阿里云
专有云敏捷版
容器服务

用户指南

产品版本：V1.1.0

文档版本：20190326

为了无法计算的价值 | 阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云文档中所有内容，包括但不限于图片、页面设计、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表本文档中的内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

表 -1: 格式约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	**禁止**： 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	**警告**： 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	**注意**： 导出的数据中包含敏感信息，请妥善保存。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	**说明**： 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	**设置** > **网络** > **设置网络类型**
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

法律声明.....I
通用约定.....I
1 什么是容器服务.....1
2 规划和准备.....2
3 如何使用特权账号.....3
4 快速入门.....5
4.1 登录容器服务管理控制台.....5
4.2 使用镜像创建应用.....6
5 集群.....9
5.1 通过kubectl连接Kubernetes集群.....9
5.2 查看集群概览.....9
5.3 查看节点列表.....10
5.4 删除集群.....11
6 节点.....13
6.1 查看节点列表.....13
6.2 管理节点标签.....14
7 存储.....16
7.1 使用NFS数据卷.....16
8 命名空间.....19
8.1 创建命名空间.....19
8.2 为命名空间配置资源额度.....21
8.3 编辑命名空间.....23
8.4 删除命名空间.....24
9 应用.....26
9.1 使用镜像创建应用.....26
9.2 使用编排模板创建应用.....28
9.3 通过Kubernetes Dashboard创建应用.....30
9.4 通过命令创建应用.....32
9.5 指定节点调度.....33
9.6 创建服务.....34
9.7 查看服务.....37
9.8 更新服务.....38
9.9 服务伸缩.....38
9.10 删除服务.....40

9.11 查看容器.....40
9.12 变更容器配置.....42
10 配置项.....43
10.1 创建配置项.....43
10.2 在 Pod 中使用配置项.....45
10.3 修改配置项.....48
10.4 删除配置项.....49
11 保密字典.....51
11.1 创建保密字典.....51
11.2 查看密钥.....53
11.3 编辑密钥.....53
11.4 删除密钥.....54
12 负载均衡.....56
12.1 通过负载均衡（Server Load Balancer）访问服务.....56
12.2 Ingress监控配置.....60
12.3 Ingress 支持.....62
13 创建配额.....68

1 什么是容器服务

Kubernetes是流行的开源容器编排技术。为了让用户可以方便地使用Kubernetes管理容器应用，容器服务-专有云Agility版提供了Kubernetes集群支持。

您可以通过容器服务控制台创建一个安全高可用的Kubernetes集群，整合阿里云虚拟化、存储、网络和安全能力，提供高性能可伸缩的容器应用管理能力，简化集群的搭建和扩容等工作，让您专注于容器化的应用的开发与管理。

