云容器实例

常见问题

文档版本 01

发布日期 2023-04-30

华为技术有限公司

商标声明


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

注意

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

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

安全声明

漏洞处理流程

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

目 录

1 产品咨询类

1.1 CCI 服务的开源第三方中包含的公网地址声明是什么?

CCI服务提供 Serverless Container（无服务器容器）引擎，让用户无需创建和管理服务器集群即可直接运行容器。在CCI服务组件开源依赖中，包含三方开源依赖k8s.io/kubernetes、go.etcd.io/etcd，其中涉及"http://metadata.google.internal."、"https://www.googleapis.com"、"https://storage.googleapis.com"等域名的公网地址均为开源仓内自带的公网地址，CCI服务不涉及与此类公网地址的连接，也不会与此类公网地址进行任何数据交换。

k8s.io/kubernetes开源社区链接：**https://github.com/kubernetes/kubernetes**

go.etcd.io/etcd开源社区链接：**https://github.com/etcd-io/etcd**

2 基本概念类

2.1 什么是云容器实例?

云容器实例（Cloud Container Instance，CCI）服务提供 Serverless Container（无服务器容器）引擎，让您无需创建和管理服务器集群即可直接运行容器。

Serverless 是一种架构理念，是指不用创建和管理服务器、不用担心服务器的运行状态（服务器是否在工作等），只需动态申请应用需要的资源，把服务器留给专门的维护人员管理和维护，进而专注于应用开发，提升应用开发效率、节约企业IT成本。传统上使用 Kubernetes 运行容器，首先需要创建运行容器的 Kubernetes 服务器集群，然后再创建容器负载。云容器实例的 Serverless Container 就是从使用角度，无需创建、管理 Kubernetes 集群，也就是从使用的角度看不见服务器（Serverless），直接通过控制台、kubectl、Kubernetes API 创建和使用容器负载，且只需为容器所使用的资源付费。

云容器实例有如下功能：

- 自动化持续交付
  一键运行DevOps的CI流程生成的容器镜像，保障CI/CD流程全自动化。
- 应用运行时全托管
  提供无状态工作负载的运行时托管，保障应用稳定运行。
- 极速弹性扩缩容
  支持用户自定义弹性伸缩策略，且能在1秒内实现弹性扩缩容。
- 应用高可用保障
  支持多实例同时对外提供服务，保障用户业务高可靠，并提供全局负载均衡能力。
- 应用容器状态监控
  提供容器健康状态检查和容器的运行时指标实时监控。
- 数据持久化存储
  支持挂载网络存储卷，保障业务数据持久化存储。

2.2 云容器实例和云容器引擎有什么区别?

华为云提供高性能、高可用、高安全的企业级容器服务，通过CNCF官方认证的两种Kubernetes服务供用户选择，包括云容器引擎（CCE）与云容器实例（CCI）。

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群，支持运行容器，提供了Kubernetes集群管理、容器应用全生命周期管理、应用服务网格、Helm应用模板、插件管理、应用调度、监控与运维等容器全栈能力，为您提供一站式容器平台服务。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

详细介绍请查看**什么是云容器引擎**。

云容器实例（Cloud Container Instance，简称CCI）服务提供 Serverless Container（无服务器容器）引擎，让您无需创建和管理服务器集群即可直接运行容器。通过CCI您只需要管理运行在Kubernetes上的容器化业务，无需管理集群和服务器即可在CCI上快速创建和运行容器负载，使容器应用零运维，使企业聚焦业务核心，为企业提供了Serverless化全新一代的体验和选择。

而Serverless是一种架构理念，是指不用创建和管理服务器、不用担心服务器的运行状态（服务器是否在工作等），只需动态申请应用需要的资源，把服务器留给专门的维护人员管理和维护，进而专注于应用开发，提升应用开发效率、节约企业IT成本。传统上使用 Kubernetes 运行容器，首先需要创建运行容器的 Kubernetes 服务器集群，然后再创建容器负载。

详细介绍请查看**什么是云容器实例**。

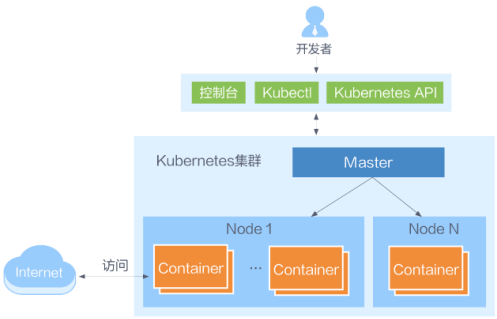
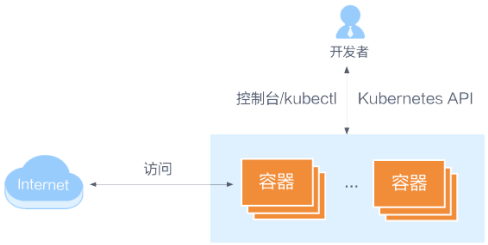
基本介绍

表 2-1 CCE 和 CCI 基本介绍

云容器引擎CCE	云容器实例CCI
云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群，支持运行Docker容器，提供了Kubernetes集群管理、容器应用全生命周期管理、应用服务网格、Helm应用模板、插件管理、应用调度、监控与运维等容器全栈能力，为您提供一站式容器平台服务。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。 详细介绍请查看**什么是云容器引擎**。	云容器实例（Cloud Container Instance， CCI）服务提供Serverless Container（无服务器容器）引擎，让您无需创建和管理服务器集群即可直接运行容器。通过CCI您只需要管理运行在Kubernetes上的容器化业务，无需管理集群和服务器即可在CCI上快速创建和运行容器负载，使容器应用零运维，使企业聚焦业务核心，为企业提供了Serverless化全新一代的体验和选择。 而Serverless是一种架构理念，是指不用创建和管理服务器、不用担心服务器的运行状态（服务器是否在工作等），只需动态申请应用需要的资源，把服务器留给专门的维护人员管理和维护，进而专注于应用开发，提升应用开发效率、节约企业IT成本。传统上使用Kubernetes运行容器，首先需要创建运行容器的Kubernetes服务器集群，然后再创建容器负载。 详细介绍请查看**什么是云容器实例**。

