data /var/paas/run/etcd-event/data-bak
```

4. 在含有etcd的节点上执行恢复命令。

```
./ucs-ctl restore etcd 备份文件压缩包路径
```

示例如下：

```
./ucs-ctl restore etcd /home/ggz/gpu-test/backup-file-20230625164904.tar.gz
```

若回显如下命令，则etcd的节点数据恢复成功：

```
Restore the etcd snapshot successfully.
```

5. 对etcd节点重启etcd服务，重启过程需要等待几分钟。

```
mv /var/paas/kubernetes/etcd*.manifest /var/paas/kubernetes/manifests
```

等待服务重新启动：

```
crictl ps | grep etcd
```

若查询到etcd容器说明服务已经重启，此时该节点的etcd数据得到恢复。

说明

etcd数据恢复需要每个含有etcd的节点上单独执行1-5步骤。

单master节点故障恢复

步骤1 在执行机上执行单节点故障恢复命令。

```
./ucs-ctl restore node 节点ip --name 集群名
```

步骤2 其中节点ip为故障节点的ip，示例如下。

```
./ucs-ctl restore node 192.168.0.87 --name gpu-test
```

步骤3 若回显如下命令，则单master节点故障恢复成功：

```
restore node 192.168.0.87 successfully.
```

步骤4 在每个含有etcd的节点上进行etcd数据恢复，具体命令请参考步骤1-5。

----结束

多master节点故障恢复

步骤1 对所有master节点进行系统清理。将执行机目录下的卸载脚本拷贝到3台master节点并执行。（脚本路径为/var/paas/.ucs-package/ucs-onpremise/scripts/uninstall_node.sh）

```
scp -r /var/paas/.ucs-package/ucs-onpremise/scripts/uninstall_node.sh root@{目标节点IP}:/root
sudo bash /root/uninstall_node.sh
```

步骤2 在执行机上执行集群故障恢复命令。

```
./ucs-ctl restore cluster 集群名 -b 备份文件压缩包路径
```

示例命令：

```
./ucs-ctl restore cluster gpu-test -b /home/ggz/gpu-test/backup-file-20230625164904.tar.gz
```


步骤3 若回显如下命令，则多master节点故障恢复成功：

```
restore cluster successfully.
```

步骤4 在每个含有etcd的节点上进行etcd数据恢复，具体命令请参考步骤1-5。

----结束

1.4 附着集群

1.4.1 附着集群概述

附着集群指满足CNCF（Cloud Native Computing Foundation）标准的第三方Kubernetes集群，如AWS（EKS）、GCP（GKE）以及自建的Kubernetes集群。

附着集群的管理流程如图1-18所示。

图 1-18 附着集群管理流程

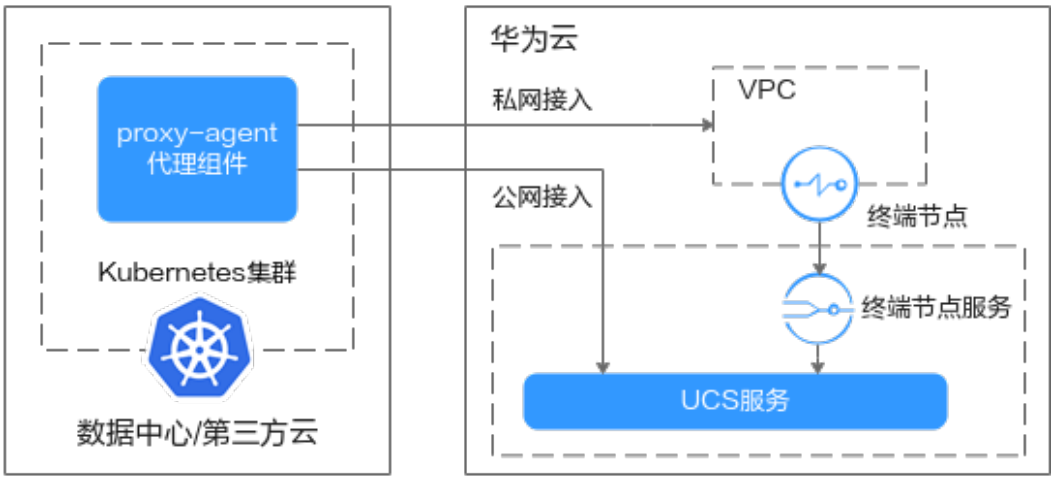
接入网络模式

对于附着集群，各集群提供商或本地数据中心对于网络入方向的端口规则有差异，防止特定端口外的入站通信。因此UCS使用集群网络代理的连接方式，如图1-19所示，无需在防火墙上启用任何入方向端口，仅通过集群代理程序的方式在出方向与UCS服务建立会话。

附着集群接入网络的方法有两种，具有不同的优点：

- **公网接入**：具有弹性灵活、成本低、易接入的优点。
- **私网接入**：可获得更加高速、低时延、稳定安全的体验。

图 1-19 集群接入原理



1.4.2 注册附着集群（公网接入）

本章节讲述附着集群的注册及公网接入流程。

约束与限制

- **华为云账号**用户需具备UCS FullAccess和VPCEndpoint Administrator权限。
- 若集群地域位于境外，应确保您的行为符合所适用的法律法规要求。
- 请确保注册的集群为通过CNCF一致性认证，且版本在1.19至1.28之间的Kubernetes集群。

前提条件

- 已创建一个准备接入UCS的集群，并且集群状态正常。
- 集群中部署proxy-agent组件的节点必须可以进行公网访问，可选择挂载EIP或使用NAT网关的方式。
- 已获取待添加集群的KubeConfig文件，具体操作步骤因厂商而异，请参见**KubeConfig**。关于KubeConfig文件的更多说明请参考**使用kubeconfig文件组织集群访问**。

步骤一：注册集群

步骤1 登录UCS控制台。

步骤2 在左侧导航栏中选择“容器舰队”，单击附着集群选项卡中的“注册集群”按钮。

步骤3 参考**表1-16**填写待添加集群的基础信息，其中带“*”的参数为必填参数。

表 1-16 注册集群基础信息配置

参数	参数说明
集群名称*	输入集群的自定义名称，需以小写字母开头，由小写字母、数字、中划线（-）组成，且不能以中划线（-）结尾。
集群服务商*	选择一个集群服务商。
所属区域*	选择集群所在的区域。
集群标签	非必填项，以键值对的形式为集群添加标签，可以通过标签实现集群的分类。键值对可自定义，以字母或者数字开头和结尾，由字母、数字、连接符（-）、下划线（_）、点号（.）组成，且63个字符之内。
上传KubeConfig*	上传kubectl的配置文件来完成集群认证，支持JSON或YAML格式。获取KubeConfig文件的操作步骤因厂商而异，请参见**KubeConfig**。
选择Context*	选择对应的Context。在完成KubeConfig文件上传后，选项列表将自动获取文件中的“contexts”字段。 默认值为KubeConfig文件中“current-context”字段指定的Context，若文件中无此字段则需要从列表中手动选择。

参数	参数说明
容器舰队	选择集群所属的舰队。 舰队用于权限精细化管理，一个集群只能加入一个舰队。若不选择舰队，集群注册成功后将显示在“未加入舰队的集群”页签下，后续还可以再添加至舰队中。 不支持在注册集群阶段选择已开通集群联邦能力的舰队，如果一定要加入这个舰队，请在集群注册成功后，再添加到该舰队中。关于集群联邦的介绍，请参见**开通集群联邦**章节。 如需新建舰队，请参见**管理容器舰队**。

步骤4 单击“确定”，集群注册成功后如**图1-20**所示，请在30分钟内接入网络。您可选择集群的接入方式或单击右上角按钮查看详细的网络接入流程。

如您未在30分钟内接入网络，将会导致集群注册失败，可单击右上角按钮重新注册集群。如果已经接入但数据未采集上来，请等待2分钟后刷新集群。

图 1-20 集群等待接入状态

----结束

步骤二：接入网络

在UCS控制台成功添加集群后，集群状态将会显示为“等待接入”，此时UCS并未打通与集群的网络连接，因此需要在集群中配置网络代理来接入网络。

步骤1 登录UCS控制台。

步骤2 单击待接入集群栏的“公网接入”，下载集群代理agent的配置文件。

说明

集群代理配置存在私有秘钥信息，每个集群代理配置仅能下载一次，请您妥善保管。

步骤3 使用kubectl连接集群，使用如下命令在集群中创建一个名为“agent.yaml”的YAML文件（该文件名称可自定义），并将**步骤2**中的agent配置内容粘贴到YAML文件中。

```
vim agent.yaml
```

步骤4 在待接入集群中执行以下命令部署代理。

```
kubectl apply -f agent.yaml
```

步骤5 查看集群代理部署状态。

```
kubectl -n kube-system get pod | grep proxy-agent
```

如果部署成功，预期输出如下：

```
proxy-agent-5f7d568f6-6fc4k 1/1 Running 0 9s
```

步骤6 查看集群代理运行状态。


```
kubectl -n kube-system logs <Agent Pod Name> | grep "Start serving"
```

如果正常运行，日志预期输出如下：

```
Start serving
```

步骤7 前往UCS控制台刷新集群状态，集群处于“运行中”。

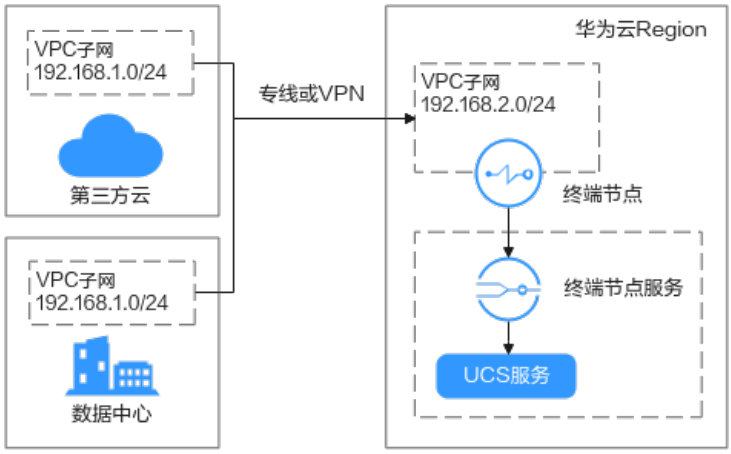
----结束

1.4.3 注册附着集群（私网接入）

位于本地数据中心和第三方云上的附着集群通过公网接入UCS存在一定的安全风险，用户需要稳定安全的集群接入方式，此时可以使用私网接入的方式将集群纳入UCS进行管理。

私网连接方式是通过云专线（DC）或虚拟专用网络（VPN）服务将云下网络与云上虚拟私有云（VPC）连通，并利用VPC终端节点通过内网与UCS服务建立连接，具有高速、低时延、安全的优势。

图 1-21 私网接入原理



约束与限制

- **华为云账号**用户需具备UCS FullAccess和VPCEndpoint Administrator权限。
- 若集群地域位于境外，应确保您的行为符合所适用的法律法规要求。
- 请确保注册的集群为通过CNCF一致性认证，且版本在1.19至1.28之间的Kubernetes集群。
- 附着集群通过私网接入时，由于网络限制，镜像仓库功能的使用可能受限：

  私网接入的集群无法通过私网下载SWR镜像仓库中的镜像，请确保工作负载运行的节点可访问公网。

前提条件

- 已创建一个准备接入UCS的集群，并且集群状态正常。
- 在UCS提供服务的区域中创建一个VPC，具体操作请参见**创建虚拟私有云和子网**。

说明

该VPC子网网段不能与IDC或第三方云中已使用的网络网段重叠，否则将无法接入集群。例如，IDC中已使用的VPC子网为192.168.1.0/24，那么华为云VPC中不能使用192.168.1.0/24这个子网。

- 已获取待添加集群的KubeConfig文件，具体操作步骤因厂商而异，请参见**KubeConfig**。关于KubeConfig文件的更多说明请参考**使用kubeconfig文件组织集群访问**。

步骤一：准备网络环境

须知

云下、云上网络打通后，建议从本地数据中心服务器ping目标VPC下的华为云服务器私网IP，以验证网络是否成功连接。

将线下自有IDC或第三方云厂商的网络环境与华为云VPC打通。

- 虚拟专用网络（VPN）方案：请参见**通过VPN连接云下数据中心与云上VPC**。
- 云专线（DC）方案：请参见**用户通过单专线静态路由访问VPC**或**用户通过单专线BGP协议访问VPC**。

步骤二：注册集群

步骤1 登录UCS控制台。

步骤2 在左侧导航栏中选择“容器舰队”，单击附着集群选项卡中的“注册集群”按钮。

步骤3 参考**表1-17**填写待添加集群的基础信息，其中带“*”的参数为必填参数。

表 1-17 注册集群基础信息配置

参数	参数说明
集群名称*	输入集群的自定义名称，需以小写字母开头，由小写字母、数字、中划线（-）组成，且不能以中划线（-）结尾。
集群服务商*	选择一个集群服务商。
所属区域*	选择集群所在的区域。
集群标签	非必填项，以键值对的形式为集群添加标签，可以通过标签实现集群的分类。键值对可自定义，以字母或者数字开头和结尾，由字母、数字、连接符（-）、下划线（_）、点号（.）组成，且63个字符之内。
上传KubeConfig*	上传kubectl的配置文件来完成集群认证，支持JSON或YAML格式。获取KubeConfig文件的操作步骤因厂商而异，请参见**KubeConfig**。

参数	参数说明
选择Context*	选择对应的Context。在完成KubeConfig文件上传后，选项列表将自动获取文件中的“contexts”字段。 默认值为KubeConfig文件中“current-context”字段指定的Context，若文件中无此字段则需要从列表中手动选择。
容器舰队	选择集群所属的舰队。 舰队用于权限精细化管理，一个集群只能加入一个舰队。若不选择舰队，集群注册成功后将显示在“未加入舰队的集群”页签下，后续还可以再添加至舰队中。 不支持在注册集群阶段选择已开通集群联邦能力的舰队，如果一定要加入这个舰队，请在集群注册成功后，再添加到该舰队中。关于集群联邦的介绍，请参见**开通集群联邦**章节。 如需新建舰队，请参见**管理容器舰队**。

步骤4 单击“确定”，集群注册成功后如**图1-22**所示，请在30分钟内接入网络。您可选择集群的接入方式或单击右上角按钮查看详细的网络接入流程。

如您未在30分钟内接入网络，将会导致集群注册失败，可单击右上角按钮重新注册集群。如果已经接入但数据未采集上来，请等待2分钟后刷新集群。

图 1-22 集群等待接入状态

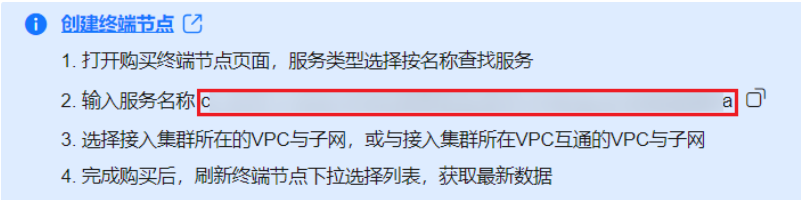
----结束

步骤三：购买终端节点

步骤1 登录UCS控制台，单击待接入集群栏的“单击接入”进入集群接入界面，单击“私网接入”。

步骤2 查看“创建终端节点”中的服务名称，单击，记录服务名称。

图 1-23 创建终端节点



步骤3 登录VPC终端节点控制台，单击“创建终端节点”，创建连接不同服务的终端节点。

步骤4 选择终端节点的区域。

步骤5 选择“按名称查找服务”，输入所记录的服务名称，并单击“验证”。

图 1-24 购买终端节点

步骤6 选择**步骤一：准备网络环境**中与集群网络连通的虚拟私有云以及对应的子网。

步骤7 根据需求选择终端节点的“节点IP”为“自动分配”或“手动分配”。

步骤8 配置完其他参数后，单击“立即购买”，并进行规格确认。

- 规格确认无误，单击“提交”，任务提交成功。
- 参数信息配置有误，需要修改，单击“上一步”，修改参数，然后单击“提交”。

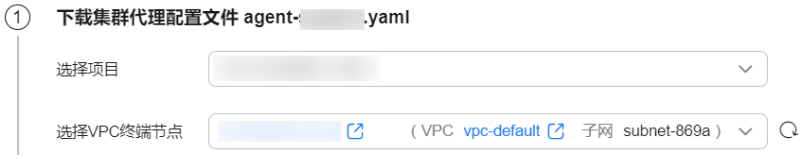
----结束

步骤四：接入集群

步骤1 登录UCS控制台，在“等待接入”状态下的目标集群栏中单击“私网接入”。

步骤2 选择项目，再选择**步骤三：购买终端节点**中创建的终端节点。

图 1-25 选择终端节点



步骤3 将2中的agent配置文件上传至节点。

步骤4 单击“安装集群代理agent配置”，在待接入集群中执行如下命令，可单击右侧 直接复制命令。

图 1-26 安装集群代理 agent 配置

须知

- 私网接入的集群无法通过私网下载SWR镜像仓库中的镜像，请确保工作负载运行的节点可访问公网。
- 拉取proxy-agent容器镜像要求集群需要具备公网访问能力，或将proxy-agent镜像上传至集群可访问的镜像仓库，否则将导致proxy-agent部署失败。

步骤5 前往UCS控制台刷新集群状态，集群处于“运行中”。

----结束

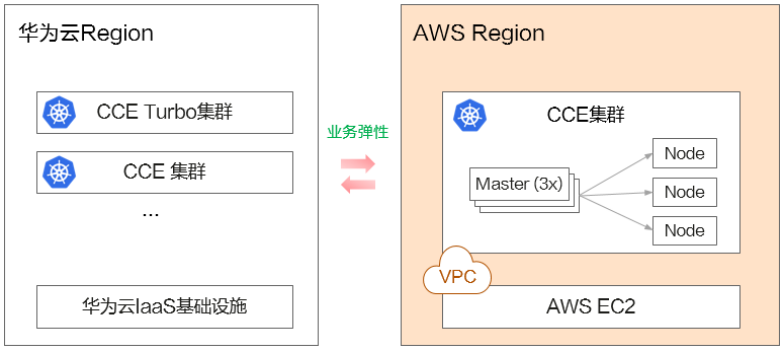
1.5 多云集群

1.5.1 多云集群概述

多云集群是指由UCS所提供，在第三方云服务供应商（如AWS）基础设施上运行的Kubernetes集群，其本质是在AWS或Azure基础设施上构建CCE集群。借助多云集群，您可以使用统一的API、工具和配置，将工作负载部署到多个云中，实现多云环境下一致的管理体验。

说明

目前仅支持在AWS基础设施上构建多云集群。



接入网络模式

多云集群仅支持公网接入方式，公网接入具有弹性灵活、成本低、易接入的优点。

1.5.2 安装多云集群的业务规划

1.5.2.1 基础软件规划

节点的操作系统、内核版本等基础软件需要符合表1中的版本要求。

表 1-18 基础软件规划

系统架构	系统类型	网络模型支持	操作系统版本	内核版本限制
x86	Ubuntu 20.04	Cilium	检查命令：**cat /etc/lsb-release** DISTRIB_DESCRIPTION="Ubuntu 20.04.4 LTS"	检查命令：**uname -r** 5.15.0-1017-aws

说明

- Cilium是一种网络插件，支持BGP、eBPF等网络协议，更多内容请参见Cilium官方文档。
- 多云集群采用containerd作为容器引擎，如果节点操作系统已经安装containerd、runC组件，UCS将直接使用上述组件。

1.5.2.2 数据规划

在AWS基础设施上构建多云集群时，将自动在AWS控制台创建以下资源，请确保资源配额足够：

表 1-19 资源数量

资源类型	EC2	NAT	VPC	子网	路由表	互联网网关	弹性IP	安全组	网络ACL	ELB	网络接口	存储卷

数量	3台	3个	1个	6个	7个	1个	3个	5个	1个	1个	4个	6块

表 1-20 EC2 资源规格

节点类型	数量	CPU (Cores)	Mem (GiB)	root盘	非root盘	备注
集群管理节点	3	8	32	100	200	t3.2xlarge型号
集群计算节点	按需	8	32	100	200	数量按需可扩展

表 1-21 IAM 权限

权限类型	权限名称
IAMRole	AWSIAMRoleNodes、AWSIAMRoleControlPlane、AWSIAMRoleControllers
IAMInstanceProfile	AWSIAMInstanceProfileNodes、AWSIAMInstanceProfileControlPlane、AWSIAMInstanceProfileControllers
IAMManagedPolicy	AWSIAMManagedPolicyCloudProviderNodes、AWSIAMManagedPolicyCloudProviderControlPlane、AWSIAMManagedPolicyControllers

1.5.3 注册多云集群

本章节将介绍UCS on AWS集群的注册过程。注册完成后，集群将通过公网自动接入UCS。

约束与限制

仅**华为云账号**以及拥有AWS账号权限的用户可进行集群注册的操作。

前提条件

- 已在UCS控制台申请多云集群试用。
- UCS集群配额充足，AWS资源配额充足。
- 已在AWS控制台创建访问密钥。具体操作请参见**如何获取访问密钥AK/SK**。

操作步骤

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击多云集群选项卡中的“注册集群”按钮。

步骤3 参考下表填写待添加集群的基础信息，其中带“*”的参数为必填参数。

表 1-22 注册集群基础信息配置

参数	参数说明
集群类型*	选择“多云集群”
云资源提供商*	选择“AWS”
集群名称*	输入集群的自定义名称，需以小写字母开头，由小写字母、数字、中划线（-）组成，且不能以中划线（-）结尾。
所属区域*	选择集群所在的区域，为对应的AWS的Region。请确保所选Region下的资源配额充足。
集群版本*	选择“1.23”
高可用*	选择“是”。系统将自动创建3个EC2作为集群控制节点。
集群标签	非必填项，以键值对的形式为集群添加标签，可以通过标签实现集群的分类。键值对可自定义，以字母或者数字开头和结尾，由字母、数字、连接符（-）、下划线（_）、点号（.）组成，且63个字符之内。
容器舰队	选择集群所属的舰队。 舰队用于权限精细化管理，一个集群只能加入一个舰队。若不选择舰队，集群注册成功后将显示在“未加入舰队的集群”页签下，后续还可以再添加至舰队中。 不支持在注册集群阶段选择已开通集群联邦能力的舰队，如果一定要加入这个舰队，请在集群注册成功后，再添加到该舰队中。关于集群联邦的介绍，请参见**开通集群联邦**章节。 如需新建舰队，请参见**管理容器舰队**。
访问密钥ID*	AWS IAM处获取的访问密钥ID，即AccessKeyID。
私有访问密钥*	AWS IAM处获取的私有访问密钥，即SecretAccessKey。
容器网段*	创建的Kubernetes集群的容器网段。
服务网段	创建的Kubernetes集群的服务网段。

步骤4 单击“确定”，集群注册成功后，等待自动接入。

----结束

1.6 单集群管理

1.6.1 单集群管理概述

UCS容器集群管理控制台提供了标准Kubernetes集群的统一管理功能，为您提供独立的集群操作入口，支持对某个集群单独进行管理。

- 对于华为云集群（CCE Standard和CCE Turbo集群），UCS中的集群控制台的功能与CCE控制台保持一致，管理CCE集群的操作指导请参见**CCE用户指南**。
- 对于附着集群、本地集群和多云集群，UCS中的集群控制台提供基本的Kubernetes资源管理能力，例如管理集群内的节点、工作负载、服务与路由、容器存储、配置项与密钥、命名空间等资源。

须知

附着集群和本地集群的控制台权限由UCS进行单独管理，需要**华为云账号**或具备**UCS FullAccess**权限的用户在UCS控制台的“权限管理”页面进行配置。

如何进入集群的管理控制台

舰队中的集群和未加入舰队的集群，进入集群控制台的方法不同，分别如下所述：

- 舰队中的集群：在“容器舰队”页面选择目标集群所在的舰队，单击舰队名称进入详情页，选择左侧导航栏的“容器集群”，单击目标集群名称进入集群控制台。
- 未加入舰队的集群：在“未加入舰队的集群”页面找到目标集群，单击集群名称后进入集群控制台。

1.6.2 节点管理

1.6.2.1 查看集群中节点

将集群接入UCS后，您可在集群控制台查看集群中的节点信息。

操作步骤

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“节点管理”，查看集群中的节点信息。

步骤3 单击操作列的“查看实例列表”，可查看运行在当前节点上的所有实例。

步骤4 单击操作列的“事件”，可查看节点事件。

步骤5 单击操作列的“更多 > 禁止调度”，可将节点快速设置为不可调度，新建Pod将无法调度至该节点。关于更多的污点设置，请参见**为节点添加标签/污点**。

----结束

1.6.2.2 为节点添加标签/污点

UCS支持为节点打上不同的标签，以定义节点的不同属性，通过这些标签，您可以快速的了解各个节点的特点。

污点（Taint）能够使节点排斥某些特定的Pod，从而避免将Pod调度到该节点上，通过添加污点，您可以实现各节点负载的合理分配。

节点标签使用场景

节点标签的主要使用场景有两类。

- 节点分类：通过添加标签对节点进行分类。
- 工作负载与节点的亲和与反亲和：
  - 有的工作负载需要的CPU大，有的工作负载需要的内存大，有的工作负载需要IO大，可能会影响其他工作负载正常工作等等，此时建议给节点添加不同标签。在部署工作负载的时候，就可以选择相应标签的节点亲和部署，保证系统正常工作；反之，可以使用节点的反亲和部署。
  - 一个系统可以分为多个模块，每个模块由多个微服务组成，为保证后期运维的高效，可以将节点打上对应模块的标签，让各模块的的工作负载部署到各自的节点上，互不干扰、利于维护。

节点固有标签

创建节点后，UCS会为节点添加固有标签，这些标签是无法编辑和删除的。节点固有标签的含义请参见**表1-23**。

表 1-23 节点固有标签

键	值
failure-domain.beta.kubernetes.io/region	表示节点当前所在区域
failure-domain.beta.kubernetes.io/zone	表示节点所在区域的可用区
beta.kubernetes.io/arch	表示节点处理器架构 例如：amd64，表示AMD64位的处理器
beta.kubernetes.io/os	表示节点的操作系统 例如：linux，表示Linux操作系统
kubernetes.io/availablezone	表示节点所在区域的可用区
kubernetes.io/hostname	表示节点主机名称
os.architecture	表示节点处理器架构 例如：amd64，表示AMD64位的处理器
os.name	表示节点的操作系统名称 例如：EulerOS_2.0_SP2，表示欧拉2.2的版本
os.version	表示节点内核版本

污点（Taints）说明

污点格式为“Key=Value:Effect”，Key和Value作为污点的标签，Value可以为空，Effect用于描述污点的效果。当前Effect支持如下几个效果。

- NoSchedule：不能容忍此污点的 Pod 不会被调度到节点上，但是现有 Pod 不会从节点中逐出。
- NoExecute：表示不能容忍此污点的 Pod 不会被调度到节点上，同时会将节点上已存在的Pod驱逐。

容忍度（Toleration）说明

容忍度应用于Pod上，允许（但并不要求）Pod 调度到带有与之匹配的污点的节点上。

污点和容忍度相互配合，可以用来避免 Pod 被分配到不合适的节点上。每个节点上都可以拥有一个或多个污点，而对这些污点没有设置容忍度的Pod，将不会被调度到该节点上。

在 Pod 中设置容忍度的示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

上面示例中表示节点上存在键名为“key1”，键值为“value1”，且效果为“NoSchedule”的污点时，该Pod能够调度到节点上。

容忍度还可以按如下方式进行设置，表示当节点上存在键名为“key1”，且效果为“NoSchedule”的污点时，该Pod也可以调度到节点上。

```
tolerations:
- key: "key1"
  operator: "Exists"
  effect: "NoSchedule"
```

管理节点标签/污点

步骤1 登录集群控制台。

步骤2 在左侧导航栏中单击“节点管理”，在节点列表中选择节点，并单击“标签与污点管理”。

步骤3 单击⊕按钮，设置节点标签/污点。如需执行多项操作，可多次添加，最多支持10条操作。

图 1-27 添加标签/污点

- 选择“添加”或“删除”操作。
- 选择操作对象为“K8S标签”或“污点（Taints）”。
- 填写需要增加标签/污点的“键”和“值”。
- 如选择操作对象为“污点（Taints）”，需选择污点效果，关于污点效果说明请参见**污点（Taints）说明**。

步骤4 单击“确定”，对所选节点执行标签/污点操作。

----结束

1.6.2.3 创建与删除节点（仅多云集群）

查看集群中节点

集群添加到UCS后，可在集群控制台查看集群中节点信息。

步骤1 登录集群控制台，单击目标集群名称进入集群详情页。

步骤2 在左侧导航栏中单击“节点管理”，查看集群中的节点信息。

步骤3 单击操作列的“查看实例列表”，可查看运行在当前节点上的所有实例。

步骤4 单击操作列的“事件”，可查看节点事件。

步骤5 单击操作列的“更多 > 禁止调度”，可将节点快速设置为不可调度，新建Pod将无法调度至该节点。

----结束

创建节点

步骤1 登录集群控制台，单击目标集群名称进入集群详情页。

步骤2 在左侧导航栏中单击“节点管理”，单击右上角“创建节点”。

步骤3 输入节点名称，选择需要的节点规格，可根据实际情况选择磁盘大小及数据盘数量。

步骤4 配置完成后，单击“下一步：规格确认”。

步骤5 规格确认无误后单击“提交”；如有问题，单击“上一步”返回修改。

说明

- 节点创建过程中，可用区、节点类型、容器引擎、操作系统、系统盘及数据盘类型都不可选。
- 数据盘可分局实际情况增加数量，最多可额外增加4块。
- 数据盘和系统盘最小默认大小为100GB。

----结束

删除节点

步骤1 登录集群控制台，单击目标集群名称进入集群详情页。

步骤2 在左侧导航栏中单击“节点管理”，进入节点管理界面。

步骤3 选择多个需要删除的节点，单击“节点名称”上方“更多”，选择“删除”，即可批量删除节点。

步骤4 如果需要删除单个节点，单击该节点后方的“更多”，选择“删除”。

步骤5 单击“是”，即可删除节点。

----结束

1.6.3 工作负载

1.6.3.1 无状态负载

工作负载即Kubernetes对一组Pod的抽象模型，用于描述业务的运行载体，包括Deployment、StatefulSet、Job、DeamonSet等。

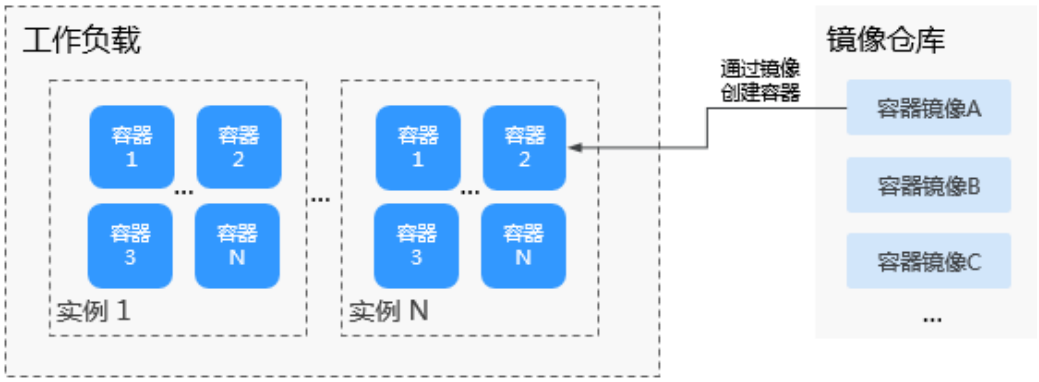
工作负载基本概念

- 无状态工作负载（即Kubernetes中的Deployment）：实例之间完全独立、功能相同，具有弹性伸缩、滚动升级等特性。如：nginx、wordpress，创建无状态工作负载请参见**创建无状态工作负载**。
- 有状态工作负载（即Kubernetes中的StatefulSet）：实例之间不完全独立，具有稳定的持久化存储和网络标示，以及有序的部署、收缩和删除等特性。如：mysql-HA、etcd，创建有状态工作负载请参见**创建有状态工作负载**。
- 守护进程集（即Kubernetes中的DeamonSet）：在集群的每个节点上运行一个Pod，且保证只有一个Pod，适合一些系统层面的应用，如日志收集、资源监控等，创建守护进程集请参见**创建守护进程集**。

工作负载与容器间的关系

如**图1-28**所示，一个工作负载由一个或多个实例（Pod）组成。一个实例由一个或多个容器组成，每个容器都对应一个容器镜像。对于无状态工作负载，实例都是完全相同的。

图 1-28 工作负载与容器的关系



工作负载生命周期说明

表 1-24 状态说明

状态	说明
运行中	所有实例都处于运行中才是运行中。
未就绪	所有实例处于pending状态。
升级中	触发升级动作后，工作负载会处于升级中。
可用	当多实例无状态工作负载运行过程中部分实例异常，可用实例不为0，工作负载会处于可用状态。
删除中	触发删除操作后，工作负载会处于删除中状态。

创建无状态工作负载

步骤1 （可选）若基于我的镜像创建工作负载，用户首先需要将镜像上传至容器镜像服务，上传镜像的方式请参见**镜像管理**。若基于开源镜像中心创建工作负载，则无需上传镜像。

步骤2 在集群控制台选择“工作负载 > 无状态负载”，单击“镜像创建”。

步骤3 参照**表1-25**设置基本信息，其中带“*”标志的参数为必填参数。

表 1-25 工作负载基本信息

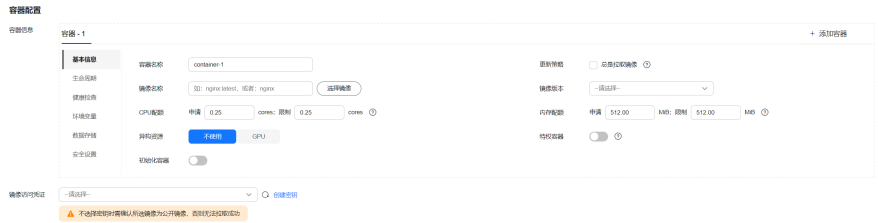
参数	参数说明
* 负载名称	新建工作负载的名称，命名必须唯一。
集群名称	新建工作负载所在的集群，无需填写。
* 命名空间	在单集群中，不同命名空间中的数据彼此隔离。使应用可以共享同个集群的服务，也能够互不干扰。若您不设置命名空间，系统会默认使用default命名空间。

参数	参数说明
* 实例数量	工作负载的实例数量。工作负载可以有一个或多个实例，用户可以设置具体实例个数，默认为2，可自定义设置为1。 每个工作负载实例都由相同的容器部署而成。设置多个实例主要用于实现高可靠性，当某个实例故障时，工作负载还能正常运行。若使用单实例，节点异常或实例异常会导致服务异常。
描述	工作负载描述信息。
时区同步	容器将和节点使用相同时区。时区同步功能开启后，在“数据存储 > 本地磁盘”中，将会自动添加HostPath类型的磁盘，请勿修改删除该磁盘。

步骤4 设置工作负载容器配置。

Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器并分别进行设置。

图 1-29 容器配置



- 容器信息：Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。
  - 基本信息：请参见**表1-26**。

表 1-26 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 ■ 我的镜像：当前区域下镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 ■ 镜像中心：开源镜像仓库中的官方镜像。 ■ 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。

参数	说明
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	■ 申请：容器需要使用的最小CPU值，默认0.25Core。 ■ 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	■ 申请：容器需要使用的内存最小值，默认512MiB。 ■ 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。
特权容器	特权容器是指容器里面的程序具有一定的特权。 若选中，容器将获得超级权限，例如可以操作宿主机上面的网络设备、修改内核参数等。

  - 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
  - 健康检查：设置健康检查可以在容器运行过程中定时检查容器的健康状况，详情请参见**设置容器健康检查**。
  - 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
  - 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。
  - 安全设置：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。
- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

步骤5 （可选）单击服务配置栏的＋，进行工作负载服务配置。

若工作负载需要和其它服务互访，或需要被公网访问，您需要添加服务（Service），设置访问方式。工作负载访问的方式决定了这个工作负载的网络属性，不同访问方式的工作负载可以提供不同网络能力，操作详情请参见**服务**。

您也可以在创建完工作负载之后再创建Service，参见**集群内访问（ClusterIP）**和**节点访问（NodePort）**。

- Service名称：新增服务名称，用户可自定义，服务名称必须唯一。
- 访问类型：
  - 集群内访问（ClusterIP）：只能集群内访问服务。
  - 节点访问（NodePort）：可以通过集群内任意节点访问到服务。
  - 负载均衡（LoadBalancer）：通过弹性负载均衡从公网访问到工作负载。
- 服务亲和（仅节点访问、负载均衡设置）：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- 端口配置：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口（仅节点访问设置）：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。
- 注解：支持key/value对格式，请您根据自身业务以及厂家要求进行注解配置，填写完成后单击“添加”。

步骤6 （可选）单击“展开高级配置”，设置工作负载高级配置。

- 升级策略：指定无状态负载的升级方式，包括整体替换升级和逐步滚动升级，详细参数说明请参见**工作负载升级配置**。
  - 滚动升级：滚动升级将逐步用新版本的实例替换旧版本的实例，升级的过程中，业务流量会同时负载均衡分布到新的和旧的实例上，因此业务不会中断。
  - 替换升级：先删除旧实例，再创建新实例。升级过程中业务会中断。
- 调度策略：您可设置亲和（affinity）与反亲和（anti-affinity）实现Pod的计划性调度，详细信息请参见**调度策略（亲和与反亲和）**。
- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。
- 容忍策略：当工作负载实例所在的节点不可用时，系统将实例重新调度到其它可用节点的时间窗，默认为300秒。
  - 容忍策略与节点的污点能力配合使用，允许（不强制）负载调度到带有与之匹配的污点的节点上，也可用于控制负载所在的节点被标记污点后负载的驱逐策略，详细内容请参考**示例教程**。
  - 单击“容忍策略”下的 ＋，可新增策略。相关参数说明见**容忍策略**。

步骤7 配置完成后，单击“创建工作负载”。返回无状态工作负载列表查看工作负载状态。

在工作负载列表中，待工作负载状态为“运行中”，工作负载创建成功。

----结束

相关操作

通过集群控制台，您还可以执行表1-27中的操作。

表 1-27 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建工作负载。
查看Pod详情	单击已创建工作负载的名称，在“实例列表”页签下，可以查看Pod相关详情信息。 ● 事件：可以设置查询条件，比如设置事件产生的时间段或搜索事件名称，查看相关事件。 ● 容器列表：可以查看相应实例的容器名称、状态、镜像以及启停次数等。 ● 查看YAML：可以查看相应实例的YAML文件。
编辑YAML	单击工作负载名称后的“编辑YAML”，可查看并编辑当前工作负载的YAML文件。
升级	1. 单击工作负载名称后的“升级”。 2. 更改工作负载信息。 3. 单击“升级工作负载”提交已修改的信息。
回退	单击工作负载名称后的“回退”，选择一个历史版本进行回退。
重新部署	单击工作负载名称后的“更多 > 重新部署”，并单击“是”进行确认，重新部署将重启负载下的全部容器组Pod。
关闭升级	单击工作负载名称后的“更多 > 关闭升级”，并单击“是”进行确认。 ● 工作负载标记为“升级已关闭”后，对其进行的升级将不会被应用到实例。 ● 正在进行中的滚动升级也会被暂停。
删除	单击工作负载名称后的“更多 > 删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的工作负载。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

1.6.3.2 有状态负载

创建有状态工作负载

步骤1 （可选）若基于我的镜像创建工作负载，用户首先需要将镜像上传至容器镜像服务，上传镜像的方式请参见**镜像管理**。若基于开源镜像中心创建工作负载，则无需上传镜像。

步骤2 在集群控制台选择“工作负载 > 有状态负载”，单击“镜像创建”。

步骤3 参照**表1-28**设置基本信息，其中带“*”标志的参数为必填参数。

表 1-28 工作负载基本信息

参数	参数说明
* 负载名称	新建工作负载的名称，命名必须唯一。
集群名称	新建工作负载所在的集群，无需填写。
* 命名空间	在单集群中，不同命名空间中的数据彼此隔离。使应用可以共享同个集群的服务，也能够互不干扰。若您不设置命名空间，系统会默认使用default命名空间。
* 实例数量	工作负载的实例数量。工作负载可以有一个或多个实例，用户可以设置具体实例个数，默认为2，可自定义设置为1。 每个工作负载实例都由相同的容器部署而成。设置多个实例主要用于实现高可靠性，当某个实例故障时，工作负载还能正常运行。若使用单实例，节点异常或实例异常会导致服务异常。
描述	工作负载描述信息。
时区同步	容器将和节点使用相同时区。时区同步功能开启后，在“数据存储 > 本地磁盘”中，将会自动添加HostPath类型的磁盘，请勿修改删除该磁盘。

步骤4 设置工作负载容器配置。

Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器并分别进行设置。

图 1-30 容器配置

- 容器信息：Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。
  - 基本信息：请参见**表1-29**。

表 1-29 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 ■ 我的镜像：当前区域下镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 ■ 镜像中心：开源镜像仓库中的官方镜像。 ■ 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	■ 申请：容器需要使用的最小CPU值，默认0.25Core。 ■ 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	■ 申请：容器需要使用的内存最小值，默认512MiB。 ■ 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。
特权容器	特权容器是指容器里面的程序具有一定的特权。 若选中，容器将获得超级权限，例如可以操作宿主机上面的网络设备、修改内核参数等。

- 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
- 健康检查：设置健康检查可以在容器运行过程中定时检查容器的健康状况，详情请参见**设置容器健康检查**。
- 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
- 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异

常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。

- 安全设置：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。

- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

步骤5 设置工作负载实例间发现服务配置。

StatefulSet实例间发现通过无头服务（Headless Service）实现，无头服务并不会分配Cluster IP，并且查询会返回所有Pod的DNS记录，这样就可查询到所有Pod的IP地址。

- Service名称：输入工作负载所对应的服务名称，用于集群内工作负载间的互相访问。该服务主要用于实例的内部发现，不需要有单独的IP地址，也不需要做负载均衡。
- 端口配置：
  - 端口名称：端口名称用于给容器端口命名，通常以端口用途命名。
  - 服务端口：输入服务端口。
  - 容器端口：输入容器的监听端口。

步骤6 （可选）单击服务配置栏的＋，进行工作负载服务配置。

若工作负载需要和其它服务互访，或需要被公网访问，您需要添加服务（Service），设置访问方式。工作负载访问的方式决定了这个工作负载的网络属性，不同访问方式的工作负载可以提供不同网络能力，操作详情请参见**服务**。

您也可以在创建完工作负载之后再创建Service，参见**集群内访问（ClusterIP）**和**节点访问（NodePort）**。

- Service名称：新增服务名称，用户可自定义，服务名称必须唯一。
- 访问类型：
  - 集群内访问（ClusterIP）：只能集群内访问服务。
  - 节点访问（NodePort）：可以通过集群内任意节点访问到服务。
  - 负载均衡（LoadBalancer）：通过弹性负载均衡从公网访问到工作负载。
- 服务亲和（仅节点访问、负载均衡设置）：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- 端口配置：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口（仅节点访问设置）：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。

    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。
- 注解：支持key/value对格式，请您根据自身业务以及厂家要求进行注解配置，填写完成后单击“添加”。

步骤7 （可选）单击“展开高级配置”，设置工作负载高级配置。

- 升级策略：指定有状态负载的升级方式，包括整体替换升级和逐步滚动升级，详细参数说明请参见**工作负载升级配置**。
  - 滚动升级：滚动升级将逐步用新版本的实例替换旧版本的实例，升级的过程中，业务流量会同时负载均衡分布到新的和旧的实例上，因此业务不会中断。
  - 替换升级：有状态工作负载的替换升级，需要手动删除旧实例，再创建新实例。升级过程中业务会中断。
- 实例管理策略：
  - 有序策略：默认实例管理策略，有状态负载会逐个的、按顺序的进行部署、删除、伸缩实例, 只有前一个实例部署Ready或者删除完成后，有状态负载才会操作后一个实例。
  - 并行策略：支持有状态负载并行创建或者删除所有的实例，有状态负载发生变更时立刻在实例上生效。
- 调度策略：您可设置亲和（affinity）与反亲和（anti-affinity）实现Pod的计划性调度，详细信息请参见**调度策略（亲和与反亲和）**。
- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。
- 容忍策略：当工作负载实例所在的节点不可用时，系统将实例重新调度到其它可用节点的时间窗，默认为300秒。
  - 容忍策略与节点的污点能力配合使用，允许（不强制）负载调度到带有与之匹配的污点的节点上，也可用于控制负载所在的节点被标记污点后负载的驱逐策略，详细内容请参考**示例教程**。
  - 单击“容忍策略”下的 ，可新增策略。相关参数说明见**容忍策略**。

步骤8 配置完成后，单击“创建工作负载”。返回有状态工作负载列表查看工作负载状态。

在工作负载列表中，待工作负载状态为“运行中”，工作负载创建成功。

----结束

相关操作

通过集群控制台，您还可以执行**表1-30**中的操作。

表 1-30 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建工作负载。

操作	说明
查看Pod详情	单击已创建工作负载的名称，在“实例列表”页签下，可以查看Pod相关详情信息。 ● 事件：可以设置查询条件，比如设置事件产生的时间段或搜索事件名称，查看相关事件。 ● 容器列表：可以查看相应实例的容器名称、状态、镜像以及启停次数等。 ● 查看YAML：可以查看相应实例的YAML文件。
编辑YAML	单击工作负载名称后的“编辑YAML”，可查看并编辑当前工作负载的YAML文件。
升级	1. 单击工作负载名称后的“升级”。 2. 更改工作负载信息。 3. 单击“升级工作负载”提交已修改的信息。
回退	单击工作负载名称后的“回退”，选择一个历史版本进行回退。
重新部署	单击工作负载名称后的“更多 > 重新部署”，并单击“是”进行确认，重新部署将重启负载下的全部容器组Pod。
关闭升级	单击工作负载名称后的“更多 > 关闭升级”，并单击“是”进行确认。 ● 工作负载标记为“升级已关闭”后，对其进行的升级将不会被应用到实例。 ● 正在进行中的滚动升级也会被暂停。
删除	单击工作负载名称后的“更多 > 删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的工作负载。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

1.6.3.3 守护进程集

创建守护进程集

步骤1 （可选）若基于我的镜像创建工作负载，用户首先需要将镜像上传至容器镜像服务，上传镜像的方式请参见**镜像管理**。若基于开源镜像中心创建工作负载，则无需上传镜像。

步骤2 在集群控制台选择“工作负载 > 守护进程集”，单击“镜像创建”。

步骤3 参照**表1-31**设置基本信息，其中带“*”标志的参数为必填参数。

表 1-31 工作负载基本信息

参数	参数说明
* 名称	新建工作负载的名称，命名必须唯一。
集群名称	新建工作负载所在的集群，无需填写。
* 命名空间	在单集群中，不同命名空间中的数据彼此隔离。使应用可以共享同个集群的服务，也能够互不干扰。若您不设置命名空间，系统会默认使用default命名空间。
描述	工作负载描述信息。
时区同步	容器将和节点使用相同时区。时区同步功能开启后，在“数据存储 > 本地磁盘”中，将会自动添加HostPath类型的磁盘，请勿修改删除该磁盘。

步骤4 设置工作负载容器配置。

Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器并分别进行设置。

图 1-31 容器配置

- 容器信息：Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。
  - 基本信息：请参见**表1-32**。

表 1-32 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 ■ 我的镜像：当前区域下镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 ■ 镜像中心：开源镜像仓库中的官方镜像。 ■ 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。

参数	说明
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	■ 申请：容器需要使用的最小CPU值，默认0.25Core。 ■ 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	■ 申请：容器需要使用的内存最小值，默认512MiB。 ■ 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。
特权容器	特权容器是指容器里面的程序具有一定的特权。 若选中，容器将获得超级权限，例如可以操作宿主机上面的网络设备、修改内核参数等。

  - 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
  - 健康检查：设置健康检查可以在容器运行过程中定时检查容器的健康状况，详情请参见**设置容器健康检查**。
  - 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
  - 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。
  - 安全设置：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。
- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

步骤5 （可选）单击服务配置栏的＋，进行工作负载服务配置。

若工作负载需要和其它服务互访，或需要被公网访问，您需要添加服务（Service），设置访问方式。工作负载访问的方式决定了这个工作负载的网络属性，不同访问方式的工作负载可以提供不同网络能力，操作详情请参见**服务**。

您也可以在创建完工作负载之后再创建Service，参见**集群内访问（ClusterIP）**和**节点访问（NodePort）**。

- Service名称：新增服务名称，用户可自定义，服务名称必须唯一。
- 访问类型：
  - 集群内访问（ClusterIP）：只能集群内访问服务。
  - 节点访问（NodePort）：可以通过集群内任意节点访问到服务。
  - 负载均衡（LoadBalancer）：通过弹性负载均衡从公网访问到工作负载。
- 服务亲和（仅节点访问、负载均衡设置）：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- 端口配置：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口（仅节点访问设置）：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。
- 注解：支持key/value对格式，请您根据自身业务以及厂家要求进行注解配置，填写完成后单击“添加”。

步骤6 （可选）单击“展开高级配置”，设置工作负载高级配置。

- 升级策略：指定守护进程集的升级方式，包括整体替换升级和逐步滚动升级，详细参数说明请参见**工作负载升级配置**。
  - 滚动升级：滚动升级将逐步用新版本的实例替换旧版本的实例，升级的过程中，业务流量会同时负载均衡分布到新的和旧的实例上，因此业务不会中断。
  - 替换升级：守护进程集的替换升级，需要手动删除旧实例，再创建新实例。升级过程中业务会中断。
- 调度策略：您可设置亲和（affinity）与反亲和（anti-affinity）实现Pod的计划性调度，详细信息请参见**调度策略（亲和与反亲和）**。
- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。
- 容忍策略：当工作负载实例所在的节点不可用时，系统将实例重新调度到其它可用节点的时间窗，默认为300秒。
  - 容忍策略与节点的污点能力配合使用，允许（不强制）负载调度到带有与之匹配的污点的节点上，也可用于控制负载所在的节点被标记污点后负载的驱逐策略，详细内容请参考**示例教程**。
  - 单击“容忍策略”下的 ，可新增策略。相关参数说明见**容忍策略**。

步骤7 配置完成后，单击“创建工作负载”。返回守护进程集工作负载列表查看工作负载状态。

在工作负载列表中，待工作负载状态为“运行中”，工作负载创建成功。

----结束

相关操作

通过集群控制台，您还可以执行表1-33中的操作。

表 1-33 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建工作负载。
查看Pod详情	单击已创建工作负载的名称，在“实例列表”页签下，可以查看Pod相关详情信息。 ● 事件：可以设置查询条件，比如设置事件产生的时间段或搜索事件名称，查看相关事件。 ● 容器列表：可以查看相应实例的容器名称、状态、镜像以及启停次数等。 ● 查看YAML：可以查看相应实例的YAML文件。
编辑YAML	单击工作负载名称后的“编辑YAML”，可查看并编辑当前工作负载的YAML文件。
升级	1. 单击工作负载名称后的“升级”。 2. 更改工作负载信息。 3. 单击“升级工作负载”提交已修改的信息。
回退	单击工作负载名称后的“回退”，选择一个历史版本进行回退。
重新部署	单击工作负载名称后的“更多 > 重新部署”，并单击“是”进行确认，重新部署将重启负载下的全部容器组Pod。
关闭升级	单击工作负载名称后的“更多 > 关闭升级”，并单击“是”进行确认。 ● 工作负载标记为“升级已关闭”后，对其进行的升级将不会被应用到实例。 ● 正在进行中的滚动升级也会被暂停。
删除	单击工作负载名称后的“更多 > 删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的工作负载。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

1.6.3.4 任务和定时任务

任务概述

任务管理对应Kubernetes中的Job，分为普通任务和定时任务。

普通任务（Job）是Kubernetes用来控制批处理型任务的资源对象。批处理业务与长期伺服业务（Deployment、StatefulSet）的主要区别是批处理业务的运行有头有尾，而长期伺服业务在用户不停止的情况下永远运行。Job管理的Pod根据用户的设置把任务成功完成就自动退出了。成功完成的标志根据不同的spec.completions策略而不同：单Pod型任务有一个Pod成功就标志成功。定数成功型任务保证有N个任务全部成功标志成功。工作队列型任务根据应用确认的全局成功而标志成功。

定时任务（CronJob）即定时任务，是基于时间的Job，类似于Linux系统的crontab，在指定的时间周期运行指定的Job，即：

- 在给定时间点只运行一次。
- 在给定时间点周期性地运行。

CronJob的典型用法如下所示：

- 在给定的时间点调度Job运行。
- 创建周期性运行的Job，例如数据库备份、发送邮件。

创建普通任务

普通任务可用于创建仅执行一次的批处理任务，任务执行完成后会自动退出。使用场景为在创建工作负载前，执行普通任务，将镜像上传至镜像仓库。

步骤1 （可选）普通任务需要基于镜像创建，若选择私有镜像，用户首先需要将镜像上传至镜像仓库。

步骤2 登录集群控制台，在左侧导航栏中选择“工作负载”，选择“普通任务”页签，并单击右上角“镜像创建”。

步骤3 配置工作负载的信息。

基本信息

- 负载类型：选择普通任务Job。
- 负载名称：填写工作负载的名称。
- 命名空间：选择工作负载的命名空间，默认为default。您可以单击后面的“创建命名空间”，命名空间的详细介绍请参见**创建命名空间**。
- 实例数量：填写实例的数量，也就是Pod的数量。

容器配置

- 容器信息：Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。
  - 基本信息：请参见**表1-34**。

表 1-34 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 ■ 我的镜像：当前区域下镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 ■ 镜像中心：开源镜像仓库中的官方镜像。 ■ 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	■ 申请：容器需要使用的最小CPU值，默认0.25Core。 ■ 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	■ 申请：容器需要使用的内存最小值，默认512MiB。 ■ 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在Pod内的应用容器启动之前运行。详细说明请参见**Init 容器**。
特权容器	特权容器是指容器里面的程序具有一定的特权。 若选中，容器将获得超级权限，例如可以操作宿主机上面的网络设备、修改内核参数等。

- 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
- 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
- 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。

- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

高级配置

- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。
- 任务设置：
  - 并行数：任务负载执行过程中允许同时创建的最大实例数，并行数应不大于实例数。
  - 超时时间：当任务执行超出该时间时，任务将会被标识为执行失败，任务下的所有实例都会被删除。为空时表示不设置超时时间。

步骤4 任务创建完成后，在“普通任务”列表中可查看已创建的普通任务。

待状态为“处理中”，普通任务创建成功。

----结束

创建定时任务

定时任务可用于创建按照设定时间和周期重复执行的任务，任务执行完成后会自动退出。使用场景为在某个固定时间点，为所有运行中的节点做时间同步。

步骤1 （可选）定时任务需要基于镜像创建，若选择私有镜像，用户首先需要将镜像上传至镜像仓库。

步骤2 登录集群控制台，在左侧导航栏中选择“工作负载”，选择“定时任务”页签，并单击右上角“创建负载”。

步骤3 配置工作负载的信息。

基本信息

- 负载类型：选择定时任务CronJob。
- 负载名称：填写工作负载的名称。
- 命名空间：选择工作负载的命名空间，默认为default。您可以单击后面的“创建命名空间”，命名空间的详细介绍请参见**创建命名空间**。

容器配置

- 容器信息：Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器。
  - 基本信息：请参见**表1-35**。

表 1-35 基本信息参数说明

参数	说明
容器名称	为容器命名。

参数	说明
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 ■ 我的镜像：当前区域下镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 ■ 镜像中心：开源镜像仓库中的官方镜像。 ■ 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	■ 申请：容器需要使用的最小CPU值，默认0.25Core。 ■ 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	■ 申请：容器需要使用的内存最小值，默认512MiB。 ■ 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。
特权容器	特权容器是指容器里面的程序具有一定的特权。 若选中，容器将获得超级权限，例如可以操作宿主机上面的网络设备、修改内核参数等。

  - 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
  - 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

定时规则

- 并发策略：支持如下三种模式：
  - Forbid：在前一个任务未完成时，不创建新任务。
  - Allow：定时任务不断新的任务，会抢占集群资源。

  - Replace：当到达新任务创建时间点，而前一个任务未完成时，新的任务会取代前一个任务。

- 定时规则：指定新建定时任务在何时执行，YAML中的定时规则通过CRON表达式实现。
  - 以固定周期执行定时任务，支持的周期单位为分钟、小时、日、月。例如，每30分钟执行一次任务，对应的CRON表达式为“*/30 * * * *”，执行时间将从单位范围内的0值开始计算，如**00:00:00**、**00:30:00**、**01:00:00**、**...**。
  - 以固定时间（按月）执行定时任务。例如，在每个月1日的0时0分执行任务，对应的CRON表达式为“0 0 1 */1 *”，执行时间为******-01-01 00:00:00**、******-02-01 00:00:00**、**...**。
  - 以固定时间（按周）执行定时任务。例如，在每周一的0时0分执行任务，对应的CRON表达式为“0 0 * * 1”，执行时间为******-**-01 周一 00:00:00**、******-**-08 周一 00:00:00**、**...**。
  - 自定义CRON表达式：关于CRON表达式的用法，可参考**CRON**。

    说明

    - 以固定时间（按月）执行定时任务时，在某月的天数不存在的情况下，任务将不会在该月执行。例如设置天数为30，而2月份没有30号，任务将跳过该月份，在3月30号继续执行。
    - 由于CRON表达式的定义，这里的固定周期并非严格意义的周期。将从0开始按周期对其时间单位范围（例如单位为分钟时，则范围为0~59）进行划分，无法整除时最后一个周期会被重置。因此仅在周期能够平均划分其时间单位范围时，才能表示准确的周期。

      举个例子，周期单位为小时，因为“/2、/3、/4、/6、/8和/12”可将24小时整除，所以可以表示准确的周期；而使用其他周期时，在新的一天开始时，最后一个周期将会被重置。比如CRON式为“* */12 * * *”时为准确的周期，每天的执行时间为**00:00:00**和**12:00:00**；而CRON式为“* */13 * * *”时，每天的执行时间为**00:00:00**和**13:00:00**，在第二天0时，虽然没到13个小时的周期还是会被刷新。

- 任务记录：可以设置保留执行成功或执行失败的任务个数，设置为0表示不保留。

高级配置

- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。

步骤4 任务创建完成后，在“定时任务”列表中可查看已创建的定时任务。

待状态为“已启动”，定时任务创建成功。

----结束

相关操作

- 事件：可以设置查询条件，比如设置事件产生的时间段或搜索事件名称，查看相关事件。
- 实例/任务列表：可以查看相应实例/任务的名称、状态等。
  - 事件：实例产生的事件信息，保存时间为1小时。
  - 实例列表：查看实例名称、状态、重启次数等。
  - 查看YAML：查看对应实例的YAML文件。
  - 删除：删除实例。
- 查看/编辑YAML：可以查看/编辑工作负载的YAML文件。

- 删除：删除任务。
- 停止（仅定时任务支持）：停止定时任务。

1.6.3.5 容器组

容器组（Pod）是Kubernetes中最小的可部署单元。一个Pod（容器组）包含了一个应用程序容器（某些情况下是多个容器）、存储资源、一个唯一的网络IP地址、以及一些确定容器该如何运行的选项。Pod容器组代表了Kubernetes中一个独立的应用程序运行实例，该实例可能由单个容器或者几个紧耦合在一起的容器组成。

YAML 创建

步骤1 登录集群控制台，选择“工作负载 > 容器组”，单击“YAML创建”。

步骤2 在弹出的“YAML创建容器组”窗中对YAML文件进行编辑。

步骤3 单击“确定”，完成YAML创建。

----结束

相关操作

- 事件：可以设置查询条件，比如设置事件产生的时间段或搜索事件名称，查看相关事件。
- 容器列表：可以查看相应实例的容器名称、状态、镜像以及启停次数等。
- 查看YAML：可以查看相应实例的YAML文件。

1.6.3.6 设置容器规格

操作场景

在创建工作负载时为添加的容器设置资源限制，可以对工作负载中每个实例所用的CPU配额、内存配额进行申请和限制。

配置含义

在CPU配额和内存配额设置中，申请值与限制值的含义如下：

- 申请值用于预分配资源，当集群中的节点没有申请值所要求的资源数量时，容器会创建失败。
- 限制值用于设置容器使用资源的最大上限，避免异常情况下节点资源消耗过多，避免极端场景下因节点资源耗尽导致已运行的容器业务被驱逐。

说明

创建工作负载时，建议设置CPU和内存的资源上下限。同一个节点上部署的工作负载，对于未设置资源上下限的工作负载，如果其异常资源泄露会导致其它工作负载分配不到资源而异常。未设置资源上下限的工作负载，工作负载监控信息也会不准确。

配置说明

- CPU配额：

表 1-36 CPU 配额说明

参数	说明
CPU申请	容器使用的最小CPU需求，作为容器调度时资源分配的判断依赖。只有当节点上可分配CPU总量 ≥ 容器CPU申请数时，才允许将容器调度到该节点。
CPU限制	容器能使用的CPU最大值。

建议配置方法：

节点的实际可用分配CPU量 ≥ 当前实例所有容器CPU限制值之和 ≥ 当前实例所有容器CPU申请值之和，节点的实际可用分配CPU量请在“资源管理 > 节点管理”中对应节点的“可分配资源”列下查看“CPU: ** Core”。

- 内存配额：

表 1-37 内存配额说明

参数	说明
内存申请	容器使用的最小内存需求，作为容器调度时资源分配的判断依赖。只有当节点上可分配内存总量 ≥ 容器内存申请数时，才允许将容器调度到该节点。
内存限制	容器能使用的内存最大值。当内存使用率超出设置的内存限制值时，该实例可能会被重启进而影响工作负载的正常使用。

建议配置方法：

节点的实际可用分配内存量 ≥ 当前节点所有容器内存限制值之和 ≥ 当前节点所有容器内存申请值之和，节点的实际可用分配内存量请在“资源管理 > 节点管理”中对应节点的“可分配资源”列下查看“内存: ** GiB”。

说明

可分配资源：可分配量按照实例请求值(request)计算，表示实例在该节点上可请求的资源上限，不代表节点实际可用资源。 计算公式为：

- 可分配CPU = CPU总量 - 所有实例的CPU请求值 - 其他资源CPU预留值
- 可分配内存 = 内存总量 - 所有实例的内存请求值 - 其他资源内存预留值

使用示例

以集群包含一个资源为4Core 8GB的节点为例，已经部署一个包含两个实例的工作负载到该集群上，并设置两个实例（实例1，实例2）的资源为{CPU申请，CPU限制，内存申请，内存限制}={1Core，2Core，2GB，2GB}。

那么节点上CPU和内存的资源使用情况如下：

- 节点CPU可分配量=4Core-（实例1申请的1Core+实例2申请的1Core）=2Core
- 节点内存可分配量=8GB-（实例1申请的2GB+实例2申请的2GB）=4GB

因此节点还剩余2Core 4GB的资源可供下一个新增的实例使用。

1.6.3.7 设置容器生命周期

操作场景

回调函数可以在容器的生命周期的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以注册相应的钩子函数。

目前提供的生命周期回调函数如下所示：

- **启动命令：**容器将会以该启动命令启动，请参见**启动命令**。
- **启动后处理：**容器启动后触发，请参见**启动后处理**。
- **停止前处理：**容器停止前触发。设置停止前处理，确保升级或实例删除时可提前将实例中运行的业务排水。详细请参见**停止前处理**。

启动命令

在默认情况下，镜像启动时会运行默认命令，如果想运行特定命令或重写镜像默认值，需要进行相应设置。

Docker的镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时将运行镜像制作时提供的默认的命令和参数，Docker将这两个字段定义为ENTRYPOINT和 CMD。

如果在创建工作负载时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令ENTRYPOINT、CMD，规则如下：

表 1-38 容器如何执行命令和参数

镜像 ENTRYPOINT	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

步骤1 在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“启动命令”页签，输入运行命令和运行参数。

表 1-39 容器启动命令

命令方式	操作步骤
运行命令	在容器中执行指定的命令。命令行可通过执行bash或执行二进制的方式实现，您可以参照示例配置需要执行的命令。 若运行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（""）。 **说明** 多命令时，运行命令建议用/bin/sh或其他的shell，其他全部命令作为参数来传入。
运行参数	输入控制容器运行命令参数，例如--port=8080。 若参数有多个，多个参数以换行分隔。

----结束

启动后处理

步骤1 在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“启动后处理”页签，设置启动后处理的参数。

表 1-40 启动后处理-参数说明

参数	说明

命令行方式	在容器中执行指定的命令。命令行可通过执行bash或执行二进制的方式实现，您可以参照示例配置需要执行的命令。
	命令的格式为Command Args[1] Args[2]…（Command为系统命令或者用户自定义可执行程序，如果未指定路径则在默认路径下寻找可执行程序），如果需要执行多条命令，建议采用将命令写入脚本执行的方式。**不支持后台执行和异步执行的命令。**
	如需要执行的命令如下：
	`exec:`
	`  command:`
	`  - /install.sh`
	`  - install_agent`
	请在执行脚本中填写: /install install_agent。这条命令表示容器创建成功后将执行install.sh。

HTTP请求方式	发起一个HTTP调用请求。配置参数如下： ● 路径：请求的URL路径，可选项。 ● 端口：请求的端口，必选项。 ● 主机地址：请求的IP地址，可选项，默认是容器所在的节点IP。

----结束

停止前处理

步骤1 在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“停止前处理”页签，设置停止前处理的命令。

表 1-41 停止前处理

参数	说明
命令行方式	在容器中执行指定的命令。命令行可通过执行bash或执行二进制的方式实现，您可以参照示例配置需要执行的命令。 命令的格式为Command Args[1] Args[2]…（Command为系统命令或者用户自定义可执行程序，如果未指定路径则在默认路径下寻找可执行程序），如果需要执行多条命令，建议采用将命令写入脚本执行的方式。 如需要执行的命令如下： exec: command: - /uninstall.sh - uninstall_agent 请在执行脚本中填写: /uninstall uninstall_agent。这条命令表示容器结束前将执行uninstall.sh。
HTTP请求方式	发起一个HTTP调用请求。配置参数如下： ● 路径：请求的URL路径，可选项。 ● 端口：请求的端口，必选项。 ● 主机地址：请求的IP地址，可选项，默认是容器所在的节点IP。

----结束

YAML 样例

本节以nginx为例，说明kubectl命令设置容器生命周期的方法。

在以下配置文件中，您可以看到postStart命令在容器目录/bin/bash下写了个**install.sh**命令。 preStop执行**uninstall.sh**命令。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        command:
        - sleep 3600                    #启动命令
```



```
    imagePullPolicy: Always
    lifecycle:
      postStart:
        exec:
          command:
          - /bin/bash
          - install.sh              #启动后命令
      preStop:
        exec:
          command:
          - /bin/bash
          - uninstall.sh            #停止前命令
    name: nginx
  imagePullSecrets:
  - name: default-secret
```

1.6.3.8 设置容器健康检查

操作场景

健康检查是指容器运行过程中，根据用户需要，定时检查容器健康状况。若不配置健康检查，如果容器内应用程序异常，Pod将无法感知，也不会自动重启去恢复。最终导致虽然Pod状态显示正常，但Pod中的应用程序异常的情况。

Kubernetes提供了三种健康检查的探针：

- **存活探针：**livenessProbe，用于检测容器是否正常，类似于执行ps命令检查进程是否存在。如果容器的存活检查失败，集群会对该容器执行重启操作；若容器的存活检查成功则不执行任何操作。
- **就绪探针：**readinessProbe，用于检查用户业务是否就绪，如果未就绪，则不转发流量到当前实例。一些程序的启动时间可能很长，比如要加载磁盘数据或者要依赖外部的某个模块启动完成才能提供服务。这时候程序进程在，但是并不能对外提供服务。这种场景下该检查方式就非常有用。如果容器的就绪检查失败，集群会屏蔽请求访问该容器；若检查成功，则会开放对该容器的访问。
- **启动探针**：startupProbe，用于探测应用程序容器什么时候启动了。 如果配置了这类探测器，就可以控制容器在启动成功后再进行存活性和就绪检查， 确保这些存活、就绪探针不会影响应用程序的启动。 这可以用于对启动慢的容器进行存活性检测，避免它们在启动运行之前就被终止。

检查方式

- **HTTP 请求检查**

  HTTP 请求方式针对的是提供HTTP/HTTPS服务的容器，集群周期性地对该容器发起HTTP/HTTPS GET请求，如果HTTP/HTTPS response返回码属于200~399范围，则证明探测成功，否则探测失败。使用HTTP请求探测必须指定容器监听的端口和HTTP/HTTPS的请求路径。

  例如：提供HTTP服务的容器，HTTP检查路径为：/health-check；端口为：80；主机地址可不填，默认为容器实例IP，此处以172.16.0.186为例。那么集群会周期性地对容器发起如下请求：GET http://172.16.0.186:80/health-check。

图 1-32 HTTP 请求检查

- **TCP 端口检查**

  对于提供TCP通信服务的容器，集群周期性地对该容器建立TCP连接，如果连接成功，则证明探测成功，否则探测失败。选择TCP端口探测方式，必须指定容器监听的端口。

  例如：有一个nginx容器，它的服务端口是80，对该容器配置了TCP端口探测，指定探测端口为80，那么集群会周期性地对该容器的80端口发起TCP连接，如果连接成功则证明检查成功，否则检查失败。

图 1-33 TCP 端口检查

- **执行命令检查**

  命令检查是一种强大的检查方式，该方式要求用户指定一个容器内的可执行命令，集群会周期性地在容器内执行该命令，如果命令的返回结果是0则检查成功，否则检查失败。

  对于上面提到的TCP端口检查和HTTP请求检查，都可以通过执行命令检查的方式来替代：

  - 对于TCP端口探测，可以使用程序对容器的端口尝试connect，如果connect成功，脚本返回0，否则返回-1。
  - 对于HTTP请求探测，可以使用脚本来对容器进行wget。

    wget http://127.0.0.1:80/health-check

    并检查response的返回码，如果返回码在200~399 的范围，脚本返回0，否则返回-1。如下图：

图 1-34 执行命令检查

须知

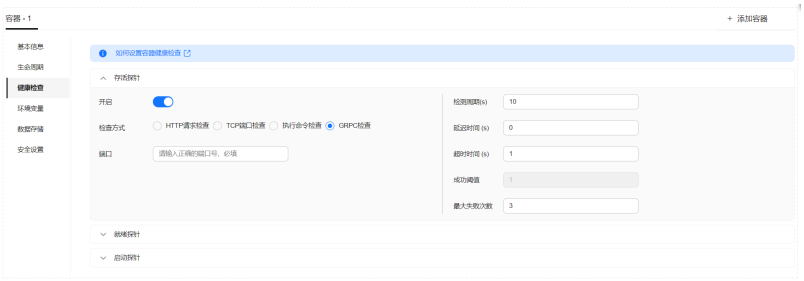
- 必须把要执行的程序放在容器的镜像里面，否则会因找不到程序而执行失败。
- 如果执行的命令是一个shell脚本，由于集群在执行容器里的程序时，不在终端环境下，因此不能直接指定脚本为执行命令，需要加上脚本解析器。比如脚本是**/data/scripts/health_check.sh**，那么使用执行命令检查时，指定的程序应该是：

```
sh
/data/scripts/health_check.sh
```

- **GRPC检查**

  GRPC检查可以为GRPC应用程序配置启动、活动和就绪探针，而无需暴露任何HTTP端点，也不需要可执行文件。Kubernetes可以通过GRPC连接到工作负载并查询其状态。

图 1-35 GRPC 检查



须知

- 使用GRPC检查时，您的应用需支持**GRPC健康检查协议**。
- 与HTTP和TCP探针类似，如果配置错误，都会被认作是探测失败，例如错误的端口、应用未实现健康检查协议等。

公共参数说明

表 1-42 公共参数说明

参数	参数说明
检测周期（periodSeconds）	探针检测周期，单位为秒。 例如，设置为30，表示每30秒检测一次。
延迟时间（initialDelaySeconds）	延迟检查时间，单位为秒，此设置与业务程序正常启动时间相关。 例如，设置为30，表明容器启动后30秒才开始健康检查，该时间是预留给业务程序启动的时间。
超时时间（timeoutSeconds）	超时时间，单位为秒。 例如，设置为10，表明执行健康检查的超时等待时间为10秒，如果超过这个时间，本次健康检查就被视为失败。若设置为0或不设置，默认超时等待时间为1秒。
成功阈值（successThreshold）	探测失败后，被视为成功的最小连续成功数。 默认值是 1，最小值是 1。 存活和启动探测的这个值必须是 1。
最大失败次数（failureThreshold）	当探测失败时重试的次数。 存活探测情况下的放弃就意味着重新启动容器。就绪探测情况下的放弃 Pod 会被打上未就绪的标签。 默认值是 3。最小值是 1。

YAML 示例

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      exec:
        command:
          - cat
          - /tmp/healthy
      initialDelaySeconds: 5
```

```
    periodSeconds: 5
  startupProbe:
    httpGet:
      path: /healthz
      port: 80
    failureThreshold: 30
    periodSeconds: 10
```

1.6.3.9 设置环境变量

操作场景

环境变量是指容器运行环境中设定的一个变量，环境变量可以在工作负载部署后修改，为工作负载提供极大的灵活性。

通过控制台设置的环境变量与Dockerfile中的“ENV”效果相同。

须知

容器启动后，容器中的内容不应修改。如果修改配置项（例如将容器应用的密码、证书、环境变量配置到容器中），当容器重启（例如节点异常重新调度Pod）后，会导致配置丢失，业务异常。

配置信息应通过入参等方式导入容器中，以免重启后配置丢失。

环境变量支持如下几种方式设置。

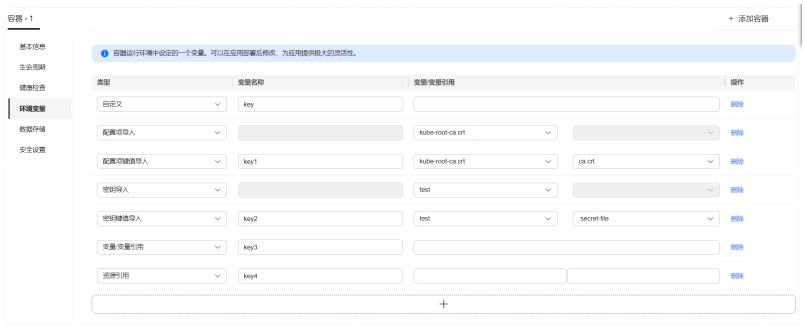
- **自定义**：自行填写变量名称及变量值。
- **配置项导入**：将配置项中所有键值都导入为环境变量。
- **配置项键值导入**：将配置项中某个键的值导入作为某个环境变量的值。例如将configmap-example这个配置项中configmap_key的值configmap_value导入为环境变量key1的值，导入后容器中有一个名为key1的环境变量，其值为configmap_value。
- **密钥导入**：将密钥中所有键值都导入为环境变量。
- **密钥键值导入**：将密钥中某个键的值导入作为某个环境变量的值。例如将secret-example这个配置项中secret_key的值secret_value导入为环境变量key2的值，导入后容器中有一个名为key2的环境变量，其值为secret_value。
- **变量引用**：用Pod定义的字段作为环境变量的值，例如Pod的名称。
- **资源引用**：用Container定义的字段作为环境变量的值，例如容器的CPU限制。

添加环境变量

步骤1 在创建工作负载时，配置容器信息，选择“环境变量”。

步骤2 设置环境变量。

图 1-36 添加环境变量



----结束

YAML 样例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: env-example
  template:
    metadata:
      labels:
        app: env-example
    spec:
      containers:
        - name: container-1
          image: nginx:alpine
          imagePullPolicy: Always
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              cpu: 250m
              memory: 512Mi
          env:
            - name: key                    # 自定义
              value: value
            - name: key1                   # 配置项键值导入
              valueFrom:
                configMapKeyRef:
                  name: configmap-example
                  key: key1
            - name: key2                   # 密钥键值导入
              valueFrom:
                secretKeyRef:
                  name: secret-example
                  key: key2
            - name: key3                   # 变量引用，用Pod定义的字段作为环境变量的值
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: key4                   # 资源引用，用Container定义的字段作为环境变量的值
              valueFrom:
                resourceFieldRef:
                  containerName: container1
                  resource: limits.cpu
```

```
            divisor: 1
      envFrom:
        - configMapRef:                # 配置项导入
            name: configmap-example
        - secretRef:                   # 密钥导入
            name: secret-example
    imagePullSecrets:
      - name: default-secret
```

环境变量查看

如果configmap-example和secret-example的内容如下。

```
$ kubectl get configmap configmap-example -oyaml
apiVersion: v1
data:
  configmap_key: configmap_value
kind: ConfigMap
...

$ kubectl get secret secret-example -oyaml
apiVersion: v1
data:
  secret_key: c2VjcmV0X3ZhbHVl              # c2VjcmV0X3ZhbHVl为secret_value的base64编码
kind: Secret
...
```

则进入Pod中查看的环境变量结果如下。

```
$ kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
env-example-695b759569-lx9jp   1/1     Running   0          17m

$ kubectl exec env-example-695b759569-lx9jp  -- printenv
/ # env
key=value                           # 自定义环境变量
key1=configmap_value                  # 配置项键值导入
key2=secret_value                   # 密钥键值导入
key3=env-example-695b759569-lx9jp     # Pod的metadata.name
key4=1                              # container1这个容器的limits.cpu，单位为Core，向上取整
configmap_key=configmap_value         # 配置项导入，原配置项中的键值直接会导入结果
secret_key=secret_value               # 密钥导入，原密钥中的键值直接会导入结果
```

1.6.3.10 工作负载升级配置

在实际应用中，升级是一个常见的场景，Deployment、StatefulSet和DaemonSet都能够很方便的支撑应用升级。

通过控制台配置工作负载升级

步骤1 在创建工作负载时，单击“展开高级配置”。

步骤2 参考**表1-43**，设置升级策略。

表 1-43 参数说明

参数	描述
升级方式	设置不同的升级策略，有如下两种。 ● RollingUpdate：滚动升级，即逐步创建新Pod再删除旧Pod，为默认策略。 ● Recreate：替换升级，即先把当前Pod删掉再重新创建Pod。
最大无效实例数（maxUnavailable）	与spec.replicas相比，可以有多少个Pod失效，也就是删除的比例，默认值是25%，比如spec.replicas为4，那升级过程中就至少有3个Pod存在，即删除Pod的步伐是1。同样这个值也可以设置成数字。 仅Deployment支持配置。
最大浪涌（maxSurge）	与spec.replicas相比，可以有多少个Pod存在，默认值是25%，比如spec.replicas为 4，那升级过程中就不能超过5个Pod存在，即按1个的步伐升级，实际升级过程中会换算成数字，且换算会向上取整。这个值也可以直接设置成数字。 仅Deployment支持配置。
实例可用最短时间（minReadySeconds）	指定新创建的 Pod 在没有任意容器崩溃情况下的最小就绪时间， 只有超出这个时间 Pod 才被视为可用。默认值为 0（Pod 在准备就绪后立即将被视为可用）。
最大保留版本数（revisionHistoryLimit）	用来设定出于回滚目的所要保留的旧 ReplicaSet 数量。这些旧 ReplicaSet 会消耗 etcd 中的资源，并占用 kubectl get rs 的输出。 每个 Deployment 修订版本的配置都存储在其 ReplicaSets 中；因此，一旦删除了旧的 ReplicaSet， 将失去回滚到 Deployment 的对应修订版本的能力。 默认情况下，系统保留 10 个旧 ReplicaSet，但其理想值取决于新 Deployment 的频率和稳定性。
升级最大时长（progressDeadlineSeconds）	指定系统在报告 Deployment 进展失败 之前等待 Deployment 取得进展的秒数。 这类报告会在资源状态中体现为 Type=Progressing、Status=False、Reason=ProgressDeadlineExceeded。Deployment 控制器将持续重试 Deployment。 将来，一旦实现了自动回滚，Deployment 控制器将在探测到这样的条件时立即回滚 Deployment。 如果指定，则此字段值需要大于 .spec.minReadySeconds 取值。
缩容时间窗（terminationGracePeriodSeconds）	优雅删除时间，默认为30秒，删除Pod时发送SIGTERM终止信号，然后等待容器中的应用程序终止执行，如果在terminationGracePeriodSeconds时间内未能终止，则发送SIGKILL的系统信号强行终止。

图 1-37 升级策略

----结束

通过控制台回退工作负载版本

回退也称为回滚，即当发现升级出现问题时，让应用回到原版本。Deployment可以非常方便的回退到原版本。

步骤1 在集群控制台选择“工作负载”，单击需要回退的工作负载名称。

步骤2 选择“版本记录”页签，找到希望回退至的版本，单击“回退到此版本”，单击“确定”，等待工作负载版本回退。

----结束

通过命令行配置升级示例

Deployment的升级可以是声明式的，也就是说只需要修改Deployment的YAML定义即可，比如使用kubectl edit命令将上面Deployment中的镜像修改为nginx:alpine。修改完成后再查询ReplicaSet和Pod，发现创建了一个新的ReplicaSet，Pod也重新创建了。

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME               DESIRED   CURRENT   READY     AGE
nginx-6f9f58dffd   2         2         2         1m
nginx-7f98958cdf   0         0         0         48m

$ kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-6f9f58dffd-tdmqk   1/1       Running   0          1m
nginx-6f9f58dffd-tesqr   1/1       Running   0          1m
```

Deployment可以通过maxSurge和maxUnavailable两个参数控制升级过程中同时重新创建Pod的比例，这在很多时候是非常有用，配置如下所示。

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

在前面的例子中，由于spec.replicas是2，如果maxSurge和maxUnavailable都为默认值25%，那实际升级过程中，maxSurge允许最多3个Pod存在（向上取整，2*1.25=2.5，取整为3），而maxUnavailable则不允许有Pod Unavailable（向上取整，2*0.75=1.5，取整为2），也就是说在升级过程中，一直会有2个Pod处于运行状态，每次新建一个Pod，等这个Pod创建成功后再删掉一个旧Pod，直至Pod全部为新Pod。

通过命令行回退工作负载版本

例如上面升级的新版镜像有问题，可以执行kubectl rollout undo命令进行回滚。

```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

Deployment之所以能如此容易的做到回滚，是因为Deployment是通过ReplicaSet控制Pod的，升级后之前ReplicaSet都一直存在，Deployment回滚做的就是使用之前的ReplicaSet再次把Pod创建出来。Deployment中保存ReplicaSet的数量可以使用revisionHistoryLimit参数限制，默认值为10。

1.6.3.11 调度策略（亲和与反亲和）

创建工作负载时可以使用nodeSelector选择Pod要部署的节点，其实Kubernetes还支持更精细、更灵活的调度机制，那就是亲和（affinity）与反亲和（anti-affinity）调度。

Kubernetes支持节点和Pod两个层级的亲和与反亲和。通过配置亲和与反亲和规则，可以允许您指定硬性限制或者偏好，例如将前台Pod和后台Pod部署在一起、某类应用部署到某些特定的节点、不同应用部署到不同的节点等等。

节点亲和（nodeAffinity）

通过nodeSelector可以让Pod只部署在具有特定标签的节点上。如下所示，Pod只会部署在拥有gpu=true这个标签的节点上。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:              # 节点选择，当节点拥有gpu=true标签时才在节点上创建Pod
    gpu: true
...
```

通过节点亲和性规则配置，也可以做到同样的事情，如下所示。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 3
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu: 100m
            memory: 200Mi
          limits:
            cpu: 100m
            memory: 200Mi
      imagePullSecrets:
      - name: default-secret
```



```
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: gpu
              operator: In
              values:
              - "true"
```

看起来这要复杂很多，但这种方式可以得到更强的表达能力，后面会进一步介绍。

这里affinity表示亲和，nodeAffinity表示节点亲和，requiredDuringSchedulingIgnoredDuringExecution非常长，不过可以将这个分作两段来看：

- 前半段requiredDuringScheduling表示下面定义的规则必须强制满足（require）才会调度Pod到节点上。
- 后半段IgnoredDuringExecution表示已经在节点上运行的Pod不需要满足下面定义的规则，即去除节点上的某个标签，那些需要节点包含该标签的Pod不会被重新调度。

另外操作符operator的值为In，表示标签值需要在values的列表中，其他operator取值如下。

- NotIn：标签的值不在某个列表中
- Exists：某个标签存在
- DoesNotExist：某个标签不存在
- Gt：标签的值大于某个值（字符串比较）
- Lt：标签的值小于某个值（字符串比较）

需要说明的是并没有nodeAntiAffinity（节点反亲和），因为NotIn和DoesNotExist可以提供相同的功能。

下面来验证这段规则是否生效，假设某集群有如下三个节点。

```
$ kubectl get node
NAME          STATUS   ROLES    AGE   VERSION
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

首先给192.168.0.212这个节点打上gpu=true的标签。

```
$ kubectl label node 192.168.0.212 gpu=true
node/192.168.0.212 labeled

$ kubectl get node -L gpu
NAME          STATUS   ROLES    AGE   VERSION                        GPU
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

创建这个Deployment，可以发现所有的Pod都部署在了192.168.0.212这个节点上。

```
$ kubectl create -f affinity.yaml
deployment.apps/gpu created

$ kubectl get pod -o wide
NAME                  READY   STATUS    RESTARTS   AGE   IP           NODE
gpu-6df65c44cf-42xw4     1/1     Running   0          15s   172.16.0.37   192.168.0.212
gpu-6df65c44cf-jzjvs     1/1     Running   0          15s   172.16.0.36   192.168.0.212
gpu-6df65c44cf-zv5cl     1/1     Running   0          15s   172.16.0.38   192.168.0.212
```

节点优先选择规则

上面讲的requiredDuringSchedulingIgnoredDuringExecution是一种**强制**选择的规则，节点亲和还有一种优先选择规则，即preferredDuringSchedulingIgnoredDuringExecution，表示会根据规则**优先**选择哪些节点。

为演示这个效果，先为上面的集群添加一个SAS磁盘的节点，并打上DISK=SAS的标签，为另外三个节点打上DISK=SSD的标签。

```
$ kubectl get node -L DISK,gpu
NAME           STATUS   ROLES    AGE     VERSION                          DISK     GPU
192.168.0.100   Ready    <none>   7h23m   v1.15.6-r1-20.3.0.2.B001-15.30.2   SAS
192.168.0.212   Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD      true
192.168.0.94    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD
192.168.0.97    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD
```

下面定义一个Deployment，要求Pod优先部署在SAS磁盘的节点上，可以像下面这样定义，使用preferredDuringSchedulingIgnoredDuringExecution规则，给SAS设置权重（weight）为80，而gpu=true权重为20，这样Pod就优先部署在SAS的节点上。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            preference:
              matchExpressions:
              - key: DISK
                operator: In
                values:
                - SSD
          - weight: 20
            preference:
              matchExpressions:
              - key: gpu
                operator: In
                values:
                - "true"
```

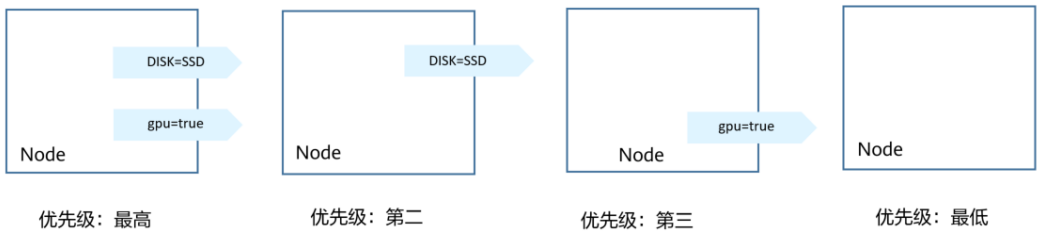
来看实际部署后的情况，可以看到部署到192.168.0.212这个节点上的Pod有5个，而192.168.0.100上只有2个。

```
$ kubectl create -f affinity2.yaml
deployment.apps/gpu created

$ kubectl get po -o wide
NAME                   READY   STATUS    RESTARTS   AGE     IP           NODE
gpu-585455d466-5bmcz   1/1     Running   0          2m29s   172.16.0.44   192.168.0.212
gpu-585455d466-cg2l6   1/1     Running   0          2m29s   172.16.0.63   192.168.0.97
gpu-585455d466-f2bt2   1/1     Running   0          2m29s   172.16.0.79   192.168.0.100
gpu-585455d466-hdb5n   1/1     Running   0          2m29s   172.16.0.42   192.168.0.212
gpu-585455d466-hkgvz   1/1     Running   0          2m29s   172.16.0.43   192.168.0.212
gpu-585455d466-mngvn   1/1     Running   0          2m29s   172.16.0.48   192.168.0.97
gpu-585455d466-s26qs   1/1     Running   0          2m29s   172.16.0.62   192.168.0.97
gpu-585455d466-sxtzm   1/1     Running   0          2m29s   172.16.0.45   192.168.0.212
gpu-585455d466-t56cm   1/1     Running   0          2m29s   172.16.0.64   192.168.0.100
gpu-585455d466-t5w5x   1/1     Running   0          2m29s   172.16.0.41   192.168.0.212
```

上面这个例子中，对于节点排序优先级如下所示，有个两个标签的节点排序最高，只有SSD标签的节点排序第二（权重为80），只有gpu=true的节点排序第三，没有的节点排序最低。

图 1-38 优先级排序顺序



这里您看到Pod并没有调度到192.168.0.94这个节点上，这是因为这个节点上部署了很多其他Pod，资源使用较多，所以并没有往这个节点上调度，这也侧面说明preferredDuringSchedulingIgnoredDuringExecution是优先规则，而不是强制规则。

工作负载亲和（podAffinity）

节点亲和的规则只能影响Pod和节点之间的亲和，Kubernetes还支持Pod和Pod之间的亲和，例如将应用的前端和后端部署在一起，从而减少访问延迟。Pod亲和同样有requiredDuringSchedulingIgnoredDuringExecution和preferredDuringSchedulingIgnoredDuringExecution两种规则。

来看下面这个例子，假设有个应用的后端已经创建，且带有app=backend的标签。

```
$ kubectl get po -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP           NODE
backend-658f6cb858-dlrz8   1/1     Running   0          2m36s   172.16.0.67   192.168.0.100
```

将前端frontend的pod部署在backend一起时，可以做如下Pod亲和规则配置。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
```

```
replicas: 3
template:
  metadata:
    labels:
      app:  frontend
  spec:
    containers:
    - image:  nginx:alpine
      name:  frontend
      resources:
        requests:
          cpu:  100m
          memory:  200Mi
        limits:
          cpu:  100m
          memory:  200Mi
    imagePullSecrets:
    - name: default-secret
    affinity:
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: kubernetes.io/hostname
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - backend
```

创建frontend然后查看，可以看到frontend都创建到跟backend一样的节点上了。

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67   192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70   192.168.0.100
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71   192.168.0.100
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72   192.168.0.100
```

这里有个**topologyKey**字段（拓扑域），意思是先圈定topologyKey指定的范围，然后再选择下面规则定义的内容。这里每个节点上都有kubernetes.io/hostname，所以看不出topologyKey起到的作用。

如果backend有两个Pod，分别在不同的节点上。

```
$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-5bpd6    1/1     Running   0          23m     172.16.0.40   192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          2m36s   172.16.0.67   192.168.0.100
```

给192.168.0.97和192.168.0.94打一个prefer=true的标签。

```
$ kubectl label node 192.168.0.97 prefer=true
node/192.168.0.97 labeled
$ kubectl label node 192.168.0.94 prefer=true
node/192.168.0.94 labeled

$ kubectl get node -L prefer
NAME           STATUS   ROLES    AGE   VERSION                          PREFER
192.168.0.100   Ready    <none>   44m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.212   Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.97    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
```

将podAffinity的topologyKey定义为prefer。

```
    affinity:
      podAffinity:
```

```
      requiredDuringSchedulingIgnoredDuringExecution:
      - topologyKey: prefer
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - backend
```

调度时，先圈定拥有prefer标签的节点，这里也就是192.168.0.97和192.168.0.94，然后再匹配app=backend标签的Pod，从而frontend就会全部部署在192.168.0.97上。

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-5bpd6    1/1     Running   0          26m     172.16.0.40   192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67   192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70   192.168.0.97
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71   192.168.0.97
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72   192.168.0.97
```

工作负载反亲和（podAntiAffinity）

前面讲了Pod的亲和，通过亲和将Pod部署在一起，有时候需求却恰恰相反，需要将Pod分开部署，例如Pod之间部署在一起会影响性能的情况。

下面例子中定义了反亲和规则，这个规则表示Pod不能调度到拥有app=frontend标签Pod的节点上，也就是下面将frontend分别调度到不同的节点上（每个节点只有一个Pod）。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 5
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
```

```
            operator: In
            values:
            - frontend
```

创建并查看，可以看到每个节点上只有一个frontend的Pod，还有一个在Pending，因为在部署第5个时4个节点上都有了app=frontend的Pod，所以第5个一直是Pending。

```
$ kubectl create -f affinity4.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP            NODE
frontend-6f686d8d87-8dlsc   1/1     Running   0          18s   172.16.0.76   192.168.0.100
frontend-6f686d8d87-d6l8p   0/1     Pending   0          18s   <none>        <none>
frontend-6f686d8d87-hgcq2   1/1     Running   0          18s   172.16.0.54   192.168.0.97
frontend-6f686d8d87-q7cfq   1/1     Running   0          18s   172.16.0.47   192.168.0.212
frontend-6f686d8d87-xl8hx   1/1     Running   0          18s   172.16.0.23   192.168.0.94
```

通过控制台配置调度策略

步骤1 在创建工作负载时，在“高级设置”中找到“调度策略”。

表 1-44 节点亲和性设置

参数名	参数描述
必须满足	即硬约束，设置必须要满足的条件，对应于requiredDuringSchedulingIgnoredDuringExecution，多条规则间是一种“或”的关系，即只需要满足一条规则即会进行调度。
尽量满足	即软约束，设置尽量满足的条件，对应于preferredDuringSchedulingIgnoredDuringExecution，无论是满足其中一条或者是都不满足都会进行调度。

步骤2 在“节点亲和性”、“工作负载亲和性”、“工作负载反亲和性”下单击 添加调度策略。在弹出的窗口中可以直接添加策略、指定节点或指定可用区。

指定节点和指定可用区本质也是通过标签实现，只是通过控制台提供了更为便捷的操作。指定节点使用的是 kubernetes.io/hostname 标签，可用区使用的是 failure-domain.beta.kubernetes.io/zone 标签。

表 1-45 调度策略设置

参数名	参数描述
标签名	对应节点的标签，可以使用默认的标签也可以用户自定义标签。

参数名	参数描述
操作符	可以设置六种匹配关系（In, NotIn, Exists, DoesNotExist. Gt, and Lt）。 ● In：是否在标签值的列表中 ● NotIn：是否不在标签值的列表中 ● Exists：某个标签存在 ● DoesNotExist：某个标签不存在 ● Gt：标签的值大于某个值（字符串比较） ● Lt：标签的值小于某个值（字符串比较）
标签值	请填写标签值。
命名空间	仅支持在工作负载亲和/工作负载反亲和调度策略中使用。 指定调度策略生效的命名空间。
拓扑域	仅支持在工作负载亲和/工作负载反亲和调度策略中使用。 先圈定拓扑域（topologyKey）指定的范围，然后再选择策略定义的内容。
权重	仅支持在“尽量满足”策略中添加。

----结束

1.6.3.12 容忍策略

容忍策略允许调度器将Pod调度至带有对应污点的节点上，需要与节点污点配合使用。每个节点可以添加一个或多个污点，对于未设置节点容忍策略的Pod，调度器会根据集群上的污点效果进行选择性调度，以避免Pod被分配到不合适的节点上。

通过控制台配置容忍策略

步骤1 登录UCS控制台。

步骤2 在创建工作负载时，单击“下一步：调度与差异化”。

步骤3 添加污点容忍策略。

参数名	参数描述
污点键	节点的污点键。
操作符	● Equal：设置此操作符表示准确匹配指定污点键（必填）和污点值的节点。如果不填写污点值，则表示可以与所有污点键相同的污点匹配。 ● Exists：设置此操作符表示匹配存在指定污点键的节点，此时容忍度不能指定污点值。若不填写污点键则可以容忍全部污点。

参数名	参数描述
污点值	• 如果操作符的值是 Exists，则value属性可省略。 • 如果操作符的值是 Equal，则表示其key与value之间的关系是Equal（等于）。 • 如果不指定操作符属性，则默认值为Equal。
污点策略	• 全部：表示匹配所有污点效果。 • NoSchedule：表示匹配污点效果为NoSchedule的污点。 • NoExecute：表示匹配污点效果为NoExecute的污点。
容忍时间窗	即tolerationSeconds参数，当污点策略为NoExecute时支持配置。 在容忍时间窗内，Pod还会在拥有污点的节点上运行，超出时间后会被驱逐。

----结束

1.6.4 服务与路由

1.6.4.1 服务

服务（Service）为集群中的工作负载提供了固定的访问方式，集群控制台支持创建以下服务：

- **集群内访问（ClusterIP）**
  表示工作负载暴露给同一集群内其他工作负载访问的方式，可以通过“集群内部域名”访问。集群内部域名格式为“<自定义的访问方式名称>.<工作负载所在命名空间>.svc.cluster.local”，例如“nginx.default.svc.cluster.local”。
- **节点访问（NodePort）**
  表示工作负载可以从集群外部访问。节点访问（NodePort）是指在每个节点的IP上开放一个静态端口，通过静态端口对外暴露服务。当集群中的节点绑定了EIP时，通过请求<EIP>:<NodePort>，也可实现从公网访问工作负载。
- **负载均衡（LoadBalancer）**
  通过弹性负载均衡从公网访问工作负载，一般用于系统中需要暴露到公网的服务。访问方式由公网弹性负载均衡ELB服务地址以及设置的访问端口组成，例如“10.117.117.117:80”。

集群内访问（ClusterIP）

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“服务”，选择“服务”页签，并选择服务所在命名空间。如需新建命名空间，请参见**创建命名空间**。

步骤3 单击右上角“创建服务”，设置集群内访问参数。

- **Service名称：**自定义服务名称，可与工作负载名称保持一致。
- **访问类型**：选择“集群内访问 ClusterIP”。

- **命名空间：**工作负载所在命名空间。
- **选择器：**添加标签，Service根据标签选择Pod，填写后单击“添加”。也可以引用已有工作负载的标签，单击“引用负载标签”，在弹出的窗口中选择负载，然后单击“确定”。
- **端口配置：**
  - 协议：请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问工作负载时使用，端口范围为1-65535，可任意指定。
  - 容器端口：工作负载程序实际监听的端口，需用户确定。例如nginx程序实际监听的端口为80。

步骤4 单击“确定”，创建Service。

----结束

节点访问（NodePort）

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“服务”，选择“服务”页签，并选择服务所在命名空间。如需新建命名空间，请参见**创建命名空间**。

步骤3 单击右上角“创建服务”，设置集群内访问参数。

- **Service名称：**自定义服务名称，可与工作负载名称保持一致。
- **访问类型**：选择“节点访问 NodePort”。
- **服务亲和：**
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- **命名空间：**工作负载所在命名空间。
- **选择器：**添加标签，Service根据标签选择Pod，填写后单击“添加”。也可以引用已有工作负载的标签，单击“引用负载标签”，在弹出的窗口中选择负载，然后单击“确定”。
- **端口配置：**
  - 协议：请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。

步骤4 单击“确定”，创建Service。

----结束

负载均衡（LoadBalancer）

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“服务”，选择“服务”页签，并选择服务所在命名空间。如需新建命名空间，请参见**创建命名空间**。

步骤3 单击右上角“创建服务”，设置集群内访问参数。

- **Service名称：**自定义服务名称，可与工作负载名称保持一致。
- **访问类型**：选择“负载均衡 LoadBalancer”。
- **服务亲和：**
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- **命名空间：**工作负载所在命名空间。
- **选择器：**添加标签，Service根据标签选择Pod，填写后单击“添加”。也可以引用已有工作负载的标签，单击“引用负载标签”，在弹出的窗口中选择负载，然后单击“确定”。
- **端口配置：**
  - 协议：请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
- **注解：**支持key/value对格式，请您根据自身业务以及厂家要求进行注解配置，填写完成后单击“添加”。

步骤4 单击“确定”，创建Service。

----结束

1.6.4.2 路由

Ingress使用弹性负载均衡作为流量入口对外提供访问，在四层负载均衡访问方式的基础上支持了URI配置，通过对应的URI将访问流量分发到对应的服务。用户可根据域名和路径对转发规则进行自定义，完成对访问流量的细粒度划分。该访问方式由公网弹性负载均衡ELB服务地址、设置的访问端口组成、定义的URI组成，例如：10.117.117.117:80/helloworld。

操作步骤

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“服务”，选择“路由”页签，并选择路由所在命名空间。如需新建命名空间，请参见**创建命名空间**。

步骤3 单击右上角“创建路由”，设置路由配置参数。

- **名称**：新增路由的名称，用户可自定义。
- **命名空间**：路由所在命名空间。
- **TLS配置**：
  - 服务器证书：选择IngressTLS类型的服务器证书。若无符合条件的证书，可单击“创建IngressTLS类型的密钥证书”，请参考**创建密钥**创建一个指定类型的密钥证书。
  - SNI（Server Name Indication）：输入域名并选择对应的证书。SNI是TLS的扩展协议，在该协议下允许同一个IP地址和端口号下对外提供多个基于TLS的访问域名，且不同的域名可以使用不同的安全证书。
- **转发策略配置**：请求的访问地址与转发规则匹配时（转发规则由域名、URL组成，例如：10.117.117.117:80/helloworld），此请求将被转发到对应的目标Service处理。您可添加多条转发策略。
  - 域名：可选项，输入实际访问的域名地址。请确保所填写的域名已注册并备案，一旦配置了域名规则后，必须使用域名访问。
  - URL：需要注册的访问路径，例如：/healthz。该访问路径需与后端应用暴露的URL一致，否则将返回404错误。
  - 目标服务名称：选择服务名称，需要先创建NodePort服务，具体可参考**节点访问（NodePort）**。
  - 目标服务访问端口：选择目标服务后，对应的容器端口将自动获取。
- **Ingress Class**：支持选择集群内已创建的Ingress Class，或手动输入规划的Ingress Class名称。
- **注解**：支持key/value对格式，请您根据自身业务以及厂家要求进行注解配置，填写完成后单击“添加”。

步骤4 单击“确定”，创建Ingress。

----结束

1.6.5 容器存储

集群挂载存储卷声明时，需要集群提供商具备存储类（StorageClass）功能，以实现存储卷的动态创建。您可前往集群控制台的“存储”页面，在“存储类”页签下查看集群支持的存储类。更多StorageClass相关内容，请参见**存储类**。

创建存储卷声明

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“存储”，在“存储卷声明”页签单击右上角“YAML创建”。

步骤3 填写或导入存储卷声明YAML。

步骤4 单击“创建”。

----结束

创建存储卷

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“存储”，在“存储卷”页签单击右上角“YAML创建”。

步骤3 填写或导入存储卷YAML。

步骤4 单击“确定”。

----结束

1.6.6 配置项与密钥

1.6.6.1 创建配置项

配置项（ConfigMap）是一种用于存储工作负载所需配置信息的资源类型，内容由用户决定。配置项创建完成后，可在容器工作负载中作为文件或者环境变量使用。

配置项允许您将配置文件从容器镜像中解耦，从而增强容器工作负载的可移植性。

配置项价值如下：

- 使用配置项功能可以帮您管理不同环境、不同业务的配置。
- 方便您部署相同工作负载的不同环境，配置文件支持多版本，方便您进行更新和回滚工作负载。
- 方便您快速将您的配置以文件的形式导入到容器中。

创建配置项

步骤1 登录集群控制台，单击左侧导航栏的“配置与密钥”，选择“配置项”页签。您还可以直接创建配置项或基于YAML来创建，若希望通过YAML创建，请跳转至**步骤4**。

步骤2 选择需要创建配置项的命名空间。

步骤3 直接创建配置项。单击“创建配置项”。

参照**表1-46**设置新增配置参数。

表 1-46 新建配置参数说明

参数	参数说明
名称	新建的配置项名称，同一个命名空间里命名必须唯一。
命名空间	新建配置项所在的命名空间，默认为当前查看的命名空间。
描述	配置项的描述信息。
配置项数据	工作负载配置的数据可以在容器中使用，或被用来存储配置数据。 单击＋，输入键、值。其中，“键”代表配置名；“值”代表配置内容。

参数	参数说明
配置标签	标签以Key/value键值对的形式附加到各种对象上（如工作负载、节点、服务等）。 标签定义了这些对象的可识别属性，用来对它们进行管理和选择。 1. 输入标签键、值。 2. 单击“确认添加”。

步骤4 基于YAML创建配置项。在创建配置项页面右侧单击“YAML创建”。

说明

若需要通过上传文件的方式创建资源，请确保资源描述文件已创建。支持json或yaml格式，详细请参见**ConfigMap资源文件配置说明**。

您可以导入或直接编写文件内容，格式为YAML或JSON。

- 方式一：导入编排文件。
  单击“导入”，导入格式为YAML或JSON的文件。编排内容中可直接展示YAML或JSON文件的内容。
- 方式二：直接编排内容。
  在编排内容区域框中，输入YAML或JSON文件内容。

步骤5 配置完成后，单击“确定”。

配置项列表中会出现新创建的配置项。

----结束

ConfigMap 资源文件配置说明

ConfigMap资源文件支持json和yaml两种文件格式，且文件大小不得超过2MB。

- json文件格式
  文件名称为configmap.json，配置示例如下：

