天翼云 容器实例

用户使用指南

中国电信股份有限公司云计算分公司

目 录

目录

1 产品介绍 *4*

1.1 产品定义 **4**

1.2 产品优势 **4**

1.3 产品功能 **5**

1.4 应用场景 **6**

1.4 基本概念 **7**

1.5 使用限制 **9**

2 快速入门 *10*

3 操作指南 *14*

3.1 控制台说明 **14**

3.2 配置中心 **15**

3.2.1 配置项 15

3.2.2 私密凭证 18

3.3 镜像仓库 **24**

3.3.1 创建镜像仓库 24

3.3.2 上传镜像 25

3.3.3 操作仓库 25

3.4 应用 **26**

3.4.1 无状态 . . . 26

3.4.2 容器组 . . . 27

3.5 网络管理 . . . 30

3.5.1 服务 . . . 30

3.5.2 路由 . . . 31

3.6 为应用挂载数据卷 . . . 33

3.7 应用管理运维 . . . 35

3.7.1 应用监控 . . . 35

4 常见问题 . . . 37

4.1 容器实例和容器引擎有什么区别？ . . . 37

4.2 什么是环境变量？ . . . 37

4.3 如果不挂载云存储的话，容器运行产生的数据存储在哪里？ . . . 37

4.4 部署应用有哪几种方式？有什么区别？ . . . 38

4.5 镜像、容器、应用的关系是什么？ . . . 38

5 参考知识 . . . 40

5.1 ConfigMap配置项要求 . . . 40

1 产品介绍

1.1 产品定义

容器实例（CT-CCI Cloud Container Instance，简称 CCI）提供 Serverless 容器运行服务，无需创建和管理底层服务器集群，只需打包好容器镜像，即可运行容器。 通过容器实例可以快速部署容器应用，部署应用时可直接运行在容器或容器组（Pod），省去底层资源运维和管理工作，同时支持监控，方便掌控容器运行状态，让您专注应用开发，提升应用开发效率。

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。

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、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。

1.2 产品优势

- **易用：无服务器容器**

  Serverless Container架构，用户只需要提交容器镜像，无需关心底层服务器，不需要预先创建集群和集群维护，提升容器易用性。

- **开放兼容**

原生支持Kubernetes、Docker接口，支持Docker镜像格式，兼容开源社区生态。

- **安全隔离**

  容器应用Pod之间相互隔离防止互相干扰。

- **秒级启动**

  实例快速启动，从容面对突发访问，不需要提前预估集群和业务流量，按需扩容。

- **弹性伸缩**

  支持用户自定义弹性伸缩策略，实现秒级弹性伸缩。

1.3 产品功能

应用运行时全托管

提供无状态负载的运行时托管，保障应用稳定运行。

应用高可用保障

支持一个负载对应多个Pod副本，保障用户业务高可靠。

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的全方位实时监控。

网络管理

支持的服务类型包括：

集群内访问（ClusterIP），它是基于四层TCP和UDP协议转发。对已创建的服务可以进行更新、删除操作。

Ingress（七层负载均衡），基于七层的HTTP和HTTPS协议转发，通过对应的URL将访问流量分发到对应的服务。同时，服务根据不同URL实现不同的功能。对已创建的ingress可以进行更新、删除操作。

配置中心

配置中心包括配置项和私密凭据两个模块。

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，对已创建的配置项进行编辑、删除。

私密凭据是一种用于存储应用所需要认证信息、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，可在容器应用中作为文件或者环境变量使用，对已创建的私密凭据可以删除。

镜像仓库

创建镜像仓库，上传镜像。

用户可创建镜像仓库，仓库分为公有、私有且可以互相转换性质，在仓库列表页可对仓库进行删除操作。

用户进入已创建的仓库助攻可以根据自己的业务需求进行镜像的上传。

1.4 应用场景

- **场景一 在线业务弹性**

  根据业务流量，自动的业务扩容

  根据业务流量自动对业务进行扩容，不需要人工干预，避免流量激增扩容不及时导致系统故障，以及平时大量闲置资源造成的浪费。

  - **简单可控**

    便捷获取计算资源，兼容 Kubernetes。

  - **快速弹性**

    快速启动和弹性伸缩。

  - **按需使用**

    容器实例按需创建释放，不需要预先准备底层资源。

- **场景二 事件驱动的无服务器容器引擎**

  Serverless 容器基础设施

  Serverless 的容器基础设施，通过事件触发，提供高并发、低成本的容器实例调度及事

件处理能力。

- **复杂事件调度**

  提供稳定的运行环境，满足复杂事件的运行诉求。

- **弹性扩展**

  快速启动和弹性伸缩。

1.4 基本概念

容器（Container）

镜像和容器的关系，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。

容器组（Pod)

Pod是Kubernetes创建或部署的最小单位。一个Pod封装一个或多个容器、存储资源、一个独立的网络IP以及管理控制容器运行方式的策略选项。

镜像（Image）

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。

无状态负载（Deployment）

无状态工作负载支持弹性伸缩与滚动升级，适用于实例完全独立、功能相同的场景，如：nginx、wordpress等。一个Deployment可以包含一个或多个Pod，每个Pod的角色相同，所以系统会自动为

Deployment的多个Pod分发请求。Deployment中的所有Pod共享存储卷。

服务（Service）

Pod是有生命周期的，它们可以被创建，也可以被销毁，然而一旦被销毁生命就永远结束。通过Pod Controller能够动态地创建和销毁Pod（例如，需要进行扩缩容，或者执行滚动升级）。每个Pod都会获取它自己的IP地址，但这些IP地址不总是稳定可依赖的。

Service定义了这样一种抽象：一个Pod的逻辑分组，一种可以访问它们的策略（通常称为微服务）。这一组Pod能够被Service访问到，通常是通过Label Selector实现的。

路由（Ingress）

Service和Pod仅可在内部网络中通过IP地址访问，外部的请求需要通过负载均衡转发到Service在Node上暴露的NodePort上，然后再由kube-proxy将其转发给相关的Pod。

Ingress是授权入站连接到达集群服务的规则集合。您可以给Ingress配置外部可访问的URL、负载均衡、SSL、基于名称的虚拟主机等。

配置项（ConfigMap）

ConfigMap用于保存配置数据的键值对，可以用来保存单个属性，也可以用来保存配置文件。

ConfigMap跟Secret很类似，但它可以更方便地处理不包含敏感信息的字符串。

秘钥凭据(Secret)

Secret是Kubernetes中一种加密存储的资源对象，用户可以将认证信息、证书、私钥等保存在密钥中，在容器启动时以环境变量等方式加载到容器中。容器实例基于Kubernetes的负载模型增强了容器安全隔离、负载快速部署、弹性负载均衡、弹性扩缩容等重要能力。容器实例提供Kubernetes原生API，且提供图形化

