容器服务

产品文档

文档目录

- 产品简介
  - 产品概述
  - 功能优势
  - 集群管理
  - 代码构建
  - 镜像仓库
  - 编排管理
  - 负载均衡
  - 配置管理
  - 应用生命周期管理
  - 网络管理
  - 云硬盘管理
  - 日志管理
  - 事件管理
  - Quota管理
  - 应用场景
- 购买指南
  - 计费概述
- 快速入门
  - 第1步：创建集群
  - 第2步：配置集群
  - 第3步：创建应用
  - 第4步：探查应用
  - 第5步：管理应用
- 操作指南
  - 操作指南
  - 业务总览
  - 代码构建
  - 镜像仓库
  - 应用列表
    - 应用列表
    - 1.如何创建一个应用?
    - 2.应用管理
    - 3.应用详情
    - 4.实例详情

- 云硬盘
  - 云硬盘
  - 1.云硬盘列表
- 应用编排
  - 应用编排
  - 1.如何创建一个编排模版?
  - 2.编排模版列表
  - 3.编排详情
  - 4.如何通过编排模版创建一个应用?
  - 5.编排应用的展示和管理
- 配置管理
  - 配置管理
  - 1. ConfigMap和Secret的区别
  - 2. 如何创建配置组和配置项
  - 3. 创建应用使用配置组

最佳实践
- 最佳实践
- 一、选择镜像
- 二、创建应用
- 三、查看应用
- 四、操作应用
  - 1. 弹性伸缩
  - 2. 灰度升级

运维指南
- 集群创建
- 集群运维

故障处理

常见问题
- 什么是容器服务?
- 容器服务适用于什么场景?
- 如何开始使用容器服务?
- 集群
- 网络
- 存储
- 监控

app接口
- app接口
- 创建app

- 删除app
- 获取app信息
- 灰度升级
- 扩容缩容
- 终止app
- 启动app

instance接口
- instance接口
- 实例列表
- 获取实例端口映射信息
- 实例信息
- 终止实例
- 启动实例
- 升级实例
- 获取实例访问入口

container接口
- container接口
- 查询容器
- 容器列表

volume接口
- volume接口
- 创建云盘
- 查询云盘
- 删除云盘
- 扩展云盘

pod编排接口
- pod编排接口
- 创建Pod编排
- 更新Pod编排

compose编排接口
- compose编排接口
- 创建compose编排
- 获取compose编排
- 删除compose编排

名字服务接口
- 名字服务接口
- 获取app名字服务
- 实例端口映射信息

ConfigMap接口
ConfigMap接口
验证ConfigMap名称
验证ConfigMap配置项名称
创建ConfigMap
获取ConfigMap
添加或修改ConfigMap的配置项
获取ConfigMap某个Key对应的内容
获取ConfigMap某个Key关联的app信息
删除ConfigMap的某个配置项
删除整个ConfigMap
Secret接口
Secret接口
验证Secret名称
验证Secret配置项名称
创建Secret
获取Secret
添加或修改Secret的配置项
获取Secret某个Key对应的内容
获取Secret某个Key关联的app信息
删除Secret的某个配置项
删除整个Secret

产品简介

产品概述

最近更新时间: 2019-11-02 06:56:13

容器服务 容器服务是基于原生kubernetes打造的容器云平台，可以调度CPU或GPU等海量计算资源，运行各种计算框架，监控任务执行结果，让数据中心的所有资源被合理使用，协同完成各类应用场景的服务和计算。容器服务为容器化应用提供了从构建到交付到运行的一整套解决方案。通过对原生kubernetes的产品化，提升了可用性、易用性、以及共享容器集群的安全性等，帮助用户降低成本，提高效率。容器服务免费试用，涉及的其他云产品另外单独计费。

功能优势

最近更新时间: 2019-11-02 06:54:59

容器服务提供多种功能，包括集群管理、代码构建、镜像仓库、编排管理、负载均衡、配置管理、应用生命周期管理、网络管理、云硬盘管理、日志管理、事件管理、quota管理等。

集群管理

最近更新时间: 2019-11-02 06:53:56

- 支持集群创建、删除、伸缩等操作。
- 可以对集群内主机进行管理、优先级管理、业务管理、容器实例配置管理等。
- 集群内完善的监控指标，支持自定义告警策略。

代码构建

最近更新时间: 2019-11-02 06:52:20

- 支持项目管理，代码仓库关联，代码分支指定。
- 支持持续集成，构建镜像，以及跟踪构建过程等功能。
- 支持云Dockerfile，默认和自定义镜像tag，构建各流程的日志、构建记录等展示、生成的镜像添加新tag等功能。

镜像仓库

最近更新时间: 2019-11-02 06:51:02

- 支持多级镜像：个人镜像、业务镜像，还可以查看公共镜像，镜像具有role based权限管理。
- 提供镜像构建自动化流程，实现通过配置管理库下载源代码，自动化编译和构建，并制作成新的容器镜像。
- 统计常用镜像，从镜像部署应用，可以查看镜像详情，历史版本等。
- 提供自动镜像安全扫描功能。
- 支持镜像同步功能。
- 支持P2P镜像下载。

编排管理

最近更新时间: 2019-11-02 06:49:17

- 编排支持kubernetes编排和compose编排。
- 编排模板中可见关系图、YAML编码和操作记录。
- 可以从编排模板作为入口直接部署编排。
- 可以查看，修改编排的YAML文件，并可以展示编排模板的操作记录。

负载均衡

最近更新时间: 2019-11-02 06:47:48

- 负载均衡与应用生命周期独立。
- 可以为同一应用或多个应用的多个容器配置统一访问地址，均衡各容器实例数据流量，也可以单独对每个实例解绑/绑定负载均衡。
- 当应用扩容缩容或者实例迁移时，系统都可以透明的自动完成动态的负载均衡绑定。

配置管理

最近更新时间: 2019-11-02 06:45:54

- 包括configMap和secret。
- 所有配置可以通过前台页面进行创建、编辑、删除、查询等操作。
- 应用可以与配置进行关联

应用生命周期管理

最近更新时间: 2019-11-02 06:43:30

- 应用视图管理：Stack□app□instance的层次管理，并且有对应的层次视图。
- 应用操作管理：Stack、app、instance都支持创建、删除操作。其中app还新增了停止、启动和重启、复制等操作。
- 应用调度属性：提供多种部署模式，亲和/反亲和调度策略，标签调度等。
- 应用升级/回退：支持滚动升级，以及灰度升级。允许指定个别实例升级/回退到特定版本。允许同一个应用的实例具有多个不同的版本。
- 应用弹性伸缩：支持自动扩容缩容，以及主动扩容缩容。支持扩容时指定版本号。通用服务在缩容操作时，可以指定缩容后的实例数，也可以对特定实例做选择性的缩容操作。
- 应用健康检查：支持如liveness Probe、Readiness Probe等探针脚本配置，用户自定义检测应用的状态。
- 应用操作记录：对应用的任何操作进行记录，以备审计
- 对应用的探查方式：支持控制台、日志、操作记录、监控、事件管理多等种方式。
- 对应用的访问方式：提供应用/实例的访问入口，绑定域名，负载均衡等访问方式，也可以使用实例名直接访问。

网络管理

最近更新时间: 2019-11-02 06:40:22

- 同时支持四种网络模式选择：overlay网络，Floating IP，NAT，以及host模式。
- 在每一台主机上的各个容器，也可以独立的使用各自的网络模式，互不影响。
- 支持对静态IP的前台导入管理；

云硬盘管理

最近更新时间: 2019-11-02 06:39:02

- 支持普通云硬盘和内置云硬盘，支持快照功能。
- 云硬盘操作可以有创建、销毁、挂载、扩容。
- 可以从云硬盘创建快照，也可以从快照创建云盘。支持云盘的迁移。支持对云盘的离线和在线扩容。
- 对云硬盘的操作具有权限控制，并提供多集群多租户视图。
- 支持独占式云盘和共享型云盘。

日志管理

最近更新时间: 2019-11-02 06:37:09

- 对系统日志和应用日志进行统一管理：无需用户修改应用程序，应用日志即可进入日志管理体系。提升日志/数据的IO性能；
- 对完成的应用/容器日志进行保存；
- 提供日志的分级查询，以及全文检索；
- 提供对查询日志的图形化统计；
- 提供对本地日志文件的web查看；
- 可以通过控制台提供对日志文件的操作。

事件管理

最近更新时间: 2019-11-02 06:35:24

- 提供对各类对象(如app，pod)的事件记录，并且不受kubernetes的有限事件限制
- 可以从web页面查看各类事件。

Quota管理

最近更新时间: 2021-09-15 10:10:48

- 对所有资源进行quota管理，包括cpu、内存，磁盘，网络等。
- 并且修改quota准入策略，确保高优先级重要应用经过准入后可以分配到资源。

应用场景

最近更新时间: 2019-11-02 03:34:53

• 持续集成与持续交付 实现了从开发、测试到发布整个流程一体化、标准化、自动化。通过不断地代码构建和自动化软件交付，让软件产品高效迭代，从而快速成熟、更加安全可控。 ü 一键提交，自动构建，随处运行。 ü 横向、纵向扩展开发到运维整个流程设计，满足不同项目多样化需求。 ü 自动化单元测试，及时发现问题。 ü 自动部署测试环境，提前模拟上线测试（安装测试），减少上线风险。 ü 版本电子流流转，全程信息全面可见，方便问题定位。

• 微服务架构 最近微服务在社区和企业中都逐步流行起来，微服务是用一组小而专的服务来构建一个应用，服务独立运行在不同的进程中，服务之间通过轻量的通讯机制来交互，并且服务可以通过自动化部署方式独立部署。微服务的核心理念是将大型、复杂的应用分解为小且内聚的服务，这些小的服务便于扩展和部署，它们可以简单的和当下流行的Docker、DevOps、云计算联系起来。