```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "paas-broker-app-017",
    "namespace": "test"
  },
  "data": {
    "context": "{\"applicationComponent\":{\"properties\":{\"custom_spec\":{}},\"node_name\":\"paas-broker-app\",\"stack_id\":\"0177eae1-89d3-cb8a-1f94-c0feb7e91d7b\"},\"softwareComponents\":[{\"properties\":{\"custom_spec\":{}},\"node_name\":\"paas-broker\",\"stack_id\":\"0177eae1-89d3-cb8a-1f94-c0feb7e91d7b\"}]}"
  }
}
```

- yaml文件格式
  文件名称为configmap.yaml，配置示例如下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
```

```
  namespace: default
data:
  data-1: "value-1"
  data-2: "value-2"
```

相关操作

通过集群控制台，您还可以执行表1-47中的操作。

表 1-47 相关操作

操作	说明
查看详情	单击配置项名称即可查看配置项数据详情。
编辑YAML	单击配置项名称后的“编辑YAML”，可查看并编辑当前配置项的YAML文件。
更新	1. 单击配置项名称后的“更新”。 2. 根据表1-46更改信息。 3. 单击“确定”提交已修改的信息。
删除	单击配置项名称后的“删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的配置项。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

1.6.6.2 创建密钥

密钥（Secret）是一种用于存储工作负载所需要认证信息、密钥的敏感信息等的资源类型，内容由用户决定。资源创建完成后，可在容器工作负载中作为文件或者环境变量使用。

创建密钥

步骤1 登录集群控制台，单击左侧导航栏的“配置与密钥 > 密钥”。您还可以直接创建密钥或基于YAML来创建，若希望通过YAML创建，请跳转至步骤4。

步骤2 选择需要创建密钥的命名空间。

步骤3 单击“创建密钥”。

参照表1-48设置基本信息。

表 1-48 基本信息说明

参数	参数说明
名称	新建的密钥的名称，同一个命名空间内命名必须唯一。
命名空间	新建密钥所在的命名空间，默认为当前查看的命名空间。

参数	参数说明
描述	新建密钥的描述。
密钥类型	新建的密钥类型。 • Opaque：一般密钥类型。在高敏感场景下，建议先通过数据加密服务加密敏感数据后，再存入Secret中。 • kubernetes.io/dockerconfigjson：存放拉取私有仓库镜像所需的认证信息。如选择此类型的密钥，需要额外输入镜像仓库地址。 • IngressTLS：存放7层负载均衡服务所需的证书。如选择此类型的密钥，需要上传证书文件及私钥文件。 • 其他：若需要创建其他类型的密钥，请手动输入密钥类型。
密钥数据	工作负载密钥的数据可以在容器中使用。 • 当密钥为Opaque类型时：输入键、值。其中“值”必须使用Base64编码，勾选“自动Base64编码”即可自动将输入的值转换为Base64编码。手动进行Base64编码的方法请参见**如何进行Base64编码**。 • 当密钥为dockerconfigjson类型时：输入私有镜像仓库的用户名和密码。
密钥标签	标签以Key/value键值对的形式附加到各种对象上（如工作负载、节点、服务等）。 标签定义了这些对象的可识别属性，用来对它们进行管理和选择。 1. 输入键、值。 2. 单击“确认添加”

步骤4 基于YAML文件创建密钥。在创建配置项页面右侧单击“YAML创建”。

说明

若需要通过上传文件的方式创建资源，请确保资源描述文件已创建。支持json或yaml格式，详细请参见**Secret资源文件配置说明**。

您可以导入或直接编写文件内容，格式为YAML或JSON。

- 方式一：导入编排文件。

  单击“导入”，导入格式为YAML或JSON的文件。编排内容中可直接展示YAML或JSON文件的内容。

- 方式二：直接编排内容。

  在编排内容区域框中，输入YAML或JSON文件内容。

步骤5 配置完成后，单击“确定”。

密钥列表中会出现新创建的密钥。

----结束

Secret 资源文件配置说明

本章节主要介绍Secret类型的资源描述文件的配置示例。

例如现在有一个工作负载需要获取账号密码，可以通过Secret来实现：

- yaml文件格式

  定义的Secret文件secret.yaml内容如下。其中Value需要用Base64，Base64编码方法请参见**如何进行Base64编码**。

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret          # secret的名称
  namespace: default      #命名空间，默认为default
data:
  username: bXktdXNlcm5hbWUK  #用户名，需要用Base64编码
  password: ******  #需要用Base64编码
type: Opaque      # type建议不要做修改
```

- json文件格式

  定义的Secret文件secret.json内容如下。

```
{
  "apiVersion": "v1",
  "kind": "Secret",
  "metadata": {
    "name": "mysecret",
    "namespace": "default"
  },
  "data": {
    "username": "bXktdXNlcm5hbWUK",
    "password": "******"
  },
  "type": "Opaque"
}
```

相关操作

密钥创建完成后，您还可以执行**表1-49**中的操作。

说明

kube-system命名空间下的密钥资源不能通过控制台更新，也不能删除，只能查看。

表 1-49 其他操作

操作	说明
编辑YAML	单击密钥名称后的“编辑YAML”，可编辑当前密钥的YAML文件。
更新密钥	1. 选择需要更新的密钥名称，单击“更新”。 2. 根据**表1-48**更改信息。 3. 单击“确定”。
删除密钥	选择要删除的密钥，单击“删除”。 根据系统提示删除密钥。

操作	说明
批量删除密钥	1. 勾选需要删除的密钥名称。 2. 单击页面左上角的“批量删除”，删除勾选的密钥。 3. 根据系统提示删除密钥。

如何进行 Base64 编码

对字符串进行Base64加密，可以直接使用“echo -n 要编码的内容 | base64”命令即可，示例如下：

```
root@ubuntu:~# echo -n "待编码内容" | base64
******
```

1.6.7 KubeConfig

1.6.7.1 获取 KubeConfig 文件

KubeConfig文件用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），通常与kubectl命令行工具或其他客户端结合使用。详细介绍可参见**Kubernetes文档**。

本章节将指导您获取集群的KubeConfig文件，由于各集群提供商的KubeConfig文件格式略有差异，请根据您的集群进行相应的操作。

须知

KubeConfig文件中包含集群认证信息，存在安全泄露风险，请您务必妥善保存。

华为云集群

步骤1 登录CCE控制台，单击目标集群名进入集群信息页面。

步骤2 在“连接信息”中单击kubectl后的“配置”按钮。

步骤3 参照页面中的提示信息，下载kubectl配置文件（公网地址变更后需重新下载）。

步骤4 使用**步骤3**中下载的配置文件接入集群，详细步骤请继续参考**注册附着集群（公网接入）**或**注册附着集群（私网接入）**。

----结束

第三方云厂商集群

由于第三方云厂商集群提供的KubeConfig文件格式存在差异，您需要自行创建一个具有所有集群资源操作权限的ServiceAccount，并获取这个ServiceAccount的token，用于配置UCS支持的KubeConfig文件。

步骤1 通过kubectl连接集群。

步骤2 创建ucs-service-account.yaml文件。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ucs-user
---
apiVersion: v1
kind: Secret
metadata:
  name: ucs-user-token
  annotations:
    kubernetes.io/service-account.name: "ucs-user"
type: kubernetes.io/service-account-token
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ucs-user-role
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - get
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: ucs-user-role-binding
subjects:
  - kind: ServiceAccount
    name: ucs-user
    namespace: default
roleRef:
  kind: ClusterRole
  name: ucs-user-role
  apiGroup: rbac.authorization.k8s.io
```

步骤3 在集群中执行以下命令创建ServiceAccount。

kubectl apply -f ucs-service-account.yaml

步骤4 使用以下命令获取token。

kubectl get secret ucs-user-token -n *default* **-oyaml | grep token: | awk '{print $2}' | base64 -d ;echo**

步骤5 配置KubeConfig文件。

参考以下示例创建一个kubeconfig.yaml文件，并将token替换为**步骤4**中获取的值。

kubeconfig.yaml：

```
kind: Config
apiVersion: v1
preferences: {}
clusters:
  - name: internalCluster
    cluster:
      server: 'https://kubernetes.default.svc.cluster.local:443'
      insecure-skip-tls-verify: true
users:
  - name: ucs-user
```

```
    user:
      token: 'MIIFbAYJKo*****'
contexts:
  - name: internal
    context:
      cluster: internalCluster
      user: ucs-user
current-context: internal
```

KubeConfig文件中的关键参数说明如下：

参数名	参数值	说明	是否可选
server	'https://kubernetes.default.svc.cluster.local:443'	APIServer的集群内访问地址。由于部分厂商集群对APIServer地址做了外部访问限制，可能导致UCS无法正常接入集群，因此建议使用集群内访问地址。	必选
insecure-skip-tls-verify	true	如使用该参数，表示跳过证书认证，参数值必须为true。	二选一 **说明** 当server字段为集群内访问地址时，优选跳过证书认证。
certificate-authority-data	base64加密字符串	如使用该参数，表示集群开启双向认证，参数值为经base64加密后的服务端证书。 原生K8s集群的服务端证书默认地址为master节点的“/etc/kubernetes/pki/ca.crt”。	
token	base64加密字符串	用户以token方式进行认证，参数值为**步骤4**中获取的token值。	三选一 **说明** 优选token方式，UCS不支持除这三种方式外的其他认证方式。
• client-certificate-data • client-key-data	base64加密字符串	用户以证书加私钥的方式进行认证。 • client-certificate-data：经base64加密后的客户端证书。 • client-key-data：经base64加密后的客户端私钥。	
• username • password	字符串	用户通过用户名密码进行认证。 • username：访问集群的用户名。 • password：用户名对应的密码。	

步骤6 使用**步骤5**中配置的KubeConfig文件接入集群，详细步骤请继续参考**注册附着集群（公网接入）**或**注册附着集群（私网接入）**。

说明

使用UCS期间，创建的ServiceAccount、ClusterRole、ClusterRoleBinding对象均不能删除，否则token将会失效。

如集群不再接入UCS，可使用**kubectl delete -f ucs-service-account.yaml**命令删除UCS创建的SA对象。

----结束

自建集群

如果您的集群是通过Kubernetes官方二进制文件或Kubeadm等部署工具搭建的标准集群，可直接使用以下方法获取KubeConfig文件。

该方法不适用于云服务商提供的商用集群，商用集群的KubeConfig文件获取请参考**第三方云厂商集群**。

步骤1 登录集群Master节点。

步骤2 查看集群访问凭证。默认情况下，自建集群的配置文件路径为Master节点的“$HOME/.kube/config”，如您的集群指定了其他KubeConfig配置文件，请自行更换路径。

cat $HOME/.kube/config

步骤3 复制该凭证内容。

步骤4 在本地创建一个YAML文件，将上一步中复制的凭证内容粘贴至该文件并保存。

步骤5 使用**步骤4**中的YAML文件接入集群，详细步骤请继续参考**注册附着集群（公网接入）**或**注册附着集群（私网接入）**。

----结束

1.6.7.2 更新 KubeConfig 文件

本章节将指导您更新集群的KubeConfig文件，以便应对集群证书信息泄露或过期情况，或进行例行的安全维护。

更新KubeConfig文件的操作仅适用于附着集群与伙伴云集群。

前提条件

- 集群未加入任何舰队。
- 集群安装了anp-agent插件，以保证新的KubeConfig文件能与集群完成一次连通性探测。

操作步骤

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在左侧导航栏选择“容器舰队”，单击“未加入舰队的集群”，下拉找到需要更新KubeConfig文件的集群。

步骤3 单击集群页签右上角的。

图 1-39 更新 kubeconfig 文件

步骤4 选择上传本地KubeConfig文件，并选择与原地址一致的Context地址。

图 1-40 上传 KubeConfig 文件

步骤5 单击“确定”提交更新。

----结束

1.6.8 自定义资源

自定义资源（Custom Resource Definitions, CRD）允许用户创建一个与Deployment、Service类似的定制资源对象，用户可以通过kubectl命令来创建和访问这种自定义资源，为用户提供模块化的Kubernetes扩展，详情请参考**使用CustomResourceDefinition 扩展 Kubernetes API**。

操作步骤

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“自定义资源”，在右上角单击“YAML创建”。

步骤3 在线编辑或选择导入自定义资源的YAML文件，单击“确定”。

步骤4 其他操作：

- 单击操作列“查看YAML”，可查看自定义资源的YAML内容。
- 单击操作列“查看资源”，可查看集群中已有的自定义资源实例。

----结束

1.6.9 命名空间

集群控制台可以创建只作用于当前集群的命名空间，用于当前集群创建工作负载、创建任务等，支持对命名空间进行配额管理或删除命名空间，但所有操作只作用于当前集群。

- 其中默认创建default命名空间只支持管理配额不支持删除。
- 集群自带的命名空间，如：kube-public和kube-system，既不支持管理配额也不支持删除。

创建命名空间

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“命名空间”，单击右上角“创建命名空间”，配置参数。

- 命名空间名称：新建命名空间名称，命名必须唯一。
- 描述：对命名空间的描述。
- 配额管理：开启后可设置资源配额。资源配额可以限制命名空间下的资源使用，进而支持以命名空间为粒度的资源划分。

  若暂不启用，可以在命名空间创建后，在列表右侧单击配额管理进行配置，请参见**设置命名空间配额**。

步骤3 单击“确定”。

----结束

删除命名空间

须知

删除命名空间将会删除该命名空间相关的所有数据资源，请谨慎操作。

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“命名空间”，选择需要删除的命名空间，单击“更多 > 删除”。

----结束

设置命名空间配额

资源配额可以限制命名空间下的资源使用，进而支持以命名空间为粒度的资源划分。

通过设置命名空间级别的资源配额，实现多团队或多用户在共享集群资源的情况下限制团队、用户可以使用的资源总量，包括限制命名空间下创建某一类型对象的数量以及对象消耗计算资源（CPU、内存）的总量。

须知

系统创建的kube-public、kube-system等命名空间不支持设置资源配额。

步骤1 登录集群控制台。

步骤2 在左侧导航栏中选择“命名空间”，单击对应命名空间后的“管理配额”。

步骤3 设置资源配额。

须知

- 所有配额均默认为不限制，如需设置资源配额，请输入大于等于1的整型数值。若手动限制CPU或内存的配额，则创建工作负载时必须指定CPU或内存请求值。
- 配额累计使用量包含系统默认创建的资源，如default命名空间下系统默认创建的Kubernetes服务（该服务可通过后端kubectl工具查看）等，故建议命名空间下的资源配额略大于实际期望值以去除系统默认创建资源的影响。

- CPU（Core）：限制命名空间下工作负载实例（Pod）能申请CPU资源的最大值，单位为“核”。
- 内存（MiB）：限制命名空间下工作负载实例能申请内存资源的最大值，单位为MiB。
- 有状态工作负载（StatefulSet）：限制命名空间下能创建有状态负载的最大数量。
- 无状态工作负载（Deployment）：限制命名空间下能创建无状态负载的最大数量。
- 普通任务（Job）：限制命名空间下能创建普通任务的最大数量。
- 定时任务（CronJob）：限制命名空间下能创建定时任务的最大数量。
- 容器组（Pod）：限制命名空间下能创建Pod的最大数量，包含停止状态的Pod。
- 容器组（不包含停止状态的Pod）：限制命名空间下能创建Pod的最大数量，不包含停止状态的Pod。
- 服务（Service）：限制命名空间下能创建服务的最大数量，包含停止状态的Service。
- 服务（不包含停止状态的Service）：限制命名空间下能创建服务的最大数量，不包含停止状态的Service。
- 存储卷声明（PersistentVolumeClaim）：限制命名空间下能创建存储卷声明的最大数量。
- 配置项（ConfigMap）：限制命名空间下能创建配置项的最大数量。
- 密钥（Secret）：限制命名空间下能创建密钥的最大数量。

步骤4 单击“确定”。

----结束

1.6.10 工作负载弹性伸缩（HPA）

HPA策略即Horizontal Pod Autoscaling，是Kubernetes中实现Pod水平自动伸缩的功能。该策略在Kubernetes社区HPA功能的基础上，增加了应用级别的冷却时间窗和扩缩容阈值等功能。

前提条件

使用HPA前需要在集群内安装能够提供Metrics API的插件（详情请参见**对Metrics API的支持**）：

- metrics-server：metrics-server从kubelet公开的Summary API中采集度量数据，提供基础资源使用指标，例如容器CPU和内存使用率。
  - 为本地集群安装metrics-server，请参见**metrics-server**。

  - 为其他集群安装metrics-server，请参见**社区官方文档**。对于附着集群，您也可安装对应厂商所提供的metric-server插件。
- Prometheus：Prometheus是一套开源的系统监控报警框架，能够采集丰富的Metrics（度量数据），因此除基础资源指标外，Prometheus还支持提供自定义指标。

约束与限制

- 需要创建弹性扩缩容策略的集群至少有一个实例， 如果没有实例则会自动进行扩容。
- 如果集群内未安装系统指标采集插件，负载伸缩策略会无法生效。
- 目前本地集群仅支持metrics-server插件来提供Metrics API，未来会开放更多插件供选择。

操作步骤

步骤1 登录UCS集群控制台。

- 如果是未加入舰队集群，直接单击集群名即可进入集群控制台。
- 如果是已加入容器舰队的集群，先进入对应容器舰队控制台，选择左侧 “集群管理”内的“容器集群”，再进入对应集群控制台。

步骤2 左侧导航栏内选择“负载伸缩策略”，单击右上角的“创建 HPA 策略”。

步骤3 填写待创建HPA策略的参数：

表 1-50 HPA 策略参数配置

参数	参数说明
策略名称	新建策略的名称，请自定义。
命名空间	请选择工作负载所在的命名空间。
关联工作负载	请选择要设置HPA 策略的工作负载。
实例范围	请输入最小实例数和最大实例数。 策略触发时，工作负载实例将在此范围内伸缩。

系统策略	● 指标：可选择“CPU利用率”或“内存利用率”。 **说明** 利用率 = 工作负载容器组（Pod）的实际使用量 / 申请量 ● 期望值：请输入期望资源平均利用率。 期望值表示所选指标的期望值，通过向上取整（当前指标值 / 期望值 × 当前实例数）来计算需要伸缩的实例数。 **说明** HPA在计算扩容、缩容实例数时，会选择最近5分钟内实例数的最大值。 ● 容忍范围：容忍度默认为0.1。指标处于范围内时不会触发伸缩，期望值必须在容忍范围之间。 当指标值大于缩容阈值且小于扩容阈值时，不会触发扩容或缩容。**阈值仅在1.15及以上版本的集群中支持**。 **须知** 可以设置多条系统策略。

----结束

1.6.11 插件管理

1.6.11.1 kube-prometheus-stack 插件

插件简介

kube-prometheus-stack通过使用Prometheus Operator和Prometheus，提供简单易用的端到端Kubernetes集群监控能力，同时还具备自定义插件规格、对接Grafana、高可用、节点亲和等能力。

kube-prometheus-stack插件的核心组件包括prometheusOperator、prometheus、alertmanager、thanosSidecar、thanosQuery、adapter、kubeStateMetrics、nodeExporter、grafana。

- prometheusOperator：根据自定义资源（Custom Resource Definition / CRDs）来部署和管理Prometheus Server，同时监控这些自定义资源事件的变化来做相应的处理，是整个系统的控制中心。
- prometheus（Server）：Operator根据自定义资源Prometheus类型中定义的内容而部署Prometheus Server集群，这些自定义资源可以看作是用来管理Prometheus Server集群的StatefulSets资源。
- alertmanager：插件的告警中心，主要用于接收Prometheus发送的告警并通过去重、分组、分发等能力管理告警信息。
- thanosSidecar：高可用场景和Prometheus运行在同一个Pod中，用于实现普罗指标数据的持久化存储。
- thanosQuery：普罗高可用时PromQL查询的入口，能够对来自Store或Prometheus的相同指标进行重复数据删除。
- adapter（custom-metrics-apiserver）：将自定义指标聚合到原生的Kubernetes API Server。
- kube-state-metrics：将Prometheus的metrics数据格式转换成Kubernetes API接口能识别的格式。kube-state-metrics组件在默认配置下，不采集Kubernetes资源

的所有labels和annotation。如需采集，请参考**如何修改kube-state-metrics组件的采集配置？**章节进行配置。

- nodeExporter：每个节点上均有部署，收集Node级别的监控数据。
- grafana：可视化浏览普罗监控数据。Grafana会默认创建大小为5 GiB的存储卷，卸载插件时Grafana的存储卷不随插件被删除。
- clusterProblemDetector：用于监控集群异常。

插件部署模式

kube-prometheus-stack插件在部署时支持Agent模式和Server模式。

- Agent模式下，插件占用集群资源较低，为集群提供普罗指标采集能力，但不支持基于自定义普罗语句的HPA及健康诊断功能。
- Server模式支持基于自定义普罗语句的HPA及健康诊断功能，依赖PVC，内存消耗较大。

注意事项

kube-prometheus-stack为系统监控插件，当集群资源不足时，Kubernetes会优先保证插件Pod的调度。

权限说明

kube-prometheus-stack插件中的node-exporter组件会监控Docker的存储磁盘空间，需要读取宿主机的/var/run/dockersock的获取Docker的info的数据。

node-exporter运行需要以下特权：

- cap_dac_override：读取Docker的info的数据。

升级插件

步骤1 选择一个容器舰队或者未加入舰队的集群。

图 1-41 选择舰队或未加入舰队的集群

步骤2 单击“容器洞察 > 集群总览”页签查看已开启监控的集群，选择待升级插件的集群，单击操作列的“查看详情”，进入概览页。

步骤3 页面右上角会展示kube-prometheus-stack插件的版本，当安装的插件版本非最新版本时，可选择升级插件，体验插件的最新功能。

----结束

不同规格的资源配额要求

安装kube-prometheus-stack插件时，需确保集群中有足够的CPU、内存等可调度资源。Agent模式默认规格的资源配额要求请参见**表1-51**，Server模式下不同插件规格的资源配额要求请参见**表1-52**。

表 1-51 Agent 模式默认规格的资源配额要求

插件规格	容器实例	CPU配额		内存配额	
默认规格	prometheusOperator	申请：100m	限制：500m	申请：100Mi	限制：500Mi
	prometheus	申请：500m	限制：4	申请：1Gi	限制：8Gi
	kube-state-metrics	申请：200m	限制：500m	申请：200Mi	限制：500Mi
	nodeExporter	申请：200m	限制：500m	申请：200Mi	限制：1Gi
	grafana	申请：100m	限制：500m	申请：200Mi	限制：2Gi

表 1-52 Server 模式不同规格的资源配额要求

插件规格	容器实例	CPU配额		内存配额	
演示规格（100容器以内）	prometheusOperator	申请：200m	限制：500m	申请：200Mi	限制：500Mi
	prometheus	申请：500m	限制：2	申请：2Gi	限制：8Gi
	alertmanager	申请：200m	限制：1	申请：200Mi	限制：1Gi
	thanosSidecar	申请：100m	限制：1	申请：100Mi	限制：2Gi
	thanosQuery	申请：500m	限制：2	申请：500Mi	限制：4Gi
	adapter	申请：400m	限制：2	申请：400Mi	限制：1Gi
	kube-state-metrics	申请：200m	限制：500m	申请：200Mi	限制：500Mi
	nodeExporter	申请：200m	限制：500m	申请：200Mi	限制：1Gi

插件规格	容器实例	CPU配额		内存配额	
	grafana	申请：200m	限制：500m	申请：200Mi	限制：2Gi
	clusterProblemDetector	申请：100m	限制：200m	申请：200Mi	限制：400Mi
小规格（2000容器以内）	prometheusOperator	申请：200m	限制：500m	申请：200Mi	限制：500Mi
	prometheus	申请：4	限制：8	申请：16Gi	限制：32Gi
	alertmanager	申请：500m	限制：1	申请：500Mi	限制：1Gi
	thanosSidecar	申请：500m	限制：1	申请：500Mi	限制：2Gi
	thanosQuery	申请：2	限制：4	申请：2Gi	限制：16Gi
	adapter	申请：2	限制：4	申请：4Gi	限制：16Gi
	kube-state-metrics	申请：500m	限制：1	申请：500Mi	限制：1Gi
	nodeExporter	申请：200m	限制：500m	申请：200Mi	限制：1Gi
	grafana	申请：200m	限制：500m	申请：200Mi	限制：2Gi
	clusterProblemDetector	申请：200m	限制：500m	申请：300Mi	限制：1Gi
中规格（5000容器以内）	prometheusOperator	申请：500m	限制：1	申请：500Mi	限制：1Gi
	prometheus	申请：8	限制：16	申请：32Gi	限制：64Gi
	alertmanager	申请：500m	限制：1	申请：500Mi	限制：2Gi
	thanosSidecar	申请：1	限制：2	申请：1Gi	限制：4Gi
	thanosQuery	申请：2	限制：4	申请：2Gi	限制：16Gi
	adapter	申请：2	限制：4	申请：16Gi	限制：32Gi
	kube-state-metrics	申请：1	限制：2	申请：1Gi	限制：2Gi
	nodeExporter	申请：200m	限制：500m	申请：200Mi	限制：1Gi
	grafana	申请：200m	限制：500m	申请：200Mi	限制：2Gi

插件规格	容器实例	CPU配额		内存配额	
	clusterProblemDetector	申请：200m	限制：1	申请：400Mi	限制：2Gi
大规格（超过5000容器）	prometheusOperator	申请：500m	限制：1	申请：500Mi	限制：2Gi
	prometheus	申请：8	限制：32	申请：64Gi	限制：128Gi
	alertmanager	申请：1	限制：2	申请：1Gi	限制：4Gi
	thanosSidecar	申请：2	限制：4	申请：2Gi	限制：8Gi
	thanosQuery	申请：2	限制：4	申请：2Gi	限制：32Gi
	adapter	申请：2	限制：4	申请：32Gi	限制：64Gi
	kube-state-metrics	申请：1	限制：3	申请：1Gi	限制：3Gi
	nodeExporter	申请：200m	限制：500m	申请：200Mi	限制：1Gi
	grafana	申请：200m	限制：500m	申请：200Mi	限制：2Gi
	clusterProblemDetector	申请：200m	限制：1	申请：400Mi	限制：2Gi

1.6.11.2 log-agent 插件

log-agent是基于开源fluent-bit和opentelemetry构建的云原生日志采集插件，支持基于CRD的日志采集策略，可以根据您配置的策略规则，对集群中的容器标准输出日志、容器文件日志、节点日志及K8s事件日志进行采集与转发。

log-agent插件的核心组件包括fluent-bit、cop-logs、log-operator和otel-collector。

- fluent-bit：日志收集器，以DaemonSet形式安装在每个节点。
- cop-logs：负责采集侧配置文件生成及更新的组件。
- log-operator：负责解析及更新日志规则的组件。
- otel-collector：负责集中式日志转发的组件，将fluent-bit收集的日志转发到云日志服务LTS。

不同规格的资源配额要求

安装log-agent插件时，需确保集群中有足够的CPU、内存等可调度资源，不同插件规格的资源配额要求请参见**表1-53**。

表 1-53 不同规格的资源配额要求

插件规格	容器实例	CPU配额		内存配额	
小规格（实例数为1）	fluent-bit	申请：100m	限制：500m	申请：200Mi	限制：500Mi
	cop-logs	申请：100m	限制：1	申请：100Mi	限制：500Mi
	log-operator	申请：100m	限制：500m	申请：100Mi	限制：500Mi
	otel-collector	申请：200m	限制：1	申请：1Gi	限制：2Gi
大规格（实例数为2）	fluent-bit	申请：100m	限制：500m	申请：200Mi	限制：500Mi
	cop-logs	申请：100m	限制：1	申请：100Mi	限制：500Mi
	log-operator	申请：100m	限制：500m	申请：100Mi	限制：500Mi
	otel-collector	申请：200m	限制：1	申请：1Gi	限制：2Gi

安装 log-agent 插件

本地集群安装log-agent插件的具体操作请参见**云原生日志采集插件**。

1.6.11.3 metrics-server

从 Kubernetes 1.8开始，Kubernetes 通过 Metrics API 提供资源使用指标，例如容器CPU和内存使用率。这些度量可以由用户直接访问（例如：通过使用kubectl top命令），或者由集群中的控制器（例如：Horizontal Pod Autoscaler）使用来进行决策，具体的组件为Metrics Server。

Metrics Server是集群核心资源监控数据的聚合器，在UCS中Metrics Server的对应插件为“metrics-server”，您可以在UCS的集群控制台中快速安装本插件。

安装本插件后，可在集群控制台内的“负载伸缩”页签下，创建HPA策略，具体请参见**创建工作负载弹性伸缩（HPA）**。

社区官方项目：**https://github.com/kubernetes-sigs/metrics-server**。

约束和限制

目前metrics-server插件仅支持UCS本地集群，暂不支持其他UCS集群。

安装插件

步骤1 进入UCS内的集群控制台。

- 如果是未加入舰队集群，直接单击集群名即可进入集群控制台。
- 如果是已加入容器舰队的集群，先进入对应容器舰队控制台，选择左侧 “集群管理”内的“容器集群”，再进入对应集群控制台。

步骤2 左侧导航栏内选择“插件管理”，在“可安装插件”中单击metrics-server的安装按钮。

步骤3 在“安装插件”页面进行规格配置，该插件可配置“单实例”、“高可用”和“自定义”三种规格，选择后单击“安装”。

说明

- 在本地集群中，metrics-server插件的最大实例数依赖manage节点数量，如果想要使用“自定义”规格创建更多的metrics-server实例，请先对manage节点进行扩容。
- manage节点在本地集群内使用tag和taint进行管理，因此对manage节点进行扩容只需要为集群内非manage节点打上标签和污点即可，具体步骤如下：
  1. 进入UCS集群控制台，单击左侧导航栏内的“节点管理”。
  2. 选中待转的非manage节点并单击“标签与污点管理”。
  3. 单击新增批量操作，新增一条更新内容：“添加/更新”-“K8S标签”-“cop.manage”-“manage”。
  4. 单击新增批量操作，新增一条更新内容：“添加/更新”-“污点（Taints）”-“role”-“manage”-“NoSchedule”。
  5. 完成上述两条后，单击确定即可完成manage节点的扩容。

步骤4 插件安装完成后，在已安装插件内单击metrics-server插件，可以看到具体的插件实例在集群内的部署情况。

----结束

升级插件

步骤1 登录UCS内的集群控制台，在左侧导航栏里选择“插件管理”。

步骤2 在“已安装插件”内，如果版本标签旁边显示“存在新版请升级”提示，那就可用单击metrics-server下的升级按钮对插件进行升级。

说明

- 如果升级按钮处于冻结状态，则说明当前插件版本是最新的版本，不需要进行升级操作。
- 升级“metrics-server”插件时，会替换原先节点上的旧版本的“metrics-server”插件，安装最新版本的“metrics-server”插件以实现功能的快速升级。

步骤3 参考安装插件内的参数说明配置参数后，单击“升级”即可升级metrics-server插件。

----结束

编辑插件

步骤1 登录UCS内的集群控制台，在左侧导航栏里选择“插件管理”。

步骤2 在“已安装插件”内选择metrics-server，单击插件页签下方的“编辑”按钮。

步骤3 参考安装插件内的参数说明配置参数后，单击“确定”即可修改metrics-server插件的相关配置。

----结束

卸载插件

步骤1 登录UCS内的集群控制台，单击左侧导航栏内的“插件管理”。

步骤2 在“已安装插件”内找到metrics-server，单击“卸载”按钮。

步骤3 在弹出的“卸载插件”按钮中选择“是”，即可将metrics-server从集群内删除。

说明

卸载metrics-server插件后，需要安装其他提供Metrics API的插件，否则已创建的负载伸缩策略将会变得不可用，请谨慎卸载.

----结束

1.6.11.4 volcano

插件简介

Volcano是一个基于Kubernetes的批处理平台，提供了机器学习、深度学习、生物信息学、基因组学及其他大数据应用所需要而Kubernetes当前缺失的一系列特性。

Volcano提供了高性能任务调度引擎、高性能异构芯片管理、高性能任务运行管理等通用计算能力，通过接入AI、大数据、基因、渲染等诸多行业计算框架服务终端用户。(目前Volcano项目已经在Github开源)

Volcano针对计算型应用提供了作业调度、作业管理、队列管理等多项功能，主要特性包括：

- 丰富的计算框架支持：通过CRD提供了批量计算任务的通用API，通过提供丰富的插件及作业生命周期高级管理，支持TensorFlow，MPI，Spark等计算框架容器化运行在Kubernetes上。
- 高级调度：面向批量计算、高性能计算场景提供丰富的高级调度能力，包括成组调度，优先级抢占、装箱、资源预留、任务拓扑关系等。
- 队列管理：支持分队列调度，提供队列优先级、多级队列等复杂任务调度能力。

项目开源地址：**https://github.com/volcano-sh/volcano**

安装插件

步骤1 登录UCS控制台，单击集群名称进入集群，单击左侧导航栏的“插件管理”，找到**Volcano**，单击“安装”。

步骤2 该插件可配置“单实例”、“高可用”或自定义规格。

选择自定义时，volcano-controller和volcano-scheduler的建议值如下：

- 小于100个节点，可使用默认配置，即CPU的申请值为500m，限制值为2000m；内存的申请值为500Mi，限制值为2000Mi。
- 高于100个节点，每增加100个节点（10000个Pod），建议CPU的申请值增加500m，内存的申请值增加1000Mi；CPU的限制值建议比申请值多1500m，内存的限制值建议比申请值多1000Mi。

说明

申请值推荐计算公式：

- CPU申请值：计算“目标节点数 * 目标Pod规模”的值，并在表1-54中根据“集群节点数 * Pod规模”的计算值进行插值查找，向上取最接近规格的申请值及限制值。

  例如2000节点和2w个Pod的场景下，“目标节点数 * 目标Pod规模”等于4000w，向上取最接近的规格为700/7w（“集群节点数 * Pod规模”等于4900w），因此建议CPU申请值为4000m，限制值为5500m。

- 内存申请值：建议每1000个节点分配2.4G内存，每1w个Pod分配1G内存，二者叠加进行计算。（该计算方法相比表1-54中的建议值会存在一定的误差，通过查表或计算均可）

  即：内存申请值 = 目标节点数/1000 * 2.4G + 目标Pod规模/1w * 1G。

  例如2000节点和2w个Pod的场景下，内存申请值 = 2 * 2.4G + 2 * 1G = 6.8G

表 1-54 volcano-controller 和 volcano-scheduler 的建议值

集群节点数/Pod规模	CPU Request(m)	CPU Limit(m)	Memory Request(Mi)	Memory Limit(Mi)
50/5k	500	2000	500	2000
100/1w	1000	2500	1500	2500
200/2w	1500	3000	2500	3500
300/3w	2000	3500	3500	4500
400/4w	2500	4000	4500	5500
500/5w	3000	4500	5500	6500
600/6w	3500	5000	6500	7500
700/7w	4000	5500	7500	8500

步骤3 配置volcano默认调度器配置参数，详情请参见表1-55。

```
colocation_enable: ''
default_scheduler_conf:
  actions: 'allocate, backfill'
  tiers:
    - plugins:
        - name: 'priority'
        - name: 'gang'
        - name: 'conformance'
    - plugins:
        - name: 'drf'
        - name: 'predicates'
        - name: 'nodeorder'
    - plugins:
        - name: 'cce-gpu-topology-predicate'
        - name: 'cce-gpu-topology-priority'
        - name: 'cce-gpu'
    - plugins:
        - name: 'nodelocalvolume'
        - name: 'nodeemptydirvolume'
        - name: 'nodeCSIscheduling'
        - name: 'networkresource'
```

表 1-55 Volcano 插件配置参数说明

插件	功能	参数说明	用法演示

binpack	将Pod调度到资源使用较高的节点以减少资源碎片	• binpack.weight：binpack插件本身在所有插件打分中的权重
		• binpack.cpu：CPU资源在资源比重的比例，默认是1
		• binpack.memory：memory资源在所有资源中的比例，默认是1
		• binpack.resources：资源类型。
			- plugins:
			  - name: binpack
			  arguments:
			  binpack.weight: 10
			  binpack.cpu: 1
			  binpack.memory: 1
			  binpack.resources: nvidia.com/gpu, example.com/foo
			
			binpack.resources.nvidia.com/gpu: 2
			
			binpack.resources.example.com/foo: 3

conformance	跳过关键Pod，比如在kube-system命名空间的Pod，防止这些Pod被驱逐	-	-
gang	将一组Pod看做一个整体去分配资源	-	-
priority	使用用户自定义负载的优先级进行调度	-	-

overcommit	将集群的资源放到一定倍数后调度，提高负载入队效率。负载都是deployment的时候，建议去掉此插件或者设置扩大因子为2.0。	overcommit-factor: 扩大因子，默认是1.2	- plugins:
			  - name: overcommit
			  arguments:
			  overcommit-factor: 2.0

drf	根据作业使用的主导资源份额进行调度，用的越少的优先	-	-
predicates	预选节点的常用算法，包括节点亲和，Pod亲和，污点容忍，node ports重复，volume limits，volume zone匹配等一系列基础算法	-	-

插件	功能	参数说明	用法演示
nodeorder	优选节点的常用算法	• nodeaffinity.weight：节点亲和性优先调度，默认值是1 • podaffinity.weight：Pod亲和性优先调度，默认值是1 • leastrequested.weight：资源分配最少的的节点优先，默认值是1 • balancedresource.weight：node上面的不同资源分配平衡的优先，默认值是1 • mostrequested.weight：资源分配最多的的节点优先，默认值是0 • tainttoleration.weight：污点容忍高的优先调度，默认值是1 • imagelocality.weight：node上面有Pod需要镜像的优先调度，默认值是1 • selectorspread.weight: 把Pod均匀调度到不同的节点上，默认值是0 • volumebinding.weight: local pv延迟绑定调度，默认值是1 • podtopologyspread.weight: Pod拓扑调度，默认值是2	- plugins: - name: nodeorder arguments: leastrequested.weight: 1 mostrequested.weight: 0 nodeaffinity.weight: 1 podaffinity.weight: 1 balancedresource.weight: 1 tainttoleration.weight: 1 imagelocality.weight: 1 volumebinding.weight: 1 podtopologyspread.weight: 2
cce-gpu-topology-predicate	GPU拓扑调度预选算法	-	-
cce-gpu-topology-priority	GPU拓扑调度优选算法	-	-

插件	功能	参数说明	用法演示
cce-gpu	结合UCS的GPU插件支持GPU资源分配，支持小数GPU配置	-	-
numaaware	numa拓扑调度	weight: 插件的权重	-
networkresource	支持预选过滤ENI需求节点，参数由CCE传递，不需要手动配置	NetworkType: 网络类型（eni或者vpc-router类型）	-
nodelocalvolume	支持预选过滤不符合local volume需求节点	-	-
nodeemptydirvolume	支持预选过滤不符合emptydir需求节点	-	-
nodeCSIscheduling	支持预选过滤everest组件异常节点	-	-

步骤4 单击“安装”。

----结束

在控制台中修改 volcano-scheduler 配置

Volcano允许用户在安装，升级，编辑时，编写Volcano调度器配置信息，并将配置内容同步到volcano-scheduler-configmap里。

当前小节介绍如何使用自定义配置，以便用户让volcano-scheduler能更适合自己的场景。

说明

仅Volcano 1.7.1及以上版本支持该功能。在新版插件界面上合并了原plugins.eas_service和resource_exporter_enable等选项，以新选项default_scheduler_conf代替。

您可登录CCE控制台，单击集群名称进入集群，单击左侧导航栏的“插件管理”，在右侧找到**Volcano**，单击“安装”或“升级”，并在“参数配置”中设置Volcano调度器配置参数。

- 使用resource_exporter配置，示例如下：

```
{
  "ca_cert": "",
  "default_scheduler_conf": {
    "actions": "allocate, backfill",
    "tiers": [
      {
        "plugins": [
```



```
            {
               "name": "priority"
            },
            {
               "name": "gang"
            },
            {
               "name": "conformance"
            }
         ]
      },
      {
         "plugins": [
            {
               "name": "drf"
            },
            {
               "name": "predicates"
            },
            {
               "name": "nodeorder"
            }
         ]
      },
      {
         "plugins": [
            {
               "name": "cce-gpu-topology-predicate"
            },
            {
               "name": "cce-gpu-topology-priority"
            },
            {
               "name": "cce-gpu"
            },
            {
               "name": "numa-aware" # add this also enable resource_exporter
            }
         ]
      },
      {
         "plugins": [
            {
               "name": "nodelocalvolume"
            },
            {
               "name": "nodeemptydirvolume"
            },
            {
               "name": "nodeCSIscheduling"
            },
            {
               "name": "networkresource"
            }
         ]
      }
   ]
  },
  "server_cert": "",
  "server_key": ""
}
```

开启后可以同时使用volcano-scheduler的numa-aware插件功能和resource_exporter功能。

- 使用eas_service配置，示例如下：

```
{
  "ca_cert": "",
  "default_scheduler_conf": {
     "actions": "allocate, backfill",
```



```
    "tiers": [
      {
        "plugins": [
          {
            "name": "priority"
          },
          {
            "name": "gang"
          },
          {
            "name": "conformance"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "drf"
          },
          {
            "name": "predicates"
          },
          {
            "name": "nodeorder"
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "cce-gpu-topology-predicate"
          },
          {
            "name": "cce-gpu-topology-priority"
          },
          {
            "name": "cce-gpu"
          },
          {
            "name": "eas",
            "custom": {
              "availability_zone_id": "",
              "driver_id": "",
              "endpoint": "",
              "flavor_id": "",
              "network_type": "",
              "network_virtual_subnet_id": "",
              "pool_id": "",
              "project_id": "",
              "secret_name": "eas-service-secret"
            }
          }
        ]
      },
      {
        "plugins": [
          {
            "name": "nodelocalvolume"
          },
          {
            "name": "nodeemptydirvolume"
          },
          {
            "name": "nodeCSIscheduling"
          },
          {
            "name": "networkresource"
          }
        ]
```

```
        }
      ]
    },
    "server_cert": "",
    "server_key": ""
}
```

- 使用ief配置，示例如下：

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
                    },
                    {
                        "name": "predicates"
                    },
                    {
                        "name": "nodeorder"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "cce-gpu-topology-predicate"
                    },
                    {
                        "name": "cce-gpu-topology-priority"
                    },
                    {
                        "name": "cce-gpu"
                    },
                    {
                        "name": "ief",
                        "enableBestNode": true
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "nodelocalvolume"
                    },
                    {
                        "name": "nodeemptydirvolume"
                    },
                    {
                        "name": "nodeCSIscheduling"
                    },
                    {
                        "name": "networkresource"
                    }
```



```
            ]
          }
        ]
      },
      "server_cert": "",
      "server_key": ""
}
```

保留原 volcano-scheduler-configmap 配置

假如在某场景下希望插件升级后时沿用原配置，可参考以下步骤：

步骤1 查看原volcano-scheduler-configmap配置，并备份。

示例如下：

```
# kubectl edit cm volcano-scheduler-configmap -n kube-system
apiVersion: v1
data:
  default-scheduler.conf: |-
    actions: "enqueue, allocate, backfill"
    tiers:
    - plugins:
      - name: priority
      - name: gang
      - name: conformance
    - plugins:
      - name: drf
      - name: predicates
      - name: nodeorder
      - name: binpack
        arguments:
          binpack.cpu: 100
          binpack.weight: 10
          binpack.resources: nvidia.com/gpu
          binpack.resources.nvidia.com/gpu: 10000
    - plugins:
      - name: cce-gpu-topology-predicate
      - name: cce-gpu-topology-priority
      - name: cce-gpu
    - plugins:
      - name: nodelocalvolume
      - name: nodeemptydirvolume
      - name: nodeCSIscheduling
      - name: networkresource
```

步骤2 在控制台“参数配置”中填写自定义修改的内容：

```
{
    "ca_cert": "",
    "default_scheduler_conf": {
        "actions": "enqueue, allocate, backfill",
        "tiers": [
            {
                "plugins": [
                    {
                        "name": "priority"
                    },
                    {
                        "name": "gang"
                    },
                    {
                        "name": "conformance"
                    }
                ]
            },
            {
                "plugins": [
                    {
                        "name": "drf"
```



```
            },
            {
               "name": "predicates"
            },
            {
               "name": "nodeorder"
            },
            {
               "name": "binpack",
               "arguments": {
                  "binpack.cpu": 100,
                  "binpack.weight": 10,
                  "binpack.resources": "nvidia.com/gpu",
                  "binpack.resources.nvidia.com/gpu": 10000
               }
            }
         ]
      },
      {
         "plugins": [
            {
               "name": "cce-gpu-topology-predicate"
            },
            {
               "name": "cce-gpu-topology-priority"
            },
            {
               "name": "cce-gpu"
            }
         ]
      },
      {
         "plugins": [
            {
               "name": "nodelocalvolume"
            },
            {
               "name": "nodeemptydirvolume"
            },
            {
               "name": "nodeCSIscheduling"
            },
            {
               "name": "networkresource"
            }
         ]
      }
   ]
  },
  "server_cert": "",
  "server_key": ""
}
```

说明

使用该功能时会覆盖原volcano-scheduler-configmap中内容，所以升级时务必检查是否在volcano-scheduler-configmap做过修改。如果是，需要把修改内容同步到升级界面里。

----结束

相关操作

- 动态资源超卖
- NUMA亲和性调度

版本记录

须知

建议升级到跟集群配套的最新volcano版本。

表 1-56 集群版本配套关系

集群版本	支持的插件版本
v1.25	1.7.1、1.7.2
v1.23	1.7.1、1.7.2
v1.21	1.7.1、1.7.2
v1.19.16	1.3.7、1.3.10、1.4.5、1.7.1、1.7.2
v1.19	1.3.7、1.3.10、1.4.5
v1.17（停止维护）	1.3.7、1.3.10、1.4.5
v1.15（停止维护）	1.3.7、1.3.10、1.4.5

表 1-57 CCE 插件版本记录

插件版本	支持的集群版本	更新特性
1.9.1	/v1.19.16.*|v1.21.*|v1.23.*|v1.25.*/	• 修复networkresource插件计数pipeline pod占用subeni问题 • 修复binpack插件对资源不足节点打分问题 • 修复对结束状态未知的pod的资源的处理 • 优化事件输出 • 默认高可用部署
1.7.2	/v1.19.16.*|v1.21.*|v1.23.*|v1.25.*/	• Volcano 支持 Kubernetes 1.25版本 • 提升Volcano调度性能。
1.7.1	/v1.19.16.*|v1.21.*|v1.23.*|v1.25.*/	Volcano 支持 Kubernetes 1.25版本
1.6.5	/v1.19.*|v1.21.*|v1.23.*/	• 支持作为CCE的默认调度器 • 支持混部场景下统一调度
1.4.5	/v1.17.*|v1.19.*|v1.21.*/	• volcano-scheduler的部署方式由statefulset调整为deployment，修复节点异常时Pod无法自动迁移的问题

插件版本	支持的集群版本	更新特性
1.4.2	/v1.15.*|v1.17.*|v1.19.*|v1.21.*/	• 修复跨GPU分配失败问题 • 适配更新后的EAS API
1.3.3	/v1.15.*|v1.17.*|v1.19.*|v1.21.*/	• 修复GPU异常导致的调度器崩溃问题；修复特权Init容器准入失败问题
1.3.1	/v1.15.*|v1.17.*|v1.19.*/	• 升级Volcano框架到最新版本 • 支持Kubernetes 1.19版本 • 添加numa-aware插件 • 修复多队列场景下Deployment扩缩容的问题 • 调整默认开启的算法插件
1.2.5	/v1.15.*|v1.17.*|v1.19.*/	• 修复某些场景下OutOfcpu的问题 • 修复queue设置部分capability情况下Pod无法调度问题 • 支持volcano组件日志时间与系统时间保持一致 • 修复队列间多抢占问题 • 修复ioaware插件在某些极端场景下结果不符合预期的问题 • 支持混合集群
1.2.3	/v1.15.*|v1.17.*|v1.19.*/	• 修复因为精度不够引发的训练任务OOM的问题 • 修复CCE1.15以上版本GPU调度的问题，暂不支持任务分发时的CCE版本滚动升级 • 修复特定场景下队列状态不明的问题 • 修复特定场景下作业挂载PVC panic的问题 • 修复GPU作业无法配置小数的问题 • 添加ioaware插件 • 添加ring controller

1.6.11.5 gpu-device-plugin

插件简介

gpu-device-plugin插件是支持在容器中使用GPU显卡的设备管理插件，集群中使用GPU节点时必须安装本插件。

约束与限制

- 下载的驱动必须是后缀为“.run”的文件。
- 仅支持Nvidia Tesla驱动，不支持GRID驱动。
- 安装或重装插件时，需要保证驱动下载链接正确且可正常访问，插件对链接有效性不做额外校验。
- gpu-device-plugin插件仅提供驱动的下载及安装脚本执行功能，插件的状态仅代表插件本身功能正常，与驱动是否安装成功无关。
- 如您使用A100/A800的多GPU卡机型，您需要手动安装与版本驱动对应的nvidia-fabricmanager服务才可以正常使用，详情请参见**安装nvidia-fabricmanager服务**。

安装插件

步骤1 登录UCS控制台，单击集群名称进入集群，在左侧导航栏中选择“插件中心”。

步骤2 在“可安装插件”中找到**gpu-device-plugin**，单击“安装”。

步骤3 在安装插件页面，填写插件配置。

- 插件规格：可配置“默认”或“自定义”规格，请根据实际情况选择。
- 容器：选择“自定义”规格时支持设置。
- Nvidia驱动：您可使用CCE提供的驱动地址或手动填写自定义Nvidia驱动的地址，集群下全部GPU节点将使用相同的驱动。

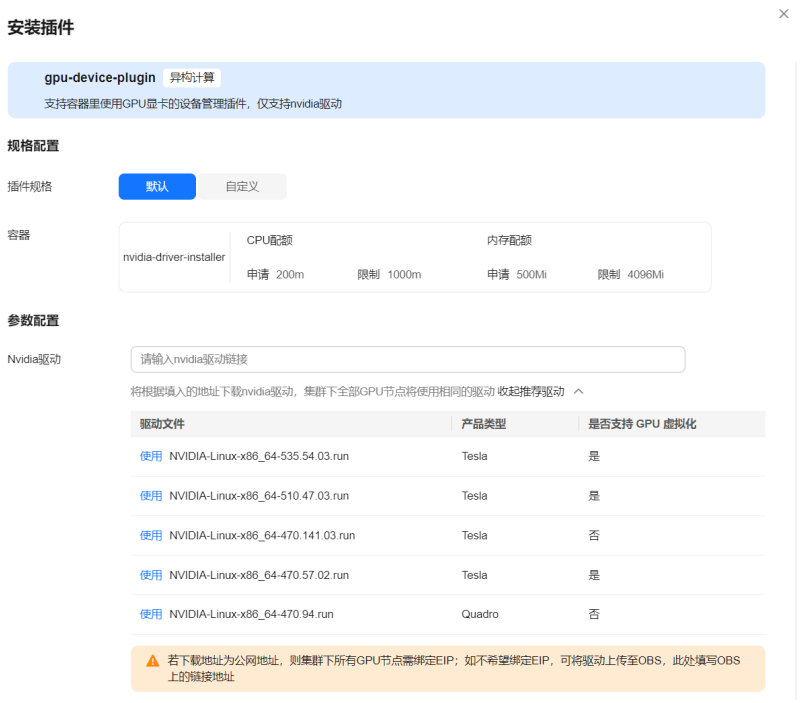
  GPU虚拟化功能仅支持470.57.02、470.103.01、470.141.03、510.39.01、510.47.03版本的GPU驱动。

  建议您使用CCE提供的驱动地址，以满足驱动版本的要求。

须知

- 如果下载链接为公网地址，例如地址为nvidia官网地址https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run，则各GPU节点均需要绑定EIP。获取驱动链接方法请参考**获取驱动链接-公网地址**。
- 若下载链接为OBS上的链接，无需绑定EIP 。获取驱动链接方法请参考**获取驱动链接-OBS地址**。
- 请确保Nvidia驱动版本与GPU节点适配。
- 更改驱动版本后，需要重启节点才能生效。
- 对于linux 5.x内核系统，如Huawei Cloud EulerOS 2.0或ubuntu 22.04，建议使用470及以上版本驱动。

图 1-42 安装 gpu-device-plugin



步骤4 单击“安装”，安装gpu-device-plugin插件的任务即可提交成功。

----结束

验证插件

插件安装完成后，在GPU节点及调度了GPU资源的容器中执行nvidia-smi命令，验证GPU设备及驱动的可用性。

GPU节点

```
cd /usr/local/nvidia/bin &&./nvidia-smi
```

容器

```
nvidia-smi
```

能正常返回GPU信息，说明设备可用，插件安装成功。

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.118.02   Driver Version: 440.118.02   CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:21:01.0 Off |                    0 |
| N/A   31C    P0    23W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

获取驱动链接-公网地址

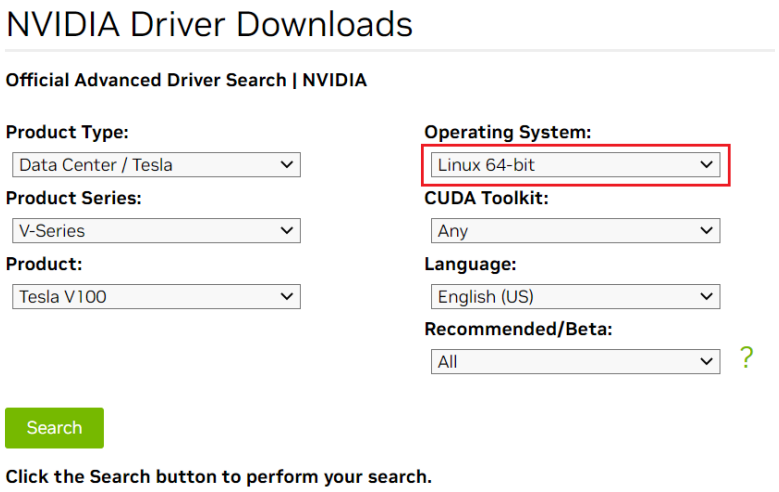
步骤1 登录CCE控制台。

步骤2 创建节点，在节点规格处选择要创建的GPU节点，选中后下方显示的信息中可以看到节点的GPU显卡型号。

步骤3 登录到**nvidia**网站。

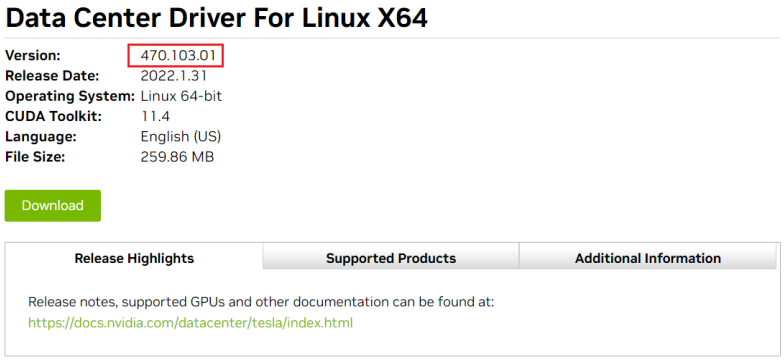
步骤4 如**图1-43**所示，在“NVIDIA驱动程序下载”框内选择对应的驱动信息。其中“操作系统”必须选**Linux 64-bit**。

图 1-43 参数选择



步骤5 驱动信息确认完毕，单击“搜索”按钮，会跳转到驱动信息展示页面，该页面会显示驱动的版本信息如**图1-44**，单击“下载”到下载页面。

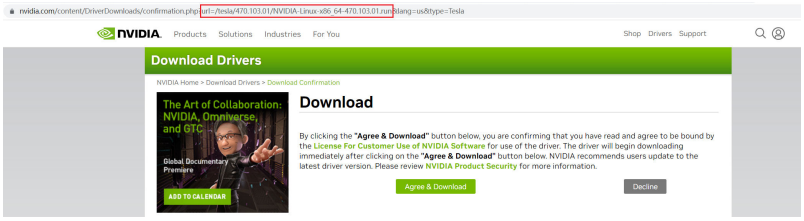
图 1-44 驱动信息



步骤6 获取驱动软件链接方式分两种：

- 方式一：如**图1-45**，在浏览器的链接中找到*url=/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run*的路径，补齐全路径为https://us.download.nvidia.com/tesla/470.103.01/NVIDIA-Linux-x86_64-470.103.01.run，该方式节点需要绑定EIP 。
- 方式二：如**图1-45**，单击“下载”按钮下载驱动，然后上传到OBS，获取软件的链接，该方式节点不需要绑定EIP。

图 1-45 获取链接



----结束

获取驱动链接-OBS 地址

步骤1 将驱动上传到对象存储服务OBS中，并将驱动文件设置为公共读，方法请参见**上传文件**。

说明

节点重启时会重新下载驱动进行安装，请保证驱动的OBS桶链接长期有效。

步骤2 在OBS管理控制台左侧导航栏选择“对象存储”。

步骤3 在桶列表单击待操作的桶，进入“概览”页面。

步骤4 在左侧导航栏，单击“对象”。

步骤5 找到目标对象，单击“更多>复制对象URL”，复制驱动链接。

图 1-46 获取链接

----结束

安装 nvidia-fabricmanager 服务

A100/A800 GPU支持 NvLink & NvSwitch，若您使用多GPU卡的机型，需额外安装与驱动版本对应的nvidia-fabricmanager服务使GPU卡间能够互联，否则可能无法正常使用GPU实例。

本文以驱动版本470.103.01为例，您可参考以下步骤进行安装，请根据实际情况需要替换驱动版本。

步骤1 登录需要安装nvidia-fabricmanager服务的GPU节点，该节点需绑定EIP用以下载nvidia-fabricmanager服务。

步骤2 安装与驱动版本对应的nvidia-fabricmanager服务，您可通过**官方下载**操作系统和驱动版本对应的安装包。

- **CentOS操作系统**

  以CentOS 7为例：

  ```
  driver_version=470.103.01
  wget https://developer.download.nvidia.cn/compute/cuda/repos/rhel7/x86_64/cuda-drivers-fabricmanager-${driver_version}-1.x86_64.rpm
  rpm -ivh nvidia-fabric-manager-${driver_version}-1.x86_64.rpm
  ```

- **Ubuntu等其他操作系统**

  以Ubuntu 18.04为例：

  ```
  driver_version=470.103.01
  driver_version_main=$(echo $driver_version | awk -F '.' '{print $1}')
  wget https://developer.download.nvidia.cn/compute/cuda/repos/ubuntu1804/x86_64/nvidia-fabricmanager-${driver_version_main}_${driver_version}-1_amd64.deb
  dpkg -i nvidia-fabricmanager-${driver_version_main}_${driver_version}-1_amd64.deb
  ```

步骤3 启动nvidia-fabricmanager服务。

```
systemctl enable nvidia-fabricmanager
systemctl start nvidia-fabricmanager
```

步骤4 查看nvidia-fabricmanager服务状态。

```
systemctl status nvidia-fabricmanager
```

----结束

相关链接

- **GPU插件及驱动相关问题的排查思路**
- **工作负载异常：GPU相关**
- **GPU调度**

1.6.11.6 e-backup 插件

插件简介

E-Backup是云原生存储系统(Everest2.0)中负责云原生应用数据保护的子系统。它支持用户将应用数据（k8s资源）和业务数据（pv卷中的数据）备份到OBS桶中，也允许用户将某次备份数据恢复到指定的K8s集群中。

E-Backup通过备份和恢复两个子功能提供对以下使用场景的支持：

- 单集群下的容灾

  周期性地对集群内部署的应用进行备份，在集群或者应用被破坏时通过恢复功能将应用重新部署到集群中，继续向外提供服务，实现应用的容灾。

- 同集群/跨集群的克隆

  对于需要大批量部署到多个集群中的应用，特别是应用已经在某个集群工作一段时间后需要增添实例的情况。首先对处于工作状态的应用进行备份，随后恢复到同集群的不同Namespace下或者其他集群中，实现应用的克隆。

- 跨集群/跨云的迁移

  由于网络、成本、业务地点变动等原因，需要将应用从某个集群迁移到跨Region的另一个集群，或者从其他云的集群迁入CCE。对迁出集群中的应用进行备份，而后恢复到迁入集群中，实现应用的迁移。

约束与限制

- 目标集群Kubernetes版本需为1.15及以上，且集群中至少包含一个可用节点。
- 集群在安装插件时，需要保证集群可正常拉取SWR镜像。
- 备份/恢复过程中，需尽量保证集群处于稳态，不要触发增、删、改等变更行为，以免出现备份/恢复失败或不完整的情况。若集群发生变更，建议等15分钟后，集群处于稳态，再做备份操作。
- E-Backup插件集成开源的Restic组件完成PV数据备份，会对备份时间点的数据做自有快照，并上传数据，不影响用户后续数据的读写，但Restic不进行文件内容的校验和业务一致性校验，其特性遵循restic约束。
- Restic组件占用内存大小与初次备份的PV卷数据大小有关。若PV卷数据大于300G，建议采用云存储提供的迁移方式。若使用应用数据管理功能迁移大量PV数据，可修改restic实例的资源配额，具体操作方式请参见**修改插件资源配额**。
- E-Backup插件遵循开源Velero和Restic插件的约束，例如在恢复过程中Service会清除ClusterIP以适应源集群和目标集群间的差异。
- 若在CCE集群中使用了扩展加密的Secret类型（ cfe/secure-opaque ），在恢复到其他集群时，需要提前手动创建同名、同类型的Secret（ 不同集群加解密物料不同 ），以免恢复的应用无法成功运行。

安装插件

须知

安装E-Backup插件后，请谨慎卸载，否则可能导致已有的备份无法恢复。因为E-Backup插件在执行备份恢复任务时，依赖自定义资源**BackupStorageLocation**及其Secret，该资源在卸载后重新安装会发生更改。

步骤1 登录集群控制台。

步骤2 在左侧导航栏中单击“插件管理”，在可安装插件栏中单击E-Backup插件下的“安装”按钮。

步骤3 参照**表1-58**进行插件规格配置。

表 1-58 E-Backup 插件规格配置

参数	参数说明
插件规格	单实例部署。
容器	设置插件容器实例的资源配额。 ● velero：提供K8s元数据备份/恢复支持。 ● restic：提供应用数据存储卷备份/恢复支持。 **说明** ● 集群中需要预留足够的资源，若资源不足，插件实例将无法调度。 ● 申请值需小于等于限制值，否则无法成功创建。 ● 请根据备份/恢复数据量大小，适当调整资源限制以避免插件故障。

步骤4 进行参数配置，当前支持配置以下参数。

volumeWorkerNum：代表并发执行数据卷备份的工作数量，默认为3。

```
{
   "volumeWorkerNum": 3
}
```

步骤5 单击“安装”后，返回“插件管理”页面查看已安装插件，插件状态为“运行中”，表明该插件已在当前集群中安装成功。

----结束

修改插件资源配额

步骤1 登录集群控制台。

步骤2 在左侧导航栏中单击“插件管理”，在已安装插件栏中单击E-Backup插件下的“编辑”按钮。

步骤3 修改插件规格配置，相关参数说明请参见**表1-58**。

步骤4 单击“确定”，插件状态为“升级中”。待升级完成后，修改后的插件配置将会生效。

----结束

2 容器舰队

2.1 容器舰队概述

容器舰队

舰队是多个集群的集合，您可以使用舰队来实现关联集群的分类。舰队还可以实现多集群的统一管理，包括权限管理、安全策略、配置管理以及多集群编排等统一管理的能力。

容器舰队使用限制

- 仅**华为云账号**且具备**UCS FullAccess**权限的用户可进行舰队的创建、删除操作。
- 一个集群只能加入一个舰队。

容器舰队支持的能力范围

集群接入UCS后，您可以将其加入容器舰队并开通集群联邦能力，以进行多集群管理。针对已接入UCS的集群（无论是否加入容器舰队）、未开通集群联邦能力的容器舰队、已开通集群联邦能力的容器舰队，UCS所支持的能力范围有所不同，如**表2-1**所示。

表 2-1 容器舰队支持的能力范围

能力	已接入UCS的集群	未开通集群联邦能力的容器舰队	已开通集群联邦能力的容器舰队
集群联邦多集群管理	-	-	√
流量分发	√	-	-
可观测性	√	√	√
服务网格	-	√	√
云原生服务中心	√	-	-

能力	已接入UCS的集群	未开通集群联邦能力的容器舰队	已开通集群联邦能力的容器舰队
策略中心	√	√	√
配置管理	√	-	-
流水线	-	-	√
权限管理	√	√	√

2.2 管理容器舰队

本小节为您介绍创建容器舰队并为其添加集群、关联权限，移出、注销舰队中的集群和删除容器舰队的具体操作步骤，指导您使用UCS容器舰队能力。

创建舰队

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”，在“容器舰队”页签下单击“创建容器舰队”。

步骤2 填写舰队信息。

图 2-1 创建容器舰队

- 舰队名称：自定义舰队的名称，需以小写字母开头，由小写字母、数字、中划线（-）组成，且不能以中划线（-）结尾。
- 添加集群：列表中显示当前未加入舰队的集群，可以在创建舰队时添加集群，也可以在舰队创建完成后添加。如不选择任何集群，则会创建一个空的舰队，完成创建后请参考**添加集群**。
- 描述：添加舰队的描述信息。

说明

集群加入容器舰队后，将获得所选容器舰队的权限，失去原有权限。

步骤3 单击“确定”，创建舰队。

----结束

添加集群

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在目标舰队栏中单击“添加集群”，或单击右上角的 按钮。

您也可以单击舰队名称进入舰队详情页，在“容器集群”页面单击右上角“添加集群”。

图 2-2 为舰队添加集群

步骤3 勾选一个或多个已有集群。一个集群只能加入一个舰队，因此列表中显示的集群均为未加入舰队的集群。

图 2-3 添加集群

说明

- 集群加入容器舰队，将拥有所选容器舰队的权限，但会失去原有赋予的权限。
- 如果该舰队开通了集群联邦，集群会自动接入集群联邦。关于集群联邦的介绍，请参见**开通集群联邦**章节。

步骤4 单击“确定”，完成集群添加。

----结束

关联权限

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在目标舰队栏中，单击右上角的  按钮。

图 2-4 为舰队关联权限

步骤3 在弹出的页面单击“修改容器舰队权限”或“关联权限”，打开修改权限页面，将已创建好的权限和舰队的命名空间关联起来。

图 2-5 修改权限

- 命名空间：支持“全部命名空间”和“指定命名空间”。全部命名空间包括当前舰队已有的命名空间和舰队后续新增的命名空间；“指定命名空间”即表示您自己选择命名空间的范围，UCS服务提供了几个常见的命名空间供您选择（如default、kube-system、kube-public），您也可以新增命名空间，但要自行确保新增的命名空间在集群中存在。

  请注意，选择的命名空间仅对权限中命名空间级资源生效，不影响权限中的集群资源。关于命名空间级和集群级资源的介绍，请参见**Kubernetes资源对象**章节。
- 关联权限：从下拉列表中选择权限，支持一次性选择多个权限，以达到批量授权的目的。

如果针对不同命名空间，关联的权限不同（例如：为default命名空间关联readonly权限，为development命名空间关联develop权限），可以单击 + 按钮添加多组授权关系。

步骤4 单击“确定”，完成权限的关联。

如果后续需要修改舰队的权限，采用同样的方法，重新选择命名空间和权限即可。

----结束

移出舰队中的集群

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下单击舰队名称，进入舰队详情页。

步骤3 在左侧导航栏中选择“容器集群”，在目标集群栏中，单击右上角的按钮。

步骤4 仔细阅读注意事项，确认风险后单击“确定”。

集群移出舰队后，会展示在“未加入舰队的集群”页签，您仍然可以将集群重新添加至舰队中。具体操作请参见**管理未加入舰队的集群**。

----结束

注销舰队中的集群

步骤1 登录UCS控制台，在左侧导航栏中单击“容器舰队”。

步骤2 在“容器舰队”页签下单击舰队名称，进入舰队详情页。

步骤3 在左侧导航栏中选择“容器集群”，在目标集群栏中，单击右上角的按钮。

步骤4 弹出“注销集群”对话框，仔细阅读注意事项，确认风险后单击“确定”。

步骤5 （可选）对于附着集群来说，注销集群成功后，还需前往目标集群卸载代理组件。

kubectl -n kube-system delete deployments/proxy-agent secret/proxy-agent-cert

步骤6 （可选）对于本地集群来说，注销集群成功后，您可以手动执行卸载命令，在本地主机环境中删除集群，清理资源：

./ucs-ctl delete cluster [集群名称]

说明

如果命令执行失败，请参考**如何手动清理本地集群节点?** 处理。

----结束

删除舰队

如果容器舰队不再使用，可以将其删除。删除时有两个限制条件：舰队中无集群；舰队已关闭集群联邦。如果舰队中有集群，可以先将集群**移出舰队**，再添加至其他舰队中；如果舰队已开通集群联邦，请参考**关闭集群联邦**关闭。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到目标舰队，单击右上角按钮。

步骤3 在弹出的提示框中单击“是”，完成舰队的删除。

----结束

2.3 管理未加入舰队的集群

注册时未选择舰队的集群，或者从舰队中移出的集群，会展示在“未加入舰队的集群”页签中。本小节指导您管理未加入舰队的集群，进行加入舰队、关联权限等操作。

加入舰队

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 选择“未加入舰队的集群”页签，在目标集群栏中单击右上角的按钮。

步骤3 选择一个容器舰队。集群加入舰队后，集群的权限将被舰队的替换，请谨慎操作。

步骤4 选择容器舰队后，界面会提示当前权限和调整后权限，确认无误后单击“确定”。

集群加入舰队成功后，会展示在对应的舰队中，后续由舰队来统一管理。

----结束

关联权限

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 选择“未加入舰队的集群”页签，在目标集群栏中单击右上角的按钮。

图 2-6 集群列表

步骤3 在弹出的页面单击“关联权限”，打开“修改权限”页面，将已创建好的权限和集群的命名空间关联起来。

- 命名空间：支持“全部命名空间”和“指定命名空间”。全部命名空间包括当前集群已有的命名空间和集群后续新增的命名空间；“指定命名空间”即表示您自己选择命名空间的范围，UCS服务提供了几个常见的命名空间供您选择（如default、kube-system、kube-public），您也可以新增命名空间，但要自行确保新增的命名空间在集群中存在。

  请注意，选择的命名空间仅对权限中命名空间级资源生效，不影响权限中的集群资源。关于命名空间级和集群级资源的介绍，请参见**Kubernetes资源对象**章节。
- 关联权限：从下拉列表中选择权限，支持一次性选择多个权限，以达到批量授权的目的。

如果针对不同命名空间，关联的权限不同（例如：为default命名空间关联readonly权限，为development命名空间关联develop权限），可以单击 + 按钮添加多组授权关系。

步骤4 单击“确定”，完成权限的关联。

如果后续需要修改集群的权限，采用同样的方法，重新选择命名空间和权限即可。

----结束

注销集群

步骤1 登录UCS控制台，在左侧导航栏中单击“容器舰队”。

步骤2 选择“未加入舰队的集群”页签，在目标集群栏中单击右上角的 按钮。

步骤3 弹出“注销集群”对话框，仔细阅读注意事项，确认风险后单击“确定”。

步骤4 （可选）对于附着集群来说，注销集群成功后，还需前往目标集群卸载代理组件。

kubectl -n kube-system delete deployments/proxy-agent secret/proxy-agent-cert

步骤5 （可选）对于本地集群来说，注销集群成功后，您可以手动执行卸载命令，在本地主机环境中删除集群，清理资源：

./ucs-ctl delete cluster [集群名称]

说明

如果命令执行失败，请参考**如何手动清理本地集群节点?** 处理。

----结束

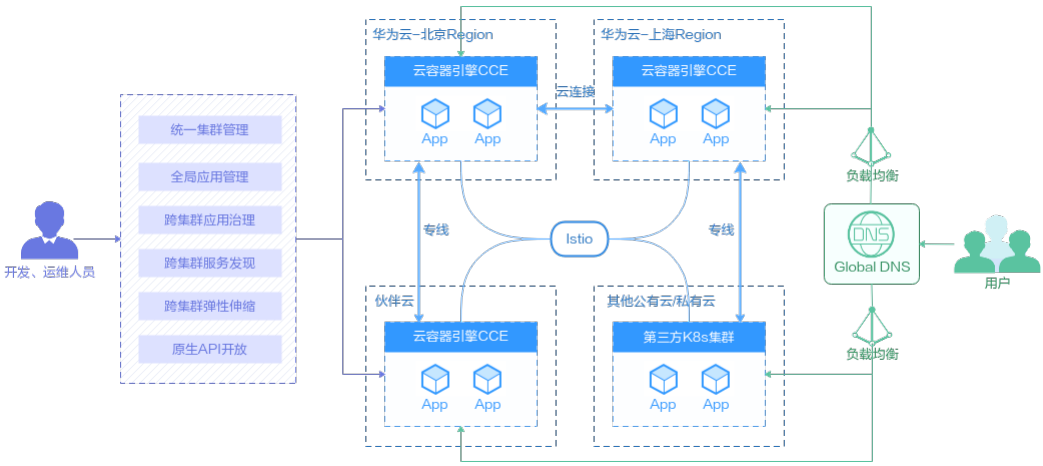
3 集群联邦

3.1 集群联邦概述

集群联邦

集群联邦是基于CNCF项目**Karmada**所提供的多云容器编排能力，旨在管理跨云、跨地域场景下的多集群应用，为您提供多集群统一管理、应用部署、服务发现、弹性伸缩、故障迁移等能力。

图 3-1 集群联邦架构



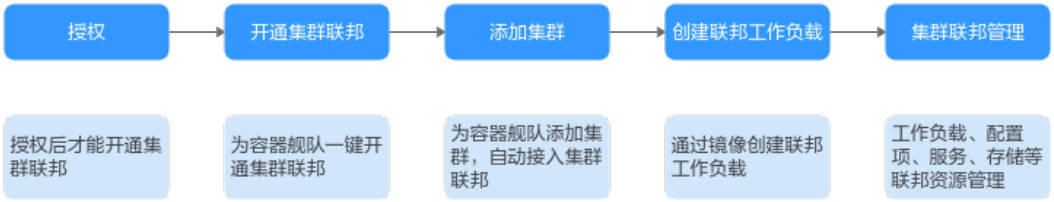
集群联邦使用限制

仅**华为云**或具备**UCS FullAccess**权限的用户可进行集群联邦的开通、关闭操作。

集群联邦使用流程

集群联邦的使用流程如**图3-2**所示。

图 3-2 集群联邦使用流程



集群联邦能力与容器舰队绑定，若您需要使用集群联邦进行多集群管理，请进行以下操作：

- 将需要管理的集群**接入UCS**，并将其**加入容器舰队**。
- 为容器舰队**开通集群联邦**能力，并**通过kubectl连接集群联邦**。
- （可选）为了使用最新的功能，您可以将旧版本的集群联邦**升级至最新版本**。

3.2 开通集群联邦

开通集群联邦

集群联邦和容器舰队绑定在一起，只需要为容器舰队一键开通集群联邦，就可以方便地操作集群联邦了。

集群联邦的开通包含两个阶段：一、开通集群联邦，二、集群接入联邦。容器舰队开通集群联邦能力后，容器舰队内的成员集群将自动接入联邦。

开通集群联邦有配额限制，并且对容器舰队中的集群有一些约束，**请在开通前仔细阅读**，避免集群联邦开通失败。

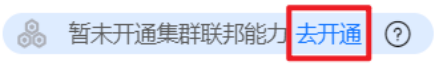
表 3-1 集群约束

项目	约束
集群版本	容器舰队中的所有集群版本必须为1.19及以上。
集群状态	容器舰队中的所有集群状态必须为“运行中”。
集群网络状态	• CCE集群、CCE Turbo集群：CCE集群所在region为新加坡的，开通联邦时UCS将自动在集群所属VPC下创建VPCEP打通网络，非该region则需要集群节点具备公网访问能力，例如为节点绑定公网地址，用以拉取公网镜像。 • 其他集群：成功接入UCS即可。
配额	集群联邦配额为1，即只能为一个容器舰队开通集群联邦。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到目标舰队，页面提示暂未开通集群联邦能力，单击“去开通”按钮。

图 3-3 开通集群联邦



步骤3 在弹出的提示框中单击“确定”，等待集群联邦开通成功。

当集群不满足约束条件时，界面会弹出报错信息，请按照提示修改，然后重新开通集群联邦。

开通集群联邦大约需要10分钟，请耐心等待。您可以单击集群联邦状态，查看详细的开通进度。开通成功后，容器舰队顶部的提示信息变为“集群联邦能力已开通 ，集群接入成功”。

----结束

添加集群

容器舰队开通集群联邦后，可以继续为舰队添加集群，添加后，集群会自动接入集群联邦。一个集群联邦最多可接入20个集群。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在目标舰队栏中单击“添加集群”，或单击右上角的 按钮。

您也可以单击舰队名称进入舰队详情页，在“容器集群”页面单击右上角“添加集群”。

步骤3 勾选一个或多个已有集群。一个集群只能加入一个舰队，因此列表中显示的集群均为未加入舰队的集群。

说明

请确保所选择的集群符合**表3-1**中的约束条件，否则会出现集群添加成功，但是接入集群联邦失败的情况。如果出现这种情况，请参考**舰队开通集群联邦后，添加集群报错如何解决?** 进行修复。

步骤4 单击“确定”，完成集群添加。

----结束

联邦管理

容器舰队开通集群联邦成功后，舰队详情页面将自动解锁联邦管理功能，如**图3-4**所示。

图 3-4 联邦管理

接下来，您可以创建联邦工作负载、服务、存储等联邦资源，完成业务部署。还可以根据业务需要完成一些高阶操作，比如多集群应用的多活容灾、弹性伸缩。

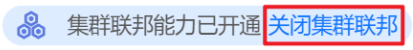
关闭集群联邦

如果不需要继续使用集群联邦，可以将其关闭。关闭集群联邦后，运行在工作负载上的服务不受影响，请放心操作。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到目标舰队，单击“关闭集群联邦”按钮。

图 3-5 关闭集群联邦



步骤3 在弹出的提示框中单击“确定”，关闭集群联邦。

----结束

常见问题

舰队开通集群联邦后，添加集群报错如何解决?

3.3 通过 kubectl 连接集群联邦

本文介绍如何使用kubectl连接集群联邦。

权限说明

kubectl访问集群联邦是通过集群联邦上生成的配置文件（kubeconfig.json）进行认证。kubeconfig.json文件内包含用户信息，UCS根据用户信息的权限判断kubectl有权限访问哪些Kubernetes资源。即哪个用户获取kubeconfig.json文件，kubeconfig.json就拥有哪个用户的信息，这样使用kubectl访问时就拥有这个用户的权限。

约束限制

- 出于安全考虑，集群联邦apiserver不提供公网访问地址。UCS通过在您提供的VPC和子网中创建终端节点，并将该终端节点连接到集群联邦apiserver，来打通访问联邦的网络。对于每个集群联邦，UCS在同一VPC中仅会创建一个终端节点。如果VPC中已有连接该联邦apiserver的终端节点，则会进行复用。
- 当前仅支持“亚太-新加坡”区域的项目下载kubectl配置文件。

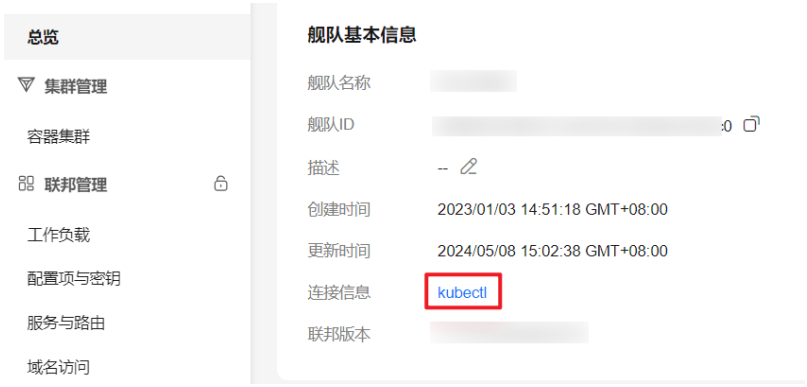
前提条件

- 使用kubectl连接集群联邦，需要先**开通集群联邦**，并保证集群联邦已开启且状态正常。
- 当前仅支持从VPC中的客户端通过kubectl连接集群联邦，如您在VPC中没有客户端机器，请创建一台。
- 已下载kubectl命令行工具并上传至客户端机器。kubectl下载地址请参见**Kubernetes 版本发布页面**。
- 确保至少具有“iam:clustergroups:get”自定义策略。

通过 kubectl 连接集群联邦

步骤1 登录UCS控制台，单击目标容器舰队名进入“容器舰队信息”页面，在“舰队基本信息”中单击“kubectl”。

图 3-6 kubectl 连接信息



步骤2 参照页面中的提示信息，选择对应的项目名称、虚拟私有云(VPC)、控制节点子网以及有效期，单击“下载”，下载kubectl配置文件。

下载下来的文件名为kubeconfig.json。

图 3-7 kubectl 连接联邦实例

须知

- kubeconfig.json文件中存在安全泄露风险，请您务必妥善保存。
- kubectl配置文件有效期可根据实际需求选择，下拉列表内可选范围为：5年、1年、6个月、30天、15天、14天、13天...1天，最短为1天。

步骤3 在执行机上安装和配置kubectl。

1. 拷贝kubectl及其配置文件到上述所选的vpc和子网下的执行机的/home目录下。
2. 登录到您的执行机，配置kubectl。

```
cd /home
chmod +x kubectl
mv -f kubectl /usr/local/bin
mkdir -p $HOME/.kube
mv -f kubeconfig.json $HOME/.kube/config
```

----结束

集群联邦支持的资源及操作

集群联邦支持的Kubernetes资源及相关操作见表3-2。表内为“√”表明集群联邦支持对该Kubernetes资源进行该操作，表内为“部分支持”表明集群联邦部分支持对该Kubernetes资源进行该操作，表内为空则表明集群联邦不支持对该Kubernetes资源进行该操作。

表 3-2 集群联邦支持的资源及操作

组/版本	资源	GET	LIST	WATCH	CREATE	UPDATE	PATCH	DELETE
core/v1	pods	√	√	√	√	√	√	√
	pods/log	√						
	pods/exec	√			√			

组/版本	资源	GET	LIST	WATCH	CREATE	UPDATE	PATCH	DELETE
	pods/status	√						
	configmaps	√	√	√	√	√	√	√
	secrets	√	√	√	√	√	√	√
	services	√	√	√	√	√	√	√
	nodes	√	√	√		√	√	
	namespaces	√	√	√	√	√	√	√
	endpoints	√	√					
	events	√	√					
	limitranges	√	√					
	resourcequotas	√	√					
	persistentvolumeclaims	√	√					
	persistentvolumes	√	√					
	serviceaccounts	√	√					
admissionregistration.k8s.io/v1	mutatingwebhookconfigurations	√	√					
	validatingwebhookconfigurations	√	√					
apiextensions.k8s.io/v1	customresourcedefinitions	√	√	√	√	√	√	√
apiregistration.k8s.io/v1	apiservices	√	√					
apps/v1	deployments	√	√	√	√	√	√	√
	deployments/scale	√				√		
	deployments/status	√						
	daemonsets	√	√	√	√	√	√	√
	daemonsets/status	√						
	statefulsets	√	√	√	√	√	√	√
	statefulsets/status	√						
	replicasets	√	√					

组/版本	资源	GET	LIST	WATCH	CREATE	UPDATE	PATCH	DELETE
autoscaling/(v1、v2、v2beta1、v2beta2)	horizontalpodautoscalers	√	√	√	√	√	√	√
batch/v1	jobs	√	√	√	√	√	√	√
	jobs/status	√						
	cronjobs	√	√	√	√	√	√	√
	cronjobs/status	√						
discovery.k8s.io/v1	endpointslices	√	√					
events.k8s.io/v1	events	√	√					
networking.k8s.io/v1	ingresses	√	√	√	√	部分支持	部分支持	√
	ingressclasses	√	√					
	networkpolicies	√	√					
policy/(v1、v1beta1)	poddisruptionbudgets	√	√	√	√	√	√	√
rbac.authorization.k8s.io/v1	clusterrolebindings	√	√	√	√	√	√	√
	clusterroles	√	√	√	√	√	√	√
	rolebindings	√	√	√	√	√	√	√
	roles	√	√	√	√	√	√	√
storage.k8s.io/v1	storageclasses	√	√					

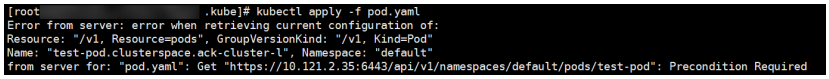
⚠ 注意

- 对于集群中的自定义资源，在集群联邦中**注册**该CRD后，才可支持通过集群联邦入口进行操作。
- Ingress对象的UPDATE和PATCH操作仅支持集群联邦控制面中的资源，不支持成员集群中的资源。

常见问题

- 如果您在访问联邦资源时出现如下错误提示，说明您没有对应资源的操作权限，请参考**集群联邦RBAC授权**添加授权。

- 如果您在访问联邦和集群内资源时出现错误提示“Precondition Required”，可能是由于网络问题或成员集群故障导致集群失联，请按以下步骤进行排查：



  a. 检查成员集群的工作状态与接入状态是否正常，具体操作可参考**附着集群接入失败如何解决？**。

  b. 检查成员集群内部署的proxy-agent是否正常工作，具体操作可参考**proxy-agent部署失败如何解决？**。

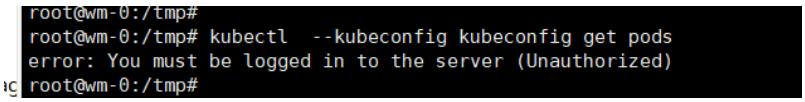
- 如果出现“no such host”问题，请按以下步骤进行排查：

  a. 查询vpcep节点是否存在，是否被误删，使用以下命令获取vpcep终端节点id：

```
server=`cat config | jq '.clusters[0].cluster.server'` echo ${server:15:36}
```

  b. 检查上述终端节点是否存在，如果存在，检查执行机是否和vpcep终端节点位于同一VPC中，二者之间网络是否连通。

- 如果出现“You must be logged in to the server (Unauthorized)”问题，请按以下步骤进行排查：



  a. 校验证书是否有误：

  将证书保存至临时文件中。

```
cd ~/.kube
cat config | jq '.clusters[0].cluster."certificate-authority-data"' | tr -d '"' | base64 -d > ca.crt
cat config | jq '.users[0].user."client-certificate-data"' | tr -d '"' | base64 -d > tls.crt
cat config | jq '.users[0].user."client-key-data"' | tr -d '"' | base64 -d > tls.key
```

  进行校验。

```
openssl verify -CAfile ca.crt tls.crt
```

  如果输出"tls.crt: OK"说明CA无误，否则需要重新下载KubeConfig。

  b. 输入如下命令，检查证书公钥和私钥是否匹配：

```
diff -eq <(openssl x509 -pubkey -noout -in tls.crt) <(openssl rsa -pubout -in tls.key)
```

  如果输出"writing RSA key"，说明证书公钥和私钥匹配，否则说明不匹配，需要重新下载KubeConfig，验证完成后请删除临时文件：

```
rm -f ca.crt tls.crt tls.key
```

  c. 检查证书是否过期。

  首先保存证书到临时文件：

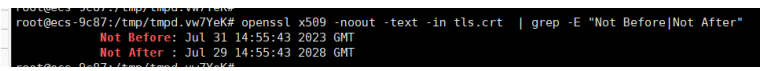
```
cd ~/.kube
cat config | jq '.users[0].user."client-certificate-data"' | tr -d '"' | base64 -d > tls.crt
```

  查看证书有效期。

```
openssl x509 -noout -text -in tls.crt | grep -E "Not Before|Not After"
```

  会输出如下图所示的证书有效区间，请确认当前证书是否在有效期内，如果过期请重新下载KubeConfig，并删除临时文件。