控制台，让您能够拥有完整的端到端使用体验，使用容器实例前，建议您先了解相关的基本概念。

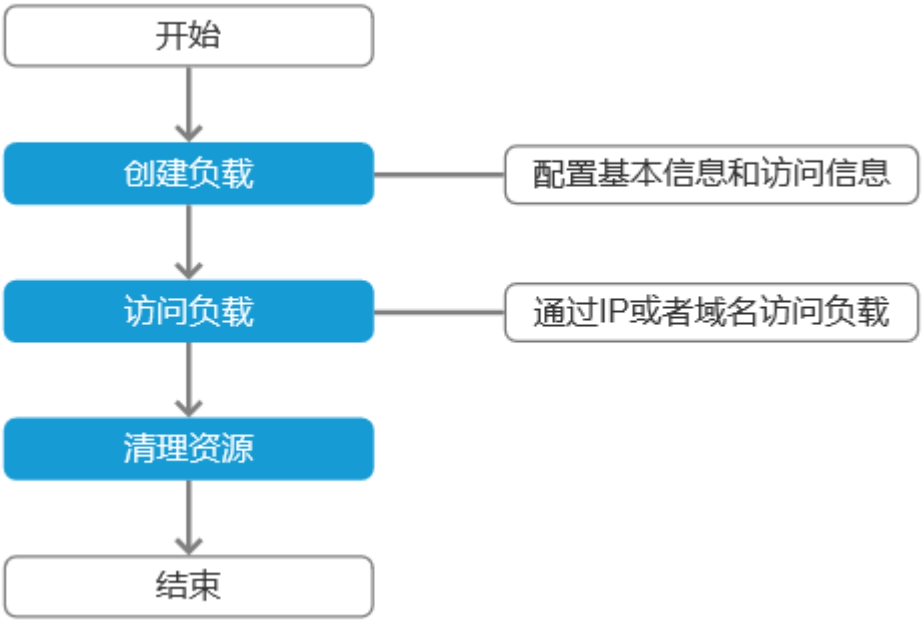
1.5 使用限制

公测期间，用户可创建容器实例的总的资源量不超过8C16G。

2 快速入门

使用 Nginx 创建一个负载

本节以创建一个 Nginx 负载为例介绍如何使用容器实例，步骤如下图所示。



步骤 1：创建负载-配置基本信息

1. 登录容器实例管理控制台。
2. 在左侧导航栏中选择“应用管理 > 无状态（Deployment）”，在右侧页面单击“创建无状态负载”。
3. 添加基本信息。
   - **负载名称**：例如 nginx。
   - **命名空间**：系统给租户分配命名空间。
   - **Pod 数量**：本例中修改 Pod 数量为 1。
   - **Pod 规格**：选择通用计算型，CPU 0.5 核，内存 1GB。

容器配置

在天翼云官方镜像选择 nginx 镜像，使用默认镜像版本。

步骤2：创建负载-配置访问信息

配置负载访问信息。

选择负载访问方式，如下：

- 集群内访问：选择此类型，系统将自动分配一个仅集群内部可以访问的虚拟 IP，供集群内部的容器访问；

  本例中，配置服务名称为“nginx”， 选择配置为“集群内访问”， 容器端口：容器中应用启动监听的端口。访问端口：ClusterIP 的服务端口，映射到容器的应用端口。两端口分别填为 80。

单击“下一步”，进入高级配置中选择升级策略：滚动升级、替换升级。选择滚动升级后。

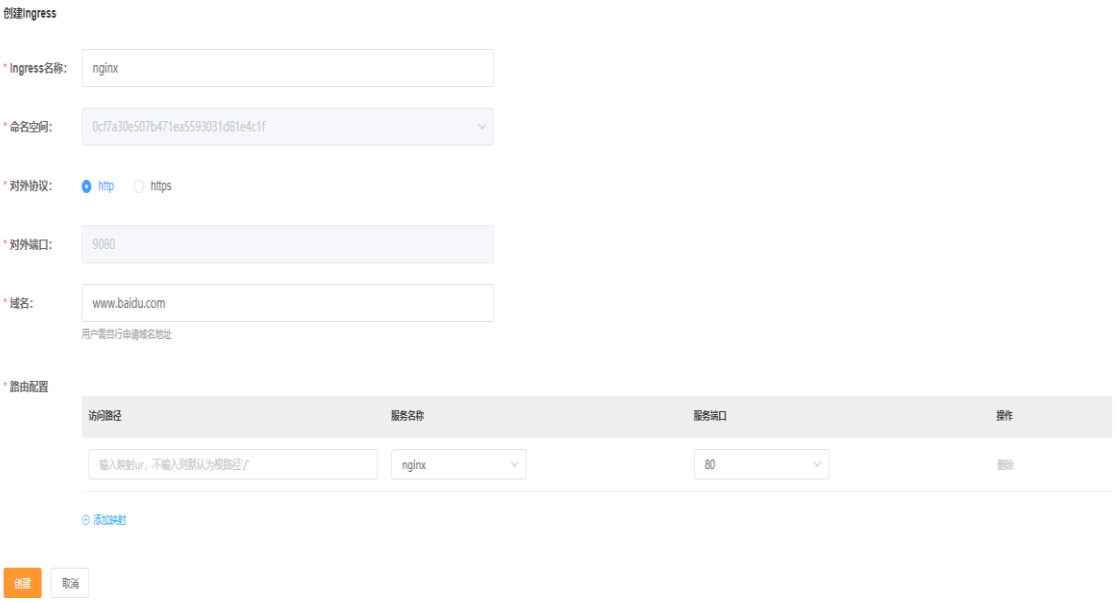
单击”下一步”进入规格确定页，确定所选规格是否正确，如没有问题的话，点击确认下单，“返回无状态负载列表”。

步骤3：访问工作负载

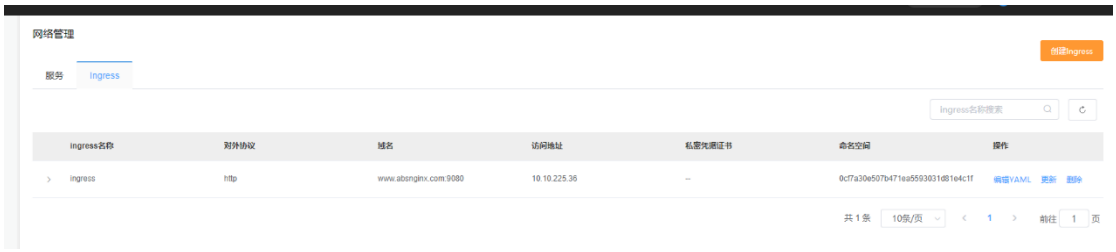
工作负载创建成功后，您可以通过浏览器访问 nginx。

前提条件：需要到网络管理中先配置 Ingress。

配置 Ingress 名称为“nginx”，对外协议为“HTTP”、填写域名，对路由进行设置，其中服务名“nginx”，服务端口为 80。点击创建，用户可以通过域名来访问容器中部署的服务。



1. 单击网络管理，选择 ingess，进入 ingress 列表页。
2. 选择对应的访问地址，并拷贝该访问地址。



3. 在浏览器中访问

3 操作指南

3.1 控制台说明

登录容器实例CCI后，可在概览界面查看说明，在没有订购容器实例的情况下，该步骤说明下的按钮是订购容器实例。页面左侧导航是CCI服务的目录，包括概览、应用管理、配置中心、资源。

应用：

无状态：对镜像进行部署，部署后应用为无状态应用(即kubernetes中Deployments:在运行中始终不保存任何数据或状态，实例之间完全独立、功能相同的特性。如：nginx、wordpress。平台能够为无状态容器应用选取合适的资源进行安装部署，并支持运行管理操作，如伸缩、升级、删除等；

容器组：是Kubernetes部署应用或服务的最小的基本单位。一个容器组封装多个应用容器、存储资源、一个独立的网络IP以及管理控制容器运行方式的策略选项。在容器组列表中可以对容器组进行监控、删除等；

网络管理：

服务：是Kubernetes的基本操作单元，是真实应用服务的抽象，每一个服务后面都有很多对应的容器来提供支持，通过 Kube-Proxy 的 port 和服务 selector 决定服务请求传递给后端的容器，对外表现为一个单一访问接口，外部不需要了解后端如何运行，这给扩展或维护后端带来很大的好处;

路由：Ingress 是授权入站连接到达集群服务的规则集合。你可以通过 Ingress 配置提供外部可访问的 URL、负载均衡、SSL、基于名称的虚拟主机等。用户通过 POST Ingress 资源到 API server 的方式来请求 Ingress。

配置中心：

配置项：是一种用于存储应用所需配置信息的资源类型，内容由用户决定。资源创建完成后，可在容器应用中加载使用。例如，在“数据卷”中加载资源文件，使其成为容器中的文件，或者在“环境变量”中加载，使其成为容器中的环境变量；