• 传统应用及有状态应用 对于大部分企业应用来说，还没有或者不能完全改造成微服务应用，而这些应用也希望能够使用到容器化，持续集成与持续交付，云上的统一调度和资源管理、容灾等特性，容器服务可以借助Tapp应用类型对这类应用提供支持，我们也称之为通用服务。

• 离线应用 越来越多的计算框架已经被广大数据工程师喜爱并大规模的使用，容器服务利用优化的Quota的弹性管理、优先级调度、Tapp、准入控制、P2P 镜像下载等特性为这些离线计算提供支持，让用户无需关注繁琐的集群管理，专注业务。

• 弹性伸缩 容器具有极其轻量、高效的特点，使得容器化的服务可以被快速的启动、迁移、裁撤等。同时借助容器服务的自动扩缩容和主动扩缩容机制，让业务可以根据各自的运行情况，资源使用指标，以及自定义指标等进行自动或者主动的弹性伸缩。

购买指南

计费概述

最近更新时间: 2019-11-02 06:58:06

容器服务免费试用，涉及的其他云产品另外单独计费

快速入门

第1步：创建集群

最近更新时间: 2019-11-02 07:10:04

在集群创建界面点击新建，然后依次填写集群信息、选择机型、云主机配置，以及信息确认即可创建一个容器集群。对创建好的集群还可以进行扩容、删除等操作。

第2步：配置集群

最近更新时间: 2019-11-02 07:10:35

对于企业级用户，如果需要多个业务共享一个集群，并且对各个业务有资源的隔离和保证时，建议对集群做一些简单的配置管理，比如创建namespace，设置各个namespace的quota等。

第3步：创建应用

最近更新时间: 2019-11-02 07:04:27

可以从应用列表或者镜像仓库的镜像中创建应用，设置应用的集群信息、资源配置、实例信息、重启策略、云盘等基本信息，也可以关联配置，设置调度管理，以及进行日志/数据路径、网络模式、负载均衡、告警设置等高级设置。设置之后，点击“一键部署”就可完成应用的发布。

第4步：探查应用

最近更新时间: 2019-11-02 07:02:28

当应用部署到容器平台之后，有多种方式可以对该应用进行探查。为具有前端页面的应用自动提供访问入口，以及默认的负载均衡，方便查看业务页面。可以通过操作记录审计对应用的各种操作。事件管理包括对应用的事件和实例的事件，可以查看历史的所有事件。可以直接通过控制台的方式进入容器，并进行各种操作。

第5步：管理应用

最近更新时间: 2019-11-02 07:00:43

应用首次发布后，后续还需要对应用做进一步的管理，主要的操作包括删除、复制、重启、停止等，也可以对应用做扩缩容以及升级等操作。扩缩容分为主动扩缩容和自动扩缩容。

操作指南

操作指南

最近更新时间: 2019-11-30 15:26:21

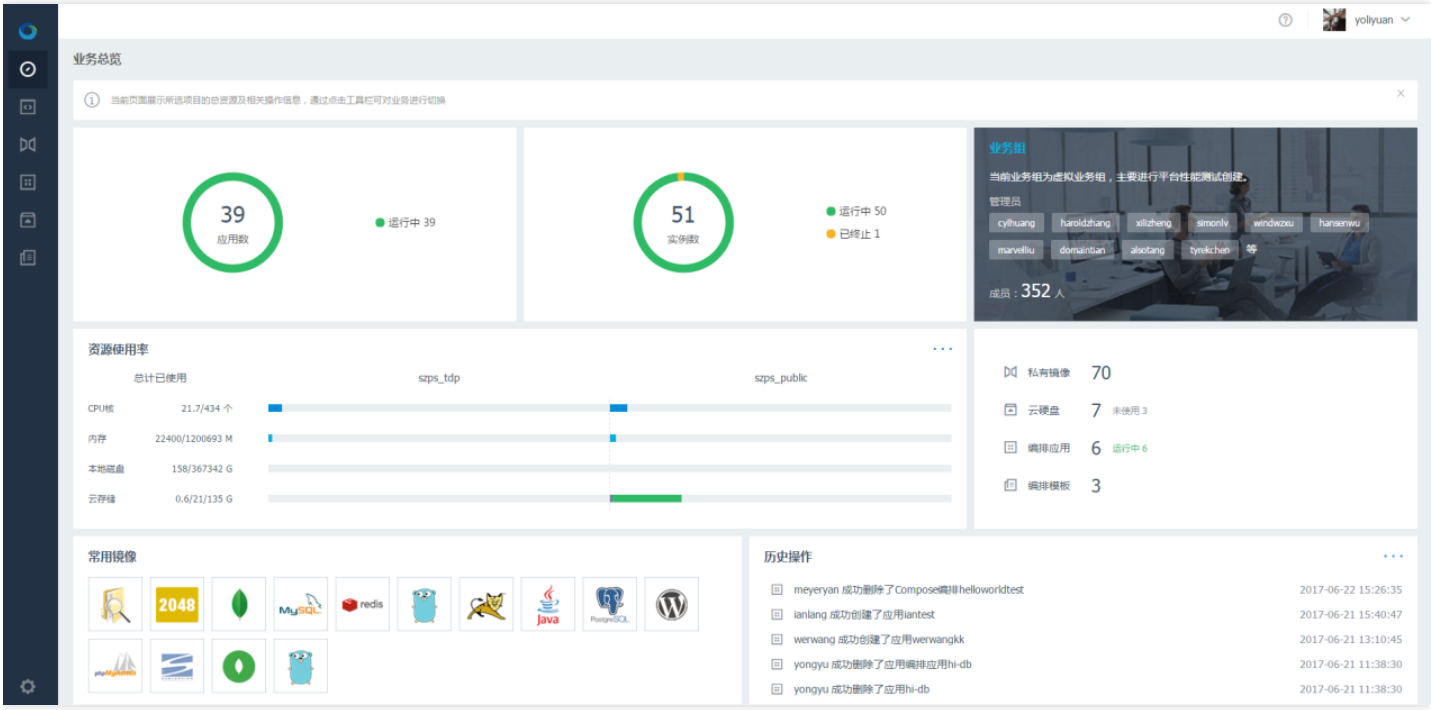
首次进入管理台，可看到以下指引界面，您可以通过切换阅读简单了解目前平台为您提供的服务：

接下来我们将详细介绍容器服务为您提供的从代码构建到应用部署的全流程的服务。

业务总览

最近更新时间: 2019-11-30 15:26:21

业务总览页面是容器服务的第一个页面，概要展示了用户所选项目的总资源及相关操作信息：



- 业务基本信息： 应用数、实例数及状态（可点击跳转到应用列表）；
- 业务组基本信息： 管理员、成员数；
- 业务所有的集群资源及资源使用状态，点击右上角的图表“更多”可以查看详细的资源趋势图；
- 私有镜像、云硬盘及编排数概览；
- 常用镜像列表（可点击镜像跳转到该镜像详情页面）；
- 历史操作记录： 同样可以点击右上角查看更多历史操作记录。

代码构建

最近更新时间: 2019-11-22 17:26:29

点击左侧导航第二个入口可切换至代码构建，代码构建可以帮助开发者将已经写好的代码同步并打包成镜像至容器平台，这样就可以直接通过镜像部署应用让业务运行起来，达到全流程化的操作。同时在后续的代码更新中也可以通过自动构建的方式更新代码到容器平台生成新的版本。

• 1.如何创建一个构建项目?

1. 点击创建项目按钮;

2. 按照要求依次填写代码来源和构建项目基本信息。

3. 点击快速创建完成项目构建。

其中常遇到的问题有以下几点:

o 什么是代码分支? 代码分支即为代码在仓库中保存的目录，在选择代码分支前需先点击同步代码方可选择。 o 什么是Dockerfile路径? Dockerfile是一个镜像的表示，可以通过Dockerfile来描述构建镜像的步骤；Dockerfile路径即为构建生成镜像的存储路径，默认为根目录可不填写，亦可根据个人习惯填写存储路径。 o 开启自动构建会怎样? 开启自动构建后，当在代码仓库中push最新的代码后，将自动触发代码构建，在原有镜像下生成新的镜像版本；不会影响正在运行中的应用。 • 2.构建列表 创建完成后页面将自动跳转后代码构建页面，展示相应的构建列表；其他情况下可以点击页面标题处的面包屑返回。

在构建列表中，你可以进行以下操作:

o 点击构建项目当前行可选中构建项目进行删除操作，删除构建项目不影响已生成的镜像;

o 点击构建项目名称，可进入构建详情;

o 点击代码源可进入代码来源地址;

o 点击镜像，则进入构建生成的镜像详情页面，正在构建中的镜像没有详情页面。

• 3.构建详情页面

点击构建项目的名称进入构建详情页面，详情页面展示构建项目的详情信息构建历史和日志，点击构建记录项可展开详细的日志信息。 同时也可以对当前日志进行手动构建、删除和修改Dockerfile路径等操作。

镜像仓库

最近更新时间: 2019-11-22 17:26:29

镜像仓库中展示了所有可用镜像内容。按照常用、我的镜像、业务镜像和全部，对镜像进行了分类，其中：

- 常用镜像：是指部署使用过的镜像
- 我的镜像：由代码构建生成，且放置于个人目录中的镜像，仅自己可见、可操作。
- 业务镜像：由代码构建生成，且属于业务的镜像，业务成员均可见、可操作；默认展示已加入的所有业务的镜像。

您可以通过分类tab快速删选镜像，也可以通过搜索进行检索。

应用列表

应用列表

最近更新时间: 2019-11-02 13:41:32

应用列表是整个容器平台的核心功能点，前面我们知道了如何把已经编写好的代码构建成为镜像，接下来就来看一下如何通过镜像生成应用，让业务跑起来。

1.如何创建一个应用?

最近更新时间: 2019-11-22 17:29:42

1.如何创建一个应用?