Kubernetes支持对容器化应用程序的部署、扩展和管理。它具有以下功能：

集群管理

- 通过控制台一键创建Kubernetes集群。
- 提供容器优化的OS镜像，提供**稳定测试和安全加固**的Kuberntes和Docker版本。

一站式容器生命周期管理

- **网络**

  提供针对阿里云优化的高性能VPC/ENI网络插件性能。
- **存储**

  支持NFS 和CEPH存储。

  支持存储卷动态创建。
- **日志**

  支持和第三方开源日志解决方案集成。
- **监控**

  支持容器级别和VM级别的监控。您还可以和第三方开源监控解决方案进行集成。
- **应用管理**

  支持应用监控，应用弹性伸缩。

2 规划和准备

在使用阿里云容器服务-专有云Agility版之前，您需要规划后端的服务器，专有网络VPC。

3 如何使用特权账号

您可通过天基平台获取特权账号，为当前账号授予最大的权限。

前提条件

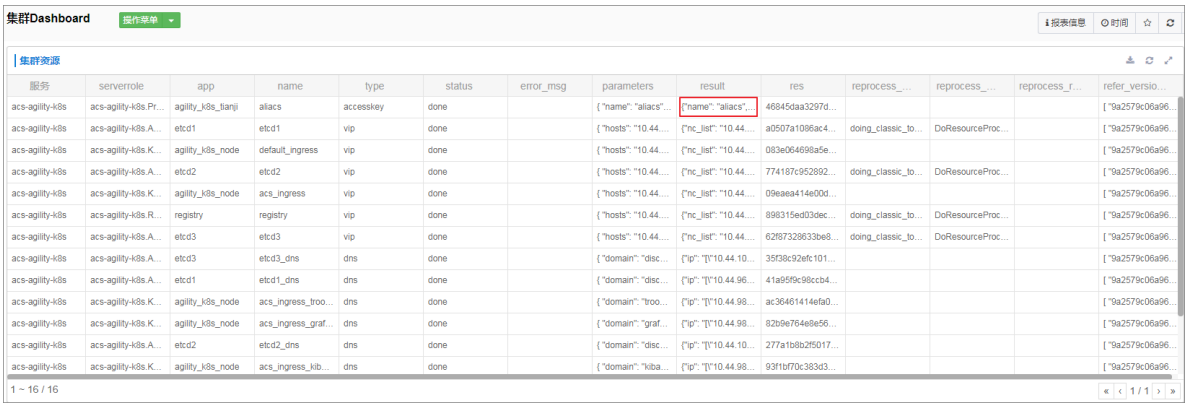
您需要拥有一个Apsara Stack运维系统的账号。

操作步骤

1. 用 ASO 提供的账号登录Apsara Stack运维系统 。
2. 单击左侧菜单栏**产品运维管理**，单击右侧**容器服务Kubernetes版**控制台，此时进入阿里云容器服务 - Apsara Stack Agility控制台，此时可以看到右上角是 xxx 用户的身份登录的。

3. 退出阿里云容器服务 - Apsara Stack Agility控制台，登录天基控制台。
4. 在天基控制台中，在顶部菜单栏单击**运维** > **集群运维**，单击AgilityOnePc 集群。

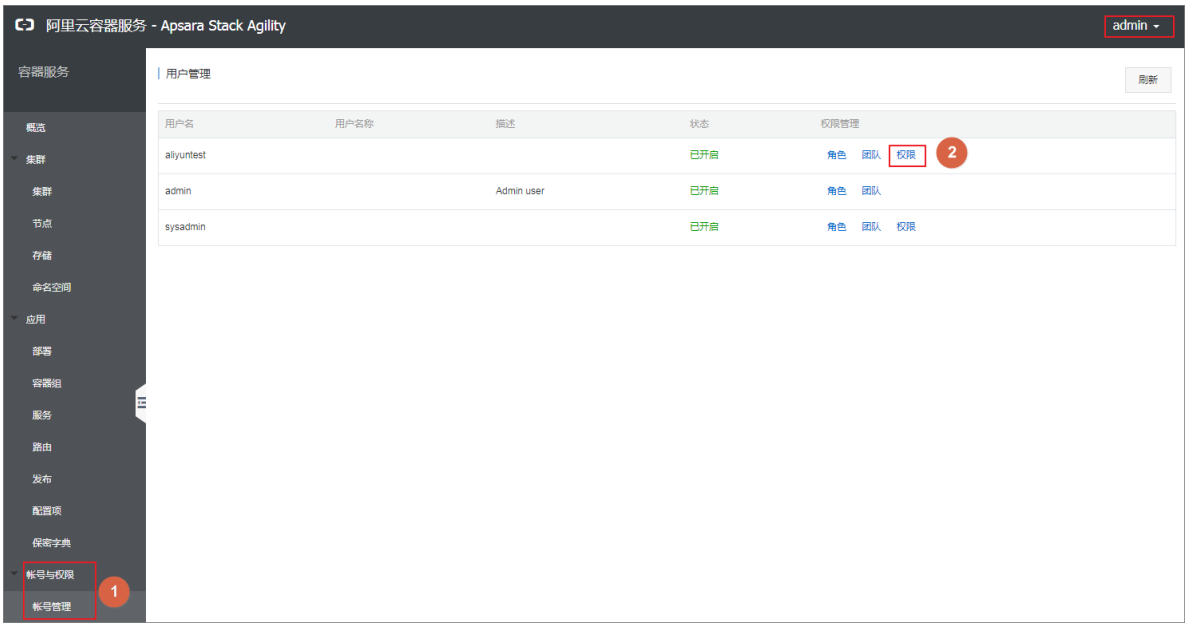
5. 在集群Dashboard中，查看集群资源中 accesskey，如下：



result列中对应的内容如下：

```
{"name": "aliacs", "accesskey-id": "****************", "
accesskey-secret": "************************", "user": "aliacs
@aliyun.com", "password": "**************************", "id":
 "********************"}
```