私密凭据：是一种用于存储应用所需要认证信息、密钥的敏感信息等的资源类型，内容由用户定。资源创建完成后，可在容器应用中作为文件或者环境变量使用。

镜像仓库：是用户订购容器实例后，在一段时间内所拥有得资源以及数量，以及部署应用后所占用得资源大小。

3.2 配置中心

3.2.1 配置项

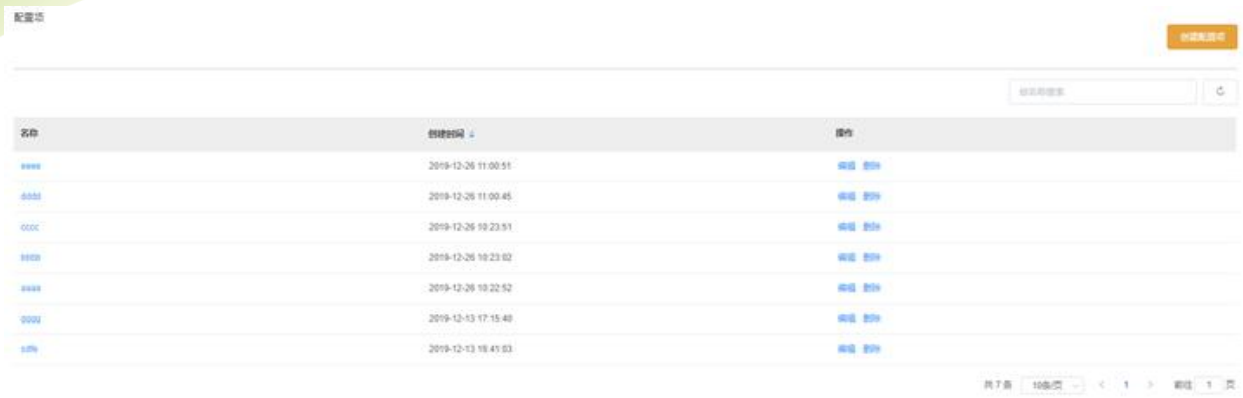
创建配置项：

配置项是一种用于存储应用所需配置信息的资源类型，内容由用户决定。资源创建完成后，可在容器应用中加载使用。例如，在“数据卷”中加载资源文件，使其成为容器中的文件，或者在“环境变量”中加载，使其成为容器中的环境变量。

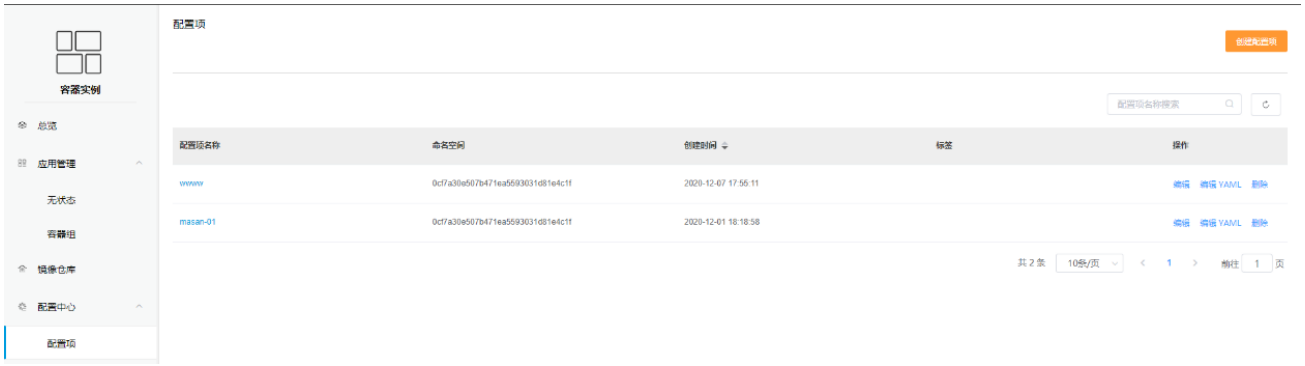
操作步骤：

1） 在控制中心中，选择【计算】【容器实例】，进入容器实例界面；

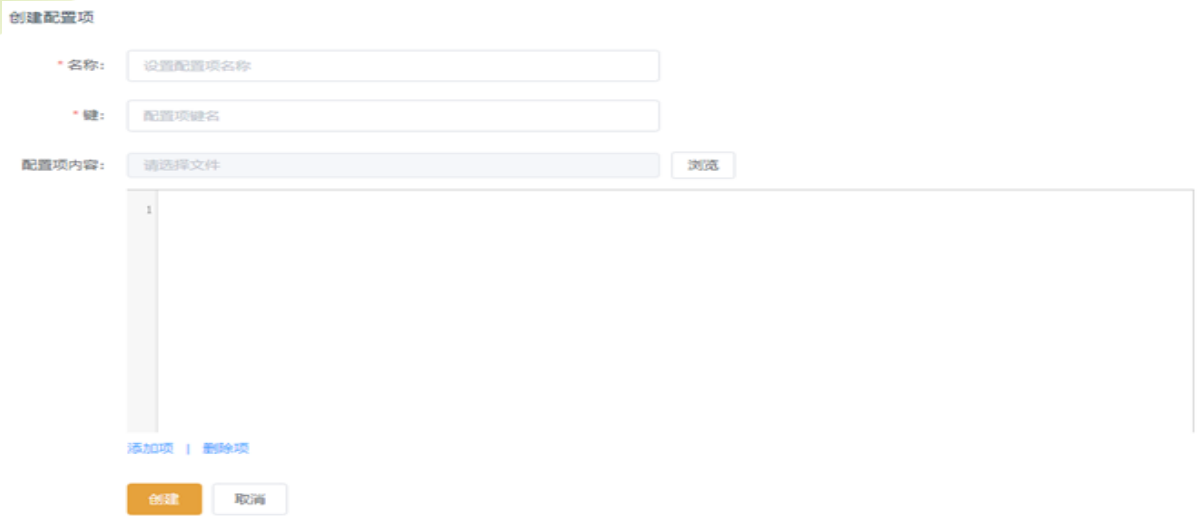
2） 单击左侧导航栏的【配置中心】【配置项】，单击【创建配置项】；



3）参照下表设置新增配置参数，其中带“*”标志的参数为必填参数。输入完成后，单击【创键】；



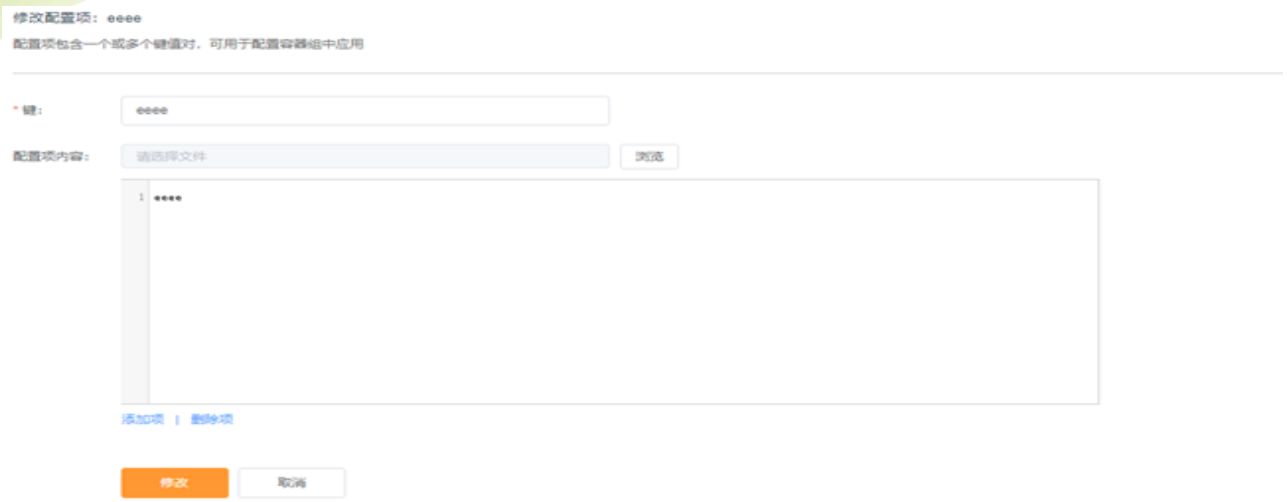
参数	参数说明
*名称	新建配置项的名称，同一个命名空间里命名必须唯一。
添加项	增加一对键、配置项内容。
键	配置项的键值。键值只能由字母、数字、句点、连字符和下划线组成
配置项内容	配置项的内容，通过上传文件来表示。