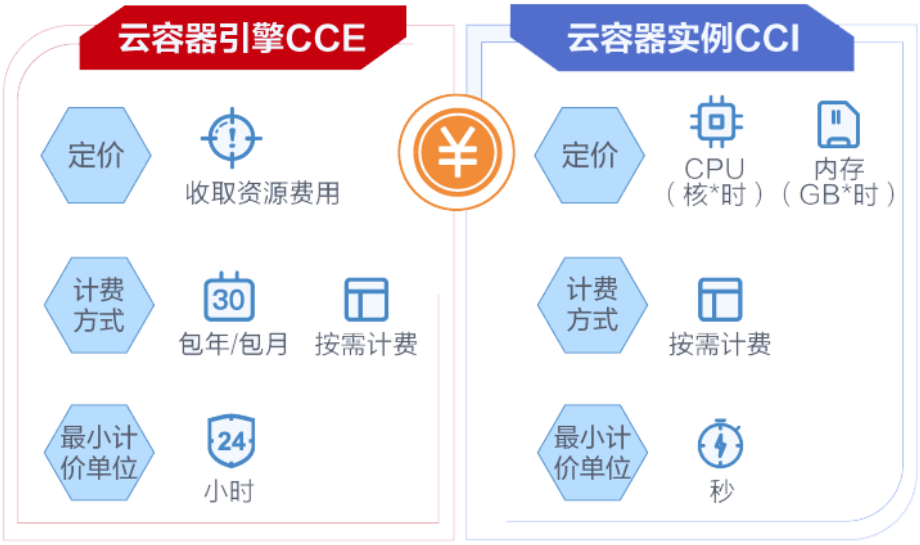
创建方式

表 2-2 创建方式不同

云容器引擎CCE	云容器实例CCI
CCE是基于Kubernetes的托管式容器管理服务，可以提供原生Kubernetes体验，可以一键创建原生Kubernetes集群，与社区能力基本一致。 使用CCE，您需要创建集群和节点，简单、低成本、高可用，无需管理Master节点。 	CCI提供 Serverless Container引擎，在华为云上部署容器时，您不需要购买和管理ECS，可以直接在华为云上运行容器和Pod，为您省去底层ECS的运维和管理工作。 使用CCI，您无需创建集群，无需创建和管理Master节点及Work节点，可直接启动应用程序。 

收费方式

图 2-1 CCE 和 CCI 收费方式区别



应用场景

图 2-2 CCE 和 CCI 应用场景区别

集群创建

图 2-3 CCE 和 CCI 创建集群区别

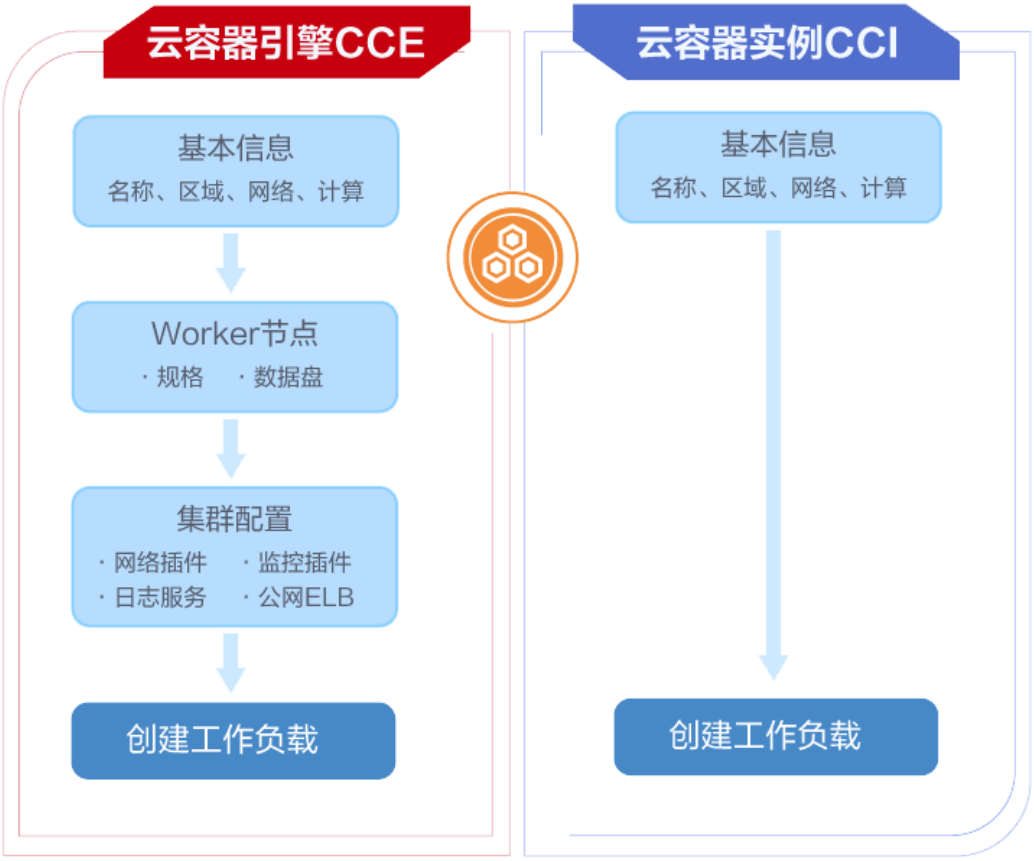


表 2-3 CCE 和 CCI 创建集群区别

云容器引擎CCE	云容器实例CCI
基本信息（名称、区域、网络、计算）-> Worker节点 -> 集群配置 -> 创建工作负载	基本信息（名称、区域、网络、计算）->创建工作负载

CCE 与 CCI 两者的配合

通过安装Virtual-Kubelet插件，可以在短时高负载场景时，将部署在CCE上的无状态工作负载（Deployment）、有状态工作负载（StatefulSet）、普通任务（Job）三种资源类型的容器实例（Pod），弹性创建到华为云云容器实例CCI服务上，以减少集群扩容带来的消耗。

具体功能如下：

- 支持容器实例实现秒级弹性伸缩：在集群资源不足时，无需新增节点，virtual-kubelet插件将自动为您在云容器实例CCI侧创建容器实例，减少运维成本。
- 无缝对接华为云容器镜像服务SWR，支持使用公用镜像和私有镜像。

- 支持CCI容器实例的事件同步、监控、日志、exec、查看状态等操作。
- 支持查看虚拟弹性节点的节点容量信息。
- 支持CCE和CCI两侧实例的service网络互通。

详情请参见**华为云CCE弹性伸缩至CCI**。

2.3 什么是环境变量?

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，为工作负载提供了极大的灵活性。

在CCI中设置环境变量与Dockerfile中的“ENV”效果相同。

2.4 什么是服务?

服务定义了实例及访问实例的途径，如单个稳定的IP地址和相应的DNS名称。

为了解决组件间的通信问题，CCI使用服务名称代替IP地址，从而实现组件间的相互访问。在创建工作负载时会指定服务名称。

2.5 什么是 mcore?

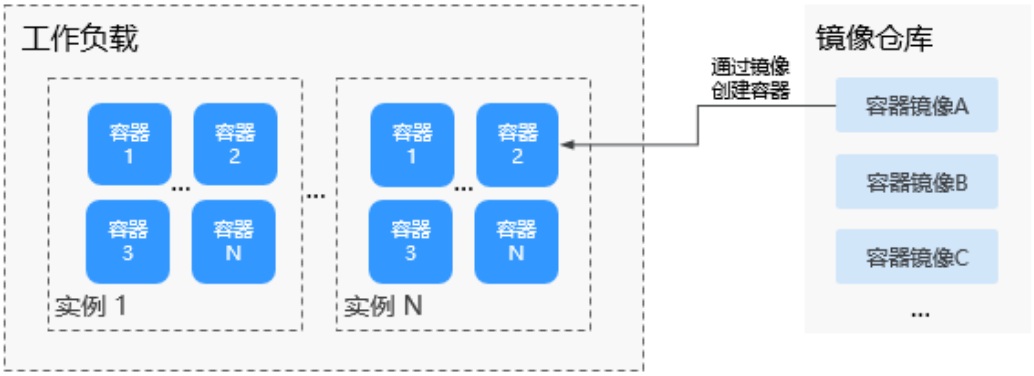
mcore是CPU使用单位，为千分之一个核。通常容器工作负载的CPU用量以mcore为单位衡量。

2.6 镜像、容器、工作负载的关系是什么?