```
rm -f tls.crt
```

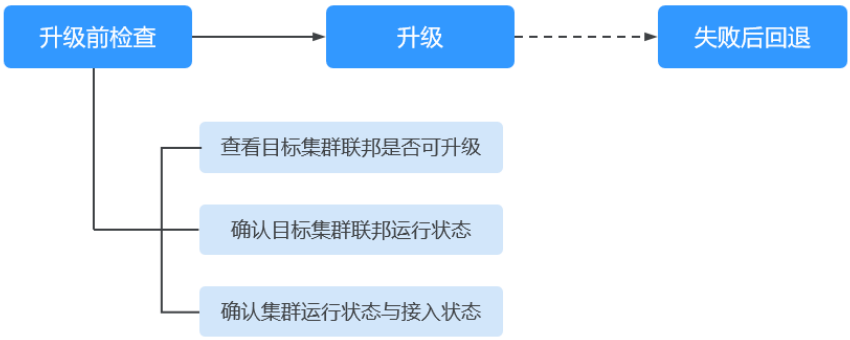


3.4 升级集群联邦

在新的联邦版本发布后，您可以对现有联邦版本进行升级，以便使用新版本支持的相关功能。您可通过**集群联邦升级路径**查看各个版本的特性说明。

联邦升级流程包括升级前检查、升级和失败后回退几个步骤，如**集群联邦升级流程**所示，您可以通过UCS控制台可视化升级联邦版本。

图 3-8 联邦升级流程



1. 升级前检查

   升级集群联邦前，UCS会对联邦运行状态、集群运行状态、集群接入状态三方面进行检查，尽可能避免升级失败。如有检查异常项，请按控制台提示排查并修复。

2. 升级

   执行集群联邦升级。

3. 失败后回退

   升级若失败，可以选择重新升级或回退至原有联邦版本。

集群联邦升级路径

UCS在发布集群联邦的最新版本后，会对该版本所做的变更进行说明。如下表格详细介绍了联邦版本能够升级到的目标版本，以及版本差异。

表 3-3 联邦版本说明

版本	说明
v1.7.0-r14	修复了使用kubectl操作联邦资源的一些bug。

升级集群联邦

UCS支持查看现有联邦版本，并升级集群联邦至更新的版本。

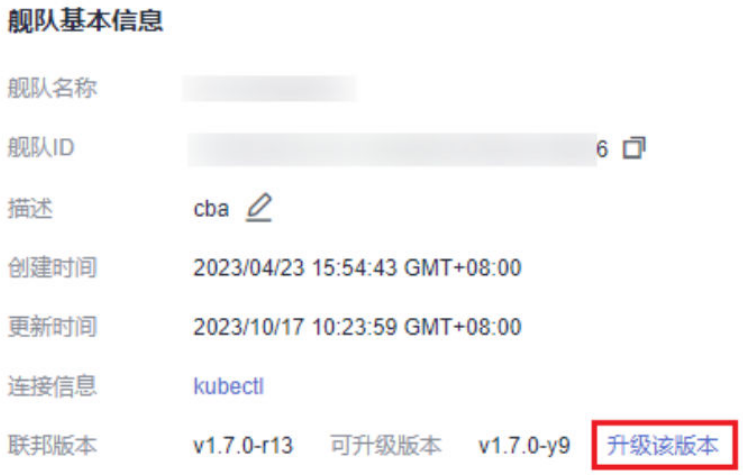
⚠ 注意

集群联邦升级过程中，不允许进行集群移入、移出操作，不允许进行联邦操作，否则可能造成集群联邦升级失败。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到需要升级联邦版本的目标舰队，单击目标舰队名称，在舰队基本信息中单击“升级该版本”。

图 3-9 升级联邦版本



步骤3 在弹出的升级告知中查看能够升级到的版本，并单击“下一步”进行升级前检查。

步骤4 若通过检查，则单击“开始升级”进行升级，升级过程大约需要2分钟。

若未通过检查，请单击“排查原因”，参考文档进行错误修复。

步骤5 右上角状态重新显示为“集群联邦能力已开通”表明升级成功，您可以在舰队基本信息中查看新的版本号。

右上角状态显示为“集群联邦能力升级失败”表明升级失败，请执行**集群联邦版本回退**。

----结束

回退集群联邦

若联邦版本升级失败，UCS支持重新对集群联邦进行升级，或将集群联邦回退至原有版本。

⚠ 注意

- 若未出现集群联邦升级失败情况，则联邦版本不可回退。
- 联邦版本回退过程中，不允许进行集群移入、移出操作，不允许进行联邦操作。
- 若在排查失败原因后进行重试、回滚仍不成功，请提交工单，联系技术支持人员处理。

步骤1 单击“集群联邦能力升级失败”，查看集群联邦升级失败原因。

步骤2 若需要重新尝试联邦版本升级，可单击“重试”，再次进行升级，具体操作参考**联邦版本升级**。

步骤3 若需要回退至原有版本，则单击“回退”，在弹窗中单击“确认”，集群联邦即可回退至现有版本。

----结束

3.5 工作负载

3.5.1 创建工作负载

3.5.1.1 无状态负载

UCS集群联邦可实现多个不同区域、不同云的Kubernetes管理，支持统一的全局应用部署，可将Deployment、StatefulSet、DaemonSet等不同类型的工作负载部署到集群联邦下的集群。

在运行中始终不保存任何数据或状态的工作负载称为“无状态负载 Deployment”，例如nginx。您可以通过控制台或kubectl命令行创建无状态负载。

创建无状态负载

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“工作负载”，在“无状态负载”页签中单击右上角“镜像创建”。

说明

若使用已有的YAML创建工作负载，请单击右上角“YAML创建”。

步骤4 设置工作负载基本信息。

- 负载类型：选择“无状态负载”。
- 负载名称：新增工作负载的名称，命名必须唯一。
- 命名空间：选择工作负载所在命名空间。如需新建命名空间，请参见**创建命名空间**。
- 描述：工作负载的描述信息。
- 实例数量：设置多集群的工作负载中各集群的实例数。用户可以设置具体实例个数，默认为2。每个工作负载实例都由相同的容器部署而成。在UCS中可以通过设置弹性扩缩容策略，根据工作负载资源使用情况，动态调整工作负载实例数。

步骤5 设置工作负载容器配置。

Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器并分别进行设置。

图 3-10 容器配置

- 基本信息：

  表 3-4 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 – 我的镜像：当前区域下华为云镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 – 镜像中心：开源镜像仓库中的官方镜像。 – 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	– 申请：容器需要使用的最小CPU值，默认0.25Core。 – 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	– 申请：容器需要使用的内存最小值，默认512MiB。 – 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。

- 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
- 健康检查：设置健康检查可以在容器运行过程中定时检查容器的健康状况，详情请参见**设置容器健康检查**。

- 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
- 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。
- 安全设置：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。
- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

步骤6 （可选）单击服务配置栏的＋，进行工作负载服务配置。

若工作负载需要和其它服务互访，或需要被公网访问，您需要添加服务（Service），设置访问方式。工作负载访问的方式决定了这个工作负载的网络属性，不同访问方式的工作负载可以提供不同网络能力，操作详情请参见**服务与路由**。

您也可以在创建完工作负载之后再创建Service，参见**集群内访问（ClusterIP）**和**节点访问（NodePort）**。

- Service名称：新增服务名称，用户可自定义，服务名称必须唯一。
- 访问类型：
  - 集群内访问（ClusterIP）：只能集群内访问服务。
  - 节点访问（NodePort）：可以通过集群内任意节点访问到服务。
- 服务亲和（仅节点访问设置）：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- 端口配置：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口（仅节点访问设置）：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。

步骤7 （可选）单击“展开高级配置”，设置工作负载高级配置。

- 升级策略：指定无状态负载的升级方式，包括整体替换升级和逐步滚动升级，详细参数说明请参见**配置工作负载升级策略**。
  - 滚动升级：滚动升级将逐步用新版本的实例替换旧版本的实例，升级的过程中，业务流量会同时负载均衡分布到新的和旧的实例上，因此业务不会中断。
  - 替换升级：先删除旧实例，再创建新实例。升级过程中业务会中断。

- 调度策略：您可设置亲和（affinity）与反亲和（anti-affinity）实现Pod的计划性调度，详细信息请参见**配置调度策略（亲和与反亲和）**。
- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。
- 容忍策略：当工作负载实例所在的节点不可用时，系统将实例重新调度到其它可用节点的时间窗，默认为300秒。

步骤8 单击“下一步：调度与差异化”，对选择的集群进行调度与差异化配置。在选择可调度集群后，可对容器进行“差异化配置”。

- 集群调度策略：
  - 调度方式：
    - 集群权重：手动设置各集群的权重，工作负载在各集群的实例数将根据设置的权重比例进行分配。
    - 自动均衡：工作负载将根据资源余量在可调度的集群中自动选择集群进行部署。
  - 部署集群：选择工作负载可调度的集群，集群个数请您根据自身业务进行确定。
    - “集群权重”模式下，需手动设置各集群权重值，权重非0的集群将自动勾选为可调度集群，权重为0则表示该集群不可调度。状态非正常的集群无法设置权重。
    - “自动均衡”模式下，单击集群即可将其勾选为可调度集群。
- 差异化配置：

  工作负载在不同的集群中部署可进行差异化的配置。在选择可调度集群后单击对应集群右上角  ，即可对每个集群进行差异化配置，差异化后的容器配置只对该集群生效。

  具体参数说明请参见**容器配置**。

步骤9 设置完成后，单击“创建工作负载”，完成创建后，可单击“返回工作负载列表”查看所创建的工作负载。

----结束

3.5.1.2 有状态负载

在运行过程中会保存数据或状态的工作负载称为“有状态工作负载（statefulset）”，创建的Pod拥有持久型标识符，Pod迁移或销毁重启后，标识符仍会保留。有状态负载不支持弹性伸缩，适用于需要使用持久化存储的场景，如ETCD等。

创建有状态负载

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“工作负载”，切换至“有状态负载”页签，并单击右上角“镜像创建”。

说明

若使用已有的YAML创建工作负载，请单击右上角“YAML创建”。

步骤4 设置工作负载基本信息。

- 负载类型：选择“有状态负载”。
- 负载名称：新增工作负载的名称，命名必须唯一。
- 命名空间：选择工作负载所在命名空间。如需新建命名空间，请参见**创建命名空间**。
- 描述：工作负载的描述信息。
- 实例数量：设置多集群的工作负载中各集群的实例数。用户可以设置具体实例个数，默认为2。每个工作负载实例都由相同的容器部署而成。在UCS中可以通过设置弹性扩缩容策略，根据工作负载资源使用情况，动态调整工作负载实例数。

步骤5 设置工作负载容器配置。

Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器并分别进行设置。

图 3-11 容器配置

- 基本信息：

表 3-5 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 – 我的镜像：当前区域下华为云镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 – 镜像中心：开源镜像仓库中的官方镜像。 – 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。

参数	说明
CPU配额	– 申请：容器需要使用的最小CPU值，默认0.25Core。 – 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	– 申请：容器需要使用的内存最小值，默认512MiB。 – 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。

- 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
- 健康检查：设置健康检查可以在容器运行过程中定时检查容器的健康状况，详情请参见**设置容器健康检查**。
- 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
- 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。
- 安全设置：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。
- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

步骤6 设置工作负载实例间发现服务配置。

StatefulSet实例间发现通过无头服务（Headless Service）实现，无头服务并不会分配Cluster IP，并且查询会返回所有Pod的DNS记录，这样就可查询到所有Pod的IP地址。

- Service名称：输入工作负载所对应的服务名称，用于集群内工作负载间的互相访问。该服务主要用于实例的内部发现，不需要有单独的IP地址，也不需要做负载均衡。
- 端口配置：
  - 端口名称：端口名称用于给容器端口命名，通常以端口用途命名。
  - 服务端口：输入服务端口。
  - 容器端口：输入容器的监听端口。

步骤7 （可选）单击服务配置栏的＋，进行工作负载服务配置。

若工作负载需要和其它服务互访，或需要被公网访问，您需要添加服务（Service），设置访问方式。工作负载访问的方式决定了这个工作负载的网络属性，不同访问方式的工作负载可以提供不同网络能力，操作详情请参见**服务与路由**。

您也可以在创建完工作负载之后再创建Service，参见**集群内访问（ClusterIP）**和**节点访问（NodePort）**。

- Service名称：新增服务名称，用户可自定义，服务名称必须唯一。
- 访问类型：
  - 集群内访问（ClusterIP）：只能集群内访问服务。
  - 节点访问（NodePort）：可以通过集群内任意节点访问到服务。
- 服务亲和（仅节点访问设置）：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- 端口配置：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口（仅节点访问设置）：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。

步骤8 （可选）单击“展开高级配置”，设置工作负载高级配置。

- 升级策略：指定有状态负载的升级方式，包括整体替换升级和逐步滚动升级，详细参数说明请参见**配置工作负载升级策略**。
  - 滚动升级：滚动升级将逐步用新版本的实例替换旧版本的实例，升级的过程中，业务流量会同时负载均衡分布到新的和旧的实例上，因此业务不会中断。
  - 替换升级：有状态工作负载的替换升级，需要手动删除旧实例，再创建新实例。升级过程中业务会中断。
- 实例管理策略：
  - 有序策略：默认实例管理策略，有状态负载会逐个的、按顺序的进行部署、删除、伸缩实例, 只有前一个实例部署Ready或者删除完成后，有状态负载才会操作后一个实例。
  - 并行策略：支持有状态负载并行创建或者删除所有的实例，有状态负载发生变更时立刻在实例上生效。
- 调度策略：您可设置亲和（affinity）与反亲和（anti-affinity）实现Pod的计划性调度，详细信息请参见**配置调度策略（亲和与反亲和）**。
- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。

步骤9 单击“下一步”，对选择的集群进行调度与差异化配置。在选择可调度集群后，可对容器进行“差异化配置”。

- 集群调度策略：
  - 调度方式：
    - 复制分发：工作负载将在勾选的所有集群中进行部署。
  - 部署集群：单击集群即可将其勾选为工作负载可调度的集群，集群个数请您根据自身业务进行确定。
- 差异化配置：

  工作负载在不同的集群中部署可进行差异化的配置。在选择可调度集群后单击对应集群右上角  ，即可对每个集群进行差异化配置，差异化后的容器配置只对该集群生效。

  具体参数说明请参见**容器配置**。

步骤10 设置完成后，单击“创建工作负载”完成创建。

----结束

3.5.1.3 守护进程集

守护进程集（DaemonSet）保证集群下全部（或某些）节点上均运行一个Pod，新节点添加到集群内也会自动部署Pod，有节点从集群移除时，该节点上的Pod也会被回收。适用于常驻集群的后台程序，如日志采集等。删除DaemonSet将会删除它创建的所有Pod。

创建守护进程集

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“工作负载”，切换至“守护进程集”页签，并单击右上角“镜像创建”。

说明

若使用已有的YAML创建工作负载，请单击右上角“YAML创建”。

步骤4 设置工作负载基本信息。

- 负载类型：选择“守护进程集”。
- 负载名称：新增工作负载的名称，命名必须唯一。
- 命名空间：选择工作负载所在命名空间。如需新建命名空间，请参见**创建命名空间**。
- 描述：工作负载的描述信息。

步骤5 设置工作负载容器配置。

Pod中可以配置多个容器，您可以单击右侧“添加容器”为Pod配置多个容器并分别进行设置。

图 3-12 容器配置

- 基本信息：

  表 3-6 基本信息参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。 – 我的镜像：当前区域下华为云镜像仓库中的镜像。若无可用的镜像，可单击“上传镜像”进行上传。 – 镜像中心：开源镜像仓库中的官方镜像。 – 共享镜像：由他人账号共享的私有镜像，详情请参见**共享私有镜像**。
镜像版本	选择需要部署的镜像版本。
更新策略	镜像更新/拉取策略。勾选“总是拉取镜像”表示每次都从镜像仓库拉取镜像；如不勾选则优先使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	– 申请：容器需要使用的最小CPU值，默认0.25Core。 – 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	– 申请：容器需要使用的内存最小值，默认512MiB。 – 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 关于CPU/内存配额申请和限制的具体说明请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。

- 生命周期：设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。目前提供的生命周期回调函数有启动命令、启动后处理、停止前处理，详情请参见**设置容器生命周期**。
- 健康检查：设置健康检查可以在容器运行过程中定时检查容器的健康状况，详情请参见**设置容器健康检查**。

- 环境变量：容器运行环境中设定的一个变量，通过环境变量设置的配置项不会随着Pod生命周期结束而变化，详情请参见**设置环境变量**。
- 数据存储：配置容器存储，可以使用本地存储和存储卷声明（PVC）。建议使用PVC将工作负载Pod数据存储在云存储上。若存储在本地磁盘上，节点异常无法恢复时，本地磁盘中的数据也将无法恢复。容器存储相关内容请参见**容器存储**。
- 安全设置：对容器权限进行设置，保护系统和其他容器不受其影响。请输入用户ID，容器将以当前用户权限运行。
- 镜像访问凭证：用于访问镜像仓库的凭证。该凭证仅访问私有镜像仓库时使用，如所选镜像为公开镜像，则无需选择密钥。密钥的创建方法请参见**创建密钥**。

步骤6 （可选）单击服务配置栏的＋，进行工作负载服务配置。

若工作负载需要和其它服务互访，或需要被公网访问，您需要添加服务（Service），设置访问方式。工作负载访问的方式决定了这个工作负载的网络属性，不同访问方式的工作负载可以提供不同网络能力，操作详情请参见**服务与路由**。

您也可以在创建完工作负载之后再创建Service，参见**集群内访问（ClusterIP）**和**节点访问（NodePort）**。

- Service名称：新增服务名称，用户可自定义，服务名称必须唯一。
- 访问类型：
  - 集群内访问（ClusterIP）：只能集群内访问服务。
  - 节点访问（NodePort）：可以通过集群内任意节点访问到服务。
- 服务亲和（仅节点访问设置）：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- 端口配置：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口（仅节点访问设置）：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。

步骤7 （可选）单击“展开高级配置”，设置工作负载高级配置。

- 升级策略：指定守护进程集的升级方式，包括整体替换升级和逐步滚动升级，详细参数说明请参见**配置工作负载升级策略**。
  - 滚动升级：滚动升级将逐步用新版本的实例替换旧版本的实例，升级的过程中，业务流量会同时负载均衡分布到新的和旧的实例上，因此业务不会中断。
  - 替换升级：守护进程集的替换升级，需要手动删除旧实例，再创建新实例。升级过程中业务会中断。

- 调度策略：您可设置亲和（affinity）与反亲和（anti-affinity）实现Pod的计划性调度，详细信息请参见**配置调度策略（亲和与反亲和）**。
- 标签与注解：您可以单击“添加”为Pod增加标签或注解，新增标签或注解的键不能与已有的重复。

步骤8 单击“下一步：调度与差异化”，对选择的集群进行调度与差异化配置。在选择可调度集群后，可对容器进行“差异化配置”。

- 调度策略：
  - 调度方式：
    - 复制分发：工作负载将在勾选的所有集群中进行部署。
  - 部署集群：单击集群即可将其勾选为工作负载可调度的集群，集群个数请您根据自身业务进行确定。
- 差异化配置：

  工作负载在不同的集群中部署可进行差异化的配置。在选择可调度集群后单击对应集群右上角  ，即可对每个集群进行差异化配置，差异化后的容器配置只对该集群生效。

  具体参数说明请参见**容器配置**。

步骤9 设置完成后，单击“创建工作负载”完成创建。

----结束

3.5.2 容器设置

3.5.2.1 容器基本信息

工作负载是Kubernetes对一组Pod的抽象模型，用于描述业务的运行载体，一个Pod可以封装1个或多个容器，您可以单击右上方的“添加容器”，添加多个容器镜像并分别进行设置。

图 3-13 添加容器

表 3-7 镜像参数说明

参数	说明
容器名称	为容器命名。
镜像名称	单击后方“选择镜像”，选择容器使用的镜像。
镜像版本	选择需要部署的镜像版本。

参数	说明
更新策略	镜像更新/拉取策略。可以勾选“总是拉取镜像”，表示每次都从镜像仓库拉取镜像；如不勾选则优使用节点已有的镜像，如果没有这个镜像再从镜像仓库拉取。
CPU配额	• 申请：容器需要使用的最小CPU值，默认0.25Core。 • 限制：允许容器使用的CPU最大值。建议设容器配额的最高限额，避免容器资源超额导致系统故障。
内存配额	• 申请：容器需要使用的内存最小值，默认512MiB。 • 限制：允许容器使用的内存最大值。如果超过，容器会被终止。 申请和限制的具体请参见**设置容器规格**。
初始化容器	选择容器是否作为初始化容器。 Init 容器是一种特殊容器，在 Pod 内的应用容器启动之前运行。详细说明请参见**Init 容器**。

3.5.2.2 设置容器规格

操作场景

在创建工作负载时为添加的容器设置资源限制，可以对工作负载中每个实例所用的CPU配额、内存配额进行申请和限制。

配置说明

- CPU配额：

  表 3-8 CPU 配额说明

参数	说明
CPU申请	容器使用的最小CPU需求，作为容器调度时资源分配的判断依赖。只有当节点上可分配CPU总量 ≥ 容器CPU申请数时，才允许将容器调度到该节点。
CPU限制	容器能使用的CPU最大值。

  建议配置方法：

  节点的实际可用分配CPU量 ≥ 当前实例所有容器CPU限制值之和 ≥ 当前实例所有容器CPU申请值之和，节点的实际可用分配CPU量请在“集群管理”中对应集群的“节点管理”页面下查看。

- 内存配额：

表 3-9 内存配额说明

参数	说明
内存申请	容器使用的最小内存需求，作为容器调度时资源分配的判断依赖。只有当节点上可分配内存总量 ≥ 容器内存申请数时，才允许将容器调度到该节点。
内存限制	容器能使用的内存最大值。当内存使用率超出设置的内存限制值时，该实例可能会被重启进而影响工作负载的正常使用。

建议配置方法：

节点的实际可用分配内存量 ≥ 当前节点所有容器内存限制值之和 ≥ 当前节点所有容器内存申请值之和，节点的实际可用分配内存量请在“集群管理”中对应集群的“节点管理”页面下查看。

说明

可分配资源：可分配量按照实例请求值(request)计算，表示实例在该节点上可请求的资源上限，不代表节点实际可用资源。 计算公式为：

- 可分配CPU = CPU总量 - 所有实例的CPU请求值 - 其他资源CPU预留值
- 可分配内存 = 内存总量 - 所有实例的内存请求值 - 其他资源内存预留值

使用示例

以集群包含一个资源为4Core 8GB的节点为例，已经部署一个包含两个实例的工作负载到该集群上，并设置两个实例（实例1，实例2）的资源为{CPU申请，CPU限制，内存申请，内存限制}={1Core，2Core，2GB，2GB}。

那么节点上CPU和内存的资源使用情况如下：

- 节点CPU可分配量=4Core-（实例1申请的1Core+实例2申请的1Core）=2Core
- 节点内存可分配量=8GB-（实例1申请的2GB+实例2申请的2GB）=4GB

因此节点还剩余2Core 4GB的资源可供下一个新增的实例使用。

3.5.2.3 设置容器生命周期

操作场景

设置生命周期回调函数可在容器的特定阶段执行调用，比如容器在停止前希望执行某项操作，就可以设置相应的函数。

目前提供的生命周期回调函数如下所示：

- **启动命令：**容器将会以该启动命令启动，请参见**启动命令**。
- **启动后处理：**容器启动后触发，请参见**启动后处理**。
- **停止前处理：**容器停止前触发。设置停止前处理，确保升级或实例删除时可提前将实例中运行的业务排水。详细请参见**停止前处理**。

启动命令

在默认情况下，镜像启动时会运行默认命令，如果想运行特定命令或重写镜像默认值，需要进行相应设置。

Docker的镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时将运行镜像制作时提供的默认的命令和参数，Docker将这两个字段定义为ENTRYPOINT和 CMD。

如果在创建工作负载时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令ENTRYPOINT、CMD，规则如下：

表 3-10 容器如何执行命令和参数

镜像 ENTRYPOINT	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

步骤1 登录UCS控制台，进入集群联邦页面，在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“启动命令”页签，输入运行命令和运行参数。

表 3-11 容器启动命令

命令方式	操作步骤
运行命令	输入可执行的命令，例如“/run/server”。 若运行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（""）。 **说明** 多命令时，运行命令建议用/bin/sh或其他的shell，其他全部命令作为参数来传入。
运行参数	输入控制容器运行命令参数，例如--port=8080。 若参数有多个，可添加运行参数。

----结束

启动后处理

步骤1 登录UCS控制台，进入集群联邦页面，在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“启动后处理”页签，设置启动后处理的参数。

表 3-12 启动后处理-参数说明

参数	说明
命令行方式	在容器中执行指定的命令，配置为需要执行的命令。命令的格式为Command Args[1] Args[2]…（Command为系统命令或者用户自定义可执行程序，如果未指定路径则在默认路径下寻找可执行程序），如果需要执行多条命令，建议采用将命令写入脚本执行的方式。 如需要执行的命令如下： exec: command: - /install.sh - install_agent 请在执行脚本中填写: /install install_agent。这条命令表示容器创建成功后将执行install.sh。
HTTP请求方式	发起一个HTTP调用请求。配置参数如下： ● 路径：请求的URL路径，可选项。 ● 端口：请求的端口，必选项。 ● 主机地址：请求的IP地址，可选项，默认为实例IP。

----结束

停止前处理

步骤1 登录UCS控制台，进入集群联邦页面，在创建工作负载时，配置容器信息，选择“生命周期”。

步骤2 在“停止前处理”页签，设置停止前处理的命令。

表 3-13 停止前处理

参数	说明
命令行方式	在容器中执行指定的命令，配置为需要执行的命令。命令的格式为Command Args[1] Args[2]…（Command为系统命令或者用户自定义可执行程序，如果未指定路径则在默认路径下寻找可执行程序），如果需要执行多条命令，建议采用将命令写入脚本执行的方式。 如需要执行的命令如下： exec: command: - /uninstall.sh - uninstall_agent 请在执行脚本中填写: /uninstall uninstall_agent。这条命令表示容器结束前将执行uninstall.sh。

参数	说明
HTTP请求方式	发起一个HTTP调用请求。配置参数如下： ● 路径：请求的URL路径，可选项。 ● 端口：请求的端口，必选项。 ● 主机地址：请求的IP地址，可选项，默认为实例IP。

----结束

YAML 样例

本节以nginx为例，说明kubectl命令设置容器生命周期的方法。

在以下配置文件中，您可以看到postStart命令在容器目录/bin/bash下写了个**install.sh**命令。preStop执行**uninstall.sh**命令。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        command:
        - sleep 3600                    #启动命令
        imagePullPolicy: Always
        lifecycle:
          postStart:
            exec:
              command:
              - /bin/bash
              - install.sh              #启动后命令
          preStop:
            exec:
              command:
              - /bin/bash
              - uninstall.sh            #停止前命令
        name: nginx
      imagePullSecrets:
      - name: default-secret
```

3.5.2.4 设置容器健康检查

操作场景

健康检查是指容器运行过程中，根据用户需要，定时检查容器健康状况。若不配置健康检查，如果容器内应用程序异常，Pod将无法感知，也不会自动重启去恢复。最终导致虽然Pod状态显示正常，但Pod中的应用程序异常的情况。

Kubernetes提供了两种健康检查的探针：

- **存活探针：**livenessProbe，用于检测容器是否正常，类似于执行ps命令检查进程是否存在。如果容器的存活检查失败，集群会对该容器执行重启操作；若容器的存活检查成功则不执行任何操作。
- **就绪探针：**readinessProbe，用于检查用户业务是否就绪，如果未就绪，则不转发流量到当前实例。一些程序的启动时间可能很长，比如要加载磁盘数据或者要依赖外部的某个模块启动完成才能提供服务。这时候程序进程在，但是并不能对外提供服务。这种场景下该检查方式就非常有用。如果容器的就绪检查失败，集群会屏蔽请求访问该容器；若检查成功，则会开放对该容器的访问。

检查方式

- **HTTP 请求检查**

  HTTP 请求方式针对的是提供HTTP/HTTPS服务的容器，集群周期性地对该容器发起HTTP/HTTPS GET请求，如果HTTP/HTTPS response返回码属于200~399范围，则证明探测成功，否则探测失败。使用HTTP请求探测必须指定容器监听的端口和HTTP/HTTPS的请求路径。

  例如：提供HTTP服务的容器，HTTP检查路径为：/health-check；端口为：80；主机地址可不填，默认为容器实例IP，此处以172.16.0.186为例。那么集群会周期性地对容器发起如下请求：GET http://172.16.0.186:80/health-check。

  图 3-14 HTTP 请求检查

- **TCP 端口检查**

  对于提供TCP通信服务的容器，集群周期性地对该容器建立TCP连接，如果连接成功，则证明探测成功，否则探测失败。选择TCP端口探测方式，必须指定容器监听的端口。

  例如：有一个nginx容器，它的服务端口是80，对该容器配置了TCP端口探测，指定探测端口为80，那么集群会周期性地对该容器的80端口发起TCP连接，如果连接成功则证明检查成功，否则检查失败。

  图 3-15 TCP 端口检查

- **执行命令检查**

  命令检查是一种强大的检查方式，该方式要求用户指定一个容器内的可执行命令，集群会周期性地在容器内执行该命令，如果命令的返回结果是0则检查成功，否则检查失败。

对于上面提到的TCP端口检查和HTTP请求检查，都可以通过执行命令检查的方式来替代：

- 对于TCP端口探测，可以使用程序来对容器的端口进行connect，如果connect成功，脚本返回0，否则返回-1。
- 对于HTTP请求探测，可以使用脚本来对容器进行wget。

  wget http://127.0.0.1:80/health-check

  并检查response 的返回码，如果返回码在200~399 的范围，脚本返回0，否则返回-1。如下图：

图 3-16 执行命令检查

须知

- 必须把要执行的程序放在容器的镜像里面，否则会因找不到程序而执行失败。
- 如果执行的命令是一个shell脚本，由于集群在执行容器里的程序时，不在终端环境下，因此不能直接指定脚本为执行命令，需要加上脚本解析器。比如脚本是**/data/scripts/health_check.sh**，那么使用执行命令检查时，指定的程序应该是**sh /data/scripts/health_check.sh**。究其原因是集群在执行容器里的程序时，不在终端环境下。

公共参数说明

表 3-14 公共参数说明

参数	参数说明
检测周期（periodSeconds）	探针检测周期，单位为秒。 例如，设置为30，表示每30秒检测一次。
延迟时间（initialDelaySeconds）	延迟检查时间，单位为秒，此设置与业务程序正常启动时间相关。 例如，设置为30，表明容器启动后30秒才开始健康检查，该时间是预留给业务程序启动的时间。
超时时间（timeoutSeconds）	超时时间，单位为秒。 例如，设置为10，表明执行健康检查的超时等待时间为10秒，如果超过这个时间，本次健康检查就被视为失败。若设置为0或不设置，默认超时等待时间为1秒。

参数	参数说明
成功阈值（successThreshold）	探测失败后，被视为成功的最小连续成功数。 默认值是 1，最小值是 1。 存活探测的这个值必须是 1。
最大失败次数（failureThreshold）	当探测失败时重试的次数。 存活探测情况下的放弃就意味着重新启动容器。就绪探测情况下的放弃 Pod 会被打上未就绪的标签。 默认值是 3。最小值是 1。

YAML 示例

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: nginx:alpine
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
    readinessProbe:
      exec:
        command:
          - cat
          - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
```

3.5.2.5 设置环境变量

操作场景

环境变量是指容器运行环境中设定的一个变量，环境变量可以在工作负载部署后修改，为工作负载提供极大的灵活性。

UCS中设置的环境变量与Dockerfile中的“ENV”效果相同。

须知

容器启动后，容器中的内容不应修改。如果修改配置项（例如将容器应用的密码、证书、环境变量配置到容器中），当容器重启（例如节点异常重新调度Pod）后，会导致配置丢失，业务异常。

配置信息应通过入参等方式导入容器中，以免重启后配置丢失。

环境变量支持如下几种方式设置。

- **自定义**
- **配置项导入**：将配置项中所有键值都导入为环境变量。
- **配置项键值导入**：将配置项中某个键的值导入作为某个环境变量的值。例如将configmap-example这个配置项中configmap_key的值configmap_value导入为环境变量key1的值，导入后容器中有一个名为key1的环境变量，其值为configmap_value。
- **密钥导入**：将密钥中所有键值都导入为环境变量。
- **密钥键值导入**：将密钥中某个键的值导入作为某个环境变量的值。例如将secret-example这个配置项中secret_key的值secret_value导入为环境变量key2的值，导入后容器中有一个名为key2的环境变量，其值为secret_value。
- **变量/变量引用**：用Pod定义的字段作为环境变量的值，例如Pod的名称。
- **资源引用**：用Container定义的字段作为环境变量的值，例如容器的CPU限制。

添加环境变量

步骤1 登录UCS控制台，进入集群联邦页面，在创建工作负载时，配置容器信息，选择“环境变量”。

步骤2 设置环境变量。

图 3-17 设置环境变量

----结束

YAML 样例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: env-example
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: env-example
```

```
template:
  metadata:
    labels:
      app: env-example
  spec:
    containers:
      - name: container-1
        image: nginx:alpine
        imagePullPolicy: Always
        resources:
          requests:
            cpu: 250m
            memory: 512Mi
          limits:
            cpu: 250m
            memory: 512Mi
        env:
          - name: key                    # 自定义
            value: value
          - name: key1                   # 配置项键值导入
            valueFrom:
              configMapKeyRef:
                name: configmap-example
                key: key1
          - name: key2                   # 密钥键值导入
            valueFrom:
              secretKeyRef:
                name: secret-example
                key: key2
          - name: key3                   # 变量引用，用Pod定义的字段作为环境变量的值
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
          - name: key4                   # 资源引用，用Container定义的字段作为环境变量的值
            valueFrom:
              resourceFieldRef:
                containerName: container1
                resource: limits.cpu
                divisor: 1
        envFrom:
          - configMapRef:                # 配置项导入
              name: configmap-example
          - secretRef:                   # 密钥导入
              name: secret-example
    imagePullSecrets:
      - name: default-secret
```

环境变量查看

如果configmap-example和secret-example的内容如下。

```
$ kubectl get configmap configmap-example -oyaml
apiVersion: v1
data:
  configmap_key: configmap_value
kind: ConfigMap
...

$ kubectl get secret secret-example -oyaml
apiVersion: v1
data:
  secret_key: c2VjcmV0X3ZhbHVl              # c2VjcmV0X3ZhbHVl为secret_value的base64编码
kind: Secret
...
```

则进入Pod中查看的环境变量结果如下。


```
$ kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
env-example-695b759569-lx9jp   1/1     Running   0          17m

$ kubectl exec env-example-695b759569-lx9jp  -- printenv
/ # env
key=value                           # 自定义环境变量
key1=configmap_value                  # 配置项键值导入
key2=secret_value                   # 密钥键值导入
key3=env-example-695b759569-lx9jp     # Pod的metadata.name
key4=1                              # container1这个容器的limits.cpu，单位为Core，向上取整
configmap_key=configmap_value         # 配置项导入，原配置项中的键值直接会导入结果
secret_key=secret_value               # 密钥导入，原密钥中的键值直接会导入结果
```

3.5.2.6 配置工作负载升级策略

在实际应用中，升级是一个常见的场景，Deployment、StatefulSet和DaemonSet都能够很方便的支撑应用升级。

设置不同的升级策略，有如下两种。

- RollingUpdate：滚动升级，即逐步创建新Pod再删除旧Pod，为默认策略。
- Recreate：替换升级，即先把当前Pod删掉再重新创建Pod。

图 3-18 工作负载升级策略

升级参数说明

- 最大浪涌（maxSurge）

  与spec.replicas相比，可以有多少个Pod存在，默认值是25%，比如spec.replicas为 4，那升级过程中就不能超过5个Pod存在，即按1个的步伐升级，实际升级过程中会换算成数字，且换算会向上取整。这个值也可以直接设置成数字。

  仅Deployment在“滚动升级”方式下支持配置。

- 最大无效实例数（maxUnavailable）

  与spec.replicas相比，可以有多少个Pod失效，也就是删除的比例，默认值是25%，比如spec.replicas为4，那升级过程中就至少有3个Pod存在，即删除Pod的步伐是1。同样这个值也可以设置成数字。

  仅Deployment、DaemonSet在“滚动升级”方式下支持配置。

- 实例可用最短时间（minReadySeconds）

  指定新创建的 Pod 在没有任意容器崩溃情况下的最小就绪时间， 只有超出这个时间 Pod 才被视为可用。默认值为 0（Pod 在准备就绪后立即将被视为可用）。

  仅Deployment、DaemonSet支持配置。

- 最大保留版本数（revisionHistoryLimit）

  用来设定出于回滚目的所要保留的旧 ReplicaSet 数量。 这些旧 ReplicaSet 会消耗 etcd 中的资源，并占用 kubectl get rs 的输出。 每个工作负载修订版本的配置都存储在其 ReplicaSets 中；因此，一旦删除了旧的 ReplicaSet， 将失去回滚到

工作负载的对应修订版本的能力。 默认情况下，系统保留 10 个旧 ReplicaSet，但其理想值取决于新工作负载的频率和稳定性。

- 升级最大时长（progressDeadlineSeconds）

  指定系统在报告 Deployment 进展失败 之前等待 Deployment 取得进展的秒数。这类报告会在资源状态中体现为 Type=Progressing、Status=False、Reason=ProgressDeadlineExceeded。Deployment 控制器将持续重试 Deployment。 将来，一旦实现了自动回滚，Deployment 控制器将在探测到这样的条件时立即回滚 Deployment。

  如果指定，则此字段值需要大于 .spec.minReadySeconds 取值。

  仅Deployment支持配置。

- 缩容时间窗（terminationGracePeriodSeconds）：

  优雅删除时间，默认为30秒，删除Pod时发送SIGTERM终止信号，然后等待容器中的应用程序终止执行，如果在terminationGracePeriodSeconds时间内未能终止，则发送SIGKILL的系统信号强行终止。

升级示例

Deployment的升级可以是声明式的，也就是说只需要修改Deployment的YAML定义即可，比如使用kubectl edit命令将上面Deployment中的镜像修改为nginx:alpine。修改完成后再查询ReplicaSet和Pod，发现创建了一个新的ReplicaSet，Pod也重新创建了。

```
$ kubectl edit deploy nginx

$ kubectl get rs
NAME            DESIRED   CURRENT   READY     AGE
nginx-6f9f58dffd   2         2         2         1m
nginx-7f98958cdf   0         0         0         48m

$ kubectl get pods
NAME                   READY     STATUS    RESTARTS   AGE
nginx-6f9f58dffd-tdmqk   1/1       Running   0          1m
nginx-6f9f58dffd-tesqr   1/1       Running   0          1m
```

Deployment可以通过maxSurge和maxUnavailable两个参数控制升级过程中同时重新创建Pod的比例，这在很多时候是非常有用，配置如下所示。

```
spec:
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
```

在前面的例子中，由于spec.replicas是2，如果maxSurge和maxUnavailable都为默认值25%，那实际升级过程中，maxSurge允许最多3个Pod存在（向上取整，2*1.25=2.5，取整为3），而maxUnavailable则不允许有Pod Unavailable（向上取整，2*0.75=1.5，取整为2），也就是说在升级过程中，一直会有2个Pod处于运行状态，每次新建一个Pod，等这个Pod创建成功后再删掉一个旧Pod，直至Pod全部为新Pod。

回滚

回滚也称为回退，即当发现升级出现问题时，让应用回到原版本。Deployment可以非常方便的回滚到原版本。

例如上面升级的新版镜像有问题，可以执行kubectl rollout undo命令进行回滚。


```
$ kubectl rollout undo deployment nginx
deployment.apps/nginx rolled back
```

Deployment之所以能如此容易的做到回滚，是因为Deployment是通过ReplicaSet控制Pod的，升级后之前ReplicaSet都一直存在，Deployment回滚做的就是使用之前的ReplicaSet再次把Pod创建出来。Deployment中保存ReplicaSet的数量可以使用revisionHistoryLimit参数限制，默认值为10。

3.5.2.7 配置调度策略（亲和与反亲和）

Kubernetes支持节点和Pod两个层级的亲和（affinity）与反亲和（anti-affinity）调度。通过配置亲和与反亲和规则，可以允许您指定硬性限制或者偏好，例如将前台Pod和后台Pod部署在一起、某类应用部署到某些特定的节点、不同应用部署到不同的节点等等。

通过控制台配置调度策略

步骤1 登录UCS控制台，进入集群联邦页面。

步骤2 在创建工作负载时，在“高级配置”中找到“调度策略”。

表 3-15 节点亲和性设置

参数名	参数描述
必须满足	即硬约束，设置必须要满足的条件，对应于requiredDuringSchedulingIgnoredDuringExecution，多条规则间是一种“或”的关系，即只需要满足一条规则即会进行调度。
尽量满足	即软约束，设置尽量满足的条件，对应于preferredDuringSchedulingIgnoredDuringExecution，无论是满足其中一条或者是都不满足都会进行调度。

步骤3 在“节点亲和性”、“工作负载亲和性”、“工作负载反亲和性”下单击 添加调度策略。

表 3-16 调度策略设置

参数名	参数描述
标签名	对应节点的标签，可以使用默认的标签也可以用户自定义标签。

参数名	参数描述
操作符	可以设置六种匹配关系（In, NotIn, Exists, DoesNotExist. Gt, and Lt）。 ● In：是否在标签值的列表中 ● NotIn：是否不在标签值的列表中 ● Exists：某个标签存在 ● DoesNotExist：某个标签不存在 ● Gt：标签的值大于某个值（字符串比较） ● Lt：标签的值小于某个值（字符串比较）
标签值	请填写标签值。
命名空间	仅支持在工作负载亲和/工作负载反亲和调度策略中使用。 指定调度策略生效的命名空间。
拓扑域	仅支持在工作负载亲和/工作负载反亲和调度策略中使用。 先圈定拓扑域（topologyKey）指定的范围，然后再选择策略定义的内容。
权重	仅支持在“尽量满足”策略中添加。

----结束

节点亲和（nodeAffinity）

Pod模板中可以通过nodeSelector设置让Pod创建在拥有指定标签的节点上。如下所示，Pod只会部署在拥有gpu=true这个标签的节点上。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:              # 节点选择，当节点拥有gpu=true标签时才在节点上创建Pod
    gpu: true
...
```

通过节点亲和性规则配置，也可以做到同样的事情，如下所示。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 3
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
```

```
    name:  gpu
    resources:
      requests:
        cpu: 100m
        memory: 200Mi
      limits:
        cpu: 100m
        memory: 200Mi
  imagePullSecrets:
  - name: default-secret
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: gpu
            operator: In
            values:
            - "true"
```

看起来这要复杂很多，但这种方式可以得到更强的表达能力，后面会进一步介绍。

这里affinity表示亲和，nodeAffinity表示节点亲和，requiredDuringSchedulingIgnoredDuringExecution非常长，不过可以将这个分作两段来看：

- 前半段requiredDuringScheduling表示下面定义的规则必须强制满足（require）才会调度Pod到节点上。
- 后半段IgnoredDuringExecution表示已经在节点上运行的Pod不需要满足下面定义的规则，即去除节点上的某个标签，那些需要节点包含该标签的Pod不会被重新调度。

另外操作符operator的值为In，表示标签值需要在values的列表中，其他operator取值如下。

- NotIn：标签的值不在某个列表中
- Exists：某个标签存在
- DoesNotExist：某个标签不存在
- Gt：标签的值大于某个值（字符串比较）
- Lt：标签的值小于某个值（字符串比较）

需要说明的是并没有nodeAntiAffinity（节点反亲和），因为NotIn和DoesNotExist可以提供相同的功能。

下面来验证这段规则是否生效，假设某集群有如下三个节点。

```
$ kubectl get node
NAME          STATUS   ROLES    AGE   VERSION
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```

首先给192.168.0.212这个节点打上gpu=true的标签。

```
$ kubectl label node 192.168.0.212 gpu=true
node/192.168.0.212 labeled

$ kubectl get node -L gpu
NAME          STATUS   ROLES    AGE   VERSION                       GPU
192.168.0.212   Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.94    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.97    Ready    <none>   13m   v1.15.6-r1-20.3.0.2.B001-15.30.2
```


创建这个Deployment，可以发现所有的Pod都部署在了192.168.0.212这个节点上。

```
$ kubectl create -f affinity.yaml
deployment.apps/gpu created

$ kubectl get pod -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP            NODE
gpu-6df65c44cf-42xw4   1/1     Running   0          15s   172.16.0.37   192.168.0.212
gpu-6df65c44cf-jzjvs   1/1     Running   0          15s   172.16.0.36   192.168.0.212
gpu-6df65c44cf-zv5cl   1/1     Running   0          15s   172.16.0.38   192.168.0.212
```

节点优先选择规则

上面讲的requiredDuringSchedulingIgnoredDuringExecution是一种**强制**选择的规则，节点亲和还有一种优先选择规则，即preferredDuringSchedulingIgnoredDuringExecution，表示会根据规则**优先**选择哪些节点。

为演示这个效果，先为上面的集群添加一个SSD磁盘的节点，并打上DISK=SSD的标签，为另外三个节点打上DISK=SAS的标签。

```
$ kubectl get node -L DISK,gpu
NAME            STATUS   ROLES    AGE     VERSION                        DISK     GPU
192.168.0.100   Ready    <none>   7h23m   v1.15.6-r1-20.3.0.2.B001-15.30.2   SSD
192.168.0.212   Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SAS      true
192.168.0.94    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SAS
192.168.0.97    Ready    <none>   8h      v1.15.6-r1-20.3.0.2.B001-15.30.2   SAS
```

下面定义一个Deployment，要求Pod优先部署在SSD磁盘的节点上，可以像下面这样定义，使用preferredDuringSchedulingIgnoredDuringExecution规则，给SSD设置权重（weight）为80，而gpu=true权重为20，这样Pod就优先部署在SSD的节点上。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  gpu
  labels:
    app:  gpu
spec:
  selector:
    matchLabels:
      app: gpu
  replicas: 10
  template:
    metadata:
      labels:
        app:  gpu
    spec:
      containers:
      - image:  nginx:alpine
        name:  gpu
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            preference:
              matchExpressions:
              - key: DISK
```

```
            operator: In
            values:
            - SSD
        - weight: 20
          preference:
            matchExpressions:
            - key: gpu
              operator: In
              values:
              - "true"
```

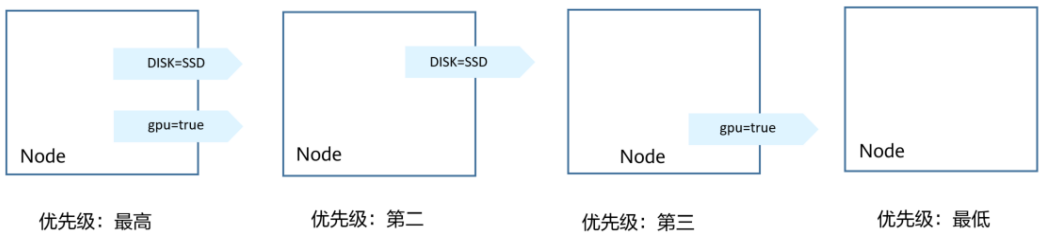
来看实际部署后的情况，可以看到部署到192.168.0.212这个节点上的Pod有5个，而192.168.0.100上只有2个。

```
$ kubectl create -f affinity2.yaml
deployment.apps/gpu created

$ kubectl get po -o wide
NAME                   READY   STATUS    RESTARTS   AGE     IP            NODE
gpu-585455d466-5bmcz   1/1     Running   0          2m29s   172.16.0.44   192.168.0.212
gpu-585455d466-cg2l6   1/1     Running   0          2m29s   172.16.0.63   192.168.0.97
gpu-585455d466-f2bt2   1/1     Running   0          2m29s   172.16.0.79   192.168.0.100
gpu-585455d466-hdb5n   1/1     Running   0          2m29s   172.16.0.42   192.168.0.212
gpu-585455d466-hkgvz   1/1     Running   0          2m29s   172.16.0.43   192.168.0.212
gpu-585455d466-mngvn   1/1     Running   0          2m29s   172.16.0.48   192.168.0.97
gpu-585455d466-s26qs   1/1     Running   0          2m29s   172.16.0.62   192.168.0.97
gpu-585455d466-sxtzm   1/1     Running   0          2m29s   172.16.0.45   192.168.0.212
gpu-585455d466-t56cm   1/1     Running   0          2m29s   172.16.0.64   192.168.0.100
gpu-585455d466-t5w5x   1/1     Running   0          2m29s   172.16.0.41   192.168.0.212
```

上面这个例子中，对于节点排序优先级如下所示，有个两个标签的节点排序最高，只有SSD标签的节点排序第二（权重为80），只有gpu=true的节点排序第三，没有的节点排序最低。

图 3-19 优先级排序顺序



这里您看到Pod并没有调度到192.168.0.94这个节点上，这是因为这个节点上部署了很多其他Pod，资源使用较多，所以并没有往这个节点上调度，这也侧面说明preferredDuringSchedulingIgnoredDuringExecution是优先规则，而不是强制规则。

工作负载亲和（podAffinity）

节点亲和的规则只能影响Pod和节点之间的亲和，Kubernetes还支持Pod和Pod之间的亲和，例如将应用的前端和后端部署在一起，从而减少访问延迟。Pod亲和同样有requiredDuringSchedulingIgnoredDuringExecution和preferredDuringSchedulingIgnoredDuringExecution两种规则。

来看下面这个例子，假设有个应用的后端已经创建，且带有app=backend的标签。

```
$ kubectl get po -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-dlrz8   1/1     Running   0          2m36s   172.16.0.67   192.168.0.100
```

将前端frontend的pod部署在backend一起时，可以做如下Pod亲和规则配置。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
            cpu:  100m
            memory:  200Mi
      imagePullSecrets:
      - name: default-secret
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend
```

创建frontend然后查看，可以看到frontend都创建到跟backend一样的节点上了。

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67   192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70   192.168.0.100
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71   192.168.0.100
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72   192.168.0.100
```

这里有个**topologyKey**字段（拓扑域），意思是先圈定topologyKey指定的范围，然后再选择下面规则定义的内容。这里每个节点上都有kubernetes.io/hostname，所以看不出topologyKey起到的作用。

如果backend有两个Pod，分别在不同的节点上。

```
$ kubectl get po -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP            NODE
backend-658f6cb858-5bpd6   1/1     Running   0          23m     172.16.0.40   192.168.0.97
backend-658f6cb858-dlrz8   1/1     Running   0          2m36s   172.16.0.67   192.168.0.100
```

给192.168.0.97和192.168.0.94打一个prefer=true的标签。

```
$ kubectl label node 192.168.0.97 prefer=true
node/192.168.0.97 labeled
$ kubectl label node 192.168.0.94 prefer=true
node/192.168.0.94 labeled
```



```
$ kubectl get node -L prefer
NAME           STATUS   ROLES    AGE   VERSION                          PREFER
192.168.0.100   Ready    <none>   44m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.212   Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2
192.168.0.94    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
192.168.0.97    Ready    <none>   91m   v1.15.6-r1-20.3.0.2.B001-15.30.2   true
```

将podAffinity的topologyKey定义为prefer。

```
    affinity:
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: prefer
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - backend
```

调度时，先圈定拥有prefer标签的节点，这里也就是192.168.0.97和192.168.0.94，然后再匹配app=backend标签的Pod，从而frontend就会全部部署在192.168.0.97上。

```
$ kubectl create -f affinity3.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE     IP             NODE
backend-658f6cb858-5bpd6    1/1     Running   0          26m     172.16.0.40    192.168.0.97
backend-658f6cb858-dlrz8    1/1     Running   0          5m38s   172.16.0.67    192.168.0.100
frontend-67ff9b7b97-dsqzn   1/1     Running   0          6s      172.16.0.70    192.168.0.97
frontend-67ff9b7b97-hxm5t   1/1     Running   0          6s      172.16.0.71    192.168.0.97
frontend-67ff9b7b97-z8pdb   1/1     Running   0          6s      172.16.0.72    192.168.0.97
```

工作负载反亲和（podAntiAffinity）

前面讲了Pod的亲和，通过亲和将Pod部署在一起，有时候需求却恰恰相反，需要将Pod分开部署，例如Pod之间部署在一起会影响性能的情况。

下面例子中定义了反亲和规则，这个规则表示Pod不能调度到拥有app=frontend标签Pod的节点上，也就是下面将frontend分别调度到不同的节点上（每个节点只有一个Pod）。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:   frontend
  labels:
    app:  frontend
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 5
  template:
    metadata:
      labels:
        app:  frontend
    spec:
      containers:
      - image:  nginx:alpine
        name:  frontend
        resources:
          requests:
            cpu:  100m
            memory:  200Mi
          limits:
```



```
        cpu:  100m
        memory:  200Mi
    imagePullSecrets:
    - name: default-secret
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: kubernetes.io/hostname
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - frontend
```

创建并查看，可以看到每个节点上只有一个frontend的Pod，还有一个在Pending，因为在部署第5个时4个节点上都有了app=frontend的Pod，所以第5个一直是Pending。

```
$ kubectl create -f affinity4.yaml
deployment.apps/frontend created

$ kubectl get po -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP            NODE
frontend-6f686d8d87-8dlsc   1/1     Running   0          18s   172.16.0.76   192.168.0.100
frontend-6f686d8d87-d6l8p   0/1     Pending   0          18s   <none>        <none>
frontend-6f686d8d87-hgcq2   1/1     Running   0          18s   172.16.0.54   192.168.0.97
frontend-6f686d8d87-q7cfq   1/1     Running   0          18s   172.16.0.47   192.168.0.212
frontend-6f686d8d87-xl8hx   1/1     Running   0          18s   172.16.0.23   192.168.0.94
```

3.5.2.8 配置调度与差异化

集群调度策略

当前界面支持集群权重和自动均衡两种策略。

通过控制台配置调度策略

步骤1 登录UCS控制台。

步骤2 在创建工作负载时，单击“下一步：调度与差异化”。

步骤3 添加调度策略。

表 3-17 调度策略

策略	描述
集群权重策略	需要您选择集群并配置分发权重，按照集群权重配比分发Pod。
自动均衡策略	按照剩余资源量自动选择集群分发Pod，不需要额外配置。

----结束

集群容忍策略

集群容忍策略允许调度器将Pod调度至带有对应污点的集群上，需要与集群污点配合使用。

默认的集群容忍策略

在创建工作负载时，UCS会为您的负载配置默认的容忍策略。默认的容忍策略会在某集群出现故障后为其添加表3-18所示的污点，待持续时长超出容忍时长后会自动驱逐该集群上所有Pod。

⚠ 注意

在驱逐所有故障集群上的Pod后，待该集群恢复正常，UCS不会按照原有调度策略将Pod迁回该集群。若您需要继续执行原调度策略配置，可以对工作负载进行**重新调度**。

表 3-18 集群异常状态污点

污点键	容忍策略
cluster.karmada.io/not-ready	集群状态为not ready时自动添加该污点，持续时长超出容忍时长后会自动驱逐该集群上所有Pod。
cluster.karmada.io/unreachable	集群状态为不可用时自动添加该污点，持续时长超出容忍时长后会自动驱逐该集群上所有Pod。

通过控制台配置集群容忍策略

步骤1 登录UCS控制台。

步骤2 在创建工作负载时，单击“下一步：调度与差异化”。

步骤3 添加污点容忍策略。

参数名	参数描述
污点键	集群的污点键。
操作符	• Equal：设置此操作符表示准确匹配指定污点键（必填）和污点值的节点。如果不填写污点值，则表示可以与所有污点键相同的污点匹配。 • Exists：设置此操作符表示匹配存在指定污点键的节点，此时容忍度不能指定污点值。若不填写污点键则可以容忍全部污点。
污点值	• 如果操作符的值是 Exists，则value属性可省略。 • 如果操作符的值是 Equal，则表示其key与value之间的关系是Equal（等于）。 • 如果不指定操作符属性，则默认值为Equal。
污点策略	• 全部：表示匹配所有污点效果。 • NoSchedule：表示匹配污点效果为NoSchedule的污点。 • NoExecute：表示匹配污点效果为NoExecute的污点。

参数名	参数描述
容忍时间窗	即tolerationSeconds参数，当污点策略为NoExecute时支持配置。 在容忍时间窗内，Pod还会在拥有污点的节点上运行，超出时间后会被驱逐。若不配置容忍时间窗，则Pod永远不会被驱逐。

----结束

3.5.3 管理工作负载

操作场景

工作负载创建后，您可以对其执行查看详情升级、编辑YAML、重新部署、重新调度、删除等操作。

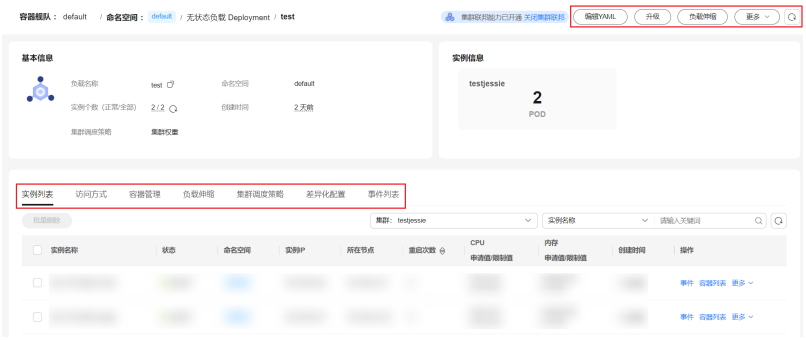
表 3-19 工作负载管理

操作	描述
查看详情	可查看Pod和工作负载的基本信息、事件和状态等，并对工作负载的配置进行修改。
编辑YAML	可通过在线YAML编辑窗对工作负载的YAML文件进行修改和下载。普通任务的YAML文件仅支持查看、复制和下载。
升级	可以通过更换镜像或镜像版本实现工作负载的快速升级，业务无中断。
重新部署	工作负载可以进行重新部署操作，重新部署后将重启负载下的全部容器组Pod，仅无状态工作负载可用。
重新调度	工作负载可以进行重新调度，重新调度后将按照已有的调度策略进行调度，仅无状态工作负载可用。
删除	若工作负载无需再使用，您可以将工作负载或任务删除。工作负载或任务删除后，将无法恢复，请谨慎操作。

查看详情

单击已创建工作负载的名称，可以进入工作负载详情页。在该页面中可以查看Pod和工作负载的基本信息、事件和状态等，并对工作负载的配置进行修改。

图 3-20 工作负载详情页



编辑 YAML

可通过在线YAML编辑窗对无状态工作负载、有状态工作负载、守护进程集、定时任务和容器组的YAML文件进行修改和下载。普通任务的YAML文件仅支持查看、复制和下载。本文以无状态工作负载为例说明如何在线编辑YAML。

步骤1 登录UCS控制台，进入一个已有的容器舰队，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“编辑YAML”，在弹出的“编辑YAML”窗中可对当前工作负载的YAML文件进行修改。

步骤3 单击“确定”，完成修改。

步骤4 （可选）在“编辑YAML”窗中，单击“下载”，可下载该YAML文件。

----结束

升级

您可以通过UCS控制台实现工作负载的快速升级。本文以无状态工作负载为例说明如何进行升级。

步骤1 登录UCS控制台，进入一个已有的容器舰队，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“升级”。

步骤3 请根据业务需求进行工作负载的升级，参数设置方法与创建工作负载时一致。

步骤4 更新完成后，单击“升级工作负载”，并手动确认YAML文件差异后提交升级。

----结束

重新部署（仅无状态工作负载可用）

重新部署将重启负载下的全部容器组Pod。本文以无状态工作负载为例说明如何重新部署工作负载。

步骤1 登录UCS控制台，进入一个已有的容器舰队，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“更多 > 重新部署”。

步骤3 在弹出的提示框中单击“是”，即可完成工作负载的重新部署。

----结束

重新调度（仅无状态工作负载可用）

重新调度将按照配置的调度策略将工作负载调度至集群中。本文以无状态工作负载为例说明如何重新调度工作负载。

步骤1 登录UCS控制台，进入一个已有的容器舰队，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“更多 > 重新调度”。

步骤3 在弹出的提示框中单击“是”，即可完成工作负载的重新调度。

----结束

说明

由于集群故障所导致的工作负载容灾调度，待集群恢复后，可通过重新调度将工作负载按照原有调度策略重新分发。

删除

若工作负载无需再使用，您可以将工作负载或任务删除。工作负载或任务删除后，将无法恢复，请谨慎操作。本文以无状态工作负载为例说明如何使用删除功能。

步骤1 登录UCS控制台，进入一个已有的容器舰队，在左侧导航栏中选择“工作负载”。

步骤2 选择“无状态负载”页签，单击工作负载后的“更多 > 删除”。

步骤3 在弹出的提示框中单击“是”，即可删除工作负载。

----结束

3.6 配置项与密钥

3.6.1 配置项（ConfigMap）

配置项（ConfigMap）允许您将配置文件从容器镜像中解耦，从而增强容器工作负载的可移植性。

配置项价值如下：

- 使用配置项功能可以帮您管理不同环境、不同业务的配置。
- 方便您部署相同工作负载的不同环境，配置文件支持多版本，方便您进行更新和回滚工作负载。
- 方便您快速将您的配置以文件的形式导入到容器中。

说明

- 通过UCS控制台创建配置项后，配置项将默认处于“未部署”状态，需要在创建或更新工作负载时进行挂载，请参见**配置项挂载**。
- 当配置项被挂载至工作负载后，每个关联工作负载的所属集群中都会创建一个同名的配置项。

创建配置项

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“配置项与密钥”，切换至“配置项”页签。

步骤4 选择需要创建配置项的命名空间，并单击右上角“创建配置项”。

步骤5 参照表3-20设置新增配置参数。

表 3-20 新建配置参数说明

参数	参数说明
名称	新建的配置项名称，同一个命名空间里命名必须唯一。
命名空间	新建配置项所在的命名空间，默认为当前查看的命名空间。
描述	配置项的描述信息。
配置项数据	工作负载配置的数据可以在容器中使用，或被用来存储配置数据。 单击＋，输入键、值。其中，“键”代表配置名；“值”代表配置内容。
配置标签	标签以Key/value键值对的形式附加到各种对象上（如工作负载、节点、服务等）。 标签定义了这些对象的可识别属性，用来对它们进行管理和选择。 1. 输入标签键、值。 2. 单击“确认添加”。

步骤6 配置完成后，单击“确定”。

----结束

使用配置项

配置项创建后，您可以在创建工作负载时设置容器存储，将配置项挂载到容器中，然后在容器中挂载路径下就可以读取到配置项的内容，操作步骤请参见**配置项挂载**。

相关操作

通过UCS控制台，您还可以执行表3-21中的操作。

表 3-21 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建配置项。
查看详情	单击配置项名称即可查看配置项数据详情。

操作	说明
编辑YAML	单击配置项名称后的“编辑YAML”，可查看并编辑当前配置项的YAML文件。
更新	1. 单击配置项名称后的“更新”。 2. 根据表3-20更改信息。 3. 单击“确认”提交已修改的信息。
删除	单击配置项名称后的“删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的配置项。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.6.2 密钥（Secret）

密钥（Secret）是一种用于存储工作负载所需要认证信息、密钥的敏感信息等的资源类型，内容由用户决定。

说明

- 通过UCS控制台创建密钥后，密钥将默认处于“未部署”状态，需要在创建或更新工作负载时进行挂载，请参见密钥挂载。
- 当密钥被挂载至工作负载后，每个关联工作负载的所属集群中都会创建一个同名的密钥。

创建密钥

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“配置项与密钥”，切换至“密钥”页签。

步骤4 选择需要创建密钥的命名空间，并单击右上角“创建密钥”。

步骤5 参照表3-22设置基本信息。

表 3-22 基本信息说明

参数	参数说明
名称	新建的密钥的名称，同一个命名空间内命名必须唯一。
命名空间	新建密钥所在的命名空间，默认为当前查看的命名空间。
描述	新建密钥的描述。

参数	参数说明
密钥类型	新建的密钥类型。 • Opaque：一般密钥类型。在高敏感场景下，建议先通过数据加密服务加密敏感数据后，再存入Secret中。 • kubernetes.io/dockerconfigjson：存放拉取私有仓库镜像所需的认证信息。如选择此类型的密钥，需要额外输入镜像仓库地址。 • IngressTLS：存放7层负载均衡服务所需的证书。如选择此类型的密钥，需要上传证书文件及私钥文件。 • 其他：若需要创建其他类型的密钥，请手动输入密钥类型。
密钥数据	工作负载密钥的数据可以在容器中使用。 • 当密钥为Opaque类型时：输入键、值。其中“值”必须使用Base64编码，勾选“自动Base64编码”即可自动将输入的值转换为Base64编码。手动进行Base64编码的方法请参见**如何进行Base64编码**。 • 当密钥为dockerconfigjson类型时：输入私有镜像仓库的用户名和密码。
标签	标签以Key/value键值对的形式附加到各种对象上（如工作负载、节点、服务等）。 标签定义了这些对象的可识别属性，用来对它们进行管理和选择。 1. 单击“添加”。 2. 输入键、值。

步骤6 配置完成后，单击“确认”。

密钥列表中会出现新创建的密钥。

----结束

使用密钥

密钥创建后，您可以在创建工作负载时设置容器存储，将密钥挂载到容器中，然后在容器中挂载路径下就可以读取到密钥的内容，操作步骤请参见**密钥挂载**。

如何进行 Base64 编码

对字符串进行Base64加密，可以直接使用“echo -n 要编码的内容 | base64”命令即可，示例如下：

```
echo -n "待编码内容" | base64
```

相关操作

通过UCS控制台，您还可以执行**表3-23**中的操作。

表 3-23 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建密钥。
查看详情	单击配置项名称即可查看密钥数据详情。
编辑YAML	单击密钥名称后的“编辑YAML”，可查看到当前密钥的YAML文件。
更新	1. 单击密钥名称后的“更新”。 2. 根据表3-22更改信息。 3. 单击“确认”提交已修改的信息。
删除	单击密钥名称后的“删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的密钥。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.7 服务与路由

3.7.1 服务与路由概述

集群为满足多种复杂场景下的工作负载访问，提供了不同的访问方式，从而满足不同的业务需求。

须知

- 通过UCS控制台创建服务（Service）、路由（Ingress）后，每个关联工作负载的所属集群中都会创建一个同名的Service、Ingress。
- 您可以在集群中对UCS自动创建的Service、Ingress进行修改或删除。但如果不同步修改UCS中服务或路由的设置，最终已修改或删除的Service、Ingress会被UCS重建。因此建议您直接通过UCS控制台修改。
- 当您的集群出现异常时，Service资源对象会迁移至健康集群。当您的集群恢复正常时，需要手动修改Service模板才能重新部署。

- **集群内访问（ClusterIP）**

  表示工作负载暴露给同一集群内其他工作负载访问的方式，可以通过“集群内部域名”访问。集群内部域名格式为“<自定义的访问方式名称>.<工作负载所在命名空间>.svc.cluster.local”，例如“nginx.default.svc.cluster.local”。

- **节点访问（NodePort）**

  表示工作负载可以从集群外部访问。节点访问（NodePort）是指在每个节点的IP上开放一个静态端口，通过静态端口对外暴露服务。当集群中的节点绑定了EIP时，通过请求<EIP>:<NodePort>，也可实现从公网访问工作负载。

- **负载均衡（LoadBalancer）**

  通过弹性负载均衡从公网访问工作负载，一般用于系统中需要暴露到公网的服务，与通过绑定弹性IP的节点访问方式相比拥有更高的可靠性。访问方式由公网弹性负载均衡ELB服务地址以及设置的访问端口组成，例如“10.117.117.117:80”。

- **路由（Ingress）**

  采用了增强型弹性负载均衡，在四层负载均衡访问方式的基础上支持了URI配置，通过对应的URI将访问流量分发到对应的服务。同时，服务根据不同URI实现不同的功能。该访问方式由公网弹性负载均衡ELB服务地址、设置的访问端口组成、定义的URI组成，例如：10.117.117.117:80/helloworld。

3.7.2 服务（Service）

3.7.2.1 集群内访问（ClusterIP）

集群内访问表示工作负载暴露给同一集群内其他工作负载访问的方式，可以通过“集群内部域名”访问。集群内部域名格式为“<自定义的访问方式名称>.<工作负载所在命名空间>.svc.cluster.local”，例如“nginx.default.svc.cluster.local”。

添加方式

您可以在创建工作负载时设置访问方式，也可以在工作负载创建完成后添加访问方式。

- 方式一：创建工作负载时配置，请参见**创建工作负载时设置**。
- 方式二：工作负载创建完成后设置，请参见**创建工作负载后设置**。

创建工作负载时设置

创建Deployment、StatefulSet、DaemonSet等不同类型的工作负载时添加Service的方法一致。

步骤1 参考**创建无状态负载**、**创建有状态负载**或**创建守护进程集**，在服务配置步骤，单击 ＋，进行工作负载服务配置。

- **Service名称**：自定义服务名称，取值范围为1-50字符。
- **访问类型**：选择“集群内访问 ClusterIP”。
- **端口配置**：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。

图 3-21 工作负载服务配置

步骤2 设置完成后，单击“确认”。

步骤3 单击“下一步：调度与差异化”，进行集群调度与差异化配置。设置完成后，单击“创建工作负载”完成创建。

步骤4 获取访问地址。

1. 单击左侧导航栏“服务与路由”，选择“服务”页签。
2. 单击所添加的Service名称进入“服务详情”界面，获取部署集群的访问地址。

----结束

创建工作负载后设置

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“服务与路由”，选择“服务”页签。

步骤4 选择服务所在命名空间，并单击右上角“创建服务”。如需新建命名空间，请参见**创建命名空间**。

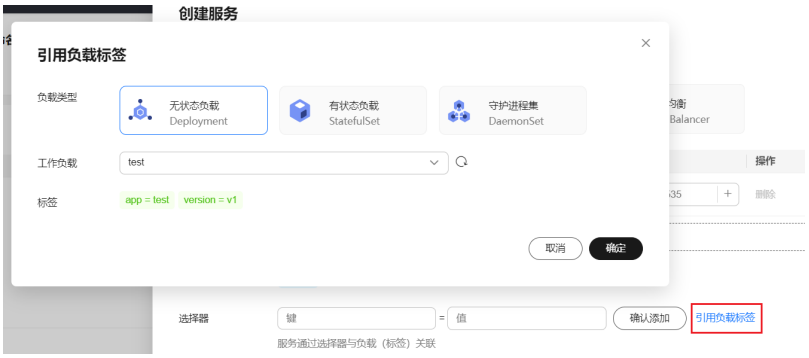
步骤5 设置访问参数。

图 3-22 创建服务

- **Service名称**：自定义服务名称，可与工作负载名称保持一致。
- **服务类型**：选择“集群内访问 ClusterIP”。
- **端口配置**：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
- **命名空间**：服务所在命名空间。
- **选择器**：服务通过选择器与负载（标签）关联。单击“引用负载标签”，可选择已有的工作负载。
  - 负载类型：选择需要关联的负载类型。

- 工作负载：选择一个已有的工作负载。如工作负载列表未显示，请单击 刷新。
- 标签：选择工作负载后自动获取对应的标签，不可修改。

图 3-23 引用负载标签



步骤6 单击“确认”。创建成功后可在“服务”页签的列表中查看。

----结束

相关操作

通过UCS控制台，您还可以执行表3-24中的操作。

表 3-24 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建服务。
查看详情	1. 选择服务所在的命名空间。 2. （可选）根据服务名称进行搜索。 3. 单击服务名称即可查看服务详情，包括基本信息以及各集群的部署信息。 4. 在服务详情页的部署集群栏中单击“查看YAML”，可查看各个集群中部署的服务实例YAML，并支持下载。
编辑YAML	单击服务名称后的“编辑YAML”，可查看并编辑当前服务的YAML文件。
更新	1. 单击服务名称后的“更新”。 2. 根据服务参数更改信息。 3. 单击“确认”提交已修改的信息。
删除	单击服务名称后的“删除”，并单击“是”进行确认。

操作	说明
批量删除	1. 勾选需要删除的服务。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.7.2.2 节点访问（NodePort）

节点访问（NodePort）表示工作负载可以从集群外部访问，节点访问方式是在每个节点上都开放一个静态端口。当外部流量访问到节点端口时，会被路由到集群中自动创建的ClusterIP服务，最终访问到工作负载。如果集群中的节点绑定了EIP，用户通过请求<EIP>:<NodePort>，也可实现工作负载的公网访问。

添加方式

您可以在创建工作负载时设置访问方式，也可以在无状态工作负载创建完成后添加访问方式。

- 方式一：创建工作负载时配置，请参见**创建工作负载时设置**。
- 方式二：工作负载创建完成后设置，请参见**创建工作负载后设置**。

创建工作负载时设置

创建Deployment、StatefulSet、DaemonSet等不同类型的工作负载时添加Service的方法一致。

步骤1 参考**创建无状态负载**、**创建有状态负载**或**创建守护进程集**，在服务配置步骤，单击＋，进行工作负载服务配置。

- **Service名称**：自定义服务名称，取值范围为1-50字符。
- **访问类型**：选择“节点访问 NodePort”。
- **服务亲和**：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- **端口配置**：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。

步骤2 设置完成后，单击“确认”。

步骤3 单击“下一步：调度与差异化”，进行集群调度与差异化配置。设置完成后，单击“创建工作负载”完成创建。

步骤4 获取访问地址。

1. 单击左侧导航栏“服务与路由”，选择“服务”页签。
2. 单击所添加的Service名称进入“服务详情”界面，获取部署集群的访问地址。如果集群下节点有绑定弹性IP，则可以通过集群下关联实例所在节点弹性IP地址 + 节点端口的形式访问后端负载。

----结束

创建工作负载后设置

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“服务与路由”，选择“服务”页签。

步骤4 选择服务所在命名空间，并单击右上角“创建服务”。如需新建命名空间，请参见**创建命名空间**。

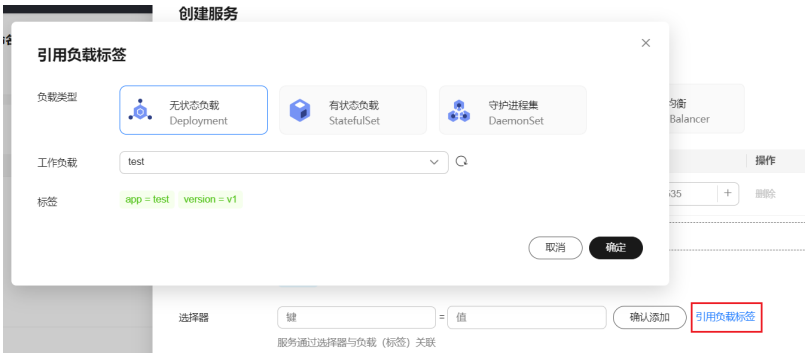
步骤5 设置访问参数。

图 3-24 创建服务

- **Service名称**：自定义服务名称，可与工作负载名称保持一致。
- **服务类型**：选择“节点访问 NodePort”。
- **服务亲和**：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- **端口配置**：
  - 协议：TCP或UDP，请根据业务的协议类型选择。

  - 服务端口：容器端口映射到集群虚拟IP上的端口，用虚拟IP访问应用时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
  - 节点端口：容器端口映射到节点私有IP上的端口，用私有IP访问应用时使用，端口范围为30000-32767，建议选择“自动生成”。
    - 自动生成：系统会自动分配端口号。
    - 指定端口：指定固定的节点端口，默认取值范围为30000-32767。若指定端口时，请确保同个集群内的端口唯一性。
- **命名空间**：服务所在命名空间。
- **选择器**：服务通过选择器与负载（标签）关联。单击“引用负载标签”，可选择已有的工作负载。
  - 负载类型：选择需要关联的负载类型。
  - 工作负载：选择一个已有的工作负载。如工作负载列表未显示，请单击 刷新。
  - 标签：选择工作负载后自动获取对应的标签，不可修改。

图 3-25 引用负载标签



步骤6 单击“确认”。创建成功后可在“服务”页签的列表中查看。

步骤7 获取访问地址。

1. 单击左侧导航栏“服务与路由”，选择“服务”页签。
2. 单击所添加的Service名称进入“服务详情”界面，获取部署集群的访问地址。如果集群下节点有绑定弹性IP，则可以通过集群下关联实例所在节点弹性IP地址 + 节点端口的形式访问后端负载。

----结束

相关操作

通过UCS控制台，您还可以执行表3-25中的操作。

表 3-25 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建服务。
查看详情	1. 选择服务所在的命名空间。 2. （可选）根据服务名称进行搜索。 3. 单击服务名称即可查看服务详情，包括基本信息以及各集群的部署信息。 4. 在服务详情页的部署集群栏中单击“查看YAML”，可查看各个集群中部署的服务实例YAML，并支持下载。
编辑YAML	单击服务名称后的“编辑YAML”，可查看并编辑当前服务的YAML文件。
更新	1. 单击服务名称后的“更新”。 2. 根据**访问参数**更改信息。 3. 单击“确认”提交已修改的信息。
删除	单击服务名称后的“删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的服务。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.7.2.3 负载均衡（LoadBalancer）

通过弹性负载均衡从公网访问工作负载，一般用于系统中需要暴露到公网的服务。访问方式由公网弹性负载均衡ELB服务地址以及设置的访问端口组成，例如“10.117.117.117:80”。

前提条件

请确保已有可用的工作负载，若没有请参照**工作负载**先创建工作负载。

添加方式

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

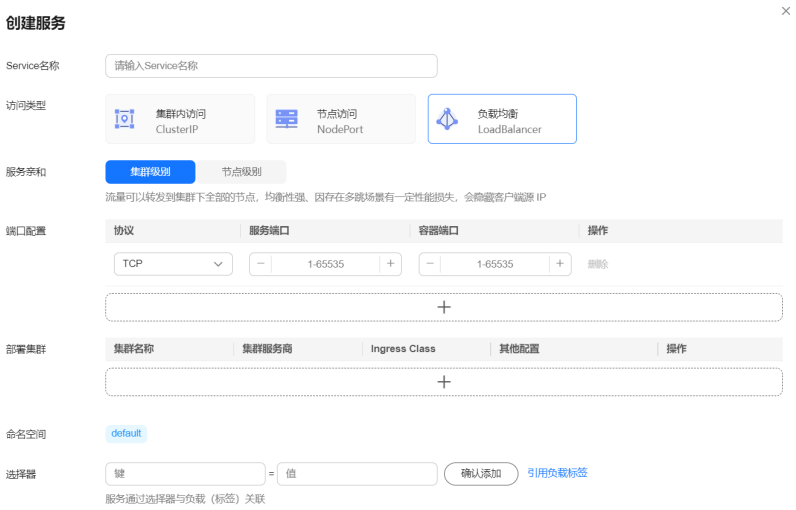
步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“服务与路由”，选择“服务”页签。

步骤4 选择服务所在命名空间，并单击右上角“创建服务”。如需新建命名空间，请参见**创建命名空间**。

步骤5 设置访问参数。

图 3-26 创建服务



- **Service名称**：自定义服务名称，取值范围为1-50字符。
- **访问类型**：选择“负载均衡 LoadBalancer”。
- **服务亲和**：
  - 集群级别：集群下所有节点的IP+访问端口均可以访问到此服务关联的负载，服务访问会因路由跳转导致一定性能损失，且无法获取到客户端源IP。
  - 节点级别：只有通过负载所在节点的IP+访问端口才可以访问此服务关联的负载，服务访问没有因路由跳转导致的性能损失，且可以获取到客户端源IP。
- **端口配置**：
  - 协议：TCP或UDP，请根据业务的协议类型选择。
  - 服务端口：容器端口映射到负载均衡实例的端口，通过负载均衡对外暴露服务时使用，端口范围为1-65535，可任意指定。
  - 容器端口：容器镜像中应用程序实际监听的端口，需用户确定。例如：nginx程序实际监听的端口为80。
- **部署集群**：选择负载均衡部署的集群，并完成负载均衡的差异化设置。

图 3-27 添加部署集群

  - **CCE集群**：

- 负载均衡器：仅支持集群所在VPC下的负载均衡实例。
- 分配策略：

  加权轮询算法：根据不同的权重将请求分配到后端服务器。

  加权最少连接：将请求分发给（当前连接/权重）比值最小的后端服务器进行处理。

  源IP算法：将客户端IP请求固定分配给一台服务器，实现获取同一个session。
- 会话保持类型：默认不启用，可选择“源IP地址”。负载均衡监听是基于IP地址的会话保持，即来自同一IP地址的访问请求转发到同一台后端服务器上。
- 健康检查：默认不启用。此处健康检查是设置负载均衡的健康检查配置，支持TCP和HTTP协议，其参数详细解释参见**表3-26**。

  表 3-26 健康检查参数说明

参数	说明	示例
检查路径	当“协议”为HTTP时设置。指定健康检查的URL地址的路径。检查路径只能以/开头，长度范围[1-80]。	/
端口	健康检查端口号，取值范围[1，65535]。 健康检查默认使用业务端口（Service的NodePort和容器端口）作为健康检查的端口。	80
检查周期	每次健康检查响应的最大间隔时间。 取值范围[1-50]。	5
超时时间（秒）	每次健康检查响应的最大超时时间。 取值范围[1-50]。	10
最大重试次数	健康检查最大的重试次数，取值范围[1-10]。	5

  - **其他云**：访问注释支持key/value对格式，请您根据自身业务以及厂家要求进行注解配置。

- **命名空间**：服务所在命名空间。
- **选择器**：服务通过选择器与负载（标签）关联。单击“引用负载标签”，可选择已有的工作负载。
  - 负载类型：选择需要关联的负载类型。
  - 工作负载：选择一个已有的工作负载。如工作负载列表未显示，请单击 刷新。

- 标签：选择工作负载后自动获取对应的标签，不可修改。

图 3-28 引用负载标签

步骤6 单击“确认”。

步骤7 获取访问地址。

1. 单击左侧导航栏“服务与路由”，选择“服务”页签。
2. 单击所添加的Service名称进入“服务详情”界面，获取部署集群的访问地址。您可以通过负载均衡的弹性IP地址 + 端口的形式访问后端负载。

----结束

相关操作

通过UCS控制台，您还可以执行表3-27中的操作。

表 3-27 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建服务。
查看详情	1. 选择服务所在的命名空间。 2. （可选）根据服务名称进行搜索。 3. 单击服务名称即可查看服务详情，包括基本信息以及各集群的部署信息。 4. 在服务详情页的部署集群栏中单击“查看YAML”，可查看各个集群中部署的服务实例YAML，并支持下载。
编辑YAML	单击服务名称后的“编辑YAML”，可查看并编辑当前服务的YAML文件。
更新	1. 单击服务名称后的“更新”。 2. 根据服务参数更改信息。 3. 单击“确认”提交已修改的信息。
删除	单击服务名称后的“删除”，并单击“是”进行确认。

操作	说明
批量删除	1. 勾选需要删除的服务。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.7.3 路由（Ingress）

Ingress使用弹性负载均衡作为流量入口对外提供访问，在四层负载均衡访问方式的基础上支持了URI配置，通过对应的URI将访问流量分发到对应的服务。用户可根据域名和路径对转发规则进行自定义，完成对访问流量的细粒度划分。该访问方式由公网弹性负载均衡ELB服务地址、设置的访问端口、定义的URI组成，例如：10.117.117.117:80/helloworld。

前提条件

请确保已有可用的工作负载，若没有请参照**工作负载**先创建工作负载。

添加方式

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

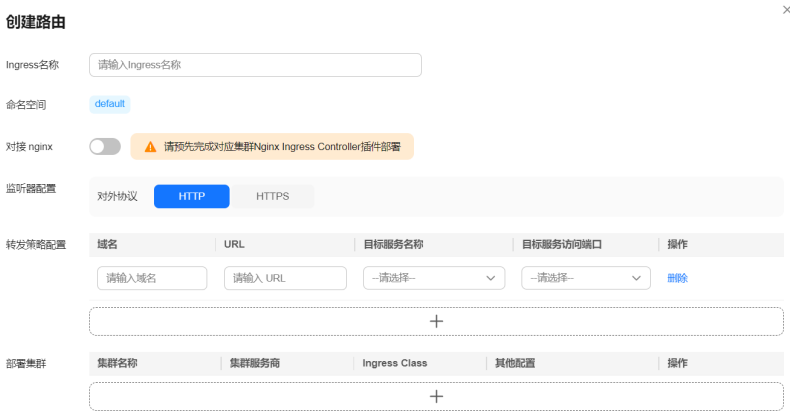
步骤3 在左侧导航栏中选择“服务与路由”，选择“路由”页签。

步骤4 选择需要创建路由的命名空间，并单击右上角“创建路由”。如需新建命名空间，请参见**创建命名空间**。

图 3-29 选择命名空间

步骤5 设置路由配置参数。

图 3-30 创建路由



- **Ingress名称**：新增路由的名称，用户可自定义。
- **命名空间**：路由所在命名空间。
- **对接nginx**：Ingress Controller分为ELB型和Nginx型。UCS支持上述两种Ingress Controller类型，其中ELB Ingress Controller基于弹性负载均衡服务（ELB）实现流量转发；而Nginx Ingress Controller使用Kubernetes社区维护的模板与镜像，通过Nginx组件完成流量转发。
  - ELB Ingress：不开启“对接nginx”。
  - Nginx Ingress：单击  开启“对接nginx”。

    ⚠ 注意

    在创建Nginx Ingress前应为对应集群安装Nginx Ingress Controller插件，安装插件的具体操作请参见：

    - 为CCE集群安装插件请参见**通过控制台创建Nginx Ingress**。
    - 为本地集群安装插件请参见**使用L7负载均衡Ingress-nginx**。
    - 为其他类型集群安装插件请参见开源社区文档**Nginx Ingress Controller**。
- **监听器配置**：选择对外协议，支持HTTP和HTTPS。若对外协议选择HTTPS，请选择IngressTLS类型的服务器证书。若无符合条件的证书，可单击“创建IngressTLS类型的密钥证书”，参考**密钥（Secret）**创建一个指定类型的密钥证书。
  - SNI（Server Name Indication）：SNI是TLS的扩展协议，在该协议下允许同一个IP地址和端口号下对外提供多个基于TLS的访问域名，且不同的域名可以使用不同的安全证书。
- **转发策略配置**：请求的访问地址与转发规则匹配时（转发规则由域名、URL组成，例如：10.117.117.117:80/helloworld），此请求将被转发到对应的目标Service处理。您可添加多条转发策略。
  - 域名：可选项，输入实际访问的域名地址。请确保所填写的域名已注册并备案，一旦配置了域名规则后，必须使用域名访问。
  - URL：需要注册的访问路径，例如：/healthz。该访问路径需与后端应用暴露的URL一致，否则将返回404错误。
  - 目标服务名称：选择服务名称，需要先创建NodePort服务，具体可参考**节点访问（NodePort）**。
  - 目标服务访问端口：选择目标服务后，对应的容器端口将自动获取。
- **部署集群**：选择需要部署的集群。

图 3-31 添加部署集群

- **CCE集群**：
  - 对外端口：开放在负载均衡服务地址的端口，可任意指定。
  - 负载均衡器：仅支持集群所在VPC下的负载均衡实例。如果没有可选的负载均衡器实例，请单击“创建负载均衡器”，创建完成后单击刷新按钮。

    注意

    创建Nginx Ingress时，您无需手动选择负载均衡器，在插件安装阶段已经完成负载均衡器的关联。
- **其他云**

  图 3-32 添加部署集群

  - Ingress Class：支持选择集群内已创建的Ingress Class，或手动输入规划的Ingress Class名称。
  - 注解：支持key/value对格式，请您根据自身业务以及厂家要求进行标签配置。
- 如需创建内部负载均衡器，您需要根据集群的云服务商添加相应的注解，请参见**内部负载均衡器**。

步骤6 单击“确认”。创建成功后可在“路由（Ingress）”页签的列表中查看。

步骤7 获取访问地址。

1. 单击左侧导航栏“服务与路由”，选择“路由”页签。
2. 单击所添加的Ingress名称进入“路由详情”界面，查看对应的负载均衡器及监听器端口配置。您可以通过负载均衡器的弹性IP地址 + 监听器端口 + URL的形式访问后端负载，例如10.117.117.117:8088/helloworld。

----结束

相关操作

通过UCS控制台，您还可以执行表3-28中的操作。

表 3-28 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建路由。
查看详情	1. 选择路由所在的命名空间。 2. （可选）根据路由名称进行搜索。 3. 单击路由名称即可查看路由详情，包括基本信息以及各集群的部署信息。 4. 在路由详情页的部署集群栏中单击“查看YAML”，可查看各个集群中部署的路由实例YAML，并支持下载。
编辑YAML	单击路由名称后的“编辑YAML”，可查看并编辑当前路由的YAML文件。
更新	1. 单击路由名称后的“更新”。 2. 根据**路由参数**更改信息。 3. 单击“确认”提交已修改的信息。
删除	单击路由名称后的“删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的路由。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.8 多集群 Ingress

3.8.1 MCI 概述

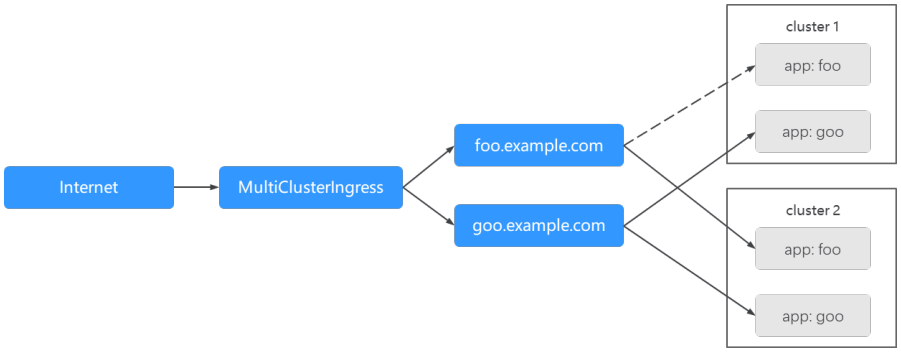
为什么需要 MCI

在传统的Kubernetes集群中，每个集群都有其负载均衡器和Ingress，这使得在多个集群之间进行负载均衡和流量路由变得困难。UCS为多集群提供了统一的流量入口MCI

（Multi Cluster Ingress），帮助您更加简单高效地跨集群、跨区域进行负载均衡和流量路由，进而提高应用程序的可用性和可靠性。

MCI是一种多集群Ingress资源对象，可以根据指定规则将外部流量路由到多个集群内的Pod。使用MCI，用户可自定义转发规则，完成对访问流量的细粒度划分。图1展示了流量如何从MCI流向集群：从域名 foo.example.com 流入的流量流向了两个集群中带有标签 app:foo 的 Pod，从域名goo.example.com 流入的流量流向了两个集群中具有标签 app:goo的 Pod。

图 3-33 MCI 示意图



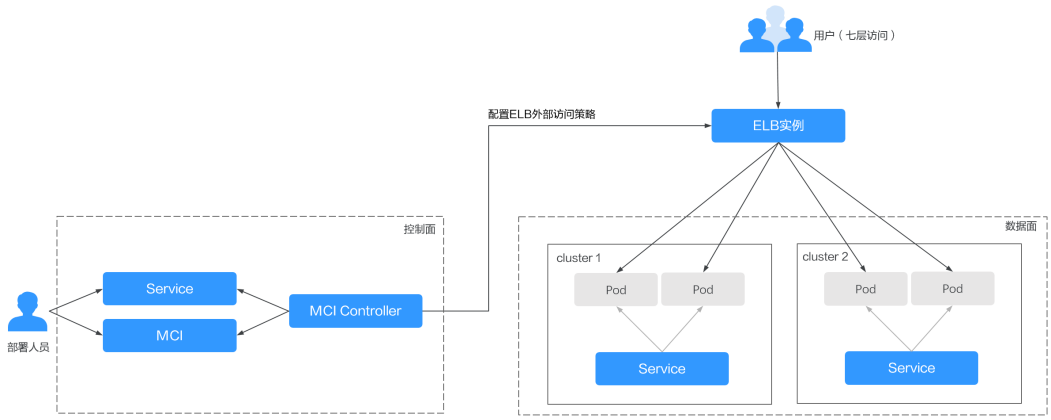
具体来说，MCI的优势在于：

- 多集群负载均衡：MCI为用户提供统一入口，将流量分发到多个Kubernetes集群中，与集群所处位置无关。
- 流量路由：使用MCI，用户可根据不同条件（URL、HTTP头、源IP等）自定义转发规则，将流量路由到不同集群，实现更灵活的流量控制。
- 高可用：MCI通过健康检查和流量自动切换，实现多集群、多区域的高可用性。
- 可扩展性：MCI发现和管理多个Kubernetes集群上的应用程序资源，实现应用程序自动扩展和部署。
- 安全性：MCI支持TLS安全策略和证书管理，保障应用程序的安全性。

MCI 的工作原理

MCI的功能主要通过请求转发执行器MCI Controller实现。MCI Controller部署在集群联邦控制面，实时监控资源对象的变化，解析MCI对象定义的规则并负责将请求转发到相应的后端Service。

图 3-34 MCI Controller 工作原理



MCI Controller支持在同一个ELB实例（即同一IP）下设置不同域名、端口和转发规则，其工作原理如图2，实现流程如下：

1. 部署人员在集群联邦控制面创建工作负载，并为其配置Service对象。
2. 部署人员在集群联邦控制面创建MCI对象，并在MCI中配置流量访问规则，包括关联的负载均衡器、URL以及访问的后端Service及端口等。
3. MCI Controller监控到MCI对象发生变化，会根据MCI中定义的规则，在ELB侧重新配置监听器以及后端服务器路由。
4. 当用户进行访问时，流量根据ELB中配置的转发策略被转发到对应的后端Service关联的各个工作负载。

3.8.2 使用 MCI

约束限制

- 当前MCI仅支持CCE Turbo 1.21及以上版本集群。
- 当前MCI仅ELB实例与集群处于同一个VPC内，成员集群间容器网段不冲突，确保ELB实例与容器pod IP网络可达。
- 当为同一Service同时配置MCI与MCS时，该Service将会下发至MCS中配置的下发集群、访问集群以及对应工作负载的部署集群。

准备工作

- 如您在同一VPC下没有可用的ELB，需要先创建ELB实例，具体请参考**创建独享型负载均衡器**。该ELB实例需要满足以下条件：
  - ELB为独享型。
  - ELB必须支持应用型（HTTP/HTTPS）。
  - ELB网络类型必须支持私网（有私有IP地址）。
- MCI为跨集群后端工作负载提供统一入口和七层网络访问，因此需要在联邦中提前部署可用的工作负载（Deployment）和服务（Service）。若您无可用工作负载和服务，请参考**无状态负载**和**集群内访问（ClusterIP）**创建。

创建 MCI 对象

步骤1 使用kubectl连接集群联邦，详细操作请参见使用kubectl连接集群联邦。

步骤2 创建并编辑 mci.yaml 文件，文件内容定义如下所示，参数定义请参见**表3-29**。

vi mci.yaml

```
apiVersion: networking.karmada.io/v1alpha1
kind: MultiClusterIngress
metadata:
  name: nginx-ingress
  namespace: default
  annotations:
    karmada.io/elb.conditions.nginx-svc:
      '[{
          "type": "header",
          "headerConfig": {
              "key":"x-header",
              "values": [
              "green"
              ]
          }
      }]'
```

```
    karmada.io/elb.id: 90f9f782-1243-41cc-a57d-6157f6cb85bf
    karmada.io/elb.projectid: 65382450e8f64ac0870cd180d14e684b
    karmada.io/elb.health-check-flag: "on"
    karmada.io/elb.health-check-option.nginx-svc: '{"protocol":"TCP"}'
spec:
  ingressClassName: public-elb
  rules:
  - host: demo.localdev.me
    http:
      paths:
      - backend:
          service:
            name: nginx-svc       # 准备一个名为nginx-svc的联邦service
            port:
              number: 8080        # 端口号为8080
        path: /web
        pathType: Prefix
```

MCI对象的结构体定义与networking.kubernetes.io/v1版本Ingress一致，不同之处在于后端服务需要填写为联邦Service，即在UCS控制台创建的Service，具体请参见集群内访问（ClusterIP）。

须知

在配置MCI文件内容的过程中需要遵守的约束条件如下：

- apiVersion，kind，name必须指定。
- spec下不允许填写TLS和DefaultBackend字段。
- rules、paths不能为空。
- host必须是DNS名称，不可以是IP地址。
- service中所指定的后端服务必须是存在的、且输入的相关信息（如端口）是正确的，否则会导致访问服务失败。若您已经创建了参数信息错误的MCI对象，请参考步骤4中的命令更新该MCI对象。
- paths中，配置的高级转发策略（karmada.io/elb.conditions.{service name}）越多的后端（backend）在paths中的位置应该越靠前。因为backend在paths中配置的位置越靠前，其转发优先级越高。

  示例：如果为后端X配置两条转发策略a和b，为后端Y配置一条转发规则a，则此时paths中X的配置顺序应在Y之前，否则，同时符合a、b两条转发规则的流量将按照优先级顺序全部转发至Y中。
- backend下不允许填写resource字段。
- path值需要是绝对路径；不合法的path值：invalidPathSequences = []string{"//", "/./", "/../", "%2f","%2F"}，invalidPathSuffixes = []string{"/..", "/."}。
- pathType合法值：Exact、Prefix、ImplementationSpecific。

表 3-29 关键参数说明

参数	是否必填	参数类型	描述
karmada.io/elb.id	是	String	MCI关联的elb的id，不允许为空。长度为1-32个字符。

参数	是否必填	参数类型	描述
karmada.io/elb.projectid	是	String	MCI关联的elb所属的项目ID。长度为1-32个字符。
karmada.io/elb.port	否	String	MCI关联的elb的端口，不填默认为80。支持值为1-65535。
karmada.io/elb.health-check-flag	否	String	是否启用健康检查，可选值为： • on：开启 • off：不开启 不填写时默认为off。
karmada.io/elb.health-check-option	否	**HealthCheck** Object	健康检查参数，详情请参见**HealthCheck**。 **说明** 健康检查参数配置示例： karmada.io/elb.health-check-option.nginx-svc: '{"protocol":"TCP","delay":"5","connect_port":"80","timeout":"1","max_retries":"1","path":"/wd"}'
karmada.io/elb.conditions.{service name}	否	Array of **Condition** Object	高级转发策略，详情请参见**Condition**。{service name}请修改为联邦Service的名称。
karmada.io/elb.lb-algorithm.{service name}	否	String	转发算法： • ROUND_ROBIN：加权轮询算法。 • LEAST_CONNECTIONS：加权最少连接算法。 • SOURCE_IP：源IP算法。 不填写时默认为ROUND_ROBIN。 {service name}请修改为联邦Service的名称。
ingressClassName	是	String	ingressClass名称。取值必须为public-elb。
host	否	String	为服务访问域名配置，默认为""，表示域名全匹配。请确保所填写的域名已注册并备案，一旦配置了域名规则后，必须使用域名访问。

参数	是否必填	参数类型	描述
backend	否	**Backend** Object	后端，是Service 和端口名称的组合， 。 对于发往MCI的 HTTP（和 HTTPS）请求，如果与规则中的主机和路径匹配， 则会被发送到所列出的后端。 **注意** 后端在paths中的配置顺序决定了策略的转发优先级。 示例：如果为后端X配置两条转发策略a和b，为后端Y配置一条转发规则a，则此时paths中X的配置顺序应在Y之前，否则，同时符合a、b两条转发规则的流量将按照优先级顺序全部转发至Y中。
path	是	String	为路由路径，用户自定义设置。所有外部访问请求需要匹配host和path。 **说明** 此处添加的访问路径要求后端应用内存在相同的路径，否则转发无法生效。 例如，Nginx应用默认的Web访问路径为“/usr/share/nginx/html”，在为Ingress转发策略添加“/test”路径时，需要应用的Web访问路径下也包含相同路径，即“/usr/share/nginx/html/test”，否则将返回404。

参数	是否必填	参数类型	描述
pathType	是	String	路径类型。 • ImplementationSpecific: 匹配方法取决于具体Ingress Controller的实现。在CCE中会使用ingress.beta.kubernetes.io/url-match-mode指定的匹配方式。 • Exact：精确匹配 URL 路径，且区分大小写。 • Prefix：前缀匹配，且区分大小写。该方式是将URL路径通过“/”分隔成多个元素 ，并且对元素进行逐个匹配。 如果URL中的每个元素均和路径匹配，则说明该URL的子路径均可以正常路由。 **说明** – Prefix匹配时每个元素均需精确匹配，如果URL的最后一个元素是请求路径中最后一个元素的子字符串，则不会匹配 。例如：/foo/bar匹配/foo/bar/baz，但不匹配/foo/barbaz。 – 通过“/”分隔元素时，若URL或请求路径以“/”结尾，将会忽略结尾的“/”。例如：/foo/bar会匹配/foo/bar/。 关于Ingress路径匹配示例，请参见示例。

表 3-30 HealthCheck 参数说明

参数	是否必填	参数类型	描述
protocol	否	String	健康检查使用的协议，支持TCP/HTTP。
connect_port	否	Int	健康检查使用的端口。支持值为1-65535。
delay	否	Int	健康检查的延迟时间，以秒为单位，1-50。

参数	是否必填	参数类型	描述
timeout	否	Int	健康检查的超时时间，以秒为单位，1-50。
path	否	String	健康检查的请求URL，当type为HTTP/HTTPS时生效。 以"/"开头，默认为"/"。支持使用字母、数字和短划线（-）、正斜线（/）、半角句号（.）、百分号（%）、半角问号（?）、井号（#）和and（&）以及扩展字符集。长度为1-80个字符。
max_retries	否	Int	最大重试次数，取值范围1-10。

表 3-31 Condition 参数说明

参数	是否必填	参数类型	描述
type	是	String	高级转发策略类型，当前仅支持header。
headerConfig	是	**headerConfig** Object	高级转发策略对象，详情请参见**headerConfig**。

表 3-32 headerConfig 参数说明

参数	是否必填	参数类型	描述
key	是	String	转发Header头。 长度限制1-40字符，只允许包含字母、数字、短划线和下划线。
values	是	String数组	转发Header头对应的值。 长度限制1-128字符，不支持空格，双引号，支持以下通配符：*（匹配0个或更多字符）和?（正好匹配1个字符）。

步骤3 执行如下命令创建MCI对象。

kubectl apply -f mci.yaml

回显如下。

```
multiClusterIngress.networking.karmada.io/nginx-ingress created
```

步骤4 创建完成后，可以执行如下命令操作MCI对象。其中nginx-ingress为MCI对象的名称。

- 获取MCI对象：**kubectl get mci nginx-ingress**
- 更新MCI对象：**kubectl edit mci nginx-ingress**
- 删除MCI对象：**kubectl delete mci nginx-ingress**

----结束

通过 MCI 访问服务

MCI对象创建成功后，您可以通过 **http://IP:port/path** 访问后端工作负载，其中IP:port为MCI关联ELB的IP和端口，path为MCI对象中定义的路径。

MCI对象中还可以设置外部域名，这样您就可以通过域名来访问到ELB，进而访问到后端服务。

```
spec:
  rules:
  - host: www.example.com       # 域名
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx    # 准备一个名为nginx的联邦service
          servicePort: 80       # 端口号为80
```

说明

域名访问依赖于域名解析，需要您将域名解析指向ELB实例的IP地址，例如您可以使用**云解析服务 DNS**来实现域名解析。

3.8.3 配置 MCI 自动切流

3.8.3.1 自动切流概述

为什么需要自动切流

MCI提供了一种跨集群的负载均衡和流量路由机制，提高了应用程序的可用性和可靠性。然而，当集群出现故障后，通过MCI分配到该集群上的请求将会失败。

UCS提供自动切流能力，可自动摘除故障集群上的流量，以进一步保障服务的可用性。自动切流能力的应用场景主要有：

- 识别集群故障并自动切流：在集群内关键组件CoreDNS功能故障后，自动探查并及时上报至控制面，将该集群上的流量摘除，由此保障服务可用性不受单个集群组件故障影响。具体切流操作请参见**配置条件触发自动切流**。
- 提前摘除流量以平滑升级：在集群管理员进行集群升级等操作前，提前将该集群上的流量摘除，操作成功后再切回流量，由此保障服务可用性不受单个集群升级影响。具体切流操作请参见**配置无条件触发自动切流**。

约束限制

当前自动切流仅支持CCE Turbo 1.21及以上版本集群。

3.8.3.2 配置无条件触发自动切流

集群管理员进行集群升级等操作，若出现升级策略不恰当、升级配置有误、操作人员执行失误等问题，可能会导致集群不可用。本小节指导您在进行集群升级前，通过创建无条件触发的Remedy对象，将MCI流量从目标集群上摘除。

创建Remedy对象可在特定触发条件下执行特定动作。集群管理员准备升级目标集群时（如member1），可以创建如下Remedy对象，将MCI流量从member1上摘除。

示例YAML定义了一个Remedy对象，触发条件为空，表示无条件触发，集群联邦控制器会立即将member1上的流量摘除。在集群升级成功之后，删除该Remedy对象，member1上的流量会自动恢复，由此保证单集群的升级不会影响服务的高可用。详细的Remedy对象参数说明请参见表3-33。