4）创建成功后，服务列表中会出现已创建的配置项，对列表中的配置项可以编辑、删除的操作。在详情页中可以对该配置项进行编辑YAML、修改、删除的操作。

编辑YAML，如图：

修改配置项，如图：



3.2.2 私密凭证

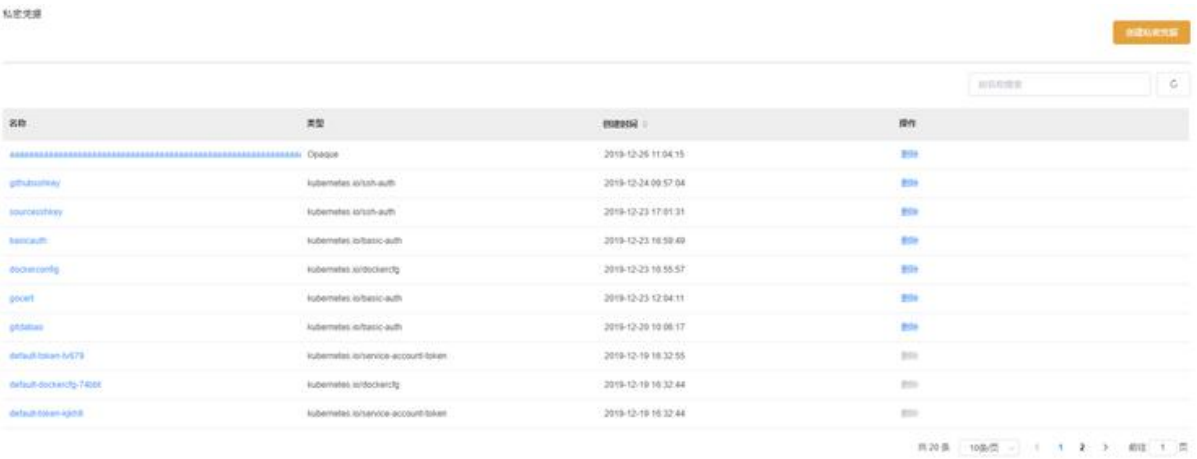
3.2.2.1 创建私密凭证

私密凭据是一种用于存储应用所需要认证信息、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，可在容器应用中作为文件或者环境变量使用。

操作步骤：

1） 在控制中心中，选择【计算】【容器实例】，进入容器实例界面；

2） 单击左侧导航栏的【配置中心】【私密凭据】，单击【创建私密凭据】；

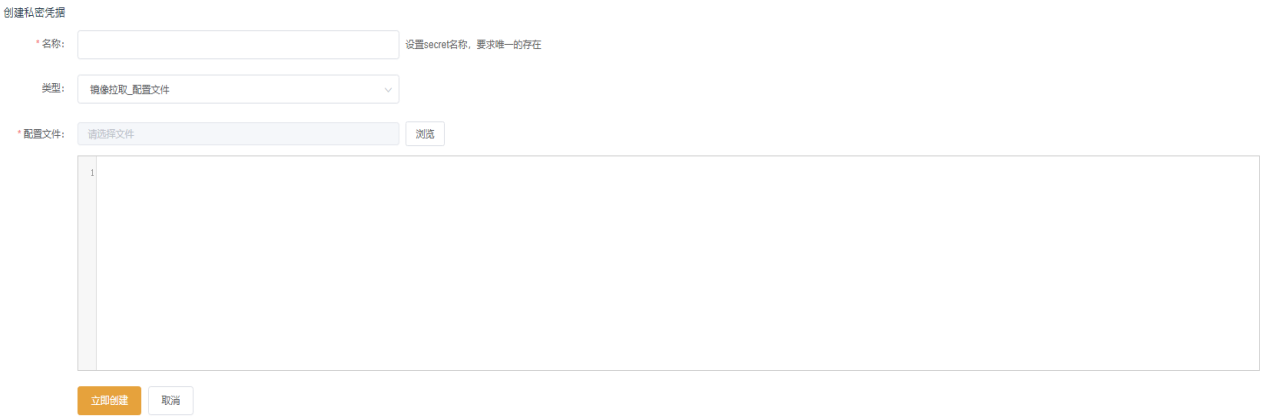


3） 参照下表

设置新增私密凭证参数，其中带“*”标志的参数为必填参数。输入完成后，单击【创建】；

参数	参数说明
*名称	新建私密凭据的名称，必须唯一。
*类型	Opaque：一般密钥类型。 镜像拉取_认证信息：存放拉取私有仓库镜像所需的认证信息。 镜像拉取_配置文件：存放拉取私有仓库镜像所需的配置文件。 镜像拉取_基础认证：存放服务所需的证书。 源码拉取_SSH Key：通过密钥进行拉取代码。
类型为：Opaque	
添加项	增加一对键、配置项内容。
键	配置项的键值。键值只能由字母、数字、句点、连字符和下划线组成。
配置项内容	配置项的内容，通过上传文件来表示。

参数	参数说明
类型为：镜像拉取_认证信息	
镜像仓库地址	输入镜像仓库的地址。
用户名	该镜像仓库的用户名称。
密码	用户名称所对应的的密码。
邮箱	该用户的邮箱地址，必须为user@domain格式。



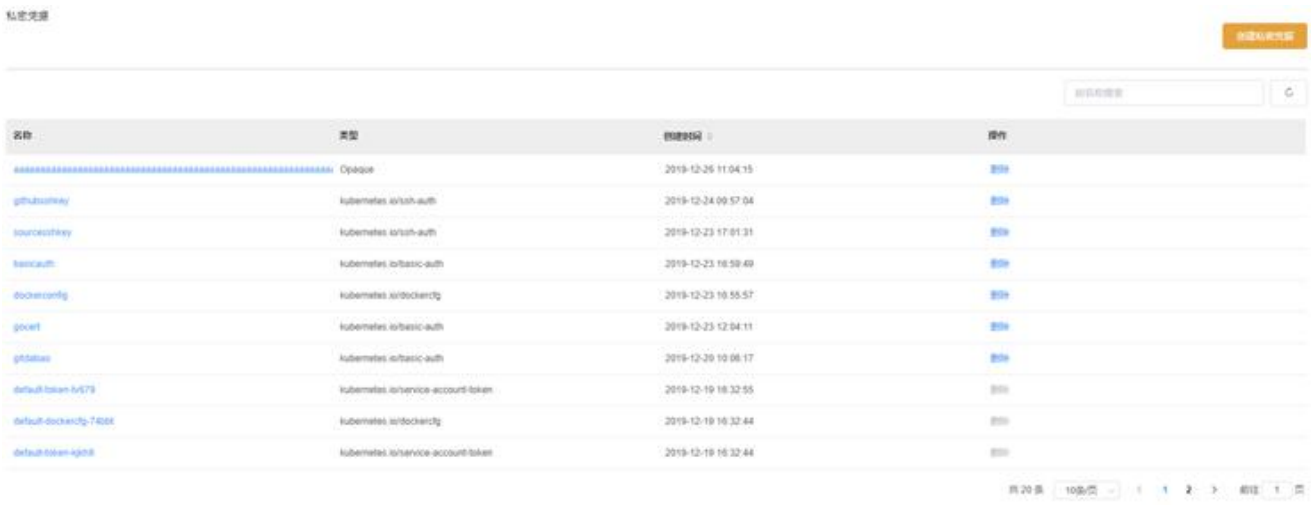
3.2.2.2 创建私密凭证

私密凭据是一种用于存储应用所需要认证信息、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，可在容器应用中作为文件或者环境变量使用。