- 镜像：容器镜像是一个特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的配置参数（如匿名卷、环境变量、用户等）。镜像不包含任何动态数据，其内容在构建之后也不会被改变。
- 容器：镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的类和实例一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。
- 工作负载：工作负载是在 Kubernetes 上运行的应用程序。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。

镜像、容器、工作负载之间的关系请参见下图。

图 2-4 镜像、容器、工作负载的关系



2.7 什么是安全容器?

安全容器这个概念主要与普通容器进行比较的。

和普通容器相比，它最主要的区别是每个容器（准确地说是pod）都运行在一个单独的微型虚拟机中，拥有独立的操作系统内核，以及虚拟化层的安全隔离。因为云容器实例采用的是共享多租集群，因此容器的安全隔离比用户独立拥有私有Kubernetes集群有更严格的要求。通过安全容器，不同租户之间的容器之间，内核、计算资源、存储和网络都是隔离开的。保护了用户的资源和数据不被其他用户抢占和窃取。

2.8 能否使用 kubectl 管理容器实例?

支持，具体请参见**https://support.huaweicloud.com/devg-cci/cci_kubectl.html**。

2.9 CCI 资源包中的核时怎么理解?

1 核*时 = 1 * 3600（核*秒）

1 核*时 ： 1核的CPU连续跑1个小时所用的资源量

1 核*秒： 1核的CPU连续跑1秒所用的资源量

案例一：

假设用户的Deployment是2.5核的，连续运行了2个小时，那么它所消耗的资源量为：2.5 * 2 = 5（核*时）= 5* 3600（核*秒）。

案例二：

假设当前是730核*时，那么最大可以1小时内运行一个730核的容器，也可以730小时内运行一个1核的容器。

详细信息您可以参考**计费说明**。

3 工作负载异常

3.1 定位思路

当工作负载状态异常时，建议先查看事件。

在CCI控制台中，单击左侧导航栏的“工作负载”，单击异常工作负载名称，进入详情页面，在Pod列表中，单击异常实例左边的，显示该实例的详情，单击事件页签。

图 3-1 查看事件

3.2 事件一：重新拉取镜像失败

工作负载详情中，如果事件中提示“重新拉取镜像失败”，请参照如下方式来排查原因。

排查项一：kubectl 创建工作负载时未指定 imagePullSecret

以创建一个名为nginx的deployment为例，请排查yaml文件中是否存在**imagePullSecrets**字段（如下yaml示例中的加粗字段），表示pull镜像时的secret名称。

需要使用**容器镜像服务**的镜像时，参数值固定为**imagepull-secret**。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:alpine
        imagePullPolicy: Always
        name: nginx
      imagePullSecrets:
        - name: imagepull-secret
```

排查项二：填写的镜像地址错误

CCI支持拉取**容器镜像服务**中的镜像来创建工作负载。

容器镜像服务中的镜像需要使用镜像的“下载指令”，上传镜像后，您可以在容器镜像服务的镜像中获取，如下图所示。

图 3-2 镜像地址

排查项三：IAM 用户没有镜像下载权限

如果您开通了企业管理，您需要在账号下的容器镜像服务中给IAM用户添加权限，IAM用户才能使用账号下的私有镜像。

给IAM用户添加权限有如下两种方法：

- 在镜像详情中为IAM用户添加授权，授权完成后，IAM用户享有读取/编辑/管理该镜像的权限，具体请参见**在镜像详情中添加授权**。

- 在组织中为IAM用户添加授权，使IAM用户对组织内所有镜像享有读取/编辑/管理的权限，具体请参见**在组织中添加授权**。

排查项四：镜像打包使用的 docker 版本过低

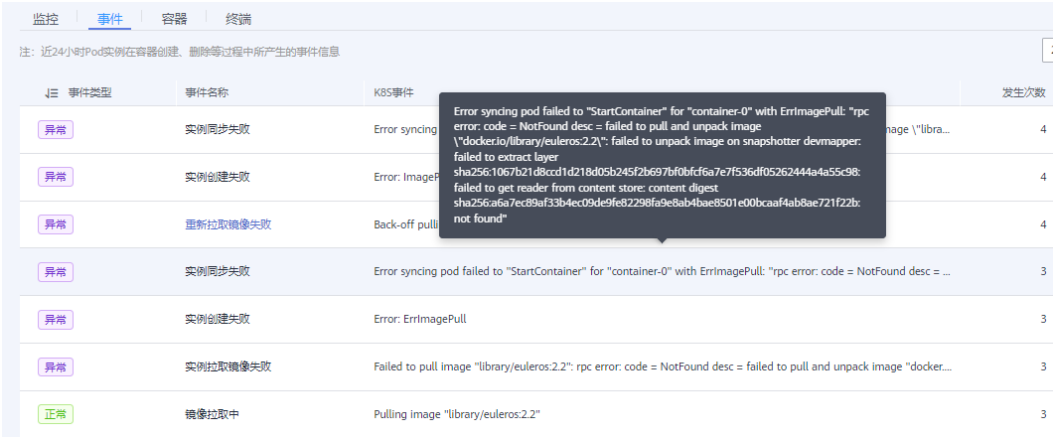
业务负载拉取镜像失败，报错信息如下：

failed to pull and unpack image "****": failed to unpack image on snapshotter devmapper: failed to extract layer

sha256:xxxxxx: failed to get reader from content store: content digest sha256:xxxxxx: not found

错误如下图所示：

图 3-3 报错信息



该错误出现的原因：镜像构建时使用的docker版本过低（<v1.10），部分镜像打包标准社区已经不再支持。

解决方案：请使用新版本 docker 运行时（>= docker v1.11）重新构建镜像后上传到SWR（容器镜像服务），升级负载镜像版本，重新拉取即可。

3.3 事件二：重新启动容器失败

工作负载详情中，如果事件中提示“重新启动容器失败”，请按照如下方式来排查原因。

排查项一：查看端口是否冲突

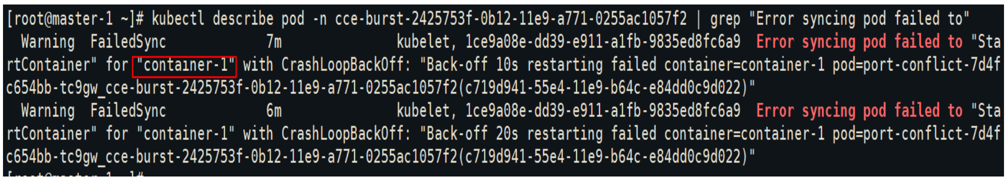
步骤1 按照**使用kubectl**配置好kubectl。

步骤2 在页面上单击失败的工作负载，进入负载详情界面，查看Pod列表，获取Pod名字。

步骤3 查看失败的容器的名称。

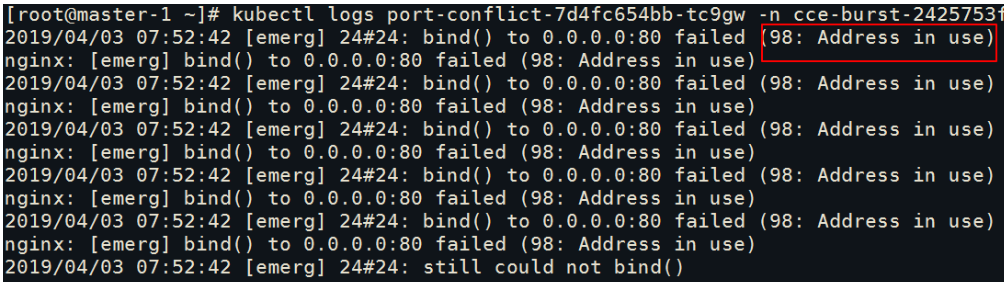
kubectl describe pod $name -n $namespace | grep "Error syncing pod failed to"

图 3-4 查看失败的容器的名称