```
apiVersion: remedy.karmada.io/v1alpha1
kind: Remedy
metadata:
  name: foo
spec:
  clusterAffinity:
    clusterNames:
      - member1
  actions:
  - TrafficControl
```

表 3-33 Remedy 参数说明

参数	描述
spec.clusterAffinity.clusterNames	策略关注的集群名列表。仅在该列表中的集群会执行指定动作，为空时不会执行任何动作。
spec.decisionMatches	触发条件列表。当上述集群列表中指定的集群满足任一触发条件时，即会执行指定动作。当列表为空时，表示无条件触发。
conditionType	触发条件的类型。当前仅支持ServiceDomainNameResolutionReady类型，即CPD上报的CoreDNS域名解析状态。
operator	判断逻辑，仅支持Equal和NotEqual两种值，即等于和不等于。
conditionStatus	触发条件的状态。
actions	策略要执行的动作，目前仅支持TrafficControl，即流量控制。

3.8.3.3 配置条件触发自动切流

本小节指导您配置条件触发自动切流，以识别集群CoreDNS功能故障并自动摘除流量。

为集群安装 CPD 组件识别集群

在配置自动切流前，您需要在集群中安装CPD（cluster-problem-detector）组件，以自动探测集群CoreDNS域名解析功能是否正常，并进行上报。

CPD会周期性地解析kubernetes.default域名，将解析结果更新到Node对象的conditions中。主CPD Pod会收集各个Node上的conditions，判断集群域名解析功能是否正常，并将该判断结果上报到至集群联邦控制面。

CPD以DaemonSet的形式部署，需要独立部署在每个目标集群的所有节点上。CPD的配置文件示例如下所示，请参见**表3-34**中的描述修改相关参数。

表 3-34 CPD 参数说明

参数	描述
<federation-version>	集群所在联邦的版本号。请在“容器舰队”页签下单击目标舰队名称以获取联邦版本号。
<your-cluster-name>	需要安装CPD组件的集群名称。
<kubeconfig-of-karmada>	可访问联邦控制面的kubeconfig文件内容。关于如何下载满足要求的kubeconfig文件，请参见**kubeconfig**。 **注意** • 下载kubeconfig文件时，需要选择部署集群所在的VPC，或者通过云连接、对等连接等方式打通到集群网络的VPC。 • 若kubeconfig文件中的联邦控制面地址为域名，则需要在部署文件中配置hostAliaes。
hostAliases	若kubeconfig文件中的联邦控制面地址为域名，则需要在部署文件中配置hostAliases，否则请删除YAML中的hostAliases配置。 • 将<host name of karmada server>替换为联邦控制面地址域名。获取联邦控制面地址域名请查看kubeconfig文件中的server字段，如下图所示。  • 将<ip of host name of karmada server>替换为该联邦控制面地址域名对应的IP地址。请在集群节点访问联邦控制面地址域名，获取解析后的IP地址。
coredns-detect-period	CoreDNS进行探测和上报的周期，默认5s（推荐值）。周期越短，探测和上报越频繁。
coredns-success-threshold	CoreDNS域名解析成功持续时长超过该阈值，则认为CoreDNS功能正常，默认30s（推荐值）。阈值越大，探测状态越稳定，但是灵敏度越低。阈值越低，灵敏度越高，但可能导致探测状态不稳定。

参数	描述
coredns-failure-threshold	CoreDNS域名解析失败持续时长超过该阈值，则认为CoreDNS功能故障，默认30s（推荐值）。阈值越大，探测状态越稳定，但是灵敏度越低。阈值越低，灵敏度越高，但可能导致探测状态不稳定。

```
kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: cluster-problem-detector
  namespace: kube-system
  labels:
    app: cluster-problem-detector
spec:
  selector:
    matchLabels:
      app: cluster-problem-detector
  template:
    metadata:
      labels:
        app: cluster-problem-detector
    spec:
      containers:
        - image: swr.ap-southeast-3.myhuaweicloud.com/hwofficial/cluster-problem-detector:<federation-
version>
          name: cluster-problem-detector
          command:
            - /bin/sh
            - '-c'
            - /var/paas/cluster-problem-detector/cluster-problem-detector
              --karmada-kubeconfig=/tmp/config
              --karmada-context=federation
              --cluster-name=<your-cluster-name>
              --host-name=${HOST_NAME}
              --bind-address=${POD_ADDRESS}
              --healthz-port=8081
              --detectors=*
              --coredns-detect-period=5s
              --coredns-success-threshold=30s
              --coredns-failure-threshold=30s
              --coredns-stale-threshold=60s
          env:
            - name: POD_ADDRESS
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: status.podIP
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: POD_NAMESPACE
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.namespace
            - name: HOST_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: spec.nodeName
          livenessProbe:
            httpGet:
```

```
            path: /healthz
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 3
          timeoutSeconds: 3
          periodSeconds: 5
          successThreshold: 1
          failureThreshold: 3
        readinessProbe:
          httpGet:
            path: /healthz
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 3
          timeoutSeconds: 3
          periodSeconds: 5
          successThreshold: 1
          failureThreshold: 3
        volumeMounts:
          - mountPath: /tmp
            name: karmada-config
    serviceAccountName: cluster-problem-detector
    volumes:
      - configMap:
          name: karmada-kubeconfig
          items:
            - key: kubeconfig
              path: config
        name: karmada-config
    securityContext:
      fsGroup: 10000
      runAsUser: 10000
      seccompProfile:
        type: RuntimeDefault
    hostAliases:
      hostnames:
        - <host name of karmada server>
      ip: <ip of host name of karmada server>
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cluster-problem-detector
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cpd-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:cluster-problem-detector
subjects:
  - kind: ServiceAccount
    name: cluster-problem-detector
    namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:cluster-problem-detector
rules:
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
```



```
    - list
    - watch
  - apiGroups:
    - ""
    resources:
    - nodes/status
    verbs:
    - patch
    - update
  - apiGroups:
    - ""
    - events.k8s.io
    resources:
    - events
    verbs:
    - create
    - patch
    - update
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: karmada-kubeconfig
  namespace: kube-system
data:
  kubeconfig: |+
    <kubeconfig-of-karmada>
```

检查 CPD 组件是否正常运行

部署CPD组件后，请检查CPD组件是否正常运行。

- 执行以下命令，检查节点的conditions中是否有“ServiceDomainNameResolutionReady”类型的 condition 出现，并检查该condition的“lastHeartBeatTime”是否及时更新。

  kubectl get node <node-name> -oyaml | **grep** -B4 ServiceDomainNameResolutionReady

  若节点中没有上述类型的condition出现，或者该condition“lastHeartBeatTime”长时间不更新：

  a. 请检查CPD中的pod是否处于Ready状态。

  b. 请检查成员集群中是否有“LoadCorednsConditionFailed”或者“StoreCorednsConditionFailed”类型的事件。若存在，请按事件中的错误提示进行处理。

- 执行以下命令，检查集群联邦cluster对象中是否有“ServiceDomainNameResolutionReady”类型的condition出现。

  kubectl --kubeconfig <kubeconfig-of-federation> get cluster <cluster-name> -oyaml | **grep** ServiceDomainNameResolutionReady

  若cluster对象中没有上述类型的condition：

a. 请检查CPD日志中是否出现“failed to sync corendns condition to control plane, requeuing”错误。
b. 请检查kubeconfig配置是否有误。若kubeconfig有更新，请重新部署CPD。
c. 请检查CPD所在节点的网络和下载kubeconfig文件所使用的VPC网络是否打通。

配置条件触发自动切流策略

在CPD组件部署成功并正常运行后，您需要创建Remedy对象，以在特定触发条件下执行特定动作，如在集群CoreDNS组件故障后执行MCI切流。

Remedy对象的配置文件示例如下所示。示例YAML定义了一个Remedy对象，可以通过member1或member2集群上的CPD上报CoreDNS解析功能，在功能故障时自动将该集群上的MCI进行切流。详细的Remedy对象参数说明请参见**表3-35**。

```
apiVersion: remedy.karmada.io/v1alpha1
kind: Remedy
metadata:
  name: foo
spec:
  clusterAffinity:
    clusterNames:
      - member1
      - member2
  decisionMatches:
  - clusterConditionMatch:
      conditionType: ServiceDomainNameResolutionReady
      operator: Equal
      conditionStatus: "False"
  actions:
  - TrafficControl
```

表 3-35 Remedy 参数说明

参数	描述
spec.clusterAffinity.clusterNames	策略关注的集群名列表。仅在该列表中的集群会执行指定动作，为空时不会执行任何动作。
spec.decisionMatches	触发条件列表。当上述集群列表中指定的集群满足任一触发条件时，即会执行指定动作。当列表为空时，表示无条件触发。
conditionType	触发条件的类型。当前仅支持ServiceDomainNameResolutionReady类型，即CPD上报的CoreDNS域名解析状态。
operator	判断逻辑，仅支持Equal和NotEqual两种值，即等于和不等于。
conditionStatus	触发条件的状态。
actions	策略要执行的动作，目前仅支持TrafficControl，即流量控制。

3.9 多集群 Service

3.9.1 MCS 概述

为什么需要 MCS

Service是Kubernetes中的一种资源对象，定义了一组Pod的访问方式。Service为Pod提供了一个稳定的IP地址和DNS名称，使得应用程序可以通过该IP地址或DNS名称来访问这些Pod，从而自动发现和连接到可用的Pod。在多集群场景下，为了满足数据主权、状态管理、可伸缩性等要求，需要将服务拆分至多个集群进行访问，这个过程相较单集群的服务访问而言更加困难。

MCS（Multi Cluster Service）是一种多集群Service资源对象，将Service的边界从单个集群扩展至集群联邦，帮助您简洁快速地实现跨集群服务发现和访问。

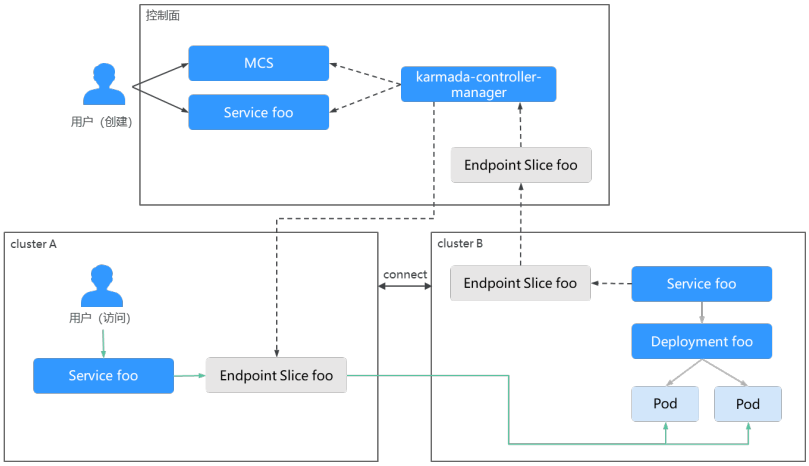
具体来说，MCS的应用场景非常广泛，典型的应用场景包括：

- 应用容灾：使用MCS，您可以在多个区域的不同集群中访问同一个Service，因此如果某个集群中应用不可用，访问请求可切换至其他集群进行处理。
- 服务共享：使用MCS，可以更便捷地在不同集群中访问公共服务（监控、日志系统等），无需为所有集群部署本地公共服务副本。
- 应用迁移：MCS桥接了不同集群服务间的通信，您可以将同一Service部署到不同集群，通过流量迁移来更轻松地进行应用迁移。

MCS 的工作原理

MCS的功能主要通过控制面组件karmada-controller-manager实现。karmada-controller-manager实时监控Service和MCS的变化，解析MCS对象定义的规则并负责将请求转发到相应的后端Service。

图 3-35 MCS 工作原理



MCS的工作原理如图3-35，实现流程如下：

1. 用户在集群联邦控制面创建工作负载，并为其配置Service对象。图中名为foo的Service部署在cluster B中。
2. 用户在集群联邦控制面创建MCS对象，并在MCS中配置访问规则，包括下发集群与目标集群名称等。图中的下发集群为cluster B，目标集群为cluster A。
3. karmada-controller-manager监控到Service和MCS对象的变化，将Service下发到源集群，并将源集群中的Endpoint Slices采集并下发到目标集群。

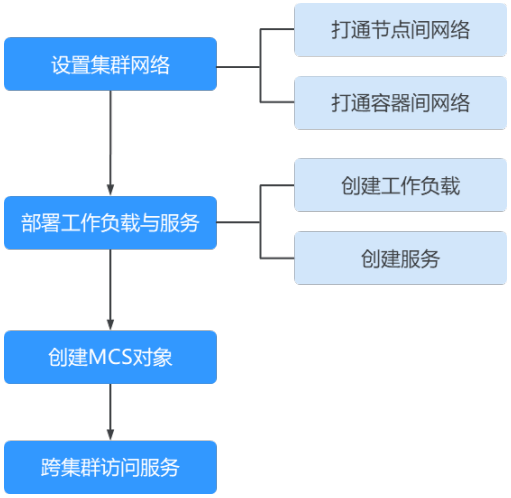
4. 用户在目标集群（cluster A）中访问部署在下发集群（cluster B）中的Service时，请求将会被路由到源集群的服务后端，从而实现跨集群服务发现及访问。

MCS 的使用流程

MCS的使用流程见**图3-36**，具体的使用流程如下：

1. 检测集群间的节点网络互通与容器网络互通，若未互通则需按照要求打通。具体操作请参见**设置集群网络**。
2. 在集群联邦中提前部署可用的工作负载和服务。具体操作请参见**准备工作**。
3. 创建MCS对象，并在MCS中配置访问规则。具体操作请参见**创建MCS对象**。
4. 跨集群访问服务，即在MCS中配置的目标集群中访问部署在下发集群中的服务。具体操作请参见**跨集群访问服务**。

图 3-36 MCS 使用流程



3.9.2 使用 MCS

3.9.2.1 设置集群网络

创建MCS前，需要保证集群间的节点网络互通与容器网络互通。

请参考**表3-36**检查集群网络情况。若集群或容器间网络还未打通，请参考表中设置方法对集群网络进行配置。若按照方法进行设置后仍无法打通网络，请参考**常见问题**进行问题排查。

表 3-36 打通集群间网络

集群间网络	检查方法	打通方法
节点网络互通	在集群A中ping集群B的节点IP，ping通则说明网络互通。	1. 设置集群网络类型 将集群网络类型设置为underlay，以支持集群间Pod通信。具体操作请参见参考**设置集群网络类型**。 2. 打通集群间网络 • CCE集群间网络打通： 同VPC下的CCE集群间网络直接互通。 跨VPC的CCE集群间网络可以通过对等连接方式打通。 • 本地集群间网络打通： 本地集群间的网络打通需设置Cilium为underlay模式，并对接主机BGP，具体操作请参见**Cilium**。 • 其他类型集群间网络打通： 请自行打通其他类型集群间网络。
容器网络互通	在集群A中curl集群B的Pod IP，curl通则说明网络互通。	

设置集群网络类型

需要保证集群的网络类型支持underlay网络，以支持集群间Pod通信。支持underlay网络的集群类型如下：

表 3-37 支持 underlay 网络的集群类型

集群类型	细分类型	网络类型	是否支持underlay网络
华为云集群	CCE集群	容器隧道网络	不支持
		VPC网络	支持
	CCE Turbo集群	云原生网络2.0	支持
本地集群	本地集群	同时支持overlay和underlay网络。 默认为overlay网络，要启用underaly网络需要手动配置，underaly网络相关介绍与操作请参见**Cilium**。	支持
多云集群	多云集群	同时支持overlay和underlay网络。 默认为overlay网络，要启用underaly网络需要手动配置，underaly网络相关介绍与操作请参见**Cilium**。	支持

集群类型	细分类型	网络类型	是否支持underlay网络
附着集群	附着集群	需支持underlay网络	取决于附着集群网络类型

常见问题

若检测结果显示集群间节点或容器访问不通，请排查以下项目：

表 3-38

排查项	排查方法	解决方案
确认集群版本是否在1.21及以上	进入集群详情页查看。	升级集群版本，具体操作请参见**升级集群**。
确认集群网络类型是否支持underlay网络	参照**表3-37**进行排查。	参照**表3-37**进行设置。
排查网络是否存在网段冲突	进入对等连接详情页查看。	修改对等连接路由中的冲突网段。

3.9.2.2 创建 MCS

约束限制

- 当前MCS仅支持v1.21及更高版本的集群创建。
- 当为同一Service同时配置MCS与MCI时，该Service将会下发至MCS中配置的下发集群、访问集群以及对应工作负载的部署集群。

准备工作

- 部署工作负载与服务

  MCS提供跨集群服务发现与访问能力，因此需要在集群联邦中提前部署可用的工作负载（Deployment）和服务（Service）。若您无可用工作负载和服务，请参考**无状态负载**和**集群内访问（ClusterIP）**创建。
- 设置集群网络

  请参照**设置集群网络**对集群间网络互通进行检查与设置。

创建 MCS 对象

步骤1 使用kubectl连接集群联邦，详细操作请参见使用kubectl连接集群联邦。

步骤2 创建并编辑 mcs.yaml 文件，文件内容定义如下所示，参数定义请参见**表3-39**。

vi mcs.yaml

示例YAML定义的MCS对象关联了名为foo的Service，支持在cluster A中访问到部署在cluster B中的该Service。

```
apiVersion: networking.karmada.io/v1alpha1
kind: MultiClusterService
metadata:
  name: foo                    # MCS对象名称
  namespace: default               # MCS对象所在命名空间名称
spec:
  types:
    - CrossCluster             # 类型为集群间服务发现
  providerClusters:             # Service的下发集群
    - name: clusterB
  consumerClusters:               # Service的访问集群
    - name: clusterA
```

表 3-39 关键参数说明

参数	是否必填	参数类型	描述
metadata.name	是	String	MCS对象的名称，应与关联Service保持一致。
metadata.namespace	否	String	MCS对象所在命名空间名称，应与关联Service所在命名空间名称保持一致。不填则默认为default。
spec.types	是	String	流量方向，实现集群间服务发现能力应配置为CrossCluster。
spec.providerClusters.name	否	String	Service的下发集群名称，应配置为Service的部署集群。不填则默认将Service下发至集群联邦内所有集群。 **注意** 若Service部署在cluster B，但在下发集群中配置了cluster A与cluster B，则该Service会同时下发至cluster A与cluster B，覆盖A中的原同名Service。
spec.consumerClusters.name	否	String	Service的访问集群名称，应配置为期望通过MCS实现跨集群访问Service的集群名称。不填则默认设置为所有集群联邦内集群可访问该Service。

步骤3 执行如下命令创建MCS对象。

kubectl apply -f mcs.yaml

步骤4 执行如下命令查看MCS对象的状态。其中foo为MCS对象的名称。

kubectl describe mcs foo

YAML文件中的status字段记录了MCS的状态，当status字段中出现如下内容时，表明Endpoint Slices下发并同步成功，集群间服务发现能力可用。

```
status:
  conditions:
  - lastTransitionTime: "2023-11-20T02:30:49Z"
```



```
message: EndpointSlices are propagated to target clusters.
  reason: EndpointSliceAppliedSuccess
  status: "True"
type: EndpointSliceApplied
```

创建完成后，可以执行如下命令操作MCS对象。其中foo为MCS对象的名称。

- 获取MCS对象：**kubectl get mcs foo**
- 更新MCS对象：**kubectl edit mcs foo**
- 删除MCS对象：**kubectl delete mcs foo**

----结束

跨集群访问服务

MCS对象创建成功后，您可以在consumerClusters.name中配置的访问集群中，访问到部署在下发集群中的Service。

创建一个Pod并进入到容器内，使用**curl *http://服务名称:端口号***命令访问Service，如下所示。

回显出现如下信息表明访问成功。

```
/ # curl http://服务名称:端口号
...
<h1>Welcome to foo!</h1>
...
```

3.10 域名访问

在多个集群中部署的应用可以通过公网域名实现统一的访问。您可以配置您的公网域名，UCS服务会以此域名作为根域名生成一个完整的应用外部访问域名。您可以通过发布域名访问，将服务，路由自动对接到华为云云解析DNS服务，以提供多云场景下应用的统一对外访问路径。同时域名访问提供了自定义权重配比等能力，可以根据您的需求控制外部流量的流向比例。

配置域名

为应用添加域名访问时，必须确保您配置的域名已在域名服务商完成注册并备案，否则域名可能无法访问。

若您已有经过注册和备案的域名，请直接至**步骤3**创建公网域名解析。

若您尚未注册域名，请先**购买公网域名**，并按照页面提示流程完成域名的备案、解析和相关配置。域名注册、备案流程如下。

步骤1 购买公网域名。以ucsclub.cn为例。

- 若您尚未购买公网域名，请先**购买公网域名**。
- 若已购买，请执行**步骤2**。

步骤2 公网域名备案。

- 若您的公网域名尚未备案，请前往**华为云备案中心**处备案。
- 若您已经备案，请执行**步骤3**。

步骤3 公网域名解析。

- 若您尚未创建公网域名解析，请**创建公网域名解析**。
- 若您已经创建，请执行**步骤4**。

步骤4 配置域名。

选择已完成以上步骤的域名，单击“配置”。

----结束

创建域名访问

根据创建无状态工作负载完成的提示页面，您可以单击“添加服务”，添加LoadBalancer类型的服务，使应用工作负载可以对外提供服务。在LoadBalancer类型的服务创建完成的提示页面中单击“创建域名访问”。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏选择“域名访问”，单击“创建域名访问”。

步骤4 设置“关联服务”参数。

- 命名空间：选择命名空间。
- 目标服务：选择目标服务，若您还没有可以关联的LoadBalancer类型的服务，请先创建服务。详细创建服务步骤见**负载均衡（LoadBalancer）**。

步骤5 单击“下一步”，配置访问模式。

- 主备模式：流量只会解析到您所选择的主集群中，可以通过修改流量配比功能，修改主备集群。
- 自适应模式：流量解析根据各集群后端实例数量自动分配权重。并且可以配置地域亲和，设置特定区域的用户流量访问特定的集群。
- 自定义模式：您可以自定义配置域名解析到每个集群的权重。并且可以配置地域亲和，设置特定区域的用户流量访问特定的集群。

步骤6 单击“创建”完成域名访问任务创建，完成该任务需要一段时间，请耐心等待。可单击“返回域名访问列表”或“查看域名访问详情”查看创建的域名访问。

----结束

修改别名

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 选择“域名访问”，单击访问名称，进入域名访问详情界面。

步骤4 单击  填写别名，填写完成后单击  。

----结束

修改流量比例

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 选择“域名访问”，单击访问名称，进入域名访问详情界面。

步骤4 在“拓扑图”页签下，单击“编辑”。

步骤5 修改完成后，单击“确定”。

----结束

查看域名访问地址

域名访问创建完成后，可以在域名访问列表中，查看域名访问地址。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 选择“域名访问”，在域名访问列表中“域名地址”即为域名访问地址。

----结束

删除域名访问

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 选择“域名访问”，在右侧域名访问列表最右侧“操作”一列，单击“删除”。

步骤4 在“删除域名访问”页面中单击“是”完成删除操作。

----结束

3.11 容器存储

3.11.1 存储概述

创建工作负载时，配置容器存储，可以使用以下类型的存储。

本地存储

通过本地磁盘存储将容器所在宿主机的文件目录挂载到容器的指定路径中（对应Kubernetes的HostPath），也可以不填写源路径（对应Kubernetes的EmptyDir），不填写时将分配主机的临时目录挂载到容器的挂载点，指定源路径的本地硬盘数据卷适用于将数据持久化存储到容器所在宿主机，EmptyDir（不填写源路径）适用于容器的临时存储。配置项（ConfigMap）是一种用于存储工作负载所需配置信息的资源类型，内容由用户决定。密钥（Secret）是一种用于存储工作负载所需要认证信息、密钥的敏感信息等的资源类型，内容由用户决定。详情请参见**挂载本地存储**。

存储卷声明

UCS支持创建持久化存储并挂载到容器的某一路径下，当容器迁移时，云存储会在新容器中重新被挂载，从而保证数据的可靠性，详情请参见**挂载存储卷**。因此建议在创建工作负载时选择挂载存储卷声明，将Pod数据存储在对应的云存储上。若存储在本地磁盘上，当节点异常无法恢复时，本地磁盘中的数据也将无法恢复。

- 华为云集群：UCS支持自动创建云硬盘（EVS）、对象存储（OBS）、文件存储（SFS）三种不同类型的云存储并挂载到华为云集群的容器路径下。
  - 云硬盘（EVS）：云硬盘可以为容器提供高可靠、高性能、规格丰富并且可弹性扩展的块存储服务，这种存储方式存放的是二进制数据，无法直接存放文件，适用于需要永久化保存的数据。
  - 文件存储（SFS）：提供按需扩展的高性能文件存储（NAS），可为容器提供文件共享访问，用于需要多读多写的文件持久化存储，适用于多种工作负载场景，包括媒体处理、内容管理和Web服务、大数据和分析应用程序等场景。
  - 对象存储（OBS）：对象存储没有总数据容量和对象/文件数量的限制，为用户提供了超大存储容量的能力，适合存放任意类型的文件，可用于海量数据存储分析、历史数据明细查询、海量行为日志分析和公共事务分析统计等场景。
- 非华为云集群：非华为云集群在使用存储卷声明挂载云存储时，需要集群提供商支持存储类功能，详情请参见**存储类**。

3.11.2 挂载本地存储

本地磁盘应用场景

使用本地磁盘有四种形式：

- 主机路径（HostPath）：将容器所在宿主机的文件目录挂载到容器指定的挂载点中，如容器需要访问宿主机的/etc/hosts，则可以使用HostPath将宿主机的/etc/hosts路径映射至容器路径。
- 临时路径（EmptyDir）：适用于临时存储、灾难恢复、共享运行时数据等场景，工作负载实例的删除或者迁移会导致临时路径被删除。生命周期与容器实例相同。容器实例消亡时，EmptyDir会被删除，数据会永久丢失。
- 配置项（ConfigMap）：将配置项挂载到容器中，然后在容器的挂载路径下就可以读取到配置项的内容。
- 密钥（Secret）：将密钥挂载到容器中，然后在容器的挂载路径下就可以读取到密钥的内容。

主机路径挂载

在容器上挂载宿主机上的文件或目录。通常用于：“容器应用程序生成的日志文件需要永久保存”或者“需要访问宿主机上Docker引擎内部数据结构的容器应用”。

步骤1 参考**创建无状态负载**、**创建有状态负载**或**创建守护进程集**，在设置容器基本信息后，选择“数据存储”，在“本地存储”页签下单击 + 添加。

图 3-37 容器存储设置

步骤2 设置添加本地磁盘参数，如表3-40。

表 3-40 主机路径挂载

参数	参数说明
存储类型	主机路径(HostPath)。
主机路径	输入主机路径，如/etc/hosts。
挂载路径	输入数据卷挂载到容器上的路径。 **须知** • 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。 • 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume。不填写时默认为根。
权限	• 只读：只能读容器路径中的数据卷。 • 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

步骤3 重复添加可增加多条设置，单击“确认”完成配置。

----结束

临时路径挂载

适用于临时存储、灾难恢复、共享运行时数据等场景，工作负载实例的删除或者迁移会导致临时路径被删除。

步骤1 参考创建无状态负载、创建有状态负载或创建守护进程集，在设置容器基本信息后，选择“数据存储”，在“本地存储”页签下单击 + 添加。

图 3-38 容器存储设置

步骤2 设置添加本地磁盘参数，如表3-41。

表 3-41 临时路径挂载

参数	参数说明
存储类型	临时路径(EmptyDir)。
磁盘介质	• 默认：即存储在硬盘上，适用于数据量大，读写效率要求低的场景。 • 内存：可以提高运行速度，但存储容量受内存大小限制。适用于数据量少，读写效率要求高的场景。
挂载路径	输入数据卷挂载到容器上的路径。 **须知** • 请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。 • 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume。不填写时默认为根。
权限	• 只读：只能读容器路径中的数据卷。 • 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

步骤3 重复添加可增加多条设置，单击“确认”完成配置。

----结束

配置项挂载

“配置项挂载”用于处理工作负载配置参数。用户需要提前创建工作负载配置，操作步骤请参见**配置项（ConfigMap）**。

步骤1 参考**创建无状态负载**、**创建有状态负载**或**创建守护进程集**，在设置容器基本信息后，选择“数据存储”，在“本地存储”页签下单击 ┼ 添加。

图 3-39 容器存储设置

步骤2 设置添加本地磁盘参数，如**表3-42**。

表 3-42 配置项挂载

参数	参数说明
存储类型	配置项(ConfigMap)。
配置项	选择对应的配置项名称。 **说明** 配置项需要提前创建，详情请参见**配置项（ConfigMap）**。
挂载路径	输入数据卷挂载到容器上的路径。 **须知** • 请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。 • 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume。不填写时默认为根。
权限	仅支持“只读”，只能读容器路径中的数据卷。

步骤3 重复添加可增加多条设置，单击“确认”完成配置。

----结束

密钥挂载

将密钥中的数据挂载到指定的容器中，密钥内容由用户决定。用户需要提前创建密钥，操作步骤请参见**密钥（Secret）**。

步骤1 参考**创建无状态负载**、**创建有状态负载**或**创建守护进程集**，在设置容器基本信息后，选择“数据存储”，在“本地存储”页签下单击 ┼ 添加。

图 3-40 容器存储设置

步骤2 设置添加本地磁盘参数，如**表3-43**。

表 3-43 密钥挂载

参数	参数说明
存储类型	密钥(Secret)。

参数	参数说明
密钥	选择对应的密钥名称。 **说明** 密钥需要提前创建，详情请参见**密钥（Secret）**。
挂载路径	输入数据卷挂载到容器上的路径。 **须知** • 请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。 • 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。
子路径	使用子路径挂载本地磁盘，实现在单一Pod中重复使用同一个Volume。不填写时默认为根。
权限	仅支持“只读”，只能读容器路径中的数据卷。

步骤3 重复添加可增加多条设置，单击“确认”完成配置。

----结束

3.11.3 挂载存储卷

存储卷声明（PVC）提供容器的持久存储管理能力，提供多云场景的统一的容器存储管理，支持将云存储按需挂载到容器，保障应用的高可靠性。

须知

- 通过UCS控制台创建存储卷声明（PVC）后，系统将自动为您在部署集群中创建一个同名的PVC，并同时创建与该PVC绑定的存储卷（PV）及其对应的存储资源。如您对Kubernetes中存储卷、存储卷声明及存储资源之间的关系不够了解，请参见**持久化存储**。
- 您可以在集群中对UCS自动创建的存储卷声明进行修改或删除。但如果不同步修改UCS中的存储卷声明设置，最终已修改或删除的存储卷声明会被UCS重建。因此建议您直接通过UCS控制台修改。
- 非华为云集群使用UCS挂载存储卷声明时，需要集群提供商具备存储类（StorageClass）功能，以实现存储卷的动态创建。请通过下列命令查询对应集群的StorageClass配置及对接的后端存储资源。更多StorageClass相关内容，请参见**存储类**。

```
kubectl get storageclass
```

使用存储卷声明挂载云存储

步骤1 参考**创建无状态负载**、**创建有状态负载**或**创建守护进程集**，在设置容器基本信息后，选择“数据存储”，在“存储卷声明PVC”页签下单击 + 添加。

图 3-41 容器存储设置

步骤2 选择需要添加的存储卷声明。如没有可选的存储卷声明，可单击“创建存储卷声明”进行添加，相关参数请参见**创建存储卷声明**。参数填写完成后，单击“确认”。

步骤3 填写容器挂载参数。

- 挂载路径：输入数据卷挂载到容器上的路径。

  须知

  - 请不要挂载在系统目录下，如“/”、“**/var/run**”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，工作负载创建失败。
  - 挂载高危目录的情况下 ，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

- 子路径：Kubernetes中数据卷挂载的subPath，指引用卷内的子路径而不是其根路径。该参数不填写时，默认挂载至根路径。
- 设置权限。
  - 只读：只能读容器路径中的数据卷。
  - 读写：可修改容器路径中的数据卷，容器迁移时新写入的数据不会随之迁移，会造成数据丢失。

步骤4 重复添加可增加多条存储卷声明设置。

----结束

3.11.4 创建存储卷声明

须知

- 通过UCS控制台创建存储卷声明（PVC）后，系统将自动为您在部署集群中创建一个同名的PVC，并同时创建与该PVC绑定的存储卷（PV）及其对应的存储资源。如您对Kubernetes中存储卷、存储卷声明及存储资源之间的关系不够了解，请参见**持久化存储**。
- 您可以在集群中对UCS自动创建的存储卷声明进行修改或删除。但如果不同步修改UCS中的存储卷声明设置，最终已修改或删除的存储卷声明会被UCS重建。因此建议您直接通过UCS控制台修改。
- 非华为云集群使用UCS挂载存储卷声明时，需要集群提供商具备存储类（StorageClass）功能，以实现存储卷的动态创建。请通过下列命令查询对应集群的StorageClass配置及对接的后端存储资源。更多StorageClass相关内容，请参见**存储类**。

  ```
  kubectl get storageclass
  ```

创建存储卷声明

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

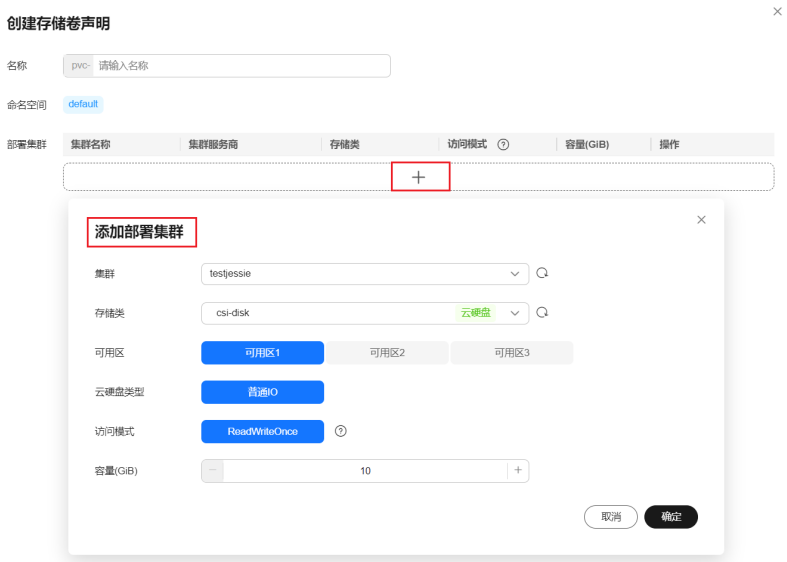
步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“容器存储”，在“存储卷声明PVC”页签下单击右上角“创建存储卷声明”。

步骤4 设置通用配置参数。

- 名称：新建存储卷声明名称，命名必须唯一。
- 命名空间：新建存储卷声明所属的命名空间，默认配置为default。
- 部署集群：单击＋，选择存储要部署的集群。

图 3-42 添加部署集群



  - 华为云集群：添加部署集群参数说明请参见表3-44。
  - 非华为云集群：添加部署集群参数说明请参见表3-45。

表 3-44 添加部署华为云集群

参数	参数说明
集群	选择部署集群为华为云集群。

参数	参数说明
存储类	• **csi-disk**：即云硬盘EVS，需指定存储可用区和云硬盘类型。 – 可用区：云硬盘所在的可用区，不同可用区下支持的云硬盘类型可能存在差异。 – 云硬盘类型：可选择普通IO、高IO或超高IO，不同IO类型对应的存储类型依次为SATA、SAS、SSD。 • **csi-nas**：即文件存储SFS。 • **csi-obs**：即对象存储OBS，需指定实例类型和对象存储类型，并添加访问密钥。 – 实例类型：对象桶或并行文件系统。并行文件系统是OBS经过优化的高性能文件系统，提供更高性能的对象访问。 – 对象存储类型：标准存储和低频访问存储。低频访问存储是高可靠、较低成本的实时访问存储服务，适用于不频繁访问（平均一年少于12次）的场景，例如文件同步/共享、企业备份。
访问模式	• 选择**csi-disk**（云硬盘）时，访问模式为ReadWriteOnce (RWO)，即存储卷只能以读写模式被单个节点同时加载。 • 选择**csi-nas**（文件存储）或**csi-obs**（对象存储）时，访问模式为ReadWriteMany (RWX)，即存储卷能够以读写模式被多个节点同时加载。
容量（GiB）	新建存储的容量，容量不能小于10GiB。 仅选择**csi-disk**（云硬盘）和**csi-nas**（文件存储）时可设置；选择**csi-obs**（对象存储）时为按需使用，无需设置容量。

表 3-45 添加部署非华为云集群

参数	参数说明
集群	选择部署集群为非华为云集群。
存储类	集群支持的存储类（StorageClass）取决于注册集群的实际环境，相关内容请参见**存储类**。
访问模式	• ReadWriteOnce (RWO)：即存储卷只能以读写模式被单个节点同时加载。 • ReadWriteMany (RWX)：即存储卷能够以读写模式被多个节点同时加载。
容量（GiB）	新建存储的容量，容量不能小于10GiB。
注解	输入键、值后单击“添加”，对应的注解将以Key/value键值对的形式附加到存储卷声明对象上。

步骤5 重复添加，可对多个集群进行差异化的存储卷声明设置。

步骤6 单击“确认”创建成功后，可单击PVC名称查看存储卷声明详情。

----结束

相关操作

通过UCS控制台，您还可以执行表3-46中的操作。

表 3-46 相关操作

操作	说明
YAML创建	单击右上角“YAML创建”，可使用已有的YAML创建服务。
查看详情	1. 选择PVC所在的命名空间。 2. （可选）根据PVC名称进行搜索。 3. 单击PVC名称即可查看存储卷声明详情，包括基本信息以及各集群的部署信息。 4. 在存储卷声明详情页的部署集群栏中单击“查看YAML”，可查看各个集群中部署的PVC实例YAML，并支持下载。
查看YAML	单击PVC名称后的“查看YAML”，可查看当前PVC的YAML文件。
更新（存储卷声明PVC扩容）	1. 单击PVC名称后的“更新”。 2. 根据PVC参数更改部署集群参数，或单击“扩容”调整存储卷容量大小。 3. 单击“确认”提交已修改的信息。
删除	单击PVC名称后的“删除”，并单击“是”进行确认。
批量删除	1. 勾选需要删除的PVC。 2. 单击左上角的“批量删除”。 3. 单击“是”进行确认。

3.12 命名空间

命名空间（Namespace）是对集群中一组资源和对象的抽象整合，可通过集群资源配额实现多个用户之间的资源划分，适用于多个团队或项目共享一个集群资源的场景。

创建命名空间

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“命名空间”，单击右上角“创建命名空间”。

步骤4 参照表3-47设置命名空间参数。

表 3-47 命名空间基本信息

参数	参数说明
命名空间名称	新建命名空间的名称，命名必须唯一。
标签	Key/value键值对格式，给命名空间添加自定义标签，定义不同属性，通过这些标签了解各个命名空间的特点。
注解	Key/value键值对格式，给命名空间添加自定义注解。
描述	输入对命名空间的描述信息。

步骤5 配置完成后，单击“确定”。

创建完成后，可单击“查看YAML”查看YAML文件，并支持下载。

----结束

使用命名空间

在创建工作负载、Service、Ingress、存储声明等场景时都会用到命名空间，以创建工作负载为例：

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“工作负载”，在“无状态负载”页签中单击右上角“镜像创建”。

步骤4 设置工作负载基本信息，选择工作负载所在的命名空间。

步骤5 继续完成工作负载其他配置的填写并创建。

----结束

删除命名空间

须知

- 在UCS控制台删除命名空间将会级联删除各个集群中的同名命名空间，并删除该命名空间相关的所有数据资源，请谨慎操作。
- 为保证UCS的正常运行，来源为“系统”或“默认”的命名空间无法被删除。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在“容器舰队”页签下找到已开通集群联邦的舰队，单击名称进入详情页。

步骤3 在左侧导航栏中选择“命名空间”，在命名空间列表中，选择需要删除的命名空间，单击“删除”。

如需同时删除多个命名空间，可勾选多个命名空间并单击“批量删除”。

步骤4 根据提示，单击“是”进行删除操作。

----结束

3.13 多集群负载伸缩

3.13.1 负载伸缩概述

为什么需要负载伸缩

由于企业应用流量的不断变化，容器工作负载的资源需求也在不断变化。在部署、管理容器工作负载时，若时刻保持业务高峰期的资源数量，会造成大量的资源浪费；若为工作负载设置资源限制，则达到资源使用上限后可能会造成应用异常。Kubernetes中的**HPA**（Horizontal Pod Autoscaler）策略可基于监控资源指标变动实现单集群工作负载自动扩缩，暂不适用于多集群工作负载。

UCS为您提供多集群工作负载的自动扩缩能力。UCS负载伸缩能力可基于工作负载的系统指标变动、自定义指标变动或固定的时间周期对工作负载进行自动扩缩，以提升多集群工作负载的可用性和稳定性。

UCS 负载伸缩的优势

UCS负载伸缩能力的优势主要在于：

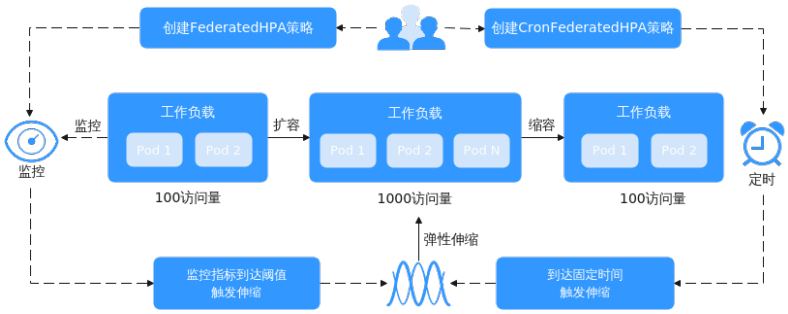
- 多集群：多集群场景下的负载伸缩，可以对集群联邦中的多个集群实行统一的负载伸缩策略。
- 高可用：在业务高峰期快速扩容以保证工作负载的可用性，在业务平缓期快速缩容以节约资源成本。
- 多功能：支持基于系统指标变动、自定义指标变动和固定时间周期进行负载伸缩，实现复杂场景下的负载伸缩。
- 多场景：使用场景广泛，典型的场景包含在线业务弹性、大规模计算训练、深度学习GPU或共享GPU的训练与推理。

负载伸缩实现机制

UCS的负载伸缩能力是由FederatedHPA和CronFederatedHPA两种负载伸缩策略所实现的，如**图3-43**所示。

- 创建FederatedHPA策略，支持基于系统指标与自定义指标对工作负载进行扩缩。指标到达所配置的期望值时，触发工作负载扩缩。
- 创建CronFederatedHPA策略，支持基于固定时间周期对工作负载进行扩缩。到达所配置的触发时间时，触发工作负载扩缩。

图 3-43 负载伸缩策略机制



约束与限制

- UCS负载伸缩策略只能作用于无状态工作负载。若您需要了解不同类型工作负载的区别，请参见工作负载。
- UCS负载伸缩策略只专注于扩缩工作负载中的Pod数量，若您需要将扩展出的Pod调度至特定集群，请按需配置工作负载的调度策略，具体操作请参见调度策略。

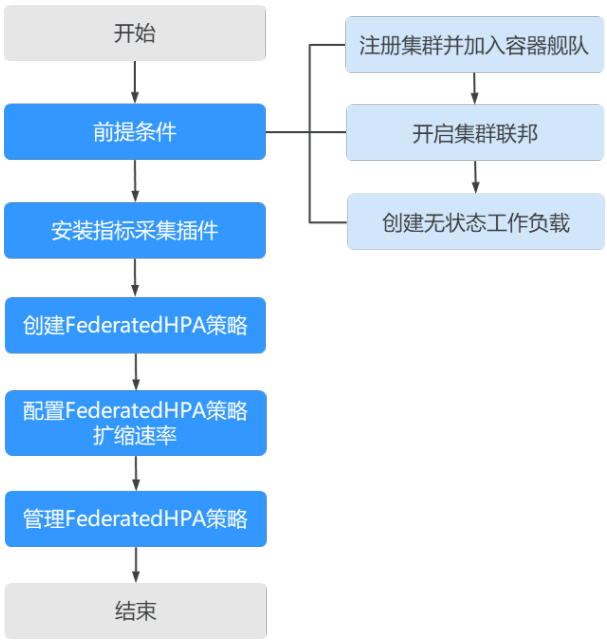
3.13.2 负载伸缩使用流程

本小节介绍FederatedHPA和CronFederatedHPA策略的使用流程。

FederatedHPA 策略使用流程

FederatedHPA策略的使用流程如图3-44所示，具体流程如下：

图 3-44 FederatedHPA 使用流程



1. 负载伸缩能力基于部署在多集群上的工作负载，因此在创建负载伸缩策略前，您需要添加集群至容器舰队、为舰队开启集群联邦能力，并创建无状态工作负载。具体操作请参见注册集群、开启集群联邦和创建工作负载。
2. 为集群安装支持采集指标数据的插件，具体操作请参见**安装指标采集插件**。

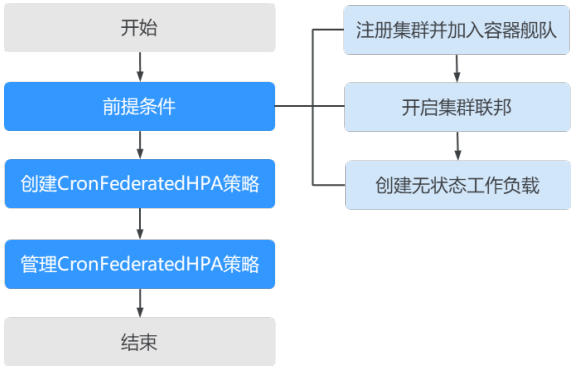
3. 创建FederatedHPA策略，具体操作请参见**创建FederatedHPA策略以按指标扩缩工作负载**。
4. 若您需要配置更加精确的工作负载扩缩速率，请参见**配置FederatedHPA策略以控制扩缩速率**。
5. 若您需要修改或删除已经创建的策略，请参见**管理FederatedHPA策略**。

CronFederatedHPA 策略使用流程

单独使用CronFederatedHPA策略

单独使用CronFederatedHPA策略时，使用流程如**图3-45**所示，具体流程如下：

图 3-45 CronFederatedHPA 单独使用流程

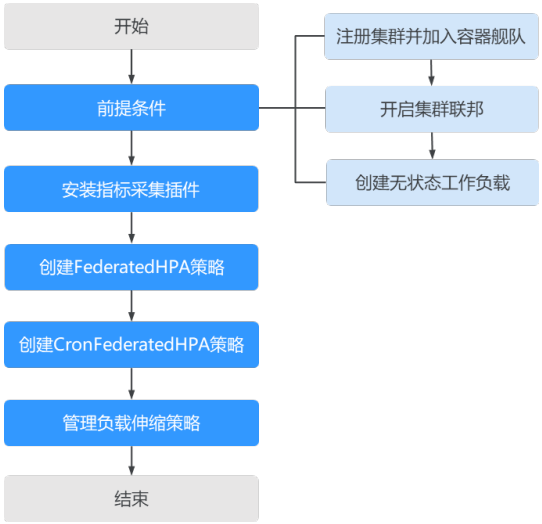


1. 负载伸缩能力基于部署在多集群上的工作负载，因此在创建负载伸缩策略前，您需要添加集群至容器舰队、为舰队开启集群联邦能力，并创建无状态工作负载。具体操作请参见注册集群、开启集群联邦和创建工作负载。
2. 创建CronFederatedHPA策略，具体操作请参见**创建CronFederatedHPA策略以定时扩缩工作负载**。
3. 若您需要修改或删除已经创建的策略，具体操作请参见**管理CronFederatedHPA策略**。

配合使用CronFederatedHPA策略与FederatedHPA策略

配合使用CronFederatedHPA策略与FederatedHPA策略时，使用流程如**图3-46**所示，具体流程如下：

图 3-46 CronFederatedHPA 与 FederatedHPA 配合使用流程



1. 负载伸缩能力基于部署在多集群上的工作负载，因此在创建负载伸缩策略前，您需要添加集群至容器舰队、为舰队开启集群联邦能力，并创建无状态工作负载。具体操作请参见注册集群、开启集群联邦和创建工作负载。
2. 为集群安装支持采集指标数据的插件，具体操作请参见**安装指标采集插件**。
3. 创建FederatedHPA策略，具体操作请参见**创建FederatedHPA策略以按指标扩缩工作负载**。
4. 创建CronFederatedHPA策略，具体操作请参见**创建CronFederatedHPA策略以定时扩缩工作负载**。
5. 若您需要修改或删除已经创建的策略，具体操作请参见**管理FederatedHPA策略**和**管理CronFederatedHPA策略**。

3.13.3 FederatedHPA 策略

3.13.3.1 FederatedHPA 工作原理

通过配置FederatedHPA策略，您可以基于工作负载的系统指标（CPU利用率、内存利用率）或自定义指标，对部署在多个集群中的无状态工作负载进行自动扩缩容。

您可以配合使用FederatedHPA策略与调度策略来实现各种功能，例如在FederatedHPA策略扩展工作负载Pod数量后，配置调度策略将扩展出的Pod调度到具有更多资源的集群，以解决单个集群的资源限制，提高故障发生时的恢复能力。

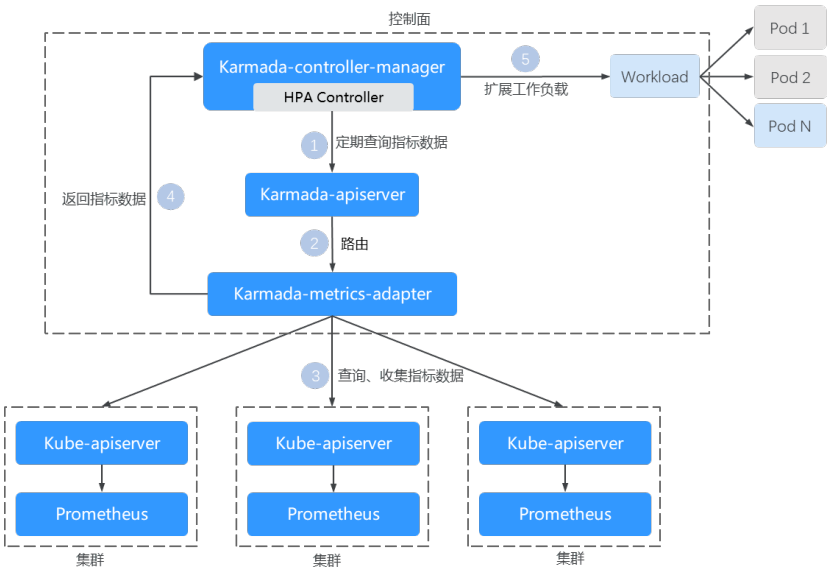
FederatedHPA 工作原理

FederatedHPA的工作原理如**图3-47**，实现流程如下：

1. HPA Controller通过API定期查询工作负载的指标数据。
2. karmada-apiserver收到查询请求，会路由到之前通过API服务注册的karmada-metrics-adapter。
3. karmada-metrics-adapter收到查询请求，会查询并收集集群中工作负载的指标数据。
4. karmada-metrics-adapter将**计算的指标数据**返回至HPA Controller。

5. HPA Controller基于返回的指标数据**计算所需的Pod扩缩数量**，并保持**负载伸缩的稳定性**。

图 3-47 FederatedHPA 工作原理



如何计算指标数据?

指标数据分为系统指标与自定义指标，计算方法如下：

- 系统指标

  主要包括CPU利用率和内存利用率两个指标，系统指标的查询与监控依赖Metrics API。例如，您希望控制工作负载对CPU资源的利用率在合理水平，可基于CPU利用率指标为其创建FederatedHPA策略。

  说明

  利用率 = 工作负载Pod的实际资源使用量 / 资源申请量

- 自定义指标

  主要提供Kubernetes Object相关的自定义监控指标，自定义指标的查询与监控依赖Custom Metrics API。例如，您可以基于每秒请求量、每秒写入次数等其他适合工作负载的自定义指标为其创建FederatedHPA策略。

若您在创建策略时设置了多个指标期望值，HPA Controller会比较每种指标变动计算出的Pod扩缩数量，选取Pod变动最大的计算结果进行扩缩。

如何计算 Pod 扩缩数量?

HPA Controller基于当前指标值和期望指标值来计算扩缩比例，再依据当前Pod数与扩缩比例计算出期望Pod数。当前Pod数与期望Pod数的计算方法如下：

- 当前Pod数 = 所有集群中状态为Ready的Pod数量

在计算期望Pod数时，HPA Controller会选择最近5分钟内计算所得的Pod数的最大值，以避免之前的自动扩缩操作还未完成，就直接执行新的扩缩的情况。

- 期望Pod数 = 当前Pod数 *（ 当前指标值 / 期望指标值 ）

例如，当以CPU利用率为扩缩容参考指标时，若当前指标值为100%，期望指标值为50%，那么按照公式计算出的期望Pod数即为当前Pod数的两倍。

如何保证负载伸缩的稳定性?

为了保证负载伸缩的稳定性，HPA controller设置了以下功能：

- 稳定窗口

  在监控到指标数据达到期待值（即满足伸缩标准）时，HPA controller会在所设定的稳定窗口期内持续检测，如果检测结果显示该时间段内的指标数据持续达到期待值，才会进行伸缩。默认扩容稳定窗口时长为0秒，缩容稳定窗口时长为300秒，支持修改。在实际配置过程中，为避免服务抖动，稳定窗口的配置的原则是快速扩容，低速缩容。

- 容忍度

  容忍度 = abs（当前指标值 / 期望指标值 -1）

  其中abs为绝对值。当指标值的变动在设定的容忍度范围之内时，不会触发工作负载的弹性伸缩。UCS负载伸缩策略默认伸缩容忍度为0.1，不支持修改。

例如，若您在创建策略时选择默认设置，则缩容会在指标数据达到期待值的1.1倍以上，且持续时间超过300秒时触发；扩容会在指标数据达到期待值的0.9倍以下，且持续时间超过0秒时触发。

3.13.3.2 安装指标采集插件

在创建FederatedHPA策略前，您需要为集群安装支持Metrics API的插件，以采集工作负载相关指标的变动。如果您已经安装了相应插件，可跳过该步骤。

选择插件

UCS提供两种插件以采集工作负载相关指标：Kubernetes Metrics Server与kube-prometheus-stack。两种插件适用的集群类型与指标类型不同，请参考**表3-48**选择插件进行安装。

表 3-48 插件选择

适用的集群类型	支持的指标类型	插件	注意事项
华为云集群	系统指标	安装Kubernetes Metrics Server或kube-prometheus-stack。	若您选择安装kube-prometheus-stack插件，在安装该插件后，需要将Prometheus注册为Metrics API的服务，具体操作请参见**通过Metrics API提供资源指标**。

适用的集群类型	支持的指标类型	插件	注意事项
	自定义指标	安装kube-prometheus-stack。	• 安装插件前，请检查您的华为云集群版本，若为v1.19以下，请先升级集群版本。 • 安装插件时，必须选择Server模式，该模式支持自定义指标。 • 安装插件后，需要将自定义指标聚合至Kubernetes API Server，具体操作请参见**将自定义指标聚合到Kubernetes API Server**。
非华为云集群	系统指标	安装Kubernetes Metrics Server。	参见**安装插件**相关文档。
	自定义指标	暂无插件支持。	非华为云集群的自定义指标采集需要自行安装**Prometheus Adapter**组件并配置自定义指标采集规则，再进行FederatedHPA策略的创建。

安装插件

选择适用的插件后，请结合**表3-48**内注意事项，参考相关文档为集群安装插件：

⚠ 注意

请为需要创建负载伸缩策略的集群联邦下所有集群安装指标采集插件，否则会造成指标采集异常，负载伸缩策略失效。

- Kubernetes Metrics Server的安装方法请参见**Kubernetes Metrics Server**。
- kube-prometheus-stack的安装方法请参见**kube-prometheus-stack**。

3.13.3.3 创建 FederatedHPA 策略以按指标扩缩工作负载

本小节将指导您创建FederatedHPA策略，以基于不同指标自动扩缩工作负载。

在创建FederatedHPA策略前，您可以参考**FederatedHPA工作原理**了解相关原理与概念。若您需要了解不同负载伸缩策略的差异，请参考**负载伸缩概述**。

约束与限制

FederatedHPA策略仅支持1.19及更高版本的集群创建。若您需要查询集群版本，请登录UCS控制台，单击需查询集群所在舰队名称，再单击“容器集群”即可。

创建 FederatedHPA 策略

控制台创建

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击已开通集群联邦的容器舰队名称，进入容器舰队详情页面。

步骤3 在左侧导航栏选择“负载伸缩”，在“指标伸缩策略”页签下，单击右上角的“创建指标伸缩策略”。

步骤4 配置FederatedHPA策略参数。

表 3-49 FederatedHPA 策略参数配置

参数	参数说明
策略名称	FederatedHPA策略的名称。输入长度范围为4-63个字符。
命名空间	请选择需要自动扩缩的工作负载所在命名空间的名称。也可新建命名空间，具体操作请参见命名空间。
生效工作负载	请选择需要设置自动扩缩的工作负载的名称。也可新建工作负载，具体操作请参见工作负载。
实例范围	触发策略时，工作负载内Pod数量所能达到的最大值与最小值。 • 最小值：请输入1-299之间的正整数。 • 最大值：请输入1-1500之间的正整数，且填写值需大于实例范围最小值。
稳定窗口时长	稳定窗口时长内指标数据持续达到期待值，才会进行扩缩。默认扩容稳定窗口时长为0秒，缩容稳定窗口时长为300秒。稳定窗口时长的详细信息请参见**如何保证负载伸缩的稳定性？**。 • 扩容：请输入0-3600之间的正整数，单位为秒。 • 缩容：请输入0-3600之间的正整数，单位为秒。
系统规则	若您需要基于系统指标对工作负载进行扩缩，则需配置该规则。 • 指标：可选择“CPU利用率”或“内存利用率”。 • 期待值：指标数据达到期待值时，触发扩缩。
自定义规则	若您需要基于自定义指标对工作负载进行扩缩，则需配置该规则。 • 自定义指标名称：在下拉框中选择自定义指标的名称。 • 指标来源：在下拉框中选择自定义指标所描述的对象类型，目前仅支持“Pod”。 • 期待值：指标数据达到期待值时，触发扩缩。 **注意** • 自定义规则仅支持1.19及更高版本的集群创建。 • 使用自定义规则时，集群中需要安装支持采集自定义指标的插件，且工作负载需正常上报并采集自定义指标，详情请参见**安装指标采集插件**。

步骤5 参数配置完成后，单击右下角“创建”，即可跳转指标伸缩策略列表查看策略详情，完成FederatedHPA策略创建。

----结束

命令行创建

步骤1 使用kubectl连接集群联邦，具体操作请参见使用kubectl连接集群联邦。

步骤2 创建并编辑fhpa.yaml文件，文件内容定义如下所示，关键参数定义请参见**表3-50**。

vi fhpa.yaml

本示例创建的FederatedHPA策略名称为hpa-example-hpa，作用于名称为hpa-example的工作负载，稳定窗口时长为扩容0秒、缩容300秒，最大Pod数为100、最小Pod数为2，包含两条系统指标规则（名称为“memory”和“cpu”）。其中，memory规则中内存利用率的期望值为50%，cpu规则中CPU利用率的期望值为60%。

```
apiVersion: autoscaling.karmada.io/v1alpha1
kind: FederatedHPA
metadata:
  name: hpa-example-hpa                       # FederatedHPA策略名称
  namespace: default                          # 工作负载所在命名空间名称
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hpa-example                         # 工作负载名称
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300         # 缩容的稳定窗口时长为300秒
    scaleUp:
      stabilizationWindowSeconds: 0           # 扩容的稳定窗口时长为0秒
  minReplicas: 2                              # 最小Pod数为2
  maxReplicas: 100                            # 最大Pod数为100
  metrics:
    - type: Resource
      resource:
        name: memory                          # 第一条规则名称
        target:
          type: Utilization                   # 指标类型为利用率
          averageUtilization: 50              # 期望的平均利用率
    - type: Resource
      resource:
        name: cpu                             # 第二条规则名称
        target:
          type: Utilization                   # 指标类型为利用率
          averageUtilization: 60              # 期望的平均利用率
```

表 3-50 关键参数说明

参数	是否必填	参数类型	描述
stabilizationWindowSeconds	否	String	缩容的稳定窗口时长，请输入0-3600之间的正整数，单位为秒，不定义时默认为300秒。 **说明** 稳定窗口时长内指标数据持续达到期待值，才会进行扩缩。稳定窗口时长的详细信息请参见**如何保证负载伸缩的稳定性？**。
stabilizationWindowSeconds	否	String	扩容的稳定窗口时长，请输入0-3600之间的正整数，单位为秒，不定义时默认为0秒。
minReplicas	是	String	触发策略时，工作负载内Pod数量所能缩容到的最小值，请输入1-299之间的正整数。

参数	是否必填	参数类型	描述
maxReplicas	是	String	触发策略时，工作负载内Pod数量所能扩容到的最大值，请输入1-1500之间的正整数，且输入值需大于实例范围最小值。
name	是	String	规则名称。基于系统指标进行弹性伸缩时，基于内存利用率的规则名称为memory，基于CPU利用率的规则名称为cpu。
type	是	String	指标类型。 ● Value：总量 ● AverageValue：平均量 = 总量 / 当前Pod数 ● Utilization：利用率 = 平均量 / 申请量
averageUtilization	是	String	指标数据达到期待值时，触发扩缩。

步骤3 执行如下命令创建FederatedHPA策略。

kubectl apply -f fhpa.yaml

回显如下表明创建成功。

```
FederatedHPA.autoscaling.karmada.io/hpa-example-hpa created
```

创建完成后，可以执行如下命令观察负载伸缩的运行效果。

- 检查工作负载的当前Pod数：kubectl get deployments
- 查看FederatedHPA策略事件（仅保留最近三条）：kubectl describe federatedhpa hpa-example-hpa

可以执行如下命令管理FederatedHPA策略。其中hpa-example-hpa为FederatedHPA策略的名称，实际情况中需修改为自己所创建的策略名称。

- 获取FederatedHPA策略：kubectl get federatedhpa hpa-example-hpa
- 更新FederatedHPA策略：kubectl edit federatedhpa hpa-example-hpa
- 删除FederatedHPA策略：kubectl delete federatedhpa hpa-example-hpa

----结束

3.13.3.4 配置 FederatedHPA 策略以控制扩缩速率

为什么需要控制扩缩速率

HPA controller默认的扩缩容总原则是：快速扩容，低速缩容。然而，若仅依靠配置稳定窗口时长，在窗口时长过后即失去了对扩缩容速率的控制能力，无法真正实现对扩缩容速率的精准控制。您可以通过配置负载伸缩策略的YAML文件中spec的behavior结构，来更精准灵活地控制FederatedHPA的自动扩缩速度。该结构支持为每个FederatedHPA策略独立配置扩缩容速率，以及为扩容与缩容配置不同的速率。

操作步骤

针对不同的业务场景，本小节提供常见的behavior结构配置方法。在其他业务场景下，如您希望缓慢扩容、快速缩容等等，可以参考下述对behavior结构的解释，针对scaleUp与scaleDown进行配置。

- 场景一：尽快扩容

  如果您希望在业务高峰期能尽快扩容，可以配置较大的Percent值。

  ```
  behavior:
    scaleUp:
      policies:
      - type: Percent
        value: 900
        periodSeconds: 60
  ```

  本示例中Percent值为900，也就是说，每个扩缩周期都按照（1 + 900%）= 10倍的速率进行扩容。例如，若工作负载中Pod数从1开始，那么在上述配置下，每隔60秒的扩容Pod数的变化如下：1 > 10 > 100 > ... 。需要注意的是，扩缩后的Pod数量不能超过FedratedHPA策略配置的最大Pod数。

  Percent类型对资源消耗的波动较大，如果您希望资源消耗可控，可以按绝对数—Pods类型来配置。

  ```
  behavior:
    scaleDown:
      policies:
      - type: Pods
        value: 10
        periodSeconds: 60
  ```

  本示例中Pods值为10，也就是说，每个扩缩周期都按照增加10个Pod的增量进行扩容。例如，若工作负载中Pod数从1开始，那么在上述配置下，每隔60秒的扩容Pod数的变化如下：1 > 11 > 21 > ... 。需要注意的是，扩缩后的Pod数量不能超过FedratedHPA策略配置的最大Pod数。

- 场景二：逐步缩容

  如果您希望在渡过业务高峰期后，对工作负载的缩容速率能更加缓慢，以提高应用的可靠性，可以配置较小的Pod值与较大的periodSeconds值。

  ```
  behavior:
    scaleDown:
      policies:
      - type: Pods
        value: 1
        periodSeconds: 600
  ```

  本示例中Pods值为1，periodSeconds值为600，也就是说：缩容的周期为600秒，每次缩容减少1个Pod。假如 pod 最开始数量为 100，每隔600秒的缩容Pod数的变化如下：100 > 99 > 98 > ⋯ 。极端情况下，若希望工作负载不进行自动缩容，可以将Percent值或Pods值配置为0。

- 场景三：默认速率

  若不对behavior进行配置，则FederatedHPA策略默认的配置为：

  ```
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 300
      policies:
      - type: Percent
        value: 100
        periodSeconds: 15
    scaleUp:
      stabilizationWindowSeconds: 0
      policies:
      - type: Percent
        value: 100
  ```

```
    periodSeconds: 15
  - type: Pods
    value: 4
    periodSeconds: 15
```

默认配置中，扩缩容的周期为15秒，每个扩缩周期都按照（1 + 100%）= 2倍的速率进行扩容或缩容，每次缩容的Pod数为4。

3.13.3.5 管理 FederatedHPA 策略

本小节将指导您对已经创建的FederatedHPA策略进行管理，包括编辑策略与删除策略。

⚠ 注意

若您在工作负载扩缩过程中对策略进行了修改或删除，则修改或删除后的策略会即刻生效。

编辑 FederatedHPA 策略

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击已开通集群联邦的容器舰队名称，进入容器舰队详情页面。

步骤3 在左侧导航栏选择“负载伸缩”，选择“指标伸缩策略”，在需要编辑的策略所在行右侧单击“编辑”，在策略详情页面可以修改策略配置，详细的参数说明请参见**表3-50**。

步骤4 修改完成后，单击“确定”。

----结束

删除 FederatedHPA 策略

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击已开通集群联邦的容器舰队名称，进入容器舰队详情页面。

步骤3 在左侧导航栏选择“负载伸缩”，选择“指标伸缩策略”，勾选需要删除的策略，可以在上方选择批量删除策略，也可在策略最右边单击“更多”选择“删除”，在弹框内单击“是”即可删除策略。

----结束

3.13.4 CronFederatedHPA 策略

3.13.4.1 CronFederatedHPA 工作原理

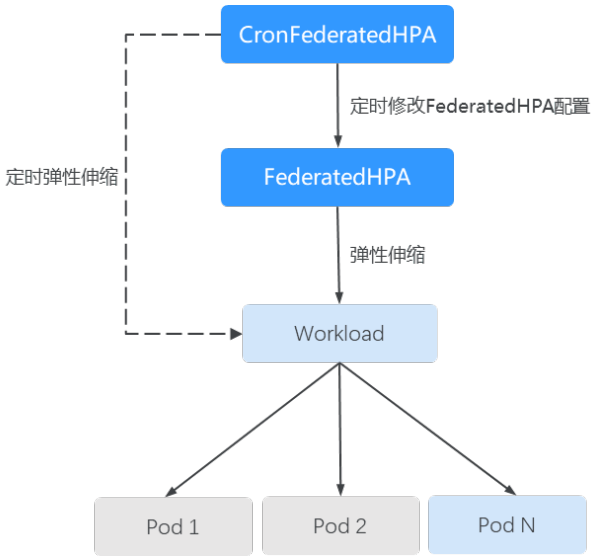
FederatedHPA是基于指标数据进行工作负载扩缩的，因此存在一定的时延。CronFederatedHPA策略基于固定的时间周期，而非指标数据对工作负载进行扩缩。

您可以对一些有周期性变化的工作负载应用CronFederatedHPA策略，在预期的业务高峰时提前扩容资源，预期的业务低谷时定时回收资源。

CronFederatedHPA 工作原理

CronFederatedHPA的工作原理如图3-48。创建CronFederatedHPA策略时，可以设定一个具体的时间，基于设定的时间调整HPA策略的最大和最小Pod数，也可以直接定时调整工作负载中的Pod数量。

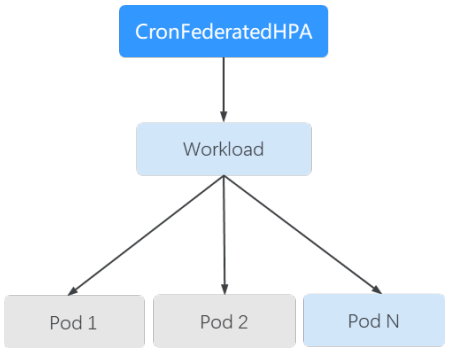
图 3-48 CronFederatedHPA 工作原理



单独使用 CronFederatedHPA

当不使用FederatedHPA策略，仅使用CronFederalHPA策略时，CronFederalHPA策略直接作用于工作负载，定时扩缩Pod数量。您可以通过设置CronFederalHPA策略的生效时间与目标Pod数，实现在固定时间段将工作负载自动扩缩至期望数量。

图 3-49 单独使用 CronFederatedHPA 策略工作原理



具体流程为：

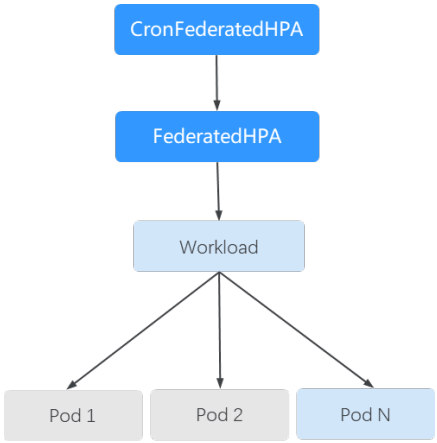
1. 创建CronFederalHPA策略，设置CronFederatedHPA的生效时间与目标Pod数。
   - 生效时间：CronFederalHPA策略会在该生效时间自动触发工作负载扩缩。
   - 目标Pod数：在到达生效时间时，所期望的Pod数。
2. 到达生效时间时，比较工作负载中的**现有Pod数**与1中设置的**目标Pod数**：目标Pod数大于现有Pod数时扩容工作负载，目标Pod数小于现有Pod数时缩容工作负载。

现有Pod数：CronFederatedHPA策略生效之前，工作负载中的Pod数量。

同时使用 CronFederatedHPA 与 FederatedHPA

当同时使用FederatedHPA策略与CronFederatedHPA策略时，CronFederatedHPA策略不再直接作用于工作负载，而是在FederatedHPA策略的基础上生效，通过定时调整FederatedHPA策略的最大和最小实例数，实现定时扩缩工作负载的作用。

图 3-50 同时使用两种策略工作原理



具体流程为：

1. 创建CronFederatedHPA策略，设置CronFederatedHPA的生效时间与目标Pod数。
   - 生效时间：CronFederatedHPA策略会在该生效时间自动触发工作负载扩缩。
   - 目标Pod数：CronFederatedHPA设定的Pod数，在CronFederatedHPA生效时用于调整FederatedHPA的最大/最小Pod数，从而间接调整工作负载中的Pod数量。
2. 到达生效时间时，比较工作负载的**现有Pod数**、FederatedHPA策略的**最大Pod数**、FederatedHPA策略的**最小Pod数**与1中设置的**目标Pod数**，根据上述四个指标的大小关系，调整FederatedHPA策略的最大和最小Pod数，FederatedHPA策略根据被调整后的最大和最小Pod数对工作负载进行扩缩。
   - 现有Pod数：CronFederatedHPA策略生效之前，工作负载中的Pod数量。
   - FederatedHPA策略的最大Pod数：工作负载的Pod数上限。
   - FederatedHPA策略的最小Pod数：工作负载的Pod数下限。

图3-51与表3-51列出了在同时使用FederatedHPA策略与CronFederatedHPA策略的情况下可能出现的扩缩容场景。您可以通过表内的示例了解在现有Pod数、最大Pod 数、最小Pod数、目标Pod数的不同大小关系下，CronFederatedHPA策略会对使用FederatedHPA策略、工作负载产生怎样的影响。

图 3-51 同时使用两种策略时的扩缩容场景

表 3-51 同时使用两种策略时的扩缩容场景示例

场景	场景说明	目标Pod数（CronFederatedHPA内设置）	现有Pod数（工作负载）	最小Pod数/最大Pod数（FederatedHPA内设置）	最终结果
场景一	**目标Pod数** < 最小Pod数 ≤ 现有Pod数 ≤ 最大Pod数	3	5	4/10	• 修改FederatedHPA的最小Pod数为3。 • 工作负载现有Pod数无修改。
场景二	**目标Pod数** = 最小Pod数 ≤ 现有Pod数 ≤ 最大Pod数	4	5	4/10	• FederatedHPA的最小Pod数无修改。 • 工作负载现有Pod数无修改。
场景三	最小Pod数 < **目标Pod数** < 现有Pod数 ≤ 最大Pod数	5	6	4/10	• 修改FederatedHPA的最小Pod数为5。 • 工作负载现有Pod数无修改。
场景四	最小Pod数 < **目标Pod数** = 现有Pod数 ≤ 最大Pod数	5	5	4/10	• 修改FederatedHPA的最小Pod数为5。 • 工作负载现有Pod数无修改。
场景五	最小Pod数 ≤ 现有Pod数 < **目标Pod数** < 最大Pod数	6	5	4/10	• 修改FederatedHPA的最小Pod数为6。 • 扩展工作负载现有Pod数至6。
场景六	最小Pod数 ≤ 现有Pod数 < **目标Pod数** = 最大Pod数	10	4	4/10	• 修改FederatedHPA的最小Pod数为10。 • 扩展工作负载现有Pod数至10。
场景七	最小Pod数 ≤ 现有Pod数 ≤ 最大Pod数< **目标Pod数**	11	4	4/10	• 修改FederatedHPA的最小Pod数为11，修改FederatedHPA的最大Pod数为11。 • 扩展工作负载现有Pod数至11。

3.13.4.2 创建 CronFederatedHPA 策略以定时扩缩工作负载

本小节将指导您创建CronFederatedHPA策略，以设置固定时间周期自动扩缩工作负载。

在创建CronFederatedHPA策略前，您可以参考**CronFederatedHPA工作原理**了解相关原理与概念。若您需要了解不同负载伸缩策略的差异，请参考**负载伸缩概述**。

约束与限制

CronFederatedHPA策略仅支持1.19及以上版本的集群创建。

创建 CronFederatedHPA 策略

控制台创建

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击已开通集群联邦的容器舰队名称，进入容器舰队详情页面。

步骤3 在左侧导航栏选择“负载伸缩”，选择“定时伸缩策略”，并单击右上角“创建定时伸缩策略”。

步骤4 参考**表3-52**配置CronFederatedHPA策略。

表 3-52 基础配置参数说明

参数	参数说明
策略名称	CronFederatedHPA策略的名称。
命名空间	需要设置自动扩缩的工作负载所在命名空间的名称。
作用对象	可选择作用于工作负载或指标伸缩策略。 • 工作负载：选择需要应用该策略的工作负载，也可新建工作负载，新建的具体操作请参见**创建工作负载**。 • 指标伸缩策略：选择已创建的指标伸缩策略，也可单击右侧新建，新建的具体操作请参见**创建FederatedHPA策略**。

步骤5 单击策略配置下的“添加规则”，参考**表3-53**配置策略规则。

表 3-53 CronFederatedHPA 策略规则配置

参数	参数说明
规则名称	CronFederatedHPA策略的名称。
目标副本数	CronFederatedHPA策略的目标Pod数。

参数	参数说明
触发时间	可选择每小时、每天、每周、每月、每年，或选择Cron表达式自行配置触发时间。 ● 每小时：在每小时的第几分钟执行一次。例如选择“5”，策略将会在每小时的第5分钟执行一次。 ● 每天：在每天具体的某一个时间执行一次，可具体到分钟。设置完成后，策略将会在每天的该时点执行。 ● 每周：在每周具体的某一天内的某一时间执行一次，可具体到分钟。设置完成后，策略将会在每周的该天该时点执行。 ● 每月：在每月具体的某一天内的某一时间执行一次，可具体到分钟。设置完成后，策略将会在每月的该天该时点执行。 ● 每年：在每年具体的某月某天内的某一时间执行一次，可具体到分钟。设置完成后，策略将会在每年的该天该时点执行。 ● Cron表达式： Cron表达式遵循以下语法规则：  例如：0 0 13 * 5 表示此任务必须在每个星期五的午夜以及每个月的 13 日的午夜开始。
时区	可选择上海时区或新加坡时区。

步骤6 设置完成后，单击“确定”，然后单击“创建”，即可跳转定时伸缩策略列表查看策略详情，完成CronFederatedHPA策略创建。

----结束

命令行创建

步骤1 使用kubectl连接集群联邦，详细操作请参见使用kubectl连接集群联邦。

步骤2 使用如下命令创建并编辑cfhpa.yaml文件。

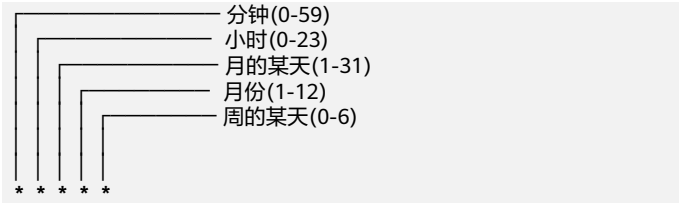
vi cfhpa.yaml

文件内容定义如下所示，参数定义请参见表3-54。本示例中构建的CronFederatedHPA策略名称为cron-federated-hpa，直接作用于名称为test的工作负载，内含两条定时规则（名称为“Scale-Up”和“Scale-Down”）。其中，Scale-Up规则为每天8:30的目标Pod数为10，Scale-Down规则为每天21:00的目标Pod数为5。

```
apiVersion: autoscaling.karmada.io/v1alpha1
kind: CronFederatedHPA
metadata:
  name: cron-federated-hpa          # CronFederatedHPA策略名称
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment                # 可选择Deployment或FederatedHPA
    name: test                      # 工作负载或FederatedHPA的名称
```

```
rules:
- name: "Scale-Up"              # 规则名称
  schedule: 30 08 * * *         # 触发时间
  targetReplicas: 10            # 目标Pod数，非负整数
  timeZone: Asia/Shanghai       # 时区
- name: "Scale-Down"            # 规则名称
  schedule: 0 21 * * *          # 触发时间
  targetReplicas: 5             # 目标Pod数，非负整数
  timeZone: Asia/Shanghai       # 时区
```

表 3-54 关键参数说明

参数	是否必填	参数类型	描述
kind	是	String	可选择Deployment或FederatedHPA。 ● 单独使用CronFederatedHPA策略：Deployment ● 同时使用FederatedHPA与CronFederatedHPA策略：FederatedHPA
name	是	String	CronFederatedHPA策略中定义的规则名称，长度为1~32字符。
schedule	是	String	策略触发时间，以Cron表达式呈现，具体遵循以下语法规则：  例如：0 0 13 * 5 表示此任务必须在每个星期五的午夜以及每个月的 13 日的午夜开始。
targetReplicas	是	String	CronFederatedHPA策略的目标Pod数。
timeZone	是	String	时区，可选择上海时区或新加坡时区。 ● 上海时区：Asia/Shanghai ● 新加坡时区：Asia/Singapore

步骤3 执行如下命令创建CronFederatedHPA策略。

kubectl apply -f cfhpa.yaml

回显如下表明创建成功。

```
CronFederatedHPA.autoscaling.karmada.io/cron-federated-hpa created
```

创建完成后，可以执行如下命令观察负载伸缩的运行效果。

- 检查工作负载的当前Pod数：kubectl get deployments
- 查看CronFederatedHPA策略事件（仅保留最近三条）：kubectl describe cronfederatedhpa cron-federated-hpa

可以执行如下命令管理CronFederatedHPA策略。其中cron-federated-hpa为CronFederatedHPA策略的名称，实际情况中需修改为自己所创建的策略名称。

- 获取CronFederatedHPA策略：kubectl get cronfederatedhpa cron-federated-hpa
- 更新CronFederatedHPA策略：kubectl edit cronfederatedhpa cron-federated-hpa
- 删除CronFederatedHPA策略：kubectl delete cronfederatedhpa cron-federated-hpa

----结束

3.13.4.3 管理 CronFederatedHPA 策略

本小节将指导您对已经创建的CronFederatedHPA策略进行管理，包括编辑策略与删除策略。

编辑 CronFederatedHPA 策略

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击已开通集群联邦的容器舰队名称，进入容器舰队详情页面。

步骤3 在左侧导航栏选择“负载伸缩”，选择“定时伸缩策略”，在需要编辑的策略所在行右侧单击“编辑”，在策略详情页面可以删除或添加策略规则。

- 如需删除策略规则，在规则后单击“删除”即可。
- 如需添加规则，单击策略配置下的“添加规则”，在弹窗内填写相关参数，然后单击“确定”。详细的参数说明请参见**表3-54**。

步骤4 修改完成后，单击“确定”。

----结束

删除 CronFederatedHPA 策略

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 单击已开通集群联邦的容器舰队名称，进入容器舰队详情页面。

步骤3 在左侧导航栏选择“负载伸缩”，选择“定时伸缩策略”，勾选需要删除的策略，可以在上方选择批量删除策略，也可在策略最右边单击“删除”，在弹框内单击“是”即可删除策略。

----结束

3.14 为集群添加标签与污点

UCS支持为集群打上不同的标签，来定义不同的属性，通过这些标签可以快速的了解各个集群的特点。而污点（taint）能够使集群排斥某些特定的Pod，从而避免Pod调度到该集群上，实现各集群负载的合理分配。

标签说明

通过给集群打上不同的标签，将集群进行分类，方便集群管理。

污点（Taints）说明

污点格式为“Key=Value:Effect”，Key和Value作为污点的标签，Value可以为空，Effect用于描述污点的效果。当前Effect支持如下两个选项：

- NoSchedule：不能容忍此污点的 Pod 不会被调度到集群上，但是现有 Pod 不会从集群中逐出。
- NoExecute：表示不能容忍此污点的 Pod 不会被调度到集群上，同时会将集群上已存在的Pod驱逐。

管理集群标签/污点

步骤1 登录UCS控制台。

步骤2 单击目标集群所在的容器舰队名称，在左侧导航栏选择“容器集群”，找到目标集群，在右上角单击 进入“标签与污点管理”。

图 3-52 标签与污点管理

步骤3 单击按钮，设置节点标签/污点。如需执行多项操作，可多次添加，最多支持10条操作。

图 3-53 添加标签/污点

- 选择“添加”或“删除”操作。
- 选择操作对象为“K8S标签”或“污点（Taints）”。
- 填写需要增加标签/污点的“键”和“值”。
- 如选择操作对象为“污点（Taints）”，需选择污点效果，关于污点效果说明请参见**污点（Taints）说明**。

步骤4 单击“确定”，对所选节点执行标签/污点操作。

----结束

3.15 集群联邦 RBAC 授权

UCS集群联邦可以实现基于华为云IAM的精细化权限管理。并在联邦中创建 Kubernetes 原生 RBAC 资源，实现联邦访问权限的精细化管理。

使用须知

- UCS的**权限管理**功能与当前联邦RBAC授权互不影响。使用UCS API时，前者生效；使用KubeConfig直接操作联邦时，后者生效。
- 在集群联邦和成员集群上分别创建的RBAC资源互相不感知、不影响。通过集群联邦入口配置的RBAC权限仅直接访问联邦时生效；直接访问成员集群时，仅成员集群上配置的RBAC生效。
- 在分配细粒度鉴权时，谨慎使用ClusterRole、ClusterRoleBinding等相关权限和角色配置，避免将资源查看权限赋予“Karmada-”为前缀的命名空间；建议使用Role、RoleBinding对指定用户命名空间的资源赋予权限。

集群联邦 RBAC 授权

UCS集群联邦使用Kubernetes原生RBAC鉴权方式，通过创建RBAC资源，为子用户授予联邦访问权限。

步骤1 使用拥有Tenant Administrator权限的用户，下载并配置好 KubeConfig 文件。详细请参见使用kubectl连接集群联邦。

步骤2 将如下内容保存为list-deploy.yaml文件。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: list-deploy-role-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: list-deploy-role
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: <user-id>     # IAM用户ID
  - apiGroup: rbac.authorization.k8s.io
    kind: Group
    name: <group-id>   # IAM用户组的ID
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: list-deploy-role
  namespace: default
rules:
  - apiGroups:
      - apps
    resources:
      - deployments
    verbs:
      - list
      - get
```

其中，<user-id>为IAM用户ID，<group-id>为用户组ID。Role与RoleBinding的字段含义详见**使用 RBAC 鉴权**。

使用如下命令创建对应的资源：

```
kubectl apply -f list-deploy.yaml
```

创建完成后，<user-id>用户或<group-id>用户组下的用户可使用查看default命名空间的deployment：

```
kubectl get deploy -n default
```

----结束

4 镜像仓库

UCS深度整合了华为云容器镜像服务（SWR）能力，支持镜像全生命周期管理，为您提供简单易用、安全可靠的镜像管理功能，帮助您快速部署容器化服务。

通过使用容器镜像服务，您无需自建和维护镜像仓库，即可享有云上的镜像安全托管及高效分发服务，获得容器上云的顺畅体验。

产品功能

- 镜像全生命周期管理

  容器镜像服务支持镜像的全生命周期管理，包括镜像的上传、下载、删除等。
- 私有镜像仓库

  容器镜像服务提供私有镜像库，并支持细粒度的权限管理，可以为不同用户分配相应的访问权限（读取、编辑、管理）。
- 镜像加速

  容器镜像服务通过华为自主专利的镜像下载加速技术，使CCE集群下载镜像时在确保高并发下能获得更快的下载速度。
- 镜像仓库触发器

  容器镜像服务支持容器镜像版本更新自动触发部署。您只需要为镜像设置一个触发器，通过触发器，可以在每次镜像版本更新时，自动更新使用该镜像部署的应用。

约束与限制

通过私网接入的附着集群可能无法使用镜像仓库功能，如需使用，请确保集群具有访问公网的能力。

上传镜像

步骤1 登录UCS控制台，在左侧导航栏中单击“镜像仓库”。

步骤2 查看当前镜像仓库的基本信息，单击此镜像仓库，进入容器镜像服务。

图 4-1 镜像仓库

步骤3 容器镜像服务上传镜像的详细操作请参见**客户端上传镜像**。

----结束

使用镜像

通过UCS管理的集群及联邦，均支持使用镜像仓库创建工作负载。镜像上传成功后，在集群中创建工作负载时可选择“镜像创建”，以UCS接管的CCE集群为例，具体操作如下：

步骤1 登录集群控制台。

步骤2 在新页面的左侧导航栏中选择“工作负载”，然后单击右上角“镜像创建”按钮。

步骤3 在“基本信息”栏输入工作负载参数，以创建无状态工作负载为例。

- 负载类型：无状态工作负载。
- 负载名称：负载名称可自定义。
- 实例数量：请根据业务需要自行选择。
- 描述：请输入描述信息。
- 时区同步：选择是否开启时区同步。开启后容器与节点使用相同时区（时区同步功能依赖容器中挂载的本地磁盘，请勿修改删除）。

步骤4 在“容器配置”栏单击“选择镜像”。

在“我的镜像”页签下，选择已上传的镜像，单击“确定”。

须知

- 如所选镜像为公开镜像，无需选取“镜像访问凭证”。
- 如所选镜像为用户在镜像仓库中上传的私有镜像，需选取“镜像访问凭证”，否则无法拉取成功。

  单击“创建密钥”，可创建镜像仓库的镜像访问凭证，具体操作参见**创建镜像密钥**。

图 4-2 容器配置

步骤5 单击“创建工作负载”，完成创建。如您想了解更多创建工作负载的步骤，请参见**无状态负载**。

----结束

创建镜像密钥

华为云集群在创建时默认生成一个名为default-secret的密钥，其中包含SWR的访问凭证，因此无需重新创建镜像密钥。

附着集群在使用SWR中的私有镜像时，需创建镜像密钥用于拉取SWR镜像，操作步骤如下：

步骤1 登录集群控制台。

步骤2 在新页面的左侧导航栏中单击“配置项与密钥”，并选择“密钥”页签。

步骤3 单击右上角“创建密钥”，并填写参数。

图 4-3 创建密钥

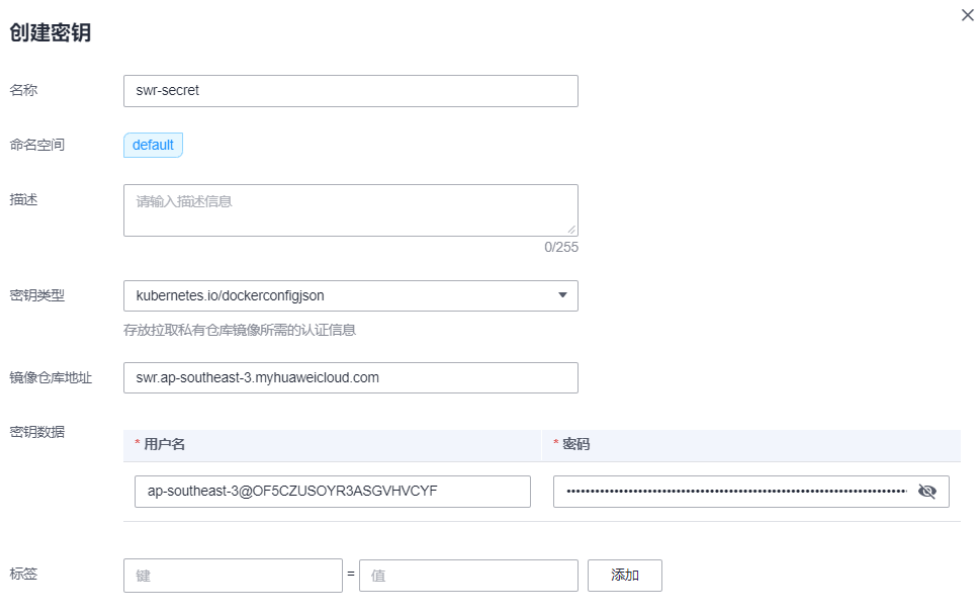


表 4-1 基本信息说明

参数	参数说明
名称	新建的密钥的名称，同一个命名空间内命名必须唯一。
命名空间	新建密钥所在的命名空间，默认为default。
描述	密钥的描述信息。
密钥类型	新建的密钥类型，选择kubernetes.io/dockerconfigjson类型，用于存放拉取私有仓库镜像所需的认证信息。

参数	参数说明
镜像仓库地址	镜像仓库地址为“swr.区域.myhuaweicloud.com”，如“亚太-新加坡”对应的镜像仓库地址为：swr.ap-southeast-3.myhuaweicloud.com。SWR的应用区域请参见**地区和终端节点**。
密钥数据	工作负载密钥的数据可以在容器中使用，输入私有镜像仓库的用户名和密码。 使用SWR时，用户名密码的获取方式如下： 1. 单击右上角用户名，前往“我的凭证 > 管理访问密钥”，单击“新增访问密钥”，即可在下载的“credentials.csv”文件中获取AK和SK信息。 AK/SK文件**仅能下载一次**，请妥善保存。更多说明请参见**访问密钥**。 2. 登录一台linux系统的计算机，执行如下命令获取登录密钥，其中*$AK*和*$SK*为上一步获取的AK/SK。 **printf "*$AK*" \| openssl dgst -binary -sha256 -hmac "*$SK*" | od -An -vtx1 | sed 's/[\n]//g' | sed 'N;s/\n//'** 3. 用户名为“[区域项目名称]@[AK]”，例如“ap-southeast-3@***”。 密码为**2**中获取的登录密钥。
密钥标签	密钥的标签。键值对形式，输入键值对后单击“添加”。

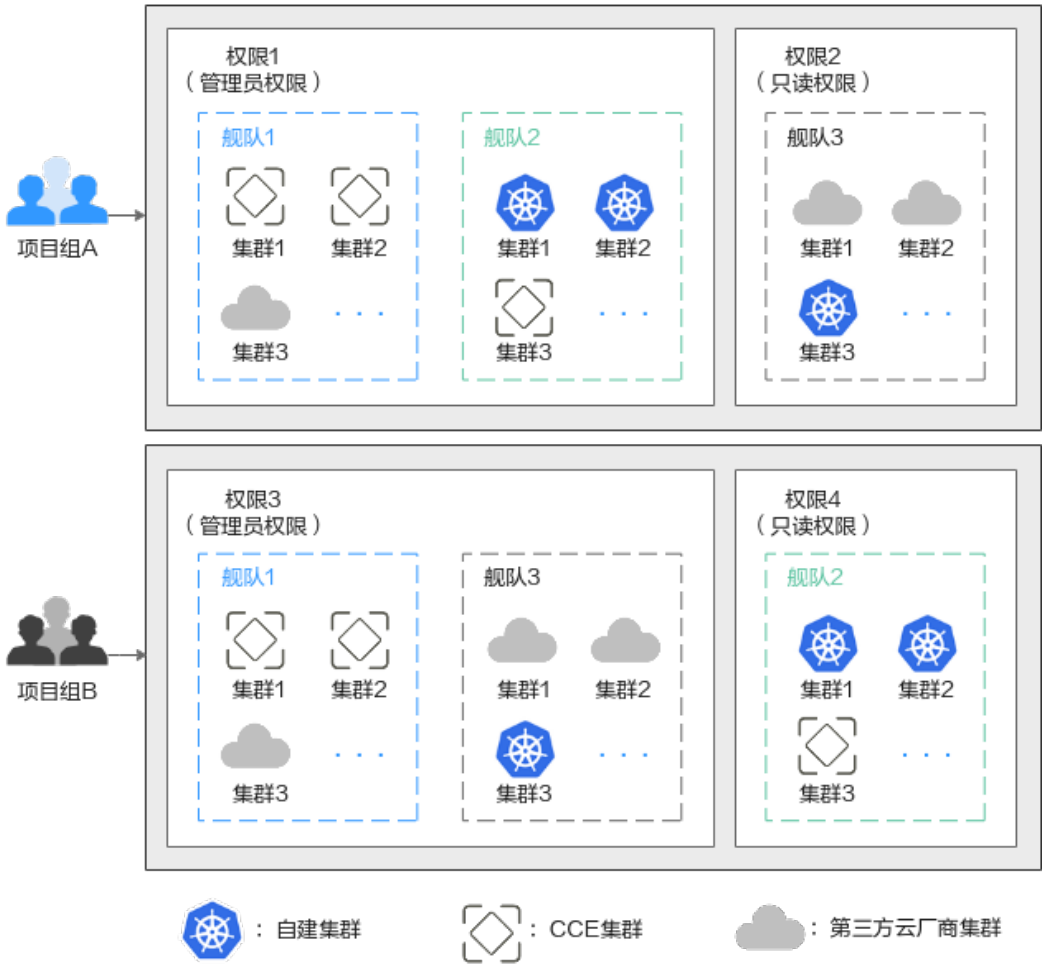
----结束

5 权限管理

5.1 UCS 权限概述

UCS在统一身份认证服务（IAM）能力基础上，为用户提供细粒度的权限管理功能，帮助用户灵活便捷地对租户下的IAM用户设置不同的UCS资源权限，结合权限策略和舰队设计，可实现企业不同部门或项目之间的权限隔离。

图 5-1 权限设计



UCS 权限类型

UCS权限管理是在IAM与Kubernetes的角色访问控制（RBAC）的能力基础上，打造的细粒度权限管理功能。支持UCS服务资源权限、集群中Kubernetes资源权限两种维度的权限控制，这两种权限针对的是不同类型的资源，在授权机制上也存在一些差异，具体如下：

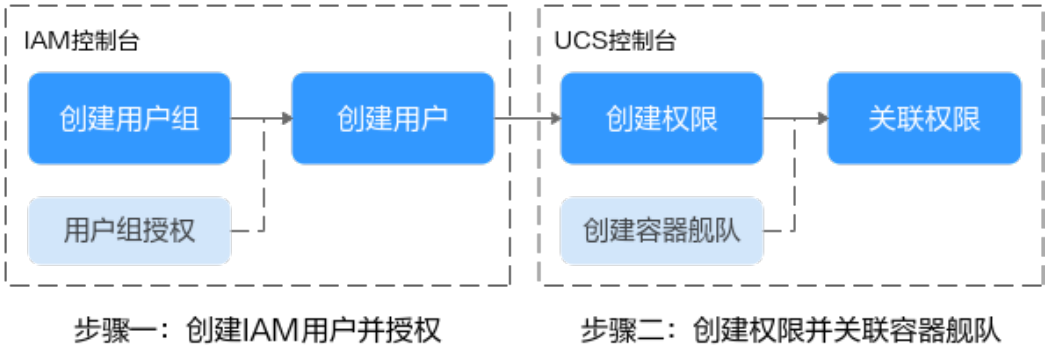
- **UCS服务资源权限**：是基于IAM系统策略的授权。UCS服务资源包括容器舰队、集群、联邦实例等等，管理员可以针对用户的角色（如开发、运维）进行差异化授权，精细控制他们对UCS资源的使用范围。
- **集群中Kubernetes资源权限**：是基于Kubernetes RBAC能力的授权，可授予针对集群内Kubernetes资源对象的细化权限，通过权限设置可以让不同的用户有操作不同Kubernetes资源对象（如工作负载、任务、服务等Kubernetes原生资源）的权限。

UCS的权限可以从使用的阶段来看，第一个阶段是创建和管理基础设施资源的权限，也就是拥有创建容器舰队、注册集群、开通集群联邦等操作的权限；第二个阶段是使用集群Kubernetes资源对象（如工作负载、服务等）的权限；第三个阶段是监控运维基础设施资源以及Kubernetes资源的权限。其中，第一个和第三个阶段属于UCS服务资源权限，在IAM控制台按照IAM系统策略的方式授予；第二个阶段属于集群中Kubernetes资源权限，由管理员在UCS控制台“权限管理”页面创建，并在“容器舰队”页面完成权限与特定舰队或集群的关联。