6. 使用其中的user和 password 登录 ASO，以同样方法登录容器服务，打开系统设置，在用户权限管理处，给需要的账号授予最大的权限。



4 快速入门

本章节介绍了如何快速部署Kubernetes集群，添加Kubernetes集群，并使用镜像创建应用，旨在引导您快速了解容器服务的快速使用。

4.1 登录容器服务管理控制台

本节介绍运维工程师等相关用户如何登录到Apsara Stack诊断系统中的容器服务Kubernetes版中。

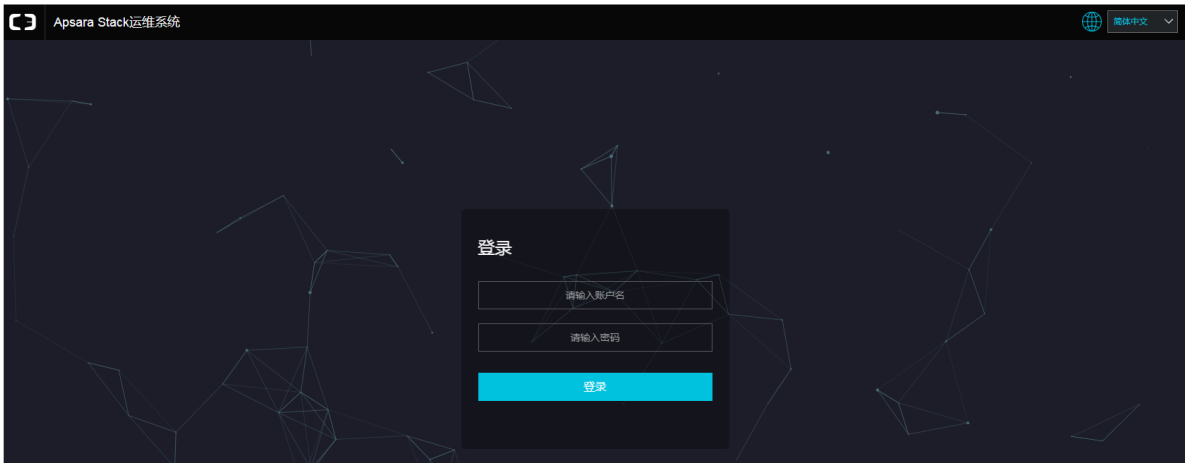
前提条件

- 已获取Apsara Stack运维系统的访问地址。Apsara Stack运维系统访问地址格式为http://*region-id*.aso.*intranet-domain-id*.com。
- 推荐使用Chrome浏览器。

操作步骤

1. 打开浏览器。
2. 在地址栏中，输入Apsara Stack运维系统的访问地址http://*region-id*.aso.*intranet-domain-id*.com，按回车键。

   图 4-1: 登录Apsara Stack运维系统页面

   说明：
   您可以单击页面右上角的下拉按钮来进行**简体中文**和**English**的语言切换。

3. 输入正确的用户名及密码。
   - 系统默认有2个用户：

- 系统管理员：管理其他用户或角色。
- 普通用户：被系统管理员授权后，才有相关的权限。

- 首次登录Apsara Stack运维控制台时，需要修改登录用户名的密码，请按照提示完成密码修改。为提高安全性，密码必须满足最小复杂度要求，即包含英文大/小写字母（A~Z、a~z）、数字（0~9）、特殊符号（！、@、#、$、% 等）。

4. 单击**登录**，进入**Apsara Stack运维系统**页面。
5. 在页面左侧导航栏中，单击**产品运维管理**，然后单击页面上的**容器服务Kubernetes版**按钮。

4.2 使用镜像创建应用

您可以在容器服务控制台使用镜像在Kubernetes集群中创建应用。

前提条件

您必须先部署一个Kubernetes集群。

使用镜像创建应用前，您需要先将镜像上传到DTR。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **部署**进入部署列表页面。
3. 单击页面右上角的**使用镜像创建**。
4. 在创建应用的页面中，根据页面提示进行配置。

表 4-1: 应用参数配置表

类别	配置	说明
应用基本信息配置	**应用名称**	所创建应用的名称。名称可以包含1~64个字符，包括数字、英文字母和连字符（-）
	部署集群	所创建应用将要部署到的集群。
	命名空间	所创建应用的命名空间。如果您不设置**命名空间**，系统会默认使用default命名空间。
应用配置	**镜像名称**	填写所使用的镜像信息。
	镜像版本	填写镜像的版本。若不指定，默认为latest。
	容器数量	可指定容器的数量，本示例是单容器Pod，若指定多个容器，会启动相同数量的Pod。