1.1 进入应用列表页面点击创建应用按钮;

1.2 选择需要部署的镜像，即业务代码； 当然你也可以在镜像仓库中选中相应的镜像，点击部署。

1.3 填写部署信息，部署信息主要包含：部署对象、基本配置、配置管理、高级配置。

• 部署配置: 即为所选镜像，可以点击更换；默认部署镜像的最新版本，您也可以切换为其他版本。

基本配置: 基本配置中除云硬盘外均为必填项，其中需注意：部署集群即为应用运行及资源消耗集群，集群资源不足会限制实例配置可选项及实例数量。

说明: 挂载云硬盘可以将应用运行结果数据保存，方便后续使用和数据共享；挂载云硬盘的应用只可包含一个实例，同一个应用可挂载多个云硬盘。应用与挂载的云硬盘需属于同一个集群下，不可跨集群挂载。

勾选使用云硬盘后需选择要挂载的云硬盘，并填写挂载地址；因此需要在云硬盘页面中创建好可用的云硬盘。挂载地址是指云硬盘在应用上的位置路径，输入格式为路径格式。云硬盘挂载到应用上之后，只有应用结束方能对云硬盘进行操作或挂载至其他云硬盘。

• 配置管理： 配置管理为非必填项，具有默认值可直接跳过不填。在应用部署时将配置导入到容器，支持环境变量和配置文件两种方式导入，同时支持无需停止服务状态下，对多个容器内配置文件进行在线更新。

高级配置: 高级配置为非必填项，具有默认值可直接跳过不填，如特殊需求可对高级配置进行修改。

• 启动命令。启动容器时将会运行的命令；指定启动命令后，镜像中指定的Entrypoint和Cmd将不再起作用。

• 环境变量： 可根据所需增加或删减环境变量参数。如配置管理与高级配置环境变量冲突，则使用高级配置内用户手动填写的值。

• 日志路径： 容器内用于存储日志的目录，该目录会被映射到主机的目录上，访问数据更快，且容器重启时，日志不受影响。

• 数据路径： 容器内用于存储数据的目录，该目录会被映射到主机的目录上，访问数据更快，且容器重启时，数据不受影响。

• 网络模式： 目前支持Overlay（虚拟网络）、Floating IP（浮动IP）、NAT（端口映射）、Host（宿主机网络）四种网络模式，可根据所需进行选择一种网络模式。

• 自动扩缩容： 启动自动扩缩容，应用运行时可以根据各个实例的实际CPU使用率自动扩容、缩容。

2.应用管理

最近更新时间: 2019-11-22 17:32:17

应用创建完成之后可点击返回应用列表页面，在应用列表页面你可以看到应用的基本信息及运行状态，点击左侧的加号，可展开查看应用下的实例运行信息。 勾选应用可对应用进行，启动、停止、弹性伸缩、灰度升级、删除等操作。其中仅删除支持批量操作。

- 启动： 停止的应用可点击再次启动。
- 停止： 各状态中的应用均可以点击进行停止，停止的应用服务将不可用，同时不再消耗资源。
- 弹性伸缩： 当需要扩大或缩小业务规模时可选择弹性伸缩；当前平台生成的应用为微服务类型的应用，应用各实例配置完全一致，弹性伸缩即增加或随机减少实例个数。
- 灰度升级： 可以选择将当前应用镜像版本升级为其他版本。
- 删除： 删除当前应用，资源将自动回收，删除后的应用不显示在应用列表，请谨慎操作。
- 说明：默认应用列表展示用户加入的所有业务的、通过所有途径创建的应用，包含后续将讲到的通过编排创建的应用。您可以通过表格右上角的搜索和筛选条件对应用进行快速查找。

3.应用详情

最近更新时间: 2019-11-22 17:32:17

点击应用名称可进入应用详情页面，应用详情页面分为上下两部分。上部分主要展示应用的基本信息镜像版本、状态、归属等，以及应用操作；下部分则详细的展示了应用相关的配置信息、实例列表、资源监控、日志、操作记录。

- 应用详情中的操作与应用列表中的操作保持一致；
- 基本信息中的镜像版本和访问地址可分别点击前往详情页面；
- 概况： 主要内容为实例配置、挂载云硬盘信息以及高级配置信息，不支持修改；
- 实例： 展示了应用下的实例列表以及相关状态、镜像版本及IP信息，点击实例名称可进入实例详情页面；
- 监控： 目前包含了应用内存和CPU核使用趋势图[y2]；
- 日志： 显示当前应用运行过程中的日志记录，至此按照关键词、主机、级别等查询；
- 操作记录： 即用户在前台针对当前应用的历史操作，及结果状态。

4.实例详情

最近更新时间: 2019-11-02 08:28:09

通过点击应用列表中展开应用下的实例名，或者应用详情中实例列表的实例名，可进入实例详情页面。 实例详情页面与应用详情页面结构类似，容器平台主持创建的应用均为微服务类型的应用，应用下各实例无实质差别，因此指定实例进行无可操作项。实例详情页面主要用于展示实例的详情信息，其中管理台为一个内嵌Webshell，您可以在这里直接对实例进行操作，语法操作与Linux一致。

云硬盘

云硬盘

最近更新时间: 2019-11-02 13:43:10

云硬盘是可灵活挂在在应用实例上的存储空间，它可以帮助应用实现：数据持久化、不同的实例之间共享和重用数据、以及数据的备份、恢复和迁移。

1.云硬盘列表

最近更新时间: 2019-11-02 08:42:06

- 点击左侧导航云硬盘->云硬盘列表，可进入云硬盘列表页面。云硬盘是作为应用的数据存储空间存在，在应用创建的时候挂载，应用结束后方能进行操作或挂载到其他应用上。云硬盘列表默认展示已加入业务的所有云硬盘，可根据右上角的搜索和筛选条件进行查询。
- 创建云硬盘：点击列表左上角创建云硬盘，在创建弹窗中填写相关信息： 其中右侧可用资源为当前所选业务集群下的资源使用状况。云硬盘存储的数据文件系统目前支持：扩展文件系统EXT4和大文件XFS。 同时我们也可以从已有的云硬盘快照中创建云硬盘，那么新建的云硬盘中将包含原有快照中的数据内容，文件系统格式也将与原有快照保持一致。
- 云硬盘操作：只有未使用的云硬盘可进行操作，选中未使用中的云硬盘可进行扩容、删除、生成快照等操作。
- 云硬盘资源管理器：云硬盘资源管理器用于辅助管理云硬盘中的文件内容，你可以在界面中可视化的管理云硬盘中的文件。点击云硬盘列表右侧的按钮可以打开或关闭云硬盘管理器，使用中的云硬盘不可进行文件管理。打开云硬盘管理器会消耗云硬盘所在集群微量资源，打开时一般需要20秒左右的加载时间。一般云硬盘资源管理器不会主动关闭，即资源不会主动回收，需手动关闭。打开资源管理器的云硬盘状态会显示为“管理中”。

应用编排

应用编排

最近更新时间: 2019-11-02 13:43:54

在实际业务中会遇到，业务代码是分多个镜像打包，而运维人员通常需要对业务的多个模块镜像进行一次性部署，这个时候就产生了应用编排的功能。您可以通过编写YAML文件的方式建立了一个编排模版，模版中可以包含多个应用、不同镜像版本、应用之间也可以建立依赖关系，然后再通过部署编排模版的方式让应用按照指定的顺序启动。

1.如何创建一个编排模版?

最近更新时间: 2019-11-22 17:50:21

1.1 点击页面左上角的创建编排，为编排模版输入一个名称；

1.2 选择所属业务，当前业务下的人员将都可见并且可以使用、编辑当前编排模版；

1.3 编写文件内容。（如果您还不知道怎么编写YAML文件，那么你可以点击左下角的示例，我们为您准备了一个示例编排，您可以按照示例编排的格式编写自己的编排文件。）

2.编排模版列表

最近更新时间: 2019-11-22 17:44:17

编排模版列表展示了已加入业务的所有编排模版，可以通过右上角的搜索和筛选进行查找。其中：

- 点击编排模版名称可进入编排详情
- 点击部署可使用当前编排模版直接部署生成应用
- 下方显示内容问当前编排模版包含的应用对应的镜像，hover至镜像logo可查看镜像名称及版本。

3.编排详情

最近更新时间: 2019-11-22 17:44:17

编排详情中显示了编排模版的详细内容，包含：编排基本信息、应用关系图、YAML文件及操作记录。

其中YAML文件支持修改更新，修改后的YAML文件不影响已部署的应用，如需对已部署的应用进行更新可直接针对应用进行操作，或更新编排模版后重新部署。

4.如何通过编排模版创建一个应用?