权限管理流程

新建IAM用户的权限管理主要流程可参见权限管理流程。

图 5-2 权限管理流程

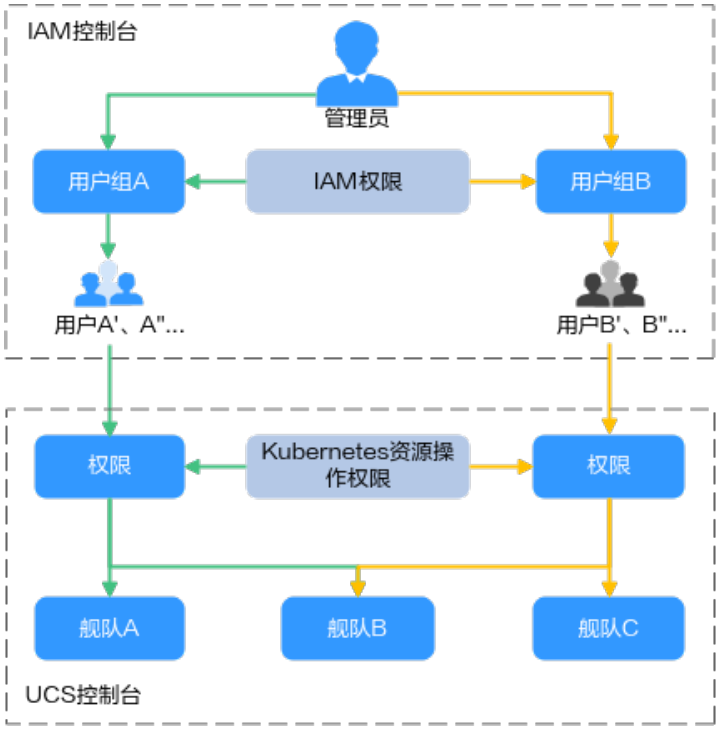


基本概念说明

UCS权限管理包含如下基本概念，其关系如图5-3所示。

- **用户：**由管理员账号在统一身份认证服务（IAM）中创建的用户，是云服务的使用人员，具有独立的身份凭证（密码和访问密钥），根据账号授予的权限使用资源。
- **用户组：**用户组是用户的集合，IAM可以通过用户组功能实现用户的授权。您创建的IAM用户，加入特定用户组后，将具备对应用户组的权限。例如，管理员为用户组授予UCS FullAccess权限后，其中的用户将具备UCS服务的管理员权限。当某个用户加入多个用户组时，此用户同时拥有多个用户组的权限，即多个用户组权限的全集。
- **权限：**由UCS管理员定义的某个或某些用户对集群中Kubernetes资源的操作范围，UCS预置了几个常用权限，包括管理员权限、只读权限、开发权限，同时也支持用户自定义权限。更多介绍请参见创建权限。
- **舰队：**舰队是多个集群的集合，管理员可以使用舰队来实现关联集群的分类。舰队还可以实现多集群的统一管理，包括权限管理、安全策略、配置管理以及多集群编排等统一管理的能力。舰队与权限是多对多的关系，即一个权限可以关联多个舰队，一个舰队也可以关联多个权限。

图 5-3 权限关系示意图



约束与限制

- 本地集群支持使用华为云IAM Token访问Kube APIServer；同时，不对UCS的系统策略（UCS FullAccess、UCS CommonOperations、UCS CIAOperations和UCS ReadOnlyAccess）进行识别。
- 对于多云集群，目前只有华为云账号可以执行集群注册的操作，暂不支持IAM系统策略。

5.2 UCS 服务资源权限（IAM 授权）

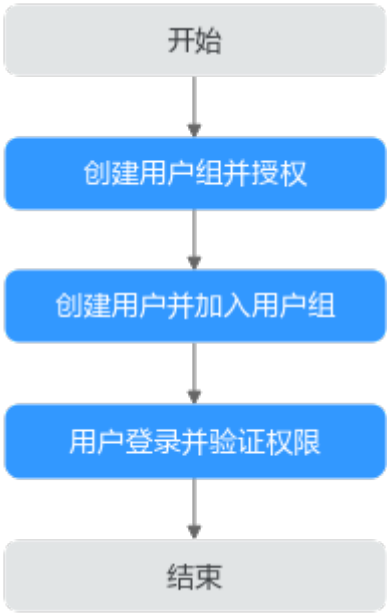
UCS服务资源权限是基于IAM系统策略的授权，UCS服务资源包括容器舰队、集群、联邦实例等等，管理员可以针对用户的角色（如开发、运维）进行差异化授权，精细控制他们对UCS资源的使用范围。

IAM通过用户组功能实现用户的授权，在给用户组授权之前，请您提前阅读用户组可以添加的**UCS系统策略**，以及**UCS功能所需的最小权限**，然后结合实际情况进行授权。若您想了解其他云服务的权限策略，请参见**系统权限**。

授权流程

本节以授予“UCS ReadOnlyAccess”权限为例介绍为用户授权的方法，操作流程如**图5-4**所示。

图 5-4 给用户授予 UCS 权限流程



1. **创建用户组并授权**。
   管理员账号在IAM控制台创建用户组，并授予UCS权限，例如：UCS ReadOnlyAccess。

   说明

   UCS部署时不区分物理区域，为全局级服务。授权时，选择授权范围方案为“所有资源”。
2. **创建用户并加入用户组**。
   在IAM控制台创建用户，并将其加入**1**中创建的用户组。
3. **用户登录**并验证权限。
   新创建的用户登录控制台，验证权限（假设当前权限仅包含UCS ReadOnlyAccess）：
   - 在“服务列表”中选择华为云UCS，进入UCS总览页，单击左侧导航栏“基础设施 > 容器舰队”，如果创建舰队和注册集群时提示无访问权限，表示“UCS ReadOnlyAccess”已生效。
   - 在“服务列表”中选择除华为云UCS以外的其他服务（如弹性云服务器ECS），若提示权限不足，表示“UCS ReadOnlyAccess”已生效。

系统策略

IAM中预置的UCS系统策略包含UCS FullAccess、UCS CommonOperations、UCS CIAOperations和UCS ReadOnlyAccess几种类型。

- UCS FullAccess：UCS服务管理员权限，拥有该权限的用户拥有服务的所有权限（包含制定权限策略、安全策略等）。
- UCS CommonOperations：UCS服务基本操作权限，拥有该权限的用户可以执行创建工作负载、流量分发等操作。
- UCS CIAOperations：UCS服务容器智能分析管理员权限。
- UCS ReadOnlyAccess：UCS服务只读权限（除容器智能分析只读权限）。

您可以通过查看策略内容来了解系统策略所支持的授权项，授权项的格式为“服务名:资源类型:操作”，支持通配符号*，通配符号*表示所有。

例如，UCS FullAccess的策略内容如下所示。可以看出，该策略包含了UCS、CCE（云容器引擎）、SWR（容器镜像服务）的所有权限，AOM（应用运维管理）、SMN（应用运维管理）、DNS（云解析服务）等服务的部分资源的操作权限。

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "ucs:*:*",
                "cce:*:*",
                "swr:*:*",
                "aom:*:get",
                "aom:*:list",
                "smn:*:list",
                "dns:*:get*",
                "dns:*:list*",
                "dns:*:get",
                "dns:*:list",
                "dns:recordset:create",
                "dns:recordset:delete",
                "dns:recordset:update",
                "dns:tag:get",
                "lts:*:get",
                "lts:*:list",
                "apm:*:get",
                "apm:*:list",
                "vpcep:epservices:*",
                "vpcep:connections:*",
                "vpcep:endpoints:*",
                "elb:*:get",
                "elb:*:list",
                "vpc:*:get",
                "vpc:*:list",
                "ief:*:get",
                "ief:*:list",
                "cgs:images:operate",
                "cgs:*:get",
                "cgs:*:list"
            ],
            "Effect": "Allow"
        }
    ]
}
```

UCS 功能所需的最小权限

华为云各服务之间存在业务交互关系，UCS也依赖其他云服务实现一些功能（如镜像仓库、域名解析），因此，上述四种系统策略经常和其他云服务的角色或策略结合使用，以达到精细化授权的目的。管理员在为IAM用户授权时，应该遵循权限最小化的安全实践原则，**表5-1**列举了UCS各功能管理员、操作、只读权限所需要的最小权限。

须知

- 如您的华为云账号为首次登录UCS控制台，需要为其授权，同意授权后，UCS将在统一身份认证服务为您创建名为ucs_admin_trust的委托，为保证UCS服务正常使用，请不要删除或者修改该委托。
- IAM用户所在用户组未授予任何权限的情况下，您将无法访问UCS控制台，请参考**表5-1**授予相关权限。

表 5-1 UCS 功能所需的最小权限

功能	权限类型	权限范围	最小权限
容器舰队	管理员权限	• 创建、删除舰队 • 注册华为云集群（CCE集群、CCE Turbo集群）、本地集群或附着集群 • 注销集群 • 将集群加入、移出舰队 • 为集群或舰队关联权限 • 开通集群联邦、联邦管理相关操作（如创建联邦工作负载、创建域名访问等）	UCS FullAccess
	只读权限	查询集群、舰队的列表或详情	UCS ReadOnlyAccess
华为云集群	管理员权限	对华为云集群及集群下所有Kubernetes资源对象（包含节点、工作负载、任务、服务等）的读写权限。	UCS FullAccess + CCE Administrator
	操作权限	对华为云集群及集群下大多数Kubernetes资源对象的读写权限，对命名空间、资源配额等Kubernetes资源对象的只读权限。	UCS CommonOperations + CCE Administrator
	只读权限	对华为云集群及集群下所有Kubernetes资源对象（包含节点、工作负载、任务、服务等）的只读权限。	UCS ReadOnlyAccess + CCE Administrator
本地/附着/多云集群	管理员权限	本地/附着/多云集群及集群下所有Kubernetes资源对象（包含节点、工作负载、任务、服务等）的读写权限。	UCS FullAccess
	操作权限	本地/附着/多云集群及集群下大多数Kubernetes资源对象的读写权限，对命名空间、资源配额等Kubernetes资源对象的只读权限。	UCS CommonOperations + UCS RBAC权限（需要包含namespaces资源对象的list权限）
	只读权限	本地/附着/多云集群及集群下所有Kubernetes资源对象（包含节点、工作负载、任务、服务等）的只读权限。	UCS ReadOnlyAccess + UCS RBAC权限（需要包含namespaces资源对象的list权限）
镜像仓库	管理员权限	容器镜像服务的所有权限，包括创建组织、上传镜像、查看镜像列表或详情、下载镜像等操作。	SWR Administrator

功能	权限类型	权限范围	最小权限
权限管理	管理员权限	• 创建、删除权限 • 查看权限列表或详情 **说明** 创建权限需要同时授予子用户IAM ReadOnlyAccess权限（IAM服务的只读权限），用于获取IAM用户列表。	UCS FullAccess + IAM ReadOnlyAccess
	只读权限	查看权限列表或详情	UCS ReadOnlyAccess + IAM ReadOnlyAccess
策略中心	管理员权限	• 启用策略中心 • 创建、停用策略实例 • 查看策略列表 • 查看策略实施详情	UCS FullAccess
	只读权限	对于已启用策略中心的舰队和集群，拥有该权限的用户可以查看策略列表和查看策略实施详情。	UCS CommonOperations 或 UCS ReadOnlyAccess
流量分发	管理员权限	创建流量策略、暂停调度策略、删除调度策略等操作。	（推荐）UCS CommonOperations + DNS Administrator 或 UCS FullAccess + DNS Administrator
	只读权限	查看流量策略列表或详情	UCS ReadOnlyAccess + DNS Administrator
容器智能分析	管理员权限	• 接入、取消接入集群 • 查看基础设施、应用负载等多维度监控数据	UCS CIAOperations

5.3 集群中 Kubernetes 资源权限（RBAC 授权）

集群中Kubernetes资源权限是基于Kubernetes RBAC能力的授权，管理员可授予用户针对集群内特定Kubernetes资源对象的操作权限，权限最终作用在舰队或未加入舰队的集群的命名空间上。

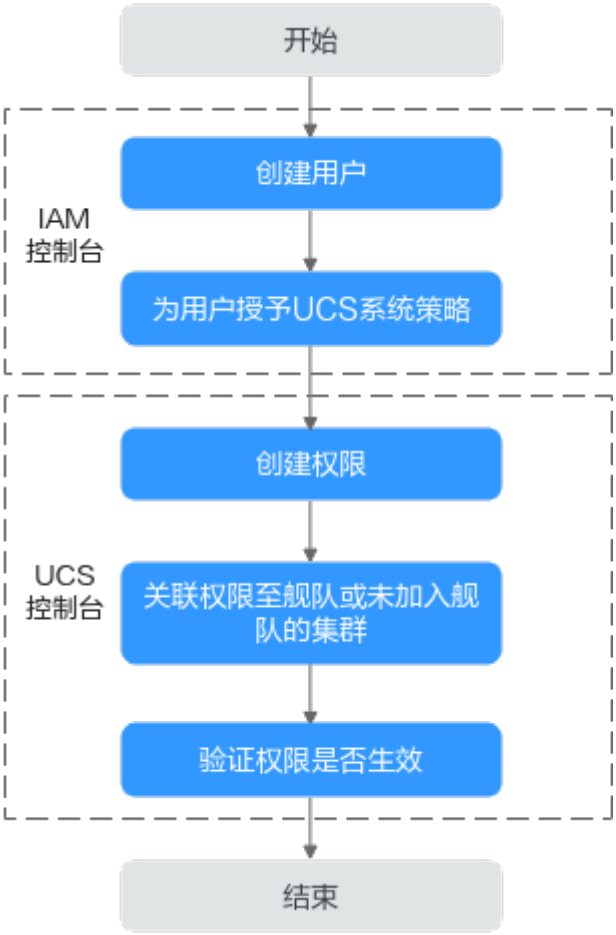
本章节以“只读权限”为例，介绍为用户授予Kubernetes资源权限的方法，操作流程如图5-5所示。

须知

UCS的集群操作权限设置仅对非华为云集群生效。对于华为云集群（CCE集群、CCE Turbo集群）的操作权限以IAM权限或者CCE RBAC权限为准。

授权流程

图 5-5 给用户授予 Kubernetes 资源权限流程



1. **创建用户**。

   管理员账号在IAM控制台创建一个用户。

2. **为用户授予UCS系统策略**。

   授予Kubernetes资源权限之前，必须先为用户授予UCS系统策略的权限，本例中应该授予UCS ReadOnlyAccess策略（UCS服务只读权限）。

3. **创建权限**。

   管理员账号在UCS控制台创建权限，选择“只读权限”的权限类型，表示对所有Kubernetes资源对象的只读权限。

4. **关联权限至舰队或未加入舰队的集群**。

   将权限关联至舰队，关联时需要选择权限作用的命名空间。也可以关联权限至未加入舰队的集群。

5. **验证权限是否生效**。
   新创建的用户登录控制台，验证只读权限是否生效。

创建权限

步骤1 登录UCS控制台，在左侧导航栏中选择“权限管理”。

步骤2 单击右上角的“创建权限”按钮。

步骤3 在弹出页面中填写权限参数。

图 5-6 创建权限

- 权限名称：自定义权限的名称，需以小写字母开头，由小写字母、数字、中划线（-）组成，且不能以中划线（-）结尾。
- 用户：在下拉列表中勾选新创建的用户名。支持选择多个用户，假设一个企业中的开发团队有多名员工，他们对资源的操作权限一样，就可以在创建权限时选择多个用户以达到批量授权的目的。
  本文以添加一个“readonly_user”用户为例。
- 权限类型：支持管理员权限、只读权限、开发权限和自定义权限。

表 5-2 权限类型说明

权限类型	说明
管理员权限	对所有集群资源对象的读写权限
只读权限	对所有集群资源对象的只读权限
开发权限	对大多数集群资源对象的读写权限，对命名空间、资源配额等集群资源对象的只读权限
自定义权限	权限由您选择的操作类型和资源对象决定

- 权限内容：表示对哪些资源可以执行哪些操作。管理员权限、只读权限、开发权限的权限内容已经模板化，您可以单击 按钮查看权限的详细内容。当权限类型选择“自定义权限”时，需要设置操作类型和资源对象。
  操作类型：包含如下类型，也支持新增操作类型（如deletecollection，表示删除多个资源）。

- get：按名称检索特定的资源对象。
- list：检索命名空间中特定类型的所有资源对象。
- watch：响应资源变化。
- create：创建资源。
- update：更新资源。
- patch：局部更新资源。
- delete：删除资源。

说明

对于全部操作推荐选择：全部。

对于只读操作推荐选择：get + list + watch。

对于读写操作推荐选择：get + list + watch + create + update + patch + delete。

资源对象：您可以选择“全部资源”或“指定资源”。全部资源对象包括当前已有的资源对象和后续新增的自定义资源对象；“指定资源”即表示您自己选择资源对象的范围，为了便于查找，UCS服务将资源对象按照工作负载、服务、配置和存储、身份验证、授权、权限、扩展、集群维度划分。

若系统资源中没有您需要的资源对象，也可以新增自定义资源对象。

如果针对不同资源对象，操作类型不同（例如：对deployments具有create、delete操作权限，对secrets具有get、list、watch操作权限），可以单击 + 按钮添加多组内容。

说明

若您想了解更多资源对象和操作类型的知识，请参阅**Kubernetes API**。

- 描述：添加权限的描述信息。

步骤4 单击“确定”。权限创建完成后，需要继续关联舰队或未加入舰队的集群，才可正常操作Kubernetes资源。

----结束

关联权限至舰队或未加入舰队的集群

舰队是多个集群的集合，舰队可以实现多集群的权限统一管理。因此建议您将集群加入舰队后，为舰队关联权限，这样舰队中的集群将具有相同的权限。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器舰队”。

步骤2 在目标舰队栏中，单击右上角的 按钮。

图 5-7 为舰队关联权限

步骤3 在弹出的页面单击“修改容器舰队权限”或“关联权限”，打开修改权限页面，将已创建好的权限和舰队的命名空间关联起来。

图 5-8 修改权限

- 命名空间：支持“全部命名空间”和“指定命名空间”。全部命名空间包括当前舰队已有的命名空间和舰队后续新增的命名空间；“指定命名空间”即表示您自己选择命名空间的范围，UCS服务提供了几个常见的命名空间供您选择（如default、kube-system、kube-public），您也可以新增命名空间，但要自行确保新增的命名空间在集群中存在。

  请注意，选择的命名空间仅对权限中命名空间级资源生效，不影响权限中的集群资源。关于命名空间级和集群级资源的介绍，请参见**Kubernetes资源对象**章节。
- 关联权限：从下拉列表中选择权限，支持一次性选择多个权限，以达到批量授权的目的。

本示例中，选择“default”命名空间，选择“readonly”权限。

如果针对不同命名空间，关联的权限不同（例如：为default命名空间关联readonly权限，为development命名空间关联develop权限），可以单击 + 按钮添加多组授权关系。

步骤4 单击“确定”，完成权限的关联。

如果后续需要修改舰队的权限，采用同样的方法，重新选择命名空间和权限即可。

----结束

验证权限是否生效

新创建的“readonly_user”用户登录控制台，验证权限是否生效（以附着集群为例）：

- 进入舰队中附着集群的控制台，选择“资源 > 工作负载”，如果可以正常查看default命名空间下的工作负载，但是查看其他命名空间下的工作负载时提示无权限，表示“只读权限”已生效。
- 进入舰队中附着集群的控制台，选择“资源 > 工作负载”，切换至default命名空间，单击右上角“创建负载”按钮，若提示无权限，表示“只读权限”已生效。

5.4 Kubernetes 资源对象

Kubernetes资源对象根据作用域区分为命名空间级和集群级。

命名空间级别

命名空间（Namespace）是Kubernetes提供的隔离机制，用于给集群中的任何资源对象进行分类、筛选和管理。

如果不同的资源对象放在不同的命名空间下，他们就会相互隔离。例如，获取所有Pod使用的命令是：

```
kubectl get pod
```

这里的Pod是有命名空间的，默认为default。指定命名空间需使用如下命令：

```
kubectl get pod -n default
```

当要获取全部命名空间下的Pod时，使用如下命令：

```
kubectl get pod --all-namespaces
```

这样就可以看到集群下所有的Pod了。

```
$ kubectl get pod --all-namespaces
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
default       nginx-dd9796d66-5chbr                      1/1     Running   0          3d1h
default       nginx-dd9796d66-xl69p                      1/1     Running   0          15d
default       sa-example                                 1/1     Running   0          10d
kube-system   coredns-6fcd88c4c-k8rtf                    1/1     Running   0          48d
kube-system   coredns-6fcd88c4c-z46p4                    1/1     Running   0          48d
kube-system   everest-csi-controller-856f8bb679-42rgw    1/1     Running   1          48d
kube-system   everest-csi-controller-856f8bb679-xs6dz    1/1     Running   0          48d
kube-system   everest-csi-driver-mkpbv                   2/2     Running   0          48d
kube-system   everest-csi-driver-v754w                   2/2     Running   0          48d
kube-system   icagent-5p44q                              1/1     Running   0          48d
kube-system   icagent-jrlbl                              1/1     Running   0          48d
monitoring    alertmanager-alertmanager-0                2/2     Running   0          29d
monitoring    cluster-problem-detector-7788f94f64-thp6s  1/1     Running   0          29d
monitoring    custom-metrics-apiserver-5f7dcf6d9-n5nrr   1/1     Running   0          19d
monitoring    event-exporter-6844c5c685-khf5t            1/1     Running   1          3d1h
monitoring    kube-state-metrics-8566d5f5c5-7kx7b        1/1     Running   0          29d
monitoring    node-exporter-7l4ml                        1/1     Running   0          29d
monitoring    node-exporter-gpxvl                        1/1     Running   0          29d
```

Pod属于命名空间级资源。此外，大多数工作负载资源、Service资源、配置与存储资源都属于命名空间级。

- **工作负载资源**

  Pod：Kubernetes部署应用或服务的最小的基本单位。

ReplicaSet：Kubernetes中的一种副本控制器，主要作用是控制由其管理的Pod，使Pod副本的数量始终维持在预设的个数。

Deployment：声明了Pod的模板和控制Pod的运行策略，适用于部署无状态的应用程序。

StatefulSet：主要用于管理有状态的应用，创建的Pod拥有根据规范创建的持久型标识符。

DaemonSet：主要用于部署常驻集群内的后台程序，例如节点的日志采集。

Job：Job控制器会创建一个或者多个Pod，这些Pod按照运行规则运行，直至运行结束。

CronJob：根据指定的预定计划周期性地运行一个Job。

- **服务发现资源**

  Service：用户在 Kubernetes中可以部署各种容器，其中一部分是通过HTTP、HTTPS协议对外提供七层网络服务，另一部分是通过TCP、UDP协议提供四层网络服务。而Kubernetes 定义的Service资源是用来管理集群中四层网络的服务访问。基于四层网络，Service提供了集群内容器服务的暴露能力。

  Ingress：Ingress能够提供七层网络下HTTP、HTTPS协议服务的暴露，同时也包括各种七层网络下的常见能力。Ingress是允许访问到集群内Service规则的集合，您可以通过配置转发规则，实现不同URL可以访问到集群内不同的Service。

- **配置与存储资源**

  ConfigMap：key-value类型的键值对，通过ConfigMap您可以将配置与运行的镜像进行解耦，使应用程序有更强的移植性。

  Secret：key-value类型的键值对，用于存储密码、令牌、密钥等敏感信息，降低直接对外暴露的风险。

  Volume：Volume（存储卷）本质是一个目录，其中可能包含一些数据，Pod中的容器可以访问该目录。Volume的生命周期与挂载它的Pod相同，但是Volume里面的文件可能在Volume消失后仍然存在，这取决于Volume的类型。

集群级别

集群级别的资源，它的作用范围要比命名空间大很多，全集群可见，并且可以调用，不属于某一个命名空间，因此资源对象的名称必须全局唯一。

集群级别的资源，不管在任何命名空间下定义，在其他的命名空间下都能看得到，在定义的时候无需指定命名空间。

常见的集群级资源有：Namespace（命名空间）、Node（节点）、Role（角色）、RoleBinding（角色绑定）、ClusterRole（集群角色）、ClusterRoleBinding（集群角色绑定）。

- **Namespace**：是Kubernetes提供的隔离机制，用于给集群中的任何资源对象进行分类、筛选和管理。

  要查询集群下所有的命名空间，可以使用如下命令：

  ```
  kubectl get ns
  ```

- **Node**：节点是组成容器集群的基本元素，可以为虚拟机或物理机。每个节点都包含运行Pod所需要的基本组件，包括Kubelet、Kube-proxy等。Node名称要全局唯一。
- **Role**：角色，其实是定义一组对Kubernetes资源（命名空间级别）的访问规则。
- **RoleBinding**：角色绑定，定义了用户和角色的关系。

- **ClusterRole**：集群角色，其实是定义一组对Kubernetes资源（集群级别，包含全部命名空间）的访问规则。
- **ClusterRoleBinding**：集群角色绑定，定义了用户和集群角色的关系。

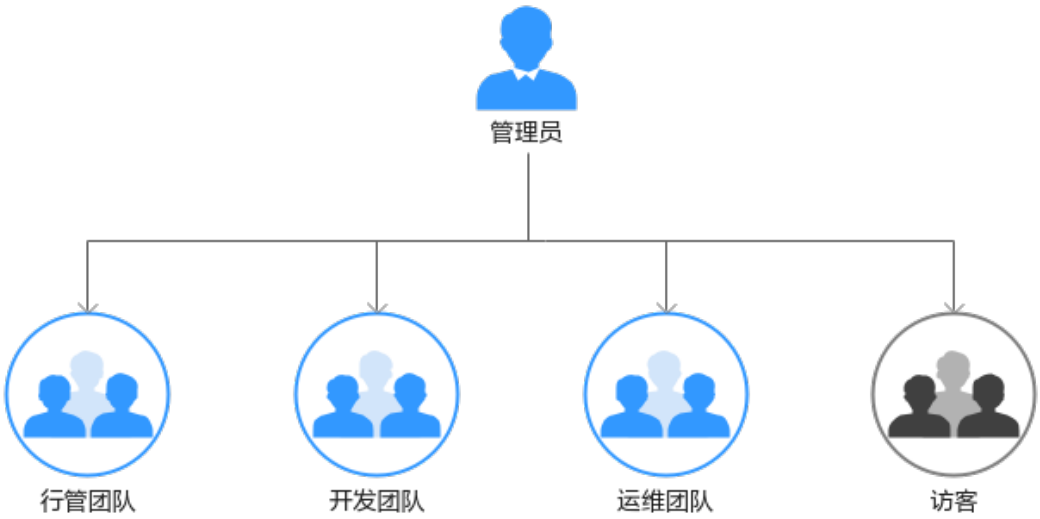
说明

Role和ClusterRole指定了可以对哪些资源做哪些动作，RoleBinding和ClusterRoleBinding将角色绑定到特定的用户、用户组或ServiceAccount上。

5.5 示例：某公司权限设计及配置

假设A公司在华为云使用UCS服务管理多集群，公司中有多个职能团队，分别负责权限分配、资源管理、创建应用、流量分发、监控运维等。结合使用IAM和UCS的权限管理，可以实现精细化授权的目标。

图 5-9 组织结构示意图



- 行管团队：负责管理公司所有资源的团队。
- 开发团队：负责业务开发的团队。
- 运维团队：负责查看并监控所有资源使用情况的团队。
- 访客：预留的只读权限团队，指那些仅具有查看资源权限的人员。

通过表5-3，给公司不同的职能团队设置不同的权限，可以实现各团队之间权限隔离，各司其职。

表 5-3 权限设置

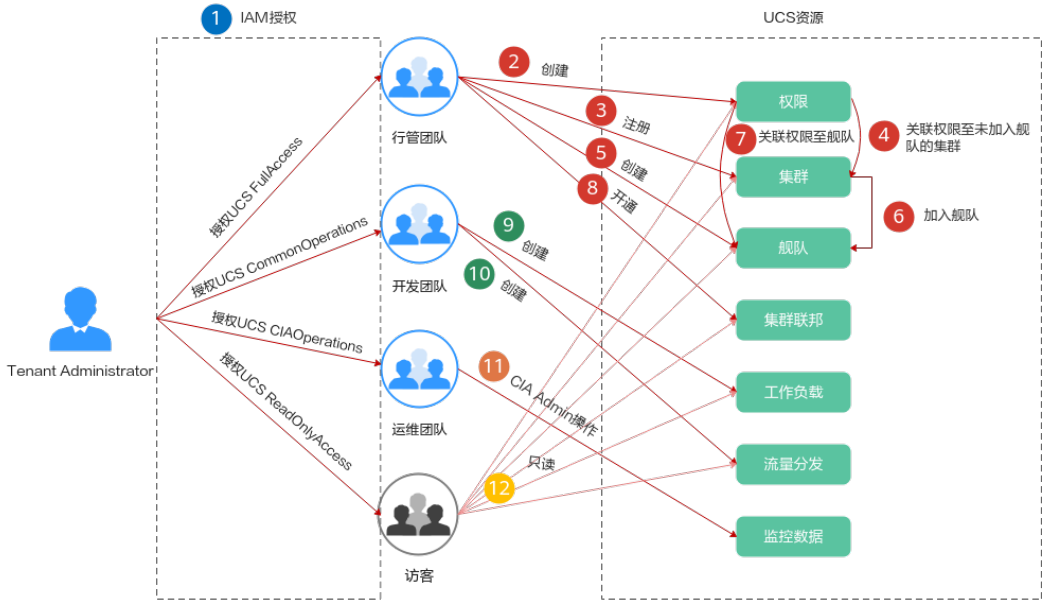
职能团队	需要授予的策略	权限说明
行管团队	UCS FullAccess	UCS服务管理员权限，拥有该权限的用户拥有服务的所有权限（包含制定权限策略、安全策略等）。

职能团队	需要授予的策略	权限说明
开发团队	UCS CommonOperations	UCS服务基本操作权限，拥有该权限的用户可以执行创建工作负载、流量分发等操作。
运维团队	UCS CIAOperations	UCS服务容器智能分析管理员权限。
访客	UCS ReadOnlyAccess	UCS服务只读权限（除容器智能分析只读权限）。

权限设计

公司不同的职能团队针对UCS资源的操作范围如下图所示：

图 5-10 UCS 资源操作全景



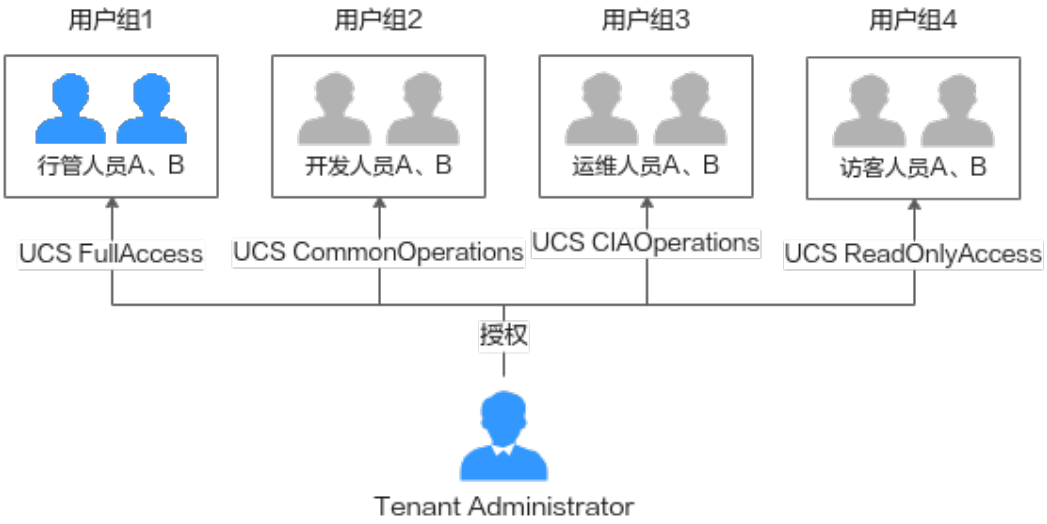
- 1：由Tenant Administrator角色的管理员为各个职能团队授权。
- 2、3、4、5、6、7、8：拥有UCS FullAccess权限的行管团队负责创建舰队、注册集群、将集群加入舰队，以及开通集群联邦能力，搭建多集群联邦基础设施。同时，行管团队创建权限，并将权限关联至舰队或未加入舰队的集群，使开发团队对具体的Kubernetes资源拥有相应的操作权限。
- 9、10：拥有UCS CommonOperations权限的开发团队执行创建工作负载、流量分发等操作。
- 11：拥有UCS CIAOperations权限的运维团队执行监控运维等操作。

- 12：拥有UCS ReadOnlyAccess权限的访客进行集群、舰队、工作负载等资源的查看操作。

管理员：IAM 授权

Tenant Administrator管理员按照图5-11方式为各个职能团队进行IAM授权，即：先创建四个用户组，分别为这些用户组授予UCS FullAccess、UCS CommonOperations、UCS CIAOperations和UCS ReadOnlyAccess权限，然后为各个用户组添加用户。

图 5-11 IAM 授权方式



例如：为开发团队创建用户组dev，授予UCS CommonOperations权限，并添加devuser1、devuser2两个用户。

图 5-12 授权记录

图 5-13 用户管理

详细的操作指导请参见**UCS服务资源权限（IAM授权）**。需要注意的是，UCS的一些功能依赖其他云服务实现，在使用这些功能时，还需要授予其他云服务的权限。例如，创建权限时需要获取IAM用户列表，因此，为行管团队授予UCS FullAccess权限的同时还需授予VDC ReadOnlyAccess权限。

行管团队：搭建基础设施、配置权限策略

步骤1 创建权限。

为开发人员创建开发权限。

图 5-14 创建开发权限

步骤2 创建舰队并将权限关联至舰队。

舰队是多个集群的集合，舰队可以实现多集群的权限统一管理。行管人员关联上一步创建的开发权限至舰队，后续加入舰队的集群将拥有舰队的权限，这样开发人员就有操作舰队中集群资源（如创建工作负载）的权限了。详细的操作指导请参见**管理容器舰队**。

步骤3 注册集群，并将它们添加到舰队中。

UCS服务支持注册华为云集群、本地集群、多云集群及附着集群，行管人员可以根据实际需求选择。详细的操作指导请参见**华为云集群**、**附着集群概述**、**本地集群概述**或**多云集群概述**。

步骤4 开通集群联邦。

集群联邦可以提供：多集群统一编排、跨集群弹性伸缩、跨集群服务发现、应用故障自动迁移等能力。集群联邦开通后，舰队内的成员集群将自动接入联邦。

----结束

开发团队：创建工作负载、流量分发

行管人员将多集群联邦基础设施搭建完成后，开发人员就可以使用这些基础设施资源了，详细的操作指导请参见**工作负载**、**流量分发**。

运维团队：查看并监控资源使用情况

运维人员利用容器智能分析提供的智能分析、仪表盘、通知配置、7x24小时守护功能，实时监控工作负载资源，分析应用健康状态，以及完成其他日常运维工作。详细的操作指导请参见**容器智能分析**。

访客：查看资源

访客（仅具有查看资源权限的人员）可执行集群、舰队、工作负载等资源的查看操作。

6 策略中心

6.1 策略中心概述

随着公司不断增加开发和生产集群的数量，确保在日益扩大的环境中创建和执行一致的配置和安全策略变得越来越具挑战性，这可能会阻碍运维效率。为了解决这个问题，UCS推出了基于OPA（Open Policy Agent）的Gatekeeper实现的策略中心功能。这一功能可以帮助您在多个集群中定义和执行一致的策略，统一资源的合规性状态。

通过策略中心，您可以创建、管理和监控跨多个集群（容器舰队）的策略实施情况。这样，您可以确保所有集群都遵循相同的安全和合规要求，从而提高运维效率。这种集中式策略管理能力使您能够更容易地应对复杂的企业环境，同时确保所有资源在任何时候都处于合规状态。UCS策略中心将帮助您实现更高的运维效率和更强的安全性。

UCS策略中心的主要优势包括：

- 一致的策略管理
  以集中、一致的方式将一组安全合规策略应用到多个容器舰队、集群中。
- 确保资源安全性
  持续审计资源，以确保资源满足安全合规要求并且不出现违反策略的行为。
- 全局资源合规性视图
  涵盖所有资源的全面合规性视图，大规模保护和管理集群资源。

6.2 策略定义与策略实例的基本概念

策略定义

在创建策略实例之前，您需要先定义一个策略定义，它描述了强制执行约束的Rego代码和约束的模式。约束的模式允许管理员像调整函数参数一样微调约束的行为。策略定义中的Rego代码描述了强制执行的具体逻辑，根据您的需求来实现不同的合规性规则。而约束模式则提供了一种灵活的方式，让管理员在创建策略实例时，能够根据实际需求调整策略行为，以满足不同场景的合规控制需求。策略定义的详细说明请参见**官方文档**。

下面是一个示例策略定义，它要求所有在约束中描述的标签都必须存在。


```
apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredLabels
      validation:
        # Schema for the `parameters` field
        openAPIV3Schema:
          type: object
          properties:
            labels:
              type: array
              items:
                type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8srequiredlabels
        violation[{"msg": msg, "details": {"missing_labels": missing}}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("you must provide labels: %v", [missing])
        }
```

策略实例

策略实例用于告知Gatekeeper应执行哪个策略定义以及如何执行。策略实例的详细说明请参见**官方文档**。

下面是一个策略实例示例，该示例使用先前提到的K8sRequiredLabels策略定义，确保Gatekeeper在所有命名空间上强制执行指定的标签。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
spec:
  enforcementAction: deny
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    labels: ["gatekeeper"]
```

改策略实例示例指定了使用“K8sRequiredLabels”策略定义并将执行策略的操作设置为“deny”，这意味着Gatekeeper将拒绝违反策略的请求。“match”部分限制了此策略实例仅适用于命名空间资源。“parameters”部分指定了必须存在于资源上的标签，本例中为“gatekeeper”。

6.3 启用策略中心

当您首次使用策略中心功能时，需要执行“启用”操作。您可以选择为整个容器舰队启用该功能，或者仅为尚未加入舰队的集群启用。启用策略中心功能后，系统将自动为您选择的舰队或集群安装Gatekeeper插件。

约束与限制

- 仅**华为云账号**或具备**UCS FullAccess**权限的用户可进行策略中心的启用操作。
- 为非华为云集群启用策略中心前，请确保集群能够拉取公网镜像。
- 启用策略中心功能后，系统将在舰队或集群上安装Gatekeeper插件。需要注意的是，插件会占用部分集群资源（如表6-1所示）。因此，在启用策略中心功能之前，请确保您的集群具有足够的资源。这将有助于确保策略中心功能的顺利部署，同时避免对现有工作负载的性能产生负面影响。

  表 6-1 Gatekeeper 插件的资源占用情况

CPU	Mem
Requests：100m * 3 Limits：1000m * 3	Requests：256Mi * 3 Limits：512Mi * 3

  说明

  “* 3”表示有3个Pod。

- 当舰队或集群处于启用中的状态时，请避免在该舰队或集群上执行任何操作。在启用过程中执行操作可能会影响启用的成功。

操作步骤

步骤1 登录UCS控制台，在左侧导航栏中选择“策略中心”。

步骤2 单击页面中的“启用”按钮，弹出“启用策略管理功能”对话框。

步骤3 在下拉列表中选择容器舰队或集群，单击“确定”，返回主页面。

您将会看到舰队或集群的策略管理状态显示为启用中。请耐心等待大约3分钟，策略管理将成功启用。

如果在启用策略管理功能时出现“The thorttling threshold has been reached: policy ip over ratelimit”，说明因为启用集群较多被限流了，请稍等一会再重试即可。

----结束

6.4 创建和管理策略实例

策略实例是用于指导Gatekeeper执行特定策略定义以及执行方式的指令集。它们充当规则集合，帮助您在Kubernetes集群中强制执行安全策略和一致性。本节将指导您如何创建和管理策略实例。

前提条件

已为容器舰队或集群启用策略中心功能。

约束与限制

如果用户通过kubectl命令删除了集群中的策略实例，您需要在界面上先删除相应的策略实例，然后重新创建。这样，系统才会再次下发新的策略实例到集群中。

创建策略实例

步骤1 登录UCS控制台，在左侧导航栏中选择“策略中心”。

步骤2 在列表中找到已启用策略中心功能的容器舰队或集群，单击“创建策略实例”。

步骤3 填写如下参数：

图 6-1 创建策略实例

- **策略定义**：从内置的33个策略定义中选择一个，以便为您的集群或容器舰队配置资源审计规则。尽管当前不支持自定义策略定义，但这些预定义的策略定义基本可以满足您在合规性和安全性方面的需求。策略定义的详细介绍请参阅**策略定义库概述**。
- **策略执行方式**：包括拦截和告警两种方式。拦截表示不符合策略要求的资源将无法创建，告警表示不符合策略要求的资源仅告警提醒，仍可以正常创建。
- **策略生效范围**：选择生效的命名空间。

步骤4 单击“创建”，创建完成后系统会自动分发策略，如果分发成功，策略实例将在集群中生效。

策略实例分发成功后可在集群中执行符合策略实例的动作，此时该动作可正常执行；若在集群中执行不符合策略实例的动作，该动作将被拒绝掉或者上报告警事件。

----结束

修改/删除策略实例

作为平台工程师，您通常需要定期审视和更新策略实例，或者删除一些不再使用的策略实例。要执行这些操作，请参考以下步骤：

步骤1 登录UCS控制台，在左侧导航栏中选择“策略中心”。

步骤2 在列表中找到已启用策略中心功能的容器舰队或集群，单击舰队或集群名称，进入详情页面。

步骤3 在“策略实施详情”中单击“策略实例”页签。

步骤4 找到待操作的策略实例，单击操作列的“编辑”或“删除”，完成相关参数的修改，或者删除策略实例。

----结束

查看策略实施状态

在容器舰队或集群的策略详情页，可以查看策略实施详情和状态，以及舰队或集群的不合规资源、告警事件和强制拦截事件的情况。您可以根据这些数据评估集群的合规情况，并及时采取修复措施。

图 6-2 策略实施详情

说明

- 当前，不合规资源统计的上报时间大约在15至30分钟之间。
- 如果下发策略实例后未对违规资源进行拦截或告警，请检查是否已开启ValidatingAdmissionPolicy特性门控、是否已启用ValidatingAdmissionWebhook和MutatingAdmissionWebhook准入控制器，详情请参考**准入控制器的作用是什么?**

6.5 示例：使用策略中心实现 Kubernetes 资源合规性治理

假设您是一家大型企业的平台工程师，负责整个基础设施环境的安全策略配置和管理，确保企业多个团队使用集群资源的合规性。利用UCS策略中心，您可以：

- 创建统一的策略实例：使用UCS策略中心创建一个统一的策略实例，包含所有团队需要遵循的安全和合规规定。这样一来，您可以确保所有团队在使用集群资源时遵循相同的标准。
- 自动化策略部署：在UCS策略中心启用并创建策略实例后，系统会自动将这些策略应用到多个集群中，这将大大减少手动配置的时间和出错的可能性。
- 监控策略实施情况：UCS策略中心能够提供诸如集群合规率、不合规资源、告警事件数以及强制拦截事件等多种监控数据。这些数据有助于您迅速发现和解决策略实施过程中的问题。

本教程将向您展示如何使用策略中心来实现Kubernetes资源的合规性治理指南。操作流程如下：

启用策略中心

步骤1 登录UCS控制台，在左侧导航栏中选择“策略中心”。

步骤2 单击页面中的“启用”按钮，弹出“启用策略管理功能”对话框。

步骤3 在下拉列表中选择容器舰队或集群，单击“确定”，返回主页面。

您将会看到舰队或集群的策略管理状态显示为启用中。请耐心等待大约3分钟，策略管理将成功启用。

----结束

创建策略实例

步骤1 登录UCS控制台，在左侧导航栏中选择“策略中心”。

步骤2 在列表中找到已启用策略中心功能的容器舰队或集群，单击“创建策略实例”。

步骤3 填写如下参数：

图 6-3 创建策略实例

- **策略定义**：从内置的33个策略定义中选择一个，本教程以“k8srequiredlabels”为例，该策略定义的作用是要求资源包含指定的标签，其值与提供的正则表达式匹配。这里设置标签的key为owner，正则表达式为^[a-zA-Z]+.agilebank.demo$。
- **策略执行方式**：包括拦截和告警两种方式。拦截表示不符合策略要求的资源将无法创建，告警表示不符合策略要求的资源仅告警提醒，仍可以正常创建。本教程以“拦截”执行方式为例。

- **策略生效范围**：选择生效的命名空间。本教程以“default”命名空间为例。

步骤4 单击“创建”，创建完成后系统会自动分发策略，如果分发成功，策略实例将在集群中生效。

----结束

验证策略实例是否生效

策略实例分发成功后可在集群中执行符合策略实例的动作，此时该动作可正常执行；若在集群中执行不符合策略实例的动作，该动作将被拒绝掉（取决于设置的策略执行方式）。

尝试在集群中创建一个Pod，定义标签为：owner: user.agilebank.demo，符合策略实例，Pod可以创建成功。

如果在创建Pod时不包含策略实例中定义的标签，则Pod创建不成功，同时在“不合规资源”页签会生成相应的记录。

6.6 使用策略定义库

6.6.1 策略定义库概述

UCS为您提供了预置的策略定义库，通过这个库，您可以创建具体的策略实例，进而将策略实例定义细节的任务委托给具备专业知识的个人或团队。这种做法不仅实现了关注点的隔离，还将策略实例的逻辑与定义进行了分离。

为了帮助您更好地理解策略定义的工作原理，每个预置策略定义都包含以下三个部分：一个示例策略实例，用于展示如何使用该策略定义；一个违反策略实例的资源定义，用于说明不符合该策略要求的资源样例；一个符合策略实例的资源定义，用于展示满足策略要求的资源样例。这些内容将帮助您更清晰地了解各种策略定义的应用场景及其执行标准。

所有策略实例都包含一个“match”部分，这部分定义了策略实例应用的目标对象。通过“match”部分，您可以精准指定策略实例适用的资源类型、命名空间或其他特定条件，从而确保策略实例仅对满足这些条件的对象起作用。

表6-2为安全类策略定义，共16个，它们专注于确保集群和资源的安全性；表6-3为合规类策略定义，总共17个，它们针对不同方面的合规要求。

表 6-2 安全类策略定义

策略定义名称	类型	推荐级别	生效对象	参数
k8spspvolumetypes	安全	L3	Pod	volumes：数组

策略定义名称	类型	推荐级别	生效对象	参数
k8spspallowedusers	安全	L3	Pod	exemptImages：字符串数组 runAsUser • rule：字符串 • ranges – min：整型 – max：整型 runAsGroup • rule：字符串 • ranges – min：整型 – max：整型 supplementalGroups • rule：字符串 • ranges – min：整型 – max：整型 fsGroup • rule：字符串 • ranges – min：整型 – max：整型
k8spspselinuxv2	安全	L3	Pod	allowedSELinuxOptions：对象数组，包含level、role、type、user四个字符串对象 exemptImages：字符串数组
k8spspseccomp	安全	L3	Pod	allowedLocalhostFiles：数组 allowedProfiles：数组 exemptImages：字符串数组
k8spspreadonlyrootfilesystem	安全	L3	Pod	exemptImages：字符串数组
k8spspprocmount	安全	L3	Pod	exemptImages：字符串数组 procMount：字符串
k8spspprivilegedcontainer	安全	L3	Pod	exemptImages：字符串数组

策略定义名称	类型	推荐级别	生效对象	参数
k8spsphostnetworkingports	安全	L3	Pod	exemptImages：字符串数组 hostNetwork • max：整型 • min：整型
k8spsphostnamespace	安全	L3	Pod	无
k8spsphostfilesystem	安全	L3	Pod	allowedHostPaths • pathPrefix：字符串
k8spspfsgroup	安全	L3	Pod	rule：字符串，支持MayRunAs、MustRunAs和RunAsAny ranges • max：整型 • min：整型
k8spspforbiddensysctls	安全	L3	Pod	allowedSysctls：数组 forbiddenSysctls：数组
k8spspflexvolumes	安全	L3	Pod	allowedFlexVolumes：数组
k8spspcapabilities	安全	L3	Pod	allowedCapabilities：数组 exemptImages：字符串数组 requiredDropCapabilities：数组
k8spspapparmor	安全	L3	Pod	allowedProfiles：数组 exemptImages：字符串数组
k8spspallowprivilegeescalationcontainer	安全	L3	Pod	exemptImages：字符串数组

表 6-3 合规类策略定义

策略定义名称	类型	推荐级别	生效对象	参数
k8srequiredprobes	合规	L1	Pod	probes：数组 probeTypes：数组

策略定义名称	类型	推荐级别	生效对象	参数
k8srequiredlabels	合规	L1	Deployment	labels • key / allowedRegex：键值对数组 message：字符串
k8srequiredannotations	合规	L1	Pod	annotations • key / allowedRegex：键值对数组 message：字符串
k8sreplicalimits	合规	L1	Deployment、ReplicaSet、CronJob	ranges • min_replicas：整型 • max_replicas：整型
noupdateserviceaccount	合规	L1	Pod	allowedGroups：数组 allowedUsers：数组
k8simagedigests	合规	L1	Pod	exemptImages：字符串数组
k8sexternalips	合规	L1	Service	allowedIPs：字符串数组
k8sdisallowedtags	合规	L1	Pod	tags：字符串数组 exemptImages：字符串数组
k8srequiredresources	合规	L1	Pod	exemptImages：字符串数组 limits • cpu • memory requests • cpu • memory
k8scontainerratios	合规	L1	Pod	ratio：字符串 cpuRatio：字符串 exemptImages：字符串数组
k8scontainerrequests	合规	L1	Pod	cpu：字符串 memory：字符串 exemptImages：字符串数组
k8scontainerlimits	合规	L1	Pod	cpu：字符串 memory：字符串 exemptImages：字符串数组

策略定义名称	类型	推荐级别	生效对象	参数
k8sblockwildcardingress	合规	L1	Ingress	无
k8sblocknodeport	合规	L1	Service	无
k8sblockloadbalancer	合规	L1	Pod	无
k8spspautomountserviceaccounttokenpod	合规	L1	Pod	无
k8sallowedrepos	合规	L1	Pod	repos：字符串数组

6.6.2 k8spspvolumetypes

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  volumes：数组

作用

约束PodSecurityPolicy中的“volumes”字段类型。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters的volumes字段定义了允许的类型列表。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPVolumeTypes
metadata:
  name: psp-volume-types
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    volumes:
    # - "*" # * may be used to allow all volume types
    - configMap
    - emptyDir
```

```
  - projected
  - secret
  - downwardAPI
  - persistentVolumeClaim
  #- hostPath #required for allowedHostPaths
  - flexVolume #required for allowedFlexVolumes
```

符合策略实例的资源定义

示例中volumes中的类型均在上述定义的允许范围内，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-volume-types-allowed
  labels:
    app: nginx-volume-types
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  - name: nginx2
    image: nginx
    volumeMounts:
    - mountPath: /cache2
      name: demo-vol
  volumes:
  - name: cache-volume
    emptyDir: {}
  - name: demo-vol
    emptyDir: {}
```

不符合策略实例的资源定义

示例中volumes中的类型（hostPath）不在上述定义的允许范围内，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-volume-types-disallowed
  labels:
    app: nginx-volume-types
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  - name: nginx2
    image: nginx
    volumeMounts:
    - mountPath: /cache2
      name: demo-vol
  volumes:
  - name: cache-volume
    hostPath:
      path: /tmp # directory location on host
  - name: demo-vol
    emptyDir: {}
```

6.6.3 k8spspallowedusers

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

```
exemptImages: 字符串数组
runAsUser:
  rule: 字符串
  ranges:
    - min: 整型
      max: 整型
runAsGroup:
  rule: 字符串
  ranges:
    - min: 整型
      max: 整型
supplementalGroups:
  rule: 字符串
  ranges:
    - min: 整型
      max: 整型
fsGroup:
  rule: 字符串
  ranges:
    - min: 整型
      max: 整型
```

作用

约束PodSecurityPolicy中的runAsUser、runAsGroup、supplementalGroups和fsGroup字段。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中定义了对runAsUser、runAsGroup、supplementalGroups和fsGroup等字段的约束。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAllowedUsers
metadata:
  name: psp-pods-allowed-user-ranges
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    runAsUser:
      rule: MustRunAs # MustRunAsNonRoot # RunAsAny
      ranges:
        - min: 100
          max: 200
    runAsGroup:
      rule: MustRunAs # MayRunAs # RunAsAny
      ranges:
        - min: 100
          max: 200
    supplementalGroups:
      rule: MustRunAs # MayRunAs # RunAsAny
```

```
    ranges:
      - min: 100
        max: 200
  fsGroup:
    rule: MustRunAs # MayRunAs # RunAsAny
    ranges:
      - min: 100
        max: 200
```

符合策略实例的资源定义

示例中runAsUser等参数均在范围内，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-users-allowed
  labels:
    app: nginx-users
spec:
  securityContext:
    supplementalGroups:
      - 199
    fsGroup: 199
  containers:
    - name: nginx
      image: nginx
      securityContext:
        runAsUser: 199
        runAsGroup: 199
```

不符合策略实例的资源定义

示例中runAsUser等参数不在范围内，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-users-disallowed
  labels:
    app: nginx-users
spec:
  securityContext:
    supplementalGroups:
      - 250
    fsGroup: 250
  containers:
    - name: nginx
      image: nginx
      securityContext:
        runAsUser: 250
        runAsGroup: 250
```

6.6.4 k8spspselinuxv2

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  allowedSELinuxOptions：对象数组，包含level、role、type、user四个字符串对象

exemptImages：字符串数组

作用

约束Pod定义SELinux配置的允许列表。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中的allowedSELinuxOptions定义了参数的允许列表。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPSELinuxV2
metadata:
  name: psp-selinux-v2
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedSELinuxOptions:
      - level: s0:c123,c456
        role: object_r
        type: svirt_sandbox_file_t
        user: system_u
```

符合策略实例的资源定义

示例中seLinuxOptions参数均在参数列表中，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx-selinux-allowed
    labels:
        app: nginx-selinux
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      seLinuxOptions:
        level: s0:c123,c456
        role: object_r
        type: svirt_sandbox_file_t
        user: system_u
```

不符合策略实例的资源定义

示例中seLinuxOptions参数不在参数列表中，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx-selinux-disallowed
    labels:
        app: nginx-selinux
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      seLinuxOptions:
        level: s1:c234,c567
```



```
    user: sysadm_u
    role: sysadm_r
    type: svirt_lxc_net_t
```

6.6.5 k8spspseccomp

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  allowedLocalhostFiles：数组

  allowedProfiles：数组

  exemptImages：字符串数组

作用

约束PodSecurityPolicy上的“seccomp.security.alpha.kubernetes.io/allowedProfileNames”注解。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中allowedProfiles定义了允许的注解。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPSeccomp
metadata:
  name: psp-seccomp
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedProfiles:
    - runtime/default
    - docker/default
```

符合策略实例的资源定义

示例中的container.seccomp.security.alpha.kubernetes.io/nginx注解的value在设定的值列表中，符合策略定义。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-seccomp-allowed
  annotations:
    container.seccomp.security.alpha.kubernetes.io/nginx: runtime/default
  labels:
    app: nginx-seccomp
spec:
  containers:
  - name: nginx
    image: nginx
```


不符合策略实例的资源定义

示例中的container.seccomp.security.alpha.kubernetes.io/nginx注解的value没在设定的值列表中，不符合策略定义。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-seccomp-disallowed
  annotations:
    container.seccomp.security.alpha.kubernetes.io/nginx: unconfined
  labels:
    app: nginx-seccomp
spec:
  containers:
  - name: nginx
    image: nginx
```

6.6.6 k8spspreadonlyrootfilesystem

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  exemptImages：字符串数组

作用

约束PodSecurityPolicy中的“readOnlyRootFilesystem”字段。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPReadOnlyRootFilesystem
metadata:
  name: psp-readonlyrootfilesystem
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
```

符合策略实例的资源定义

示例中readOnlyRootFilesystem字段为true，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-readonlyrootfilesystem-allowed
  labels:
    app: nginx-readonlyrootfilesystem
spec:
  containers:
  - name: nginx
    image: nginx
```

```
  securityContext:
    readOnlyRootFilesystem: true
```

不符合策略实例的资源定义

示例中readOnlyRootFilesystem字段为false，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-readonlyrootfilesystem-disallowed
  labels:
    app: nginx-readonlyrootfilesystem
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      readOnlyRootFilesystem: false
```

6.6.7 k8spspprocmount

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  exemptImages：字符串数组

  procMount：字符串

作用

约束PodSecurityPolicy中的“allowedProcMountTypes”字段。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中指定了procMount的值为Default。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPProcMount
metadata:
  name: psp-proc-mount
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    procMount: Default
```

符合策略实例的资源定义

示例中securityContext字段中的procMount为Default，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
```

```
  name: nginx-proc-mount-disallowed
  labels:
    app: nginx-proc-mount
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      procMount: Default
```

不符合策略实例的资源定义

示例中securityContext字段中的procMount为Unmasked，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-proc-mount-disallowed
  labels:
    app: nginx-proc-mount
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      procMount: Unmasked
```

6.6.8 k8spspprivilegedcontainer

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  exemptImages：字符串数组

作用

禁止PodSecurityPolicy中的“privileged”字段为true。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPPrivilegedContainer
metadata:
  name: psp-privileged-container
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    excludedNamespaces: ["kube-system"]
```

符合策略实例的资源定义

示例中privileged设置为false，符合策略实例。


```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privileged-allowed
  labels:
    app: nginx-privileged
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: false
```

不符合策略实例的资源定义

示例中privileged设置为true，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privileged-disallowed
  labels:
    app: nginx-privileged
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
```

6.6.9 k8spsphostnetworkingports

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

```
exemptImages: 字符串数组
hostNetwork:
  max: 整型
  min: 整型
```

作用

约束PodSecurityPolicy中的“hostNetwork”和“hostPorts”字段。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中hostNetwork为true时，使用的端口必须在指定的端口范围内。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostNetworkingPorts
metadata:
  name: psp-host-network-ports
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
```

```
    hostNetwork: bool
    min: 80
    max: 9000
```

符合策略实例的资源定义

示例中hostNetwork设置成了false，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-networking-ports-allowed
  labels:
    app: nginx-host-networking-ports
spec:
  hostNetwork: false
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 9000
      hostPort: 80
```

不符合策略实例的资源定义

示例中hostNetwork设置成了true，但是端口未在指定范围内，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-networking-ports-disallowed
  labels:
    app: nginx-host-networking-ports
spec:
  hostNetwork: true
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 9001
      hostPort: 9001
```

6.6.10 k8spsphostnamespace

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：无

作用

限制PodSecurityPolicy中的“hostPID”和“hostIPC”字段。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostNamespace
```

```
metadata:
  name: psp-host-namespace
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
```

符合策略实例的资源定义

示例中hostPID和hostIPC均为false，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-namespace-allowed
  labels:
    app: nginx-host-namespace
spec:
  hostPID: false
  hostIPC: false
  containers:
  - name: nginx
    image: nginx
```

不符合策略实例的资源定义

示例中hostPID和hostIPC均为true，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-namespace-disallowed
  labels:
    app: nginx-host-namespace
spec:
  hostPID: true
  hostIPC: true
  containers:
  - name: nginx
    image: nginx
```

6.6.11 k8spsphostfilesystem

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  ```
  allowedHostPaths:
    readOnly: 布尔值
    pathPrefix: 字符串
  ```

作用

约束PodSecurityPolicy中的“hostPath”字段的参数。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中的allowedHostPaths指定了字段的值。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostFilesystem
metadata:
  name: psp-host-filesystem
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedHostPaths:
    - readOnly: true
      pathPrefix: "/foo"
```

符合策略实例的资源定义

示例中hostPath中pathPrefix以/foo开头，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-filesystem
  labels:
    app: nginx-host-filesystem-disallowed
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - mountPath: /cache
          name: cache-volume
          readOnly: true
  volumes:
    - name: cache-volume
      hostPath:
        path: /foo/bar
```

不符合策略实例的资源定义

示例中hostPath中pathPrefix以/tmp开头，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-filesystem
  labels:
    app: nginx-host-filesystem-disallowed
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
      readOnly: true
  volumes:
  - name: cache-volume
    hostPath:
      path: /tmp # directory location on host
```

6.6.12 k8spspfsgroup

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

```
rule: 字符串，支持MayRunAs、MustRunAs和RunAsAny
ranges
  max: 整型
  min: 整型
```

作用

控制PodSecurityPolicy中的“fsGroup”字段在限制范围内。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPFSGroup
metadata:
  name: psp-fsgroup
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    rule: "MayRunAs" #"MustRunAs" #"MayRunAs", "RunAsAny"
    ranges:
    - min: 1
      max: 1000
```

符合策略实例的资源定义

示例中fsGroup设为了500，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: fsgroup-disallowed
spec:
  securityContext:
    fsGroup: 500 # directory will have group ID 500
  volumes:
    - name: fsgroup-demo-vol
      emptyDir: {}
  containers:
    - name: fsgroup-demo
      image: busybox
      command: ["sh", "-c", "sleep 1h"]
      volumeMounts:
        - name: fsgroup-demo-vol
          mountPath: /data/demo
```

不符合策略实例的资源定义

示例中fsGroup设为了2000，不符合策略实例。


```
apiVersion: v1
kind: Pod
metadata:
  name: fsgroup-disallowed
spec:
  securityContext:
    fsGroup: 2000   # directory will have group ID 2000
  volumes:
  - name: fsgroup-demo-vol
    emptyDir: {}
  containers:
  - name: fsgroup-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: fsgroup-demo-vol
      mountPath: /data/demo
```

6.6.13 k8spspforbiddensysctls

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：
  allowedSysctls：数组
  forbiddenSysctls：数组

作用

约束PodSecurityPolicy中的“sysctls”字段不能使用的name。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中的forbiddenSysctls定义了sysctls中不能允许的名称。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPForbiddenSysctls
metadata:
  name: psp-forbidden-sysctls
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    forbiddenSysctls:
    # - "*" # * may be used to forbid all sysctls
    - kernel.*
```

符合策略实例的资源定义

示例中sysctls的name符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-forbidden-sysctls-disallowed
```



```
  labels:
    app: nginx-forbidden-sysctls
spec:
  containers:
    - name: nginx
      image: nginx
  securityContext:
    sysctls:
      - name: net.core.somaxconn
        value: "1024"
```

不符合策略实例的资源定义

示例中sysctls的name（ kernel.msgmax ）不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-forbidden-sysctls-disallowed
  labels:
    app: nginx-forbidden-sysctls
spec:
  containers:
    - name: nginx
      image: nginx
  securityContext:
    sysctls:
      - name: kernel.msgmax
        value: "65536"
      - name: net.core.somaxconn
        value: "1024"
```

6.6.14 k8spspflexvolumes

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：
  allowedFlexVolumes：数组

作用

约束PodSecurityPolicy中的allowedFlexVolumes字段类型。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中的allowedFlexVolumes字段定义了允许的driver类型列表。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPFlexVolumes
metadata:
  name: psp-flexvolume-drivers
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
```



```
  allowedFlexVolumes: #[]
  - driver: "example/lvm"
  - driver: "example/cifs"
```

符合策略实例的资源定义

示例中flexVolume中的类型均在上述定义的允许范围内，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-flexvolume-driver-allowed
  labels:
    app: nginx-flexvolume-driver
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /test
      name: test-volume
      readOnly: true
  volumes:
  - name: test-volume
    flexVolume:
      driver: "example/lvm"
```

不符合策略实例的资源定义

示例中flexVolume中的类型不在上述定义的允许范围内，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-flexvolume-driver-disallowed
  labels:
    app: nginx-flexvolume-driver
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /test
      name: test-volume
      readOnly: true
  volumes:
  - name: test-volume
    flexVolume:
      driver: "example/testdriver" #"example/lvm"
```

6.6.15 k8spspcapabilities

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：
  allowedCapabilities：数组
  exemptImages：字符串数组
  requiredDropCapabilities：数组

作用

限制PodSecurityPolicy中的“allowedCapabilities”和“requiredDropCapabilities”字段。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中定义了allowedCapabilities和requiredDropCapabilities的列表。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPCapabilities
metadata:
  name: capabilities-demo
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
  parameters:
    allowedCapabilities: ["something"]
    requiredDropCapabilities: ["must_drop"]
```

符合策略实例的资源定义

示例capabilities中的各项参数符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      securityContext:
        capabilities:
          add: ["something"]
          drop: ["must_drop", "another_one"]
      resources:
        limits:
          cpu: "100m"
          memory: "30Mi"
```

不符合策略实例的资源定义

示例capabilities中的参数不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
```

```
    image: openpolicyagent/opa:0.9.2
    args:
      - "run"
      - "--server"
      - "--addr=localhost:8080"
    securityContext:
      capabilities:
        add: ["disallowedcapability"]
    resources:
      limits:
        cpu: "100m"
        memory: "30Mi"
```

6.6.16 k8spspapparmor

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：
  allowedProfiles：数组
  exemptImages：字符串数组

作用

约束AppArmor字段列表。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters的allowedProfiles字段定义了允许的值列表。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAppArmor
metadata:
  name: psp-apparmor
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedProfiles:
    - runtime/default
```

符合策略实例的资源定义

示例中apparmor的值在上述定义的允许范围内，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-apparmor-allowed
  annotations:
    # apparmor.security.beta.kubernetes.io/pod: unconfined # runtime/default
    container.apparmor.security.beta.kubernetes.io/nginx: runtime/default
  labels:
    app: nginx-apparmor
spec:
```

```
  containers:
  - name: nginx
    image: nginx
```

不符合策略实例的资源定义

示例中apparmor的值不在上述定义的允许范围内，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-apparmor-disallowed
  annotations:
    # apparmor.security.beta.kubernetes.io/pod: unconfined # runtime/default
    container.apparmor.security.beta.kubernetes.io/nginx: unconfined
  labels:
    app: nginx-apparmor
spec:
  containers:
  - name: nginx
    image: nginx
```

6.6.17 k8spspallowprivilegeescalationcontainer

基本信息

- 策略类型：安全
- 推荐级别：L3
- 生效资源类型：Pod
- 参数：

  exemptImages：字符串数组

作用

约束PodSecurityPolicy中的“allowPrivilegeEscalation”字段为false。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAllowPrivilegeEscalationContainer
metadata:
  name: psp-allow-privilege-escalation-container
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
```

符合策略实例的资源定义

示例中allowPrivilegeEscalation的值为false，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privilege-escalation-allowed
  labels:
    app: nginx-privilege-escalation
```

```
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      allowPrivilegeEscalation: false
```

不符合策略实例的资源定义

示例中allowPrivilegeEscalation的值不为false，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-privilege-escalation-disallowed
  labels:
    app: nginx-privilege-escalation
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      allowPrivilegeEscalation: true
```

6.6.18 k8srequiredprobes

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Pod
- 参数：
  probes：数组
  probeTypes：数组

作用

要求Pod具有Readiness或Liveness Probe。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters展示了probes的类型和probeTypes。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredProbes
metadata:
  name: must-have-probes
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    probes: ["readinessProbe", "livenessProbe"]
    probeTypes: ["tcpSocket", "httpGet", "exec"]
```

符合策略实例的资源定义

Pod中有livenessProbe和readinessProbe，probeType为tcpSocket，符合策略实例。


```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod1
spec:
  containers:
  - name: tomcat
    image: tomcat
    ports:
    - containerPort: 8080
    livenessProbe:
      tcpSocket:
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 10
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
  volumes:
  - name: cache-volume
    emptyDir: {}
```

不符合策略实例的资源定义

Pod中有livenessProbe，但是没有定义probeType，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod1
spec:
  containers:
  - name: nginx-1
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    livenessProbe:
      # tcpSocket:
      #   port: 80
      # initialDelaySeconds: 5
      # periodSeconds: 10
    volumeMounts:
    - mountPath: /tmp/cache
      name: cache-volume
  - name: tomcat
    image: tomcat
    ports:
    - containerPort: 8080
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
  volumes:
  - name: cache-volume
    emptyDir: {}
```

6.6.19 k8srequiredlabels

基本信息

- 策略类型：合规
- 推荐级别：L1

- 生效资源类型：*
- 参数：

```
labels: 键值对数组，key/ allowedRegex
  key: a8r.io/owner
    # Matches email address or github user
    allowedRegex: ^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}|[a-z]{1,39})$
```

作用

要求资源包含指定的标签，其值与提供的正则表达式匹配。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中指定了提示信息message以及label的约束定义。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: all-must-have-owner
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    message: "All namespaces must have an `owner` label that points to your company username"
    labels:
      - key: owner
        allowedRegex: "^[a-zA-Z]+.agilebank.demo$"
```

符合策略实例的资源定义

示例包含策略实例中定义的label，符合策略实例。

```
apiVersion: v1
kind: Namespace
metadata:
  name: allowed-namespace
  labels:
    owner: user.agilebank.demo
```

不符合策略实例的资源定义

示例不包含策略实例中定义的label，不符合策略实例。

```
apiVersion: v1
kind: Namespace
metadata:
  name: disallowed-namespace
```

6.6.20 k8srequiredannotations

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：*
- 参数：


```
annotations: 键值对数组，key/ allowedRegex
 key: a8r.io/owner
   # Matches email address or github user
   allowedRegex: ^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}|[a-z]{1,39})$
```

作用

要求资源包含指定的annotations，其值与提供的正则表达式匹配。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters中指定了提示信息message以及annotations的约束定义。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredAnnotations
metadata:
  name: all-must-have-certain-set-of-annotations
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
  parameters:
    message: "All services must have a `a8r.io/owner` and `a8r.io/runbook` annotations."
    annotations:
      - key: a8r.io/owner
        # Matches email address or github user
        allowedRegex: ^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}|[a-z]{1,39})$
      - key: a8r.io/runbook
        # Matches urls including or not http/https
        allowedRegex: ^(http:\/\/www\.|https:\/\/www\.|http:\/\/|https:\/\/)?[a-z0-9]+([\-\.]{1}[a-z0-9]+)*\.[a-z]
{2,5}(:[0-9]{1,5})?(\/.*)?$
```

符合策略实例的资源定义

示例中的annotations符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: allowed-service
  annotations:
    a8r.io/owner: "dev-team-alfa@contoso.com"
    a8r.io/runbook: "https://confluence.contoso.com/dev-team-alfa/runbooks"
spec:
  ports:
  - name: http
    port: 80
    targetPort: 8080
  selector:
    app: foo
```

不符合策略实例的资源定义

示例中的annotations没有配置值，不符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: disallowed-service
spec:
  ports:
  - name: http
    port: 80
```



```
  targetPort: 8080
selector:
  app: foo
```

6.6.21 k8sreplicalimits

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：*
- 参数：

```
ranges:
  min_replicas: 整型
  max_replicas: 整型
```

作用

要求具有“spec.replicas”字段的对象（Deployments、ReplicaSets等）在定义的范围内。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，parameters定义范围为3到50。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sReplicaLimits
metadata:
  name: replica-limits
spec:
  match:
    kinds:
      - apiGroups: ["apps"]
        kinds: ["Deployment"]
  parameters:
    ranges:
    - min_replicas: 3
      max_replicas: 50
```

符合策略实例的资源定义

Replicas设为了3，符合策略实例。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: allowed-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

不符合策略实例的资源定义

Replicas设为了100，不符合策略实例。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: disallowed-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 100
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

6.6.22 noupdateserviceaccount

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：*
- 参数：
  allowedGroups：数组
  allowedUsers：数组

作用

拒绝白名单外的资源更新ServiceAccount。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，pararmeters中定义了允许的组列表allowedGroups和允许的用户列表allowedUsers。

```
# IMPORTANT: Before deploying this policy, make sure you allow-list any groups
# or users that need to deploy workloads to kube-system, such as cluster-
# lifecycle controllers, addon managers, etc. Such controllers may need to
# update service account names during automated rollouts (e.g. of refactored
# configurations). You can allow-list them with the allowedGroups and
# allowedUsers properties of the NoUpdateServiceAccount Constraint.
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: NoUpdateServiceAccount
metadata:
  name: no-update-kube-system-service-account
spec:
  match:
    namespaces: ["kube-system"]
    kinds:
    - apiGroups: [""]
      kinds:
      # You can optionally add "Pod" here, but it is unnecessary because
```



```
    # Pod service account immutability is enforced by the Kubernetes API.
    - "ReplicationController"
  - apiGroups: ["apps"]
    kinds:
    - "ReplicaSet"
    - "Deployment"
    - "StatefulSet"
    - "DaemonSet"
  - apiGroups: ["batch"]
    kinds:
    # You can optionally add "Job" here, but it is unnecessary because
    # Job service account immutability is enforced by the Kubernetes API.
    - "CronJob"
 parameters:
  allowedGroups: []
  allowedUsers: []
```

符合策略实例的资源定义

没有更新ServiceAccount，符合策略实例。

```
# Note: The gator tests currently require exactly one object per example file.
# Since this is an update-triggered policy, at least two objects are technically
# required to demonstrate it. Due to the gator requirement, we only have one
# object below. The policy should allow changing everything but the
# serviceAccountName field.
kind: Deployment
apiVersion: apps/v1
metadata:
  name: policy-test
  namespace: kube-system
  labels:
    app: policy-test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: policy-test-deploy
  template:
    metadata:
      labels:
        app: policy-test-deploy
    spec:
      # Changing anything except this field should be allowed by the policy.
      serviceAccountName: policy-test-sa-1
      containers:
      - name: policy-test
        image: ubuntu
        command:
        - /bin/bash
        - -c
        - sleep 99999
```

6.6.23 k8simagedigests

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Pod
- 参数：

  exemptImages：字符串数组

作用

容器镜像必须包含digest。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sImageDigests
metadata:
  name: container-image-must-have-digest
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
```

符合策略实例的资源定义

容器镜像包含digest，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/
opa:0.9.2@sha256:04ff8fce2afd1a3bc26260348e5b290e8d945b1fad4b4c16d22834c2f3a1814a
```

不符合策略实例的资源定义

容器镜像不包含digest，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
```

6.6.24 k8sexternalips

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Service
- 参数：
  allowedIPs：字符串数组

作用

限制服务externalIP仅为允许的IP地址列表。

策略实例示例

服务的externalIP仅允许allowedIPs中定义的IP。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sExternalIPs
metadata:
  name: external-ips
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
  parameters:
    allowedIPs:
      - "203.0.113.0"
```

符合策略实例的资源定义

externalIPs中的IP为允许列表中的IP，符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: allowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 8080
  externalIPs:
    - 203.0.113.0
```

不符合策略实例的资源定义

externalIPs中的IP不为允许列表中的IP，不符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: disallowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 8080
  externalIPs:
    - 1.1.1.1
```

6.6.25 k8sdisallowedtags

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Pod

- 参数：
  tags：字符串数组
  exemptImages：字符串数组

作用

约束容器镜像tag。

策略实例示例

以下策略实例展示了策略定义生效的资源类型，pararmeters中表示不允许容器镜像tag为latest。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sDisallowedTags
metadata:
  name: container-image-must-not-have-latest-tag
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
  parameters:
    tags: ["latest"]
    exemptImages: ["openpolicyagent/opa-exp:latest", "openpolicyagent/opa-exp2:latest"]
```

符合策略实例的资源定义

容器镜像tag不为latest，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
```

不符合策略实例的资源定义

容器镜像tag为latest，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed-2
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:latest
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
```


6.6.26 k8sdisallowanonymous

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：RoleBinding、ClusterRoleBinding
- 参数：
  allowedRoles：字符串数组

作用

不允许将白名单以外的ClusterRole和Role关联到system:anonymous User和system:unauthenticated Group。

策略实例示例

示例展示了ClusterRole和Role资源仅能关联到allowedRoles中定义的Role。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sDisallowAnonymous
metadata:
  name: no-anonymous
spec:
  match:
    kinds:
      - apiGroups: ["rbac.authorization.k8s.io"]
        kinds: ["ClusterRoleBinding"]
      - apiGroups: ["rbac.authorization.k8s.io"]
        kinds: ["RoleBinding"]
  parameters:
    allowedRoles:
      - cluster-role-1
```

符合策略实例的资源定义

ClusterRole关联到cluster-role-1 Role中，符合策略实例。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-role-1
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:unauthenticated
```

不符合策略实例的资源定义

ClusterRole关联到cluster-role-2 Role中，不符合策略实例。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
```

```
metadata:
  name: cluster-role-binding-2
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-role-2
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:unauthenticated
```

6.6.27 k8srequiredresources

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Pod
- 参数：

```
exemptImages：字符串数组
limits
  cpu
  memory
requests
  cpu
  memory
```

作用

约束容器资源使用。

策略实例示例

必须配置内存的Limit，CPU和内存的Request。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredResources
metadata:
  name: container-must-have-cpu-requests-memory-limits-and-requests
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    limits:
      - memory
    requests:
      - cpu
      - memory
```

符合策略实例的资源定义

已经配置内存的Limit，CPU和内存的Request，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
```

```
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        limits:
          cpu: "100m"
          memory: "1Gi"
        requests:
          cpu: "100m"
          memory: "1Gi"
```

不符合策略实例的资源定义

没有配置内存的Limit，CPU和内存的Request，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        limits:
          memory: "2Gi"k8sexternalips
```

6.6.28 k8scontainerratios

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Service
- 参数：
  ratio：字符串
  cpuRatio：字符串
  exemptImages：字符串数组

作用

限制服务externalIP仅为允许的IP地址列表。

策略实例示例

服务的externalIP仅允许allowedIPs中定义的IP。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sExternalIPs
metadata:
  name: external-ips
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
  parameters:
    allowedIPs:
      - "203.0.113.0"
```

符合策略实例的资源定义

externalIPs中的IP为允许列表中的IP，符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: allowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 8080
  externalIPs:
    - 203.0.113.0
```

不符合策略实例的资源定义

externalIPs中的IP不为允许列表中的IP，不符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: disallowed-external-ip
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 8080
  externalIPs:
    - 1.1.1.1
```

6.6.29 k8scontainerrequests

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Pod
- 参数：

  cpu：字符串

  memory：字符串

exemptImages：字符串数组

作用

限制CPU和内存的Request必须设置且小于配置的最大值。

策略实例示例

示例配置了CPU和内存的最大Request。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sContainerRequests
metadata:
  name: container-must-have-requests
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    cpu: "200m"
    memory: "1Gi"
```

符合策略实例的资源定义

CPU和内存的Request小于配置的最大值，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        requests:
          cpu: "100m"
          memory: "1Gi"
```

不符合策略实例的资源定义

内存的Request大于约束的最大值，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-disallowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
```



```
    requests:
      cpu: "100m"
      memory: "2Gi"
```

6.6.30 k8scontainerlimits

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Pod
- 参数：

  cpu：字符串

  memory：字符串

  exemptImages：字符串数组

作用

限制容器必须设置CPU和内存Limit，并且小于设定的最大值。

策略实例示例

示例展示了匹配的对象的CPU最大为200m，内存最大为1G。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sContainerLimits
metadata:
  name: container-must-have-limits
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    cpu: "200m"
    memory: "1Gi"
```

符合策略实例的资源定义

CPU和内存的Limit符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
  labels:
    owner: me.agilebank.demo
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
      args:
        - "run"
        - "--server"
        - "--addr=localhost:8080"
      resources:
        limits:
          cpu: "100m"
          memory: "1Gi"
```

不符合策略实例的资源定义

内存Limit超过最大值，不符合策略实例。

6.6.31 k8sblockwildcardingress

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Ingress
- 参数：无

作用

禁止ingress配置空白或通配符类型的hostname。

策略实例示例

示例展示了该策略定义的生效类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockWildcardIngress
metadata:
  name: block-wildcard-ingress
spec:
  match:
    kinds:
      - apiGroups: ["extensions", "networking.k8s.io"]
        kinds: ["Ingress"]
```

符合策略实例的资源定义

ingress配置的hostname不是空白或通配符类型，符合策略实例。

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: non-wildcard-ingress
spec:
  rules:
  - host: 'myservice.example.com'
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
```

不符合策略实例的资源定义

ingress配置的hostname是空白，不符合策略实例。

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
```



```
  name: wildcard-ingress
spec:
  rules:
  - host: ''
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wildcard-ingress
spec:
  rules:
  # Omitted host field counts as a wildcard too
  - http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
```

ingress配置的hostname是包含通配符*，不符合策略实例。

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wildcard-ingress
spec:
  rules:
  - host: '*.example.com'
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: example
            port:
              number: 80
```

6.6.32 k8sblocknodeport

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Service
- 参数：无

作用

不允许Service为NodePort类型。

策略实例示例

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockNodePort
metadata:
  name: block-node-port
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Service"]
```

符合策略实例的资源定义

Servie类型非Nodeport，符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-disallowed
spec:
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

不符合策略实例的资源定义

Servie类型是Nodeport，不符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-disallowed
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

6.6.33 k8sblockloadbalancer

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：Service
- 参数：无

作用

不允许Service为LoadBalancer类型。

策略实例示例

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockLoadBalancer
metadata:
  name: block-load-balancer
spec:
```



```
match:
  kinds:
    - apiGroups: [""]
      kinds: ["Service"]
  excludedNamespaces:
    - "ingress-nginx-private"
    - "ingress-nginx-public"
```

符合策略实例的资源定义

Servie类型非LoadBalancer，符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-allowed
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 80
```

不符合策略实例的资源定义

Servie类型是LoadBalancer，不符合策略实例。

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-disallowed
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

6.6.34 k8sblockendpointeditdefaultrole

基本信息

- 策略类型：合规
- 推荐级别：L1
- 生效资源类型：ClusterRole
- 参数：无

作用

默认情况下，许多Kubernetes都预定义了一个名为system:aggregate-to-edit的ClusterRole，k8sblockendpointeditdefaultrole策略定义禁止该ClusterRole对Endpoints进行create、patch和update操作。

策略实例示例

以下策略实例展示了策略定义生效的资源类型。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sBlockEndpointEditDefaultRole
metadata:
  name: block-endpoint-edit-default-role
```

```
spec:
  match:
    kinds:
      - apiGroups: ["rbac.authorization.k8s.io"]
        kinds: ["ClusterRole"]
```

符合策略实例的资源定义

示例中ClusterRole的生效对象中没有endpoints，符合策略实例。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: null
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
  name: system:aggregate-to-edit
rules:
- apiGroups:
  - ""
  resources:
  - pods/attach
  - pods/exec
  - secrets
  - services/proxy
  verbs:
  - get
  - list
  - watch
```

不符合策略实例的资源定义

示例中ClusterRole的生效对象中有endpoints，不符合策略实例。

```
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: null
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
  name: system:aggregate-to-edit
rules:
- apiGroups:
  - apps
  resources:
  - endpoints
  verbs:
  - create
  - delete
  - deletecollection
  - patch
  - update
```

6.6.35 k8spspautomountserviceaccounttokenpod

基本信息

- 策略类型：合规
- 推荐级别：L1

- 生效资源类型：Pod
- 参数：无

作用

约束容器不能设置automountServiceAccountToken为true。

策略实例示例

示例声明了match匹配的对象不能把automountServiceAccountToken字段设为true。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPAutomountServiceAccountTokenPod
metadata:
  name: psp-automount-serviceaccount-token-pod
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    excludedNamespaces: ["kube-system"]
```

符合策略实例的资源定义

Pod的automountServiceAccountToken字段设为false，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-automountserviceaccounttoken-allowed
  labels:
    app: nginx-not-automountserviceaccounttoken
spec:
  automountServiceAccountToken: false
  containers:
  - name: nginx
    image: nginx
```

不符合策略实例的资源定义

Pod的automountServiceAccountToken字段设为true，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-automountserviceaccounttoken-disallowed
  labels:
    app: nginx-automountserviceaccounttoken
spec:
  automountServiceAccountToken: true
  containers:
  - name: nginx
    image: nginx
```

6.6.36 k8sallowedrepos

基本信息

- 策略类型：合规
- 推荐级别：L1

- 生效资源类型：Pod
- 参数：
  repos：字符串数组

作用

容器镜像必须以指定字符串列表中的字符串开头。

策略实例示例

以下策略实例定义容器镜像必须以“openpolicyagent/”开头。

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sAllowedRepos
metadata:
  name: repo-is-openpolicyagent
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    namespaces:
      - "default"
  parameters:
    repos:
      - "openpolicyagent/"
```

符合策略实例的资源定义

容器镜像以“openpolicyagent/”开头，符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: opa-allowed
spec:
  containers:
    - name: opa
      image: openpolicyagent/opa:0.9.2
```

不符合策略实例的资源定义

容器镜像以nginx开头，不符合策略实例。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-disallowed
spec:
  containers:
    - name: nginx
      image: nginx
```


7 配置管理

应用场景

在分布式集群场景下，为了方便用户对集群进行应用部署，实现自动化应用下发功能。UCS配置管理提供从仓库资源到Kubernetes集群自动部署应用配置的核心能力，通过采用**Kustomize组织和定制资源集的方式**配置仓库，提供对华为云集群、多云集群、本地集群和附着集群进行跨命名空间、跨集群、跨舰队的配置分发与配置管理的能力。对用户部署在各集群的业务提供实时的状态观测和消息通知，以确保应用出现问题时可以快速识别和定位，保障使用客户业务App的终端用户的使用体验和服务级别目标（SLO）达成。

说明

Kustomize是一个Kubernetes应用程序配置管理工具，可提供一种简单灵活的方式来生成Kubernetes资源，并可以使得这些资源在不同的环境中用不同的方式进行配置。 Kustomize还提供了一些钩子，可以在生成资源之前和之后执行操作，例如根据生成的资源更新其他文件。Kustomize使用一种名为Kustomization的格式来描述应用程序的配置。文件通常命名为Kustomization.yaml，并在应用程序的根目录下创建。

7.1 GitOps

GitOps 概述

GitOps是使用Git仓库来管理应用的部署模板，将Git仓库作为Kubernetes集群中部署应用的唯一来源，实现应用的持续部署，实现多集群的GitOps持续交付，满足应用的高可用部署、系统组件多集群分发等需求。GitOps假设每一个基础设施都被表示为一个具有版本控制功能的存储系统中的文件，并且有一个自动化的过程可以无缝地将更改的应用同步到应用程序运行环境。

而结合Kubernetes生态中的声明式API、Controller Loop可以更好得实现这一思想，该系统从一开始就遵循声明性规范以及最终一致性和收敛性的原则。

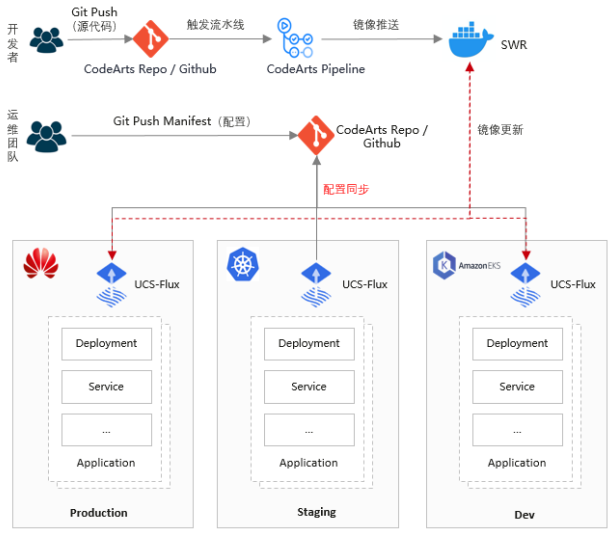
约束限制

为非华为云集群启用配置管理能力前，请确保集群能够拉取公网镜像。

GitOps 实现方式

- 开发运维人员基于Git工作流，可将现有流程，从应用开发扩展到到部署、应用生命周期管理和基础架构配置，开箱即用，客户无须运维Gitops工具。
- Gitops插件通过内置Kustomize结合Base/overlay制品组织方式和HelmRelease结合valuesFrom/valuesFiles等方式的能力，满足客户差异化的配置管理诉求。
- 将Git仓库中最新合入的制品配置信息同步部署至多个集群中，同时对应用发布行为进行版本化管理和权限控制，提供发布回滚和版本迭代控制，并进行审计跟踪。
- 所需的基础架构状态会自动应用于基础架构，而无需任何手动干预，持续监控并确保基础架构始终遵循Git存储库中的配置，确保基础设施处于理想状态。

图 7-1 GitOps 实现方式



GitOps 优势

- 简单易学：Git易于被开发者接受，易于集成，无需额外学习成本。
- 安全性高：开发者使用GitOps无需任何Kubernetes集群权限，仅需要Git仓库权限，保证集群安全可靠。
- 可靠性强：提供原生Kubernetes资源、Helm Chart资源、Kustomize等资源交付清单的版本管理，方便用户进行部署应用、增量变化和应用配置的回滚。
- 应用持续部署：Kubernetes集群和Git仓库中的应用状态自动同步，保持一致，实现应用持续部署。

GitOps 价值

- 提供原生Kubernetes资源、Helm Chart资源、Kustomize等资源交付清单的版本管理，方便用户进行部署应用、增量变化和应用配置的回滚。
- 更精细的多集群、多环境差异化配置体验：
  - 复用同一个应用组件（如多个业务线都对数据库的连接池模板复用）的交付模板，形成最佳实践模板。
  - 进行更灵活的标签替换、字符串、版本号替换/参数的动态嵌入/Patch操作。

7.2 创建配置集合

背景

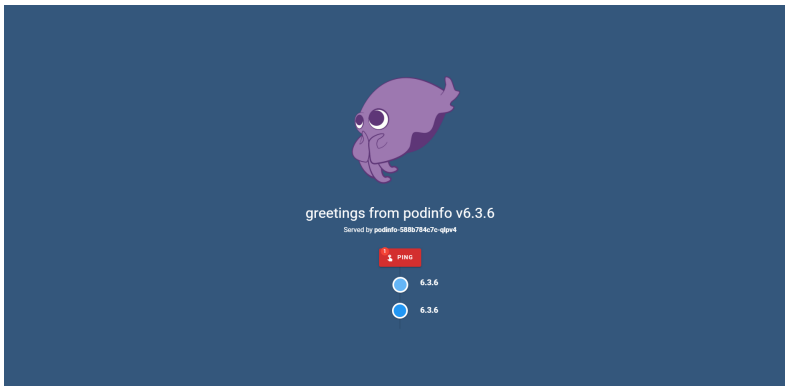
Podinfo是一个微模型的Web应用程序，它展示了在Kubernetes中运行微服务的最佳实践，其主要用作于测试和研讨。本章将用podinfo源代码来做创建配置集合的示例。

为了可以更快的、更稳定的持续地交付软件、减少后续维护工作，所以将podinfo的源代码放入**GitHub仓库**，并通过创建配置集合的方式部署到集群中，通过GitOps能力实现软件自动化部署，具体请参考**操作步骤**。

须知

- 创建podinfo源代码仓时，请先注册一个属于自己的GitHub账号，然后将podinfo所有代码fork到自己的GitHub仓库中。
- 在git仓库中定义交付资源清单文件时，不应包含敏感信息（如数据库连接密钥等）。相关敏感信息应以环境变量、加密存储的Secret等方式进行存储。

图 7-2 Podinfo 界面



操作步骤

步骤1 登录华为云控制台。

步骤2 在左侧导航栏中选择“分布式云原生”，选择“配置管理”。

步骤3 在右上角“添加集群”，选择需要启用配置管理功能的目标集群，单击确定。

步骤4 在集群概览页，选择目标集群，单击“Gitops能力”，查看Gitops插件（名称：集群名称-FluxPlugin）是否安装成功。当插件部署状态显示运行中，表示插件已部署成功。

图 7-3 集群概览页

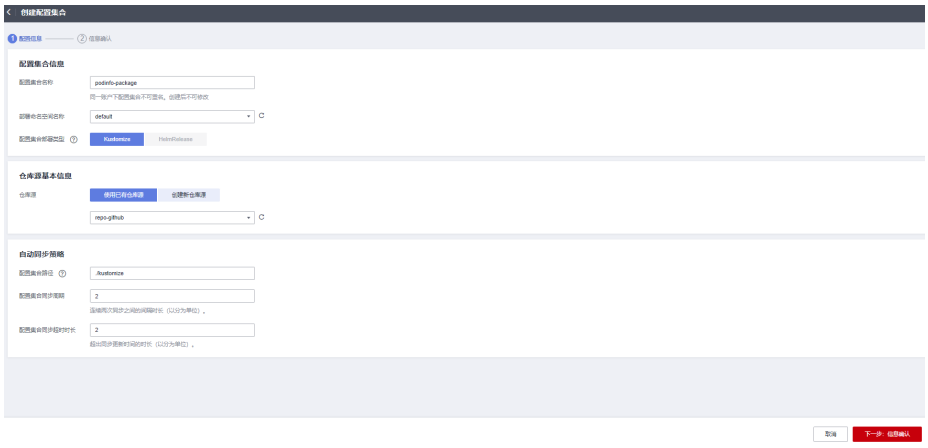
步骤5 选择“配置集合”页签，单击创建配置集合。

步骤6 选择仓库源，如果已有仓库源请参考**使用已有仓库源配置**，如果需要创建新仓库源，请参考**创建新仓库源**。

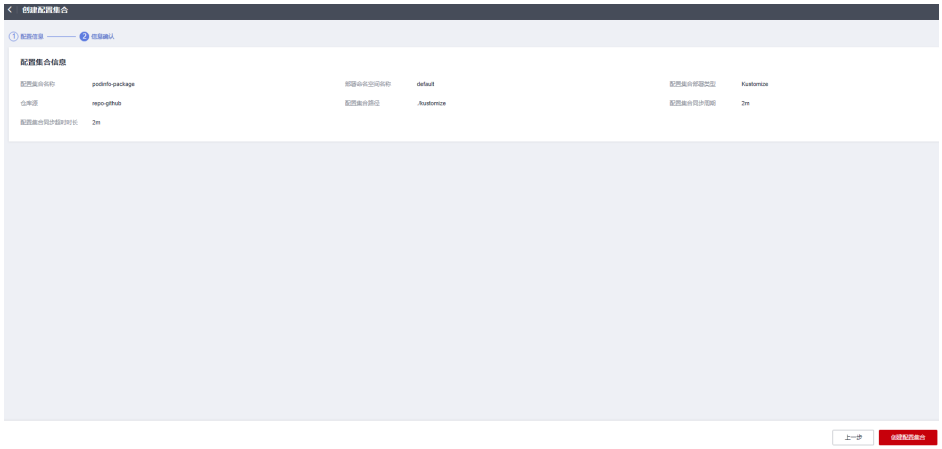
----结束

使用已有仓库源配置

步骤1 输入配置集合名称，选择部署空间名称，选择“使用已有仓库源”，选择已有仓库，在“自动同步策略”下输入“配置集合路径”（该仓库源中要同步的配置集合在结构设置上的顶层路径），单击“下一步：信息确认”；



步骤2 确认配置信息无误后，单击“创建配置集合”；如有问题，单击上一步进行修改。



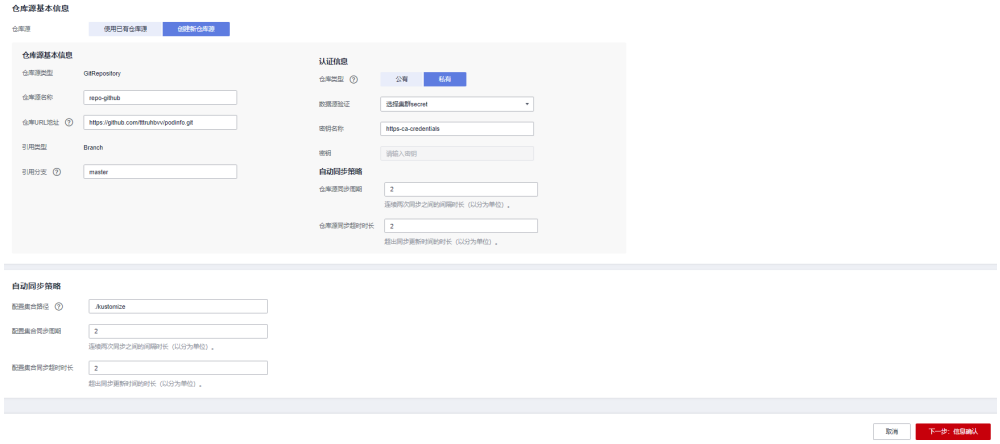
----结束

创建新仓库源

步骤1 单击“创建新仓库源”输入仓库源名称、仓库源URL地址。

步骤2 输入需要与其同步的代码库分支。

步骤3 选择数据源验证，以及输入密钥。



说明

- 选择公有类型的仓库无需进行身份验证，即可提供只读权限。
- 选择私有类型的仓库，则数据源验证可选择“选择集群secret”和“提供认证信息(SSH)”，两种方式都需要配置的秘钥进行身份验证。
- 仓库秘钥创建请参考**秘钥**。

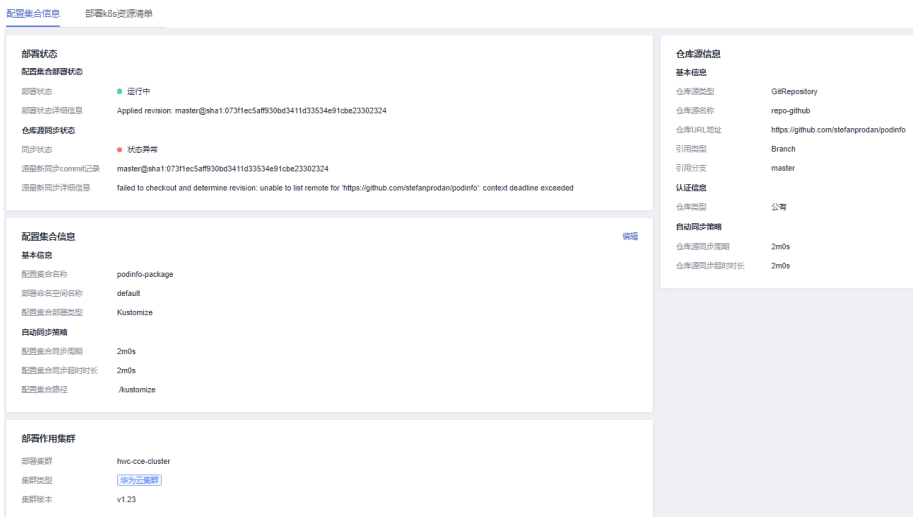
步骤4 仓库源创建完成后，在“自动同步策略”内输入“配置集合路径”，单击“下一步：信息确认”。