操作步骤：

1） 在控制中心中，选择【计算】【容器实例】，进入容器实例界面；

2） 单击左侧导航栏的【配置中心】【私密凭据】，单击【创建私密凭据】；



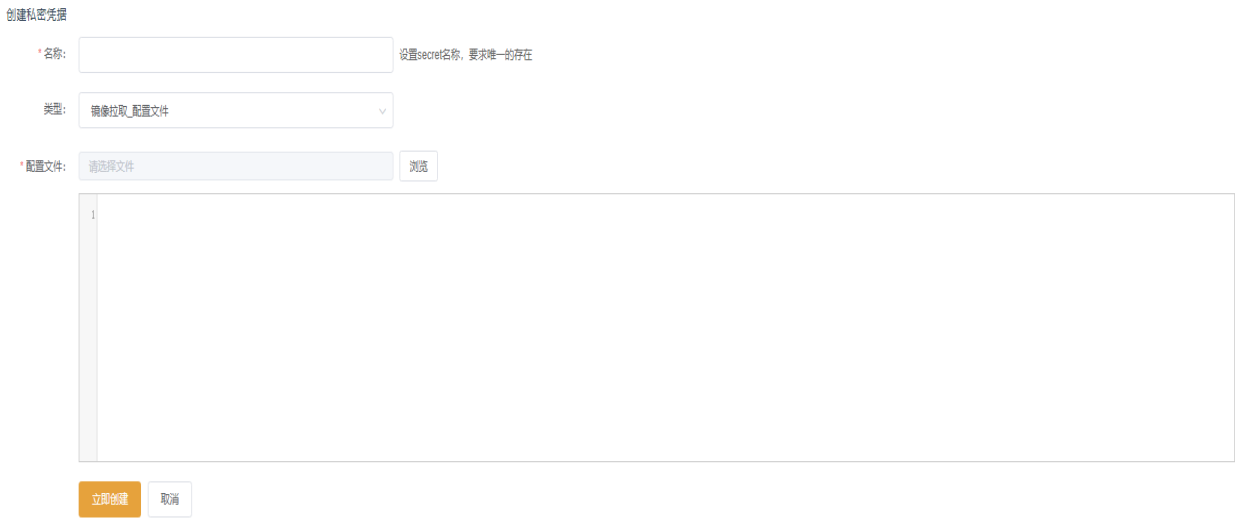
3） 参照下表：

设置新增私密凭证参数，其中带“*”标志的参数为必填参数。输入完成后，单击【创建】；

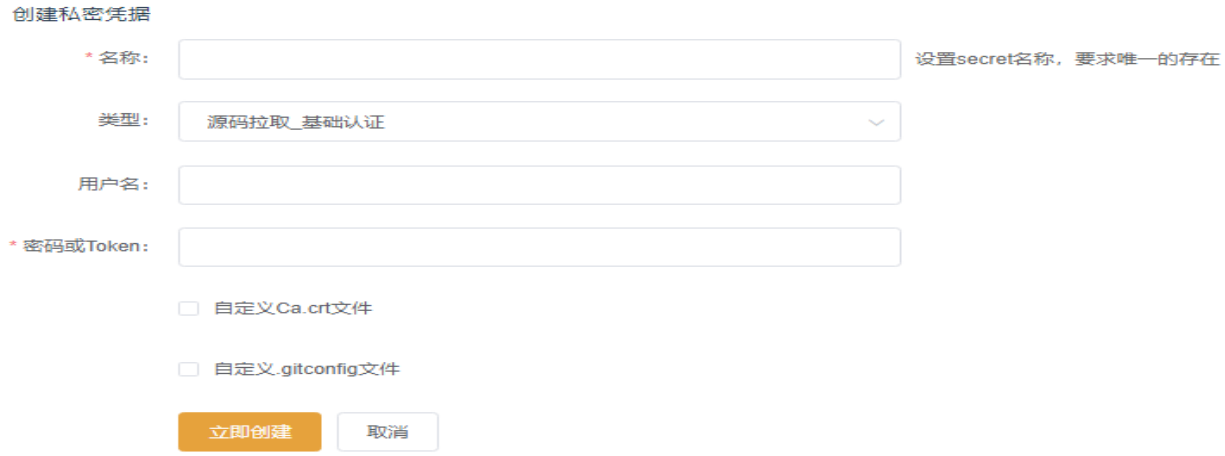
参数	参数说明
*名称	新建私密凭据的名称，必须唯一。
*类型	Opaque：一般密钥类型。 镜像拉取_认证信息：存放拉取私有仓库镜像所需的认证信息。 镜像拉取_配置文件：存放拉取私有仓库镜像所需的配置文件。 镜像拉取_基础认证：存放服务所需的证书。 源码拉取_SSH Key：通过密钥进行拉取代码。
类型为：Opaque	
添加项	增加一对键、配置项内容。
键	配置项的键值。键值只能由字母、数字、句点、连字符和下划线组成。
配置项内容	配置项的内容，通过上传文件来表示。

参数	参数说明
类型为：镜像拉取_认证信息	
镜像仓库地址	输入镜像仓库的地址。
用户名	该镜像仓库的用户名称。
密码	用户名称所对应的的密码。
邮箱	该用户的邮箱地址，必须为user@domain格式。

参数	参数说明
类型为：• 镜像拉取_配置文件	
配置文件	上传拉取私有仓库镜像所需的配置文件。



参数	参数说明
类型为：镜像拉取_基础认证	
用户名	填写用户名称
密码/Token	用户名称对应的的密码。

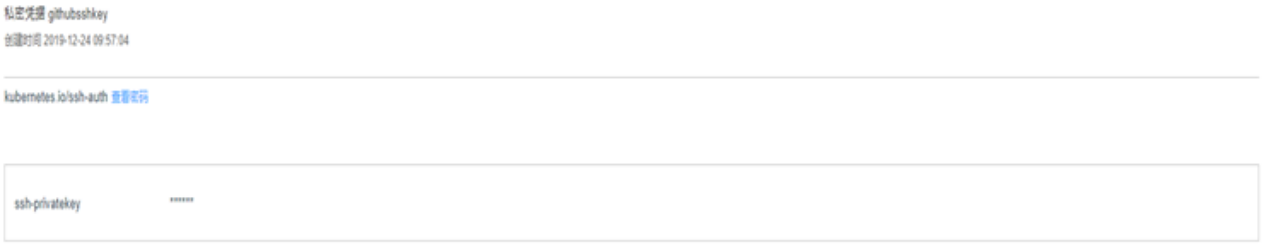


参数	参数说明
类型为：源码拉取_SSH_Key	
SSH_Key私有秘钥	通过密钥进行拉取代码。需要用户上传SSH私有密钥文件。

4）单击【创建】。等待创建成功；

5）创建成功后，服务列表中会出现已创建的秘钥凭证，对已创建的秘钥凭证可进行查看详情、删除的操作。

查看详情如下图：



3.3 镜像仓库

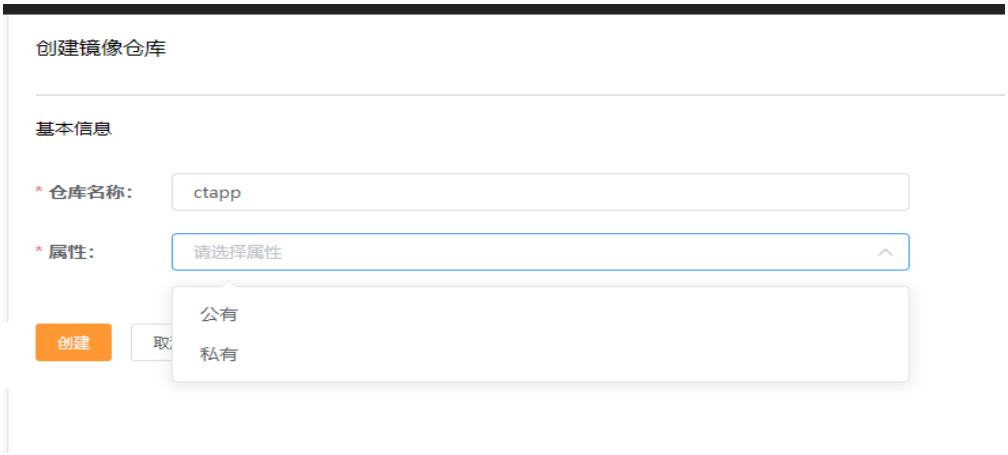
3.3.1 创建镜像仓库

操作步骤：

1）进入【镜像仓库】模块，点击【创建仓库】；

2）填入仓库名称和属性；

参数	参数说明
*仓库名称	仓库的名称，不支持中文字符
*属性	分为共有和私有



3）单击【创建】，完成镜像仓库创建。

3.3.2 上传镜像

操作步骤：

1）在 linux 服务器中，准备好构建完成的 docker 镜像或者从 dockerhub 获取镜像；

2）通过 docker tag 命令，修改镜像名称，例如 docker tag mysql:5.5 cce-registry.ctyun.cn:443/ctapp/mysql:5.5；