最近更新时间: 2019-11-22 17:44:17

4.1 选定编排模版点击部署，跳转至应用创建页面：

4.2 填写相关字段。由于应用的相关配置及所属业务已经在编排模版中设置过，因此只需填写部署名称，即当前部署的编排应用合集名称，部署集群即可一键部署。

4.3 部署完成后可返回应用列表查看已部署应用的运行状态及详情。

5.编排应用的展示和管理

最近更新时间: 2019-11-02 08:43:53

编排应用同样展示在应用列表中，编排应用的名称=部署名称+YAML中应用名称。同样您也可以点击应用列表顶部的切换按钮切换至编排应用，然后按照部署集合的方式查看部署下的全部应用。 您可以通过勾选删除部署的方式对部署下的应用进行批量删除，编排应用的操作管理及详情与普通应用保持一致。

配置管理

配置管理

最近更新时间: 2019-11-02 13:44:59

容器的配置需要通过配置文件、命令行参数和环境变量来进行的，为了使用方便，往往需要将这些整合到image里，如果配置出现变化，需要重新制作image，从而造成容器的便携性较差。为了实现应用程序的代码和配置相分离，将配置信息和image解耦，平台提供了ConfigMap和Secret两种API资源，在应用部署的时候，将配置文件、命令行参数以及环境变量注入到容器内部，对容器完全透明，保证应用程序的可移植性，不同的场景可以使用不同的配置，而无需重新制作image。

1. ConfigMap和Secret的区别

最近更新时间: 2021-09-15 10:10:48

- ConfigMap：用于存储一些非敏感的配置信息，如应用程序的ini, json等配置文件，也可保存二进制大文件
- Secret: 用来存储一些少量较为敏感的数据，确保传输的安全，如密码、密钥、token、证书等敏感内容

ConfigMap和Secret的创建和使用上是相同的，下面以ConfigMap为例介绍配置管理的功能。

2. 如何创建配置组和配置项

最近更新时间: 2019-11-22 17:46:58

2.1 点击左侧导航应用管理-》配置管理，可进入配置管理页面。选择ConfigMap子页面，点击创建配置组，选择业务和集群，输入配置组名称，点击保存，即可创建一个空的配置组。用户可以将应用使用的各种配置组织起来，方便管理。 须以小写字母或数字开头结尾，可以用-或.分隔，且只能包含-、.、小写字母和数字。

2.2 选中某个配置组，点击添加配置项，输入配置项的名称和内容，点击保存。 点击配置组旁边的加号展开，就可以刚刚创建的配置项。 注: 之前说过，配置项可以环境变量或者配置文件注入到容器中，配置项的名称作为环境变量的key或者文件名需要满足相应的命名规范：

- 环境变量：只能包含下划线(_)、大小写字母和数字
- 配置文件：名称只能包含中划线(-)、下划线(_)、点（.）、大小写字母和数字

2.3 展开配置项会看到每个配置项关联的应用个数，点击个数，会看到应用信息和关联方式，如果该配置项被应用使用，将无法删除，需要先删除所关联应用，这是为了防止应用在重启时，获取配置项失败。

3. 创建应用使用配置组

最近更新时间: 2019-11-22 17:46:58

3.1 进入创建应用页面，展开配置管理，可以选择用户创建的ConfigMap或者Secret配置组

3.2 应用使用配置组有两种方式，环境变量和配置文件

• 环境变量：选择某个配置组，右边就会显示对应的配置项，用户可以勾选某个配置项，配置项的名称会作为环境变量的key，内容作为环境变量的value，所以，配置项的名称需要满足环境变量key的命名规则，前台会过滤掉不满足规则的配置项。

• 配置文件：填写挂载目录（如果没有会自动创建），选择配置组和配置项，配置项的名称就是文件的名称，在实例中就会生成文件/etc/config/yarn-site.xml

3.3 应用创建完成，通过控制台进入实例，就可以看到生成的环境变量和配置文件

3.4 在应用详情页面，也可以很方便的看到应用使用的配置组信息和使用方式。

最佳实践

最佳实践

最近更新时间: 2019-11-02 13:28:12

本文介绍在容器服务中，如何构建一个最简单的web应用。

一、选择镜像

最近更新时间: 2019-11-22 17:46:58

进入“镜像仓库”页面的“业务镜像”，在右侧搜索框中搜索“helloworld-with-log”

从“全部”中搜索镜像

二、创建应用

最近更新时间: 2019-11-30 15:26:21

1. 为了体验灰度升级，使用“通用服务”；
2. 为了体验日志，在高级属性中填写日志路径为/data/log

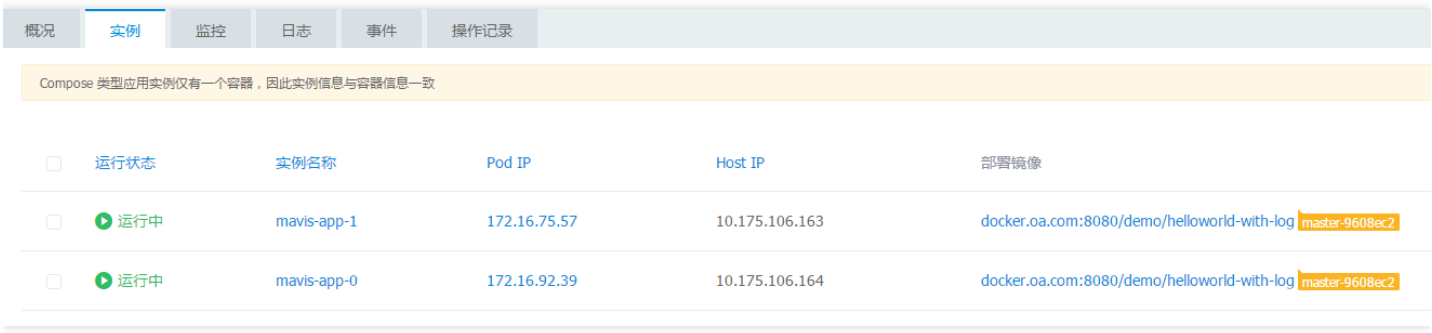
   应用创建中

三、查看应用

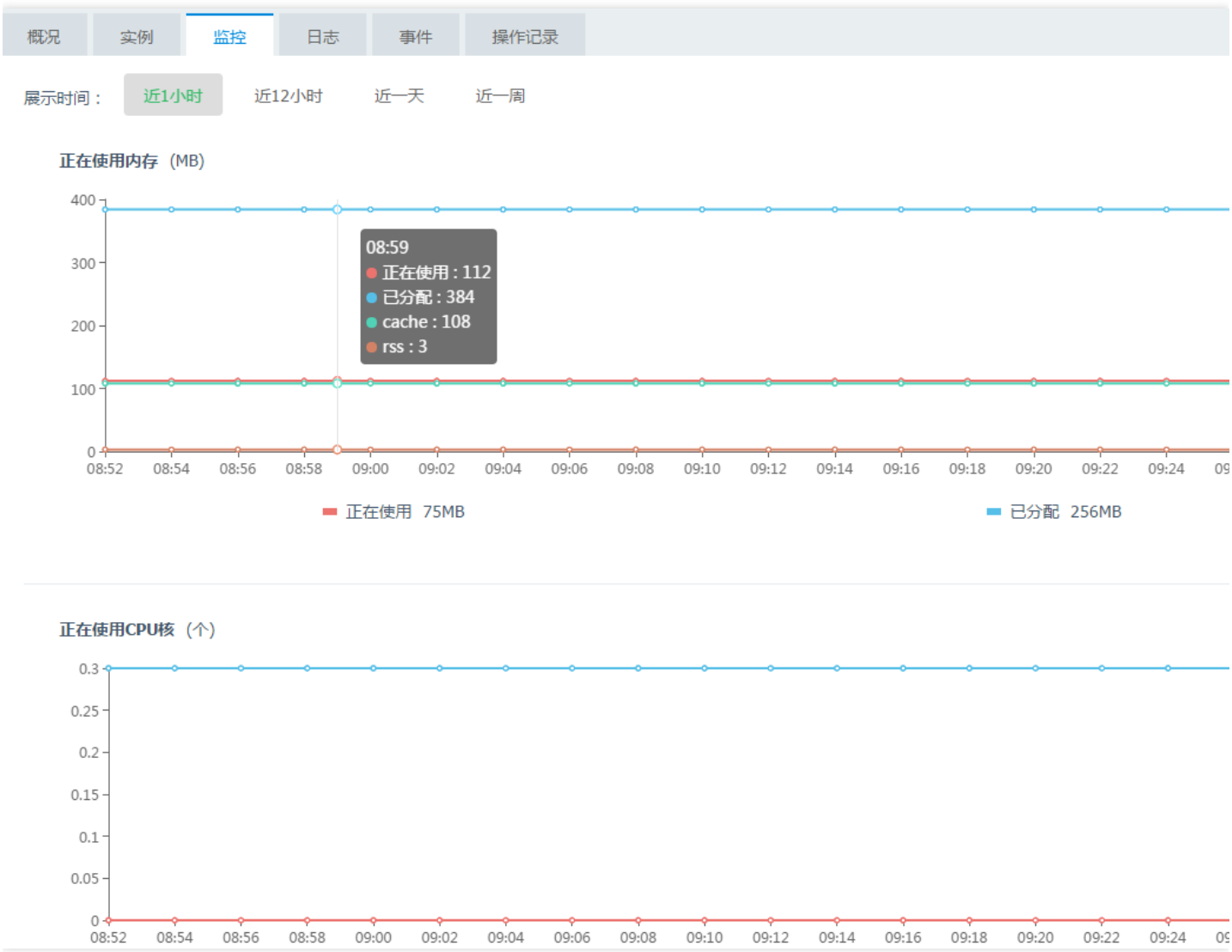
最近更新时间: 2019-11-30 15:26:21

点击应用名称，进入应用详情页面。

可以通过“访问地址”，进入应用页面。



应用的实例页面


概况
实例
监控
日志
事件
操作记录
展示时间：
近1小时
近12小时
近一天
近一周
正在使用内存 (MB)
400
300
200
100
0
08:59
正在使用 : 112
已分配 : 384
cache : 108
rss : 3
08:52
08:54
08:56
08:58
09:00
09:02
09:04
09:06
09:08
09:10
09:12
09:14
09:16
09:18
09:20
09:22
09:24
正在使用 75MB
已分配 256MB
正在使用CPU核 (个)
0.3
0.25
0.2
0.15
0.1
0.05
0

应用的监控页面

应用的日志检索

应用的事件

四、操作应用

1. 弹性伸缩

最近更新时间: 2019-11-30 15:26:21

应用的弹性伸缩

运行状态	实例名称	Pod IP	Host IP
运行中	mavis-app-0	172.16.92.39	10.175.106.164
运行中	mavis-app-1	172.16.75.57	10.175.106.163
运行中	mavis-app-2	172.16.86.37	10.175.106.152