步骤5 确认配置信息无误后，单击“创建配置集合”；如有问题，单击上一步进行修改。

----结束

查看配置集合信息

步骤1 单击集群名称，进入配置管理界面，单击配置名称，即可查看配置集合信息。



步骤2 单击“部署k8s资源清单”，可查看配置集合部署资源内容；单击“查看详情”，可查看应用的详细信息。

----结束

7.3 修改源代码

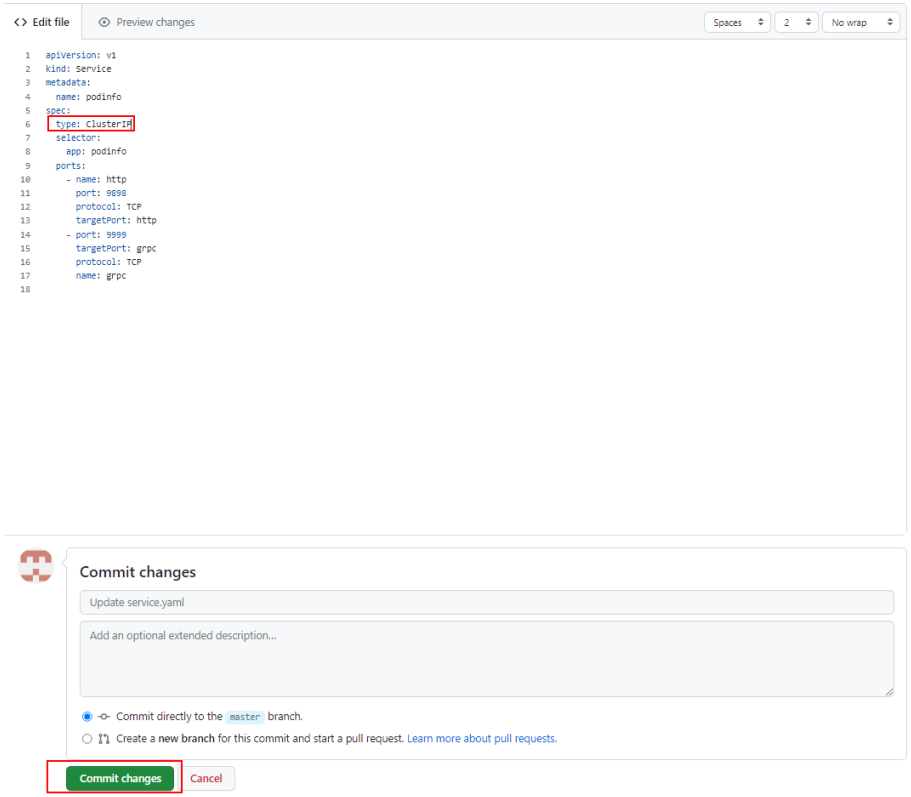
修改应用服务

如图7-4，现需要将集群下podinfo服务的访问类型由“集群内访问”改为“节点访问”，将其端口暴露到现网，具体操作如下：

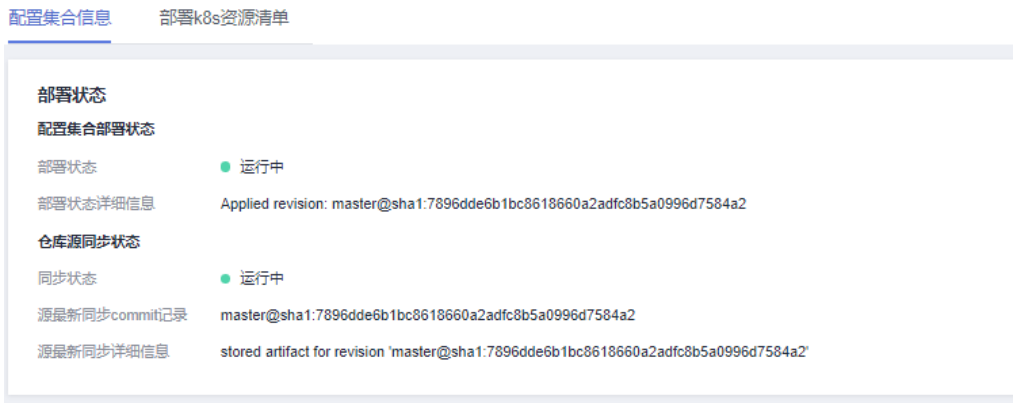
图 7-4 服务列表

步骤1 进入配置集合源代码仓库，根据根据配置集合仓库源信息，找到并打开podinfo/kustomize路径下的service.yaml文件，单击  ，将“type: ClusterIP”修改为“type: NodePort”，单击 

保存。



步骤2 登录华为云控制台，在左侧导航栏选择“分布式云原生”，选择“配置管理”，进入被修改的配置集合所在集群，单击配置集合名，查看“配置集合信息”页面，当“仓库源同步状态”为“运行中”后，表示仓库源代码已同步完成，单击“部署k8s资源清单”。



说明

修改完仓库源代码后，集群需要重新拉取部署该应用，该过程可能需要等待几分钟。

步骤3 在“部署k8s资源清单”页面下，选择资源类型为“Service”的“podinfo”资源，单击“查看详情”。

步骤4 进入“服务列表”页面下，查看“podinfo”的“访问端口 -> 容器端口 / 协议”生成的端口号。

步骤5 输入：集群弹性公网IP:http端口号（32286）即可访问该服务页面。

----结束

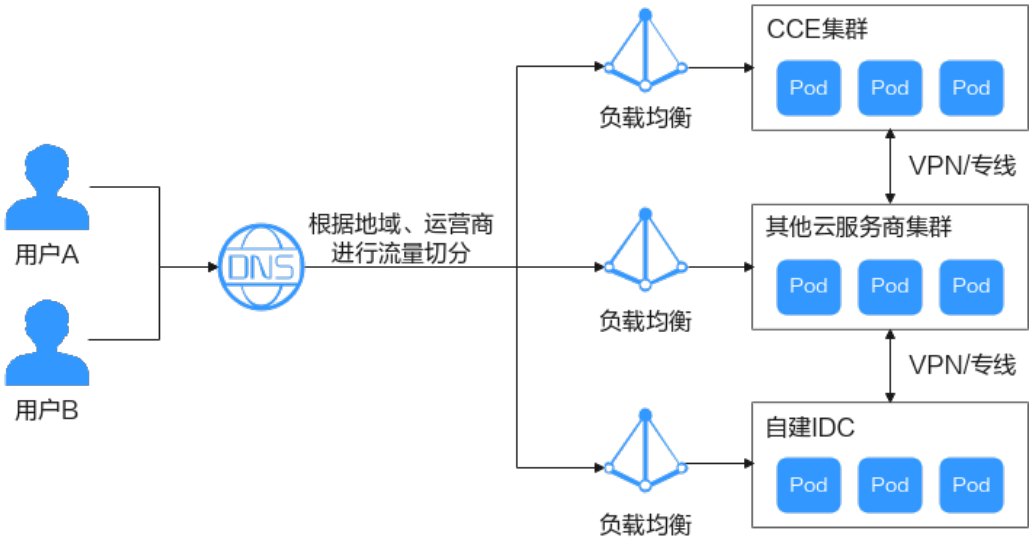
8 流量分发

8.1 流量分发概述

UCS提供的流量分发功能可基于访问位置和业务策略对全域流量进行最优化调度，支持跨云多集群服务接入和流量管理，实现智能流量分发调度，实时跨域、按需调配应用访问流量。

通过云解析服务DNS，用户的访问流量可根据运营商、地域等维度进行切分，对同一域名的访问请求进行不同的解析响应，指向不同的后端集群，从而解决了跨域、跨网访问延迟高的问题。

图 8-1 流量管理示意图



前提条件

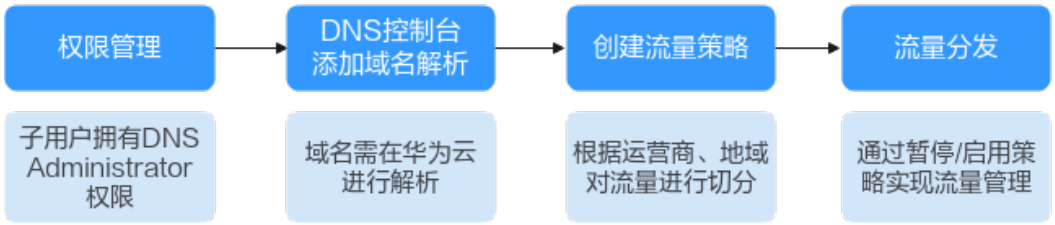
- IAM子用户如需进行流量管理，必须拥有DNS Administrator权限。
- 您需要已有一个公网域名，如果您尚未购买公网域名，请**购买**。
- 您的公网域名已经过备案，如果您的公网域名尚未备案，请前往**华为云备案中心**备案。

- 您的公网域名可以正常被解析，如果您尚未创建公网域名解析，请**创建**。

使用流程

流量分发功能的使用流程如**图8-2**所示。

图 8-2 流量分发功能使用流程



8.2 创建流量策略

步骤1 登录UCS控制台，在左侧导航栏中单击“流量分发”。

步骤2 在流量管理控制台页面，单击右上角“创建流量策略”。

步骤3 在弹出页面中，填写域名，并请至少添加一条调度策略。如您需要为多个域名创建流量策略，请依次重复**步骤3**~**步骤5**。

- 域名：域名前缀可自定义，后缀为已购买备案并且添加华为云DNS解析的公网域名。

  前缀部分由多个以点分隔的字符串组成，可包含字母、数字、汉字、中划线，中划线不能在开头或末尾，单个字符串不超过63个字符，域名总长度不超过254个字符。

  说明

  - 如果没有子域名，域名前缀部分可不填。
  - 域名后缀部分为云解析服务（DNS）中已添加解析的公网域名。如您需要管理域名，请前往DNS控制台，具体步骤请参见**公网域名管理**。
- 调度策略：可基于访问位置和业务策略对流量进行调度，详见**步骤4**。

步骤4 单击 + 按钮，添加一个调度策略后单击“确定”，如**图8-3**所示。如需为同一域名添加不同的调度策略，请重复此步骤，也可在创建完成后继续添加。

图 8-3 添加调度策略

- 集群：选择一个状态为“运行中”的目标集群，列表中将自动获取UCS接管的所有集群。
- 命名空间：选择目标服务所在的命名空间，默认为“default”。
- 服务：选择一个目标服务，仅支持访问类型为负载均衡的服务，查询结果已过滤。
- 线路类型：
  - 全网默认：必选，未匹配到已设置的线路时，会返回默认解析结果。
  - 运营商线路解析：根据访问用户所在运营商网络调度到最佳访问地址。默认值为“电信/地区默认”，支持指定运营商及地区，其中地区选择的细粒度为省级。
  - 地域解析：根据访问用户所处地理位置调度到最佳访问地址。默认值为“中国大陆/地区默认”，支持全球地域选择，其中中国大陆地区细粒度为省级，其余地域细粒度为国家/地区。

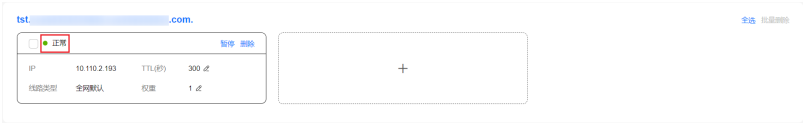
  须知

  您需要先创建一条“全网默认”线路类型的调度策略作为默认解析，再添加自定义调度策略。如果域名未添加默认线路解析记录，会造成指定线路外的地区访问失败。

- TTL：指解析记录在本地DNS服务器的缓存时间，默认值为“300s/5分钟”。如果您的服务地址经常更换，建议TTL值设置相对小些，反之，建议设置相对大些。
- 权重：当域名在同一解析线路中有多条相同类型的解析记录时，可以通过“权重”设置解析记录集的响应比例。详细信息请参考**配置权重解析**。

步骤5 单击“创建”，流量策略创建成功。

图 8-4 成功创建流量策略



----结束

8.3 管理流量策略

暂停调度策略

对于已创建的流量策略，如发生集群故障倒换等突发场景时，支持将某个调度策略暂停使用，待故障解除后再启用调度策略。下面以暂停调度策略为例进行说明，启用调度策略的方法与暂停一致，不再赘述。

步骤1 登录UCS控制台，在左侧导航栏中单击“流量分发”。

步骤2 在对应的调度策略框右上角位置单击“暂停”。

步骤3 在弹窗中二次确认暂停调度策略后，此调度策略将会被暂停使用。

----结束

删除调度策略

步骤1 登录UCS控制台，在左侧导航栏中单击“流量分发”。

步骤2 在对应的调度策略框右上角位置单击“删除”。

如您需要删除多个调度策略，可在对应的调度策略框左上角勾选此策略，然后单击界面右上角的“批量删除”。

步骤3 在弹窗中二次确认删除调度策略，删除操作将无法恢复，请您谨慎操作。

说明

删除中时请勿关闭当前弹窗或刷新页面，删除完成后弹框会自动关闭，否则可能导致部分资源残留。

----结束

9 可观测性

9.1 容器智能分析

9.1.1 容器智能分析概述

容器智能分析是华为云打造的新一代云原生容器运维平台，可实时监控应用及资源，采集各项指标及事件等数据以分析应用健康状态，提供全面、清晰、多维度数据可视化能力，兼容主流开源组件，并提供快捷故障定位的能力。

产品功能

- **容器洞察**：提供基于Kubernetes原生类型的容器监控能力，支持集群、节点、工作负载的资源全景，支持节点的资源占用、工作负载的资源消耗，以及近一小时的CPU/内存指标展示，全面监控集群的健康状态和负荷程度。
- **健康诊断**：对集群健康状态进行周期性检查，可以对集群、节点资源使用情况，工作负载、Pod资源状态进行快速诊断。
- **仪表盘**：仪表盘可将不同图表展示到同一个屏幕上，通过不同的仪表形式来展示资源数据，例如，曲线图、数字图等，进而全面、深入地掌握监控数据。

产品优势

- 容器智能分析深度整合云原生基金会（CNCF）的监控项目Prometheus，同时遵循OpenTracing/OpenTelemetry规范。对关键指标、事件等运维数据进行统一采集、存储和可视化展现，精心打造云原生应用的良好可观测性能力。
- 将云原生基础设施监控和应用负载监控进行关联，提供全栈监控，使用户能够随时随地清晰地感知基础设施和应用负载状态。
- 能够对Kubernetes集群、容器组（Pod）等进行详细监控，对业务提供端到端追踪和可视化，提供集群健康诊断能力，大大缩短问题分析定位时间。
- 提供开箱即用的插件安装、数据采集、仪表盘监控能力，相比基于开源产品打造的监控产品，在可靠性、高可用、安装部署便捷性上更具有竞争力，能够更好的为您的云原生应用保驾护航。

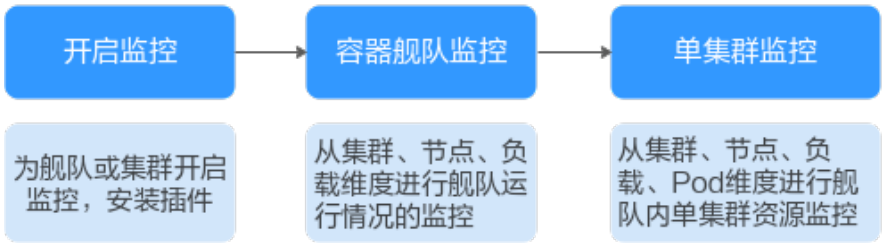
约束与限制

- 仅**华为云账号**，或者具备**UCS FullAccess**或**UCS CIAOperations**（建议）权限的用户可进行容器智能分析相关的操作。
- 本地集群和附着集群目前只支持将指标和事件上报到北京四Region的AOM 2.0和LTS服务；华为云集群目前只支持北京四、上海一Region启用容器智能分析服务。

使用流程

容器智能分析的使用流程如图 容器智能分析使用流程所示。

图 9-1 容器智能分析使用流程



9.1.2 为集群开启监控

9.1.2.1 集群监控概述

为集群开启监控才能确保您的集群处于实时守护状态。

关于插件的详细说明，请参考**kube-prometheus-stack插件**和**云原生日志采集插件**章节。

目前容器智能分析支持监控华为云集群、附着集群、本地集群、多云集群。在开启监控时，各个集群的参数配置存在差异，因此，本小节将分别介绍这五种集群的开启操作。

- **为华为云集群开启监控**
- **为本地集群开启监控**
- **为附着集群开启监控**
- **为多云集群开启监控**

插件状态说明

kube-prometheus-stack和log-agent插件的状态说明如**表9-1**所示。部分状态将影响集群进行监控开启、监控配置修改和监控关闭操作，详见后续章节的约束与限制部分。

表 9-1 插件状态说明

状态	说明
插件未安装	插件未安装
运行中	插件全部实例状态都在运行中，插件正常使用

状态	说明
安装中	插件正在安装中
升级中	插件正在更新中
回滚中	插件正在回滚中
回滚失败	插件回滚失败，可重试回滚或卸载后重新安装
删除中	插件正在删除中
部分就绪	插件下只有部分实例状态为运行中，插件部分功能可用
不可用	插件状态异常，插件不可使用。可单击插件名称查看实例异常事件
安装失败	插件安装失败，需要卸载后重新安装
升级失败	插件升级失败，可重试升级或卸载后重新安装
删除失败	插件删除失败，可重试卸载
未知	插件处于未知状态，需要卸载后重新安装

9.1.2.2 为华为云集群开启监控

本章节讲述为华为云集群开启监控的操作流程。

约束与限制

华为云集群开启监控之前，有可能已经安装了kube-prometheus-stack插件，若该插件处于“安装中”、“升级中”、“删除中”和“回滚中”状态时，不允许开启监控。插件的状态说明请参见**插件状态说明**。

前提条件

已将华为云集群注册到UCS中，具体操作请参见**华为云集群**。

操作步骤

步骤1 登录UCS控制台，在左侧导航栏中单击“容器智能分析”。

步骤2 选择一个容器舰队或者未加入舰队的集群，并单击右上角“开启监控”按钮。

图 9-2 选择舰队或未加入舰队的集群

步骤3 选择一个华为云集群。

步骤4 单击“下一步：接入配置”，完成指标采集配置。

规格配置

- 部署模式：支持Agent模式和Server模式。Agent模式占用集群资源较低，为集群提供普罗指标采集能力，但不支持基于自定义普罗语句的HPA及健康诊断功能。Server模式为集群提供普罗指标采集能力，支持基于自定义普罗语句的HPA及健康诊断功能，依赖PVC，内存消耗较大。
- 插件规格：如果部署模式选择“Agent模式”，插件规格为默认值。如果部署模式选择“Server模式”，插件规格包括演示规格（100容器以内）、小规格（2000容器以内）、中规格（5000容器以内）和大规格（超过5000容器）四种规格。不同规格对集群的CPU、内存等资源要求不同。不同插件规格占用的资源配额可参考**不同规格的资源配额要求**。

参数配置

- 对接方式：当前仅支持接入AOM服务。
- AOM实例：容器监控会将指标统一上报给AOM服务，因此需要选择一个Prometheus for CCE 类型的AOM实例。默认指标是免费的，而自定义指标将由AOM服务收费。
- 采集周期：普罗采集指标数据并上报的时间周期。取值范围10~120秒，默认为15秒。

  存储：（部署模式选择“Server模式”时需要配置）用于普罗数据的临时存储（PVC），华为云集群默认使用云硬盘（csi-disk-topology）存储类型的PVC。如果命名空间monitoring下已存在可使用的PVC（pvc-prometheus-server），则可以使用该存储作为存储源。
  - 云硬盘类型：可选择高IO、超高IO、普通IO。
  - 容量：为创建PVC时指定的容量大小或者选择Pod存储时的存储最大限制值。

  须知

  插件存储使用云硬盘会产生额外费用，请参考**价格详情**。

关于插件的详细说明请参见**kube-prometheus-stack插件**。

步骤5 单击“确认接入”，自动返回至“容器洞察 > 集群总览”页面，集群的接入状态为“安装中”。

等待集群开启成功后，列表中将显示集群的CPU使用率、CPU分配率等指标，说明集群已经处于容器智能分析的守护中了。

说明

若集群开启失败，请参考**常见问题**处理。

----结束

9.1.2.3 为本地集群开启监控

本章节讲述为本地集群开启监控的操作流程。

前提条件

已将本地集群注册到UCS中，具体操作请参见**本地集群概述**。

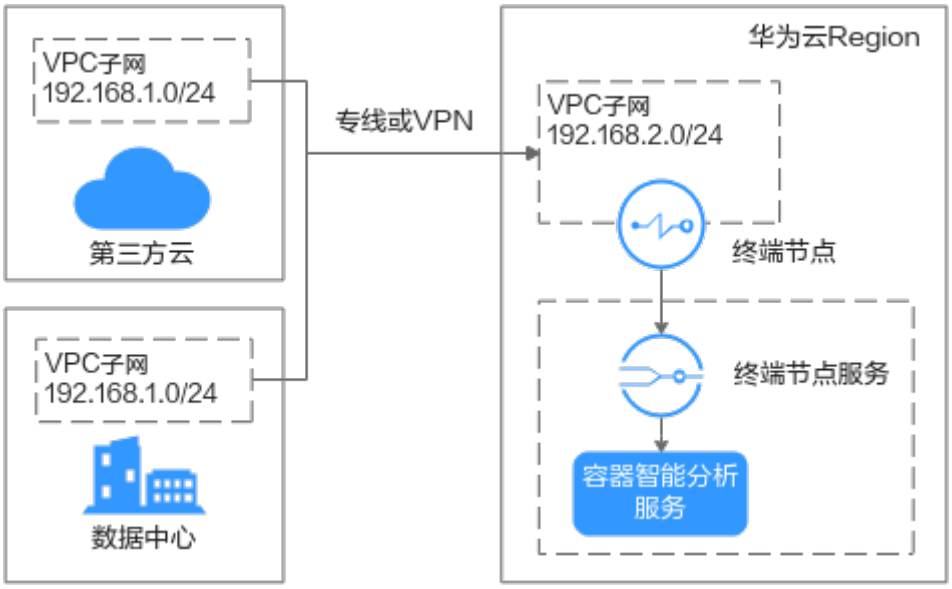
准备网络环境

本地集群的数据接入方式支持公网接入和私网接入。

- 公网接入是通过公网Internet接入，要求集群能够访问公网，具有弹性灵活、成本低、易接入的优势。如果对网络质量没有要求，只想采用更简便的方式接入，那么公网接入是个不错的选择。

  公网接入要求集群能够访问公网，请确保集群已符合此条件，否则会接入失败。
- 私网接入是通过云专线（DC）或虚拟专用网络（VPN）服务将云下网络与云上虚拟私有云（VPC）连通，并利用VPC终端节点通过内网与容器智能分析建立连接，具有高速、低时延、安全的优势。

图 9-3 私网接入原理



因此，在开启之前，您需要准备满足一个云上虚拟私有云（VPC），并将线下自有IDC的网络环境与该VPC连通。VPC子网网段不能与IDC中已使用的网络网段重叠，否则将无法接入集群，例如，IDC中已使用的VPC子网为192.168.1.0/24，那么华为云VPC中不能使用192.168.1.0/24这个子网。

网络连通可以选用如下两种方案：

- 虚拟专用网络（VPN）方案：请参见**通过VPN连接云下数据中心与云上VPC**。
- 云专线（DC）方案：请参见**用户通过单专线静态路由访问VPC**或**用户通过单专线BGP协议访问VPC**。

为集群开启监控

步骤1 登录UCS控制台，在左侧导航栏中单击“容器智能分析”。

步骤2 选择一个容器舰队或者未加入舰队的集群，并单击右上角“开启监控”按钮。

图 9-4 选择舰队或未加入舰队的集群

步骤3 选择一个本地集群。

步骤4 单击“下一步：接入配置”，完成网络配置。

- 数据接入方式：支持选择“公网接入”和“私网接入”。
- 数据上报区域：选择数据上报的区域，和已连通云下网络的VPC属于同一区域。
- 项目列表：如果开通了IAM项目，还需要选择一个项目。
- 私网接入点：数据接入方式为“私网接入”时需要配置。

  在已连通云下网络的VPC中创建VPC终端节点（VPCEP）连通容器智能分析数据上报接收点，可选择已有私网接入点，当选择新建私网接入点时将会收取0.1元/小时的VPCEP资源费用。

  创建私网接入点将会创建一个VPCEP终端节点和一个DNS内网域名，需保证主账号有相应资源的配额。另外，还需要确保页面选择的子网存在可用IP。

步骤5 完成指标采集配置。

规格配置

- 部署模式：支持Agent模式和Server模式。Agent模式占用集群资源较低，为集群提供普罗指标采集能力，但不支持基于自定义普罗语句的HPA及健康诊断功能。Server模式为集群提供普罗指标采集能力，支持基于自定义普罗语句的HPA及健康诊断功能，依赖PVC，内存消耗较大。
- 插件规格：如果部署模式选择“Agent模式”，插件规格为默认值。如果部署模式选择“Server模式”，插件规格包括演示规格（100容器以内）、小规格（2000容器以内）、中规格（5000容器以内）和大规格（超过5000容器）四种规格。不同规格对集群的CPU、内存等资源要求不同。不同插件规格占用的资源配额可参考**不同规格的资源配额要求**。

参数配置

- 对接方式：当前仅支持接入AOM服务。
- AOM实例：容器监控会将指标统一上报给AOM服务，因此需要选择一个Prometheus for CCE 类型的AOM实例。默认指标是免费的，而自定义指标将由AOM服务收费，收费标准请参见**AOM计费说明**。
- 采集周期：普罗采集指标数据并上报的时间周期。取值范围10~60秒，默认为15秒。
- 存储：（部署模式选择“Server模式”时需要配置）用于普罗数据的临时存储。本地集群支持CSI-Local存储类型。将节点所拥有的磁盘以PVC的方式提供给容器使用，使用此种存储卷时容器将固定调度至本地存储卷所在节点运行，需确保Pod与目标节点没有调度冲突。

- 存储类型：选择“CSI-Local”。
- 容量：创建PVC时指定的容量大小，此容量仅供参考，实际容量为本地目录所在盘的可用容量。
- 节点：指定普罗服务准备调度的节点，需确保普罗服务可调度到此节点。
- 节点目录：指定普罗服务存储数据的目录，请输入绝对路径，该路径将在目标节点自动创建。

关于插件的详细说明请参见**kube-prometheus-stack插件**。

步骤6 单击“确认接入”，自动返回至“容器洞察 > 集群总览”页面，集群的接入状态为“安装中”。

等待集群开启监控成功后，列表中将显示集群的CPU使用率、CPU分配率等指标，说明集群已经处于容器智能分析的守护中了。

说明

若集群开启监控失败，请参考**常见问题**处理。

----结束

9.1.2.4 为附着集群开启监控

本章节讲述为附着集群开启监控的操作流程。

前提条件

已将附着集群注册到UCS中，具体操作请参见**附着集群概述**。

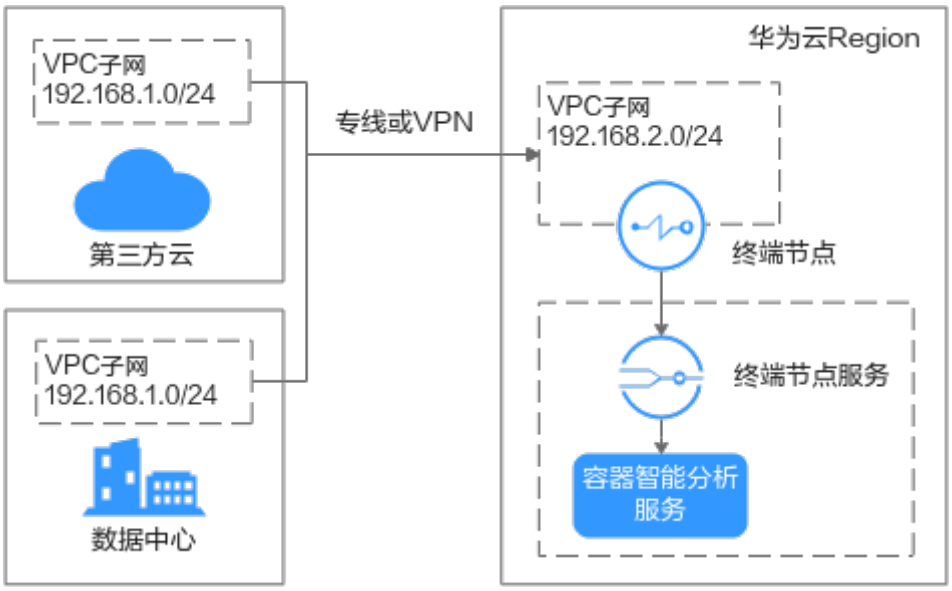
准备网络环境

附着集群的数据接入方式支持公网接入和私网接入。

- 公网接入是通过公网Internet接入，要求集群能够访问公网，具有弹性灵活、成本低、易接入的优势。如果对网络质量没有要求，只想采用更简便的方式接入，那么公网接入是个不错的选择。

  公网接入要求集群能够访问公网，请确保集群已符合此条件，否则会接入失败。
- 私网接入是通过云专线（DC）或虚拟专用网络（VPN）服务将第三方云网络与云上虚拟私有云（VPC）连通，并利用VPC终端节点通过内网与容器智能分析建立连接，具有高速、低时延、安全的优势。

图 9-5 私网接入原理



因此，在开启之前，您需要准备满足一个云上虚拟私有云（VPC），并将第三方云厂商的网络环境与该VPC连通。VPC子网网段不能与第三方云中已使用的网络网段重叠，否则将无法接入集群，例如，第三方云中已使用的VPC子网为192.168.1.0/24，那么华为云VPC中不能使用192.168.1.0/24这个子网。

网络连通可以选用如下两种方案：

- 虚拟专用网络（VPN）方案：请参见**通过VPN连接云下数据中心与云上VPC**。
- 云专线（DC）方案：请参见**用户通过单专线静态路由访问VPC**或**用户通过单专线BGP协议访问VPC**。

为集群开启监控

步骤1 登录UCS控制台，在左侧导航栏中单击“容器智能分析”。

步骤2 选择一个容器舰队或者未加入舰队的集群，并单击右上角“开启监控”按钮。

图 9-6 选择舰队或未加入舰队的集群

步骤3 选择一个附着集群。

步骤4 单击“下一步：接入配置”，完成网络配置。

- 数据接入方式：支持选择“公网接入”和“私网接入”。

- 数据上报区域：选择数据上报的区域，和已连通第三方云网络的VPC属于同一区域。
- 项目列表：如果开通了IAM项目，还需要选择一个项目。
- 私网接入点：数据接入方式为“私网接入”时需要配置。

  在已连通第三方云网络的VPC中创建VPC终端节点（VPCEP）连通容器智能分析数据上报接收点，可选择已有私网接入点，当选择新建私网接入点时将会收取0.1元/小时的VPCEP资源费用。

  创建私网接入点将会创建一个VPCEP终端节点和一个DNS内网域名，需保证主账号有相应资源的配额。另外，还需要确保页面选择的子网存在可用IP。

步骤5 完成指标采集配置。

规格配置

- 部署模式：支持Agent模式和Server模式。Agent模式占用集群资源较低，为集群提供普罗指标采集能力，但不支持基于自定义普罗语句的HPA及健康诊断功能。Server模式为集群提供普罗指标采集能力，支持基于自定义普罗语句的HPA及健康诊断功能，依赖PVC，内存消耗较大。
- 插件规格：如果部署模式选择“Agent模式”，插件规格为默认值。如果部署模式选择“Server模式”，插件规格包括演示规格（100容器以内）、小规格（2000容器以内）、中规格（5000容器以内）和大规格（超过5000容器）四种规格。不同规格对集群的CPU、内存等资源要求不同。不同插件规格占用的资源配额可参考**不同规格的资源配额要求**。

参数配置

- 对接方式：当前仅支持接入AOM服务。
- AOM实例：容器监控会将指标统一上报给AOM服务，因此需要选择一个Prometheus for CCE 类型的AOM实例。默认指标是免费的，而自定义指标将由AOM服务收费，收费标准请参见**AOM计费说明**。
- 采集周期：普罗采集指标数据并上报的时间周期。取值范围10~60秒，默认为15秒。
- 存储：（部署模式选择“Server模式”时需要配置）用于普罗数据的临时存储。
  - 存储类型：附着集群支持Emptydir和Local Storage两种存储类型。

    使用Emptydir模式普罗数据将存储在Pod中，请确保prometheus-server-0调度到的节点上的容器存储挂载容量满足所输入的容量大小。

    使用Local Storage将会在您的集群内创建monitoring命名空间（如果不存在），以及local-storage类型的PV及PVC，请保证您指定的节点上存在所输入的目录以及该目录满足所输入的容量大小。
  - 容量：创建PVC时指定的容量大小或者选择Pod存储时的存储最大限制值。

关于插件的详细说明请参见**kube-prometheus-stack插件**。

步骤6 单击“确认接入”，自动返回至“容器洞察 > 集群总览”页面，集群的接入状态为“安装中”。

等待集群开启监控成功后，列表中将显示集群的CPU使用率、CPU分配率等指标，说明集群已经处于容器智能分析的守护中了。

说明

若集群开启监控失败，请参考**常见问题**处理。

----结束

9.1.2.5 为多云集群开启监控

本章节讲述为多云集群开启监控的操作流程。

前提条件

已将多云集群注册到UCS中，具体操作请参见**多云集群概述**。

准备网络环境

多云集群的数据接入方式支持公网接入，要求集群能够访问公网，具有弹性灵活、成本低、易接入的优势。如果对网络质量没有要求，只想采用更简便的方式接入，那么公网接入是个不错的选择。

公网接入要求集群能够访问公网，请确保集群已符合此条件，否则会接入失败。

为集群开启监控

步骤1 登录UCS控制台，在左侧导航栏中单击“容器智能分析”。

步骤2 选择一个容器舰队或者未加入舰队的集群，并单击右上角“开启监控”按钮。

图 9-7 选择舰队或未加入舰队的集群

步骤3 选择一个多云集群。

步骤4 单击“下一步：接入配置”，完成网络配置。

- 数据接入方式：“公网接入”。
- 数据上报区域：选择数据上报的区域。
- 项目列表：如果开通了IAM项目，还需要选择一个项目。

步骤5 完成指标采集配置。

规格配置

- 部署模式：支持Agent模式和Server模式。Agent模式占用集群资源较低，为集群提供普罗指标采集能力，但不支持基于自定义普罗语句的HPA及健康诊断功能。Server模式为集群提供普罗指标采集能力，支持基于自定义普罗语句的HPA及健康诊断功能，依赖PVC，内存消耗较大。

- 插件规格：如果部署模式选择“Agent模式”，插件规格为默认值。如果部署模式选择“Server模式”，插件规格包括演示规格（100容器以内）、小规格（2000容器以内）、中规格（5000容器以内）和大规格（超过5000容器）四种规格。不同规格对集群的CPU、内存等资源要求不同。不同插件规格占用的资源配额可参考**不同规格的资源配额要求**。

参数配置

- 对接方式：当前仅支持接入AOM服务。
- AOM实例：容器监控会将指标统一上报给AOM服务，因此需要选择一个Prometheus for CCE 类型的AOM实例。默认指标是免费的，而自定义指标将由AOM服务收费，收费标准请参见**AOM计费说明**。
- 采集周期：普罗采集指标数据并上报的时间周期。取值范围10~60秒，默认为15秒。
- 存储：（部署模式选择“Server模式”时需要配置）用于普罗数据的临时存储。
  - 存储类型：多云集群支持Emptydir和Local Storage两种存储类型。

    使用Emptydir模式普罗数据将存储在Pod中，请确保prometheus-server-0调度到的节点上的容器存储挂载容量满足所输入的容量大小。

    使用Local Storage将会在您的集群内创建monitoring命名空间（如果不存在），以及local-storage类型的PV及PVC，请保证您指定的节点上存在所输入的目录以及该目录满足所输入的容量大小。
  - 容量：创建PVC时指定的容量大小或者选择Pod存储时的存储最大限制值。

关于插件的详细说明请参见**kube-prometheus-stack插件**。

步骤6 单击“确认接入”，自动返回至“容器洞察 > 集群总览”页面，集群的接入状态为“安装中”。

等待集群开启监控成功后，列表中将显示集群的CPU使用率、CPU分配率等指标，说明集群已经处于容器智能分析的守护中了。

说明

若集群开启监控失败，请参考**常见问题**处理。

----结束

9.1.2.6 修改监控配置

集群开启监控成功后，还可以修改监控配置，网络配置、指标采集配置和事件采集配置均支持修改。

说明

当事件采集配置从开启置为关闭，系统将会删除log-agent插件。

约束与限制

kube-prometheus-stack插件处于“安装中”、“升级中”、“删除中”、“回滚中”、“回滚失败”、“不可用”、“安装失败”、“删除失败”和“未知”状态时，不允许修改集群监控配置。

操作步骤

步骤1 登录UCS控制台，在左侧导航栏中选择“容器智能分析”。

步骤2 选择一个容器舰队或者未加入舰队的集群。

图 9-8 选择舰队或未加入舰队的集群

步骤3 单击“容器洞察 > 集群总览”页签查看已开启监控的集群，在需要修改配置的集群行单击“修改接入配置”。

步骤4 修改完成后单击“确认接入”。

----结束

9.1.2.7 关闭监控

本章节讲述为集群关闭监控的操作流程。

约束与限制

关闭监控前，请仔细阅读以下注意事项，避免数据丢失或者相关资源继续产生费用。

- kube-prometheus-stack插件处于“安装中”、“升级中”、“删除中”和“回滚中”时不允许关闭监控。
- kube-prometheus-stack插件处于“运行中”、“部分就绪”和“安装失败”时关闭监控：对于华为云集群，系统将会更新kube-prometheus-stack插件以关闭数据上报功能；对于本地集群和附着集群，系统将会卸载kube-prometheus-stack插件。
- kube-prometheus-stack插件处于“回滚失败”、“不可用”、“安装失败”、“删除失败”和“未知”状态时，关闭监控将会卸载kube-prometheus-stack插件。
- 通过私网接入的本地集群和附着集群，关闭监控时会检查私网接入点（开启监控时创建的VPCEP终端节点和DNS内网域名）是否有其他集群在使用，若没有则会删除此私网接入点。
- 华为云集群使用云硬盘（csi-disk-topology）存储类型的PVC作为插件数据的临时存储，集群关闭监控后，命名空间monitoring下的PVC不会自动删除，为避免继续产生费用，请前往CCE控制台手动删除（需要先卸载kube-prometheus-stack插件，再删除）。

操作步骤

步骤1 选择一个容器舰队或者未加入舰队的集群。

图 9-9 选择舰队或未加入舰队的集群

步骤2 单击“容器洞察 > 集群总览”页签查看已开启监控的集群，在需要关闭监控的集群行，单击“取消监控”。

步骤3 在二次确认弹窗中单击“确认”，即可为集群关闭监控。

----结束

9.1.3 容器洞察

9.1.3.1 容器洞察概述

容器洞察提供基于Kubernetes原生类型的容器监控能力，支持集群、节点、工作负载的资源全景，支持节点的资源占用、工作负载的资源消耗，以及近一小时的CPU/内存指标展示，全面监控集群的健康状态和负荷程度。

9.1.3.2 查看舰队总览

查看舰队总览。您可以选择一个容器舰队或未加入舰队的集群，查看所选范围内已开启监控的集群、以及集群中的节点、负载总览信息。

说明

本小节操作指导均以查看容器舰队的总览信息为例，若您需要查看未加入舰队集群的总览信息，请在容器洞察页面选择“其他 > 未加入舰队集群”，查看全部未加入舰队的集群、以及集群中的节点、负载总览信息。

查看舰队内集群总览

功能入口

步骤1 登录UCS控制台，在左侧导航栏中选择“容器智能分析”，选择一个容器舰队。

步骤2 单击“容器洞察 > 集群总览”页签查看已开启监控的集群，列表中显示容器舰队内所有集群的CPU使用率、CPU分配率、内存分配率和使用率等指标。

----结束

页面简介

集群总览页面，可以展示同一容器舰队下所有集群详细信息，包括集群状态、类型、所在区域、CPU/内存使用率、CPU/内存分配率、节点正常/总数，可以进行集群开启/关闭操作，**修改集群监控配置**。

词条	词条简介
集群概况	集群概况展示了舰队内集群的名称、风险等级、当前运行状态、类型、所在区域、CPU/内存的使用率与分配率等信息，且单击上方搜索框，选择属性类型，再输入对应的关键字，即可查询该条件下的集群。
集群风险概览	集群风险概览是容器智能分析针对集群健康状态进行7×24小时持续检查的功能。通过该功能，您可以对集群的风险情况、K8sWarning事件进行快速诊断，并且可以为非正常项提供恢复建议。 **说明** 集群风险概览下仅展示最近100条K8S Warning事件，更多事件请前往**集群事件**页查看。 K8S事件列表内不包含附着集群。
用量统计	此处默认统计近1小时、近8小时和近24小时的CPU/内存的平均水位。帮助您快速识别系统资源占用情况。 **说明** 您可以将鼠标悬停在图表上，以便查看每分钟的监控数据。
资源统计	资源统计涵盖了该舰队下CPU用量Top5集群、内存用量Top5集群、节点数量Top5集群以及Pod数量Top5集群等参数统计。其中单击内存/CPU右上角“可分配量”可查看其剩余可分配用量，单击节点/Pod数量右上角“异常数”可查看其出现异常的集群数量。
资源盘点	资源盘点可统计该舰队下所有集群的版本、集群运营厂商、集群类型以及集群所在区域等不同类型集群在舰队内所占比例，且单击集群版本号、运营商名称或本地集群，即可查看除该类型外，其他类型集群在该舰队下的占比情况。

查看舰队内节点总览

功能入口

步骤1 登录UCS控制台，在左侧导航栏中选择“容器智能分析”，选择一个容器舰队。

步骤2 单击“容器洞察 > 节点总览”页签查看节点总览。

----结束

页面简介

节点总览页面包含了同一舰队内已开启监控的所有集群下的节点信息，以及节点风险统计和资源占用情况。

表 9-2 节点总览页面

词条	词条简介
节点概览	节点概况展示了节点的名称、当前运行状态、CPU/内存的使用率、所属集群、节点IP地址以及节点所在地区，且单击上方搜索框，选择属性类型，再输入对应的关键字，即可查询该条件下的节点。

词条	词条简介
节点风险概览	节点风险概览统计该舰队内已开启监控的集群节点发生的K8s Warning事件，包括事件名称、类型、所属集群名称、资源类型、资源名称、事件内容、发生时间和发生次数等。单击上方搜索框，选择属性类型，再输入对应的关键字，即可查询该条件下的事件。单击，可对事件进行排序。 **说明** 仅展示最近100条K8S Warning事件，更多事件请前往**集群事件**页查看。
资源统计	资源统计涵盖了该舰队下CPU用量 Top5节点和内存用量 Top5节点。分别单击右上角内存/CPU“可分配量”可查看其剩余可分配用量。单击CPU用量Top5节点图表旁任一节点，即可隐藏其在图表中的数据，只查看其它节点数据。

查看舰队内工作负载总览

功能入口

负载总览界面提供了同一舰队内所有开启监控的集群下所有工作负载的综合信息、包括负载列表、风险概览和资源统计等。

步骤1 登录UCS控制台，在左侧导航栏中选择“容器智能分析”，选择一个容器舰队。

步骤2 单击“容器洞察 > 负载总览”页签查看同一舰队内已开启监控的集群下的所有工作负载，列表中显示工作负载的名称、状态、实例个数、CPU使用率、内存使用率等指标。列表右上角还支持按照工作负载类型筛选希望展示的工作负载。

----结束

词条	词条简介
负载概况	负载概况可展示同一舰队内所有开启监控的集群下所有工作负载列表，包含负载当前状态、实例个数（正常/全部）、所属命名空间、所属集群、CPU/内存使用率等。单击上方搜索框，选择属性类型，再输入对应的关键字，即可查询该条件下的负载。单击，可对列表内的负载进行排序。
工作负载风险概览	工作负载风险概览统计该舰队内已开启监控的集群节点发生的K8s Warning事件，包括事件名称、类型、所属集群名称、资源类型、资源名称、事件内容、发生时间和发生次数等。单击上方搜索框，选择属性类型，再输入对应的关键字，即可查询该条件下的事件。 单击，可对事件进行排序。 单击右上角条件框、可对列表内的工作负载进行筛选。 **说明** 仅展示最近100条K8S Warning事件，更多事件请前往**集群事件**页查看。

词条	词条简介
资源统计	资源统计涵盖了该舰队下CPU用量Top5工作负载、内存用量Top5工作负载、重启次数Top5工作负载和异常Pod数Top5工作负载。分别单击右上角“平均使用率”内存/CPU可查看其剩余可分配量。单击CPU用量Top5工作负载图表旁任一工作负载，即可隐藏其在图表中的数据，只查看其它工作负载数据。

9.1.3.3 查看集群情况

查看集群情况流程

在“容器洞察 > 集群总览”页面的集群统计列表中，单击集群名称跳转至单个集群的智能分析页面。本页面分为五个页签，分别为：

- “集群”页签：具体信息请参见**查看集群详情**。
- “节点”页签：具体信息请参见查看**集群内节点详情**。
- “工作负载”页签：具体信息请参见**查看集群内工作负载详情**。
- “Pod”页签：具体信息请参见**查看集群内Pod详情**。
- “事件”页签：具体信息请参见**查看集群内事件详情**。

查看集群详情

集群详情页面提供了单个集群的监控情况，包含资源概况、资源消耗TOP统计和用量统计多维度的信息概况。通过集群监控您可以及时了解集群的资源使用情况和趋势，快速响应可能存在的风险项，保证集群流畅运行。

您可以将鼠标悬停在图表上，以便查看每分钟的监控数据。

图 9-10 集群详情页面

表 9-3 集群详情页面

词条	词条简介
集群健康度	资源健康度评估包括多个维度，如健康评分、待处理风险项数、风险等级，以及诊断风险项在Master、集群、节点、工作负载和外部依赖五个方面的占比（异常数据使用红色突出显示）。欲了解更多诊断结果，请前往**健康诊断**页面查看。 **须知** 当集群所安装的kube-prometheus-stack插件的部署模式为“Server模式”时，方可以查看集群的资源健康度。
资源健康概况	资源概况涵盖了节点、工作负载和容器组三类资源中异常资源所占比例，以及命名空间的总数。此外，还包括了控制面组件和Master节点的异常占比、API Server总QPS以及API Server请求错误率。 作为集群的API服务提供者，控制面API Server的异常可能会导致整个集群无法访问，同时也会影响依赖API Server的工作负载的正常运行。为了帮助您快速识别和修复问题，资源概况中提供了API Server的总QPS和请求错误率指标。
资源消耗Top统计	在资源消耗TOP统计中，UCS服务会将CPU使用率和内存使用率排名前五的节点、无状态负载、有状态负载和Pod纳入统计范围，以帮助您识别资源消耗“大户”。 **说明** • CPU使用率 工作负载CPU使用率 = 工作负载各个Pod中CPU使用率的平均值 Pod CPU使用率 = Pod实际使用的CPU核数 / 业务容器CPU核数限制值之和（未配置限制值时采用节点总量） • 内存使用率 工作负载内存使用率 = 工作负载各个Pod中内存使用率的平均值 Pod内存使用率 = Pod实际使用的物理内存 / 业务容器物理内存限制值之和（未配置限制值时采用节点总量）
数据面监控	此处默认统计近1小时、近8小时和近24小时的各维度资源用量。如需查看更多监控信息，请单击“查看全部监控”，跳转至“仪表盘”页面，相应指导请参见**仪表盘**。

9.1.3.4 查看集群内节点情况

如果您需要监控节点的资源使用情况，可以前往容器洞察中的节点页面查看。该页面提供了指定集群下所有节点的综合信息，以及单个节点的详细监控数据，包括CPU/内存使用率、网络流入/流出速率、磁盘读/写IO等。

功能入口

步骤1 登录UCS控制台。

步骤2 在左侧导航栏中选择“容器智能分析”，在“容器洞察 > 集群总览”页面的集群统计列表中，单击集群名称，选择“节点”。

页面呈现了所有工作负载的综合信息，如需深入了解单个工作负载的监控情况，可单击工作负载名称，进入该工作负载的“概览”页面，通过切换“实例列表”、“监控”页签查看相应内容。

----结束

查看集群内节点列表

节点列表中包含节点名称、状态、IP地址、Pod（已分配/总额度）、CPU申请比率/限制比率/使用率，以及内存申请比率/限制比率/使用率等信息。

您可以通过在列表上方按照节点名称、状态、私有地址和公网地址进行筛选，快速找到需要的节点。在列表的右上角，您可以单击  按钮来导出全部节点数据，或者选择部分节点进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

节点概况展示了节点的名称、当前运行状态、CPU/内存的使用率、所属集群、节点IP地址以及节点所在地区，且单击上方搜索框，选择属性类型，再输入对应的关键字，即可查询该条件下的节点。

当节点的CPU限制比率或内存限制比率超过100%时，意味着节点资源超分，节点上的负载限制值（可使用的最大值）之和已经超过了节点规格。如果负载占用资源过高，可能会导致节点异常。

查看集群内节点详情

在节点列表中，单击需要查看详情的节点名称，进入该节点的详情页面，通过切换“概览”、“Pod列表”和“监控”页签查看相应内容。

表 9-4 节点详情页面

词条	词条描述
概览	单击节点名称，可以进入节点概览页。在这里，您可以方便地查看： ● 资源健康概况：包括节点状态、Pod数量以及异常事件。 ● 节点监控：包括近1小时、近8小时、近24小时以及自定义时间段内的监控概览，其中包括CPU使用率、内存使用率和网络流入/流出速率这些常见的监控指标。 ● Pod使用趋势：包括近1小时、近8小时、近24小时以及自定义时间段内CPU使用量、内存使用量Top5的Pod信息。

词条	词条描述
容器列表	Pod列表中包含了实例名称、状态、命名空间、实例IP、所在节点、重启次数、CPU申请/限制、内存申请/限制，以及CPU和内存使用率等详细信息。 您可以通过在列表上方按照实例名称、状态、命名空间、实例IP和所在节点进行筛选，快速找到需要的实例。 在列表的右上角，您可以单击  按钮来导出全部实例数据，或者选择部分实例进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。 单击实例名称可以查看实例的详细监控数据。更多相关内容，请参见**查看集群内Pod情况**。
监控	在此处，您可以方便地查看节点在近1小时、近8小时、近24小时以及自定义时间段内各维度资源的使用情况。 如需查看更多监控信息，请单击“查看全部仪表盘”，跳转至“仪表盘”页面，相应指导请参见**仪表盘**。

9.1.3.5 查看集群内工作负载情况

如果您需要监控工作负载的资源使用情况，可以前往容器洞察中的工作负载页面查看。该页面提供了指定集群下所有工作负载的综合信息，以及单个工作负载的详细监控数据，包括CPU/内存使用率、网络流入/流出速率、磁盘使用率等。

功能入口

步骤1 登录UCS控制台。

步骤2 在左侧导航栏中选择“容器智能分析”，在“容器洞察 > 集群总览”页面的集群统计列表中，单击集群名称，选择“负载”。

页面呈现了所有工作负载的综合信息，如需深入了解单个工作负载的监控情况，可单击工作负载名称，进入该工作负载的“概览”页面，通过切换“Pod列表”、“监控”页签查看相应内容。

----结束

查看集群内工作负载列表

工作负载列表中包含工作负载名称、状态、实例个数（正常/全部）、命名空间、镜像名称、CPU使用率，以及内存使用率等信息。

图 9-11 工作负载列表页面

您可以利用页面右上角的命名空间和工作负载类型，以及列表上方的工作负载名称、状态和命名空间进行筛选，快速定位所需的工作负载。

在列表的右上角，您可以单击  按钮来导出全部工作负载数据，或者选择部分工作负载进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

查看集群内工作负载详情

在工作负载列表中，单击需要查看详情的工作负载名称，进入该工作负载的详情页面，通过切换“概览”、“Pod列表”和“监控”页签查看相应内容。

表 9-5 工作负载详情页面

词条	词条描述
概览	单击工作负载名称，进入工作负载的“概览”页面。 ● 资源概况：包括负载状态、Pod数量（异常/总数）以及异常事件。 ● 监控概览：包括CPU使用率、内存使用率和网络流入/流出速率这些常见的监控指标。 ● Pod使用趋势：包括工作负载中各Pod的CPU使用率、CPU使用量、内存使用率和内存使用量（在图表左上角切换对应指标），并且支持查看降序Top5和升序Top5数据（在图表右上角进行切换）。
Pod列表	Pod列表中包含了实例名称、状态、命名空间、实例IP、所在节点、重启次数、CPU申请/限制、内存申请/限制，以及CPU和内存使用率等详细信息。 您可以通过在列表上方按照实例名称、状态、命名空间、实例IP和所在节点进行筛选，快速找到需要的实例。 在列表的右上角，您可以单击  按钮来导出全部实例数据，或者选择部分实例进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。 单击实例名称可以查看实例的详细监控数据。更多相关内容，请参见**查看集群内Pod情况**。
监控	在此处，您可以方便地查看工作负载在近1小时、近8小时、近24小时以及自定义时间段内各维度资源的使用情况。 如需查看更多监控信息，请单击“查看全部仪表盘”，跳转至“仪表盘”页面，相应指导请参见**仪表盘**。

9.1.3.6 查看集群内 Pod 情况

如果您需要监控Pod的资源使用情况，可以前往容器洞察中的Pod页面查看。该页面提供了指定集群下所有Pod的综合信息，以及单个Pod的详细监控数据，包括CPU/内存使用率、网络流入/流出速率、磁盘使用率等。

说明

页面中容器组、Pod以及实例是指同一个概念。

功能入口

步骤1 登录UCS控制台。

步骤2 在左侧导航栏中选择“容器智能分析”，在“容器洞察 > 集群总览”页面的集群统计列表中，单击集群名称，选择“Pod”。

页面呈现了所有实例的综合信息，如需深入了解单个实例的监控情况，可单击实例名称，进入该实例的“概览”页面，通过切换“容器列表”、“监控”页签查看相应内容。

----结束

查看集群内 Pod 列表

容器组列表中包含实例名称、状态、命名空间、实例IP、所在节点、重启次数、CPU申请/限制、内存申请/限制、CPU使用率，以及内存使用率等信息。

图 9-12 Pod 列表页面

您可以利用页面左上方的命名空间，以及列表上方的实例名称、状态、实例IP和所在节点进行筛选，快速定位所需的实例。

在列表的右上角，您可以单击  按钮来导出全部实例数据，或者选择部分实例进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。

查看集群内 Pod 详情

在Pod列表中，单击需要查看详情的实例名称，进入该实例的详情页面，通过切换“概览”、“容器列表”和“监控”页签查看相应内容。

表 9-6 Pod 详情页面

词条	词条描述
概览	单击实例名称，进入实例概览页。 ● 资源概况：Pod状态、容器数量（异常/总数）以及异常事件。 ● 监控概览：包括CPU使用率、内存使用率和网络流入/流出速率这些常见的监控指标。 ● 容器使用趋势：包括Pod中各容器的CPU使用率、CPU使用量、内存使用率和内存使用量（在图表左上角切换对应指标），并且支持查看降序Top5和升序Top5数据（在图表右上角进行切换）。
容器列表	容器列表中包含了容器名称、状态、命名空间、重启次数，以及镜像等详细信息。 您可以通过在列表上方按照容器名称、状态和命名空间进行筛选，快速找到需要的容器。 在列表的右上角，您可以单击  按钮来导出全部容器数据，或者选择部分容器进行导出，此时仅导出所选中的数据。导出的文件为“.xlsx”格式，文件命名中包含时间戳。
监控	在此处，您可以方便地查看实例在近1小时、近8小时、近24小时以及自定义时间段内各维度资源的使用情况。 如需查看更多监控信息，请单击“查看全部仪表盘”，跳转至“仪表盘”页面，相应指导请参见**仪表盘**。

9.1.3.7 查看集群内事件情况

Kubernetes事件涵盖了集群的运行状态和各类资源的调度情况，对运维人员日常观察资源的变更以及定位问题均有帮助。如果您需要监控集群内事件，可以前往“容器洞察 > 事件”页面查看。为了实现这一目标，您需要为集群安装log-agent插件，该插件可以采集Kubernetes事件，并在“容器洞察 > 事件”页面进行展示。

功能入口

步骤1 登录UCS控制台。

步骤2 在左侧导航栏中选择“容器智能分析”，在“容器洞察 > 集群总览”页面的集群统计列表中，单击集群名称，选择“事件”。

----结束

查看集群内事件详情

事件页面分为两个页签：“概览”和“事件”。在“概览”页签中，您可以查看集群中事件的总数、趋势和排序信息；在“事件”页签中，可以查看事件的详细信息，包括事件名称、类型、内容，以及触发该事件的资源的相关信息等。

表 9-7 事件详情页面

页签	页签简介
概览	“概览”页面默认展示集群中所有命名空间的事件统计信息，您也可以在右上角的下拉框中切换命名空间，以查看指定命名空间下的事件数据。 ● 事件总数：Normal和Warning事件的数量分布情况，呈现为一个圆环图。 ● Warning事件资源维度TOP5：排名前五的Warning事件数量所对应的资源信息。 ● Warning事件环比：近24小时Warning事件数与前24小时之间的比较。 ● Warning 事件趋势（24小时）：24小时内Warning事件的数量变化趋势。 ● Normal 事件趋势（24小时）：24小时内Normal事件的数量变化趋势。 ● 24小时事件数量TOP10：24小时内事件数量排名前十的事件名称。
事件	“事件”页面展示单位时间内集群中事件的详细信息，包括事件名称、类型、内容，以及触发该事件的资源的相关信息等。 **事件搜索** 事件页面的主要功能是展示按照一定条件搜索出的指定资源的事件信息，包括Normal/Warning事件趋势和详情。这样，用户可以更加方便地查看与该资源相关的事件信息。 您可以按照如下几种方式进行事件搜索。 ● 在输入框里输入要搜索的事件名称，选择命名空间或者事件类型，单击“搜索”按钮进行搜索。 ● 单击“高级搜索”，按照工作负载、节点、容器组、事件内容、资源类型或者资源名称进行搜索。 ● 也可以在左上角选择事件发生的时间范围，包括近1小时、近1天、近1周和自定义。 **事件列表** 您可以在列表中查看满足搜索条件的事件详情，包括最近发生时间、事件名称、资源类型、资源名称、事件内容、事件类型和发生次数。单击操作列的“历史事件”，在弹出的对话框中将展示当前资源类型和资源名称下的所有事件。

9.1.4 健康诊断

概述

健康诊断是容器智能分析的一个重要功能，用于诊断集群的健康状态。开通容器智能分析后，健康诊断将基于集群的配置和kube-prometheus-stack插件上报至AOM的指标，从集群、节点、工作负载、核心插件、外部依赖的维度出发，提供全面的集群健康状态检查。同时，该功能还基于Kubernetes集群的运维最佳实践，提供相应的诊断结果和修复建议。

约束与限制

- 集群版本高于v1.17。
- 集群处于“运行中”状态。

查看巡检详情

步骤1 选择一个容器舰队或者未加入舰队的集群。

图 9-13 选择舰队或未加入舰队的集群

步骤2 单击“健康诊断”页签，您可通过健康诊断功能查看各个集群的正常比例及风险数量。

图 9-14 健康诊断

步骤3 下拉到“诊断结果”，查看当前集群巡检诊断结果。

单击 ，单击“诊断详情”，可进入健康诊断详情页查看诊断项和诊断结果。

图 9-15 诊断结果

----结束

配置定时巡检规则

步骤1 选择一个容器舰队或者未加入舰队的集群。

图 9-16 选择舰队或未加入舰队的集群

步骤2 单击“容器洞察 > 集群总览”页签查看已开启监控的集群。

步骤3 单击上方“健康诊断”，进入诊断详情页，在右边开启“定时巡检”，配置定时任务启动的时间。

集群将在指定时间自动开始集群巡检任务。单个集群，每天仅支持配置一个定时巡检时间。

图 9-17 定时巡检设置

说明

也可按照**查看巡检详情**指导进入单集群巡检详情页面。

----结束

发起诊断

步骤1 按照**查看巡检详情**指导进入单集群巡检页面。

步骤2 在下方“巡检集群”中选择未巡检集群，单击“马上诊断”。

集群将开始执行诊断。诊断结束后，页面将自动刷新并展示诊断结果，其中无风险项将自动隐藏。

健康诊断将针对不同维度的巡检项，归纳Kubernetes中常见的问题，并提供相应的修复建议。您可以单击“诊断详情”查看具体诊断项的详细信息与修复建议。

图 9-18 诊断详情

----结束

支持的巡检项

表 9-8 CCE 集群巡检项

巡检维度	集群巡检场景	巡检项
集群	集群资源规划能力	集群Master节点是否高可用
		集群CPU的Request水位是否超过80%
		集群CPU的Limit水位是否超过150%
		集群内存的Request水位是否超过80%
		集群内存的Limit水位是否超过150%
		集群版本是否超期
	集群运维能力	集群kube-prometheus-stack插件状态是否正常
		集群log-agent插件状态是否正常
		集群npd插件状态是否正常

	集群配置	安全组配置是否正确
核心插件	coredns插件状态	coredns近24小时cpu使用率最大值是否超过80%
		coredns近24小时内存使用率最大值是否超过80%
		coredns近24小时是否存在域名解析失败请求次数
		coredns近24小时P99请求时延是否超过5s
		coredns插件状态
	everest插件状态	everest插件状态
		everest近24小时CPU使用率最大值是否超过80%
		everest近24小时内存使用率最大值是否超过80%
	kube-prometheus-stack插件状态	kube-prometheus-stack近24小时CPU使用率最大值是否超过80%
		kube-prometheus-stack近24小时内存使用率最大值是否超过80%
		kube-prometheus-status插件状态
		kube-prometheus-status近24小时是否出现OOM
		kube-prometheus-status在Server部署模式下，prometheus-server的PVC使用率是否超过80%
	log-agent插件状态	log-agent插件状态
		LTS日志组、日志流是否创建成功
		LTS日志组结构化是否创建成功
	autoscaler插件状态	集群在开启节点池弹性扩缩容条件下，autoscaler插件状态是否可用
节点	节点状态	节点状态是否就绪
		节点状态不可调度
		节点kubelet状态
	节点配置	节点内存的Requset是否超过80%
		节点CPU的Request是否超过80%
		节点内存的Limit检查是否超过150%

		节点CPU的Limit检查是否超过150%
	节点资源水位诊断	节点24小时内CPU使用率最大值是否超过80%
		节点24小时内内存使用率最大值是否超过80%
		节点磁盘使用率是否超过80%
		节点PID使用量是否正常
		节点24小时内是否发生OOM事件
负载	Pod状态	Pod状态检查
	Pod负载状态	Pod在24小时内是否发生OOM
		Pod的24小时内CPU使用率最大值是否超过80%
		Pod的24小时内内存使用率最大值是否超过80%
	Pod配置	Pod中的容器是否配置Request
		Pod中的容器是否配置Limit
	Pod探针配置	Pod中的容器是否配置存活探针
		Pod中的容器是否配置就绪探针
外部依赖	租户节点资源配额	租户云硬盘配额是否超过90%
		租户ECS配额充足是否超过90%

表 9-9 本地集群巡检项

巡检维度	集群巡检场景	巡检项
集群	集群资源规划能力	集群Master节点是否高可用
		集群CPU的Request水位是否超过80%
		集群CPU的Limit水位是否超过150%
		集群内存的Request水位是否超过80%
		集群内存的Limit水位是否超过150%
	集群运维能力	集群kube-prometheus-stack插件状态是否正常
		集群log-agent插件状态是否正常
核心插件	kube-prometheus-stack插件状态	kube-prometheus-stack近24小时CPU使用率最大值是否超过80%

		kube-prometheus-stack近24小时内存使用率最大值是否超过80%
		kube-prometheus-status插件状态
		kube-prometheus-status近24小时是否出现OOM
	log-agent插件状态	log-agent插件状态
		LTS日志组、日志流是否创建成功
		LTS日志组结构化是否创建成功
节点	节点状态	节点状态是否就绪
		节点状态不可调度
		节点kubelet状态
	节点配置	节点内存的Requset是否超过80%
		节点CPU的Request是否超过80%
		节点内存的Limit检查是否超过150%
		节点CPU的Limit检查是否超过150%
	节点资源水位诊断	节点24小时内CPU使用率最大值是否超过80%
		节点24小时内内存使用率最大值是否超过80%
		节点磁盘使用率是否超过80%
		节点PID使用量是否正常
		节点24小时内是否发生OOM事件
负载	Pod状态	Pod状态检查
	Pod负载状态	Pod在24小时内是否发生OOM
		Pod的24小时内CPU使用率最大值是否超过80%
		Pod的24小时内内存使用率最大值是否超过80%
	Pod配置	Pod中的容器是否配置Request
		Pod中的容器是否配置Limit
	Pod探针配置	Pod中的容器是否配置存活探针
		Pod中的容器是否配置就绪探针
外部依赖	租户节点资源配额	租户云硬盘配额是否超过90%

		租户ECS配额充足是否超过90%

表 9-10 附着集群、多云集群、伙伴云集群巡检项

巡检维度	集群巡检场景	巡检项
集群	集群资源规划能力	集群Master节点是否高可用
		集群CPU的Request水位是否超过80%
		集群CPU的Limit水位是否超过150%
		集群内存的Request水位是否超过80%
		集群内存的Limit水位是否超过150%
	集群运维能力	集群kube-prometheus-stack插件状态是否正常
核心插件	kube-prometheus-stack插件状态	kube-prometheus-stack近24小时CPU使用率最大值是否超过80%
		kube-prometheus-stack近24小时内存使用率最大值是否超过80%
		kube-prometheus-status插件状态
		kube-prometheus-status近24小时是否出现OOM
节点	节点状态	节点状态是否就绪
		节点状态不可调度
		节点kubelet状态
	节点配置	节点内存的Requset是否超过80%
		节点CPU的Request是否超过80%
		节点内存的Limit检查是否超过150%
		节点CPU的Limit检查是否超过150%
	节点资源水位诊断	节点24小时内CPU使用率最大值是否超过80%
		节点24小时内内存使用率最大值是否超过80%
		节点磁盘使用率是否超过80%
		节点PID使用量是否正常
		节点24小时内是否发生OOM事件
负载	Pod状态	Pod状态检查

	Pod负载状态	Pod在24小时内是否发生OOM
		Pod的24小时内CPU使用率最大值是否超过80%
		Pod的24小时内内存使用率最大值是否超过80%
	Pod配置	Pod中的容器是否配置Request
		Pod中的容器是否配置Limit
	Pod探针配置	Pod中的容器是否配置存活探针
		Pod中的容器是否配置就绪探针
外部依赖	租户节点资源配额	租户云硬盘配额是否超过90%
		租户ECS配额充足是否超过90%

9.1.5 仪表盘

仪表盘可将不同图表展示到同一个屏幕上，通过不同的仪表形式来展示资源数据，例如，曲线图、数字图等，进而全面、深入地掌握监控数据。

查看/切换视图

步骤1 选择一个容器舰队或者未加入舰队的集群。

图 9-19 选择舰队或未加入舰队的集群

步骤2 选择“仪表盘”页签，默认展示集群视图。

步骤3 设置查看视图的相关参数。各个视图可供设置的参数不同，请参见表9-11。

步骤4 设置视图的时间窗。

在页面右上角处，选择时间段，或者自定义时间，并单击刷新界面。

步骤5 容器智能分析仪表盘提供了预置视图，您可单击视图名称边上的“切换视图”按钮，选择需要的视图查看监控数据。系统预置视图如表9-11所示。

表 9-11 预置视图

视图名称	视图参数	视图中包含的监控指标
集群视图（默认视图）	集群	• 节点数/磁盘不可用节点数/不可用节点数 • CPU/内存使用率 • CPU/内存Requests水位 • CPU/内存Limits水位 • Pod/容器数 • CPU/内存使用量 • 网络接收/发送速率 • 网络平均接收/发送速率 • 接收/发送数据包速率 • 丢包率(接收/发送) • 磁盘IOPS(读+写) • ThroughPut(读+写)
APIServer视图	• 集群 • 实例	• 存活数 • QPS • 请求成功率(读) • 处理中请求数 • 请求速率(读/写) • 请求错误率(读/写) • 请求时延(读/写)(99分位时延) • 工作队列增加速率/深度 • 工作队列时延(99分位时延) • 内存/CPU使用量 • Go routine数
Pod视图	• 集群 • 命名空间 • pod	• 容器数/运行中容器数 • Pod状态 • 容器重启次数 • CPU/内存使用量 • CPU Throttling • 网络接收/发送速率 • 接收/发送数据包速率 • 丢包率(接收/发送) • 磁盘IOPS(读+写) • ThroughPut(读+写) • 文件系统使用率/使用量

视图名称	视图参数	视图中包含的监控指标
主机视图	• 集群 • 节点	• CPU/内存使用率 • 平均负载 • 内存使用量 • 磁盘写入/读取速率 • 磁盘空间使用 • 磁盘IO
Node视图	• 集群 • 节点	• CPU/内存使用率 • CPU/内存Requests水位 • CPU/内存Limits水位 • 内存使用量 • 网络接收/发送速率 • 接收/发送数据包速率(Pod) • 接收/发送数据包速率 • 丢包率(接收/发送) • 磁盘IOPS(读+写) • ThroughPut(读+写)
CoreDNS视图	• 集群 • 实例	• 请求速率(记录类型/区域/DO标志位) • 请求数据包(UDP/TCP) • 响应速率(响应状态码) • 响应时延 • 响应数据包(UDP/TCP) • 缓存大小 • 缓存命中率
PVC视图	• 集群 • 命名空间 • PV • PVC	• PV/PVC状态 • PVC使用量/使用率 • PVC inodes使用量/使用率 • PVC每小时/每天/每周使用率 • 一周后PVC使用量

视图名称	视图参数	视图中包含的监控指标
Kubelet	• 集群 • 实例	• 运行中Kubelet/Pod/容器 • 实际卷/期望卷/配置错误数量 • 操作速率/错误率/时延 • Pod启动速率/时延(99分位) • 存储操作速率/错误率/时延(99分位) • 控制组管理器操作速率/时延(99分位) • PLEG relist速率/间隔/时延(99分位) • RPC速率 • 请求时延(99分位) • 内存/CPU使用量 • Go routine数
Prometheus	• 集群 • job • instance	• Target同步间隔 • Target数 • 平均拉取间隔 • 拉取失败 • Appended Samples • Head中Series数/Chunks数 • 查询速率/阶段时延
Prometheus Remote Write	• 集群 • 实例 • url	• Highest Timestamp In vs. Highest Timestamp Sent • Rate5m • Rate in vs. succeeded or dropped 5m • 当前/最大/最小/期望分片数 • 分片容量 • 挂起样本数 • TSDB/远程写入当前段 • 样本丢弃率/失败率/重试率 • 入队失败重试率
工作负载	• 集群 • 命名空间 • 类型 • 工作负载	• CPU/内存使用量 • 网络接收/发送速率 • 网络平均接收/发送速率 • 接收/发送数据包速率 • 丢包率(接收/发送)

视图名称	视图参数	视图中包含的监控指标
XGPU视图	集群	• 集群--XGPU设备显存使用率 • 集群--XGPU设备算力使用率 • 节点--XGPU设备显存使用率 • 节点--XGPU设备算力使用率 • 节点--XGPU设备数量 • 节点--XGPU设备显存分配量 • GPU卡--XGPU设备显存使用率 • GPU卡--XGPU设备显存分配量 • GPU卡--XGPU设备显存分配率 • GPU卡--XGPU设备算力使用率 • GPU卡--XGPU设备数量 • GPU卡--调度策略 • GPU卡--不健康的XGPU设备数量 • 容器显存分配量 • 容器算力使用率 • 容器显存使用量 • 容器显存使用率

----结束

9.2 日志中心

9.2.1 日志中心概述

Kubernetes日志可以协助您排查和诊断问题。本文介绍UCS如何通过多种方式进行Kubernetes日志管理。

- 您可以方便地使用云原生日志采集插件采集应用日志并上报LTS，从而更好地利用LTS日志服务提供给您的各种日志统计分析等功能。具体操作，请参见**收集数据面日志**。
- 支持收集集群控制平面组件日志和Kubernetes审计日志，将日志从master节点采集到您账号的LTS日志服务的日志流中。具体操作，请参见**收集控制面组件日志**和**收集Kubernetes审计日志**。
- 支持收集集群Kubernetes事件，将Kubernetes事件从集群内采集到您账号的LTS日志服务的日志流中，以便对Kubernetes事件进行持久化存储和统计分析。具体操作，请参见**收集Kubernetes事件**。

约束限制

日志中心仅支持**华为云集群**和**本地集群**。

9.2.2 开启日志中心

云原生日志采集插件是基于开源fluent-bit和opentelemetry构建的云原生日志、Kubernetes事件采集插件。云原生日志采集插件支持基于CRD的日志采集策略，可以根据您配置的策略规则，对集群中的容器标准输出日志、容器文件日志、节点日志及Kubernetes事件日志进行采集与转发。同时支持上报Kubernetes事件到AOM，用于配置事件告警，默认上报所有异常事件和部分正常事件。

约束与限制

- 仅支持1.21及以上版本的华为云集群或本地集群。
- 每个集群限制50条日志规则。
- 不采集.gz、.tar、.zip后缀类型的日志文件。
- 采集容器文件日志时，若节点存储模式为Device Mapper模式，路径配置必须为节点数据盘挂载路径。
- 若容器运行时为containerd模式，容器标准输出日志中的多行配置暂不生效。(1.3.0及以上版本没有该限制)
- 每个集群限制单行日志采集速率不超过10000条/秒，多行日志不超过2000条/秒。
- 每个节点中，所有日志策略监听的文件数不能超过4096个文件 。
- 如果业务容器的数据目录是通过数据卷（Volume）挂载的，插件不支持采集它的父目录，需设置采集目录为完整的数据目录。
- 容器运行时间建议不小于1分钟，防止日志文件删除过快，无法及时采集。

费用说明

LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用。

本地集群需要选择网络接入方式，若选择云专线或VPN方式接入，VPC终端节点将按照时长计费。

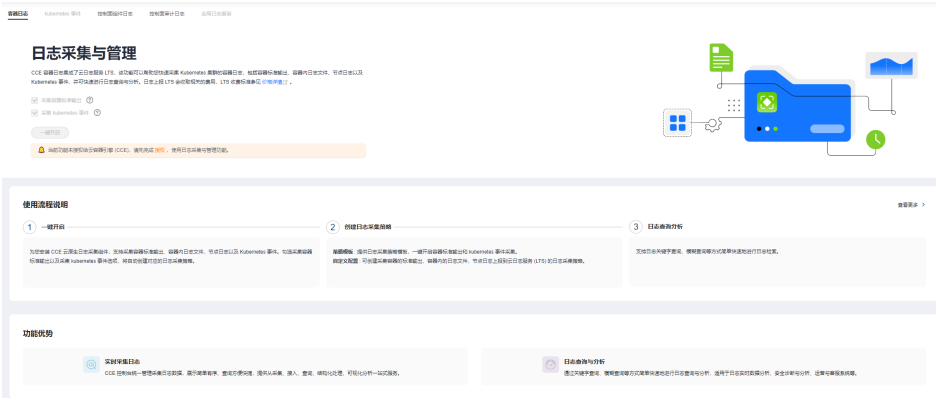
启用云原生日志采集插件采集日志

步骤1 登录UCS控制台，进入“容器舰队”，单击舰队名称进入舰队页面。

步骤2 选择“容器集群”，单击集群名称进入集群，选择左侧导航栏的“日志中心”。

步骤3 （仅华为云集群）未进行授权的用户需要先授权，已授权的用户直接跳转下一步。在弹出框中单击“确认授权”。

步骤4 （仅华为云集群）页面单击“一键开启”，等待约30秒后，页面自动跳转。



- 采集容器标准输出：开启后，将创建名为default-stdout的日志策略，并上报所有命名空间下的标准输出到云日志服务（LTS）。
- 采集Kubernetes事件：开启后，将创建名为default-event的日志策略，并上报所有命名空间下的Kubernetes事件到云日志服务（LTS）。

步骤5 （仅本地集群）为本地集群云原生日志采集插件授权。详细步骤请参见**本地集群安装云原生日志插件前置授权**。

步骤6 （仅本地集群）页面单击“立即开启”，在弹窗中进行日志采集配置与网络配置，等待约30秒后，页面自动跳转。

表 9-12 本地集群日志采集配置与网络配置

配置	描述
日志采集配置	• 采集容器标准输出：开启后，将创建名为default-stdout的日志策略，并上报所有命名空间下的标准输出到云日志服务（LTS）。 • 采集Kubernetes事件：开启后，将创建名为default-event的日志策略，并上报所有命名空间下的Kubernetes事件到云日志服务（LTS）和应用运维管理（AOM）。 • Kubernetes审计日志：采集Kubernetes审计日志并上报到云日志服务（LTS）。 • kube-apiserver日志：采集控制面组件kube-apiserver组件日志并上报到云日志服务（LTS）。 • kube-controller-manager日志：采集控制面组件kube-controller-manager组件日志并上报到云日志服务（LTS）。 • kube-scheduler日志：采集控制面组件kube-scheduler组件日志并上报到云日志服务（LTS）。
网络配置	• 公网接入：通过公网Internet接入，要求集群能够访问公网，具有弹性灵活、成本低、易接入的优势。公网接入要求集群能够访问公网，请确保集群已符合此条件，否则会接入失败。 • 云专线/VPN接入：通过云专线（DC）或虚拟专用网络（VPN）服务将云下网络与云上虚拟私有云（VPC）连通，并利用VPC终端节点通过内网与容器智能分析建立连接，具有高速、低时延、安全的优势。详情见**本地集群使用云专线/VPN上报日志**。

----结束

常见问题处理

插件中除log-operator外组件均未就绪，且出现异常事件“实例挂卷失败”。

解决方案：请查看log-operator日志，安装插件时，其余组件所需的配置文件需要log-operator生成，log-operator生成配置出错，会导致所偶遇组件无法正常启动。

9.2.3 收集数据面日志

云原生日志采集插件是基于开源fluent-bit和opentelemetry构建的云原生日志、Kubernetes事件采集插件。云原生日志采集插件支持基于CRD的日志采集策略，可以根据您配置的策略规则，对集群中的容器标准输出日志、容器文件日志、节点日志及Kubernetes事件日志进行采集与转发。同时支持上报Kubernetes事件到AOM，用于配置事件告警，默认上报所有异常事件和部分正常事件。

费用说明

LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用。

集群数据面组件说明

当前支持收集以下两种类型的控制面日志，每个日志流对应一个Kubernetes控制层面组件。关于这些组件的更多信息，请参见**Kubernetes组件**。

表 9-13 集群控制面组件说明

类别	组件	日志流	说明
数据面组件日志	default-stdout	stdout-{clusterID}	采集标准输出。默认日志组：k8s-logs-{集群ID}。
	default-event	event-{clusterID}	采集Kubernetes事件。默认日志组：k8s-logs-{集群ID}。

使用云原生日志采集插件采集日志

步骤1 查看并配置日志采集策略。

1. 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 右上角单击“日志采集策略”，将显示当前集群所有上报LTS的日志策略。

图 9-20 查看日志策略

若安装插件时勾选了采集标准输出和采集Kubernetes事件，将创建两个日志策略，并对接默认的LTS日志组、日志流。

3. 创建日志策略：单击上方“创建日志策略”，输入要采集的配置信息。

   策略模板：若安装插件时未勾选需要采集的日志策略，或者删除了对应的日志策略，可通过该方式重新创建默认日志策略。

图 9-21 使用策略模板

自定义策略：用于配置自定义日志策略。

图 9-22 自定义策略

表 9-14 自定义策略参数说明

参数	说明
日志类型	指定采集哪类日志。 - 容器标准输出：用于采集容器标准输出，可以按命名空间、工作负载名称、实例标签配置采集策略。 - 容器文件路径：用于采集容器内的日志，可以按工作负载和实例标签配置采集策略。 - 节点文件路径：用于采集节点上的日志文件，一条日志策略只能配置一个文件路径。
日志源	采集哪些容器的日志。 - 所有容器：可以指定采集某个命名空间的所有容器，如不指定则采集所有命名空间的容器。 - 指定工作负载：指定采集哪些工作负载容器的日志，可以指定采集工作负载中具体容器的日志，如不指定则采集所有容器的日志。 - 指定实例标签：根据标签指定采集哪些工作负载容器的日志，可以指定采集工作负载中具体容器的日志，如不指定则采集所有容器的日志。

参数	说明
路径配置	用于配置需要采集的日志路径。 文件路径必须以/ 开头，只能包含大写字母、小写字母、数字或特殊字符-_/*?，且长度不能超过512个字符。 文件名称只能包含大写字母、小写字母、数字或特殊字符-_*?.。 日志文件夹：请填写绝对路径。日志文件名：不支持.gz、.tar、.zip后缀类型。 最多有三级目录采用通配符匹配，且第一级目录不能使用通配符。 目录名和文件名支持完整名称和通配符模式，通配符只支持星号（*）和半角问号（?）。 星号（*）表示匹配多个任意字符。半角问号（?）表示匹配单个任意字符。例如： – 日志路径为/var/logs/* 文件名*.log，表示/var/logs下所有目录中后缀名为.log的文件。 – 日志路径为 /var/logs/app_* 文件名*.log，表示/var/logs目录下所有符合app_*格式的目录中后缀名为.log的文件。 如果业务容器的数据目录是通过数据卷（Volume）挂载的，插件不支持采集它的父目录，需设置采集目录为完整的数据目录。例如/var/log/service目录是数据卷挂载的路径，则设置采集目录为/var/log或/var/log/*将采集不到该目录下的日志，需设置采集目录为/var/log/service。
日志格式	– 单行文本 每条日志仅包含一行文本，以换行符 \n 作为各条日志的分界线。 – 多行文本 有些程序打印的日志存在一条完整的日志数据跨占多行（例如 Java 程序日志）情况，日志采集系统默认是按行采集。如果您想在日志采集系统中按整条显示日志，可以开启多行文本，采用首行正则的方式进行匹配，当选择多行文本时，需填写日志匹配格式。 例如： 需采集的日志格式如下，则需填写时间的正则匹配，在日志匹配格式处填写：\d{4}-\d{2}-\d{2} \d{2}\:\d{2}\:\d{2}.* 则下面以日期开头三行日志会作为一条完整日志。 2022-01-01 00:00:00 Exception in thread "main" java.lang.RuntimeException: Something has gone wrong, aborting! at com.myproject.module.MyProject.badMethod(MyProject.java:22) at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)

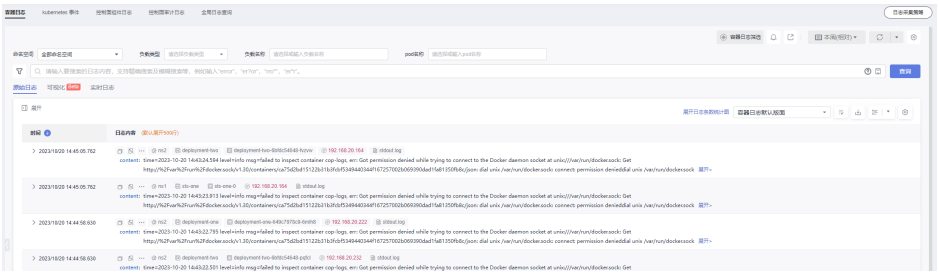
参数	说明
上报到云日志服务(LTS)	用于配置日志上报的日志组和日志流。 – 使用默认日志组/日志流：将为您自动选择默认日志组（k8s-log-{集群ID}）和默认的日志流（stdout-{集群ID}）。 – 自定义日志组/日志流：可在下拉框选择任意日志组和日志流。
日志组	日志组是云日志服务进行日志管理的基本单位。如果您未创建日志组，CCE会提示您进行创建，默认名称为k8s-log-{集群ID}，如 k8s-log-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3。
日志流	日志流（LogStream）：日志流是日志读写的基本单位，日志组中可以创建日志流，将不同类型的日志分类存储，方便对日志进一步分类管理。在安装插件或者根据模板创建日志策略时，会自动创建以下日志流： – 容器日志：默认名称为stdout-{集群ID}，如 stdout-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3 – k8s事件：默认名称为event-{集群ID}，如 event-bb7eaa87-07dd-11ed-ab6c-0255ac1001b3

4. 编辑日志策略：单击“编辑”按钮，可对已经存在的日志策略进行修改。
5. 删除日志策略：单击“删除”按钮，可对已经存在的日志策略进行删除。

步骤2 查看日志。

1. 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 日志中心下有5个页签，支持不同类型日志查看。
   – 容器日志：显示默认日志组（k8s-log-{集群ID}）下默认日志流（stdout-{集群ID}）中的所有日志数据，华为云集群支持通过工作负载搜索。

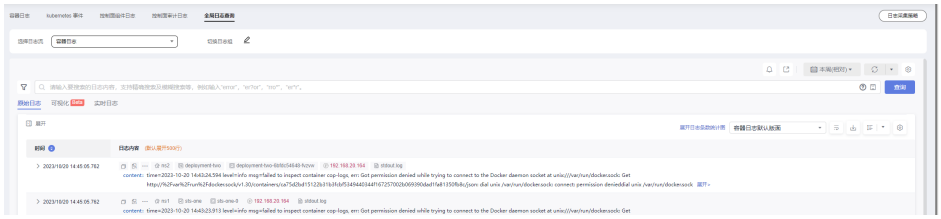
   图 9-23 容器日志查询

   – Kubernetes事件：显示默认日志组（k8s-log-{集群ID}）下默认日志流（event-{集群ID}）中的所有日志数据，用于查询集群产生的Kubernetes事件。
   – 控制面组件日志：显示默认日志组（k8s-log-{集群ID}）下默认日志流（{组件名}-{集群ID}）中的所有日志数据，用于查看集群控制面重要组件的日志信息。

- 控制面审计日志：显示默认日志组（k8s-log-{集群ID}）下默认日志流audit-{集群ID}）中的所有日志数据，用于查看集群控制面审计日志信息。
- 全局日志查询：支持查看所有日志组日志流下的日志信息。可通过选择日志流查看所选日志流中的日志信息，默认会选择集群默认日志组（k8s-log-{集群ID}），可通过单击切换日志组右侧的图标切换其他日志组。

图 9-24 全局日志查询



3. 单击右上角“日志采集策略”，单击“查看日志”，可以直接跳转至对应日志策略的日志列表。

图 9-25 查看日志

----结束

常见问题处理

1. **log-operator标准输出报错：Failed to create log group, the number of log groups exceeds the quota**

   示例：

   ```
   2023/05/05 12:17:20.799 [E] call 3 times failed, resion: create group failed, projectID: xxx, groupName: k8s-log-xxx, err: create groups status code: 400, response: {"error_code":"LTS.0104","error_msg":"Failed to create log group, the number of log groups exceeds the quota"}, url: https://lts.cn-north-4.myhuaweicloud.com/v2/xxx/groups, process will retry after 45s
   ```

   解决方案：LTS日志组有配额限制，如果出现该报错，请前往LTS下删除部分无用的日志组。限制详情请参见：日志组。

2. **配置了容器文件路径采集，采集的目录不是挂载到容器内的，且节点引擎为docker，采集不到日志。**

   解决方案：

   请检查工作负载所在节点的容器存储模式是否为deviceMapper，deviceMapper不支持采集容器内日志（创建日志策略时已提示此限制，如图9-26所示）。检查方法如下：

   a. 进入业务工作负载所在节点。

   b. 执行**docker info | grep "Storage Driver"**。

c. 若返回的Storage Driver值为devicemapper，则该日志无法采集。

图 9-26 创建日志策略

3. **日志无法上报，otel组件标准输出报错：log's quota has full**

```
2023-08-16T09:03:20.067+0800    error    exporterhelper/queued_retry.go:361    Exporting failed. Try enabling retry_
on_failure config option to retry on retryable errors    {"kind": "exporter", "data_type": "logs", "name": "lts/defaul
t-event", "error": "fail to push event data via lts exporter: read body {\"errorCode\":\"SWCSTG.ALS.200.210\",\"error
Message\":\"projectid                                    s quota has full!!\",\"result\":null} error"}
go.opentelemetry.io/collector/exporter/exporterhelper.(*retrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:361
go.opentelemetry.io/collector/exporter/exporterhelper.(*logsExporterWithObservability).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:142
go.opentelemetry.io/collector/exporter/exporterhelper.(*queuedRetrySender).send
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/queued_retry.go:295
go.opentelemetry.io/collector/exporter/exporterhelper.NewLogsExporterWithContext.func2
        go.opentelemetry.io/collector@v0.58.0/exporter/exporterhelper/logs.go:122
go.opentelemetry.io/collector/consumer.ConsumeLogsFunc.ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/consumer/logs.go:36
go.opentelemetry.io/collector/service/internal/fanoutconsumer.(*logsConsumer).ConsumeLogs
        go.opentelemetry.io/collector@v0.58.0/service/internal/fanoutconsumer/logs.go:77
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).handleEvent
        cieotelcol/receiver/k8seventsreceiver/receiver.go:138
cieotelcol/receiver/k8seventsreceiver.(*k8seventsReceiver).startWatch.func1
        cieotelcol/receiver/k8seventsreceiver/receiver.go:116
k8s.io/client-go/tools/cache.ResourceEventHandlerFuncs.OnAdd
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:232
k8s.io/client-go/tools/cache.processDeltas
        k8s.io/client-go@v0.24.3/tools/cache/controller.go:441
k8s.io/client-go/tools/cache.newInformer.func1
```

解决方案：

云日志服务（LTS）有免费赠送的额度，超出后将收费，报错说明免费额度已用完，如果需要继续使用，请前往云日志服务控制台“配置中心>配额设置”，打开“超额继续采集日志”开关。

4. 采集容器内日志，且采集目录配置了通配符，日志无法采集。

排查方法：请检查工作负载配置中Volume挂载情况，如果业务容器的数据目录是通过数据卷（Volume）挂载的，插件不支持采集它的父目录，需设置采集目录为完整的数据目录。例如/var/log/service目录是数据卷挂载的路径，则设置采集目录为/var/log或/var/log/*将采集不到该目录下的日志，需设置采集目录为/var/log/service。

解决方案：若日志生成目录为/application/logs/{应用名}/*.log，建议工作负载挂载Volume时，直接挂载/application/logs，日志策略中配置采集路径为/application/logs/*/*.log

9.2.4 收集控制面组件日志

集群支持对用户开放集群控制节点的日志信息。在日志管理页面可以选择需要上报的日志主题。支持kube-controller-manager、kube-apiserver、kube-scheduler三个组件的日志。

费用说明

LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用。

约束与限制

- 华为云集群必须为v1.21.7-r0及以上补丁版本、v1.23.5-r0及以上补丁版本或1.25版本。
- 请确保云日志服务LTS资源配额充足，LTS的默认配额请参见**基础资源**。

集群控制面组件说明

当前支持收集以下三种类型的控制面日志，每个日志流对应一个Kubernetes控制层面组件。关于这些组件的更多信息，请参见**Kubernetes组件**。

表 9-15 集群控制面组件说明

类别	组件	日志流	说明
控制面组件日志	kube-apiserver	kube-apiserver-{{clusterID}}	kube-apiserver组件是暴露Kubernetes API接口的控制层面的组件。更多信息，请参见**kube-apiserver**。
	kube-controller-manager	kube-controller-manager-{{clusterID}}	kube-controller-manager组件是Kubernetes集群内部的管理控制中心，内嵌了Kubernetes发布版本中核心的控制链路。更多信息，请参见**kube-controller-manager**。
	kube-scheduler	kube-scheduler-{{clusterID}}	kube-scheduler组件是Kubernetes集群的默认调度器。更多信息，请参见**kube-scheduler**。

开启本地集群控制面日志

集群未安装云原生日志采集插件

安装云原生日志采集插件时，可通过勾选对应控制面组件，创建默认日志采集策略，采集对应组件日志上报到LTS。安装方法见：**启用云原生日志采集插件采集日志**。

集群已安装云原生日志采集插件

1. 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 右上角单击“日志采集策略”，将显示当前集群所有上报LTS的日志策略。
3. 单击上方“创建日志策略”，输入要采集的配置信息。

   策略模板：若安装插件时未开启控制面组件的采集策略，或者删除了对应的日志策略，可通过该方式重新创建对应组件采集策略。

   图 9-27 创建日志策略

4. 日志查看：可直接在“日志中心”页面，“控制面组件日志”页签中查看，选择日志策略配置的日志流名称，即可查看上报到云日志服务（LTS）的日志。

图 9-28 查看日志

开启华为云集群控制面日志

创建集群时开启

1. 登录云容器引擎（CCE）控制台。
2. 在控制台上方导航栏，选择集群，单击“购买”。
3. 在“插件选择”页面中，勾选“云原生日志采集”

4. 在“插件配置”页面中，“云原生日志采集插件”选择“自定义安装”，单击“控制面组件日志”开启采集。
   - 采集容器标准输出：开启后，将创建名为default-stdout的日志策略，并上报所有命名空间下的标准输出到云日志服务（LTS）。
   - 采集Kubernetes事件：开启后，将创建名为default-event的日志策略，并上报所有命名空间下的Kubernetes事件到云日志服务（LTS）。
5. 配置完成后，单击右下角“规格确认”，在弹出的窗口中单击“确定”，完成创建。

已有集群中开启

1. 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 选择“控制面组件日志”页签，选择需要采集的控制面组件，单击“一键开启”。

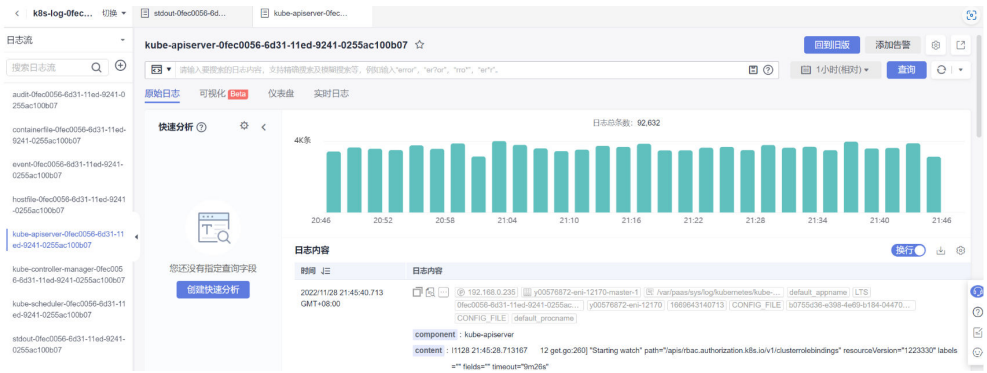
查看集群控制面组件日志

通过控制台查看目标集群控制面组件日志

1. 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 选择“控制面组件日志”页签，在控制面日志中选中需要查看的日志主题，支持的控制面组件日志请参见**集群控制面组件说明**。关于该页面的操作详情，请参见**LTS用户指南**。

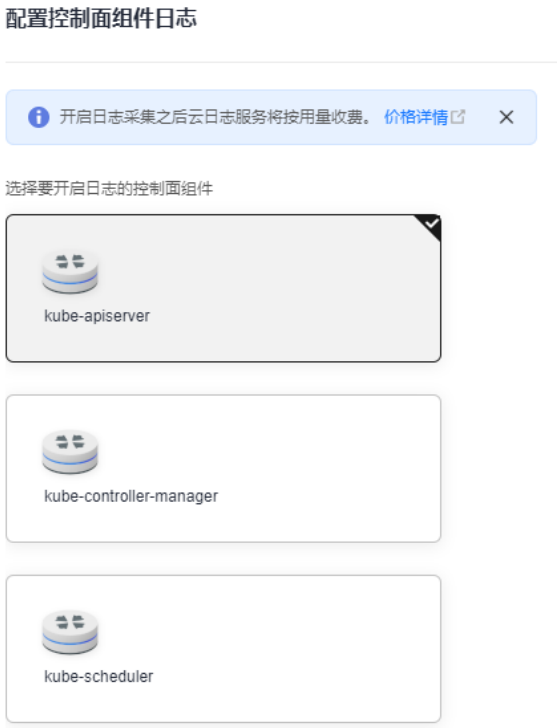
通过LTS控制台查看目标集群控制面组件日志

1. 登录LTS控制台，选择“日志管理”页面。
2. 通过集群ID查到对应的日志组，单击该日志组名称，查看日志流，详情请参见**LTS用户指南**。



关闭华为云集群控制面组件日志

1. 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。
2. 选择“控制面组件日志”页签，单击右上角“配置控制面组件日志”，在“配置控制面组件日志”中修改日志配置。



3. 选择是否开启各个组件日志，并单击“确定”。

说明

关闭集群控制面组件日志后，原有的日志流将不再更新日志，但已有的日志不会被删除，因此可能会产生LTS日志费用。

9.2.5 收集 Kubernetes 审计日志

集群支持对用户开放集群Master节点的日志信息。在日志管理页面可以选择需要上报Kubernetes审计日志到云日志服务（LTS）。

约束与限制

- 华为云集群必须为v1.21.7-r0及以上补丁版本、v1.23.5-r0及以上补丁版本或1.25版本。
- 请确保云日志服务LTS资源配额充足，LTS的默认配额请参见**基础资源**。

集群 Kubernetes 审计日志说明

表 9-16 集群 Kubernetes 审计日志说明

类别	组件	日志流	说明
控制面审计日志	audit	audit-{{clusterID}}	集群审计日志提供了与安全相关的、按时间顺序排列的记录集，记录每个用户、使用Kubernetes API的应用以及控制面自身引发的活动。

开启本地集群控制面审计日志

集群未安装云原生日志采集插件

安装云原生日志采集插件时，可通过勾选控制面审计日志，创建默认日志采集策略，采集对应组件日志上报到LTS。安装方法见：**启用云原生日志采集插件采集日志**。

集群已安装云原生日志采集插件

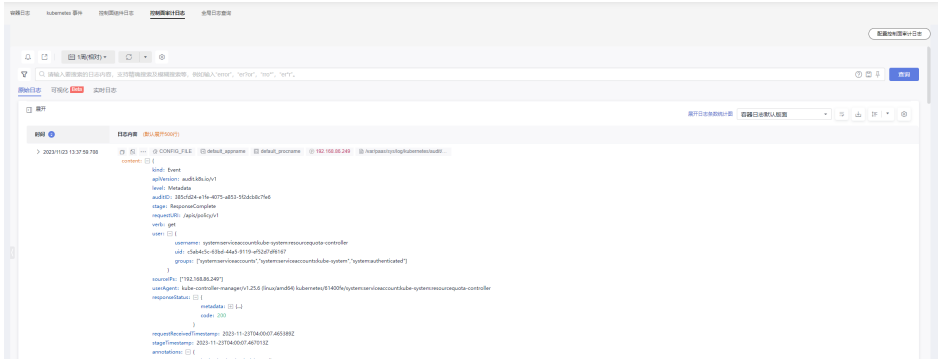
步骤1 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。