```
[root@master-1 ~]# kubectl describe pod -n cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2 | grep "Error syncing pod failed to"
  Warning  FailedSync            7m               kubelet, 1ce9a08e-dd39-e911-a1fb-9835ed8fc6a9  Error syncing pod failed to "Sta
rtContainer" for "container-1" with CrashLoopBackOff: "Back-off 10s restarting failed container=container-1 pod=port-conflict-7d4f
c654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
  Warning  FailedSync            6m               kubelet, 1ce9a08e-dd39-e911-a1fb-9835ed8fc6a9  Error syncing pod failed to "Sta
rtContainer" for "container-1" with CrashLoopBackOff: "Back-off 20s restarting failed container=container-1 pod=port-conflict-7d4f
c654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
```

步骤4 查看退出容器的错误日志。

kubectl logs $podName -n $namespace -c $containerName


```
[root@master-1 ~]# kubectl logs port-conflict-7d4fc654bb-tc9gw -n cce-burst-2425753f
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: bind() to 0.0.0.0:80 failed (98: Address in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address in use)
2019/04/03 07:52:42 [emerg] 24#24: still could not bind()
```

此种问题有如下解决方法：重新创建工作负载，并配置正确的端口，确保端口不冲突。

----结束

排查项二：用户自身业务 BUG

请检查工作负载启动命令是否正确执行，或工作负载本身bug导致容器不断重启。

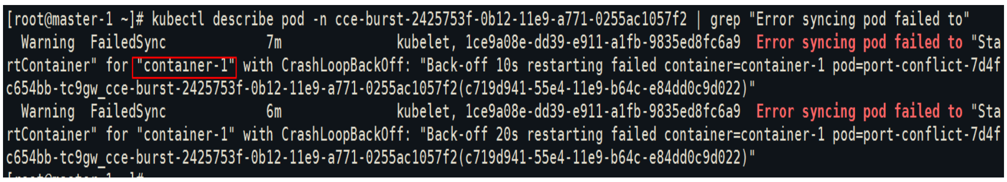
步骤1 按照**使用kubectl**配置好kubectl。

步骤2 在页面单击失败的工作负载，进入负载详情界面，查看Pod列表，获取Pod名字。

步骤3 查看失败的容器的名称。

kubectl describe pod $name -n $namespace | grep "Error syncing pod failed to"

图 3-5 查看失败的容器的名称

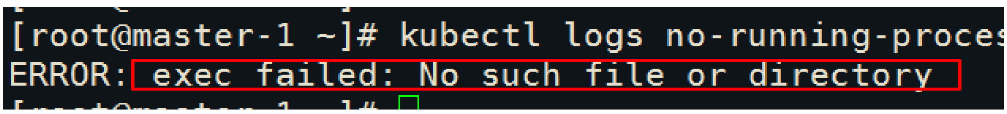

```
[root@master-1 ~]# kubectl describe pod -n cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2 | grep "Error syncing pod failed to"
  Warning  FailedSync            7m               kubelet, 1ce9a08e-dd39-e911-a1fb-9835ed8fc6a9  Error syncing pod failed to "Sta
rtContainer" for "container-1" with CrashLoopBackOff: "Back-off 10s restarting failed container=container-1 pod=port-conflict-7d4f
c654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
  Warning  FailedSync            6m               kubelet, 1ce9a08e-dd39-e911-a1fb-9835ed8fc6a9  Error syncing pod failed to "Sta
rtContainer" for "container-1" with CrashLoopBackOff: "Back-off 20s restarting failed container=container-1 pod=port-conflict-7d4f
c654bb-tc9gw_cce-burst-2425753f-0b12-11e9-a771-0255ac1057f2(c719d941-55e4-11e9-b64c-e84dd0c9d022)"
```

步骤4 查看退出容器的错误日志。

kubectl logs $podName -n $namespace -c $containerName

根据日志提示修复工作负载本身的问题。

图 3-6 容器启动命令配置不正确