类别	配置	说明
	资源限制	可指定该应用所能使用的资源上限，包括CPU和内存两种资源，防止占用过多资源。 **说明**： 其中，CPU资源的单位为millicores，即一个核的千分之一；内存的单位为Bytes，可以为Gi、Mi或Ki
	所需资源	即为该应用预留资源额度，包括CPU和内存两种资源，防止因资源不足而被其他服务或进程争抢资源，导致应用不可用。
	数据卷	支持配置hostpath类型的数据卷。hostpath类型的数据卷会将主机文件系统中的文件或目录挂载到pod 中。更多信息参见 *volumes* 关于hostpath的信息。
	环境变量	支持通过键值对的形式为Pod配置环境变量。用于给Pod添加环境标志或传递配置等，具体请参见 *Pod variable*。
	Command	为容器设置启动命令，若不配置，会使用镜像默认的设置；若配置，则会覆盖镜像里的默认设置
	Args	为容器设置启动参数，若配置，则会覆盖镜像里的默认设置， **说明**： 并且如果只配置Arguments，容器启动时，默认的Command会执行新的Arguments。Command和Arguments 在Pod创建之后不可更改。
	容器启动项	勾选stdin表示为该容器开启标准输入；勾选tty表示为该容器分配一个虚拟终端，以便于向容器发送信号。通常这两个选项是一起使用的，表示将终端 (tty) 绑定到容器的标准输入（stdin）上，比如一个交互式的程序从用户获取标准输入，并显示到终端中。
	自动伸缩	您可勾选是否开启 自动伸缩，为了满足应用在不同负载下的需求，容器服务支持服容器的弹性伸缩，即根据容器CPU 和内存资源占用情况自动调整容器数量。 • 指标：支持CPU和内存，需要和设置的所需资源类型相同。

类别	配置	说明
		• 触发条件：资源使用率的百分比，超过该使用量，容器开始扩容。 • 最大容器数量：该Deployment可扩容的容器数量上限。 • 最小容器数量：该Deployment可缩容的容器数量下限。
访问设置	**服务**	您可以从下面 2 种服务类型中进行选择。 • 不创建：即不为该应用建服务。 • 虚拟集群 IP：即 ClusterIP，指通过集群的内部 IP 暴露服务，选择该项，服务只能够在集群内部可以访问。 • 节点端口：即 NodePort，通过每个 Node 上的 IP 和静态端口（NodePort）暴露服务。NodePort 服务会路由到 ClusterIP 服务，这个 ClusterIP 服务会自动创建。通过请求 `<NodeIP>:<NodePort>`，可以从集群的外部访问一个 NodePort 服务。
	名称	默认会生成一个应用名后缀 svc 的服务名称，也可更改服务的名称。您可自主设置。
	端口映射	您需要添加服务端口和容器端口，若您选择节点端口的服务类型，还需要自己设置节点端口，防止端口出现冲突。支持 TCP/UDP 协议。

5. 完成应用配置后，单击**创建**。

   创建成功后，进入创建完成页面，您可以单击**Kubernetes控制台**进入Kubernetes Dashboard，查看应用详情。

5 集群

您可通过此章节了解容器服务Kubernetes集群的基本功能。

5.1 通过kubectl连接Kubernetes集群

如果您需要从客户端计算机连接到Kubernetes集群，请使用Kubernetes命令行客户端*kubectl*。

操作步骤

1. 从*Kubernetes*版本页面下载最新的kubectl客户端。
2. 安装和设置kubectl客户端。

   有关详细信息，参见安装和设置*kubectl*。
3. 配置集群凭据。

   您可以使用**scp**命令安全地将主节点的配置从Kubernetes集群主VM中的`/etc/kubernetes/kube.conf`复制到本地计算机的`$HOME/.kube/config`（kubectl预期凭据所在的位置）。 例如：

   ```
   mkdir $HOME/.kube
   scp root@<master-public-ip>:/etc/kubernetes/kube.conf $HOME/.kube/
   config
   ```

   配置完成后，您即可以使用kubectl从本地计算机访问Kubernetes集群。

5.2 查看集群概览

阿里云容器服务 Kubernetes 集群提供集群概览概览功能，提供应用状态、组件状态和资源监控等功能，方便您快速了解集群的健康状态信息。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**概览**，进入 Kubernetes 集群概览页面。
3. 选择所需的集群和命名空间，您可查看应用状态、组件状态和资源监控图表。
   - **应用状态**：显示当前运行的部署、容器组和副本集的状态示意图，绿色图标代表正常，黄色图标代表异常。
   - **节点状态**：显示当前集群中的节点状态示意图。
   - **组件状态**：Kubernetes 集群的组件通常部署在 kube-system 命名空间下，包括scheduler、controller-manager和 etcd 等核心组件。