步骤2 右上角单击“日志采集策略”，将显示当前集群所有上报LTS的日志策略。

步骤3 单击上方“创建日志策略”，输入要采集的配置信息。

策略模板：若安装插件时未开启控制面审计日志的采集策略，或者删除了对应的日志策略，可通过该方式重新创建控制面审计日志采集策略。

步骤4 日志查看：可直接在“日志中心”页面，“控制面审计日志”页签中查看，选择日志策略配置的日志流名称，即可查看上报到云日志服务（LTS）的日志。



----结束

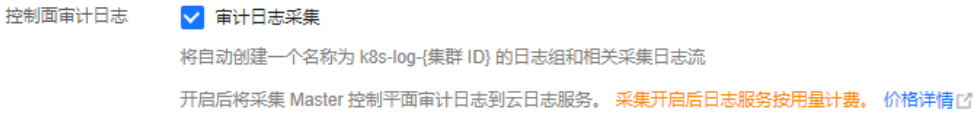
开启华为云集群控制面审计日志

创建集群时开启

步骤1 登录云容器引擎（CCE）控制台。

步骤2 在控制台上方导航栏，选择集群，单击“购买”。

步骤3 在“插件配置”页面中，“控制面审计日志”模块勾选“审计日志采集”。



----结束

已有集群中开启

步骤1 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。

步骤2 选择“控制面审计日志”页签，选择audit组件，单击“一键开启”。

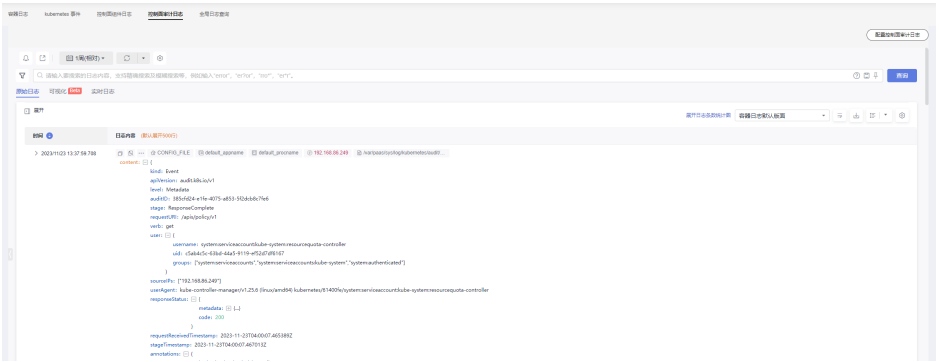
----结束

查看集群控制面审计日志

通过控制台查看目标集群控制面审计日志

步骤1 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。

步骤2 选择“控制面审计日志”页签，在控制面日志中选中需要查看的日志主题。关于该页面的操作详情，请参见**LTS用户指南**。

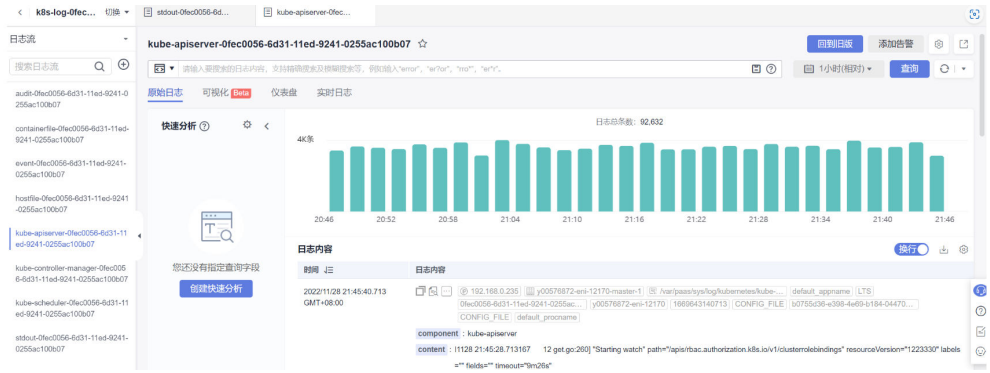


----结束

通过LTS控制台查看目标集群控制面审计日志

步骤1 登录LTS控制台，选择“日志管理”页面。

步骤2 通过集群ID查到对应的日志组，单击该日志组名称，查看日志流，详情请参见**LTS用户指南**。



----结束

关闭华为云集群控制面审计日志

步骤1 登录容器舰队控制台，单击集群名称进入集群，选择左侧导航栏的“日志中心”。

步骤2 选择“审计日志”页签，在“配置审计日志”中修改日志配置。

步骤3 取消勾选audit，并单击“确定”。

说明

关闭集群控制面审计日志后，原有的日志流将不再更新日志，但已有的日志不会被删除，因此可能会产生LTS日志费用。

----结束

9.2.6 收集 Kubernetes 事件

CCE 云原生日志采集插件插件可采集Kubernetes事件上报到云日志服务（LTS）和应用运维管理（AOM），用于保存事件信息和事件告警。

费用说明

LTS创建日志组免费，并每月赠送每个账号一定量免费日志采集额度，超过免费额度部分将产生费用。

Kubernetes 事件上报云日志服务（LTS）

集群未安装云原生日志采集插件

安装云原生日志采集插件时，可通过勾选采集Kubernetes事件，创建默认日志采集策略，采集所有事件上报到LTS。安装方法见：**收集数据面日志**

集群已安装云原生日志采集插件

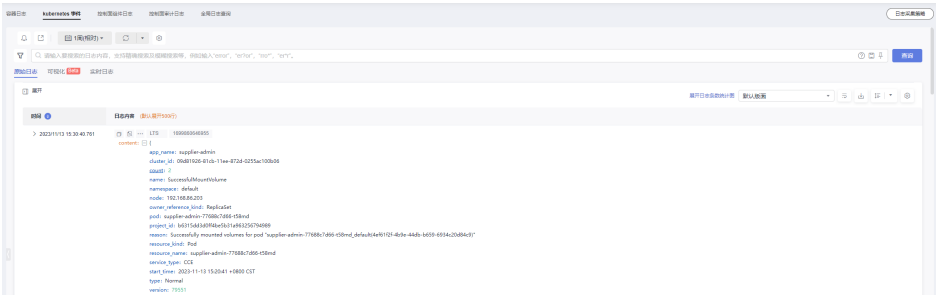
步骤1 登录云容器引擎（CCE）控制台，单击集群名称进入集群，选择左侧导航栏的“日志管理”。

步骤2 右上角单击“日志采集策略”，将显示当前集群所有上报LTS的日志策略。

步骤3 单击上方“创建日志策略”，输入要采集的配置信息。

策略模板：若安装插件时未勾选采集Kubernetes事件，或者删除了对应的日志策略，可通过该方式重新创建默认事件采集策略。

步骤4 事件查看：可直接在“日志管理”页面查看，选择日志策略配置的日志流名称，即可查看上报到云日志服务（LTS）的事件。



----结束

Kubernetes 事件上报应用运维管理（AOM）

当华为云集群版本为1.19.16、1.21.11、1.23.9或1.25.4时，安装CCE 云原生日志采集插件后，默认会将上报所有Warning级别事件以及部分Normal级别事件到应用运维管理（AOM），上报的事件可用于配置告警，安装方法见：**收集数据面日志**

本地集群可在安装插件时开启或关闭该功能。

自定义事件上报

若已上报的事件不能满足需求，可通过修改配置，修改需要上报到应用运维管理（AOM）的事件。

步骤1 在集群上执行以下命令，编辑当前的事件采集配置。

kubectl edit logconfig -n kube-system default-event-aom

步骤2 根据需要修改事件采集配置。

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  annotations:
    helm.sh/resource-policy: keep
  name: default-event-aom
  namespace: kube-system
spec:
  inputDetail:    #采集端配置
    type: event    #采集端类型，请勿修改
    event:
      normalEvents:    #Normal级别事件采集配置
        enable: true    #是否开启Normal级别事件采集
        includeNames:    #需要采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
        includeNames:    #不采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
      warningEvents:    #Warning级别事件采集配置
        enable: true    #是否开启Warning级别事件采集
        includeNames:    #需要采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
        includeNames:    #不采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
  outputDetail:
    type: AOM    #输出端类型，请勿修改
    AOM:
      events:
      - name: DeleteNodeWithNoServer    #事件名，必选
        resourceType: Namespace    #事件对应的资源类型
        severity: Major    #事件上报到AOM后的事件级别，默认Major。可选值：Critical：紧急；Major：重要；
Minor：次要；Info：提示
```

----结束

9.2.7 云原生日志采集插件

本章节主要介绍本地集群云原生日志采集插件相关内容，**开启日志中心**时会自动安装云原生日志采集插件，也可以参考本章节内容手动安装。华为云集群云原生日志采集插件相关内容请参见**云原生日志采集插件**。

插件简介

云原生日志采集插件（log-agent）是基于开源fluent-bit和opentelemetry构建的云原生日志采集插件，支持基于CRD的日志采集策略，可以根据您配置的策略规则，对集群中的容器标准输出日志、容器文件日志、节点日志及K8s事件日志进行采集与转发。安装云原生日志采集插件后，默认采集容器标准输出日志和集群内K8s事件。采集日志的详细使用方法请参见**收集数据面日志**。

约束与限制

云原生日志采集插件有如下限制：

- 仅支持1.21及以上版本集群。
- 每个集群限制50条日志规则。
- 不采集.gz、.tar、.zip后缀类型的日志文件。
- 采集容器文件日志时，若节点存储模式为Device Mapper模式，路径配置必须为节点数据盘挂载路径。
- 若容器运行时为containerd模式，容器标准输出日志中的多行配置暂不生效。
- 每个集群限制单行日志采集速率不超过10000条/秒，多行日志不超过2000条/秒。
- 容器运行时间建议不小于1分钟，防止日志文件删除过快，无法及时采集。

权限说明

云原生日志采集插件中的fluent-bit组件会根据用户的采集配置，读取各节点上容器标准输出、容器内文件日志以及节点日志并采集。

fluent-bit组件运行会使用到以下权限：

- CAP_DAC_OVERRIDE：忽略文件的 DAC 访问限制。
- CAP_FOWNER：忽略文件属主 ID 必须和进程用户 ID 相匹配的限制。
- DAC_READ_SEARCH：忽略文件读及目录搜索的 DAC 访问限制。
- SYS_PTRACE：允许跟踪任何进程。

本地集群安装云原生日志插件前置授权

由于云原生日志插件需要访问LTS和AOM两个云服务，访问云服务需要对云原生日志插件进行鉴权，本地集群云原生日志插件使用工作负载 Identity方式允许集群中的工作负载模拟IAM用户来访问云服务。

工作负载 Identity方式是将集群的公钥配置到IAM身份提供商中，并添加 ServiceAccount 与 IAM 账号映射规则。工作负载部署时挂载ServiceAccount对应的Token，使用此Token访问云服务，IAM 使用该公钥验证Token，从而无需直接使用 IAM 账号的 AK/SK 等信息，降低安全风险。

步骤1 获取本地集群私钥签发的jwks，该公钥用于验证集群签发的 ServiceAccount Token。

1. 使用kubectl连接本地集群。
2. 执行如下命令获取公钥。

   kubectl get --raw /openid/v1/jwks

   返回结果为一个 json 字符串，是当前集群的签名公钥，用于访问身份供应商。

```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "Ew29q....",
      "alg": "RS256",
      "n": "peJdm...."
    }
  ]
}
```

步骤2 在 IAM 配置身份供应商，标志当前集群在 IAM 侧的身份。

1. 登录IAM控制台,查询本地集群所在项目的ID，创建身份供应商，协议选择OpenID Connect。指定插件需要配置指定的身份供应商名称，具体请参见**表9-17**。用户组的权限配置具体操作请参见**用户组策略内容**。

表 9-17 log-agent 身份供应商配置

插件名称	身份提供商名称	客户端ID	namespace	ServiceAccountName	用户组需要开通的最小权限
log-agent	ucs-cluster-identity-{项目ID}	ucs-cluster-identity	monitoring	log-agent-serviceaccount	aom:alarm:* lts:*:*

图 9-29 修改身份提供商信息

2. 单击“确定”，然后修改身份提供商信息，需要修改的信息如表9-18所示。随后创建身份转换规则，单击“创建规则”进行创建。

图 9-30 修改身份提供商信息

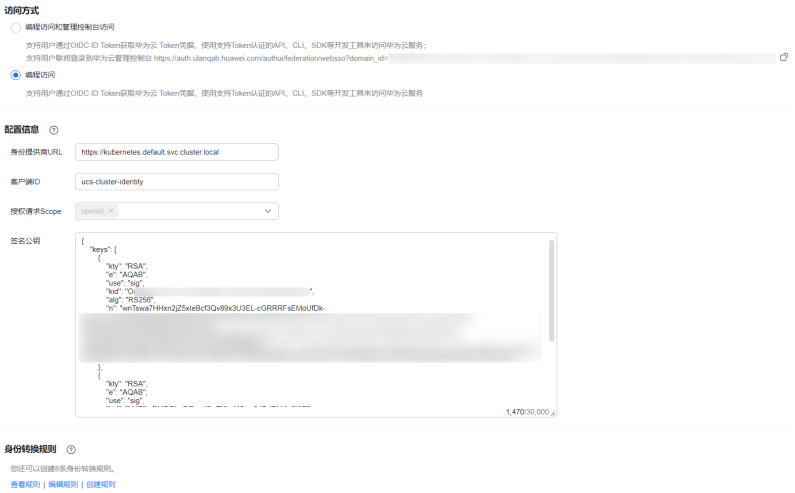
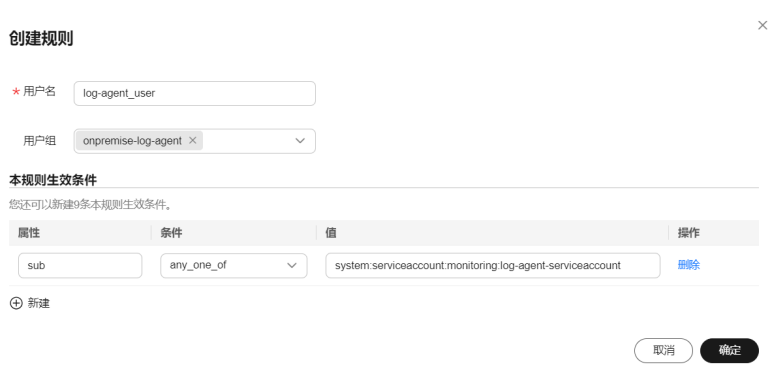


表 9-18 身份提供商配置参数说明

参数	说明
访问方式	选择“编程访问”。
配置信息	- 身份供应商 URL：https://kubernetes.default.svc.cluster.local。 - 客户端 ID：指定插件需要配置指定的客户端ID，请参见表9-17。 - 签名公钥：本地集群的 jwks，获取方法请参见步骤1。涉及多个集群时，请用逗号分隔每个集群的keys数组内容。

参数	说明
身份转换规则	身份转换规则的作用将集群内的ServiceAccount和IAM用户组做映射。 – 属性：sub – 条件：any_one_of – 值： 值的格式为：system:serviceaccount:*Namespace*:*ServiceAccountName* 其中*Namespace*请修改为需要创建ServiceAccount的命名空间，*ServiceAccountName*请修改为需要创建的ServiceAccount名称。 例如：值为system:serviceaccount:monitoring:log-agent-serviceaccount，表明在monitoring命名空间下创建一个名为log-agent的ServiceAccount，并映射到对应用户组，后续使用该 ServiceAccount获取的IAM Token就拥有了对应用户组的权限。 **说明** 本地集群中的相关插件需要配置指定的 ServiceAccountName和用户组权限才能正常工作，请参见 **表9-17**。

图 9-31 创建身份转换规则



3. 单击“确定”。

----结束

本地集群安装云原生日志采集插件

步骤1 登录UCS控制台，选择容器舰队，单击集群名称进入集群，在左侧导航栏中选择“插件中心”，在右侧找到**云原生日志采集插件**，单击“安装”。

步骤2 在安装插件页面，设置“规格配置”。

表 9-19 插件规格配置

参数	参数说明
插件规格	该插件可配置“小规格”、“大规格”或“自定义”规格。
实例数	选择上方插件规格后，显示插件中的实例数。 选择“自定义”规格时，您可根据需求调整插件实例数。
容器	log-agent插件包含以下容器，您可根据需求自定义调整规格： • fluent-bit：日志收集器，以DaemonSet形式安装在每个节点。 • cop-logs：负责采集侧配置文件生成及更新的组件。 • log-operator：负责解析及更新日志规则的组件。 • otel-collector：负责集中式日志转发的组件，将fluent-bit收集的日志转发到LTS。

步骤3 设置插件支持的“参数配置”。

Kubernetes事件上报AOM：采集Kubernetes事件并上报到应用运维管理 AOM，可在AOM配置事件告警规则。

步骤4 设置插件实例日志上报的“网络配置”。

- 公网接入：通过公网Internet接入，要求集群能够访问公网，具有弹性灵活、成本低、易接入的优势。公网接入要求集群能够访问公网，请确保集群已符合此条件，否则会接入失败。
- 云专线/VPN接入：通过云专线（DC）或虚拟专用网络（VPN）服务将云下网络与云上虚拟私有云（VPC）连通，并利用VPC终端节点通过内网与容器智能分析建立连接，具有高速、低时延、安全的优势。详情见**本地集群使用云专线/VPN上报日志**。

步骤5 完成以上配置后，单击“安装”。

----结束

组件说明

表 9-20 log-agent 组件

容器组件	说明	资源类型
fluent-bit	轻量级的日志收集器和转发器，部署在每个节点上采集日志。	DaemonSet
cop-logs	负责生成采集文件的软链接，和fluent-bit运行在同一Pod。	DaemonSet
log-operator	负责生成内部的配置文件。	Deployment
otel-collector	负责收集来自不同应用程序和服务的日志数据，集中后上报至LTS。	Deployment

版本记录

表 9-21 云原生日志采集插件版本记录

插件版本	支持的集群版本	更新特性
1.4.1	v1.21 v1.22 v1.23 v1.24 v1.25 v1.26 v1.27 v1.28	第一次发布，支持本地集群。

自定义事件上报 AOM

log-agent插件会将所有Warning级别事件以及部分Normal级别事件上报到AOM。您也可以根据需求自行设置需要上报的事件，具体方法如下：

1. 在集群上执行以下命令，编辑当前的事件采集配置。

   kubectl edit logconfig -n kube-system default-event-aom

2. 根据需要修改事件采集配置。

```
apiVersion: logging.openvessel.io/v1
kind: LogConfig
metadata:
  annotations:
    helm.sh/resource-policy: keep
  name: default-event-aom
  namespace: kube-system
spec:
  inputDetail:    #采集端配置
    type: event    #采集端类型，请勿修改
    event:
      normalEvents:    #Normal级别事件采集配置
        enable: true    #是否开启Normal级别事件采集
        includeNames:    #需要采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
        includeNames:    #不采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
      warningEvents:    #Warning级别事件采集配置
        enable: true    #是否开启Warning级别事件采集
        includeNames:    #需要采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
        includeNames:    #不采集的事件名，不指定则采集所有事件
        - NotTriggerScaleUp
  outputDetail:
    type: AOM    #输出端类型，请勿修改
    AOM:
      events:
      - name: DeleteNodeWithNoServer    #事件名，必选
        resourceType: Namespace    #事件对应的资源类型
        severity: Major    #事件上报到AOM后的事件级别，默认Major。可选值：Critical：紧急；Major：
重要；Minor：次要；Info：提示
```

log-agent 事件说明

在log-agent插件的安装和运行阶段，log-operator组件会上报事件，您可以根据这些事件来判断安装是否成功，并确定故障原因。具体如表9-22所示：

表 9-22 log-agent 事件说明

事件名称	说明
InitLTSFailed	初始化LTS日志组日志流失败
WatchAKSKFailed	监听AKSK失败
WatchAKSKSuccessful	监听AKSK成功
RequestLTSFailed	请求LTS接口失败
InitLTSSuccessful	初始化LTS日志组日志流成功
CreateWebhookConfigFailed	创建MutatingWebhookConfiguration失败
CreateWebhookConfigSuccessful	创建MutatingWebhookConfiguration成功
StartServerSuccessful	开启监听成功
StartServerFailed	开启监听失败
StartManagerFailed	开启CRD监听失败
InjectAnnotationFailed	annotation注入失败
InjectAnnotationSuccessful	annotation注入成功
UpdateLogConfigFailed	更新logconfig信息失败
GetConfigListFailed	获取CR列表失败
GenerateConfigFailed	生成fluent-bit、otel配置失败

log-agent 指标说明

log-agent插件的log-operator、fluent-bit和otel-collector组件提供了一系列指标，您可以使用AOM或Prometheus来监控这些指标，以便及时了解log-agent插件的运行情况，具体操作可参考**使用AOM监控自定义指标**或**使用Prometheus监控自定义指标**。详细的指标如下所述：

- log-operator组件（仅华为云集群）

  端口：8443；地址：/metrics；协议：https

表 9-23 指标

指标名	说明	类型
log_operator_aksk_latest_update_times	AK/SK最后更新时间	Gauge
log_operator_aksk_update_total	AK/SK更新次数	Counter
log_operator_send_request_total	发送请求数	Counter
log_operator_webhook_listen_status	Webhook监听状态	Gauge
log_operator_http_request_duration_seconds	HTTP请求时延	Histogram
log_operator_http_request_total	HTTP请求数	Counter
log_operator_webhook_request_total	Webhook请求数	Counter

- fluent-bit组件

  端口：2020；地址：/api/v1/metrics/prometheus；协议：http

  表 9-24 指标

指标名	说明	类型
fluentbit_filter_add_records_total	用于记录在过滤器中添加的记录总数	Counter
fluentbit_filter_drop_records_total	用于记录被过滤掉的日志记录数量	Counter
fluentbit_input_bytes_total	用于衡量Fluent Bit在处理日志数据时输入的总字节数	Counter
fluentbit_input_files_closed_total	用于记录关闭的文件总数	Counter
fluentbit_input_files_opened_total	用于监控Fluent Bit的文件输入插件（input plugin）打开的文件数量	Counter
fluentbit_input_files_rotated_total	用于记录Fluent Bit输入插件已经轮转的文件总数	Counter
fluentbit_input_records_total	用于衡量 Fluent Bit 在输入插件中处理的记录数	Counter

指标名	说明	类型
fluentbit_output_dropped_records_total	用于记录输出插件丢弃的记录数量	Counter
fluentbit_output_errors_total	用于监控 Fluent Bit 的输出错误数量	Counter
fluentbit_output_proc_bytes_total	用于监控 Fluent Bit 的输出插件（output plugin）处理的总字节数	Counter
fluentbit_output_proc_records_total	用于监控 Fluent Bit 的输出插件处理的记录数	Counter
fluentbit_output_retried_records_total	用于衡量 Fluent Bit 在输出数据时重试的次数	Counter
fluentbit_output_retries_total	用于衡量 Fluent Bit 在发送数据到输出插件时发生重试的次数	Counter
fluentbit_uptime	Fluent Bit 运行的时间，通常以秒为单位	Counter
fluentbit_build_info	用于显示Fluent Bit的版本和构建信息	Gauge

- otel-collector组件
  端口：8888；地址：/metrics；协议：http

  表 9-25 指标

指标名	说明	类型
otelcol_exporter_enqueue_failed_log_records	用于衡量OpenTelemetry Collector在将日志记录发送到下游系统时，由于某些原因无法成功发送的日志记录数量	Counter
otelcol_exporter_enqueue_failed_metric_points	用于衡量在将指标数据发送到后端时，由于某些原因导致无法成功发送的指标数据点的数量	Counter
otelcol_exporter_enqueue_failed_spans	用于衡量otelcol exporter在将span发送到后端时失败的次数	Counter
otelcol_exporter_send_failed_log_records	用于衡量日志记录发送失败的数量	Counter

指标名	说明	类型
otelcol_exporter_sent_log_records	用于衡量OpenTelemetry Collector（otelcol）发送的日志记录数量	Counter
otelcol_process_cpu_seconds	用于度量进程CPU使用时间的指标，它表示进程在特定时间段内使用的CPU时间，单位为秒	Counter
otelcol_process_memory_rss	是OpenTelemetry中用于监控进程内存使用情况的一个指标。其中，rss代表Resident Set Size，即进程当前占用的物理内存大小	Gauge
otelcol_process_runtime_heap_alloc_bytes	用于监控进程运行时堆内存分配的指标。它表示进程在运行时分配的堆内存的总字节数。	Gauge
otelcol_process_runtime_total_alloc_bytes	用于衡量进程在运行时分配的总字节数	Counter
otelcol_process_runtime_total_sys_memory_bytes	用于衡量进程在运行时使用的系统内存总量，单位为字节。	Gauge
otelcol_process_uptime	指OpenTelemetry收集器进程的运行时间，以秒为单位。	Counter
otelcol_receiver_accepted_log_records	用于衡量OpenTelemetry收集器接收并成功处理的日志记录数量	Counter
otelcol_receiver_refused_log_records	用于衡量接收器（receiver）拒绝接收的日志记录数量	Counter

9.2.8 本地集群使用云专线/VPN 上报日志

通过云专线（DC）或虚拟专用网络（VPN）服务将云下网络与云上虚拟私有云（VPC）连通，并利用VPC终端节点通过内网与容器智能分析建立连接，具有高速、低时延、安全的优势。

步骤一：云日志服务 VPC 终端节点授权

步骤1 在导航栏单击“工单>新建工单”。

步骤2 在“我遇到的问题所属产品/服务”的输入框中输入LTS，单击“搜索”

步骤3 问题类型选择“其他问题”，新建工单。

步骤4 输入问题描述，选择联系方式，并提交。

说明

问题描述内容建议：云日志服务VPC终端节点授权，{账号ID}。如：云日志服务VPC终端节点权限开通。

----结束

步骤二：云专线/VPN 接入

步骤1 提交工单开通云日志服务（LTS）的VPC终端节点。具体步骤请参见**步骤一：云日志服务VPC终端节点授权**。

步骤2 在本地集群详情页，编辑云原生日志采集插件配置。

- 未安装 云原生日志采集（log-agent）插件时，可在日志中心单击“立即开启”。
- 已安装 云原生日志采集（log-agent）插件时，可在插件中心编辑对应插件。

步骤3 选择对应的VPC终端节点。若不存在可用的VPC终端节点，单击“创建终端节点”以创建VPC终端节点。再次提交工单云日志服务（LTS）的VPC终端节点需要经过LTS服务审批，操作方法请参见**步骤一：云日志服务VPC终端节点授权**。

须知

创建的VPC终端节点需要和本地集群节点在同一个虚拟私有云或建立对等连接。

步骤4 单击“确定”，完成云专线/VPN接入。

----结束

10 容器迁移

10.1 容器迁移概述

华为云UCS的容器迁移服务为您提供了一种可靠、安全、灵活且高效的迁移方案。通过使用UCS，您可以将本地数据中心或其他云提供商上的Kubernetes集群中的云原生应用迁移到华为云UCS管理的Kubernetes集群中。这样，您可以实现统一的运维管理，降低管理成本并提高运维效率。

将应用从一个环境迁移到另一个环境是一项具有挑战性的任务，因此仔细的规划和准备至关重要。UCS的容器迁移服务通过以下四个阶段为您提供全流程迁移指导：

1. 集群评估：在此阶段，您需要评估源集群的状态以决定目标集群的类型。
2. 数据迁移：在这个阶段，您将迁移镜像以及依赖服务的相关数据。
3. 应用备份：在此阶段，您需要备份源集群中的应用。
4. 应用迁移：在这个阶段，您将通过备份数据恢复的方式，将源集群中的应用迁移到目标集群。

通过遵循UCS容器迁移服务的全流程迁移指导，您可以更顺利地将应用从一个环境迁移到另一个环境。

方案优势

UCS容器迁移的主要优势包括：

- **易用性**

  在集群评估、镜像迁移、应用备份和应用迁移阶段，迁移过程已实现工具化。这些工具免安装，简单轻量且配置灵活。
- **多版本**

  支持Kubernetes 1.19以上版本的集群迁移。
- **无依赖**

  迁移工具不需要任何外部依赖，可以独立运行。
- **多架构**

  迁移工具支持在Linux（x86、arm）、Windows环境中运行。
- **多场景**

支持多种场景的集群迁移，满足您不同迁移需求，具体如表10-1所示。

表 10-1 迁移场景

场景	说明
本地IDC集群迁移上云	将位于本地数据中心的Kubernetes集群迁移到华为云UCS管理的Kubernetes集群，实现应用程序的云端部署和运维管理。
第三方云集群跨云迁移	将位于其他云服务提供商的Kubernetes集群迁移到华为云UCS管理的Kubernetes集群，实现跨云迁移和统一管理。
不同Region UCS华为云集群迁移	在华为云UCS管理的Kubernetes集群间进行迁移，将应用程序从一个地理区域迁移到另一个地理区域，以满足数据合规性、延迟和可用性等需求。
同Region UCS华为云集群迁移	在同一地理区域内的华为云UCS管理的Kubernetes集群间进行迁移，实现资源优化、应用升级或其他管理需求。

- **不停机**
  迁移过程无需停机，不影响源集群业务。

10.2 容器迁移准备工作

硬件资源

在开始迁移之前，请确保您已准备了一台安装了kubectl的服务器，用于连接源集群和目标集群。该服务器需要至少拥有5GB左右的本地磁盘空间和≥8G的内存，以确保迁移工具可以正常运行，并存储相关数据，如源集群的采集数据和目标集群的推荐数据等。

迁移工具支持在Linux（x86、arm）、Windows环境中运行，因此您可以在这些操作系统中任选一种作为服务器的操作系统。

工具包

在集群评估、镜像迁移、应用备份和应用迁移阶段，迁移过程已实现工具化。您需要预先下载这些工具并将它们上传到前述服务器。

说明

对于Linux操作系统来说，使用下述工具前，需要运行**chmod u+x 工具名**命令（例如**chmod u+x kspider-linux-amd64**），授予可执行权限。

表 10-2 准备工作

工具	说明	下载链接	备注
kspider	kspider是一款用于采集源集群信息的工具，它向用户提供了集群的Kubernetes版本、规模、工作负载数量、存储以及正在使用的镜像等数据，这些信息有助于用户了解集群的当前状况，评估迁移风险，并选择合适的目标集群版本和规模。	Linux x86： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/kspider-linux-amd64** Linux arm： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/kspider-linux-arm64** Windows： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/kspider-windows-amd64.exe**	这些工具均支持在Linux（x86、arm）和Windows系统上运行，工具包解压后会包含两个二进制文件和一个应用程序，分别适用于Linux和Windows环境。 kspider工具包含： • kspider-linux-amd64 • kspider-linux-arm64 • kspider-windows-amd64.exe image-migrator工具包含： • image-migrator-linux-amd64 • image-migrator-linux-arm64 • image-migrator-windows-amd64.exe k8clone工具包含： • k8clone-linux-amd64 • k8clone-linux-arm64 • k8clone-windows-amd64.exe

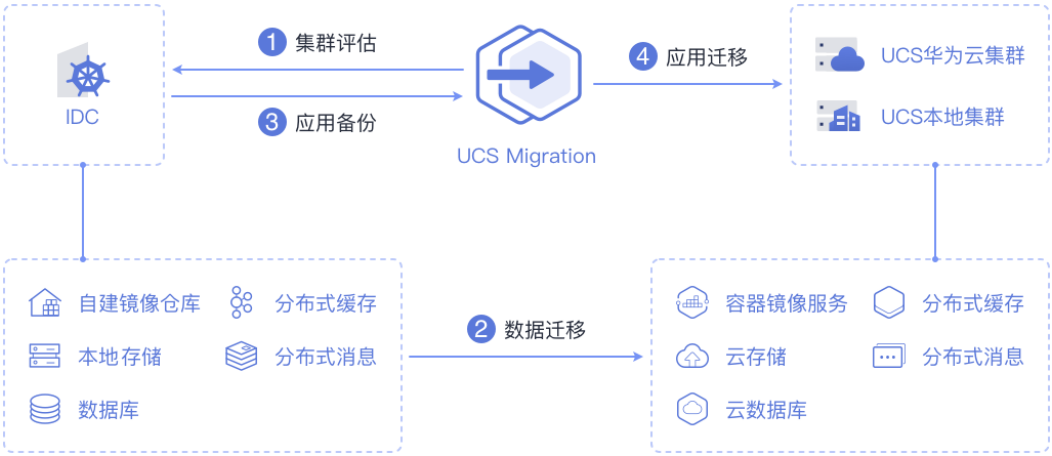
工具	说明	下载链接	备注
image-migrator	image-migrator是一个镜像迁移工具，能够自动将基于Docker Registry v2搭建的Docker镜像仓库或第三方云镜像仓库中的镜像迁移到SWR中。	Linux x86： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/image-migrator-linux-amd64** Linux arm： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/image-migrator-linux-arm64** Windows： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/image-migrator-windows-amd64.exe**	
k8clone	k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群中。	Linux x86： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/k8clone-linux-amd64** Linux arm： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/k8clone-linux-arm64** Windows： **https://ucs-migration-intl.obs.ap-southeast-3.myhuaweicloud.com/toolkits/k8clone-windows-amd64.exe**	

10.3 本地 IDC 集群迁移上云

10.3.1 本地 IDC 集群迁移上云流程

UCS容器迁移支持将本地IDC自建的Kubernetes集群应用迁移到UCS华为云集群或本地集群，实现应用程序的云端部署和运维管理。迁移流程如图10-1所示。

图 10-1 迁移流程



主要包含四个步骤：

步骤1 集群评估

在这个阶段，您将根据源集群的现状来评估适合迁移的目标集群类型。UCS的kspider工具能够自动收集源集群的信息，包括Kubernetes版本、规模、工作负载、存储等数据，并根据收集到的数据为您提供推荐的目标集群信息。具体请参见**集群评估**。

步骤2 数据迁移

在这个阶段，您将把镜像和相关依赖服务的数据迁移到云端。UCS的image-migrator是一个自动化镜像迁移工具，可以将基于Docker Registry v2搭建的Docker镜像仓库中的镜像迁移到SWR中；对于依赖服务的数据迁移，您可以使用华为云的其他云产品来配套实现。具体请参见**镜像迁移**和**依赖服务迁移**。

步骤3 应用备份

在这个阶段，您将对本地IDC集群中的应用进行备份。UCS的k8clone工具可以自动收集Kubernetes元数据，并将其以压缩包的形式保存到本地，从而实现集群中应用的备份。具体请参见**应用备份**。

步骤4 应用迁移

在这个阶段，您将利用备份数据恢复的方法，将本地IDC集群中的应用迁移到UCS华为云集群或本地集群。具体请参见**应用迁移**。

----结束

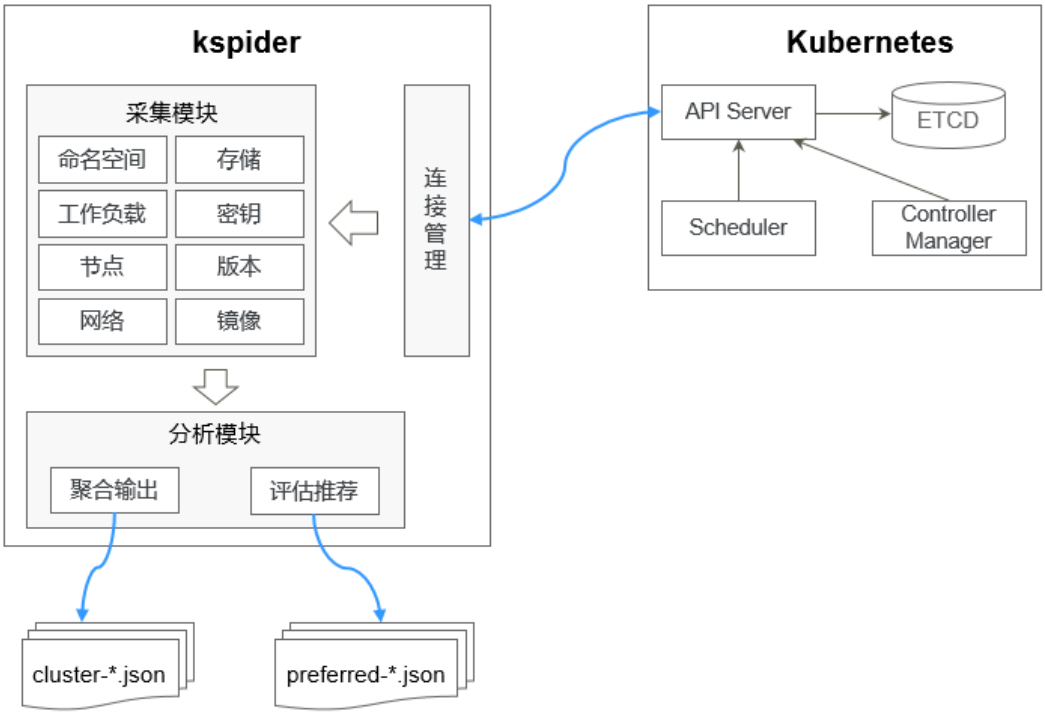
10.3.2 集群评估

将应用从一个环境迁移到另一个环境是一项具有挑战性的任务，因此您需要进行仔细的规划和准备。kspider是一款用于采集源集群信息的工具，它向用户提供了集群的Kubernetes版本、规模、工作负载数量、存储以及正在使用的镜像等数据，这些信息有助于用户了解集群的当前状况，评估迁移风险，并选择合适的目标集群版本和规模。

kspider 工作原理

kspider工具的架构如图10-2所示，包含三个模块：采集模块、连接管理和分析模块。采集模块可以收集源集群的数据，包括命名空间、工作负载、节点、网络等；连接管理模块负责与源集群的API Server建立连接；分析模块分为聚合输出和评估推荐两部分，旨在输出源集群的采集数据（生成“cluster-*.json”文件）以及提供目标集群的推荐信息（生成“preferred-*.json”文件）。

图 10-2 kspider 架构



kspider 使用方法

说明

kspider工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**kspider-linux-amd64**分别替换为**kspider-linux-arm64**或**kspider-windows-amd64.exe**。

根据**容器迁移准备工作**章节的要求，准备一台服务器并上传kspider工具，然后进行解压缩。在kspider工具所在目录下执行**./kspider-linux-amd64 -h**，您可以查看该工具的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件

中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。

- -n, --namespaces：指定采集的命名空间，默认排除了kube-system、kube-public、kube-node-lease等系统命名空间。
- -q, --quiet：静态退出。
- -s, --serial：根据采集信息输出汇聚文件（cluster-{serial}.json）和推荐文件（preferred-{serial}.json）唯一标识的序号。

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]

Aliases:
  kspider, kspider

Flags:
  -h, --help               help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet              command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

步骤一：源集群数据采集

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 使用默认参数配置，采集集群中所有命名空间的数据。执行方法：**./kspider-linux-amd64**

执行后的输出详细信息如下：

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
   Name     CPU     Memory     IP     Arch     OS     Kernel     MachineID
   10.1.18.64     4     8008284Ki     [10.1.18.64 10.1.18.64]     amd64     linux
3.10.0-1127.19.1.el7.x86_64     ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
   10.1.19.13     4     8008284Ki     [10.1.19.13 10.1.19.13]     amd64     linux
3.10.0-1127.19.1.el7.x86_64     2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
   Name     Namespace     NodeAffinity
   minio     default     false
   minio     minio     false
There are 3 Deployments:
   Name     Namespace     NodeAffinity
   rctest     default     true
   flink-operator-controller-manager     flink-operator-system     false
   rctest     minio     false
There are 1 DaemonSets:
   Name     Namespace     NodeAffinity
   ds-nginx     minio     false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
   Namespace/Name     Pods
   default/pvc-data-minio-0     default/minio-0
   minio/obs-testing     minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
   minio/pvc-data-minio-0     minio/minio-0
```

```
There are 5 PersistentVolumes:
  Name  Namespace  pvcName  scName  size  key
  pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8  minio-pv  obs-testing  csi-obs  1Gi  pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
  pvc-c7c768aa-373a-4c52-abea-e8b486d23b47  minio-pv  pvc-data-minio-0  csi-disk-sata  10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
  pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05  minio  obs-testing  csi-obs  1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
  pvc-9fd92c99-805a-4e65-9f22-e238130983c8  default  pvc-data-minio-0  csi-disk  10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
  pvc-a22fd877-f98d-4c3d-a04e-191d79883f97  minio  pvc-data-minio-0  csi-disk-sata  10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
  Name  Namespace  ServiceType
  headless-lxprus  default  ClusterIP
  kubernetes  default  ClusterIP
  minio  default  NodePort
  flink-operator-controller-manager-metrics-service  flink-operator-system  ClusterIP
  flink-operator-webhook-service  flink-operator-system  ClusterIP
  headless-lxprus  minio  ClusterIP
  minio  minio  NodePort
There are 0 Ingresses:
There are 6 Images:
  Name
  gcr.io/flink-operator/flink-operator:v1beta1-6
  flink:1.8.2
  swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
  nginx:stable-alpine-perl
  swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
  gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
  SecretType
  cfe/secure-opaque
  helm.sh/release.v1
```

在kspider执行完毕后，当前目录下将生成两个文件：

- cluster-*.json：此文件包含了源集群及应用的采集数据，这些数据可用于分析和规划迁移过程。
- preferred-*.json：此文件包含了推荐的目标集群信息。基于源集群的规模和节点规格进行初步评估，文件将提供关于目标集群版本和规模的建议。

步骤3 查看源集群及应用的采集数据。

您可以用文本编辑器或JSON查看器打开“cluster-*.json”文件以查看数据。在实际操作中，您需要将文件名中的“*”替换为实际的时间戳或序列号，以找到并打开正确的文件。

“cluster-*.json”文件说明如下：

```
{
 K8sVersion：Kubernetes版本，字符串类型
 Namespaces：命名空间数量，字符串类型
 Pods：Pod总数量，整型
 Nodes：节点总信息，以IP为key，展示节点信息
  IP地址
   CPU：CPU，字符串类型
   Arch：CPU架构，字符串类型
   Memory：内存，字符串类型
   HugePages1Gi：1G大页内存，字符串类型
   HugePages2Mi：2M大页内存，字符串类型
   OS：节点OS，字符串类型
   KernelVersion：OS内核版本，字符串类型
   RuntimeVersion：节点容器运行及版本，字符串类型
   InternalIP：内部IP，字符串类型
   ExternalIP：外部IP，字符串类型
   MachineID：节点ID，字符串类型。说明：CCE中能够保证与ECS的ID一致
```



```
 Workloads：工作负载
   Deployment：工作负载类型，支持Deployment（无状态负载）、StatefulSet（有状态负载）、DaemonSet（守护进程集）、CronJob（定时任务）、Job（普通任务）、LonePod（独立Pod）
     default：命名空间名称
       Count：数量，整型
       Items：详细信息，数组类型
         Name：工作负载名称，字符串类型
         Namespace：命名空间名称，字符串类型
         NodeAffinity：节点亲和性，布尔型
         Replicas：副本数量，整型
 Storage：存储
   PersistentVolumes：持久卷
     pv-name：以PV名称为key
       VolumeID：卷ID，字符串类型
       Namespace：命名空间，字符串类型
       PvcName：绑定PVC的名称，字符串类型
       ScName：存储类的名称，字符串类型
       Size：申请空间大小，字符串类型
       Pods：使用PV的Pod名称，字符串类型
       NodeIP：Pod所在的节点IP，字符串类型
       VolumePath：该Pod挂载节点的路径，字符串类型
   OtherVolumes：其它类型卷
     类型：AzureFile、AzureDisk、GCEPersistentDisk、AWSElasticBlockStore、Cinder、Glusterfs、NFS、CephFS、FlexVolume、FlexVolume、DownwardAPI
       卷ID/卷名称/卷共享路径等为key
         Pods：使用其的Pod，字符串类型
         NodeIP：Pod所在的节点IP，字符串类型
         卷ID/卷名称/卷共享路径等唯一标识卷信息的信息，字符串类型
 Networks：网络
   LoadBalancer：负载均衡类型
     service：网络类型，包括service和ingress
       Name：名称，字符串类型
       Namespace：命名空间名称，字符串类型
       Type：类型，字符串类型
 ExtraSecrets：扩展secret类型
   secret类型名，字符串类型
 Images：镜像
   镜像repo，字符串类型
}
```

示例：

```
{
  "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
  "Namespaces": 12,
  "Pods": 33,
  "Nodes": {
    "10.1.17.219": {
      "CPU": "4",
      "Memory": "7622944Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP9x86_64)",
      "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.17.219",
      "ExternalIP": "",
      "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
    },
    "10.1.18.182": {
      "CPU": "4",
      "Memory": "7992628Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP5)",
      "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.18.182",
```



```
    "ExternalIP": "100.85.xxx.xxx",
    "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
  }
},
"Workloads": {
  "CronJob": {},
  "DaemonSet": {
    "default": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-prometheus-node-exporter",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 3
        }
      ]
    }
  },
  "Deployment": {
    "default": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-cost-analyzer",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    },
    "kubecost": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-kube-state-metrics",
          "Namespace": "kubecost",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  },
  "Job": {},
  "LonePod": {},
  "StatefulSet": {
    "minio-all": {
      "Count": 1,
      "Items": [
        {
          "Name": "minio",
          "Namespace": "minio-all",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  }
},
"Storage": {
  "PersistentVolumes": {
    "demo": {
      "VolumeID": "demo",
      "Namespace": "fluid-demo-test",
      "PvcName": "demo",
      "ScName": "fluid",
      "Size": "100Gi",
      "Pods": "",
      "NodeIP": "",
```



```
      "VolumePath": ""
     },
     "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
      "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
      "Namespace": "minio-all",
      "PvcName": "pvc-data-minio-0",
      "ScName": "csi-disk",
      "Size": "10Gi",
      "Pods": "minio-all/minio-0",
      "NodeIP": "10.1.23.159",
      "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
     }
    },
    "OtherVolumes": {}
  },
  "Networks": {
    "LoadBalancer": {}
  },
  "ExtraSecrets": [
    "cfe/secure-opaque",
    "helm.sh/release.v1"
  ],
  "Images": [
    "nginx:stable-alpine-perl",
    "ghcr.io/koordinator-sh/koord-manager:0.6.2",
    "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
    "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
    "gcr.io/kubecost1/cost-model:prod-1.91.0",
    "gcr.io/kubecost1/frontend:prod-1.91.0"
  ]
}
```


----结束

步骤二：目标集群评估

在kspider执行完毕后，除了“cluster-*.json”文件之外，还会在当前目录下生成“preferred-*.json”文件。这个文件基于源集群的规模和节点规格进行初步评估，并提供关于目标集群版本和规模的推荐信息。这有助于您更好地规划和准备迁移过程。

“preferred-*.json”文件说明如下：

```
{
  K8sVersion：Kubernetes版本，字符串类型
  Scale：集群规模，字符串类型
  Nodes：节点信息
    CPU：CPU，字符串类型
    Memory：内存，字符串类型
    Arch：架构，字符串类型
    KernelVersion：OS内核版本，字符串类型
    ProxyMode：集群Proxy模式，字符串类型
  ELB：是否依赖ELB，布尔型
}
```

上述文件中每个字段的评估规则如下：

表 10-3 评估规则

字段	评估规则
Kubernetes版本	如果是1.21以下版本，推荐UCS集群主要发行版本（例如1.21，随着时间发展会发生变化），大于主要发行版本时，将推荐UCS集群的最新版本。

字段	评估规则
集群规模	源集群节点数＜25，推荐50节点规模 25≤源集群节点数＜100，推荐200节点规模 100≤源集群节点数＜500，推荐1000节点规模 源集群节点数≥500，推荐2000节点规模
CPU+内存	统计数量最多的那一种规格
架构	统计数量最多的那一种规格
OS内核版本	统计数量最多的那一种规格
集群Proxy模式	根据集群规模来设置，大于1000节点规模的集群，推荐使用ipvs，1000以内的推荐使用iptables。
是否依赖ELB	源集群是否有负载均衡类型的Service

示例：

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

⚠ 注意

评估结果仅供参考，最终选择什么版本、规模的目标集群还需要您综合判断。

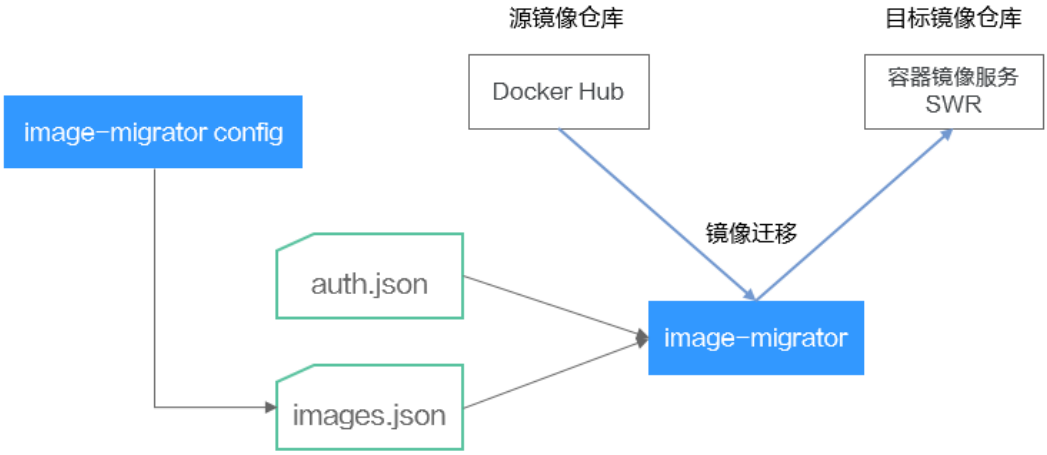
10.3.3 镜像迁移

为保证集群迁移后容器镜像可正常拉取，提升容器部署效率，建议您将自建镜像仓库迁移至华为云容器镜像服务（SWR）。UCS华为云集群和本地集群配合SWR为您提供容器自动化交付流水线，采用并行传输的镜像拉取方式，能够大幅提升容器的交付效率。

image-migrator是一个镜像迁移工具，能够自动将基于Docker Registry v2搭建的Docker镜像仓库中的镜像迁移到SWR中。

image-migrator 工作原理

图 10-3 image-migrator 工作原理



使用image-migrator工具将镜像迁移到SWR时，需要准备两个文件，一个为镜像仓库访问权限文件“auth.json”，该文件中的两个对象分别为源镜像仓库和目标镜像仓库（即Registry）的账号和密码；另一个为镜像列表文件“images.json”，文件内容由多条镜像同步规则组成，每一条规则包括一个源镜像仓库（键）和一个目标镜像仓库（值）。将这两个文件准备好以后，放在image-migrator工具所在目录下，执行一条简单的命令，就可以完成镜像的迁移。关于两个文件的详细介绍如下：

- **“auth.json”文件**

  “auth.json”为镜像仓库访问权限文件，其中每个对象为一个registry的用户名和密码。通常源镜像仓库需要具有pull以及访问tags权限，目标镜像仓库需要拥有push以及创建仓库权限。如果是匿名访问镜像仓库，则不需要填写用户名、密码等信息。“auth.json”文件的结构如下：

  ```
  {
    "源镜像仓库地址": { },
    "目标镜像仓库地址": {
       "username": "xxxxxx",
       "password": "***************",
       "insecure": true
    }
  }
  ```

  其中，

  - “源镜像仓库地址”和“目标镜像仓库地址”支持“registry”和“registry/namespace”两种形式，需要跟下述“images.json”中的registry或registry/namespace对应。images中被匹配到的url会使用对应用户名密码进行镜像同步，优先匹配“registry/namespace”的形式。

    目标镜像仓库地址如果为“registry”形式，可以从SWR控制台页面获取，具体方法如下：在“总览”页面单击右上角“登录指令”，登录指令末尾的域名即为SWR镜像仓库地址，例如swr.cn-north-4.myhuaweicloud.com。注意每个Region的地址不同，请切换到对应Region获取。如果为“registry/namespace”形式，还要将namespace替换为SWR的组织名称。
  - username：（可选）用户名，values可以填写具体取值，也可以使用“${env}”或者“$env”类型的字符串引用环境变量。
  - password：（可选）密码，values可以填写具体取值，也可以使用“${env}”或者“$env”类型的字符串引用环境变量。

- insecure：（可选）registry是否为http服务，如果是，insecure为true；默认是false。

说明

目标镜像仓库SWR的用户名形式为：区域项目名称@AK；密码为AK和SK经过加密处理后的登录密钥，详细指导请参考**获取长期有效登录指令**。

示例：

```
{
    "quay.io/coreos": { },
    "swr.cn-north-4.myhuaweicloud.com": {
        "username": "cn-north-4@RVHVMX******",
        "password": "***************",
        "insecure": true
    }
}
```

- **“images.json”文件**

该文件本质上是一个待迁移的镜像清单，由多条镜像同步规则组成，每一条规则包括一个源镜像仓库（键）和一个目标镜像仓库（值）。具体的要求如下：

a. 同步的最大单位是仓库（repository），不支持通过一条规则同步整个namespace以及registry。

b. 源仓库和目标仓库的格式与docker pull/push命令使用的镜像url类似（registry/namespace/repository:tag）。

c. 源仓库和目标仓库（如果目标仓库不为空字符串）都至少包含registry/namespace/repository。

d. 源仓库字段不能为空，如果需要将一个源仓库同步到多个目标仓库需要配置多条规则。

e. 目标仓库名可以和源仓库名不同，此时同步功能类似于：docker pull + docker tag + docker push。

f. 当源仓库字段中不包含tag时，表示将该仓库所有tag同步到目标仓库，此时目标仓库不能包含tag。

g. 当源仓库字段中包含tag时，表示只同步源仓库中的一个tag到目标仓库，如果目标仓库中不包含tag，则默认使用源tag。

h. 当目标仓库为空字符串时，会将源镜像同步到默认registry的默认namespace下，并且repository以及tag与源仓库相同，默认registry和默认namespace可以通过命令行参数以及环境变量配置。

示例如下：

```
{
    "quay.io/coreos/etcd:1.0.0": "swr.cn-north-4.myhuaweicloud.com/test/etcd:1.0.0",
    "quay.io/coreos/etcd": "swr.cn-north-4.myhuaweicloud.com/test/etcd",
    "quay.io/coreos/etcd:2.7.3": "swr.cn-north-4.myhuaweicloud.com/test/etcd"
}
```

使用image-migrator工具的config子命令可自动获取集群中工作负载正在使用的镜像，具体用法请参见**image-migrator config使用方法**。得到images.json文件后，您还可以根据需要进行修改、添加或删除。

image-migrator 使用方法

说明

image-migrator工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**image-migrator-linux-amd64**分别替换为**image-migrator-linux-arm64**或**image-migrator-windows-amd64.exe**。

在image-migrator工具所在目录下执行**./image-migrator-linux-amd64 -h**，可以查看image-migrator工具的使用方法。

- --auth：指定auth.json的路径，默认在image-migrator所在目录下。
- --images：指定images.json的路径，默认在image-migrator所在目录下。
- --log：指定image-migrator生成日志的路径，默认是image-migrator当前目录下的image-migrator.log。
- --namespace：默认的目标仓库的namespace，也就是说，如果images.json中没有指定目标仓库中的namespace，可以在执行迁移命令时指定。
- --registry：默认的目标仓库的registry，也就是说，如果images.json中没有指定目标仓库中的registry，可以在执行迁移命令时指定。
- --retries：迁移失败时的重试次数，默认为3。
- --workers：镜像搬迁的worker数量（并发数），默认是7。

```
$ ./image-migrator-linux-amd64 -h
A Fast and Flexible docker registry image images tool implement by Go.

Usage:
  image-migrator [flags]

Aliases:
  image-migrator, image-migrator

Flags:
      --auth string        auth file path. This flag need to be pair used with --images. (default "./auth.json")
  -h, --help               help for image-migrator
      --images string      images file path. This flag need to be pair used with --auth (default "./images.json")
      --log string         log file path (default "./image-migrator.log")
      --namespace string   default target namespace when target namespace is not given in the images config file, can also be set with DEFAULT_NAMESPACE environment value
      --registry string    default target registry url when target registry is not given in the images config file, can also be set with DEFAULT_REGISTRY environment value
  -r, --retries int        times to retry failed tasks (default 3)
  -w, --workers int        numbers of working goroutines (default 7)

$./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test \
--registry=swr.cn-north-4.myhuaweicloud.com --retries=2
$ ./image-migrator
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

示例如下：

./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

该命令表示将“images.json”文件中的镜像迁移至“swr.cn-north-4.myhuaweicloud.com/test”镜像仓库下，迁移失败时可以重试2次，一次可以同时搬迁5个镜像。

image-migrator config 使用方法

image-migrator工具的config子命令可用于获取集群应用中使用的镜像，在工具所在目录下生成images.json。执行**./image-migrator-linux-amd64 config -h**命令可以查看config子命令的使用方法。

- -k, --kubeconfig：指定kubectl的kubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -n, --namespaces：指定获取镜像的命名空间，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示获取所有命名空间的镜像。
- -t, --repo：指定目标仓库的地址（registry/namespace）。

```
$ ./image-migrator-linux-amd64 config -h
generate images.json

Usage:
   image-migrator config [flags]

Flags:
   -h, --help               help for config
   -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default "/root/.kube/config")
   -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
   -t, --repo string         target repo,such as swr.cn-north-4.myhuaweicloud.com/test
```

示例如下：

- 指定一个命名空间

  ./image-migrator-linux-amd64 config -n default -t swr.cn-north-4.myhuaweicloud.com/test
- 指定多个命名空间

  ./image-migrator-linux-amd64 config -n default,kube-system -t swr.cn-north-4.myhuaweicloud.com/test
- 不指定命名空间（表示获取所有命名空间的镜像）

  ./image-migrator-linux-amd64 config -t swr.cn-north-4.myhuaweicloud.com/test

镜像迁移操作步骤

步骤1 准备镜像仓库访问权限文件：auth.json。

新建一个auth.json文件，并按照格式修改，如果是匿名访问仓库，则不需要填写用户名、密码等信息。将文件放置在image-migrator所在目录下。

示例：

```
{
   "quay.io/coreos": { },
   "swr.cn-north-4.myhuaweicloud.com": {
      "username": "cn-north-4@RVHVMX******",
      "password": "***************",
      "insecure": true
   }
}
```

详细的参数说明请参见**“auth.json”文件**。

步骤2 准备镜像列表文件：images.json。

1. 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。
2. 执行镜像迁移config子命令，生成images.json文件。

   您可以参考**image-migrator config使用方法**中的方法和示例，不指定命名空间，或者指定一个、多个命名空间来获取源集群应用中使用的镜像。
3. 根据需求调整images.json文件内容，但要遵循**“images.json”文件**中所讲的八项要求。

步骤3 镜像迁移。

您可以执行默认的**./image-migrator-linux-amd64**命令进行镜像迁移，也可以根据需要设置image-migrator的参数。

例如以下命令：

./image-migrator-linux-amd64 --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

示例：

```
$ ./image-migrator-linux-amd64
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

步骤4 结果查看。

上述命令执行完毕后，回显如下类似信息：

```
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

表示按照配置，成功将38个镜像迁移到SWR仓库中。

----结束

10.3.4 依赖服务迁移

本节介绍集群依赖服务的相关数据迁移，如本地存储、数据库、分布式缓存、分布式消息等。若您的集群不涉及这些数据，或者这些数据不需要搬迁上云，可忽略本节内容。

存储迁移

- 若您的集群使用本地存储，迁移上云可以使用华为云**数据快递服务 DES**。DES服务是一种海量数据传输解决方案，支持TB到几百TB级数据上云，通过Teleport设备或硬盘（外置USB接口、SATA接口、SAS接口）向华为云传输大量数据，致力于解决海量数据传输网络成本高、传输时间长等难题。
- 若您的集群对接了对象存储，且需同步搬迁至云上，可以使用华为云**对象存储迁移服务 OMS**，帮助您将对象存储中的数据在线迁移至华为云对象存储服务OBS。
- 若您的集群使用文件存储，迁移上云可以使用华为云弹性文件服务 SFS，具体请参见**数据迁移**。

数据库迁移

若您的数据库采用集群外的非容器化部署方案，且需将数据库搬迁上云，可以使用华为云**数据复制服务 DRS**帮助完成数据库迁移。DRS服务具有实时迁移、备份迁移、实时同步、数据订阅和实时灾备等多种功能。

其他数据迁移

- 大数据场景迁移：推荐使用华为云**云数据迁移 CDM**。
- Kafka业务迁移：具体请参见华为云分布式消息服务Kafka版的**Kafka业务迁移**。
- Redis业务迁移：具体请参见华为云分布式缓存服务 DCS的**数据迁移指南**。

10.3.5 应用备份

本地IDC集群中应用的迁移包含两个步骤：应用备份和应用迁移，即备份本地IDC集群中应用，然后通过数据恢复的方式迁移至目标集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群（UCS华为云集群或本地集群）中，从而实现本地IDC集群应用的迁移上云。

须知

建议在用户业务量小时执行备份操作。

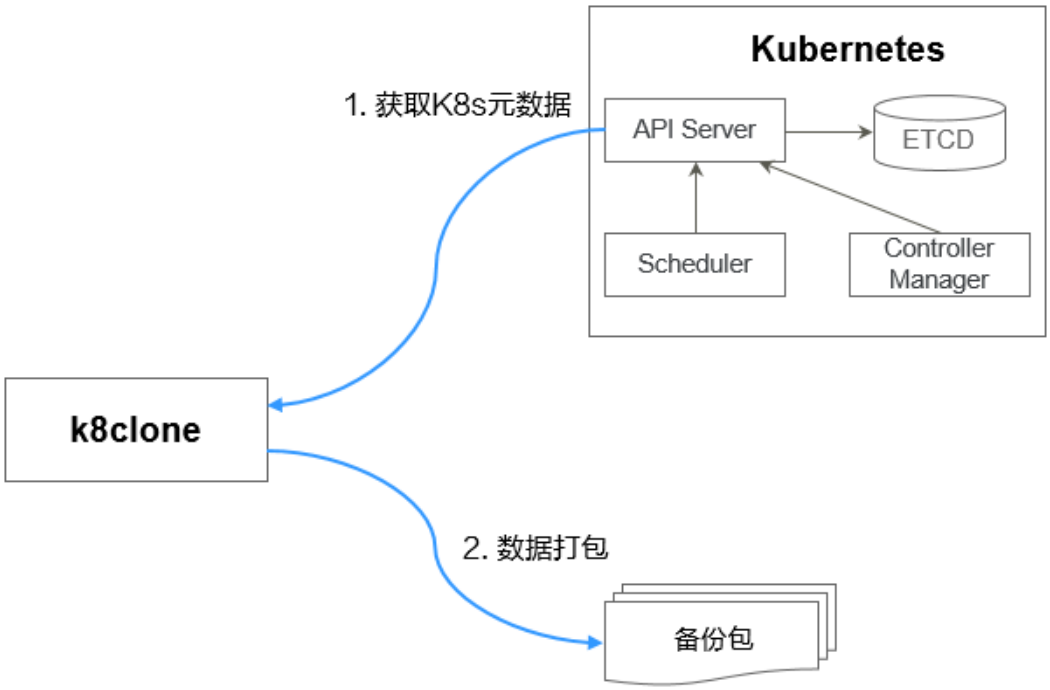
前提条件

确认云原生应用依赖的服务（镜像、存储、数据库等非集群内的数据）都已经迁移完成。

k8clone 数据备份原理

数据备份的流程参考如下：

图 10-4 数据备份流程



k8clone 备份使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 backup -h**，可以查看k8clone工具备份的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见Kubernetes文档。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -n, --namespace：备份指定命名空间的云原生应用，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示备份整个集群。
- -e, --exclude-namespaces：排除指定命名空间对象的备份，不能和--namespace一起使用。
- -x, --exclude-kind：排除指定资源类型的备份。
- -i, --include-kind : 指定资源类型的备份。
- -y, --exclude-object：排除指定资源对象的备份。
- -z, --include-object：指定资源对象的备份。
- -w, --exclude-having-owner-ref：排除拥有ownerReferences资源对象的备份，默认是false。传参过程中需要带上等号，如：-w=true 。

- -d, --local-dir：备份数据放置的路径，默认是当前目录下k8clone-dump文件夹。

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
   k8clone backup [flags]

Flags:
   -s, --api-server string            Kubernetes api-server url
   -q, --context string               Kubernetes configuration context
   -w, --exclude-having-owner-ref     Exclude all objects having an Owner Reference. The following form is not permitted for boolean flags such as '-w false', please use '-w=false'
   -x, --exclude-kind strings         Ressource kind to exclude. Eg. 'deployment'
   -i, --include-kind strings         Ressource kind to include. Eg. 'deployment'
   -e, --exclude-namespaces strings   Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all namespaces and then filters them. Don't use it with the namespace flag.
   -y, --exclude-object strings       Object to exclude. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -z, --include-object strings       Object to include. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -h, --help                         help for backup
   -k, --kubeconfig string            The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
   -d, --local-dir string             Where to dump yaml files (default "./k8clone-dump")
   -n, --namespace string             Only dump objects from this namespace
```

示例如下：

- 整个集群对象的备份，默认路径为当前目录下“k8clone-dump”文件夹

  ./k8clone-linux-amd64 backup
- 整个集群对象的备份，并指定备份数据路径

  ./k8clone-linux-amd64 backup -d ./xxxx
- 指定命名空间对象的备份

  ./k8clone-linux-amd64 backup -n default
- 排除命名空间对象的备份

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease
- 排除指定资源类型的备份

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice
- 指定资源类型的备份

  ./k8clone-linux-amd64 backup -x rolebinding
- 排除指定资源对象的备份

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns
- 指定资源对象的备份

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns
- 排除拥有ownerReferences资源对象的备份

  ./k8clone-linux-amd64 backup -w=true

应用备份操作步骤

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 进入k8clone工具所在目录，执行备份命令，即可备份数据到本地目录，并打包成压缩包。

k8clone备份使用方法的示例中提供了几种常见的备份方式，您可以按需选择，也可以自定义备份方式。

----结束

10.3.6 应用迁移

本地IDC集群中应用的迁移包含两个步骤：应用备份和应用迁移，即备份本地IDC集群中应用，然后通过数据恢复的方式迁移至目标集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群（UCS华为云集群或本地集群）中，从而实现本地IDC集群应用的迁移上云。

约束限制

当前不支持高版本集群应用向低版本集群迁移。

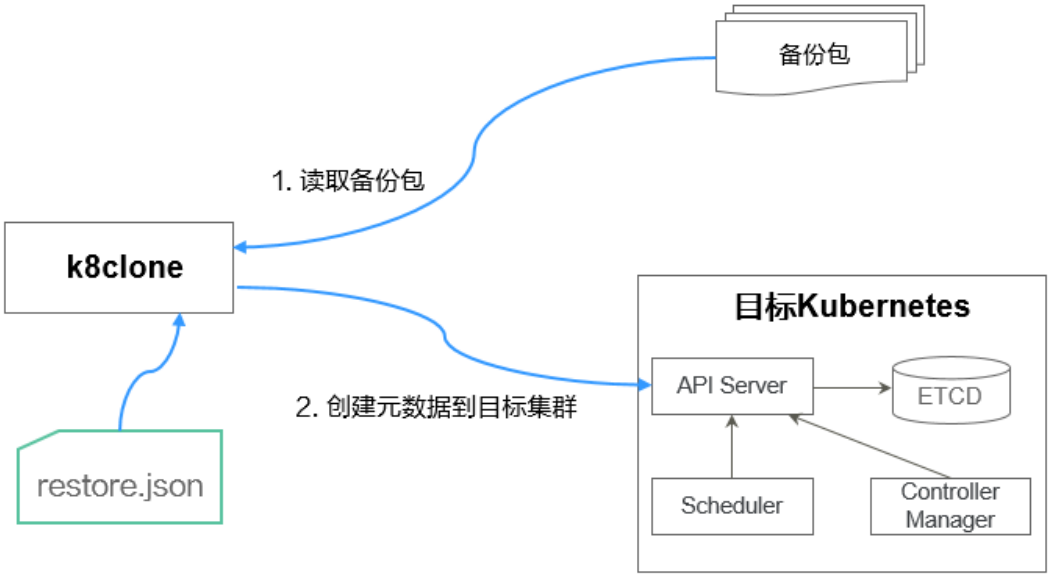
前提条件

- 确认云原生应用依赖的服务（镜像、存储、数据库等非集群内的数据）都已经迁移完成。
- 确认源集群中元数据备份数据已经下载到执行k8clone的服务器上。

k8clone 数据恢复原理

数据恢复的流程参考如下：

图 10-5 数据恢复流程



在执行恢复操作前，需要准备一个数据恢复配置文件“restore.json”，目的是在应用恢复时自动更换PVC、StatefulSet的存储类名称，以及工作负载所使用镜像的Repository地址。

文件内容如下：

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  //支持修改PVC、StatefulSet的StorageClassName
字段
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-
north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- StorageClass：支持PVC、有状态应用VolumeClaimTemplates中存储类名称按照配置进行自动更换。
- ImageRepo：支持工作负载所使用镜像的Repository地址的更换，工作负载包括Deployment（含initContainer）、StatefulSet、Orphaned Pod、Job、CronJob、Replica Set、Replication Controller、DaemonSet。

k8clone 恢复使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 restore -h**，可以查看k8clone工具恢复的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -f, --restore-conf：指定restore.json的路径，默认是k8clone工具所在目录下。
- -d, --local-dir：备份数据放置的路径，默认是k8clone工具所在目录下。

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
    k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

应用恢复操作步骤

步骤1 通过kubectl连接目标集群。具体方法可参考**使用kubectl连接集群**。

步骤2 准备数据恢复配置文件：restore.json。

新建一个restore.json文件，按照格式修改，并将文件放置在k8clone工具所在目录下。

示例：

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

步骤3 进入k8clone工具所在目录，执行恢复命令，将备份数据恢复到目标集群。

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

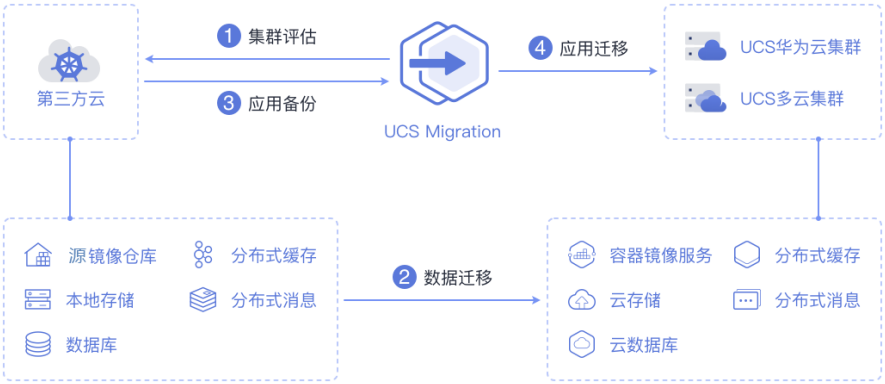
----结束

10.4 第三方云集群跨云迁移

10.4.1 第三方云集群跨云迁移流程

UCS容器迁移支持将第三方云Kubernetes托管集群应用迁移到UCS华为云集群或多云集群，实现跨云迁移和统一管理。

图 10-6 迁移流程



主要包含四个步骤：

步骤1 集群评估

在这个阶段，您将根据源集群的现状来评估适合迁移的目标集群类型。UCS的kspider工具能够自动收集源集群的信息，包括Kubernetes版本、规模、工作负载、存储等数据，并根据收集到的数据为您提供推荐的目标集群信息。具体请参见**集群评估**。

步骤2 数据迁移

在这个阶段，您将把镜像和相关依赖服务的数据迁移到云端。UCS的image-migrator是一个自动化镜像迁移工具，可以将第三方云镜像仓库中的镜像迁移到SWR中；对于依赖服务的数据迁移，您可以使用华为云的其他云产品来配套实现。

步骤3 应用备份

在这个阶段，您将对第三方云集群中的应用进行备份。UCS的k8clone工具可以自动收集Kubernetes元数据，并将其以压缩包的形式保存到本地，从而实现集群中应用的备份。具体请参见**应用备份**。

步骤4 应用迁移

在这个阶段，您将利用备份数据恢复的方法，将第三方云集群中的应用迁移到UCS华为云集群或多云集群。具体请参见**应用迁移**。

----结束

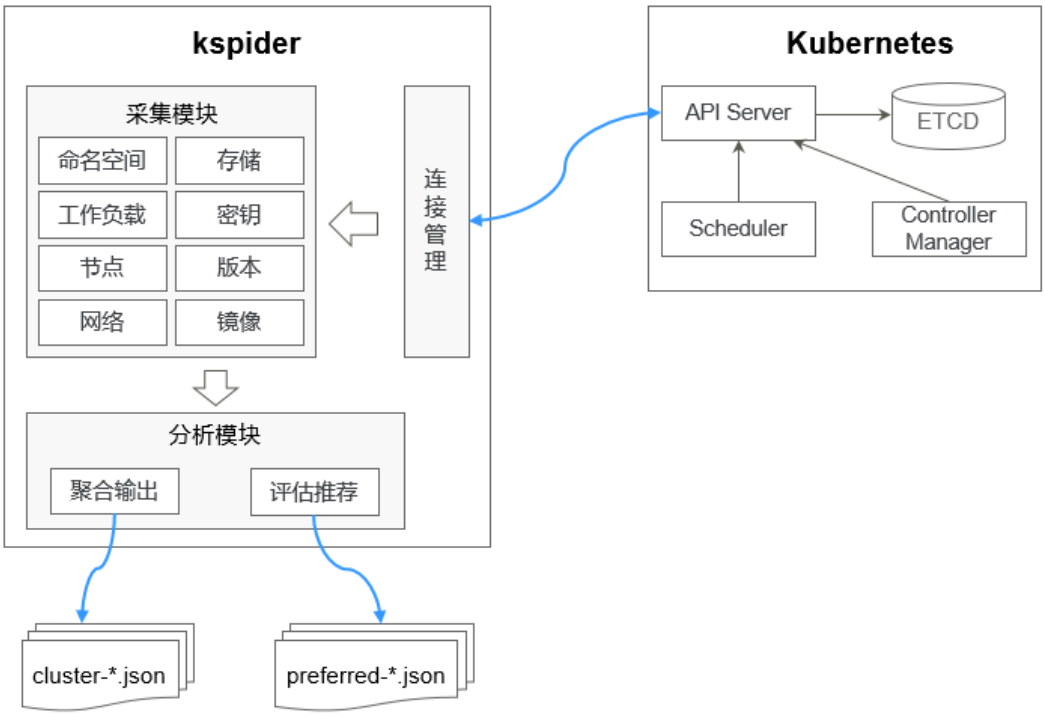
10.4.2 集群评估

将应用从一个环境迁移到另一个环境是一项具有挑战性的任务，因此您需要进行仔细的规划和准备。kspider是一款用于采集源集群信息的工具，它向用户提供了集群的Kubernetes版本、规模、工作负载数量、存储以及正在使用的镜像等数据，这些信息有助于用户了解集群的当前状况，评估迁移风险，并选择合适的目标集群版本和规模。

kspider 工作原理

kspider工具的架构如**图10-7**所示，包含三个模块：采集模块、连接管理和分析模块。采集模块可以收集源集群的数据，包括命名空间、工作负载、节点、网络等；连接管理模块负责与源集群的API Server建立连接；分析模块分为聚合输出和评估推荐两部分，旨在输出源集群的采集数据（生成“cluster-*.json”文件）以及提供目标集群的推荐信息（生成“preferred-*.json”文件）。

图 10-7 kspider 架构



kspider 使用方法

说明

kspider工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**kspider-linux-amd64**分别替换为**kspider-linux-arm64**或**kspider-windows-amd64.exe**。

根据**容器迁移准备工作**章节的要求，准备一台服务器并上传kspider工具，然后进行解压缩。在kspider工具所在目录下执行**./kspider-linux-amd64 -h**，您可以查看该工具的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -n, --namespaces：指定采集的命名空间，默认排除了kube-system、kube-public、kube-node-lease等系统命名空间。
- -q, --quiet：静态退出。
- -s, --serial：根据采集信息输出汇聚文件（cluster-{serial}.json）和推荐文件（preferred-{serial}.json）唯一标识的序号。

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]

Aliases:
  kspider, kspider

Flags:
  -h, --help              help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default "$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet             command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when the kspider is started. (default "1673853404")
```

步骤一：源集群数据采集

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 使用默认参数配置，采集集群中所有命名空间的数据。执行方法：**./kspider-linux-amd64**

执行后的输出详细信息如下：

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
   Name     CPU     Memory     IP     Arch     OS     Kernel     MachineID
   10.1.18.64     4     8008284Ki     [10.1.18.64 10.1.18.64]     amd64     linux     3.10.0-1127.19.1.el7.x86_64     ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
   10.1.19.13     4     8008284Ki     [10.1.19.13 10.1.19.13]     amd64     linux     3.10.0-1127.19.1.el7.x86_64     2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
```

```
  Name  Namespace  NodeAffinity
  minio  default  false
  minio  minio  false
There are 3 Deployments:
  Name  Namespace  NodeAffinity
  rctest  default  true
  flink-operator-controller-manager  flink-operator-system  false
  rctest  minio  false
There are 1 DaemonSets:
  Name  Namespace  NodeAffinity
  ds-nginx  minio  false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
  Namespace/Name  Pods
  default/pvc-data-minio-0  default/minio-0
  minio/obs-testing  minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
  minio/pvc-data-minio-0  minio/minio-0
There are 5 PersistentVolumes:
  Name  Namespace  pvcName  scName  size  key
  pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8  minio-pv  obs-testing  csi-obs  1Gi  pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
  pvc-c7c768aa-373a-4c52-abea-e8b486d23b47  minio-pv  pvc-data-minio-0  csi-disk-sata  10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
  pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05  minio  obs-testing  csi-obs  1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
  pvc-9fd92c99-805a-4e65-9f22-e238130983c8  default  pvc-data-minio-0  csi-disk  10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
  pvc-a22fd877-f98d-4c3d-a04e-191d79883f97  minio  pvc-data-minio-0  csi-disk-sata  10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
  Name  Namespace  ServiceType
  headless-lxprus  default  ClusterIP
  kubernetes  default  ClusterIP
  minio  default  NodePort
  flink-operator-controller-manager-metrics-service  flink-operator-system  ClusterIP
  flink-operator-webhook-service  flink-operator-system  ClusterIP
  headless-lxprus  minio  ClusterIP
  minio  minio  NodePort
There are 0 Ingresses:
There are 6 Images:
  Name
  gcr.io/flink-operator/flink-operator:v1beta1-6
  flink:1.8.2
  swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
  nginx:stable-alpine-perl
  swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
  gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
  SecretType
  cfe/secure-opaque
  helm.sh/release.v1
```

在kspider执行完毕后，当前目录下将生成两个文件：

- cluster-*.json：此文件包含了源集群及应用的采集数据，这些数据可用于分析和规划迁移过程。
- preferred-*.json：此文件包含了推荐的目标集群信息。基于源集群的规模和节点规格进行初步评估，文件将提供关于目标集群版本和规模的建议。

步骤3 查看源集群及应用的采集数据。

您可以用文本编辑器或JSON查看器打开“cluster-*.json”文件以查看数据。在实际操作中，您需要将文件名中的“*”替换为实际的时间戳或序列号，以找到并打开正确的文件。

“cluster-*.json”文件说明如下：


```
{
 K8sVersion：Kubernetes版本，字符串类型
 Namespaces：命名空间数量，字符串类型
 Pods：Pod总数量，整型
 Nodes：节点总信息，以IP为key，展示节点信息
  IP地址
    CPU：CPU，字符串类型
    Arch：CPU架构，字符串类型
    Memory：内存，字符串类型
    HugePages1Gi：1G大页内存，字符串类型
    HugePages2Mi：2M大页内存，字符串类型
    OS：节点OS，字符串类型
    KernelVersion：OS内核版本，字符串类型
    RuntimeVersion：节点容器运行及版本，字符串类型
    InternalIP：内部IP，字符串类型
    ExternalIP：外部IP，字符串类型
    MachineID：节点ID，字符串类型。说明：CCE中能够保证与ECS的ID一致
 Workloads：工作负载
  Deployment：工作负载类型，支持Deployment（无状态负载）、StatefulSet（有状态负载）、DaemonSet
（守护进程集）、CronJob（定时任务）、Job（普通任务）、LonePod（独立Pod）
    default：命名空间名称
      Count：数量，整型
      Items：详细信息，数组类型
        Name：工作负载名称，字符串类型
        Namespace：命名空间名称，字符串类型
        NodeAffinity：节点亲和性，布尔型
        Replicas：副本数量，整型
 Storage：存储
  PersistentVolumes：持久卷
    pv-name：以PV名称为key
      VolumeID：卷ID，字符串类型
      Namespace：命名空间，字符串类型
      PvcName：绑定PVC的名称，字符串类型
      ScName：存储类的名称，字符串类型
      Size：申请空间大小，字符串类型
      Pods：使用PV的Pod名称，字符串类型
      NodeIP：Pod所在的节点IP，字符串类型
      VolumePath：该Pod挂载节点的路径，字符串类型
  OtherVolumes：其它类型卷
    类型：AzureFile、AzureDisk、GCEPersistentDisk、AWSElasticBlockStore、Cinder、Glusterfs、NFS、
CephFS、FlexVolume、FlexVolume、DownwardAPI
      卷ID/卷名称/卷共享路径等为key
        Pods：使用其的Pod，字符串类型
        NodeIP：Pod所在的节点IP，字符串类型
        卷ID/卷名称/卷共享路径等唯一标识卷信息的信息，字符串类型
 Networks：网络
  LoadBalancer：负载均衡类型
    service：网络类型，包括service和ingress
      Name：名称，字符串类型
      Namespace：命名空间名称，字符串类型
      Type：类型，字符串类型
 ExtraSecrets：扩展secret类型
  secret类型名，字符串类型
 Images：镜像
  镜像repo，字符串类型
}
```

示例：

```
{
 "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
 "Namespaces": 12,
 "Pods": 33,
 "Nodes": {
  "10.1.17.219": {
   "CPU": "4",
   "Memory": "7622944Ki",
   "HugePages1Gi": "0",
   "HugePages2Mi": "0",
   "Arch": "amd64",
```



```
    "OS": "EulerOS 2.0 (SP9x86_64)",
    "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
    "RuntimeVersion": "docker://18.9.0",
    "InternalIP": "10.1.17.219",
    "ExternalIP": "",
    "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
   },
   "10.1.18.182": {
    "CPU": "4",
    "Memory": "7992628Ki",
    "HugePages1Gi": "0",
    "HugePages2Mi": "0",
    "Arch": "amd64",
    "OS": "EulerOS 2.0 (SP5)",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
    "RuntimeVersion": "docker://18.9.0",
    "InternalIP": "10.1.18.182",
    "ExternalIP": "100.85.xxx.xxx",
    "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
   }
  },
  "Workloads": {
   "CronJob": {},
   "DaemonSet": {
    "default": {
     "Count": 1,
     "Items": [
      {
       "Name": "kubecost-prometheus-node-exporter",
       "Namespace": "default",
       "NodeAffinity": false,
       "Replicas": 3
      }
     ]
    }
   },
   "Deployment": {
    "default": {
     "Count": 1,
     "Items": [
      {
       "Name": "kubecost-cost-analyzer",
       "Namespace": "default",
       "NodeAffinity": false,
       "Replicas": 1
      }
     ]
    },
    "kubecost": {
     "Count": 1,
     "Items": [
      {
       "Name": "kubecost-kube-state-metrics",
       "Namespace": "kubecost",
       "NodeAffinity": false,
       "Replicas": 1
      }
     ]
    }
   },
   "Job": {},
   "LonePod": {},
   "StatefulSet": {
    "minio-all": {
     "Count": 1,
     "Items": [
      {
       "Name": "minio",
       "Namespace": "minio-all",
```



```
          "NodeAffinity": false,
          "Replicas": 1
        }
       ]
     }
   }
 },
 "Storage": {
   "PersistentVolumes": {
     "demo": {
       "VolumeID": "demo",
       "Namespace": "fluid-demo-test",
       "PvcName": "demo",
       "ScName": "fluid",
       "Size": "100Gi",
       "Pods": "",
       "NodeIP": "",
       "VolumePath": ""
     },
     "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
       "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
       "Namespace": "minio-all",
       "PvcName": "pvc-data-minio-0",
       "ScName": "csi-disk",
       "Size": "10Gi",
       "Pods": "minio-all/minio-0",
       "NodeIP": "10.1.23.159",
       "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
     }
   },
   "OtherVolumes": {}
 },
 "Networks": {
   "LoadBalancer": {}
 },
 "ExtraSecrets": [
   "cfe/secure-opaque",
   "helm.sh/release.v1"
 ],
 "Images": [
   "nginx:stable-alpine-perl",
   "ghcr.io/koordinator-sh/koord-manager:0.6.2",
   "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
   "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
   "gcr.io/kubecost1/cost-model:prod-1.91.0",
   "gcr.io/kubecost1/frontend:prod-1.91.0"
 ]
}
```


----结束

步骤二：目标集群评估

在kspider执行完毕后，除了“cluster-*.json”文件之外，还会在当前目录下生成“preferred-*.json”文件。这个文件基于源集群的规模和节点规格进行初步评估，并提供关于目标集群版本和规模的推荐信息。这有助于您更好地规划和准备迁移过程。

“preferred-*.json”文件说明如下：

```
{
 K8sVersion：Kubernetes版本，字符串类型
 Scale：集群规模，字符串类型
 Nodes：节点信息
   CPU：CPU，字符串类型
   Memory：内存，字符串类型
   Arch：架构，字符串类型
   KernelVersion：OS内核版本，字符串类型
```



```
  ProxyMode：集群Proxy模式，字符串类型
 ELB：是否依赖ELB，布尔型
}
```

上述文件中每个字段的评估规则如下：

表 10-4 评估规则

字段	评估规则
Kubernetes版本	如果是1.21以下版本，推荐UCS集群主要发行版本（例如1.21，随着时间发展会发生变化），大于主要发行版本时，将推荐UCS集群的最新版本。
集群规模	源集群节点数＜25，推荐50节点规模 25≤源集群节点数＜100，推荐200节点规模 100≤源集群节点数＜500，推荐1000节点规模 源集群节点数≥500，推荐2000节点规模
CPU+内存	统计数量最多的那一种规格
架构	统计数量最多的那一种规格
OS内核版本	统计数量最多的那一种规格
集群Proxy模式	根据集群规模来设置，大于1000节点规模的集群，推荐使用ipvs，1000以内的推荐使用iptables。
是否依赖ELB	源集群是否有负载均衡类型的Service

示例：

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

⚠ 注意

评估结果仅供参考，最终选择什么版本、规模的目标集群还需要您综合判断。

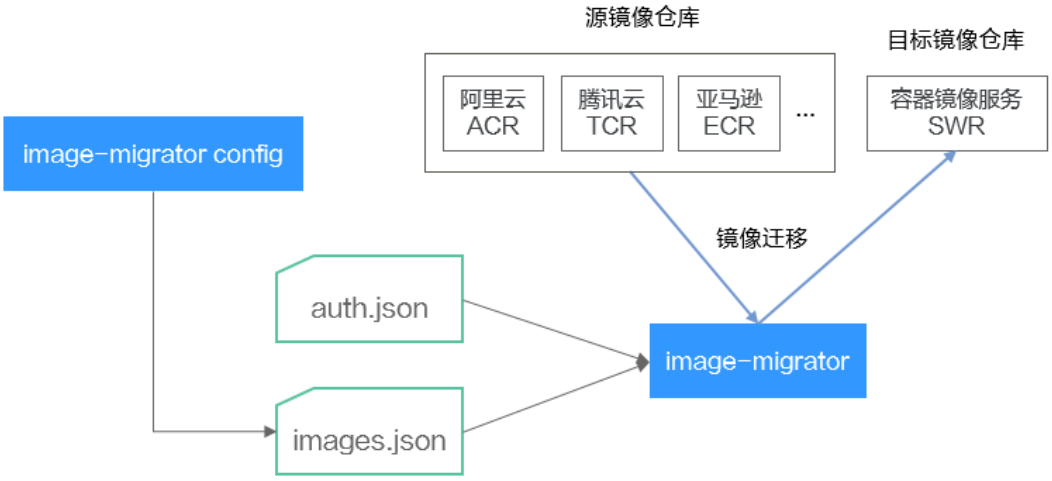
10.4.3 镜像迁移

为保证集群迁移后容器镜像可正常拉取，提升容器部署效率，建议您将第三方云镜像仓库迁移至华为云容器镜像服务（SWR）。UCS华为云集群和多云集群配合SWR为您提供容器自动化交付流水线，采用并行传输的镜像拉取方式，能够大幅提升容器的交付效率。

image-migrator是一个镜像迁移工具，能够自动将第三方云镜像仓库中的镜像迁移到SWR中。

image-migrator 工作原理

图 10-8 image-migrator 工作原理



使用image-migrator工具将镜像迁移到SWR时，需要准备两个文件，一个为镜像仓库访问权限文件“auth.json”，该文件中的两个对象分别为源镜像仓库和目标镜像仓库（即Registry）的账号和密码；另一个为镜像列表文件“images.json”，文件内容由多条镜像同步规则组成，每一条规则包括一个源镜像仓库（键）和一个目标镜像仓库（值）。将这两个文件准备好以后，放在image-migrator工具所在目录下，执行一条简单的命令，就可以完成镜像的迁移。关于两个文件的详细介绍如下：

- **“auth.json”文件**

  “auth.json”为镜像仓库访问权限文件，其中每个对象为一个registry的用户名和密码。通常源镜像仓库需要具有pull以及访问tags权限，目标镜像仓库需要拥有push以及创建仓库权限。如果是匿名访问镜像仓库，则不需要填写用户名、密码等信息。“auth.json”文件的结构如下：

  ```
  {
    "源镜像仓库地址": { },
    "目标镜像仓库地址": {
       "username": "xxxxxx",
       "password": "***************",
       "insecure": true
    }
  }
  ```

  其中，文件内各个参数的描述与填写指导请参见表10-5。

  表 10-5 auth.json 文件参数描述

参数	描述
源镜像仓库地址	支持“registry”和“registry/namespace”两种形式，需要跟下述“images.json”中的registry或registry/namespace对应。 **说明** images中被匹配到的url会使用对应用户名密码进行镜像同步，优先匹配“registry/namespace”的形式。

参数	描述
目标镜像仓库地址	支持“registry”和“registry/namespace”两种形式，需要跟下述“images.json”中的registry或registry/namespace对应。 ● 若您的镜像仓库为华为云SWR，且目标镜像仓库地址为“registry”形式，可以从SWR控制台页面获取，具体方法如下： 在“总览”页面单击右上角“登录指令”，登录指令末尾的域名即为SWR镜像仓库地址，例如swr.cn-north-4.myhuaweicloud.com。注意每个Region的地址不同，请切换到对应Region获取。 如果为“registry/namespace”形式，还要将namespace替换为SWR的组织名称。 ● 若您的镜像仓库为Amazon ECR或ACR，请登录相应厂商的镜像仓库控制台，查看镜像仓库的推送命令，获取相应镜像仓库地址。
username	用户名，values可以填写具体取值，也可以使用“${env}”或者“$env”类型的字符串引用环境变量。 ● 若您的镜像仓库为华为云SWR，则目标镜像仓库SWR的用户名形式为：区域项目名称@AK。 ● 若您的镜像仓库为Amazon ECR或ACR，请登录相应厂商的镜像仓库控制台，查看镜像仓库的推送命令，获取相应账号。
password	密码，values可以填写具体取值，也可以使用“${env}”或者“$env”类型的字符串引用环境变量。 ● 若您的镜像仓库为华为云SWR，则目标镜像仓库SWR的密码为AK和SK经过加密处理后的登录密钥，详细指导请参考**获取长期有效登录指令**。 ● 若您的镜像仓库为Amazon ECR或ACR，请登录相应厂商的镜像仓库控制台，查看镜像仓库的推送命令，获取相应密码。
insecure	registry是否为http服务，如果是，insecure为true；默认是false。

示例：

```
{
   "quay.io/coreos": { },
   "swr.cn-north-4.myhuaweicloud.com": {
      "username": "cn-north-4@RVHVMX******",
      "password": "***************",
      "insecure": true
   }
}
```

- **“images.json”文件**

  该文件本质上是一个待迁移的镜像清单，由多条镜像同步规则组成，每一条规则包括一个源镜像仓库（键）和一个目标镜像仓库（值）。具体的要求如下：

  a. 同步的最大单位是仓库（repository），不支持通过一条规则同步整个namespace以及registry。

b. 源仓库和目标仓库的格式与docker pull/push命令使用的镜像url类似（ registry/namespace/repository:tag ）。
c. 源仓库和目标仓库（ 如果目标仓库不为空字符串 ）都至少包含registry/namespace/repository。
d. 源仓库字段不能为空，如果需要将一个源仓库同步到多个目标仓库需要配置多条规则。
e. 目标仓库名可以和源仓库名不同，此时同步功能类似于：docker pull + docker tag + docker push。
f. 当源仓库字段中不包含tag时，表示将该仓库所有tag同步到目标仓库，此时目标仓库不能包含tag。
g. 当源仓库字段中包含tag时，表示只同步源仓库中的一个tag到目标仓库，如果目标仓库中不包含tag，则默认使用源tag。
h. 当目标仓库为空字符串时，会将源镜像同步到默认registry的默认namespace下，并且repository以及tag与源仓库相同，默认registry和默认namespace可以通过命令行参数以及环境变量配置。

示例如下：

```
{
    "quay.io/coreos/etcd:1.0.0": "swr.cn-north-4.myhuaweicloud.com/test/etcd:1.0.0",
    "quay.io/coreos/etcd": "swr.cn-north-4.myhuaweicloud.com/test/etcd",
    "quay.io/coreos/etcd:2.7.3": "swr.cn-north-4.myhuaweicloud.com/test/etcd"
}
```

使用image-migrator工具的config子命令可自动获取集群中工作负载正在使用的镜像，具体用法请参见**image-migrator config使用方法**。得到images.json文件后，您还可以根据需要进行修改、添加或删除。

image-migrator 使用方法

说明

image-migrator工具支持在Linux（ x86、arm ）和Windows环境中运行，使用方法相似。本文将以Linux（ x86 ）环境为例进行介绍。

若使用Linux（ arm ）或Windows环境，请将下述命令中的**image-migrator-linux-amd64**分别替换为**image-migrator-linux-arm64**或**image-migrator-windows-amd64.exe**。

在image-migrator工具所在目录下执行**./image-migrator-linux-amd64 -h**，可以查看image-migrator工具的使用方法。

- --auth：指定auth.json的路径，默认在image-migrator所在目录下。
- --images：指定images.json的路径，默认在image-migrator所在目录下。
- --log：指定image-migrator生成日志的路径，默认是image-migrator当前目录下的image-migrator.log。
- --namespace：默认的目标仓库的namespace，也就是说，如果images.json中没有指定目标仓库中的namespace，可以在执行迁移命令时指定。
- --registry：默认的目标仓库的registry，也就是说，如果images.json中没有指定目标仓库中的registry，可以在执行迁移命令时指定。
- --retries：迁移失败时的重试次数，默认为3。
- --workers：镜像搬迁的worker数量（ 并发数 ），默认是7。

```
$ ./image-migrator-linux-amd64 -h
A Fast and Flexible docker registry image images tool implement by Go.

Usage:
```



```
  image-migrator [flags]

Aliases:
  image-migrator, image-migrator

Flags:
      --auth string        auth file path. This flag need to be pair used with --images. (default "./auth.json")
  -h, --help               help for image-migrator
      --images string      images file path. This flag need to be pair used with --auth (default "./images.json")
      --log string         log file path (default "./image-migrator.log")
      --namespace string   default target namespace when target namespace is not given in the images config file, can also be set with DEFAULT_NAMESPACE environment value
      --registry string    default target registry url when target registry is not given in the images config file, can also be set with DEFAULT_REGISTRY environment value
  -r, --retries int        times to retry failed tasks (default 3)
  -w, --workers int        numbers of working goroutines (default 7)

$./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test \
--registry=swr.cn-north-4.myhuaweicloud.com --retries=2
$ ./image-migrator
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

示例如下：

./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

该命令表示将“images.json”文件中的镜像迁移至“swr.cn-north-4.myhuaweicloud.com/test”镜像仓库下，迁移失败时可以重试2次，一次可以同时搬迁5个镜像。

image-migrator config 使用方法

image-migrator工具的config子命令可用于获取集群应用中使用的镜像，在工具所在目录下生成images.json。执行**./image-migrator-linux-amd64 config -h**命令可以查看config子命令的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -n, --namespaces：指定获取镜像的命名空间，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示获取所有命名空间的镜像。
- -t, --repo：指定目标仓库的地址（registry/namespace）。

```
$ ./image-migrator-linux-amd64 config -h
generate images.json

Usage:
   image-migrator config [flags]

Flags:
   -h, --help               help for config
   -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default "/root/.kube/config")
   -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
   -t, --repo string         target repo,such as swr.cn-north-4.myhuaweicloud.com/test
```

示例如下：

- 指定一个命名空间

  ./image-migrator-linux-amd64 config -n default -t swr.cn-north-4.myhuaweicloud.com/test

- 指定多个命名空间

  ./image-migrator-linux-amd64 config -n default,kube-system -t swr.cn-north-4.myhuaweicloud.com/test

- 不指定命名空间（表示获取所有命名空间的镜像）

  ./image-migrator-linux-amd64 config -t swr.cn-north-4.myhuaweicloud.com/test

镜像迁移操作步骤

步骤1 准备镜像仓库访问权限文件：auth.json。

新建一个auth.json文件，并按照格式修改，如果是匿名访问仓库，则不需要填写用户名、密码等信息。将文件放置在image-migrator所在目录下。

示例：

```
{
    "quay.io/coreos": { },
    "swr.cn-north-4.myhuaweicloud.com": {
        "username": "cn-north-4@RVHVMX******",
        "password": "***************",
        "insecure": true
    }
}
```

详细的参数说明请参见“**auth.json**”**文件**。

步骤2 准备镜像列表文件：images.json。

1. 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。
2. 执行镜像迁移config子命令，生成images.json文件。

   您可以参考**image-migrator config使用方法**中的方法和示例，不指定命名空间，或者指定一个、多个命名空间来获取源集群应用中使用的镜像。
3. 根据需求调整images.json文件内容，但要遵循“**images.json**”**文件**中所讲的八项要求。

步骤3 镜像迁移。

您可以执行默认的**./image-migrator-linux-amd64**命令进行镜像迁移，也可以根据需要设置image-migrator的参数。

例如以下命令：

./image-migrator-linux-amd64 --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

示例：

```
$ ./image-migrator-linux-amd64
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