```
[root@master-1 ~]# kubectl logs no-running-proces
ERROR: exec failed: No such file or directory
```

此种问题的解决方案是：重新创建工作负载，并配置正确的启动命令。

----结束

排查项三：工作负载配置的健康检查执行失败

工作负载如果配置liveness型（工作负载存活探针）健康检查，当健康检查失败次数超过阈值时，会重启实例中的容器。在工作负载详情页面查看事件，如果K8S事件中出现“Liveness probe failed: ……”时，表示健康检查失败。请重新配置正确的健康检查策略。

排查项四：命名空间的资源类型错误

请检查创建命名空间时选择的资源类型是否正确，通用计算型和GPU加速型支持X86镜像。

步骤1 登录控制台，在页面上单击失败的工作负载，进入负载详情界面。

步骤2 查看Pod列表，单击实例异常Pod所在行“操作”列的“查看日志”。

步骤3 查看报错信息如下。

ERROR: exec failed: Exec format error

ERROR: hyper send process initiated event: error

----结束

其他定位方法

负载异常有可能是容器中运行的应用启动异常，这时可以通过手动执行启动命令，根据错误提示进行问题定位和修复。通常的做法如下：

1. 为工作负载配置如下类型的启动命令，这样pod启动后没有启动应用程序，没有执行任何操作。

说明

执行启动命令前，请确认镜像中**/bin/bash**命令可用。

2. Pod启动成功后，执行kubectl exec进入到pod中，手动执行启动命令，根据错误提示进行问题定位和修复。

4 容器工作负载类

4.1 为什么业务运行性能不达预期?

由于CCI服务底层资源是多租户共享的，为了保障用户业务稳定，CCI服务底层对于磁盘IO等是有流控限制的。体现在容器内，主要影响是负载对根目录rootfs的读写、负载标准日志输出数据量都会受到一定限制。如果您的业务运行性能不达预期，可以从以下几个可能的原因进行排查：

原因一：用户业务容器存在日志打印至标准输出，且日志量较大的场景。

详情说明：CCI服务底层会对标准输出的转发进行限流，如果业务的日志量比较大（>1MB/s），则推荐使用日志卷将日志上报到AOM（参考**日志管理**），或者将日志输出到FlexVolume、持久化的EVS卷、SFS卷等，并配合sidecar运行fluentbit等开源组件将日志上报到自建日志中心。对标准输出打印较大的日志量可能会由于转发限流导致业务性能受损。

原因二：用户业务容器内存在对rootfs磁盘的高IO读写的场景。

详情说明：CCI服务会对rootfs的盘进行IO限流，如果业务进程运行中会执行较高的磁盘IO（带宽>6MB/s，iops>1000）或者对磁盘IO性能比较敏感，请不要将大IO的文件操作放在rootfs，例如往容器系统盘（rootfs磁盘）高频打日志，使系统盘频繁读写，您可以将业务相关的配置文件，或者一些读写不频繁的文件放在rootfs磁盘中。大IO的文件操作需要根据业务场景选择随Pod生命周期创删的日志卷、FlexVolume或者持久化的EVS卷、SFS卷等。对rootfs进行大IO操作可能会由于磁盘限流导致业务性能受损。

如果存在上述问题场景，请根据详情说明进行调整，避免业务性能受损。

4.2 如何设置实例（Pod）数?

创建工作负载时

创建负载的时候直接设置实例（Pod）数量。

图 4-1 设置工作负载实例数量

工作负载创建后

在负载详情页“伸缩”部分，选择手动伸缩，单击，调整实例数量。

图 4-2 手动伸缩

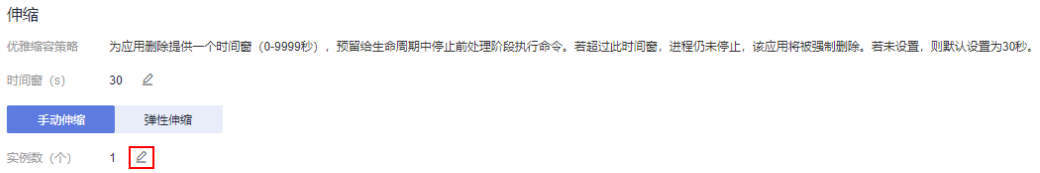
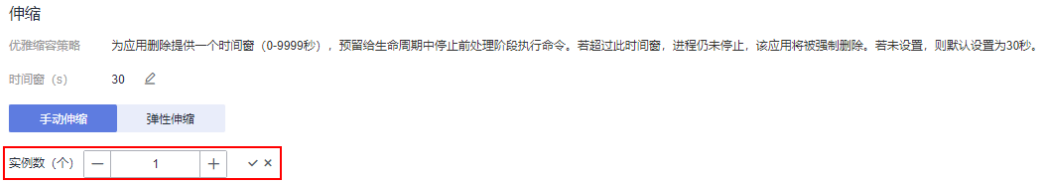


图 4-3 调整实例数



4.3 如何查看资源配额?

云容器实例对单个用户的资源数量和容量限定了配额，配额的详细信息请参见**关于配额**。

步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

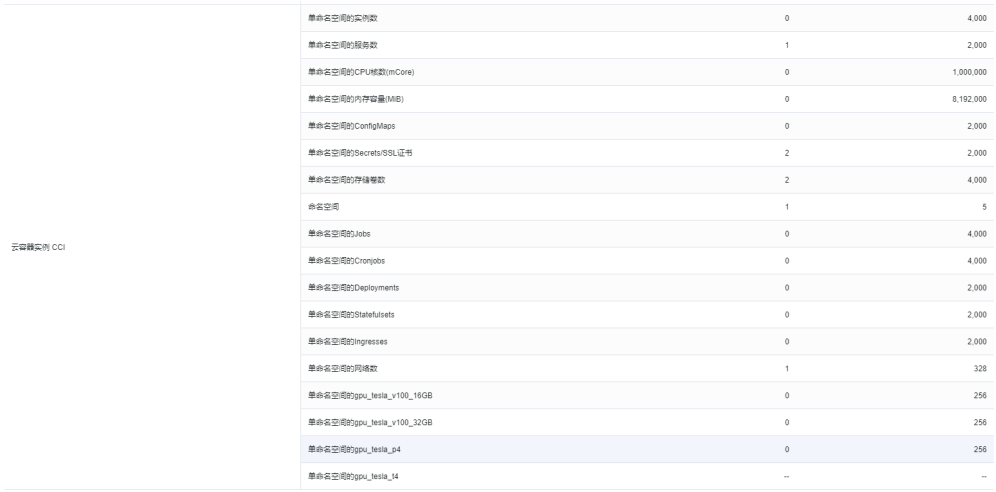
步骤3 在页面右上角，选择“资源 > 我的配额”。

系统进入“服务配额”页面。

图 4-4 我的配额

步骤4 您可以在“服务配额”页面，查看各项资源的总配额及使用情况。

图 4-5 CCI 资源配额



----结束

4.4 如何设置应用的探针?

云容器实例基于Kubernetes，提供了应用存活探针和应用业务探针，您可以在创建工作负载的时候设置，具体请参见**健康检查**。

4.5 弹性伸缩策略如何配置?

云容器实例支持告警、定时、周期三种弹性伸缩策略。具体配置方法请参见**伸缩负载**。

4.6 使用 sample 镜像创建工作负载无法运行

当您使用过容器镜像服务（SWR）但没有上传过镜像时，容器镜像服务会为您预置一个名为sample的镜像，该镜像无法运行，建议您直接使用开源镜像中心的镜像创建负载。

4.7 调用接口删除 Deployment 后怎么还能查看到 Pod?

Deployment接口提供级联删除Pod的选项propagationPolicy，可以设置propagationPolicy的值为Orphan、Foreground和Background，具体请参见**删除Deployment**。

4.8 为什么 exec 进入容器后执行 GPU 相关的操作报错?

问题现象：

exec进入容器后执行GPU相关的操作（例如nvidia-smi、使用tensorflow运行GPU训练任务等）报错“cannot open shared object file: No such file or directory”。

```
root@cc1-deployment-20196201-7ffcbd6477-khj8d:/workspace/examples/image-classification# python train_imag
Traceback (most recent call last):
  File "train_imagenet.py", line 22, in <module>
    from common import find_mxnet, data, dali, fit
  File "/workspace/examples/image-classification/common/find_mxnet.py", line 20, in <module>
    import mxnet as mx
  File "/opt/mxnet/python/mxnet/__init__.py", line 24, in <module>
    from .context import Context, current_context, cpu, gpu, cpu_pinned
  File "/opt/mxnet/python/mxnet/context.py", line 24, in <module>
    from .base import classproperty, with_metaclass, _MXClassPropertyMetaClass
  File "/opt/mxnet/python/mxnet/base.py", line 212, in <module>
    _LIB = _load_lib()
  File "/opt/mxnet/python/mxnet/base.py", line 203, in _load_lib
    lib = ctypes.CDLL(lib_path[0], ctypes.RTLD_LOCAL)
  File "/usr/lib/python3.5/ctypes/__init__.py", line 347, in __init__
    self._handle = _dlopen(self._name, mode)
OSError: libcuda.so.1: cannot open shared object file: No such file or directory
```

问题原因：

安全容器内的cuda库位置为/usr/local/nvidia/lib64，您需要添加/usr/local/nvidia/lib64到LD_LIBRARY_PATH，才能正确地找到cuda库。

解决方法：

使用kubectl exec或者前端console登录进入带GPU的容器时，先执行命令**export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/nvidia/lib64**，然后再执行其他GPU相关的操作命令。

4.9 使用 CCI 集群，在容器内部执行 systemctl 命令，需要以特权模式（ --privileged=true ）来启动容器是否支持?

目前，CCI尚不支持特权模式。

特权容器主要场景是让容器共享、操作宿主机的设备，CCI是基于kata的hypervisor虚拟化级别隔离，所以宿主机的资源对容器完全隔离。

其他场景，推荐通过k8s原生SecurityContext中更细粒度的权限策略来控制，按需使用，保障客户容器运行环境安全可靠。

4.10 Intel oneAPI Toolkit 运行 VASP 任务，为什么概率性运行失败?

Intel oneAPI Toolkit（Intel并行计算平台）运行的VASP（用于电子结构计算和量子力学-分子动力学模拟）任务对CPU硬件版本有深度依赖，在小规格Pod场景下概率性运行失败，建议切换oneAPI版本或使用4核以上Pod运行。

4.11 容器实例被驱逐

如果节点超负荷运行，无法承受的话，系统会自动清理一些任务。

建议跟据实际使用量来申请资源。

4.12 为什么界面不显示工作负载终端?

工作负载终端，显示异常。

请先排除欠费因素。

4.13 删除工作负载后，会持续扣费。

创建的工作负载，在删除后，“我的资源”中显示依然存在，且会持续扣费，建议您确认下该命名空间下是否还有Pod在运行，如需停用，请删除Pod，后台会停止计费。

5 镜像仓库类

5.1 公有镜像是否可以导出?

您无法导出或下载其他用户上传的镜像。

5.2 如何制作容器镜像?

本节指导用户通过Dockerfile定制一个简单的Web工作负载程序的容器镜像。

背景信息

如果使用官方的Nginx镜像来创建容器工作负载，在浏览器访问时则会看到默认的Nginx欢迎页面，本节以Nginx镜像为例，修改Nginx镜像的欢迎页面，定制一个新的镜像，将欢迎页面改为“Hello, CCI!”。

操作步骤

步骤1 以root用户登录容器引擎所在的虚拟机。

步骤2 创建一个名为Dockerfile的文件。

mkdir mynginx

cd mynginx

touch Dockerfile

步骤3 编辑Dockerfile。

vi Dockerfile

文件内容如下：