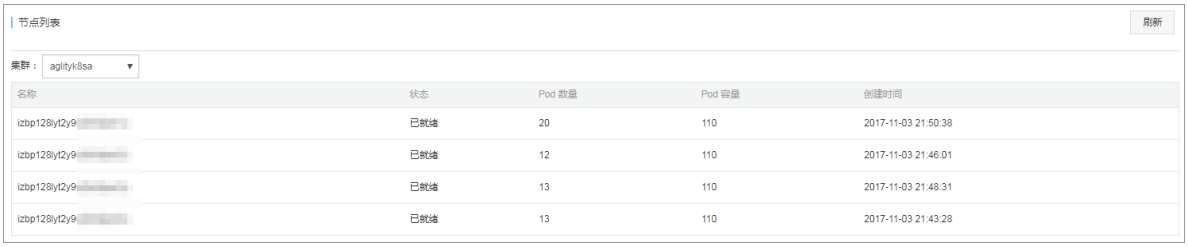
- **资源监控**：提供 CPU 和内存的监控图表。CPU 统计单位为 Cores（核），可显示小数点后 3 位，最小统计单位是 millcores，即一个核的 1/1000；内存的统计单位是 Gi，显示小数点后 3 位。更多相关信息，请参见 *Meaning of CPU*和 *Meaning of memory*。

5.3 查看节点列表

您可以通过命令、容器服务控制台的节点列表页面或者Kubernetes Dashboard的节点列表页面查看Kubernetes集群的节点列表。

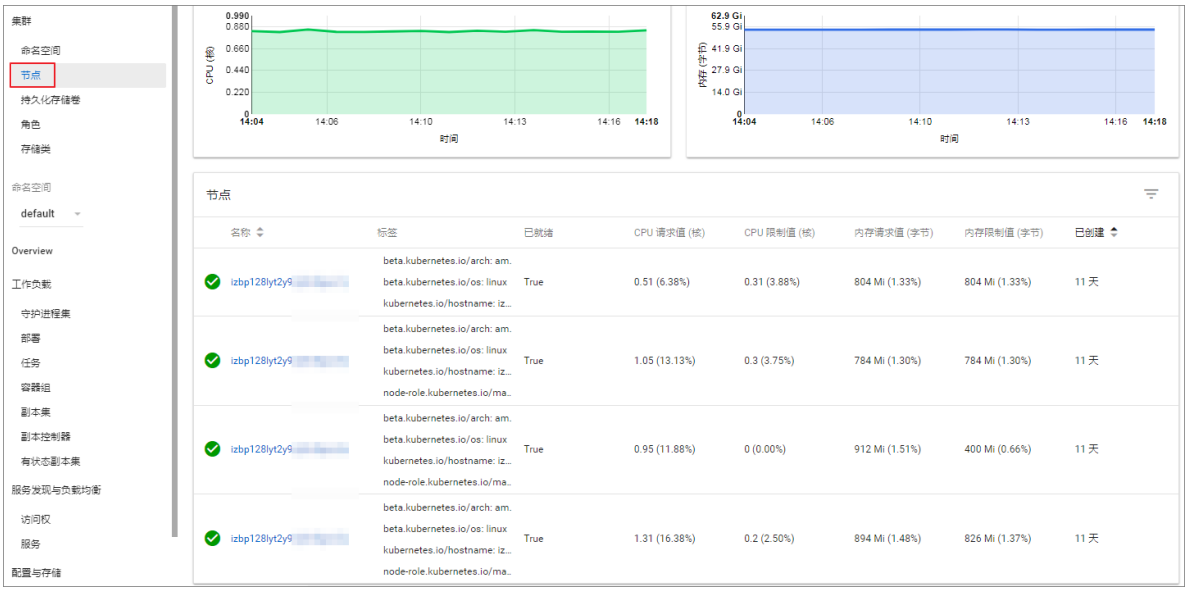
通过容器服务控制台查看

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **节点**进入节点列表页面。
3. 选择所需的集群，您可以查看该集群下的节点列表。



通过Kubernetes Dashboard查看

1. 登录容器服务管理控制台。
2. 在**Kubernetes**菜单下，单击**集群**进入Kubernetes集群列表页面。
3. 选择所需的集群并单击右侧的**控制台**，进入Kubernetes Dashboard。
4. 在Kubernetes Dashboard中，单击左侧导航栏中的**节点**。即可查看集群中的节点列表。