3）如果是私有仓库，需要使用 docker login ｛仓库地址｝，输入帐号密码后，完成登录；

4）使用命令 docker push cce-registry.ctyun.cn:443/ctapp/mysql:5.5，完成上传镜像。

3.3.3 操作仓库

操作步骤：

1） 在镜像仓库列表中，对应的仓库有【操作】，可让仓库【转为公有/私有】，【删除】仓库；

2） 原来是公有的仓库，单击【转为私有】可改变属性为私有；原来是私有的仓库，单击【转为公有】可改

变属性为公有；

3） 单击对应仓库的【删除】，点击【确定】可删除该仓库。

3.4 应用

3.4.1 无状态

若用户需要托管以docker容器打包的应用，请创建容器应用。无状态应用中各实例之间相互独立，互不依赖，任意一个Web请求完全与其他请求隔离。无状态容器应用更易实现可靠性和伸缩性。

说明：创建多个容器应用时，请确保容器应用使用的端口不冲突 ，否则部署会失败。

操作步骤：

1)在左侧导航栏中选择“应用管理 -> 无状态（Deployment）” ，在右侧页面单击“创建无状态负载” ；

2）添加基本信息；

- 负载名称：例如nginx。
- 命名空间：系统给租户分配命名空间。
- Pod数量：本例中修改Pod数量为1。
- Pod规格：选择通用计算型，CPU 0.5核，内存 1GB。

3）创建负载-配置访问信息；

4）单击“下一步” ，进入高级配置中选择升级策略：滚动升级、替换升级。选择滚动升级后；

5）单击”下一步”进入规格确定页，确定所选规格是否正确，如没有问题的话，点击确认下单， “返回无状态负载列表” ；

3.4.2 容器组

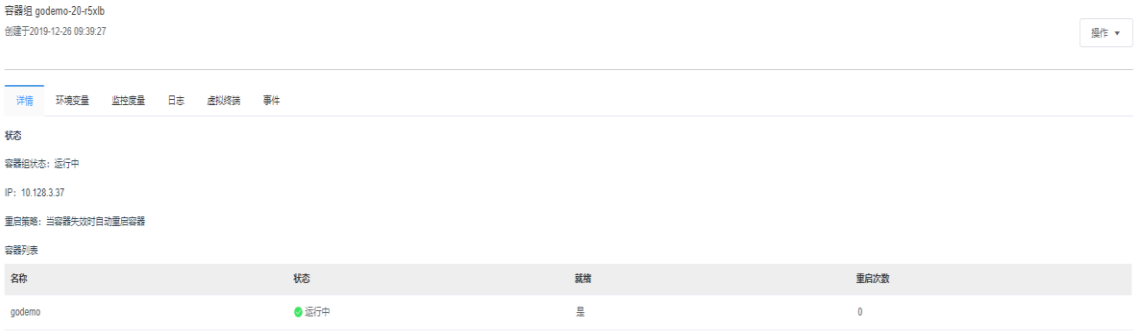
容器组是Kubernetes部署应用或服务的最小的基本单位。一个容器组可以封装多个应用容器(也可以只有一个容器）、存储资源、一个独立的网络IP以及管理控制容器运行方式的策略选项。

用户部署完镜像后，查看【容器组列表】。

容器组列表页中内容包括：名称、状态、就绪容器、重启容器、触发时间。

点击【名称】页面可以跳转到【容器组详情页】。查看容器组详情、环境变量、监控度量、日志、虚拟终端、事件

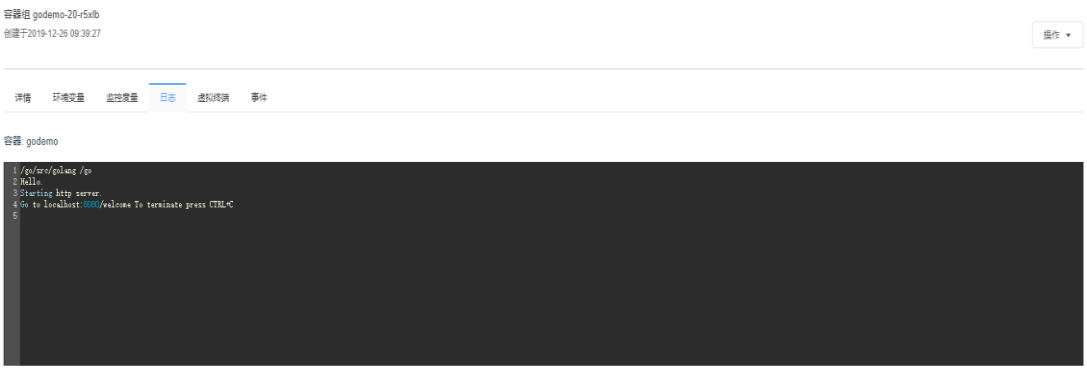
1、容器组详情_详情 界面样式：



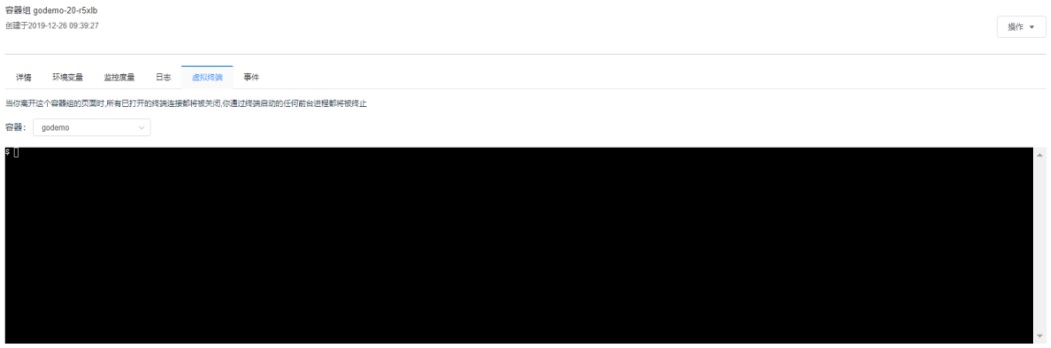
2、容器组详情_环境变量 界面样式：

3、容器组详情_监控度量 界面样式：

4、容器组详情_日志 界面样式：

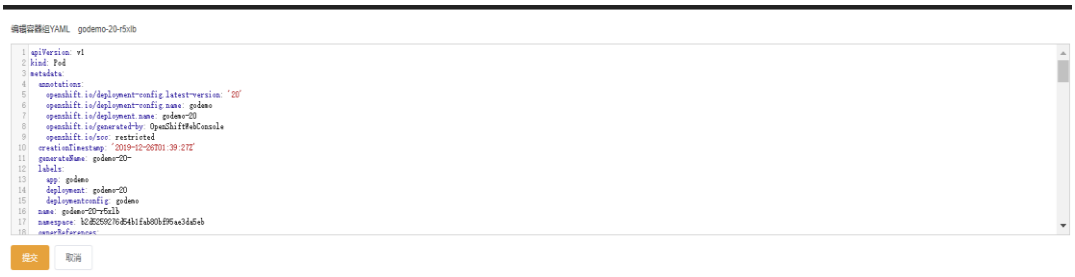


5、 容器组详情_虚拟终端 界面样式



6、 容器组详情_事件 界面样式

容器组详情页中，点击操作中的编辑YAML，如下图：



点击操作中的删除，如下图：

3.5 网络管理

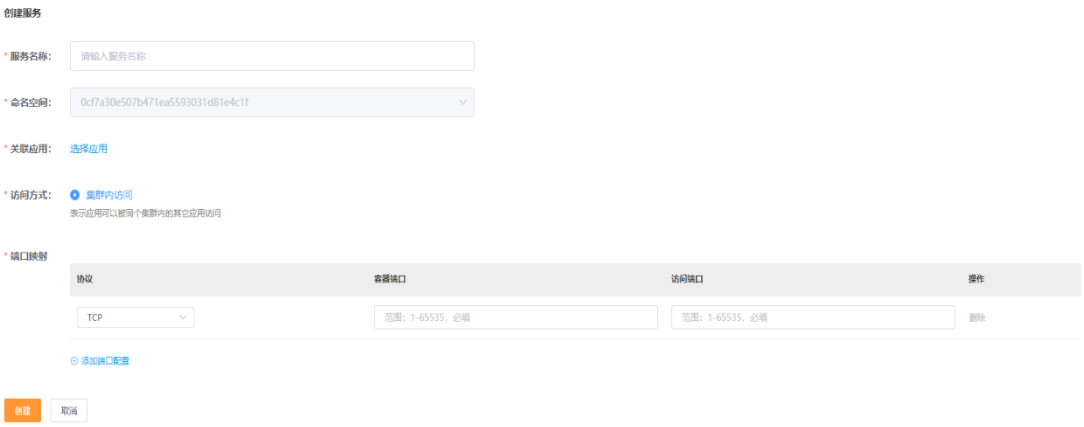
3.5.1 服务

云容器引擎为满足多种复杂场景下应用间的互相访问，提供了不同的访问方式，从而满足不同场景提供不同访问通道。本平台暂时支持的服务类型只包括：集群内访问（ClusterIP）。表示应用暴露给同一集群内其他应用访问的方式，可以通过“集群内部域名”访问。