实例数从2扩大3

扩容

操作记录到应用操作记录中

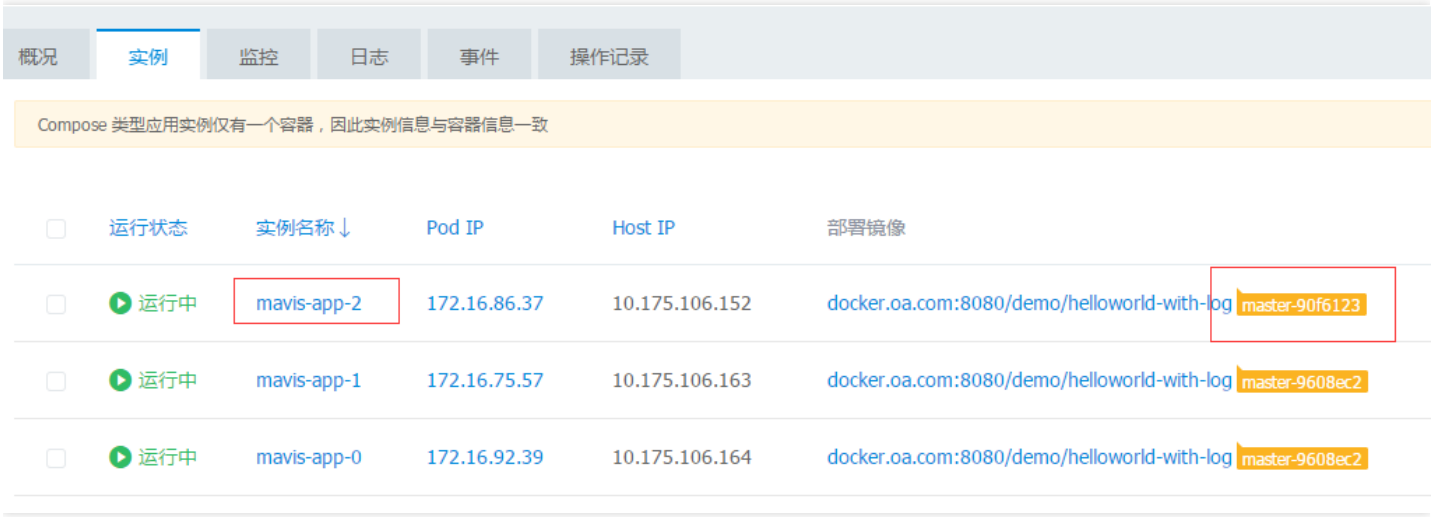
2. 灰度升级

最近更新时间: 2019-11-30 15:26:21

1） 只升级部分实例，如实例2

对2号实例做升级

选择升级的版本号



2号实例被升级 2） 灰度升级全部实例：

对应用的所有实例做升级

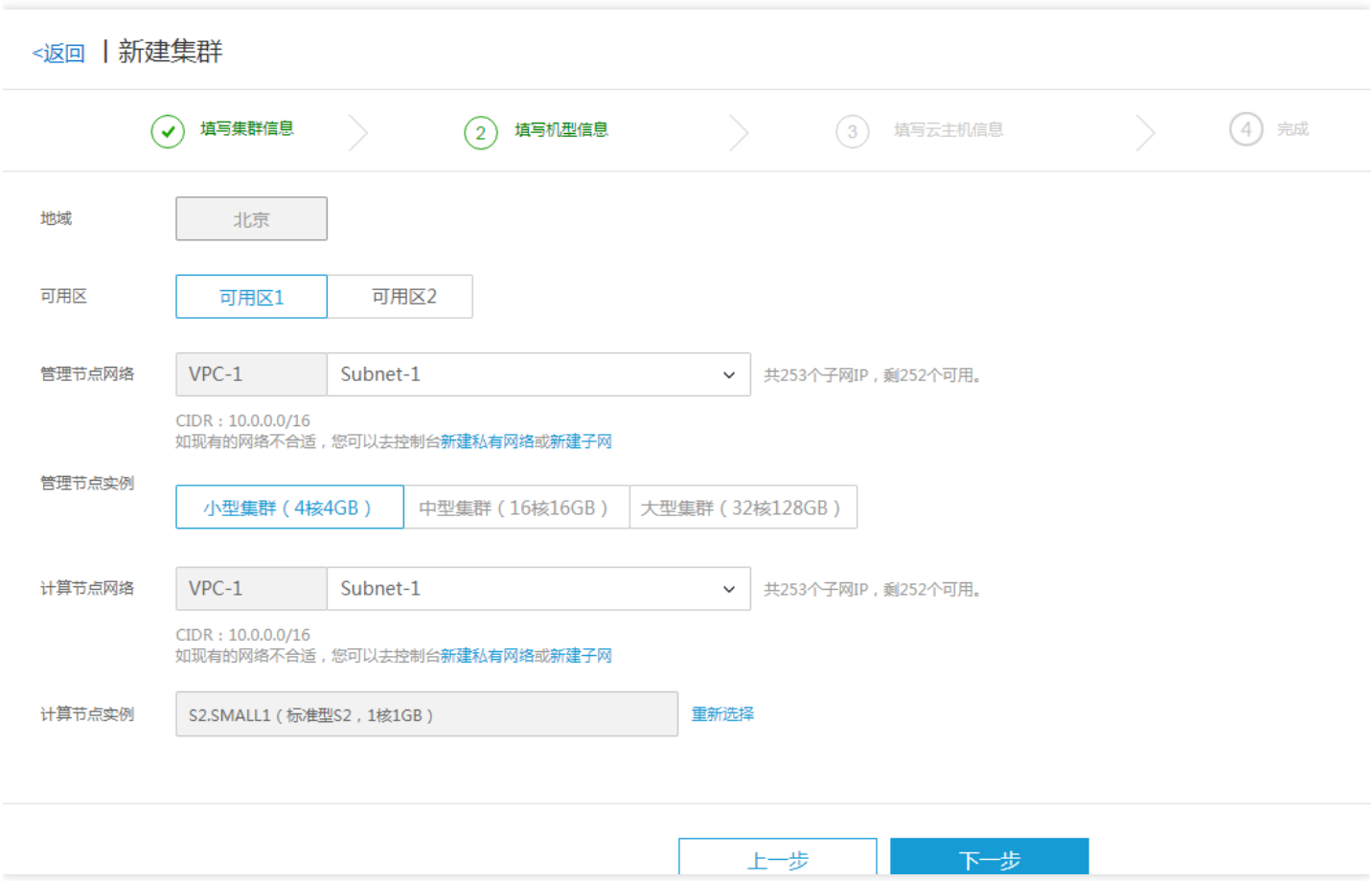
所有实例被升级

运维指南
集群创建

最近更新时间: 2021-12-14 16:18:49

填写集群信息

填写机型信息


<返回 | 新建集群
填写集群信息
2 填写机型信息
3 填写云主机信息
4 完成
地域
北京
可用区
可用区1
可用区2
管理节点网络
VPC-1
Subnet-1
共253个子网IP，剩252个可用。
CIDR：10.0.0.0/16
如现有的网络不合适，您可以去控制台新建私有网络或新建子网
管理节点实例
小型集群（4核4GB）
中型集群（16核16GB）
大型集群（32核128GB）
计算节点网络
VPC-1
Subnet-1
共253个子网IP，剩252个可用。
CIDR：10.0.0.0/16
如现有的网络不合适，您可以去控制台新建私有网络或新建子网
计算节点实例
S2.SMALL1（标准型S2，1核1GB）
重新选择
上一步
下一步

填写云主机信息

<返回 | 新建集群


填写集群信息
填写机型信息
3
填写云主机信息
4
完成
计算节点操作盘
本地硬盘
SSD云硬盘
本地硬盘固定为50GB，购买成功后，系统盘不支持更换介质，使用本地硬盘的服务器暂不支持升级CPU/内存/硬盘
计算节点数据盘
本地硬盘
100GB
300GB
500GB
100
GB
操作系统
Centos 7.4 64位
当前仅支持Centos 7.4 64位
用户名
root
密码
请输入密码
Linux机器密码需8到16位，至少包括两项字母、数字或特殊符号
确认密码
请再次输入密码
安全加固
开通
安装组件开通DDoS防护和云镜主机防护 详细介绍
云监控
开通
开通云产品监控、分析和实施告警，安装组件获取主机监控指标 详细介绍
上一步
完成