```
FROM nginx
RUN echo '<h1>Hello,CCI!</h1>' > /usr/share/nginx/html/index.html
```

其中：

- FROM语句：表示使用nginx镜像作为基础。

- RUN语句：表示执行echo命令，在显示器中显示一段Hello,CCI!的文字。

步骤4 构建容器镜像。

docker build -t nginx:v3 .

步骤5 执行以下命令，可查看到已成功部署的nginx镜像，版本为v3。

docker images

----结束

5.3 如何上传镜像?

镜像的管理是由容器镜像服务（SoftWare Repository）提供的，当前容器镜像服务提供如下两种上传镜像的方法：

- **客户端上传镜像**
- **页面上传镜像**

5.4 CCI 是否提供基础容器镜像的下载服务?

CCI中的镜像仓库是由容器镜像服务（SoftWare Repository）提供，容器镜像服务提供基础容器镜像的下载。

5.5 CCI Administrator 有上传镜像包的权限吗?

当前在CCI中上传镜像使用的是华为云的“SWR容器镜像服务”。

您还需要为账号添加**Tenant Administrator**权限。

5.6 CCI 上传镜像包需要开通什么权限?

当前在CCI中上传镜像使用的是华为云的“SWR容器镜像服务”。

需要为账号添加**Tenant Administrator**权限。SWR权限的详细内容可参考**SWR权限**。

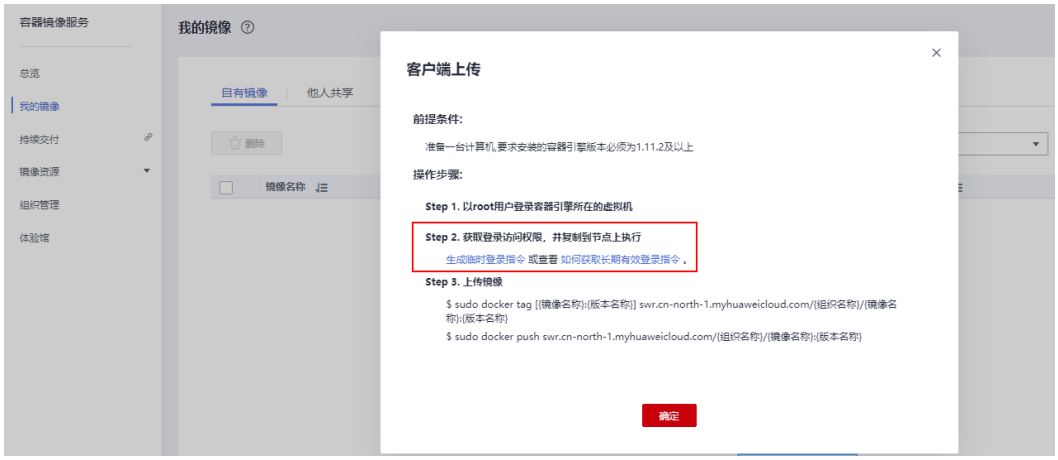
上传镜像的具体步骤可参考**客户端上传镜像**或**页面上传镜像**。

5.7 CCI 上传镜像时提示需要认证怎么办?

当前在CCI中上传镜像使用的是“SWR容器镜像服务”。

使用SWR上传镜像，您需要先获取访问权限，请参见下图。如果需了解上传镜像详细步骤，请参见**客户端上传镜像**。

图 5-1 上传镜像



6 网络管理类

6.1 如何查看虚拟私有云 VPC 的网段?

在“虚拟私有云”页面，可查看虚拟私有云的“名称/ID”和“VPC网段”。用户可以调整已创建的VPC或通过重新创建VPC调整网段。

图 6-1 查看 VPC 网段

6.2 CCI 是否支持负载均衡?

CCI支持负载均衡，当前在CCI创建工作负载的访问设置页面中，内网访问（**使用私网ELB访问**）和公网访问中的配置都是负载均衡方式。

通常所说的负载均衡一般指的是公网负载均衡，CCI对接负载均衡服务。

通过CCI创建工作负载时，在设置访问设置的页面，可以根据需要选择内网访问和外网访问，然后配置负载均衡。

1. 公网访问负载均衡，请参见**公网访问**。
2. 内网访问负载均衡，请参见**内网访问**。

6.3 CCI 如何配置 DNS 服务?

如果用户负载需要使用k8s内部域名解析，则需要安装coredns插件。此时pod的dnsPolicy需要设置为ClusterFirst。

在插件市场界面可以单击，将coredns插件安装在指定的namesapce下。

图 6-2 创建插件

如果用户负载不需要k8s内部域名解析服务，但是需要使用域名解析服务，此时pod的dnsPolicy需要设置为Default。

除了以上两种配置方式用户还可以通过设置dnsPolicy为None使用自定义dns服务。yaml示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
    - name: test
      image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
      - 1.2.3.4
    searches:
      - ns1.svc.cluster-domain.example
      - my.dns.search.suffix
    options:
      - name: ndots
        value: "2"
      - name: edns0
```

6.4 CCI 是否支持高速 IB（Infiniband）网络?

不支持。

6.5 如何从公网访问容器?

负载支持绑定ELB。用户可以给负载绑定ELB实例，通过ELB的地址从外部访问容器负载。具体操作请参见**公网访问**。

如果您使用kubectl，您也可以参见**Service**和**Ingress**，创建Service和Ingress对象，绑定ELB。

6.6 如何从容器访问公网?

您可以使用公有云平台提供的NAT网关服务。该服务能够为虚拟私有云内的容器提供网络地址转换（Network Address Translation）服务，使虚拟私有云内的容器可以共享使用弹性公网IP访问Internet。通过NAT网关的SNAT功能，可以直接访问Internet，提供超大并发数的连接服务，适用于请求量大、连接数多的服务。详细信息请参见**从容器中访问公网**。

6.7 如何处理公网无法访问负载?