操作步骤：

1）在控制中心中，选择【计算】【容器实例】，进入容器实例界面；

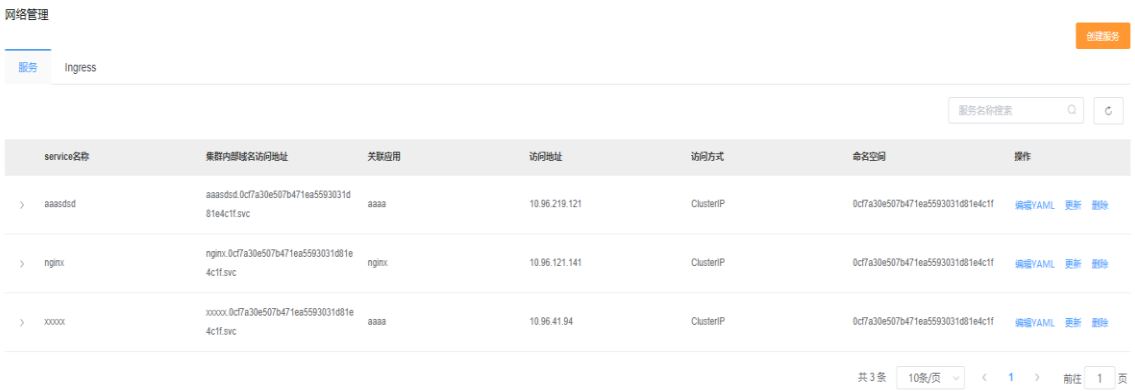
2）单击左侧导航栏的【网络管理】【服务】，单击【创建服务】；



3） 参照下表设置新增服务参数，其中带“*”的参数为必填参数；

*服务名称	新建服务的名称。
*关联应用	选择需要添加Service的工作负载。
*访问方式	默认选择“集群内访问（ClusterIP）”。
*端口映射	协议：请根据业务的协议类型选择TCP、UDP。 容器端口：应用程序实际监听的端口。 访问端口：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。 添加端口配置：点击添加端口配置新增一行端口映射。 删除：删除端口配置。一个端口配置不允许删除，删除后置灰。

4）单击【创建】，页面跳转到【服务列表项】，如下图：



3.5.2 路由

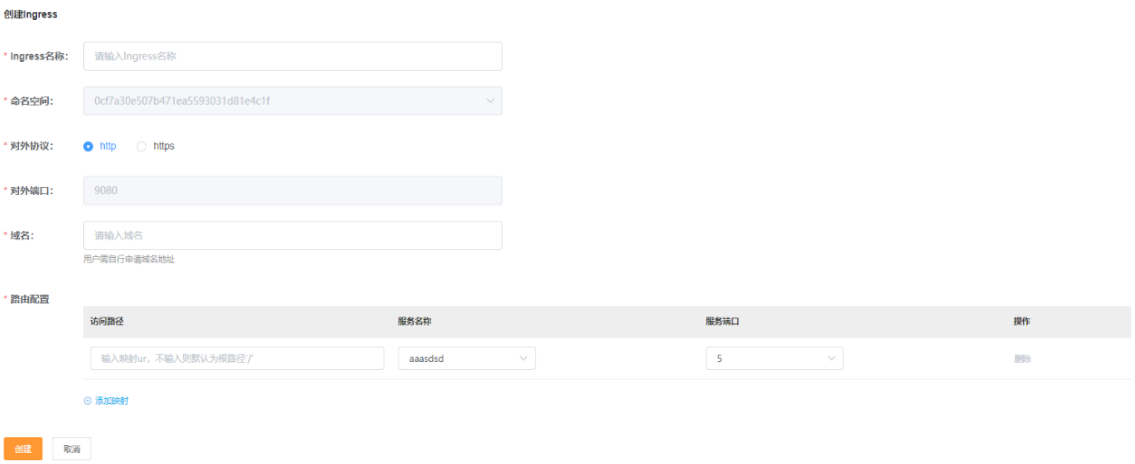
七层负载均衡支持URL配置，通过对应的URL将访问流量分发到对应的服务。同时，服务根据不同URL实现不同的功能。

路由七层负载由公网服务地址、访问端口、URL组成，例如：10.117.117.117:80/helloworld。

操作步骤：

1) 在控制中心中，选择【计算】【容器实例】，进入容器实例界面；

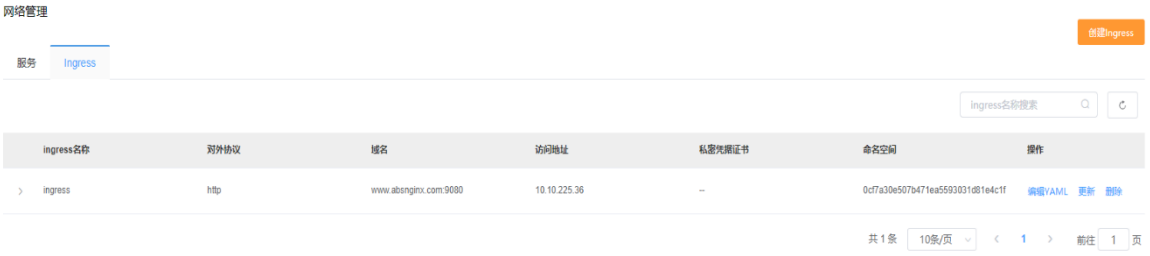
2) 单击左侧导航栏的【应用】【路由】，单击【创建路由】；



3） 参照下表设置新增路由参数，其中带“*”的参数为必填参数；

*Ingress名称	新建Ingress的名称。
*域名	实际访问的域名地址，需用户购买备案自己的域名，必填。一旦配置了域名规则，则必须使用域名访问。
访问路径	需要注册的访问路径，不输入默认为根路径。例如：/healthz。
*服务名称	和路由相关联的服务。
*服务端口	选择相应服务后对应的端口。
对外协议	支持http、https协议。

4）单击创建后，页面跳转回路由列表页，如图：



3.6 为应用挂载数据卷

Docker镜像是由多个文件系统叠加而成，当启动一个容器的时候，Docker会加载只读镜像层并在上面添加一个读写层。当删除Docker容器并通过该镜像重新启动时，之前的更改将会丢失。为了能够保存数据以及共享容器间的数据，Docker提出了数据卷的概念。简单来说，数据卷就是目录或者文件，它可以绕过默认的联合文件系统，以正常的文件或者目录的形式存在于主机上。

在Docker中，数据卷只是磁盘或另一容器中的目录。其生命周期不受管理，且Docker现在提供的卷驱动程序功能非常有限。容器实例CCI采用的是Kubernetes的数据卷的概念，Kubernetes数据卷具有明确的生命周期管理，支持多种类型的数据卷，同时实例可以使用任意数量的数据卷。

CCI支持挂载本地磁盘：

挂载本地磁盘：支持emptyDir、configMap、secret三种。

EmptyDir：用于临时存储，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；

ConfigMap：将配置文件中的key映射到容器中，可以用于挂载配置文件到指定容器目录；

Secret：将密钥中的数据挂载到指定的容器路径。

“挂载本地磁盘”的操作步骤：

1) 登录容器实例控制台，在镜像部署过程中，点击添加本地磁盘进行添加；

2) 卷类型选择**emptyDir**：容器分配到节点时系统将自动创建卷，初始内容为空。在同一个Pod中所有容器可以读写emptyDir中的相同文件。当Pod从节点上移除时，empryDir中的数据也会永久删除。通常用于临时数据的高速存储。