通过命令查看

说明：

使用命令查看Kubernetes集群的节点列表页面之前，您需要先设置通过*kubectl*连接*Kubernetes*集群。

通过kubectl连接到Kubernetes集群后，运行以下命令查看集群中的节点。

```
kubectl get nodes
```

输出示例：

```
$ kubectl get nodes
NAME                      STATUS    AGE       VERSION
iz2ze2n6ep53tch701yh9zz   Ready     19m       v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39e5az   Ready     7m        v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39e5bz   Ready     7m        v1.6.1-2+ed9e3d33a07093
iz2zef4dnn9nos8elyr32kz   Ready     14m       v1.6.1-2+ed9e3d33a07093
iz2zeitvvo8enoreufstkmz   Ready     11m       v1.6.1-2+ed9e3d33a07093
```

5.4 删除集群

您可以通过容器服务控制台删除不再使用的集群。

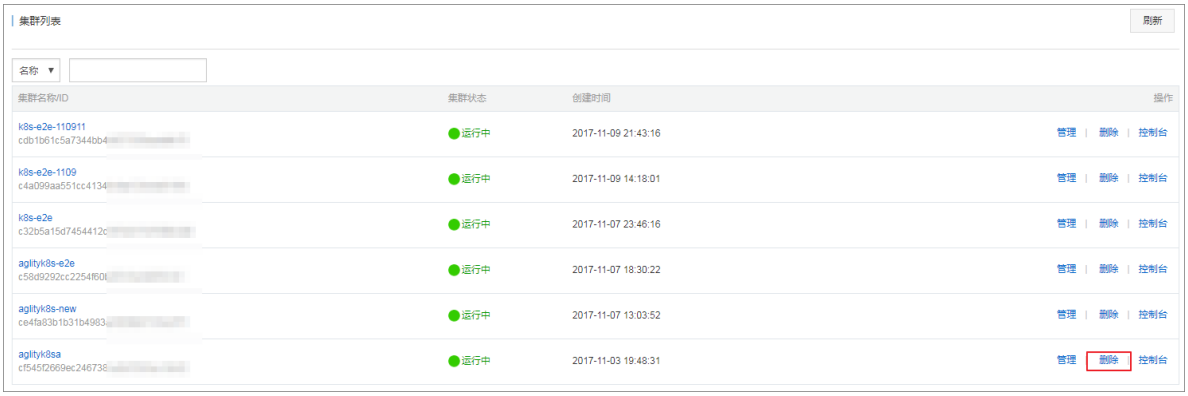
注意事项

此操作仅会从控制台界面上删除该集群，但是在后台中该集群仍然存在。

操作步骤

1. 登录容器服务管理控制台。

2. 单击左侧导航栏中的**集群** > **集群**进入Kubernetes集群列表页面。
3. 选择所需的集群并单击操作列的**删除**。



6 节点

您可查看某个集群下的节点列表；并可为节点添加标签，方便应用进行指定节点调度。

6.1 查看节点列表

您可通过多种方式查看集群的节点信息。

您可以通过命令、容器服务管理控制台的节点列表页面或者 Kubernetes Dashboard 的节点列表页面查看 Kubernetes 集群的节点列表。

通过命令查看

说明：

使用命令查看 Kubernetes 集群的节点列表页面之前，您需要先设置 *通过kubectl连接Kubernetes集群*。

通过 kubectl 连接到 Kubernetes 集群后，运行以下命令查看集群中的节点。

```
kubectl get nodes
```

输出示例：

```
$ kubectl get nodes
    NAME                        STATUS    AGE       VERSION
    iz2ze2n6ep53tch701yh9zz     Ready     19m       v1.6.1-2+ed9e3d33a0
7093
    iz2zeafr762wibijx39e5az     Ready     7m        v1.6.1-2+ed9e3d33a0
7093
    iz2zeafr762wibijx39e5bz     Ready     7m        v1.6.1-2+ed9e3d33a0
7093
    iz2zef4dnn9nos8elyr32kz     Ready     14m       v1.6.1-2+ed9e3d33a0
7093
    iz2zeitvvo8enoreufstkmz     Ready     11m       v1.6.1-2+ed9e3d33a0
7093
```

通过容器服务管理控制台查看

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **节点**，进入节点列表页面。
3. 选择所需的集群，您可以查看该集群下的节点列表。

通过 Kubernetes Dashboard 查看

1. 登录容器服务管理控制台。

2. 单击左侧导航栏中的**集群** > **集群**，进入 Kubernetes 集群列表页面。
3. 选择所需的集群并单击操作列的**控制台**，进入 Kubernetes Dashboard。
4. 在 Kubernetes Dashboard 中，单击左侧导航栏中的**节点**。即可查看集群中的节点列表。