步骤4 结果查看。

上述命令执行完毕后，回显如下类似信息：

```
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

表示按照配置，成功将38个镜像迁移到SWR仓库中。

----结束

10.4.4 依赖服务迁移

本节介绍集群依赖服务的相关数据迁移，如存储、数据库、分布式缓存、分布式消息等。若您的集群不涉及这些数据，或者这些数据不需要搬迁至华为云，可忽略本节内容。

存储迁移

- 若您的集群使用了云硬盘，跨云迁移可以使用华为云**数据快递服务 DES**。DES服务是一种海量数据传输解决方案，支持TB到几百TB级数据上云，通过Teleport设备或硬盘（外置USB接口、SATA接口、SAS接口）向华为云传输大量数据，致力于解决海量数据传输网络成本高、传输时间长等难题。
- 若您的集群使用了对象存储，跨云迁移可以使用华为云**对象存储迁移服务 OMS**。OMS服务是一种线上数据迁移服务，帮助您将其他云服务商对象存储中的数据在线迁移至华为云的对象存储服务 OBS中。
- 若您的集群使用了文件存储，跨云迁移可以使用华为云弹性文件服务 SFS，具体请参见**数据迁移**。

数据库迁移

若您的数据库需要搬迁至华为云，可以使用**数据复制服务 DRS**帮助完成数据库迁移。DRS服务具有实时迁移、备份迁移、实时同步、数据订阅和实时灾备等多种功能。

其他数据迁移

- 大数据场景迁移：推荐使用华为云**云数据迁移 CDM**。
- Kafka业务迁移：具体请参见华为云分布式消息服务Kafka版的**Kafka业务迁移**。
- Redis业务迁移：具体请参见华为云分布式缓存服务 DCS的**数据迁移指南**。

10.4.5 应用备份

第三方云集群中应用的迁移包含两个步骤：应用备份和应用迁移，即备份第三方云集群中应用，然后通过数据恢复的方式迁移至目标集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群（UCS华为云集群或多云集群）中，从而实现第三方云集群应用的迁移上云。

须知

建议在用户业务量小时执行备份操作。

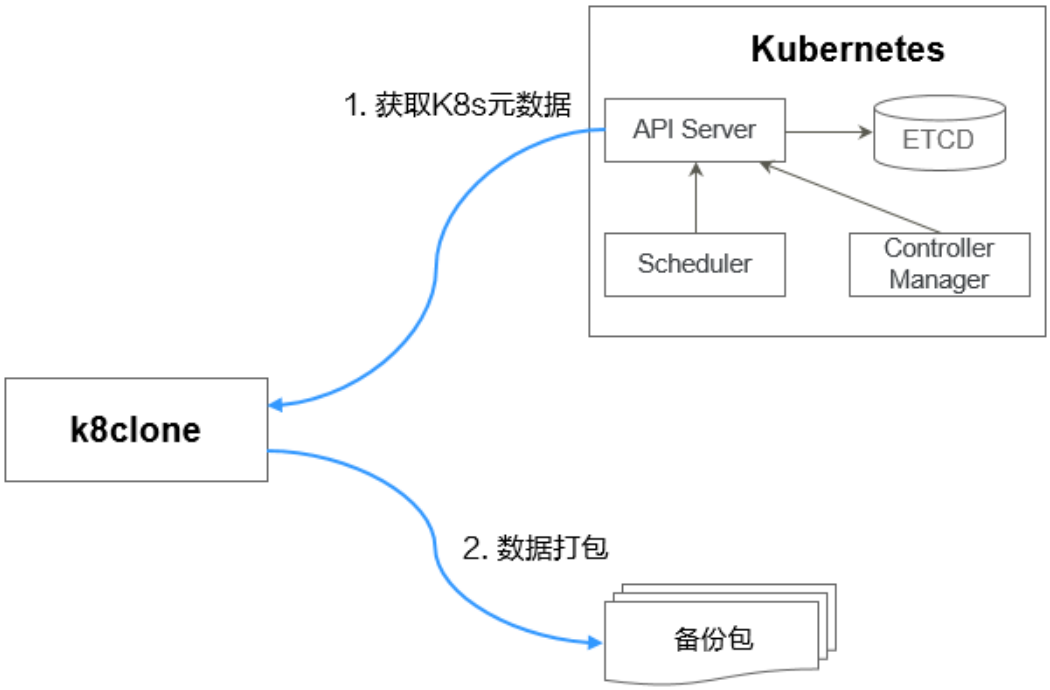
前提条件

确认云原生应用依赖的服务（镜像、存储、数据库等非集群内的数据）都已经迁移完成。

k8clone 数据备份原理

数据备份的流程参考如下：

图 10-9 数据备份流程



k8clone 备份使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 backup -h**，可以查看k8clone工具备份的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见Kubernetes文档。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -n, --namespace：备份指定命名空间的云原生应用，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示备份整个集群。

- -e, --exclude-namespaces：排除指定命名空间对象的备份，不能和--namespace一起使用。
- -x, --exclude-kind：排除指定资源类型的备份。
- -i, --include-kind : 指定资源类型的备份。
- -y, --exclude-object：排除指定资源对象的备份。
- -z, --include-object：指定资源对象的备份。
- -w, --exclude-having-owner-ref：排除拥有ownerReferences资源对象的备份，默认是false。传参过程中需要带上等号，如：-w=true 。
- -d, --local-dir：备份数据放置的路径，默认是当前目录下k8clone-dump文件夹。

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
   k8clone backup [flags]

Flags:
   -s, --api-server string            Kubernetes api-server url
   -q, --context string               Kubernetes configuration context
   -w, --exclude-having-owner-ref     Exclude all objects having an Owner Reference. The following form is not permitted for boolean flags such as '-w false', please use '-w=false'
   -x, --exclude-kind strings         Ressource kind to exclude. Eg. 'deployment'
   -i, --include-kind strings         Ressource kind to include. Eg. 'deployment'
   -e, --exclude-namespaces strings   Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all namespaces and then filters them. Don't use it with the namespace flag.
   -y, --exclude-object strings       Object to exclude. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -z, --include-object strings       Object to include. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -h, --help                         help for backup
   -k, --kubeconfig string            The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
   -d, --local-dir string             Where to dump yaml files (default "./k8clone-dump")
   -n, --namespace string             Only dump objects from this namespace
```

示例如下：

- 整个集群对象的备份，默认路径为当前目录下“k8clone-dump”文件夹

  ./k8clone-linux-amd64 backup
- 整个集群对象的备份，并指定备份数据路径

  ./k8clone-linux-amd64 backup -d ./xxxx
- 指定命名空间对象的备份

  ./k8clone-linux-amd64 backup -n default
- 排除命名空间对象的备份

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease
- 排除指定资源类型的备份

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice
- 指定资源类型的备份

  ./k8clone-linux-amd64 backup -x rolebinding
- 排除指定资源对象的备份

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns
- 指定资源对象的备份

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns

- 排除拥有ownerReferences资源对象的备份
  ./k8clone-linux-amd64 backup -w=true

应用备份操作步骤

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 进入k8clone工具所在目录，执行备份命令，即可备份数据到本地目录，并打包成压缩包。

k8clone备份使用方法的示例中提供了几种常见的备份方式，您可以按需选择，也可以自定义备份方式。

----结束

10.4.6 应用迁移

第三方云集群中应用的迁移包含两个步骤：应用备份和应用迁移，即备份第三方云集群中应用，然后通过数据恢复的方式迁移至目标集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群（UCS华为云集群或本地集群）中，从而实现本地IDC集群应用的迁移上云。

约束限制

当前不支持高版本集群应用向低版本集群迁移。

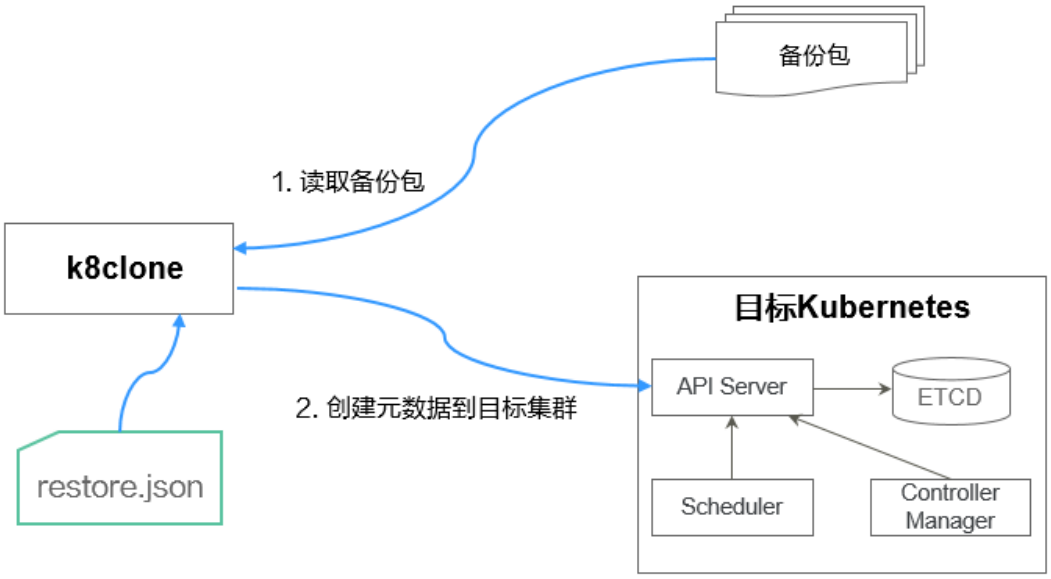
前提条件

- 确认云原生应用依赖的服务（镜像、存储、数据库等非集群内的数据）都已经迁移完成。
- 确认源集群中元数据备份数据已经下载到执行k8clone的服务器上。

k8clone 数据恢复原理

数据恢复的流程参考如下：

图 10-10 数据恢复流程



在执行恢复操作前，需要准备一个数据恢复配置文件“restore.json”，目的是在应用恢复时自动更换PVC、StatefulSet的存储类名称，以及工作负载所使用镜像的Repository地址。

文件内容如下：

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  //支持修改PVC、StatefulSet的StorageClassName字段
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- StorageClass：支持PVC、有状态应用VolumeClaimTemplates中存储类名称按照配置进行自动更换。
- ImageRepo：支持工作负载所使用镜像的Repository地址的更换，工作负载包括Deployment（含initContainer）、StatefulSet、Orphaned Pod、Job、CronJob、Replica Set、Replication Controller、DaemonSet。

k8clone 恢复使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 restore -h**，可以查看k8clone工具恢复的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件

中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。

- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -f, --restore-conf：指定restore.json的路径，默认是k8clone工具所在目录下。
- -d, --local-dir：备份数据放置的路径，默认是k8clone工具所在目录下。

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
    k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

应用恢复操作步骤

步骤1 通过kubectl连接目标集群。具体方法可参考**使用kubectl连接集群**。

步骤2 准备数据恢复配置文件：restore.json。

新建一个restore.json文件，按照格式修改，并将文件放置在k8clone工具所在目录下。

示例：

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

步骤3 进入k8clone工具所在目录，执行恢复命令，将备份数据恢复到目标集群。

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

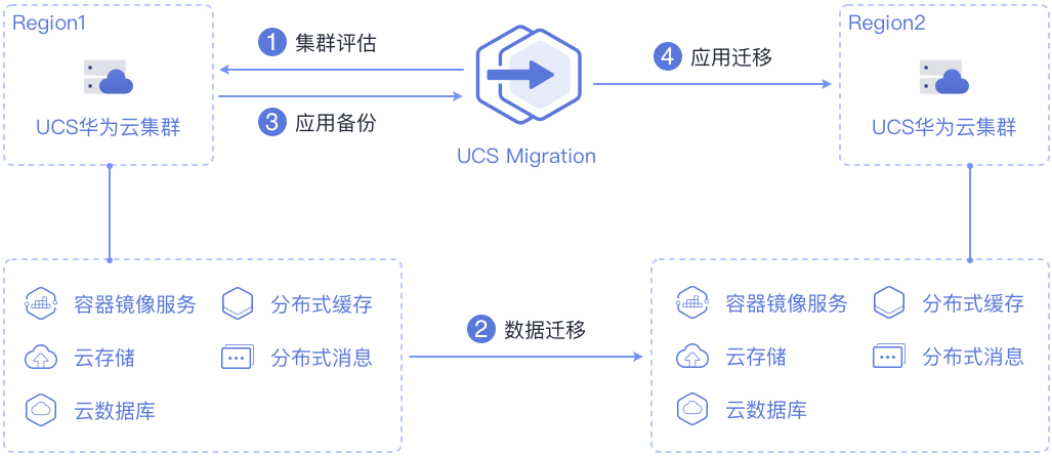
----结束

10.5 不同 Region UCS 华为云集群迁移

10.5.1 不同 Region UCS 华为云集群迁移流程

在华为云UCS管理的Kubernetes集群间进行迁移，将应用程序从一个地理区域迁移到另一个地理区域，以满足数据合规性、延迟和可用性等需求。

图 10-11 迁移流程



主要包含四个步骤：

步骤1 集群评估

在这个阶段，您将根据源集群的现状来评估适合迁移的目标集群类型。UCS的kspider工具能够自动收集源集群的信息，包括Kubernetes版本、规模、工作负载、存储等数据，并根据收集到的数据为您提供推荐的目标集群信息。具体请参见**集群评估**。

步骤2 数据迁移

在这个阶段，您将把镜像和相关依赖服务的数据迁移到目标Region。镜像的跨区域迁移可以使用容器镜像服务 SWR的“镜像同步”功能。

对于依赖服务的数据迁移，您可以查看华为云对应云产品的跨区域迁移指导来完成。具体请参见**数据迁移**。

步骤3 应用备份

在这个阶段，您将对源Region集群中的应用进行备份。UCS的k8clone工具可以自动收集Kubernetes元数据，并将其以压缩包的形式保存到本地，从而实现集群中应用的备份。具体请参见**应用备份**。

步骤4 应用迁移

在这个阶段，您将利用备份数据恢复的方法，将源Region集群中的应用迁移到目标Region集群。具体请参见**应用迁移**。

----结束

10.5.2 集群评估

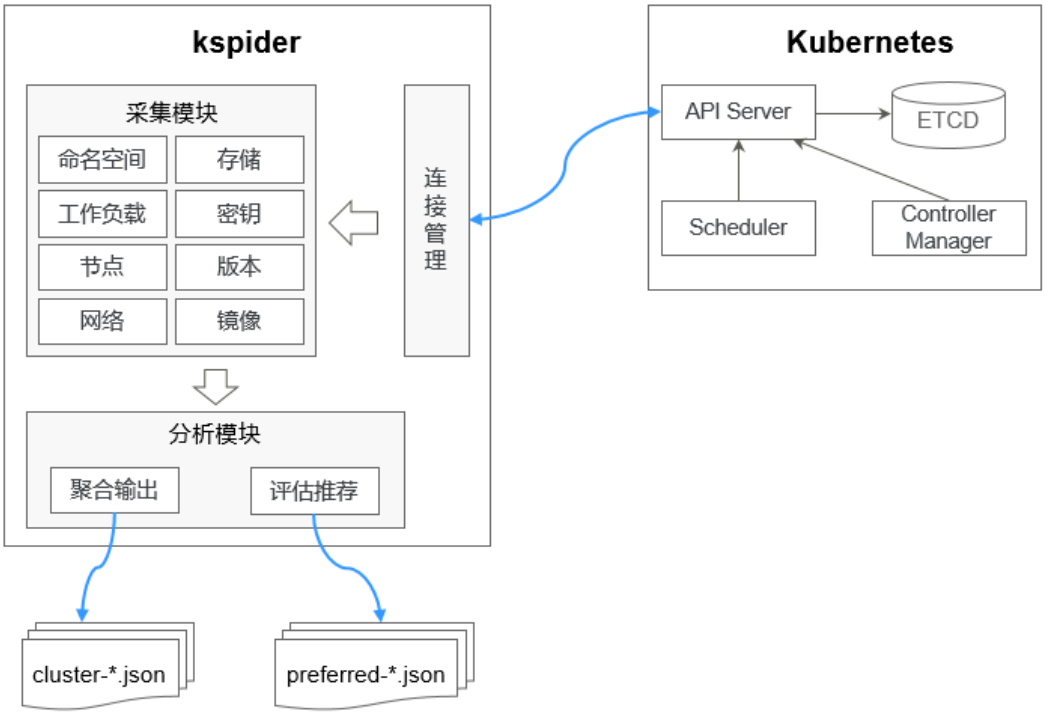
将应用从一个环境迁移到另一个环境是一项具有挑战性的任务，因此您需要进行仔细的规划和准备。kspider是一款用于采集源集群信息的工具，它向用户提供了集群的Kubernetes版本、规模、工作负载数量、存储以及正在使用的镜像等数据，这些信息有助于用户了解集群的当前状况，评估迁移风险，并选择合适的目标集群版本和规模。

kspider 工作原理

kspider工具的架构如**图10-12**所示，包含三个模块：采集模块、连接管理和分析模块。采集模块可以收集源集群的数据，包括命名空间、工作负载、节点、网络等；连

接管理模块负责与源集群的API Server建立连接；分析模块分为聚合输出和评估推荐两部分，旨在输出源集群的采集数据（生成“cluster-*.json”文件）以及提供目标集群的推荐信息（生成“preferred-*.json”文件）。

图 10-12 kspider 架构



kspider 使用方法

说明

kspider工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**kspider-linux-amd64**分别替换为**kspider-linux-arm64**或**kspider-windows-amd64.exe**。

根据**容器迁移准备工作**章节的要求，准备一台服务器并上传kspider工具，然后进行解压缩。在kspider工具所在目录下执行**./kspider-linux-amd64 -h**，您可以查看该工具的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -n, --namespaces：指定采集的命名空间，默认排除了kube-system、kube-public、kube-node-lease等系统命名空间。
- -q, --quiet：静态退出。
- -s, --serial：根据采集信息输出汇聚文件（cluster-{serial}.json）和推荐文件（preferred-{serial}.json）唯一标识的序号。

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
```



```
  kspider [flags]

Aliases:
  kspider, kspider

Flags:
  -h, --help              help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet             command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

步骤一：源集群数据采集

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 使用默认参数配置，采集集群中所有命名空间的数据。执行方法：**./kspider-linux-amd64**

执行后的输出详细信息如下：

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
   Name     CPU     Memory     IP     Arch     OS     Kernel     MachineID
   10.1.18.64     4     8008284Ki     [10.1.18.64 10.1.18.64]     amd64     linux
3.10.0-1127.19.1.el7.x86_64     ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
   10.1.19.13     4     8008284Ki     [10.1.19.13 10.1.19.13]     amd64     linux
3.10.0-1127.19.1.el7.x86_64     2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
   Name     Namespace     NodeAffinity
   minio     default     false
   minio     minio     false
There are 3 Deployments:
   Name     Namespace     NodeAffinity
   rctest     default     true
   flink-operator-controller-manager     flink-operator-system     false
   rctest     minio     false
There are 1 DaemonSets:
   Name     Namespace     NodeAffinity
   ds-nginx     minio     false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
   Namespace/Name     Pods
   default/pvc-data-minio-0     default/minio-0
   minio/obs-testing     minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
   minio/pvc-data-minio-0     minio/minio-0
There are 5 PersistentVolumes:
   Name     Namespace     pvcName     scName     size     key
   pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8     minio-pv     obs-testing     csi-obs     1Gi     pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
   pvc-c7c768aa-373a-4c52-abea-e8b486d23b47     minio-pv     pvc-data-minio-0     csi-disk-sata     10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
   pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05     minio     obs-testing     csi-obs     1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
   pvc-9fd92c99-805a-4e65-9f22-e238130983c8     default     pvc-data-minio-0     csi-disk     10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
   pvc-a22fd877-f98d-4c3d-a04e-191d79883f97     minio     pvc-data-minio-0     csi-disk-sata     10Gi
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
   Name     Namespace     ServiceType
   headless-lxprus     default     ClusterIP
```



```
  kubernetes    default    ClusterIP
  minio    default    NodePort
  flink-operator-controller-manager-metrics-service    flink-operator-system    ClusterIP
  flink-operator-webhook-service    flink-operator-system    ClusterIP
  headless-lxprus    minio    ClusterIP
  minio    minio    NodePort
There are 0 Ingresses:
There are 6 Images:
  Name
  gcr.io/flink-operator/flink-operator:v1beta1-6
  flink:1.8.2
  swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
  nginx:stable-alpine-perl
  swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
  gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
  SecretType
  cfe/secure-opaque
  helm.sh/release.v1
```

在kspider执行完毕后，当前目录下将生成两个文件：

- cluster-*.json：此文件包含了源集群及应用的采集数据，这些数据可用于分析和规划迁移过程。
- preferred-*.json：此文件包含了推荐的目标集群信息。基于源集群的规模和节点规格进行初步评估，文件将提供关于目标集群版本和规模的建议。

步骤3 查看源集群及应用的采集数据。

您可以用文本编辑器或JSON查看器打开“cluster-*.json”文件以查看数据。在实际操作中，您需要将文件名中的“*”替换为实际的时间戳或序列号，以找到并打开正确的文件。

“cluster-*.json”文件说明如下：

```
{
 K8sVersion：Kubernetes版本，字符串类型
 Namespaces：命名空间数量，字符串类型
 Pods：Pod总数量，整型
 Nodes：节点总信息，以IP为key，展示节点信息
  IP地址
   CPU：CPU，字符串类型
   Arch：CPU架构，字符串类型
   Memory：内存，字符串类型
   HugePages1Gi：1G大页内存，字符串类型
   HugePages2Mi：2M大页内存，字符串类型
   OS：节点OS，字符串类型
   KernelVersion：OS内核版本，字符串类型
   RuntimeVersion：节点容器运行及版本，字符串类型
   InternalIP：内部IP，字符串类型
   ExternalIP：外部IP，字符串类型
   MachineID：节点ID，字符串类型。说明：CCE中能够保证与ECS的ID一致
 Workloads：工作负载
  Deployment：工作负载类型，支持Deployment（无状态负载）、StatefulSet（有状态负载）、DaemonSet（守护进程集）、CronJob（定时任务）、Job（普通任务）、LonePod（独立Pod）
   default：命名空间名称
    Count：数量，整型
    Items：详细信息，数组类型
     Name：工作负载名称，字符串类型
     Namespace：命名空间名称，字符串类型
     NodeAffinity：节点亲和性，布尔型
     Replicas：副本数量，整型
 Storage：存储
  PersistentVolumes：持久卷
   pv-name：以PV名称为key
    VolumeID：卷ID，字符串类型
    Namespace：命名空间，字符串类型
```



```
      PvcName：绑定PVC的名称，字符串类型
      ScName：存储类的名称，字符串类型
      Size：申请空间大小，字符串类型
      Pods：使用PV的Pod名称，字符串类型
      NodeIP：Pod所在的节点IP，字符串类型
      VolumePath：该Pod挂载节点的路径，字符串类型
   OtherVolumes：其它类型卷
    类型：AzureFile、AzureDisk、GCEPersistentDisk、AWSElasticBlockStore、Cinder、Glusterfs、NFS、
CephFS、FlexVolume、FlexVolume、DownwardAPI
      卷ID/卷名称/卷共享路径等为key
        Pods：使用其的Pod，字符串类型
        NodeIP：Pod所在的节点IP，字符串类型
        卷ID/卷名称/卷共享路径等唯一标识卷信息的信息，字符串类型
 Networks：网络
  LoadBalancer：负载均衡类型
    service：网络类型，包括service和ingress
      Name：名称，字符串类型
      Namespace：命名空间名称，字符串类型
      Type：类型，字符串类型
 ExtraSecrets：扩展secret类型
  secret类型名，字符串类型
 Images：镜像
  镜像repo，字符串类型
}
```

示例：

```
{
  "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
  "Namespaces": 12,
  "Pods": 33,
  "Nodes": {
    "10.1.17.219": {
      "CPU": "4",
      "Memory": "7622944Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP9x86_64)",
      "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.17.219",
      "ExternalIP": "",
      "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
    },
    "10.1.18.182": {
      "CPU": "4",
      "Memory": "7992628Ki",
      "HugePages1Gi": "0",
      "HugePages2Mi": "0",
      "Arch": "amd64",
      "OS": "EulerOS 2.0 (SP5)",
      "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
      "RuntimeVersion": "docker://18.9.0",
      "InternalIP": "10.1.18.182",
      "ExternalIP": "100.85.xxx.xxx",
      "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
    }
  },
  "Workloads": {
    "CronJob": {},
    "DaemonSet": {
      "default": {
        "Count": 1,
        "Items": [
          {
            "Name": "kubecost-prometheus-node-exporter",
            "Namespace": "default",
            "NodeAffinity": false,
            "Replicas": 3
```



```
        }
       ]
     }
   },
   "Deployment": {
     "default": {
       "Count": 1,
       "Items": [
         {
           "Name": "kubecost-cost-analyzer",
           "Namespace": "default",
           "NodeAffinity": false,
           "Replicas": 1
         }
       ]
     },
     "kubecost": {
       "Count": 1,
       "Items": [
         {
           "Name": "kubecost-kube-state-metrics",
           "Namespace": "kubecost",
           "NodeAffinity": false,
           "Replicas": 1
         }
       ]
     }
   },
   "Job": {},
   "LonePod": {},
   "StatefulSet": {
     "minio-all": {
       "Count": 1,
       "Items": [
         {
           "Name": "minio",
           "Namespace": "minio-all",
           "NodeAffinity": false,
           "Replicas": 1
         }
       ]
     }
   }
 },
 "Storage": {
   "PersistentVolumes": {
     "demo": {
       "VolumeID": "demo",
       "Namespace": "fluid-demo-test",
       "PvcName": "demo",
       "ScName": "fluid",
       "Size": "100Gi",
       "Pods": "",
       "NodeIP": "",
       "VolumePath": ""
     },
     "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
       "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
       "Namespace": "minio-all",
       "PvcName": "pvc-data-minio-0",
       "ScName": "csi-disk",
       "Size": "10Gi",
       "Pods": "minio-all/minio-0",
       "NodeIP": "10.1.23.159",
       "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
     }
   },
   "OtherVolumes": {}
```



```
 },
 "Networks": {
   "LoadBalancer": {}
 },
 "ExtraSecrets": [
   "cfe/secure-opaque",
   "helm.sh/release.v1"
 ],
 "Images": [
   "nginx:stable-alpine-perl",
   "ghcr.io/koordinator-sh/koord-manager:0.6.2",
   "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
   "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
   "gcr.io/kubecost1/cost-model:prod-1.91.0",
   "gcr.io/kubecost1/frontend:prod-1.91.0"
 ]
}
```

----结束

步骤二：目标集群评估

在kspider执行完毕后，除了“cluster-*.json”文件之外，还会在当前目录下生成“preferred-*.json”文件。这个文件基于源集群的规模和节点规格进行初步评估，并提供关于目标集群版本和规模的推荐信息。这有助于您更好地规划和准备迁移过程。

“preferred-*.json”文件说明如下：

```
{
 K8sVersion：Kubernetes版本，字符串类型
 Scale：集群规模，字符串类型
 Nodes：节点信息
   CPU：CPU，字符串类型
   Memory：内存，字符串类型
   Arch：架构，字符串类型
   KernelVersion：OS内核版本，字符串类型
   ProxyMode：集群Proxy模式，字符串类型
 ELB：是否依赖ELB，布尔型
}
```

上述文件中每个字段的评估规则如下：

表 10-6 评估规则

字段	评估规则
Kubernetes版本	如果是1.21以下版本，推荐UCS集群主要发行版本（例如1.21，随着时间发展会发生变化），大于主要发行版本时，将推荐UCS集群的最新版本。
集群规模	源集群节点数＜25，推荐50节点规模 25≤源集群节点数＜100，推荐200节点规模 100≤源集群节点数＜500，推荐1000节点规模 源集群节点数≥500，推荐2000节点规模
CPU+内存	统计数量最多的那一种规格
架构	统计数量最多的那一种规格
OS内核版本	统计数量最多的那一种规格

字段	评估规则
集群Proxy模式	根据集群规模来设置，大于1000节点规模的集群，推荐使用ipvs，1000以内的推荐使用iptables。
是否依赖ELB	源集群是否有负载均衡类型的Service

示例：

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

⚠ 注意

评估结果仅供参考，最终选择什么版本、规模的目标集群还需要您综合判断。

10.5.3 数据迁移

本节介绍镜像及集群依赖服务的相关数据迁移，如云存储、云数据库、分布式缓存、分布式消息等。

镜像迁移

镜像的跨区域迁移可以使用容器镜像服务 SWR的“镜像同步”功能。

对于镜像仓库中已有的镜像，您需要执行手动镜像同步，将镜像同步到目标区域。另外，为镜像设置镜像自动同步功能，可以帮助您把最新推送的镜像自动同步到其他区域镜像仓库内，后期镜像有更新时，目标仓库的镜像也会自动更新。

具体请参见**自动同步镜像**。

云存储迁移

若您的集群使用了云硬盘或文件存储，跨区域迁移可以使用**云备份 CBR**。CBR为云上的弹性云服务器、裸金属服务器、云硬盘、文件存储、云下VMware虚拟化环境和本地文件目录，提供简单易用的备份服务，当发生病毒入侵、人为误删除等事件时，可将数据恢复到任意备份点。

具体请参见**创建云硬盘备份创建云硬盘备份**或**创建SFS Turbo备份**。

云数据库迁移

云数据库的跨区域迁移可以使用**数据复制服务 DRS**。DRS服务具有实时迁移、备份迁移、实时同步、数据订阅和实时灾备等多种功能。

其他数据迁移

- 大数据场景迁移：推荐使用**云数据迁移 CDM**。
- Kafka业务迁移：具体请参见分布式消息服务Kafka版的**Kafka业务迁移**。
- Redis业务迁移：具体请参见分布式缓存服务 DCS的**数据迁移指南**。

10.5.4 应用备份

不同Region UCS华为云集群应用的迁移包含两个步骤：应用备份和应用迁移，即备份源Region集群中应用，然后通过数据恢复的方式迁移至目标Region集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群中。

须知

建议在用户业务量小时执行备份操作。

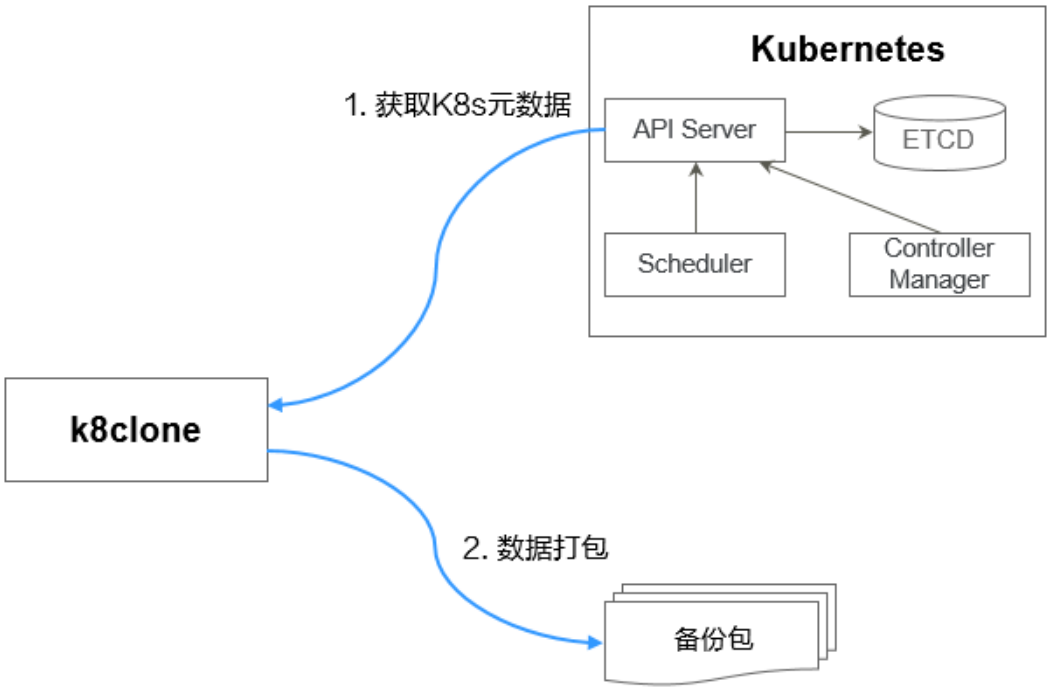
前提条件

确认云原生应用依赖的服务（镜像、存储、数据库等非集群内的数据）都已经迁移完成。

k8clone 数据备份原理

数据备份的流程参考如下：

图 10-13 数据备份流程



k8clone 备份使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 backup -h**，可以查看k8clone工具备份的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见Kubernetes文档。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -n, --namespace：备份指定命名空间的云原生应用，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示备份整个集群。
- -e, --exclude-namespaces：排除指定命名空间对象的备份，不能和--namespace一起使用。
- -x, --exclude-kind：排除指定资源类型的备份。
- -i, --include-kind : 指定资源类型的备份。
- -y, --exclude-object：排除指定资源对象的备份。
- -z, --include-object：指定资源对象的备份。
- -w, --exclude-having-owner-ref：排除拥有ownerReferences资源对象的备份，默认是false。传参过程中需要带上等号，如：-w=true 。
- -d, --local-dir：备份数据放置的路径，默认是当前目录下k8clone-dump文件夹。

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
   k8clone backup [flags]

Flags:
   -s, --api-server string           Kubernetes api-server url
   -q, --context string              Kubernetes configuration context
   -w, --exclude-having-owner-ref    Exclude all objects having an Owner Reference. The following form is not permitted for boolean flags such as '-w false', please use '-w=false'
   -x, --exclude-kind strings        Ressource kind to exclude. Eg. 'deployment'
   -i, --include-kind strings        Ressource kind to include. Eg. 'deployment'
   -e, --exclude-namespaces strings  Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all namespaces and then filters them. Don't use it with the namespace flag.
   -y, --exclude-object strings      Object to exclude. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -z, --include-object strings      Object to include. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -h, --help                        help for backup
   -k, --kubeconfig string           The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
   -d, --local-dir string            Where to dump yaml files (default "./k8clone-dump")
   -n, --namespace string            Only dump objects from this namespace
```

示例如下：

- 整个集群对象的备份，默认路径为当前目录下“k8clone-dump”文件夹

./k8clone-linux-amd64 backup

- 整个集群对象的备份，并指定备份数据路径

  ./k8clone-linux-amd64 backup -d ./xxxx

- 指定命名空间对象的备份

  ./k8clone-linux-amd64 backup -n default

- 排除命名空间对象的备份

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease

- 排除指定资源类型的备份

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice

- 指定资源类型的备份

  ./k8clone-linux-amd64 backup -x rolebinding

- 排除指定资源对象的备份

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns

- 指定资源对象的备份

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns

- 排除拥有ownerReferences资源对象的备份

  ./k8clone-linux-amd64 backup -w=true

应用备份操作步骤

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 进入k8clone工具所在目录，执行备份命令，即可备份数据到本地目录，并打包成压缩包。

k8clone备份使用方法的示例中提供了几种常见的备份方式，您可以按需选择，也可以自定义备份方式。

----结束

10.5.5 应用迁移

不同Region UCS华为云集群应用的迁移包含两个步骤：应用备份和应用迁移，即备份源Region集群中应用，然后通过数据恢复的方式迁移至目标Region集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群中。

约束限制

当前不支持高版本集群应用向低版本集群迁移。

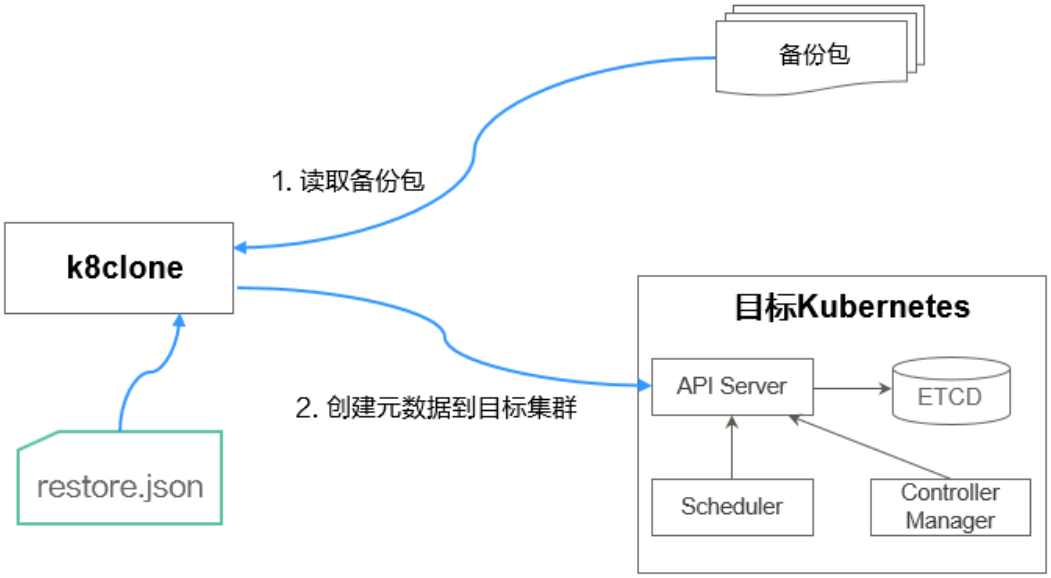
前提条件

- 确认云原生应用依赖的服务（镜像、存储、数据库等非集群内的数据）都已经迁移完成。
- 确认源集群中元数据备份数据已经下载到执行k8clone的服务器上。

k8clone 数据恢复原理

数据恢复的流程参考如下：

图 10-14 数据恢复流程



在执行恢复操作前，需要准备一个数据恢复配置文件“restore.json”，目的是在应用恢复时自动更换PVC、StatefulSet的存储类名称，以及工作负载所使用镜像的Repository地址。

文件内容如下：

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  //支持修改PVC、StatefulSet的StorageClassName字段
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- StorageClass：支持PVC、有状态应用VolumeClaimTemplates中存储类名称按照配置进行自动更换。
- ImageRepo：支持工作负载所使用镜像的Repository地址的更换，工作负载包括Deployment（含initContainer）、StatefulSet、Orphaned Pod、Job、CronJob、Replica Set、Replication Controller、DaemonSet。

k8clone 恢复使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 restore -h**，可以查看k8clone工具恢复的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -f, --restore-conf：指定restore.json的路径，默认是k8clone工具所在目录下。
- -d, --local-dir：备份数据放置的路径，默认是k8clone工具所在目录下。

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
    k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

应用恢复操作步骤

步骤1 通过kubectl连接目标集群。具体方法可参考**使用kubectl连接集群**。

步骤2 准备数据恢复配置文件：restore.json。

新建一个restore.json文件，按照格式修改，并将文件放置在k8clone工具所在目录下。

示例：

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

步骤3 进入k8clone工具所在目录，执行恢复命令，将备份数据恢复到目标集群。

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

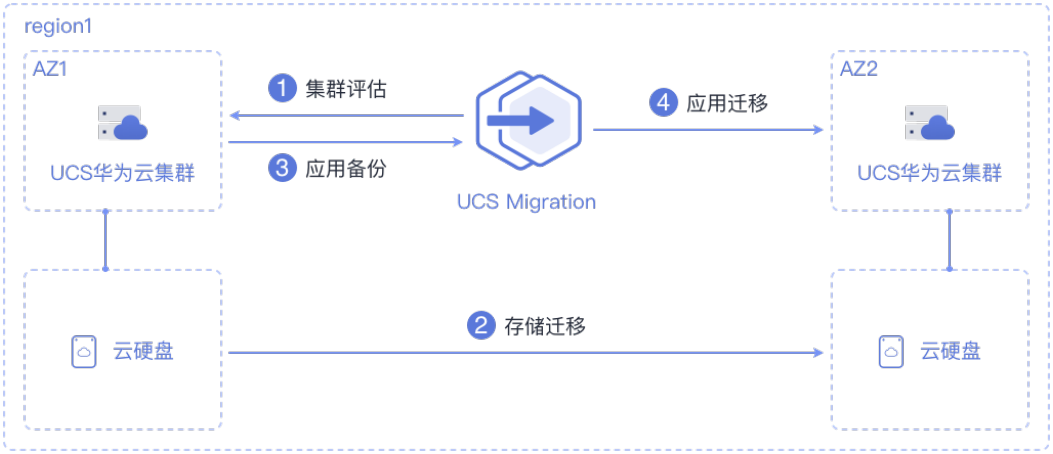
----结束

10.6 同 Region UCS 华为云集群迁移

10.6.1 同 Region UCS 华为云集群迁移流程

在同一地理区域内的华为云UCS管理的Kubernetes集群间进行迁移，实现资源优化、应用升级或其他管理需求。迁移流程如图10-15所示。

图 10-15 迁移流程



主要包含四个步骤：

步骤1 集群评估

在这个阶段，您将根据源集群的现状来评估适合迁移的目标集群类型。UCS的kspider工具能够自动收集源集群的信息，包括Kubernetes版本、规模、工作负载、存储等数据，并根据收集到的数据为您提供推荐的目标集群信息。具体请参见**集群评估**。

步骤2 存储迁移

在这个阶段，您将把云硬盘的数据迁移到目标AZ。具体请参见**存储迁移**。

步骤3 应用备份

在这个阶段，您将对源AZ集群中的应用进行备份。UCS的k8clone工具可以自动收集Kubernetes元数据，并将其以压缩包的形式保存到本地，从而实现集群中应用的备份。具体请参见**应用备份**。

步骤4 应用迁移

在这个阶段，您将利用备份数据恢复的方法，将源AZ集群中的应用迁移到目标AZ集群。具体请参见**应用迁移**。

----结束

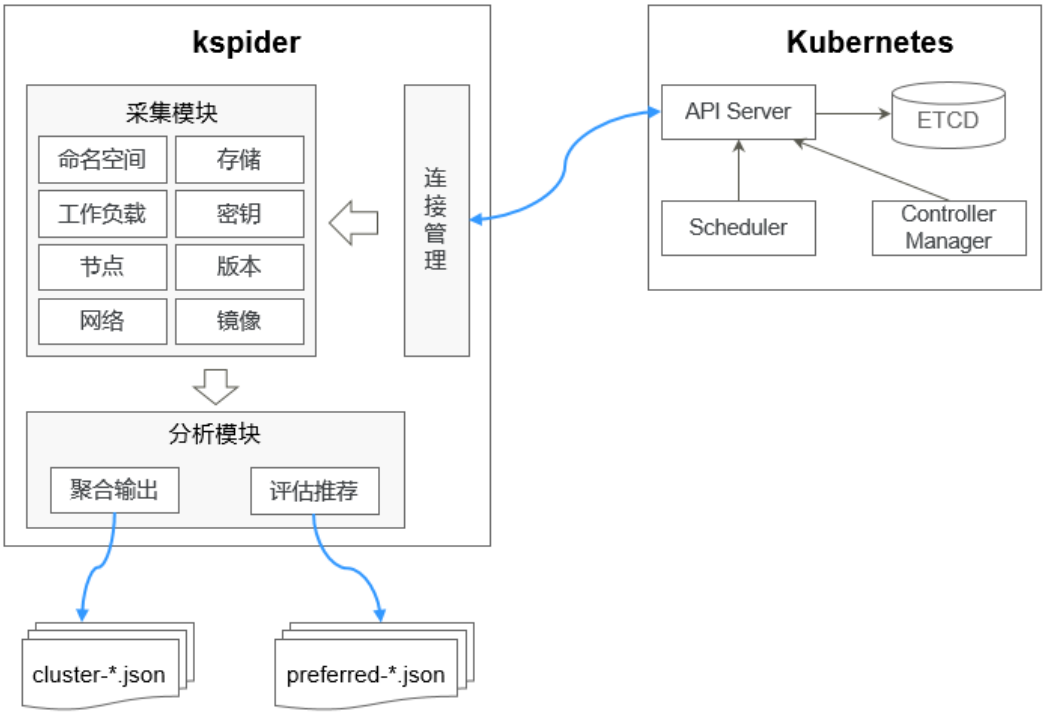
10.6.2 集群评估

将应用从一个环境迁移到另一个环境是一项具有挑战性的任务，因此您需要进行仔细的规划和准备。kspider是一款用于采集源集群信息的工具，它向用户提供了集群的Kubernetes版本、规模、工作负载数量、存储以及正在使用的镜像等数据，这些信息有助于用户了解集群的当前状况，评估迁移风险，并选择合适的目标集群版本和规模。

kspider 工作原理

kspider工具的架构如图10-16所示，包含三个模块：采集模块、连接管理和分析模块。采集模块可以收集源集群的数据，包括命名空间、工作负载、节点、网络等；连接管理模块负责与源集群的API Server建立连接；分析模块分为聚合输出和评估推荐两部分，旨在输出源集群的采集数据（生成“cluster-*.json”文件）以及提供目标集群的推荐信息（生成“preferred-*.json”文件）。

图 10-16 kspider 架构



kspider 使用方法

说明

kspider工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**kspider-linux-amd64**分别替换为**kspider-linux-arm64**或**kspider-windows-amd64.exe**。

根据**容器迁移准备工作**章节的要求，准备一台服务器并上传kspider工具，然后进行解压缩。在kspider工具所在目录下执行**./kspider-linux-amd64 -h**，您可以查看该工具的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -n, --namespaces：指定采集的命名空间，默认排除了kube-system、kube-public、kube-node-lease等系统命名空间。
- -q, --quiet：静态退出。
- -s, --serial：根据采集信息输出汇聚文件（cluster-{serial}.json）和推荐文件（preferred-{serial}.json）唯一标识的序号。

```
$ ./kspider-linux-amd64 -h
A cluster information collection and recommendation tool implement by Go.

Usage:
  kspider [flags]

Aliases:
  kspider, kspider

Flags:
  -h, --help               help for kspider
  -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default
"$HOME/.kube/config")
  -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are
specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
  -q, --quiet              command to execute silently
  -s, --serial string       User-defined sequence number of the execution. The default value is the time when
the kspider is started. (default "1673853404")
```

步骤一：源集群数据采集

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 使用默认参数配置，采集集群中所有命名空间的数据。执行方法：**./kspider-linux-amd64**

执行后的输出详细信息如下：

```
[~]# ./kspider-linux-amd64
The Cluster version is v1.15.6-r1-CCE2.0.30.B001
There are 5 Namespaces
There are 2 Nodes
   Name     CPU     Memory     IP     Arch     OS     Kernel     MachineID
   10.1.18.64     4     8008284Ki     [10.1.18.64 10.1.18.64]     amd64     linux
3.10.0-1127.19.1.el7.x86_64     ef9270ed-7eb3-4ce6-a2d8-f1450f85489a
   10.1.19.13     4     8008284Ki     [10.1.19.13 10.1.19.13]     amd64     linux
3.10.0-1127.19.1.el7.x86_64     2d889590-9a32-47e5-b947-09c5bda81849
There are 9 Pods
There are 0 LonePods:
There are 2 StatefulSets:
   Name     Namespace     NodeAffinity
   minio     default     false
   minio     minio     false
There are 3 Deployments:
   Name     Namespace     NodeAffinity
   rctest     default     true
   flink-operator-controller-manager     flink-operator-system     false
   rctest     minio     false
There are 1 DaemonSets:
   Name     Namespace     NodeAffinity
   ds-nginx     minio     false
There are 0 Jobs:
There are 0 CronJobs:
There are 4 PersistentVolumeClaims:
   Namespace/Name     Pods
   default/pvc-data-minio-0     default/minio-0
   minio/obs-testing     minio/ds-nginx-9hmds,minio/ds-nginx-4jsfg
   minio/pvc-data-minio-0     minio/minio-0
There are 5 PersistentVolumes:
   Name     Namespace     pvcName     scName     size     key
   pvc-bd36c70f-75bf-4000-b85c-f9fb169a14a8     minio-pv     obs-testing     csi-obs     1Gi     pvc-
bd36c70f-75bf-4000-b85c-f9fb169a14a8
   pvc-c7c768aa-373a-4c52-abea-e8b486d23b47     minio-pv     pvc-data-minio-0     csi-disk-sata     10Gi
1bcf3d00-a524-45b1-a773-7efbca58f36a
   pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05     minio     obs-testing     csi-obs     1Gi
pvc-4f52462b-3b4c-4191-a63b-5a36a8748c05
   pvc-9fd92c99-805a-4e65-9f22-e238130983c8     default     pvc-data-minio-0     csi-disk     10Gi
590afd05-fc68-4c10-a598-877100ca7b3f
   pvc-a22fd877-f98d-4c3d-a04e-191d79883f97     minio     pvc-data-minio-0     csi-disk-sata     10Gi
```



```
48874130-df77-451b-9b43-d435ac5a11d5
There are 7 Services:
  Name  Namespace  ServiceType
  headless-lxprus  default  ClusterIP
  kubernetes  default  ClusterIP
  minio  default  NodePort
  flink-operator-controller-manager-metrics-service  flink-operator-system  ClusterIP
  flink-operator-webhook-service  flink-operator-system  ClusterIP
  headless-lxprus  minio  ClusterIP
  minio  minio  NodePort
There are 0 Ingresses:
There are 6 Images:
  Name
  gcr.io/flink-operator/flink-operator:v1beta1-6
  flink:1.8.2
  swr.cn-north-4.myhuaweicloud.com/paas/minio:latest
  nginx:stable-alpine-perl
  swr.cn-north-4.myhuaweicloud.com/everest/minio:latest
  gcr.io/kubebuilder/kube-rbac-proxy:v0.4.0
There are 2 Extra Secrets:
  SecretType
  cfe/secure-opaque
  helm.sh/release.v1
```

在kspider执行完毕后，当前目录下将生成两个文件：

- cluster-*.json：此文件包含了源集群及应用的采集数据，这些数据可用于分析和规划迁移过程。
- preferred-*.json：此文件包含了推荐的目标集群信息。基于源集群的规模和节点规格进行初步评估，文件将提供关于目标集群版本和规模的建议。

步骤3 查看源集群及应用的采集数据。

您可以用文本编辑器或JSON查看器打开“cluster-*.json”文件以查看数据。在实际操作中，您需要将文件名中的“*”替换为实际的时间戳或序列号，以找到并打开正确的文件。

“cluster-*.json”文件说明如下：

```
{
 K8sVersion：Kubernetes版本，字符串类型
 Namespaces：命名空间数量，字符串类型
 Pods：Pod总数量，整型
 Nodes：节点总信息，以IP为key，展示节点信息
  IP地址
    CPU：CPU，字符串类型
    Arch：CPU架构，字符串类型
    Memory：内存，字符串类型
    HugePages1Gi：1G大页内存，字符串类型
    HugePages2Mi：2M大页内存，字符串类型
    OS：节点OS，字符串类型
    KernelVersion：OS内核版本，字符串类型
    RuntimeVersion：节点容器运行及版本，字符串类型
    InternalIP：内部IP，字符串类型
    ExternalIP：外部IP，字符串类型
    MachineID：节点ID，字符串类型。说明：CCE中能够保证与ECS的ID一致
 Workloads：工作负载
   Deployment：工作负载类型，支持Deployment（无状态负载）、StatefulSet（有状态负载）、DaemonSet（守护进程集）、CronJob（定时任务）、Job（普通任务）、LonePod（独立Pod）
    default：命名空间名称
      Count：数量，整型
      Items：详细信息，数组类型
        Name：工作负载名称，字符串类型
        Namespace：命名空间名称，字符串类型
        NodeAffinity：节点亲和性，布尔型
        Replicas：副本数量，整型
 Storage：存储
```



```
  PersistentVolumes：持久卷
    pv-name：以PV名称为key
      VolumeID：卷ID，字符串类型
      Namespace：命名空间，字符串类型
      PvcName：绑定PVC的名称，字符串类型
      ScName：存储类的名称，字符串类型
      Size：申请空间大小，字符串类型
      Pods：使用PV的Pod名称，字符串类型
      NodeIP：Pod所在的节点IP，字符串类型
      VolumePath：该Pod挂载节点的路径，字符串类型
  OtherVolumes：其它类型卷
    类型：AzureFile、AzureDisk、GCEPersistentDisk、AWSElasticBlockStore、Cinder、Glusterfs、NFS、
CephFS、FlexVolume、FlexVolume、DownwardAPI
      卷ID/卷名称/卷共享路径等为key
        Pods：使用其的Pod，字符串类型
        NodeIP：Pod所在的节点IP，字符串类型
        卷ID/卷名称/卷共享路径等唯一标识卷信息的信息，字符串类型
 Networks：网络
  LoadBalancer：负载均衡类型
    service：网络类型，包括service和ingress
      Name：名称，字符串类型
      Namespace：命名空间名称，字符串类型
      Type：类型，字符串类型
 ExtraSecrets：扩展secret类型
  secret类型名，字符串类型
 Images：镜像
  镜像repo，字符串类型
}
```

示例：

```
{
 "K8sVersion": "v1.19.10-r0-CCE22.3.1.B009",
 "Namespaces": 12,
 "Pods": 33,
 "Nodes": {
  "10.1.17.219": {
   "CPU": "4",
   "Memory": "7622944Ki",
   "HugePages1Gi": "0",
   "HugePages2Mi": "0",
   "Arch": "amd64",
   "OS": "EulerOS 2.0 (SP9x86_64)",
   "KernelVersion": "4.18.0-147.5.1.6.h687.eulerosv2r9.x86_64",
   "RuntimeVersion": "docker://18.9.0",
   "InternalIP": "10.1.17.219",
   "ExternalIP": "",
   "MachineID": "0c745e03-2802-44c2-8977-0a9fd081a5ba"
  },
  "10.1.18.182": {
   "CPU": "4",
   "Memory": "7992628Ki",
   "HugePages1Gi": "0",
   "HugePages2Mi": "0",
   "Arch": "amd64",
   "OS": "EulerOS 2.0 (SP5)",
   "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64",
   "RuntimeVersion": "docker://18.9.0",
   "InternalIP": "10.1.18.182",
   "ExternalIP": "100.85.xxx.xxx",
   "MachineID": "2bff3d15-b565-496a-817c-063a37eaf1bf"
  }
 },
 "Workloads": {
  "CronJob": {},
  "DaemonSet": {
   "default": {
    "Count": 1,
    "Items": [
     {
```



```
          "Name": "kubecost-prometheus-node-exporter",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 3
        }
      ]
    }
  },
  "Deployment": {
    "default": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-cost-analyzer",
          "Namespace": "default",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    },
    "kubecost": {
      "Count": 1,
      "Items": [
        {
          "Name": "kubecost-kube-state-metrics",
          "Namespace": "kubecost",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  },
  "Job": {},
  "LonePod": {},
  "StatefulSet": {
    "minio-all": {
      "Count": 1,
      "Items": [
        {
          "Name": "minio",
          "Namespace": "minio-all",
          "NodeAffinity": false,
          "Replicas": 1
        }
      ]
    }
  }
},
"Storage": {
  "PersistentVolumes": {
    "demo": {
      "VolumeID": "demo",
      "Namespace": "fluid-demo-test",
      "PvcName": "demo",
      "ScName": "fluid",
      "Size": "100Gi",
      "Pods": "",
      "NodeIP": "",
      "VolumePath": ""
    },
    "pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c": {
      "VolumeID": "82365752-89b6-4609-9df0-007d964b7fe4",
      "Namespace": "minio-all",
      "PvcName": "pvc-data-minio-0",
      "ScName": "csi-disk",
      "Size": "10Gi",
      "Pods": "minio-all/minio-0",
      "NodeIP": "10.1.23.159",
      "VolumePath": "/var/lib/kubelet/pods/5fc47c82-7cbd-4643-98cd-cea41de28ff2/volumes/
```



```
kubernetes.io~csi/pvc-fd3a5bb3-119a-44fb-b02e-96b2cf9bb36c/mount"
      }
    },
    "OtherVolumes": {}
  },
  "Networks": {
    "LoadBalancer": {}
  },
  "ExtraSecrets": [
    "cfe/secure-opaque",
    "helm.sh/release.v1"
  ],
  "Images": [
    "nginx:stable-alpine-perl",
    "ghcr.io/koordinator-sh/koord-manager:0.6.2",
    "swr.cn-north-4.myhuaweicloud.com/paas/minio:latest",
    "swr.cn-north-4.myhuaweicloud.com/everest/e-backup-test:v1.0.0",
    "gcr.io/kubecost1/cost-model:prod-1.91.0",
    "gcr.io/kubecost1/frontend:prod-1.91.0"
  ]
}
```


----结束

步骤二：目标集群评估

在kspider执行完毕后，除了“cluster-*.json”文件之外，还会在当前目录下生成“preferred-*.json”文件。这个文件基于源集群的规模和节点规格进行初步评估，并提供关于目标集群版本和规模的推荐信息。这有助于您更好地规划和准备迁移过程。

“preferred-*.json”文件说明如下：

```
{
  K8sVersion：Kubernetes版本，字符串类型
  Scale：集群规模，字符串类型
  Nodes：节点信息
    CPU：CPU，字符串类型
    Memory：内存，字符串类型
    Arch：架构，字符串类型
    KernelVersion：OS内核版本，字符串类型
    ProxyMode：集群Proxy模式，字符串类型
  ELB：是否依赖ELB，布尔型
}
```

上述文件中每个字段的评估规则如下：

表 10-7 评估规则

字段	评估规则
Kubernetes版本	如果是1.21以下版本，推荐UCS集群主要发行版本（例如1.21，随着时间发展会发生变化），大于主要发行版本时，将推荐UCS集群的最新版本。
集群规模	源集群节点数＜25，推荐50节点规模 25≤源集群节点数＜100，推荐200节点规模 100≤源集群节点数＜500，推荐1000节点规模 源集群节点数≥500，推荐2000节点规模
CPU+内存	统计数量最多的那一种规格
架构	统计数量最多的那一种规格

字段	评估规则
OS内核版本	统计数量最多的那一种规格
集群Proxy模式	根据集群规模来设置，大于1000节点规模的集群，推荐使用ipvs，1000以内的推荐使用iptables。
是否依赖ELB	源集群是否有负载均衡类型的Service

示例：

```
{
  "K8sVersion": "v1.21",
  "Scale": 50,
  "Nodes": {
    "CPU": "4",
    "Memory": "7622952Ki",
    "Arch": "amd64",
    "KernelVersion": "3.10.0-862.14.1.5.h520.eulerosv2r7.x86_64"
  },
  "ELB": false,
  "ProxyMode": "iptables"
}
```

⚠ 注意

评估结果仅供参考，最终选择什么版本、规模的目标集群还需要您综合判断。

10.6.3 存储迁移

若您的集群使用了云硬盘，需要随集群一起迁往目标AZ，迁移方法如下：

可以通过云备份服务创建云硬盘备份，再使用备份创建新的云硬盘，在配置云硬盘信息时，选择目标可用区即可。具体操作请参见**创建云硬盘备份**和**使用备份创建新云硬盘**。

10.6.4 应用备份

不同AZ UCS华为云集群应用的迁移包含两个步骤：应用备份和应用迁移，即备份源AZ集群中应用，然后通过数据恢复的方式迁移至目标AZ集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群中。

须知

建议在用户业务量小时执行备份操作。

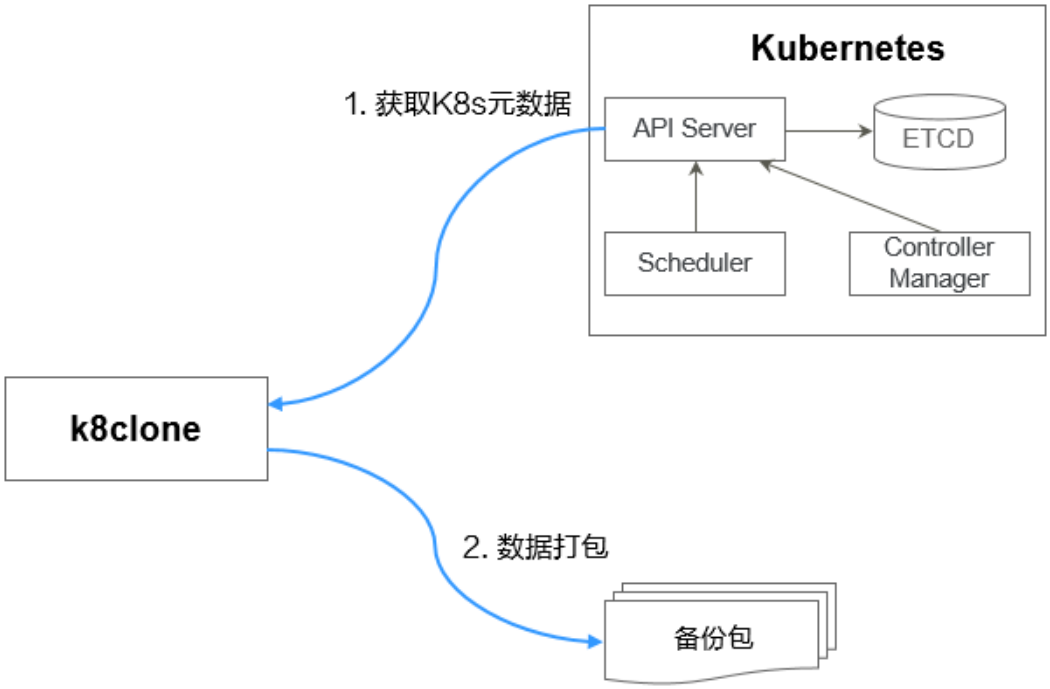
前提条件

确认云原生应用依赖的存储数据已经迁移完成。

k8clone 数据备份原理

数据备份的流程参考如下：

图 10-17 数据备份流程



k8clone 备份使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 backup -h**，可以查看k8clone工具备份的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见Kubernetes文档。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -n, --namespace：备份指定命名空间的云原生应用，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示备份整个集群。
- -e, --exclude-namespaces：排除指定命名空间对象的备份，不能和--namespace一起使用。
- -x, --exclude-kind：排除指定资源类型的备份。
- -i, --include-kind : 指定资源类型的备份。

- -y, --exclude-object：排除指定资源对象的备份。
- -z, --include-object：指定资源对象的备份。
- -w, --exclude-having-owner-ref：排除拥有ownerReferences资源对象的备份，默认是false。传参过程中需要带上等号，如：-w=true 。
- -d, --local-dir：备份数据放置的路径，默认是当前目录下k8clone-dump文件夹。

```
$ ./k8clone-linux-amd64 backup -h
Backup Workload Data as yaml files

Usage:
   k8clone backup [flags]

Flags:
   -s, --api-server string            Kubernetes api-server url
   -q, --context string               Kubernetes configuration context
   -w, --exclude-having-owner-ref     Exclude all objects having an Owner Reference. The following form is not permitted for boolean flags such as '-w false', please use '-w=false'
   -x, --exclude-kind strings         Ressource kind to exclude. Eg. 'deployment'
   -i, --include-kind strings         Ressource kind to include. Eg. 'deployment'
   -e, --exclude-namespaces strings   Namespaces to exclude. Eg. 'temp.*' as regexes. This collects all namespaces and then filters them. Don't use it with the namespace flag.
   -y, --exclude-object strings       Object to exclude. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -z, --include-object strings       Object to include. The form is '<kind>:<namespace>/<name>',namespace can be empty when object is not namespaced. Eg. 'configmap:kube-system/kube-dns'
   -h, --help                         help for backup
   -k, --kubeconfig string            The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
   -d, --local-dir string             Where to dump yaml files (default "./k8clone-dump")
   -n, --namespace string             Only dump objects from this namespace
```

示例如下：

- 整个集群对象的备份，默认路径为当前目录下“k8clone-dump”文件夹

  ./k8clone-linux-amd64 backup
- 整个集群对象的备份，并指定备份数据路径

  ./k8clone-linux-amd64 backup -d ./xxxx
- 指定命名空间对象的备份

  ./k8clone-linux-amd64 backup -n default
- 排除命名空间对象的备份

  ./k8clone-linux-amd64 backup -e kube-system,kube-public,kube-node-lease
- 排除指定资源类型的备份

  ./k8clone-linux-amd64 backup -x endpoints,endpointslice
- 指定资源类型的备份

  ./k8clone-linux-amd64 backup -x rolebinding
- 排除指定资源对象的备份

  ./k8clone-linux-amd64 backup -y configmap:kube-system/kube-dns
- 指定资源对象的备份

  ./k8clone-linux-amd64 backup -z configmap:kube-system/kube-dns
- 排除拥有ownerReferences资源对象的备份

  ./k8clone-linux-amd64 backup -w=true

应用备份操作步骤

步骤1 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。

步骤2 进入k8clone工具所在目录，执行备份命令，即可备份数据到本地目录，并打包成压缩包。

k8clone备份使用方法的示例中提供了几种常见的备份方式，您可以按需选择，也可以自定义备份方式。

----结束

10.6.5 应用迁移

不同AZ UCS华为云集群应用的迁移包含两个步骤：应用备份和应用迁移，即备份源AZ集群中应用，然后通过数据恢复的方式迁移至目标AZ集群。

k8clone是一个简便的Kubernetes元数据克隆工具，它可以将Kubernetes元数据（对象）保存为本地压缩包，然后将这些元数据恢复到目标集群中。

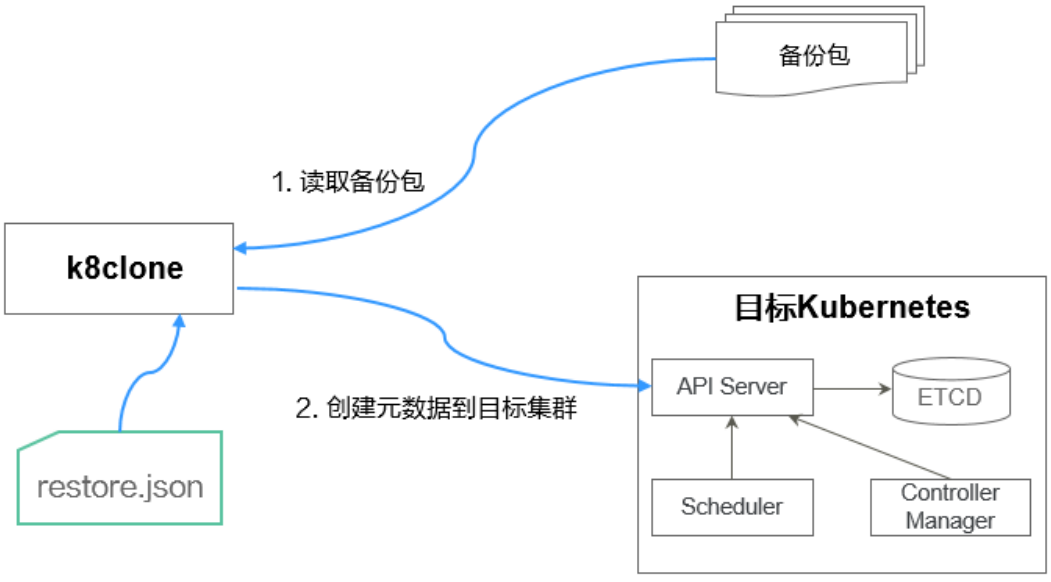
前提条件

- 确认云原生应用依赖的存储数据已经迁移完成。
- 确认源集群中元数据备份数据已经下载到执行k8clone的服务器上。

k8clone 数据恢复原理

数据恢复的流程参考如下：

图 10-18 数据恢复流程



在执行恢复操作前，需要准备一个数据恢复配置文件“restore.json”，目的是在应用恢复时自动更换PVC、StatefulSet的存储类名称，以及工作负载所使用镜像的Repository地址。

文件内容如下：

```
{
   "StorageClass":
        "OldStorageClassName": "NewStorageClassName"  //支持修改PVC、StatefulSet的StorageClassName
字段
   "ImageRepo":
        "OldImageRepo1": "NewImageRepo1",   //eg:"dockerhub.com": "cn-north-4.swr.huaweicloud.com"
        "OldImageRepo2": "NewImageRepo2",   //eg:"dockerhub.com/org1": "cn-
north-4.swr.huaweicloud.com/org2"
        "NoRepo": "NewImageRepo3"   //eg:"golang": "swr.cn-north-4.myhuaweicloud.com/paas/golang"
}
```

- StorageClass：支持PVC、有状态应用VolumeClaimTemplates中存储类名称按照配置进行自动更换。
- ImageRepo：支持工作负载所使用镜像的Repository地址的更换，工作负载包括Deployment（含initContainer）、StatefulSet、Orphaned Pod、Job、CronJob、Replica Set、Replication Controller、DaemonSet。

k8clone 恢复使用方法

说明

k8clone工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**k8clone-linux-amd64**分别替换为**k8clone-linux-arm64**或**k8clone-windows-amd64.exe**。

在k8clone工具所在目录下执行**./k8clone-linux-amd64 restore -h**，可以查看k8clone工具恢复的使用方法。

- -k, --kubeconfig：指定kubectl的KubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册Kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -s, --api-server：Kubernetes API Server URL，默认是""。
- -q, --context：Kubernetes Configuration Context，默认是""。
- -f, --restore-conf：指定restore.json的路径，默认是k8clone工具所在目录下。
- -d, --local-dir：备份数据放置的路径，默认是k8clone工具所在目录下。

```
$ ./k8clone-linux-amd64 restore -h
ProcessRestore from backup

Usage:
    k8clone restore [flags]

Flags:
  -s, --api-server string     Kubernetes api-server url
  -q, --context string        Kubernetes configuration context
  -h, --help                  help for restore
  -k, --kubeconfig string     The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config.
  -d, --local-dir string      Where to restore (default "./k8clone-dump.zip")
  -f, --restore-conf string   restore conf file (default "./restore.json")
```

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

应用恢复操作步骤

步骤1 通过kubectl连接目标集群。具体方法可参考**使用kubectl连接集群**。

步骤2 准备数据恢复配置文件：restore.json。

新建一个restore.json文件，按照格式修改，并将文件放置在k8clone工具所在目录下。

示例：

```
{
  "StorageClass": {
    "csi-disk": "csi-disk-new"
  },
  "ImageRepo": {
    "quay.io/coreos": "swr.cn-north-4.myhuaweicloud.com/paas"
  }
}
```

步骤3 进入k8clone工具所在目录，执行恢复命令，将备份数据恢复到目标集群。

示例：

./k8clone-linux-amd64 restore -d ./k8clone-dump.zip -f ./restore.json

----结束

11 流水线

11.1 流水线概述

流水线是华为云 CodeArts Pipeline 基于UCS容器舰队提供的多云场景下的自动化发布管理服务，提供调测、编排、发布、上线等一站式能力，帮助您建立敏捷高效的应用交付整体解决方案。

基于流水线对UCS容器舰队应用进行发布，可以实现在公有云、私有云、边缘云等多云场景下的跨云一站式应用发布。

前提条件

已创建UCS容器舰队，且该舰队已开通集群联邦功能，若未开通，请参考**开通集群联邦**进行开通。

流水线发布流程

图 11-1 流水线发布流程

流水线发布的流程如**图11-1**所示，具体步骤如下：

步骤1 配置项目与扩展点。在该小节，您将会为应用开通新的流水线项目，并为该项目配置跨服务权限。

步骤2 新建发布环境。在该小节，您将会为应用建立新的代码仓库，并配置环境信息以及关联的UCS集群舰队信息。

步骤3 配置发布策略。在该小节，您将会根据预置的发布模板，配置应用的发布策略。

步骤4 配置流水线及参数。在该小节，您将会对发布流程进行图形化编排，通过发布插件选择环境级别、发布环境以及产物地址。

步骤5 发布舰队应用。在该小节，您将会通过流水线，结合上述步骤的调测、编排，实现从源码构建到舰队应用发布全流程的自动化体验。

----结束

11.2 配置项目与扩展点

本节将指导您为应用开通新的流水线项目，并为该项目配置跨服务权限。

创建 Scrum 项目

步骤1 登录UCS控制台，在左侧导航栏选择“流水线”。

步骤2 单击“开始搭建你的第一个容器舰队流水线”，跳转至CodeArts服务界面，单击“立即开通”。

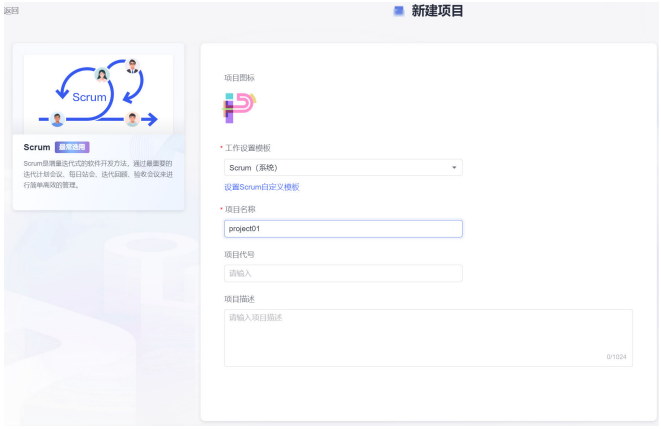
图 11-2 开通 CodeArts 服务

步骤3 单击“新建项目”，找到项目模板中的的Scrum，单击“选用”。

图 11-3 选用 Scrum 项目

步骤4 输入项目名称等信息，创建Scrum项目，创建成功后会自动跳转至Scrum项目首页。

图 11-4 创建 Scrum 项目



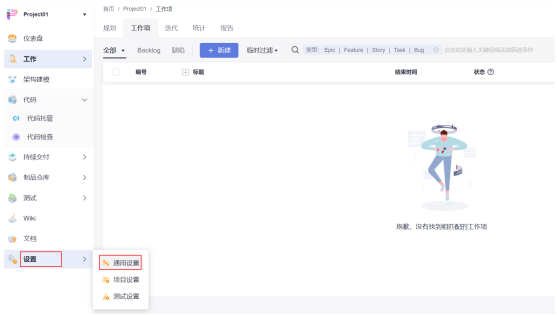
----结束

配置服务扩展点

通过新建服务扩展点并在其中配置账号的IAM权限认证信息，您可以在CodeArts侧获取到在同云服务账号、以及跨账号的场景下UCS服务中的舰队信息，完成跨服务的权限对接以及应用发布。配置服务扩展点的操作步骤如下。

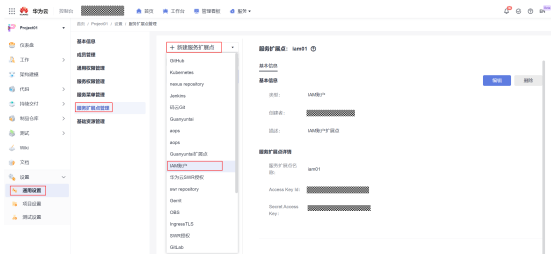
步骤1 在Scrum项目首页左侧的导航栏中选择“设置>通用设置”。

图 11-5 选择通用设置



步骤2 单击“服务扩展点管理”，并单击“新建服务扩展点”，在下拉菜单中选择“IAM账户”。

图 11-6 配置服务扩展点



步骤3 在服务扩展点中配置IAM信息，相关信息填写指导见**表11-1**。

图 11-7 新建服务扩展点

表 11-1 配置 IAM 信息参数说明

参数	说明
连接名称	扩展点连接名称。可自定义，本示例中为IAM01。
Access Key Id	访问密钥ID，获取方法请参见**获取访问密钥AK/SK**。
Secret Access Key	秘密访问密钥，获取方法请参见**获取访问密钥AK/SK**。

----结束

11.3 新建发布环境

本节将指导您为应用建立新的代码仓库、并配置环境信息以及关联的UCS集群舰队信息。

创建代码仓库

步骤1 在Scrum项目首页，搜索**创建Scrum项目**中创建的项目，单击项目名称，进入项目。

步骤2 在左侧导航栏选择“代码>代码托管”，单击“新建普通仓库”。

图 11-8 新建普通仓库

步骤3 配置代码仓库的名称、权限设置、是否公开等信息，具体配置按照**图11-9**中的默认配置进行设置。

图 11-9 代码仓库配置

----结束

新建环境

步骤1 在左侧导航栏选择“持续交付>发布管理”，单击“环境”进入环境列表页，即可查看当前发布管理下的所有环境列表。

图 11-10 查看环境

步骤2 单击“新建环境”，填写基本信息，相关参数说明如**表11-2**所示。

须知

- 当选择发布用户为当前用户时，可以直接获取对应账号的UCS服务舰队信息。
- 当选择发布用户为其他用户时，可以通过**配置项目与扩展点**配置的IAM服务扩展点获取对应账号的UCS舰队信息。

图 11-11 选择发布用户为当前用户

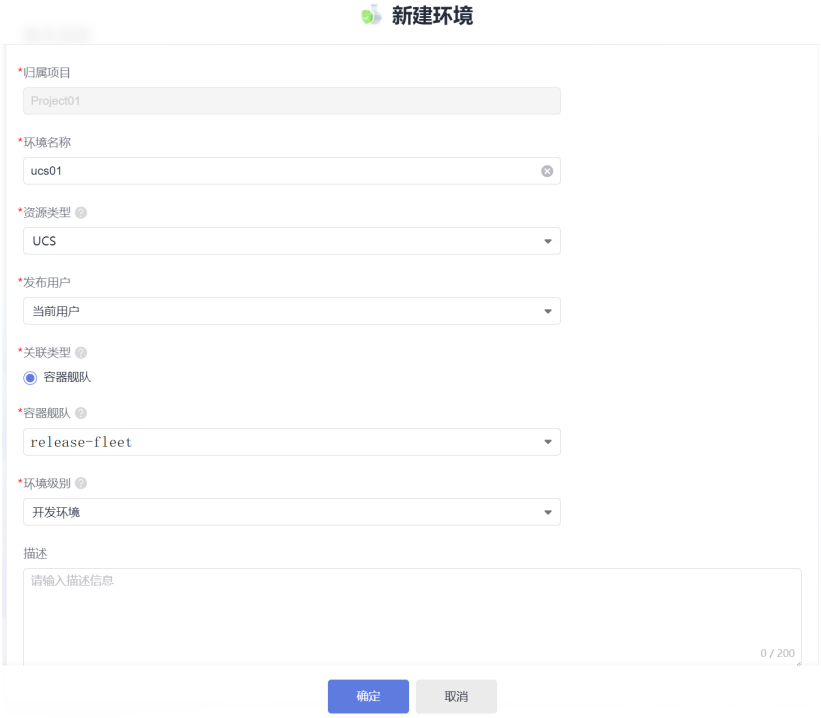


图 11-12 选择发布用户为其他用户

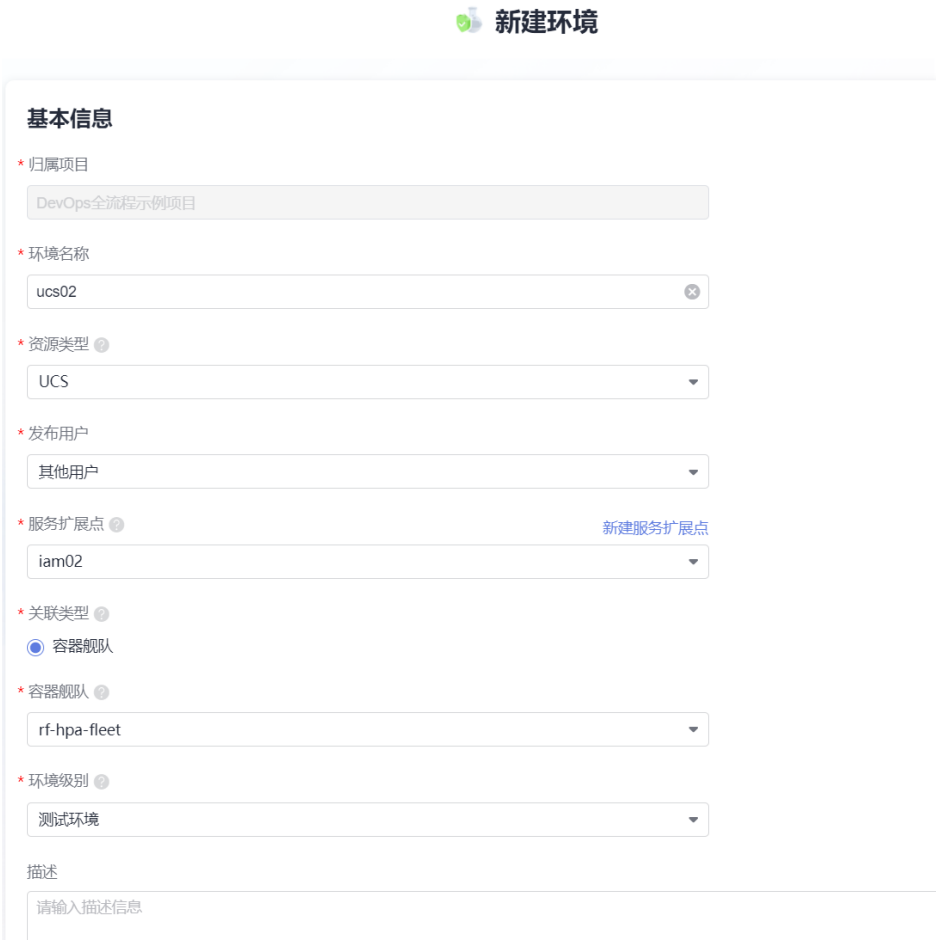


表 11-2 新建环境参数说明

参数	说明
环境名称	发布管理下环境唯一标识，创建后不可修改。
资源类型	支持多种类型资源的部署，不同类型的资源支持的部署插件不同。
发布用户	可选择当前用户或其他用户，获取对应账号的舰队信息进行应用发布。
服务扩展点	配置授权扩展点以获取UCS资源权限。创建服务扩展点请参见**获取访问密钥AK/SK**。。
关联类型	选择关联的UCS资源粒度，当前支持容器舰队。
容器舰队	选择UCS中开启联邦功能的集群舰队。
环境级别	环境类型，内置了开发环境、测试环境、预发环境和生产环境四种类型。

参数	说明
描述	环境的描述信息。可选填。

步骤3 单击“确定”，完成环境创建。创建成功后将自动跳转至环境详情页面。

----结束

11.4 配置发布策略

发布管理预置滚动升级模板，本节将指导您基于滚动升级模板，添加滚动升级插件，配置发布策略。

UCS流水线目前仅支持预置滚动升级模板。

步骤1 在环境详情页面，单击“发布策略”。

图 11-13 发布策略

步骤2 单击在自定义策略右侧的+号，然后在弹出的新建策略窗口中，根据需要选择策略模板，单击“确定”。

图 11-14 新建策略

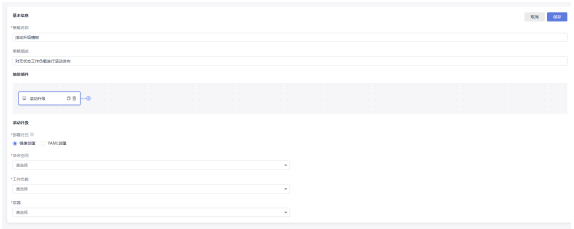
步骤3 在模板上填写基本信息，添加插件，以自定义编排任务。

其中，UCS滚动升级插件支持镜像创建和YAML创建两种部署方式：

镜像创建

选择镜像创建时，需要选中对应的舰队命名空间、工作负载与容器。部署时，流水线上的镜像将直接替换对应命名空间、工作负载与容器中的镜像。

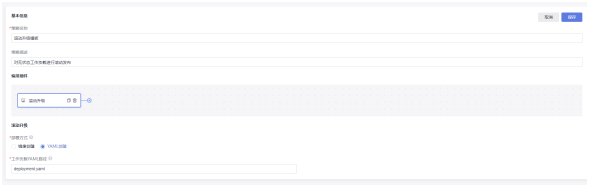
图 11-15 镜像创建



YAML创建

在代码仓内新建一个YAML文件，填写升级的工作负载YAML路径。

图 11-16 YAML 创建

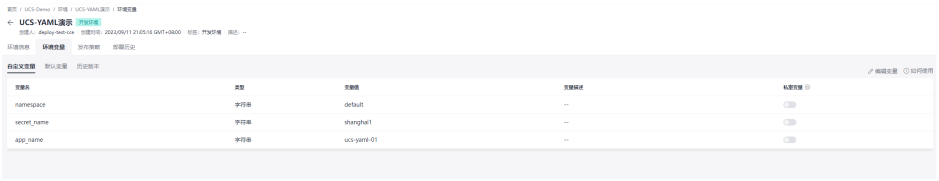


若涉及拉取私有镜像，您需要进行以下操作：

在UCS服务侧，为对应的集群配置Docker镜像仓库密钥，并记录密钥名称，具体操作参考**密钥（Secret）**。

在“发布管理>环境>环境变量”中设置环境变量，在yaml中可以通过{{}}形式引用。

图 11-17 设置环境变量



示例yaml文件：

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: {{app_name}}
  namespace: {{namespace}}
spec:
  replicas: 3
  selector:
    matchLabels:
      app: {{app_name}}
      version: v1
  template:
    metadata:
      labels:
        app: {{app_name}}
        version: v1
    spec:
      containers:
        - name: container-1
          image: {{ARTIFACT}}
          env:
            - name: PAAS_APP_NAME
              value: {{app_name}}
            - name: PAAS_NAMESPACE
              value: {{namespace}}
            - name: PAAS_PROJECT_ID
              value: {{PROJECT_ID}}
          resources:
            limits:
              cpu: 250m
              memory: 512Mi
            requests:
              cpu: 250m
```

```
        memory: 512Mi
  imagePullSecrets:
    - name: {{secret_name}}
  schedulerName: default-scheduler
```

说明

- YAML创建部署方式仅支持单个YAML文件。
- YAML文件的代码仓和分支来源于您在发布管理中配置的代码仓和分支。
- YAML路径为相对路径，当前目录为代码分支的根目录。
- YAML路径可以使用“${变量名}”引用环境变量，YAML文件中可以使用“{{变量名}}”引用环境变量。

----结束

11.5 配置流水线及参数

本节将指导您对发布流程进行图形化编排，通过发布插件选择环境级别、发布环境以及产物地址。

步骤1 在左侧导航栏选择“持续交付>流水线”，进入流水线页面。

步骤2 单击“新建流水线”，选择**创建代码仓库**中创建的代码仓。

图 11-18 新建流水线

步骤3 单击“下一步”，在系统模板中选择“新手上路”，进入流水线任务编排页面。

步骤4 根据业务具体情况编辑阶段名称，并设置每个任务的执行内容和编排详情。

图 11-19 任务编排

设置任务编排中，中间构建阶段的名称为“阶段_1”，任务类型为“build构建”。该阶段的具体操作为基于应用源码构建部署应用的镜像，详细的操作细节见**CodeArts流水线**。

设置最右侧阶段的名称为“阶段_2”，任务类型为“云原生发布”。该阶段的具体操作为根据定义的交付资源yaml文件将应用部署至UCS所属舰队中。

图 11-20 添加云原生发布任务

步骤5 单击“菜单流水线”，选择发布插件，并配置环境级别、需要发布的环境，以及产物地址。

产物地址指**步骤4**中设置的“阶段_1 Build构建”通过源码编译生成的镜像，并推送至SWR镜像仓库的地址。配置该参数时可直接输入产物地址+引用镜像版本号，也可以使用${变量名}使用环境变量引用构建产物。

图 11-21 设置云原生发布任务

步骤6 修改工作负载的YAML文件，在image字段中引用ARTIFACT默认变量，产物地址会通过ARTIFACT默认变量渲染到工作负载YAML的image字段。

```
image:{{ARTIFACT}}
```

流水线配置成功后，流水线详情页面如**图11-22**所示。

图 11-22 流水线配置成功

----结束

11.6 发布舰队应用

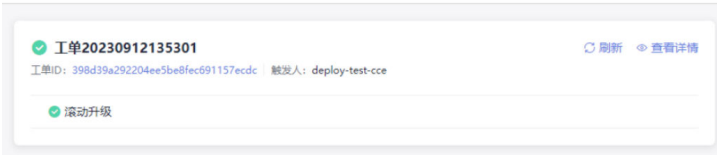
本节将指导您通过流水线，结合上述步骤的调测、编排，实现从源码构建到舰队应用发布全流程的自动化体验。

步骤1 配置流水线、运行参数和产物地址后，单击“运行”，即可执行流水线，实现编译构建和云原生发布。

图 11-23 执行流水线

步骤2 单击流水线阶段中的“发布”，并单击“任务结果”查看发布单。

图 11-24 查看发布单

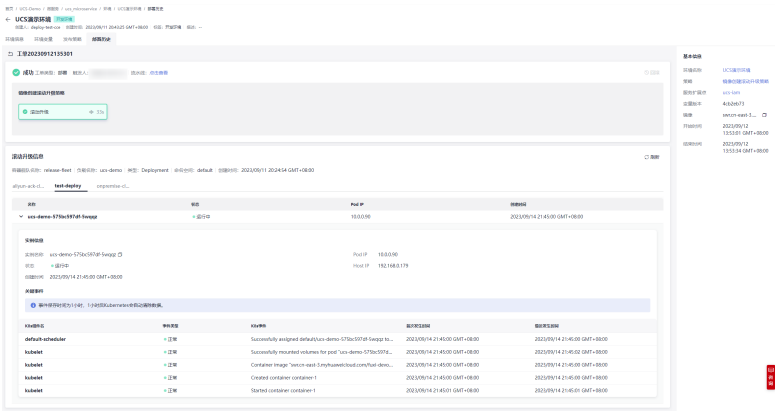


- 发布单展示本次发布的基本信息，包括：工单名称、工单号、发布任务阶段展示。
- 工单详情页面展示发布过程的详细信息，可以查看当前容器舰队下对应的各集群指定工作负载的运行情况，并且用户可以进行重试、取消等操作。

步骤3 单击“查看详情”可跳转至工单详情页。

工单详情页可查看应用在舰队内每个集群的发布过程和详细信息，可以查看到当前容器舰队下对应的华为云集群、附着集群及本地集群内工作负载的实例信息、创建时间、k8s事件信息等内容。

图 11-25 工单详情



步骤4 完成舰队应用发布后，登录UCS控制台，左侧导航栏选择“容器舰队”，单击舰队名称，左侧导航栏选择“工作负载”，查看对应的无状态工作负载是否已发布至对应集群内并正常运行。

----结束

12 错误码

如果操作请求在执行过程中出现异常导致未被处理，则会返回一条错误信息，错误信息中包括错误码和具体错误描述。

表 12-1 错误码说明

错误码	状态码	错误信息	描述
UCS.00000001	400	Failed to obtain the user information.	未能获取用户信息
UCS.00000003	400	Failed to obtain the federation information.	获取联邦信息失败
UCS.00000004	403	Request forbidden.	禁止请求
UCS.00000005	500	Database operation failed.	数据库操作失败
UCS.00000006	500	Server internal error.	服务器内部错误
UCS.00000007	500	Data transform error.	数据转换失败
UCS.00000008	500	Error add event.	添加事件失败
UCS.00000009	500	Data unmarshal error.	数据反序列化失败
UCS.00000010	500	Data marshal error.	数据序列化失败
UCS.00000011	400	Bad query parameter value.	请求参数非法
UCS.00000012	400	Invalid request body.	请求体非法

错误码	状态码	错误信息	描述
UCS.00000013	404	No requested resources found.	请求资源不存在
UCS.00000014	500	Failed to encrypt data.	加密数据失败
UCS.00000015	500	Failed to decrypt data.	解密数据失败
UCS.00000016	400	Invalid header value.	请求头非法
UCS.00000017	400	Insufficient quota	配额不足
UCS.00000018	401	Authorization failed.	授权失败
UCS.00010001	500	Failed to get iam connection.	获取IAM连接失败
UCS.00010002	403	Sub-user has no authority to create agency.	子用户无权创建委托
UCS.00010003	400	Failed to create agency.	创建委托失败
UCS.00010004	500	Failed to get role id for te_admin.	获取te_admin role失败
UCS.00010005	500	Failed to get admin token from iam.	获取admin token失败
UCS.00010006	500	Failed to get agency list from iam.	获取委托列表失败
UCS.00010007	500	Failed to get agency grants from iam.	获取委托grants失败
UCS.00010008	500	Failed to update agency role.	更新委托role失败
UCS.00010009	400	Failed to get project token by agency	通过委托获取项目token失败
UCS.00010010	400	Failed to get op_svc account domain token	获取op账号token失败
UCS.00010011	400	Failed to get project id by project name.	获取项目id失败
UCS.00010012	400	IAM agency quota insufficient, please expand agency quota	超出IAM委托配额
UCS.00010013	400	fail to get iam pdp authorize result	获取PDP鉴权结果失败

错误码	状态码	错误信息	描述
UCS.00010014	403	iam pdp authentication denied	PDP鉴权拒绝
UCS.00010015	403	iam rbac authentication denied	RBAC鉴权拒绝
UCS.00020001	500	Failed to get aeskey.	获取aeskey失败
UCS.00020002	500	Failed to get certs.	获取证书失败
UCS.00020003	500	Failed to create certs.	创建证书失败
UCS.00020003	500	Failed to delete certs.	删除证书失败
UCS.00030001	404	Cluster Not Found.	集群不存在
UCS.00030002	400	Failed to obtain the cluster information.	获取集群信息失败
UCS.00030003	400	Failed to get resourceJob info with cluster status	获取resource job失败
UCS.00040001	400	Failed to obtain the mesh information.	获取网格信息失败
UCS.00090001	500	Failed to create DNSRecord	记录集创建失败
UCS.00100001	400	Failed to publish message to smn.	发布消息到SMN失败
UCS.00100002	400	smn topic error.	SMN主题错误
UCS.00100003	400	smn subscription error.	SMN订阅错误
UCS.00110001	400	SDR failed to get billing raw data	获取计费数据失败
UCS.00110002	400	Formatting raw billing data to SDR format error	格式化计费数据失败
UCS.00120001	400	CBC failed to update resources status	更新CBC资源状态失败
UCS.00130001	400	Get UCS Agency info error	获取UCS委托错误
UCS.00140001	400	Create ClusterRole failed	ClusterRole创建失败

错误码	状态码	错误信息	描述
UCS.00140002	400	Delete ClusterRole failed	ClusterRole删除失败
UCS.00140003	400	Update ClusterRole failed	ClusterRole更新失败
UCS.00140004	400	Get ClusterRole failed	ClusterRole信息获取失败
UCS.00140005	400	Create ClusterRoleBinding failed	ClusterRoleBinding创建失败
UCS.00140006	400	Delete ClusterRoleBinding failed	ClusterRoleBinding删除失败
UCS.00140007	400	Update ClusterRoleBinding failed	ClusterRoleBinding更新失败
UCS.00140008	400	Get ClusterRoleBinding failed	ClusterRoleBinding信息获取失败
UCS.00140009	400	Create Role failed	Role创建失败
UCS.00140010	400	Delete Role failed	Role删除失败
UCS.00140011	400	Update Role failed	Role更新失败
UCS.00140012	400	Get Role failed	Role信息获取失败
UCS.00140013	400	Create RoleBinding failed	RoleBinding创建失败
UCS.00140014	400	Delete RoleBinding failed	RoleBinding删除失败
UCS.00140015	400	Update RoleBinding failed	RoleBinding更新失败
UCS.00140016	400	Get RoleBinding failed	RoleBinding信息获取失败
UCS.00150001	400	Cluster policy validate failed.	集群策略验证失败
UCS.00150002	400	ClusterGroup policy validate failed.	集群组策略验证失败
UCS.00150003	400	Cluster has enable policy.	集群已启用策略
UCS.00150004	400	ClusterGroup has enable policy.	集群组已启用策略

错误码	状态码	错误信息	描述
UCS.00150005	400	Cluster not enable policy.	集群未启用策略
UCS.00150006	400	ClusterGroup not enable policy.	集群组未启用策略
UCS.00150007	500	Get policy job failed.	获取策略任务失败
UCS.01000001	400	Failed to obtain the user information.	获取用户信息失败
UCS.01000002	429	The throttling threshold has been reached.	达到流控阈值
UCS.01000003	401	Authorization failed.	授权失败
UCS.01000004	403	Request forbidden.	禁止请求
UCS.01000005	500	Database operation failed.	数据库操作失败
UCS.01000006	500	Server internal error.	服务器内部错误
UCS.01000007	500	Data transform error.	数据转换失败
UCS.01000008	500	Error add event.	添加事件失败
UCS.01000009	500	Data unmarshal error.	数据反序列化失败
UCS.01000010	500	Data marshal error.	数据序列化失败
UCS.01000011	400	Bad query parameter value.	请求参数非法
UCS.01000012	400	Invalid request body.	请求体非法
UCS.01000013	404	No requested resources found.	请求资源不存在
UCS.01000014	500	Failed to encrypt data.	加密数据失败
UCS.01000015	500	Failed to decrypt data.	解密数据失败
UCS.01000016	400	Invalid header value.	请求头非法

错误码	状态码	错误信息	描述
UCS.01000017	400	Insufficient quota	配额不足
UCS.01000018	400	Quota info validate failed	配额参数校验失败
UCS.01000019	500	Quota update failed	配额更新失败
UCS.01010001	500	Failed to get iam connection.	获取IAM连接失败
UCS.01010002	500	Failed to get project token by agency	通过委托获取项目token失败
UCS.01010003	403	No access permission. Please contact the administrator.	无访问权限
UCS.01010004	400	get deployment region's projectID error	获取项目ID失败
UCS.01010005	400	get IAM agency's token error	获取委托token失败
UCS.01010006	400	fail to get iam pdp authorize result	获取PDP鉴权结果失败
UCS.01010007	403	iam pdp authentication denied	PDP鉴权拒绝
UCS.01010008	403	iam rbac authentication denied	RBAC鉴权拒绝
UCS.01020001	500	Failed to get aeskey.	获取aeskey失败
UCS.01020002	500	Failed to get certs.	获取证书失败
UCS.01020003	500	Failed to create certs.	创建证书失败
UCS.01020004	500	Failed to delete certs.	删除证书失败
UCS.01030001	404	Cluster Not Found.	集群不存在
UCS.01030002	400	Failed to obtain the cluster information.	获取集群信息失败
UCS.01030003	409	The same cluster already exists.	存在同名集群
UCS.01030004	400	Cluster status is unavailable, please fix cluster first.	集群状态不可用

错误码	状态码	错误信息	描述
UCS.01030005	403	No authorization for cluster	集群授权失败
UCS.01030006	400	Create resource job for cluster error	集群创建resource job失败
UCS.01030007	400	Create on-demand order for cluster error	创建按需订单失败
UCS.01030008	400	Cluster kubeconfig format error.	集群kubeconfig格式错误
UCS.01030009	400	This cluster does not support unregister	集群不支持注销
UCS.01030010	400	Failed to obtain cce cluster information.	获取cce集群信息失败
UCS.01030011	400	Cluster category not supported	不支持该集群类别
UCS.01030012	400	Register cce cluster error	注册cce集群失败
UCS.01030013	400	Register attached cluster error	注册附着集群失败
UCS.01030014	400	Register on-premise cluster error	注册本地集群失败
UCS.01030015	100	Register multi cloud cluster error	注册多云集群失败
UCS.01030016	400	Cluster has been frozen	集群已被冻结
UCS.01050001	400	RecordSet create failed.	记录集创建失败
UCS.01080001	400	Failed to obtain the federation information.	获取联邦信息失败
UCS.01080002	400	Cluster group has federalized.	舰队已开启联邦
UCS.01080003	500	Cluster group federation failed.	舰队联邦操作失败
UCS.01080004	400	Cluster group federation validate failed.	开启联邦校验失败
UCS.01080005	400	Retry join all clusters to federation failed.	重试所有集群加入联邦失败
UCS.01080006	400	Cluster group has not been federalized.	舰队未开启联邦

错误码	状态码	错误信息	描述
UCS.01080007	400	Retry join cluster to federation failed.	重试集群加入联邦失败
UCS.01090001	400	Failed to obtain the mesh information.	获取网格信息失败
UCS.01100001	403	No authorization for cluster group	舰队未授权
UCS.01100002	400	associate cluster with clustergroup error	集群加入舰队失败
UCS.01100003	400	associate cluster with rule error	舰队关联权限策略失败
UCS.01100004	409	The same clustergroup already exists.	同名舰队已存在
UCS.01100005	404	ClusterGroup Not Found.	舰队不存在
UCS.01100006	400	Cluster number in fleet exceed limit.	舰队内集群数量超过限制
UCS.01100007	400	Update associated clusters validate failed	更新关联集群校验失败
UCS.01110001	400	resource notification to SMN error	通知SMN失败
UCS.01120001	400	Create ClusterRole failed	ClusterRole创建失败
UCS.01120002	400	Delete ClusterRole failed	ClusterRole删除失败
UCS.01120003	400	Update ClusterRole failed	ClusterRole更新失败
UCS.01120004	400	Get ClusterRole failed	ClusterRole信息获取失败
UCS.01120005	400	Create ClusterRoleBinding failed	ClusterRoleBinding创建失败
UCS.01120006	400	Delete ClusterRoleBinding failed	ClusterRoleBinding删除失败
UCS.01120007	400	Update ClusterRoleBinding failed	ClusterRoleBinding更新失败
UCS.01120008	400	Get ClusterRoleBinding failed	ClusterRoleBinding信息获取失败
UCS.01120009	400	Create Role failed	Role创建失败

错误码	状态码	错误信息	描述
UCS.01120010	400	Delete Role failed	Role删除失败
UCS.01120011	400	Update Role failed	Role更新失败
UCS.01120012	400	Get Role failed	Role信息获取失败
UCS.01120013	400	Create RoleBinding failed	RoleBinding创建失败
UCS.01120014	400	Delete RoleBinding failed	RoleBinding删除失败
UCS.01120015	400	Update RoleBinding failed	RoleBinding更新失败
UCS.01120016	400	Get RoleBinding failed	RoleBinding信息获取失败
UCS.01130001	400	policy management create reconcile job failed	策略管理创建协调作业失败
UCS.01130002	400	policy management create disable job failed	策略管理创建禁用作业失败
UCS.01130003	400	cluster policy validate failed.	集群策略验证失败
UCS.01130004	400	clusterGroup policy validate failed.	集群组策略验证失败
UCS.01130005	400	cluster policy management is in installing or closing status	集群策略管理处于安装或关闭状态
UCS.01130006	400	cluster group policy management is in installing or closing status	集群组策略管理处于安装或关闭状态