<返回 | 新建集群

填写集群信息 > 填写机型信息 > 填写云主机信息 > 4 完成

集群新增成功

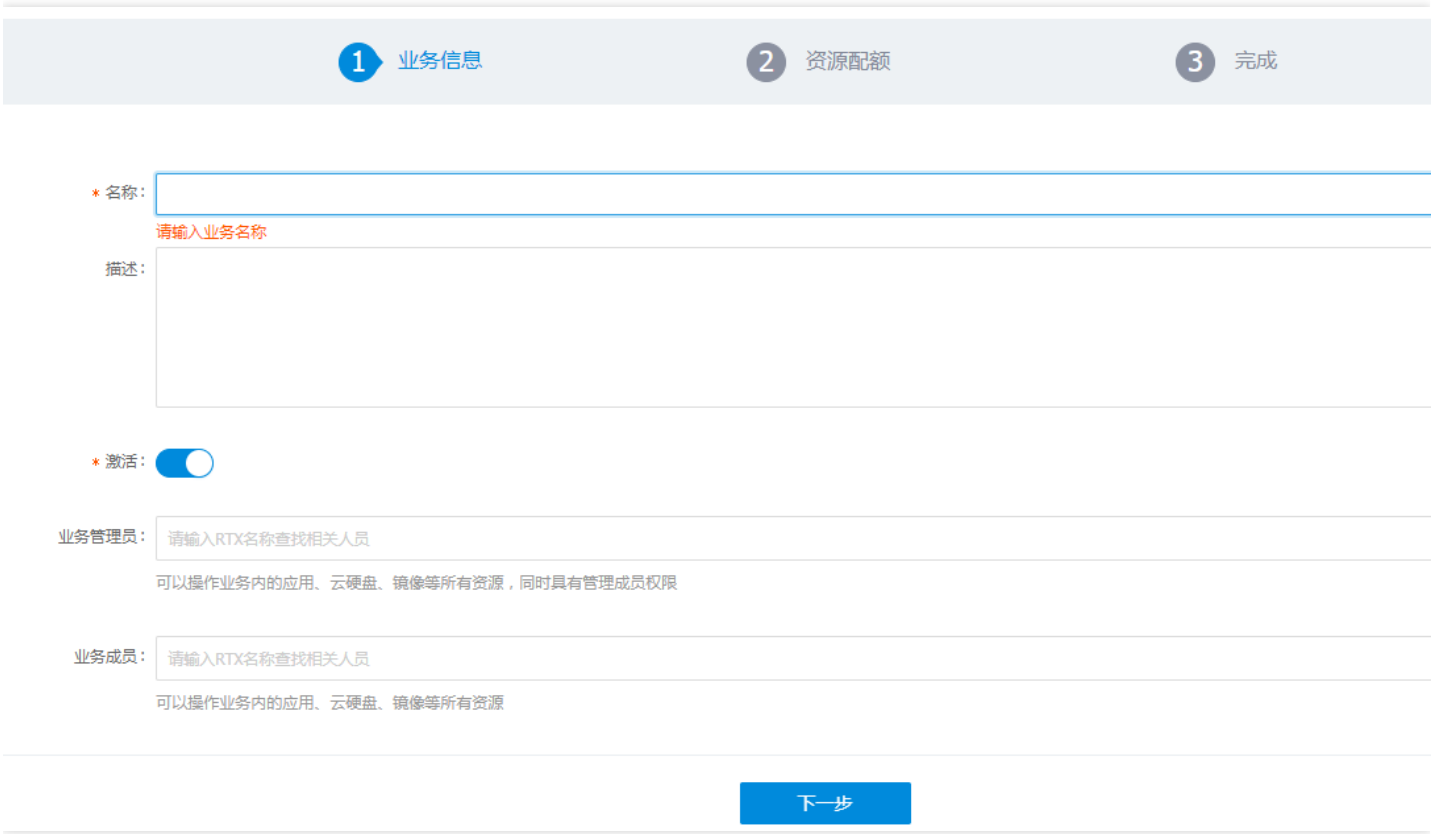
集群新增成功，10s后将自动跳转至管理页面

立即前往

集群运维

最近更新时间: 2019-11-30 15:26:21

创建新的namespace（业务），以及为该业务赋予quota信息。



1 业务信息　2 资源配额　3 完成

* 集群：

* CPU核(个)：

* 内存(MB)：

* 本地磁盘(GB)：

* 云存储(GB)：

* 网络出带宽(Mbit/s)：

* GPU(个)：

+增加集群

上一步　创 建

故障处理

最近更新时间: 2019-11-30 15:26:18

- 集群创建失败 在容器服务上创建集群时，有时会创建失败。因为集群创建时，需要调用底层IAAS接口创建虚拟机，当租户虚拟机quota不足时，有可能导致集群创建失败。
- 应用提交失败 容器服务为了保证应用在提交后可以很快运行起来，以及不同业务共享集群时可以按照quota公平的使用资源，会在应用提交时，加入资源的准入检查。当应用因为准入检查提交失败时，可以通过增加该namespace（业务）在本集群的quota来重试。
- 实例运行失败 用户的应有时候会因为各种原因导致起不来或者运行失败。定位失败有几种方式。通过日志查看，尤其是stderr。还可以通过实例的事件来定位原因。
- 我发现我成功提交了一个应用后发现应用的状态变成了失败，这是什么原因？ 首先这里提示成功是因为这个请求是异步的，说明后台接收到了这个请求。而应用状态变成fail的原因可能有多个，要查看失败原因，可以点击应用名称进入详情页面，然后打开操作记录tab可以看到失败的操作记录，在失败的操作记录上点击后可以展开查看具体的失败原因。除了创建失败，其他的操作失败情况也可以在这里进行查看。

- 我发现提交应用后，部分或者所有实例一直处于等待中状态，如何查看原因？ 等待中状态说明实例在提交到后台后没有及时在工作节点拉起，或者拉起失败了，需要进入应用详情页面后在实例列表的tab中选择有问题的实例，查看实例的具体事件。主要通过查看事件和详细说明可以获取实例pending的原因。例如如果发现是调度失败，则可以查看详细说明可以得知为何失败，一般是资源不足导致的失败会有insufficient字样，如果创建的应用是floatingip网络模式的，还可能出现MatchNodeSelector的错误字样，说明集群没有floatingip资源或者有floatingip的机器其他资源又无法满足了。这时候需要联系管理员增加资源或者申请ip了。如果调度成功了，可能会卡在别的事件上（例如正在拉取一个较大的镜像（这时候需要耐心等待一下），拉取镜像失败，或者在启动容器的过程中出错了等），可以查看最新的事件查明原因。

常见问题

什么是容器服务?

最近更新时间: 2021-10-09 11:07:17

容器服务是基于原生的 Kubernetes 进行适配和功能增强的一款生产级别的容器管理系统。虽然容器有非常多的优点，但要将大量的容器管理起来尤为困难，而 Kubernetes 作为开源的软件，在 Docker 技术的基础上，为容器化的应用提供部署运行、资源调度、服务发现和动态伸缩等一系列完整功能，能够帮助用户快速实现应用程序容器化的部署、扩展和管理。

容器服务适用于什么场景?

最近更新时间: 2021-10-09 11:06:28

容器服务可以支持开发云、测试云、微服务应用、有状态的应用、离线计算、GPU应用等等各种应用场景，是真正意义的云操作系统，让各种应用都可以共享集群资源。

如何开始使用容器服务?

最近更新时间: 2021-10-09 11:06:28

可以通过容器服务控制台进行集群的创建，集群创建完之后即可向该应用提交应用，并进行对应的一站式管理操作。

集群

最近更新时间: 2021-09-17 22:01:29

容器集群如何保证安全隔离?

您创建的集群均是独占集群，同时您可以通过VPC、安全组等功能进行安全隔离。

网络

最近更新时间: 2021-09-17 22:01:29

容器服务有几种网络模式?

容器服务为应用提供了四种网络模式，分别是Host、NAT、Floating IP和overlay网络。

同一个集群下的容器和服务能否互通么?

可以

我想在基础网络中使用容器服务，可以吗?

可以

存储

最近更新时间: 2021-09-17 22:01:29

容器怎么做持久化的存储比较好?

容器服务提供了云硬盘的功能。

容器的存储在节点路径我可以设置么?

可以。在创建应用的时候可以指定数据路径和日志路径就可以。

监控

最近更新时间: 2021-09-17 22:01:29

容器服务目前支持哪些监控?

容器服务提供了集群、节点、应用、实例等多重维度的指标监控。

容器服务支持对应用设置告警么?

支持。在创建应用可以设置，当应用创建完后，也可以随时修改该应用的告警策略。

app接口

app接口

最近更新时间: 2019-11-02 13:30:49

path 参数说明 projectName: 业务名 appName: app名

创建app

最近更新时间: 2019-11-02 11:43:07

`POST /v2/applications/{projectName}` 请求body

```
{
"name": "demoapp",
"imageRef": "nginx:1.7.9",
"resources": {
"cpu": "0.1",
"mem": "256Mi",
"disk":"1Gi",
"gpu":"0"
},
"instanceNum": 2,
"cmd": "sleep 10000",
"dockerEnv": [
{
"name": "PATH1",
"value": "/bin"
},
{
"name": "PATH2",
"value": "/usr/bin"
}
],
"portMappings": [
{
"protocol": "tcp",
"containerPort": 8088
},
{
"protocol": "udp",
"containerPort": 9999
}
],
"dockerLogDir": "/log/dir",
"dockerDataDir": "/data/dir",
"volumes": "rbd1:/data/rbd1,rbd2:/data/rbd2",
"networkType": 0,
"userName":"Jack",
"scale": {
"minReplicas": 1,
"maxReplicas": 5,
"targetCPUUtilization": 50
```

```
},
"configMap":[
{
"type": "Volume",
"name": "special-config",
"key": ["game-config", "game-config2.conf"],
"mountPath": "/etc/config"
},
{
"type": "Env",
"name": "special-config2",
"key": ["HELLO", "WORLD"]
}
],
"secret":[
{
"type": "Volume",
"name": "special-config",
"key": ["game-config", "game-config2.conf"],
"mountPath": "/etc/config"
},
{
"type": "Env",
"name": "special-config2",
"key": ["HELLO", "WORLD"]
}
],
"internalVolumes": [{
"allocateMode": "new",
"fsType": "ext4",
"volumeSize": 1,
"mountPath": "/data"
}],
"nodeLabels":{
"key1":"value1",
"key2":"value2"
}
}
```

返回

```
{
"AppID": "033613b240a2986011223370c94d1519147a0479d7a0eb6e88bb5ce20282be05",
"Cmd": "/bin/bash",
"Created": "2016-09-28T17:29:22+08:00",
"DockerDataDir": "/log/dir",
"DockerLogDir": "/log/dir",
```

```
"Environments": [
{
"name": "PATH1",
"value": "/bin"
},
{
"name": "PATH2",
"value": "/usr/bin"
}
],
"Image": "nginx:1.7.9",
"InstanceNum": 2,
"InstanceSummary": "",
"Name": "demoapp",
"PortsInfo": [
{
"protocol": "tcp",
"containerPort": 8088
},
{
"protocol": "udp",
"containerPort": 9999
}
],
"ProjectID": "abcdefg",
"ProjectName": "default",
"Status": "RUNNING",
"Username": "Jack",
"ResInfo": {
"cpu": "0.1",
"mem": "256Mi",
"disk":"1Gi"
},
"NetworkType": 0,
"Type": "Deployment",
"scale": {
"minReplicas": 1,
"maxReplicas": 5,
"targetCPUUtilization": 50
},
"configMap":[
{
"type": "Volume",
"name": "special-config",
"key": ["game-config", "game-config2.conf"],
"mountPath": "/etc/config"
},
{
```

```
"type": "Env",
"name": "special-config2",
"key": ["HELLO", "WORLD"]
}
],
"secret":[
{
"type": "Volume",
"name": "special-config",
"key": ["game-config", "game-config2.conf"],
"mountPath": "/etc/config"
},
{
"type": "Env",
"name": "special-config2",
"key": ["HELLO", "WORLD"]
}
],
"internalVolumes": [{
"allocateMode": "new",
"fsType": "ext4",
"volumeSize": 1,
"mountPath": "/data"
}],
"nodeLabels":{
"key1":"value1",
"key2":"value2"
}
}
```