6.2 管理节点标签

您可通过Web界面对节点进行标签管理。

您可以通过容器服务 Web 界面对节点进行标签管理，包括批量添加节点标签、通过标签筛选节点和快速删除节点标签。

关于如何使用节点标签实现节点调度，请参见指定节点调度。

前提条件

您已经成功创建一个 Kubernetes 集群。

批量添加节点标签

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **节点**，进入节点列表页面。
3. 选择所需的集群，在页面右上角单击**标签管理**。
4. 在节点列表中，批量选择节点，然后单击**添加标签**。
5. 在弹出的添加标签对话框中，输入标签的名称和值，然后单击**确定**。

   您可以在标签管理页面，看到批量节点具有相同的标签。

通过标签筛选节点

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **节点**，进入节点列表页面。
3. 选择所需的集群，在页面右上角单击**标签管理**。
4. 选择某个节点，单击右侧的标签，如 `group:worker`，可通过标签来筛选节点。

   您可看到通过 `group:worker` 标签成功筛选出所需的节点。

删除节点标签

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **节点**，进入节点列表页面。
3. 选择所需的集群，在页面右上角单击**标签管理**。

4. 选择某个节点，单击标签的删除图标，如 `group:worker`。

   您可以看到该节点右侧的标签消失，节点标签被删除。

7 存储

容器服务Kubernetes集群支持NFS和CEPH类型的数据卷，您需要在集群部署环境中预先创建对应的文件系统。

7.1 使用NFS数据卷

您可以在容器服务 Kubernetes 集群中使用 NFS 数据卷。

操作步骤

1. 创建 PV。

   您可以使用 yaml 文件或者通过界面创建 NFS 数据卷。

   - **使用 yaml 文件创建 PV**

     a. 登录容器服务管理控制台。

     b. 单击左侧导航栏中的**应用** > **部署**进入应用列表页面。

     c. 单击页面右上角的**使用模板创建**。

     d. 对模板进行配置来创建NFS数据卷。

     表 7-1: 通过模板创建NFS数据卷

集群	选择目标集群。数据卷将部署在该集群内。
命名空间	选择命名空间，默认为default。
示例模板	您可以选择容器服务提供的NFS yaml示例模板（Volume - NFS）或者选择**自定义**并编写您自己的yaml模板。本示例使用Volume - NFS示例模板。
模板	您可自己编辑Kubernetes Yaml 编排模板；或者单击**使用已有模板**。

     e. 完成配置后，单击**部署**。

   - **通过界面创建 NFS 数据卷**

     a. 登录容器服务管理控制台。

     b. 单击左侧导航栏中的**集群** > **存储**，并单击页面右上角的**创建**。

     c. 在创建数据卷对话框中，配置数据卷的相关参数。

表 7-2: 通过界面创建NFS数据卷

数据卷类型	本示例中为NFS。
数据卷名	创建的数据卷的名称。数据卷名在集群内必须唯一。
总量	所创建数据卷的容量。注意不能超过磁盘容量。
访问模式	默认为 ReadWriteMany。
服务IP	NFS server所在主机的IP地址。
路径	共享目录
回收策略	可设为Recycle或Retain。

d. 完成配置后，单击**创建**。

2. 创建 PVC。

   a. 登录容器服务管理控制台。

   b. 单击左侧导航栏中的**应用** > **部署**进入应用列表页面。

   c. 单击页面右上角的**使用模板创建**。

   d. 对模板进行配置来创建NFS数据卷索取（PVC）。

   表 7-3: 通过模板创建PVC

集群	选择目标集群。数据卷将部署在该集群内。
命名空间	选择命名空间，默认为default。
编排模板	您可以选择容器服务提供的数据卷索取 yaml示例模板（Resource - PersistentVolumeClaim）或者选择**自定义**并编写您自己的yaml模板。本示例使用Resource - PersistentVolumeClaim示例模板。

   e. 完成配置后，单击**创建**。

3. 创建 Pod。

   a. 登录容器服务管理控制台。

   b. 单击左侧导航栏中的**应用** > **部署**进入应用列表页面。

c. 创建一个Pod模板，用来使用PVC。

表 7-4: 通过模板创建Pod

配置	说明
集群	选择目标集群。数据卷将部署在该集群内。
命名空间	选择命名空间，默认为default。

编排模板	选择**自定义**并编写您自己的yaml模板。
	本示例中的模板如下所示。
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  name: "nfstest"`
	`spec:`
	`  containers:`
	`  - name: "nginx"`
	`  image: "nginx"`
	`  volumeMounts:`
	`  - name: nfs-pv4`
	`  mountPath: "/data"`
	`  volumes:`
	`  - name: nfs-pv4`
	`  persistentVolumeClaim:`
	`  claimName: nfs-pvc1`