配置参数：

参数	参数说明
本地磁盘名称	输入存储名称。
卷类型	• 选择卷类型为【emptyDir】； • 选择【存储介质】： – 默认：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：存储在内存中，适用于数据量少，读写效率要求高的场景。 • 单击【确定】，挂载对应的容器路径；

参数	参数说明
挂载路径	数据卷挂载到容器上的路径。 注意：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。
权限	• 只读：只能读容器路径中的数据卷。 • 可写：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。

3） 卷类型选择**configMap**：平台提供应用代码和配置文件的分离，configMap用于处理应用配置参数。用户需要提前创建应用配置，操作步骤请参见创建应用配置项；

配置参数：

参数	参数说明
本地磁盘名称	输入存储名称。
卷类型	• 选择卷类型为【configMap】； • 选择对应的configMap名称； 说明：configMap需要提前创建，请参见创建配置项(configMap)。
挂载路径	数据卷挂载到容器上的路径。
权限	只读：只能读容器路径中的数据卷。

4） 卷类型选择 secret：用户需要提前创建私密凭据，操作步骤请参见创建私密凭据；

配置参数：

3 操作指南

参数	参数说明
本地磁盘名称	输入存储名称。
卷类型	• 选择卷类型为【secret】； • 选择对应的 secret 名称； 说明：secret需要提前创建，请参见创建私密凭据(secret)。
挂载路径	数据卷挂载到容器上的路径。
权限	只读：只能读容器路径中的数据卷。

3.7 应用管理运维

3.7.1 应用监控

应用创建成功后，可通过性能监控，来监控容器的CPU和内存使用情况。

1) 登录容器实例控制台，单击左侧导航栏的【应用管理】【无状态】；

2) 单击最对应的名称，进入该名称应用的详情页面；

3) 单击【监控度量】页签；

4) 查看相应容器的CPU使用大小、内存使用大小、网络速率：

CPU使用大小：横坐标表示时间，纵坐标表示CPU使用大小。

内存使用大小：横坐标表示时间，纵坐标表示内存使用量。

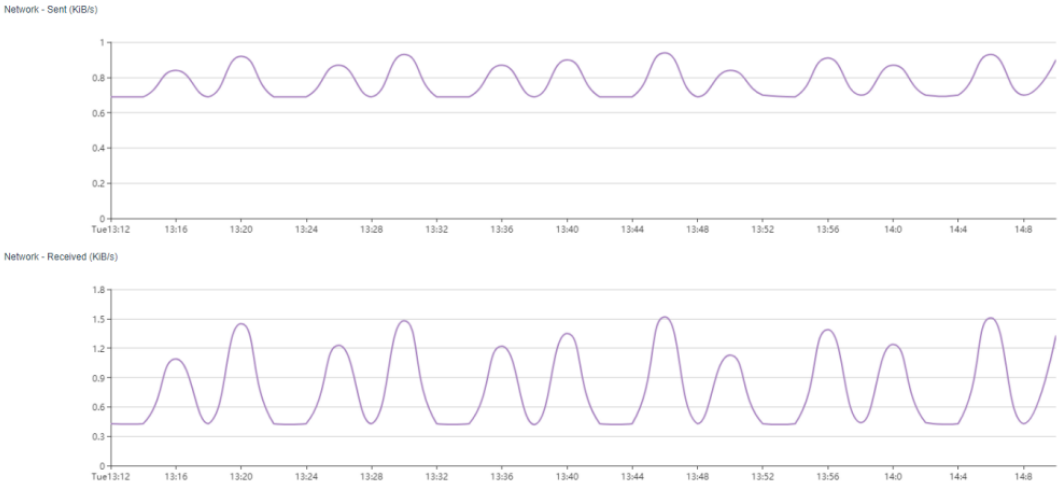
网络速率(发送)：横坐标表示时间，纵坐标表示网络速率(发送)。

网络速率(接收)：横坐标表示时间，纵坐标表示网络速率(接收)。

内存使用大小、CPU使用大小：


jenkins-6
开始于2019-12-19 15:35:49
操作
详情
环境变量
监控度量
日志
事件
容器：
jenkins
时间范围：
最近1小时
内存 (MiB)
1,500
1,200
900
600
300
0
Tue13:12
13:16
13:20
13:24
13:28
13:32
13:36
13:40
13:44
13:48
13:52
13:56
14:0
14:4
14:8
CPU (Cores)
0.03
0.025
0.02
0.015
0.01
0.005
0

网络速率(输入、输出)：


Network - Sent (KiB/s)
1
0.8
0.6
0.4
0.2
0
Tue13:12
13:16
13:20
13:24
13:28
13:32
13:36
13:40
13:44
13:48
13:52
13:56
14:0
14:4
14:8
Network - Received (KiB/s)
1.8
1.5
1.2
0.9
0.6
0.3
0

4 常见问题

4.1 容器实例和容器引擎有什么区别？

容器引擎提供高性能可扩展的容器服务，基于云主机快速构建高可靠的容器集群，兼容 Kubernetes 及 Docker 容器生态，帮助用户轻松创建和管理多样化的容器应用，并提供容器故障自愈，监控日志采集，自动弹性扩容等高效运维能力。

容器实例服务提供 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。

4.2 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，您可以在创建容器模板时设定不超过30个的环境变量；环境变量可以在应用部署后修改，为应用提供极大的灵活性。在CCE中设置环境变量与Dockerfile中的“ENV”效果相同。

4.3 如果不挂载云存储的话，容器运行产生的数据存储在哪里？

4.4 部署应用有哪几种方式？有什么区别？

目前支持三种部署方式，用户可基于自身需求选择：

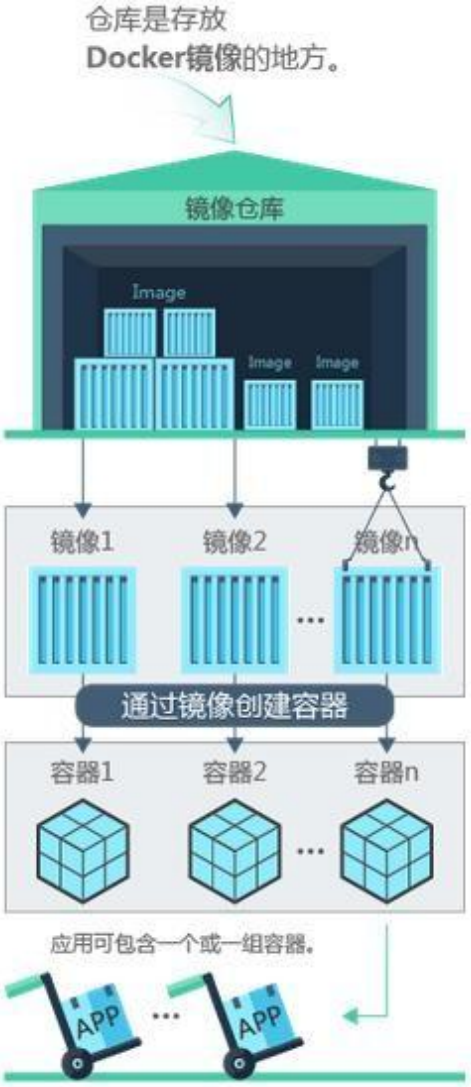
- 选择Dockerhub官方镜像：基于开源docker公有镜像创建容器应用。
- 选择天翼云镜像：基于天翼云官方平台上的私有镜像创建容器应用。
- 构建私有镜像：您可基于业务需求制作私有docker镜像。基于该私有镜像创建容器应用。

4.5 镜像、容器、应用的关系是什么？

镜像：Docker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。

镜像、容器、以及应用之间的关系请参见下图：


仓库是存放
Docker镜像的地方。
镜像仓库
Image
Image
Image
镜像1
镜像2
镜像n
通过镜像创建容器
容器1
容器2
容器n
应用可包含一个或一组容器。
APP
APP

5 参考知识

5.1 ConfigMap配置项要求

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

json格式：

```
{
 "kind": "ConfigMap",
 "apiVersion": "v1",
 "metadata": {
  "name": "paas-broker-
 app-017", "namespace":
 "lcqtest", "enable": true
 },
 "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
```

文件名称为 configmap.json，配置示例如下：

yaml格式：

文件名称为configmap.yaml，配置示例如下：

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
    #  nodeSelector:
    #    env: test-team
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 80
```