删除app

最近更新时间: 2019-11-02 11:43:07

`DELETE /v2/applications/{projectName}/{appName}?app_id=appID` **请求参数 • app_id: 查询条件，选填，如果不设置默认删除正在运行的app 返回**

```
{
"Code": 0,
"Message": ""
}
```

获取app信息

最近更新时间: 2019-11-02 11:43:07

`GET /v2/applications/{projectName}/{appName}` 请求参数 • app_id: appID,选填。如果没有指定,默认获取运行中app的信息。

灰度升级

最近更新时间: 2019-11-02 11:43:07

`POST /v2/applications/{projectName}/{appName}/rollout` 请求body

```
{
"image": "NewImageName:NewTag"
}
```

返回

```
{
"Code": 0,
"Message": ""
}
```

扩容缩容

最近更新时间: 2019-11-02 11:43:07

`POST /v2/applications/{projectName}/{appName}/scale?type=xxx` 请求参数 • type: 扩容缩容的类型，目前支持两种：manual和auto，对于manual类型，需要传scale参数; 对于auto类型，需要传body • scale: 期望的副本数 请求body

```
{
"minReplicas": 1,
"maxReplicas": 10,
"targetCPUUtilization": 50
}
```

终止app

最近更新时间: 2019-11-02 11:43:07

`POST /v2/applications/{projectName}/{appName}/kill?app_id=xxx` 请求参数 • app_id: appID，可选，如果不填则会杀死还在运行中的app

启动app

最近更新时间: 2019-11-02 11:43:07

`POST /v2/applications/{projectName}/{appName}/start?app_id=xxxxx` **请求参数 • app_id: appID, 必填**

instance接口

instance接口

最近更新时间: 2019-11-02 13:31:42

path 参数说明 projectName: 业务名 appName: app名 instanceName: 实例名 containerName: 容器名 fileName: 文件名 tAppInstanceID: tapp实例的instance下标，从0开始

实例列表

最近更新时间: 2019-11-02 11:53:48

`GET /v2/instances/{projectName}/{appName}/lists?app_id={app_id}` 请求参数 • app_id: appID • size: 单页大小 // optional, default:10 • page: 页数 // optional, default:0

获取实例端口映射信息

最近更新时间: 2019-11-02 11:53:05

`GET /v2/instances/{projectName}/{instanceName}/portmapping?instance_id={instance_id}` 请求参数 • instance_id: 实例ID • size: 单页大小 // optional, default:10 • page: 页数 // optional, default:0

实例信息

最近更新时间: 2019-11-02 11:51:36

`GET /v2/instances/{projectName}/{instanceName}?instance_id={instance_id}` 请求参数 • instance_id: 实例ID

终止实例

最近更新时间: 2019-11-02 11:50:37

`POST /v2/instances/{projectName}/{instanceName}/{tAppInstanceID}/kill?app_id={app_id}&&instance_id={instance_id}` 请求参数 • app_id: tapp的ID • instance_id: tapp下instance的id

启动实例

最近更新时间: 2019-11-02 11:48:41

`POST /v2/instances/{projectName}/{instanceName}/{tAppInstanceID}/start?app_id={app_id}&&instance_id={instance_id}` 请求参数 • app_id: tapp的ID • instance_id: tapp下instance的id

升级实例

最近更新时间: 2019-11-02 11:47:52

`POST /v2/instances/{projectName}/{instanceName}/{tAppInstanceID}/update?app_id={app_id}&&instance_id={instance_id}` 请求参数 • app_id: tapp的ID • instance_id: tapp下instance的id 请求body

```
{
"image": "public/2048:latest"
}
```

返回

```
{
"Code": 0,
"Message": ""
}
```

获取实例访问入口

最近更新时间: 2019-11-02 11:21:26

`POST /v2/instances/{projectName}/{instanceName}/url` 请求参数 • 无 请求body 无

container接口

container接口

最近更新时间: 2019-11-02 13:32:48

path 参数说明 projectName: 业务名 appName: app名 instanceName: 实例名 containerName: 容器名

查询容器

最近更新时间: 2019-11-02 11:57:16

`GET /v2/containers/{projectName}/{instanceName}/{containerName}`

容器列表

最近更新时间: 2019-11-22 18:05:14

`GET /v2/containers/{projectName}/{instanceName}/lists` 请求参数

- size: 单页大小 // optional, default:10
- page: 页数 // optional, default:0

volume接口

volume接口

最近更新时间: 2019-11-22 18:05:14

path 参数说明

projectName: 业务名

volumeId: 云盘ID

snapshotId: 快照ID

创建云盘

最近更新时间: 2019-11-22 18:05:14

`POST /v2/cinder/{projectName}/volumes` 请求body

```
{
"size": 10,
"name": "test",
"description": "volume for test",
"snapshot_id": "",
"metadata": {
"fs_type": "xfs"
}
}
```

返回

```
{
"Code": 0,
"Message": "xxx",
}
```

查询云盘

最近更新时间: 2019-11-02 12:10:15

`GET /v2/cinder/{projectName}/volumes/{volumeId}`

删除云盘

最近更新时间: 2019-11-02 12:01:52

`DELETE /v2/cinder/{projectName}/volumes/{volumeId}` 返回

```
{
"Code": 0
}
```

扩展云盘

最近更新时间: 2019-11-02 12:00:43

`POST /v2/cinder/{projectName}/volumes/{volume_id}/extend` 请求body

```
{
"new_size": 10// 单位是G
}
```

返回

```
{
"Code": 0,
"Message": "xxx",
}
```

pod编排接口

pod编排接口

最近更新时间: 2019-11-22 18:10:46

path 参数说明

projectName: 业务名

appName: app名，即pod名

创建Pod编排

最近更新时间: 2019-11-02 12:22:30

`POST /v2/podstack/{projectName}` **请求body**


```
{
"stackInstanceName": "pod666",
"content": "containers:\n- image: docker.oa.com:8080/public/mysql:latest\n name: mysql\n env:\n - name: POD_NAMESPACE\n valueFrom:\n fieldRef:\n fieldPath: metadata.namespace\n - name: MYSQL_ROOT_PASSWORD\n value: hhh\n ports:\n - containerPort: 8080\n name: admin-port\n - containerPort: 3306\n name: db-port\n resources:\n requests:\n cpu: 100m\n memory: 128Mi\n volumeMounts:\n - mountPath: /var/lib/mysql\n name: rethinkdb-storage\n- image: docker.oa.com:8080/public/2048:latest\n name: 2048app\n env:\n - name: POD_NAME\n value: 2048app\n ports:\n - containerPort: 80\n name: web\nvolumes:\n- name: rethinkdb-storage\n emptyDir: {}\nreplicas: 3\nannotation:\n- containerName: mysql\n logPath: /data1\n dataPath: /data2\n- containerName: 2048app\n logPath: /var/log",
"scale": {
"minReplicas": 1,
"maxReplicas": 5,
"targetCPUUtilization": 50
}
```

更新Pod编排

最近更新时间: 2019-11-22 18:10:46

`POST /v2/podstack/{projectName}/{appName}?app_id={appID}` **请求参数**

- app_id: podID

请求body


```
{
"content": "containers:\n- image: docker.oa.com:8080/public/mysql:latest\n name: mysql\n env:\n - name: POD_NAMESPACE\n valueFrom:\n fieldRef:\n fieldPath: metadata.namespace\n - name: MYSQL_ROOT_PASSWORD\n value: hhh\n ports:\n - containerPort: 8080\n name: admin-port\n - containerPort: 3306\n name: db-port\n resources:\n requests:\n cpu: 100m\n memory: 128Mi\n volumeMounts:\n - mountPath: /var/lib/mysql\n name: rethinkdb-storage\n- image: docker.oa.com:8080/public/2048:latest\n name: 2048app\n env:\n - name: POD_NAME\n value: 2048app\n ports:\n - containerPort: 80\n name: web\nvolumes:\n- name: rethinkdb-storage\n emptyDir: {}\nreplicas: 3\nannotation:\n- containerName: mysql\n logPath: /data1\n dataPath: /data2\n- containerName: 2048app\n logPath: /var/log"
}
```

compose编排接口

compose编排接口

最近更新时间: 2019-11-22 18:10:46

path 参数说明

projectName: 业务名

composeName: compose名

创建compose编排

最近更新时间: 2019-11-02 12:31:13

`POST /v2/composes/{projectName}` 请求body

```
{
"stackInstanceName": "compose",
"content": "c1:\n image: docker.oa.com:8080/public/helloworld\n replicas: 2\n mem_limit: 256m\n cpu_shares: 1\n environment:\n CONTAINER_NAME: c1\n app_NAME: app-c1\n links:\n - c2\n - c3\nc2:\n image: docker.oa.com:8080/public/helloworld\n replicas: 1\n mem_limit: 256m\n cpu_shares: 1\n environment:\n CONTAINER_NAME: c2\n app_NAME: app-c2\nc3:\n image: docker.oa.com:8080/public/helloworld\n replicas: 3\n mem_limit: 128m\n cpu_shares: 1\n environment:\n CONTAINER_NAME: c3\n app_NAME: app-c3\n"
}
```