1. 公网能正常访问的前提是负载已处于运行中状态，如果您的负载处于未就绪或异常状态，公网访问将无法正常使用。
2. 从负载开始创建到公网可以正常访问可需要1分钟到3分钟的时间，在此时间内网络路由尚未完成配置，请稍作等待。
3. 负载创建3分钟以后仍然无法访问。在“工作负载 -> 查看您创建的负载详情-> 选择访问配置 -> 选择访问事件”，查看访问事件，查看是否有告警事件。如下两种情况为无法访问公网的事件。
   - Listener port is repeated：ELB监听器端口重复，是由于之前发布公网访问的负载，删除之后立刻创建使用相同ELB端口的公网访问负载，ELB实际删除端口需要一定的时间，等待5-10分钟，公网访问可正常使用。
   - Create listener failed：创建ELB监听器失败，创建监听器失败的原因一般是超过配额限度，请选择其他配额充足的ELB实例。
4. 负载创建3分钟以后仍然无法访问，且无告警事件，可能原因是用户配置的容器端口实际上没有相应进程在监听，目前云容器实例服务无法检测出该类使用异常，需要您排查镜像是否有监听该容器端口。如果容器端口监听正确，此时无法访问的原因可能为ELB实例本身有问题，请排查ELB实例状态。

6.8 负载访问 504 问题定位思路

负载访问504问题一般是因为ELB实例绑定的Port到后端 CCI 负载Pod的安全组没有放通。查看CCI负载Pod使用的安全组，确保安全组规则放通ELB实例绑定的Port。

Pod绑定的安全组可以通过查看负载对应Network获得，调用**Network**接口，响应里面metadata.annotations中的network.alpha.kubernetes.io/default-security-group即为安全组ID，如下所示。

```
{
   "kind": "Network",
   "apiVersion": "networking.cci.io/v1beta1",
   "metadata": {
      "name": "namespace-test-dc1-default-network",
      "namespace": "namespace-test",
      "selfLink": "/apis/networking.cci.io/v1beta1/namespaces/namespace-test/networks/namespace-test-dc1-default-network",
      "uid": "6fb85414-af6b-11e8-b6ef-f898ef6c78b4",
      "resourceVersion": "5016899",
      "creationTimestamp": "2018-09-03T11:21:00Z",
      "annotations": {
         "network.alpha.kubernetes.io/project-id": "51bf52609f2a49c68bfda3398817b376",
         "network.alpha.kubernetes.io/default-security-group": "19c5d024-aed5-4856-b958-c0f65ce70855",
         "network.alpha.kubernetes.io/domain-id": "aadb43c0b14c4cafbccfff483d075987"
      },
      "enable": true
```



```
    },
    "spec": {
        "cidr": "192.168.244.0/23",
        "attachedVPC": "0d4080e5-546a-46c4-86fe-f3e26d685177",
        "networkType": "underlay_neutron",
        "physicalNetwork": "phy_net0",
        "networkID": "0022e356-f730-4226-802e-9cdaa6e7da17",
        "subnetID": "1ffd839d-e534-4fa8-a59d-42356335bf74",
        "availableZone": "cnnorth1a"
    },
    "status": {
        "state": "Active"
    }
}
```


进入网络控制台，根据上述操作中已获取的安全组ID，查找对应的安全组。

单击安全组名称，在安全组的入方向规则中增加如下规则。

须知

UDP类型的公网访问，健康检查依赖ICMP规则，请注意添加。

协议端口	类型	源地址
全部	IPv4	sg-test
ICMP : Echo	IPv4	0.0.0.0/0
TCP : 22	IPv4	0.0.0.0/0
TCP : 3389	IPv4	0.0.0.0/0
TCP : 1-65535	IPv4	0.0.0.0/0
UDP : 1-65535	IPv4	0.0.0.0/0

6.9 如何解决 Connection timed out 问题?

问题现象：

CCI实例创建正常，使用python django smtp服务发送邮件时，一直提示“[Errno 110] Connection timed out”错误。

问题原因：

- 客户仅购买了ELB服务未购买NAT网关服务，故只能从外部访问容器。购买NAT网关后，才可以从容器内部访问外部网络。
- 为了确保良好的网络安全环境，华为云对25端口对外发送做了限制。

解决方法：

- 方法一：使用**NAT网关服务**，该服务能够为VPC内的容器实例提供网络地址转换（Network Address Translation）服务，SNAT功能通过绑定弹性公网IP，实现私有IP向公有IP的转换，可实现VPC内的容器实例共享弹性公网IP访问Internet。详情请参见**从容器访问公网**。
- 方法二：联系技术人员，放通客户新申请的弹性公网IP的25端口。

7 存储管理类

7.1 CCI 支持的云存储有哪些，哪种存储需要设置备份?

当前CCI支持使用如下几种云存储：

- 云硬盘（ EVS ）
- 文件存储（ SFS ）
- 极速文件存储（ SFS Turbo ）
- 对象存储（ OBS ）

其中云硬盘存储需要人工配置备份策略，详情请参见**云硬盘备份管理**。

7.2 如何使用云存储?

如果您使用控制台界面创建，请参见**存储概述**。

您还可以使用kubectl创建云存储，具体请参见**使用PersistentVolumeClaim申请持久化存储**。

7.3 如果不挂载云存储的话，容器运行产生的数据存储在哪里?

如果没有挂载EVS等磁盘，应用数据存储在容器的物理机磁盘，每个Pod存储空间限制为CPU物理机磁盘为20G，GPU物理机磁盘为20G，如果为专属节点可根据客户需求进行调整。

说明

为了确保数据的安全性，在创建容器时容器引擎会从devicemapper获取虚拟化盘，其他容器引擎无法访问。

7.4 job 的 pod 已经执行完成的情况下，为什么依然有实例在挂卷等事件，并且事件信息是失败的?

问题现象：

job的Pod已经执行完成的情况下，依然有实例在挂卷等事件，并且事件信息是失败的。

图 7-1 问题截图

问题原因：

各种类型的Pod（Deployment/StatefulSet/Job/CronJob）在Node上启动时：

- 由kubelet针对该Pod创建podWorker（独立协程）负责检测Pod与关联volume的挂载情况：每隔0.3s检测当前Pod所需挂载的volume都已经挂载成功，检测超时为2min3s；如果检测周期中以及最终超时到达时Pod关联volume都没有检测到挂载成功，则上报事件“Unable to mount volumes for pod …”。
- 由kubelet中VolumeManager（独立协程）负责具体实施Pod关联volume的挂载操作。

对于long running的Pod（Deployment/StatefulSet），除了类似镜像拉取失败、存储挂载失败、容器网络分配失败、当前节点CPU/Mem不满足Pod的实际使用要求等异常场景外，Pod容器如果最终都会启动成功时，上述podWorker在几次周期后都会判定挂载成功。

而对于短时运行的Pod（Job/CronJob），由于容器中业务存在正常退出（如问题场景的GCS Demo job只执行一些echo和ls命令，总体耗时1s不到），就存在短时Pod运行退出时如果刚好在两次podwork检测volume挂载周期中，那么就会出现本问题单所述的误报，但是不影响业务使用，且实际的Job业务还是会运行超过上述时间的。