返回

```
{
"Code": 0,
"Message": ""
}
```

获取compose编排

最近更新时间: 2019-11-02 12:30:12

`GET /v2/composes/{projectName}/{composeName}`

删除compose编排

最近更新时间: 2019-11-02 12:29:27

`DELETE /v2/composes/{projectName}/{composeName}` 返回

```
{
"Code": 0,
"Message": ""
}
```

名字服务接口

名字服务接口

最近更新时间: 2019-11-22 18:10:46

path 参数说明

projectName: 业务名

appName: app名

获取app名字服务

最近更新时间: 2019-11-22 18:10:46

`GET /v2/nameservice/{projectName}/{appName}?port={port}&protocol={protocol}&instanceName={instanceName}` 请求参数

- port: 容器端口号，不为空时只返回端口号为port的地址信息 // optional, default: “”
- protocol: 端口协议，可填tcp或udp，不为空时只返回协议为protocol的地址信息 // optional, default:””
- instanceName: 实例名称，不为空时只返回指定实例的地址信息 // optional, default:””
- 注：以上三个参数皆为空时，返回app的全部地址信息

返回信息说明

- privateAddr: 容器的集群内地址
- publicAddr: 容器的集群外可访问地址，若应用的网络类型为overlay, 则此项为空

实例端口映射信息

最近更新时间: 2019-11-22 18:10:46

`GET /v2/instances/{projectName}/{instanceName}/portmapping?instance_id={instance_id}` 请求参数

- instance_id: 实例ID
- size: 单页大小 // optional, default:10
- page: 页数 // optional, default:0

ConfigMap接口

ConfigMap接口

最近更新时间: 2019-11-22 18:10:46

path 参数说明

projectName: 业务名

configmapName: ConfigMap名称

keyName：ConfigMap配置项名称

验证ConfigMap名称

最近更新时间: 2019-11-02 13:07:32

`GET /v2/configmaps/{projectName}/validate?name={configMapName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

验证ConfigMap配置项名称

最近更新时间: 2019-11-02 13:06:26

`GET /v2/configmaps/{projectName}/{configMapName}/validate?key_name={keyName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

创建ConfigMap

最近更新时间: 2019-11-02 13:04:50

根据指定的内容创建ConfigMap `POST /v2/configmaps/{projectName}` 请求Body

```
{
"name": "special-config",
"data":{
"game-config":"\nSPECIAL_LEVEL_KEY=very\nSPECIAL_TYPE_KEY=charm\nlog_level=INFO",
"HELLO":"world"
}
}
```

返回

```
{
"name":"special-config",
"projectName": "demo",
"created_at": "2016-12-08T15:34:45.276506116+08:00",
"userName": "admin",
"configNum": 2,
"data":{
"game-config":"\nSPECIAL_LEVEL_KEY=very\nSPECIAL_TYPE_KEY=charm\nlog_level=INFO",
"HELLO":"world"
}
}
```

获取ConfigMap

最近更新时间: 2019-11-02 13:03:39

`GET /v2/configmaps/{projectName}/{configMapName}` 返回


```
{
"name": "special-config",
"projectName": "demo",
"created_at": "2016-12-08T15:34:45.276506116+08:00",
"userName": "admin",
"configNum": 2,
"data": {
"game-config": "\nSPECIAL_LEVEL_KEY=very\nSPECIAL_TYPE_KEY=charm\nlog_level=INFO",
"HELLO": "World"
},
"appMapNum": {
"game-config":2,
"HELLO": 1
}
}
```

添加或修改ConfigMap的配置项

最近更新时间: 2019-11-02 13:02:23

`POST /v2/configmaps/{projectName}/{configMapName}/{keyName}` 请求Body

```
{
"data":"Kitty"
}
```

返回

```
{
"name": "special-config",
"projectName": "demo",
"created_at": "2016-12-08T15:34:45.276506116+08:00",
"userName": "admin",
"configNum": 2,
"data": {
"game-config": "\nSPECIAL_LEVEL_KEY=very\nSPECIAL_TYPE_KEY=charm\nlog_level=INFO",
"HELLO": "Kitty"
},
"appMapNum": {
"game-config":2,
"HELLO": 1
}
}
```

获取ConfigMap某个Key对应的内容

最近更新时间: 2019-11-02 13:00:25

`GET /v2/configmaps/{projectName}/{configMapName}/{keyName}` 返回


```
{
"data":"SPECIAL_LEVEL_KEY=hasdhasjk\nSPECIAL_TYPE_KEY=happyone\nlog_level=INFO"
}
```

获取ConfigMap某个Key关联的app信息

最近更新时间: 2019-11-02 12:45:49

`GET /v2/configmaps/{projectName}/{configMapName}/{keyName}/apps` 返回


```
{
"content":[
{
"appId": "79c8a4f5ae13f2239ad8b8103ecacb19ec9137088427e062ffd7807cb163e84d",
"appName": "demoapp4",
"type": "Volume",
"mountPath": "/etc/config"
},
{
"appId": "fea3f59a50eda2416942d8c7ef5f190886e5b432eec6c5759c29711eb683a82a",
"appName": "demoapp5",
"type": "Volume",
"mountPath": "/etc/config"
}
]
}
```

删除ConfigMap的某个配置项

最近更新时间: 2019-11-02 12:41:14

`DELETE /v2/configmaps/{projectName}/{configMapName}/{keyName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

删除整个ConfigMap

最近更新时间: 2019-11-02 12:38:57

`DELETE /v2/configmaps/{projectName}/{configMapName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

Secret接口

Secret接口

最近更新时间: 2019-11-22 18:10:46

path 参数说明

projectName: 业务名

secretName: Secret名称

keyName：Secret配置项名称

验证Secret名称

最近更新时间: 2019-11-02 13:22:34

`GET /v2/secrets/{projectName}/validate?name={secretName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

验证Secret配置项名称

最近更新时间: 2019-11-02 13:21:21

`GET /v2/secrets/{projectName}/{secretName}/validate?key_name={keyName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

创建Secret

最近更新时间: 2019-11-02 13:20:06

根据指定的内容创建Secret `POST /v2/secrets/{projectName}` 请求Body

```
{
"name": "special-config",
"data":{
"game-config":"ClNQRUNJQUxfTEVWRUxfS0VZPXZlcnkKU1BFQ0lBTF9UWVBFX0tFWT1jaGFybQpsb2dfbGV2ZWw9SU5GTw==",
"HELLO":"MjIyMjIyCg=="
}
}
```

返回

```
{
"name":"special-config",
"projectName": "demo",
"created_at": "2016-12-08T15:34:45.276506116+08:00",
"userName": "admin",
"configNum": 2,
"data":{
"game-config":"ClNQRUNJQUxfTEVWRUxfS0VZPXZlcnkKU1BFQ0lBTF9UWVBFX0tFWT1jaGFybQpsb2dfbGV2ZWw9SU5GTw==",
"HELLO":"MjIyMjIyCg=="
}
}
```

获取Secret

最近更新时间: 2019-11-02 13:18:56

`GET /v2/secrets/{projectName}/{secretName}` 返回


```
{
"name": "special-config",
"projectName": "demo",
"created_at": "2016-12-08T15:34:45.276506116+08:00",
"userName": "admin",
"configNum": 2,
"data": {
"game-config":"ClNQRUNJQUxfTEVWRUxfS0VZPXZlcnkKU1BFQ0lBTF9UWVBFX0tFWT1jaGFybQpsb2dfbGV2ZWw9SU5GTw==",
"HELLO":"MjIyMjIyCg=="
},
"appMapNum": {
"game-config":2,
"HELLO": 1
}
}
```

添加或修改Secret的配置项

最近更新时间: 2019-11-02 13:16:56

`POST /v2/secrets/{projectName}/{secretName}/{keyName}` 请求Body

```
{
"data":"MTIzNDU2Cg=="
}
```

返回


```
{
"name": "special-config",
"projectName": "demo",
"created_at": "2016-12-08T15:34:45.276506116+08:00",
"userName": "admin",
"configNum": 2,
"data": {
"game-config": "ClNQRUNJQUxfTEVWRUxfS0VZPXZlcnkKU1BFQ0lBTF9UWVBFX0tFWT1jaGFybQpsb2dfbGV2ZWw9SU5GTw==",
"HELLO": "MTIzNDU2Cg=="
},
"appMapNum": {
"game-config":2,
"HELLO": 1
}
}
```

获取Secret某个Key对应的内容

最近更新时间: 2019-11-02 13:14:54

`GET /v2/secrets/{projectName}/{secretName}/{keyName}` 返回


```
{
"data":"ClNQRUNJQUxfTEVWRUxfS0VZPXZlcnkKU1BFQ0lBTF9UWVBFX0tFWT1jaGFybQpsb2dfbGV2ZWw9SU5GTw=="
}
```

获取Secret某个Key关联的app信息

最近更新时间: 2019-11-02 13:12:35

`GET /v2/secrets/{projectName}/{secretName}/{keyName}/apps` 返回


```
{
"content":[
{
"appId": "79c8a4f5ae13f2239ad8b8103ecacb19ec9137088427e062ffd7807cb163e84d",
"appName": "demoapp4",
"type": "Volume",
"mountPath": "/etc/config"
},
{
"appId": "fea3f59a50eda2416942d8c7ef5f190886e5b432eec6c5759c29711eb683a82a",
"appName": "demoapp5",
"type": "Volume",
"mountPath": "/etc/config"
}
]
}
```

删除Secret的某个配置项

最近更新时间: 2019-11-02 13:10:53

`DELETE /v2/secrets/{projectName}/{secretName}/{keyName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```

删除整个Secret

最近更新时间: 2019-11-02 13:09:30

`DELETE /v2/secrets/{projectName}/{secretName}` 返回

```
{
"Code": XXX,
"Message": "XXX"
}
```