当前kubelet上述能力属于社区挂载框架既有能力。

解决方法：

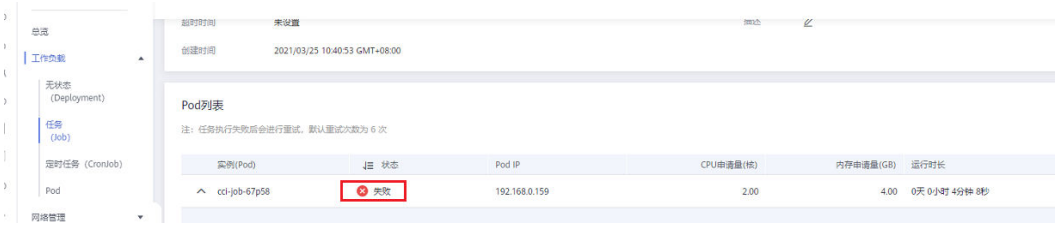
针对短时运行的Pod（Job/CronJob），可能存在由于运行时间过短而误报卷挂载超时的情况，如果这类短时运行任务属于正常退出，则该误报对业务没有影响可以忽略。

7.5 使用 OBS 存储挂载失败

问题现象：

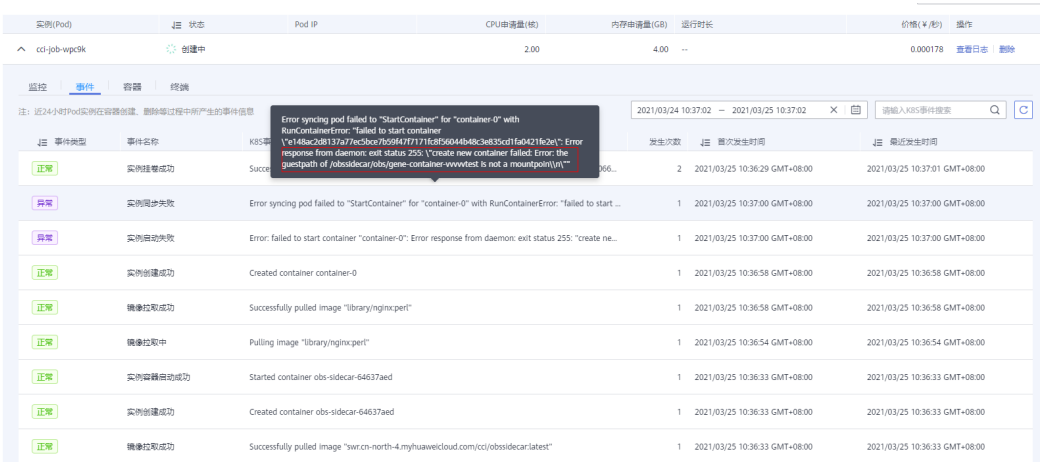
步骤1 单击“工作负载 > 任务 ”，实例状态显示“失败”。

图 7-2 Pod 状态



步骤2 单击“事件”页，查看异常事件，如下图：

图 7-3 事件类型异常

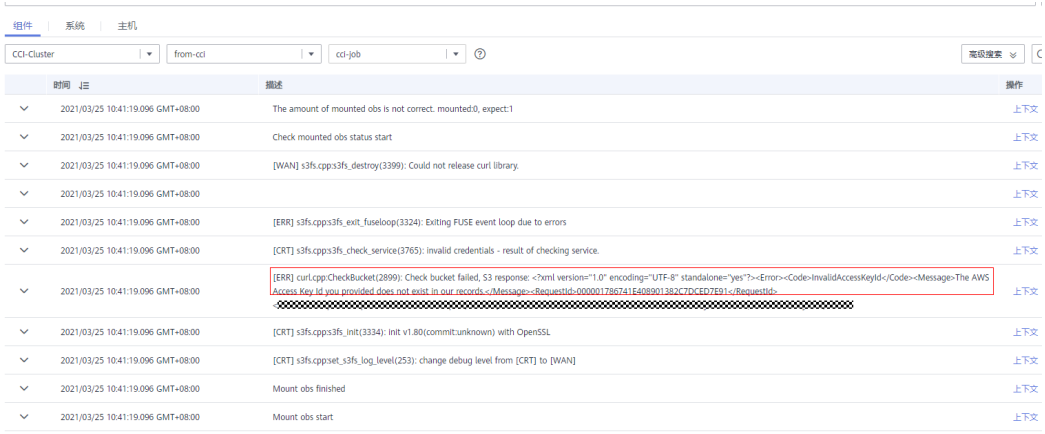


步骤3 在Pod列表，单击失败实例后的“查看日志”，跳转到应用运维管理AOM界面。

图 7-4 Pod 列表

步骤4 在应用运维管理AOM界面，单击“日志 > 日志搜索”选择组件，查看错误信息。

图 7-5 查看日志



----结束

问题原因：

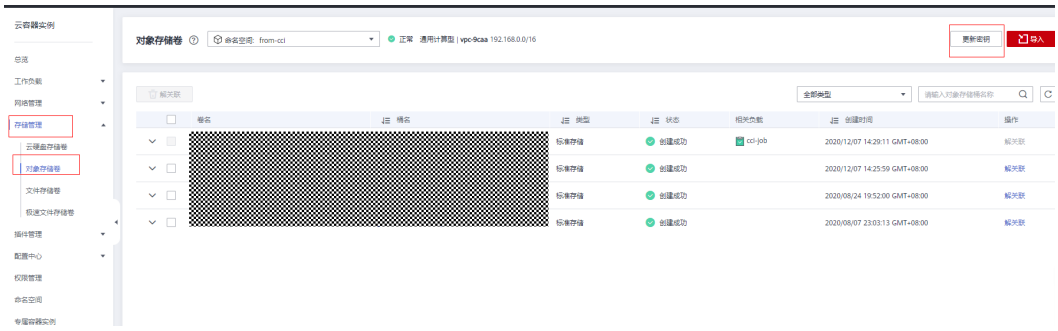
上传的AK/SK失效。

解决方法：

重新上传有效的AK/SK。

步骤1 在云容器实例CCI控制台，单击“存储管理 > 对象存储卷”， 单击“更新密钥”，重新上传有效的AK/SK。

图 7-6 更新密钥



----结束

8 日志采集类

8.1 日志出现重复/丢失的原因

日志出现重复

原因一：日志文件转储，且转储文件仍被匹配到。

详细说明：如果配置日志路径文件名中有通配符，如配置为/tmp/*.log，当/tmp/test.log文件转储为/tmp/test.001.log后，因仍被通配规则匹配到，会被视为新文件，则会被重新采集。

日志出现丢失

原因一：日志文件的存在时间小于5秒。

详细说明：CCI感知日志文件的周期为5秒。日志文件从创建到被删除或重命名的时间小于5秒，则可能该日志文件不会被采集。

9 账户类

9.1 账户有余额，仍提示欠费

问题现象：

账户重置后有余额，在CCI中新建命名空间仍然提示欠费，无法操作。

解决方法：

退出账号，重新登录或清理浏览器缓存。

9.2 资源无法删除

问题现象：

资源无法删除。

解决方案：

当前账号欠费，账号欠费会导致部分资源权限受限，需要补交欠费后在删除相关资源。