d. 完成配置后，单击**创建**。

8 命名空间

命名空间是Kubernetes编排系统的一个原生概念，充当一个虚拟的集群，用于多种工作用途，满足多用户的使用需求。

8.1 创建命名空间

您可通过界面创建命名空间。

前提条件

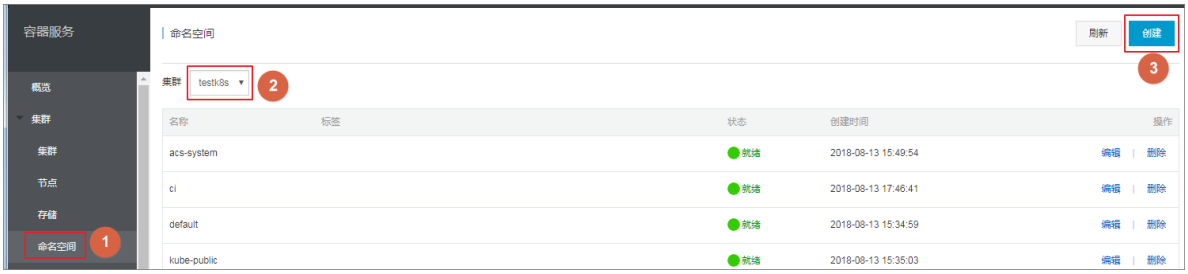
您已成功创建一个 Kubernetes 集群。

背景信息

在 Kubernetes 集群中，您可使用 Namespaces（命名空间）功能创建多个虚拟的空间，在集群用户数量较多时，多个命名空间可以有效划分工作区间，将集群资源划分为多个用途，并通过 *resource-quotas* 对命名空间的资源进行分配。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏的**集群** > **命名空间**，进入命名空间列表页面。
3. 选择所需的集群，单击页面右上角的**创建**。



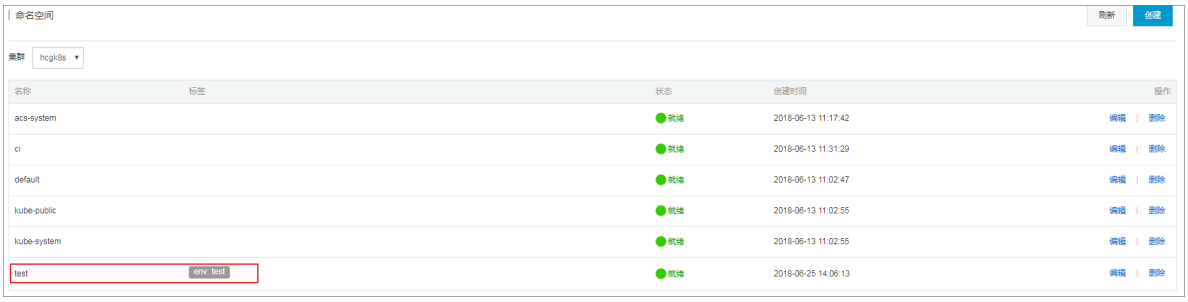
4. 在弹出的对话框中，配置命名空间。

表 8-1: 创建命名空间

配置	说明
名称	您需要设置命名空间的名称，本例中为 test 。长度为 1-63 个字符，只能包含数字、字母、和 “-” ，且首尾只能是字母或数字。
	标签：您可为命名空间添加多个标签。标签用于标识该命名空间的特点，如标识该命名空间用于测试环境。 您可输入变量名称和变量值，单击右侧的**添加**，为命名空间新增一个标签。

- ：
-

5. 完成命名空间设置后，单击**确定**。
6. 返回命名空间列表，您可看到 test 命名空间出现在列表中，命名空间成功创建。



8.2 为命名空间配置资源额度

您可通过为命名空间配置资源额度，来合理分配资源。

前提条件

- 您已成功创建一个 Kubernetes 集群。
- 您已成功创建一个示例的命名空间**test**，参见创建命名空间。
- 通过Kubectl连接到Kubernetes集群，参见*通过kubectl连接Kubernetes集群*。

背景信息

在默认的情况下，运行中的 Pod 可以无限制的使用 Node 上的 CPU 和内存，这意味着任意一个 Pod 都可以无节制地使用集群的计算资源，某个命名空间的 pod 可能会耗尽集群的资源。

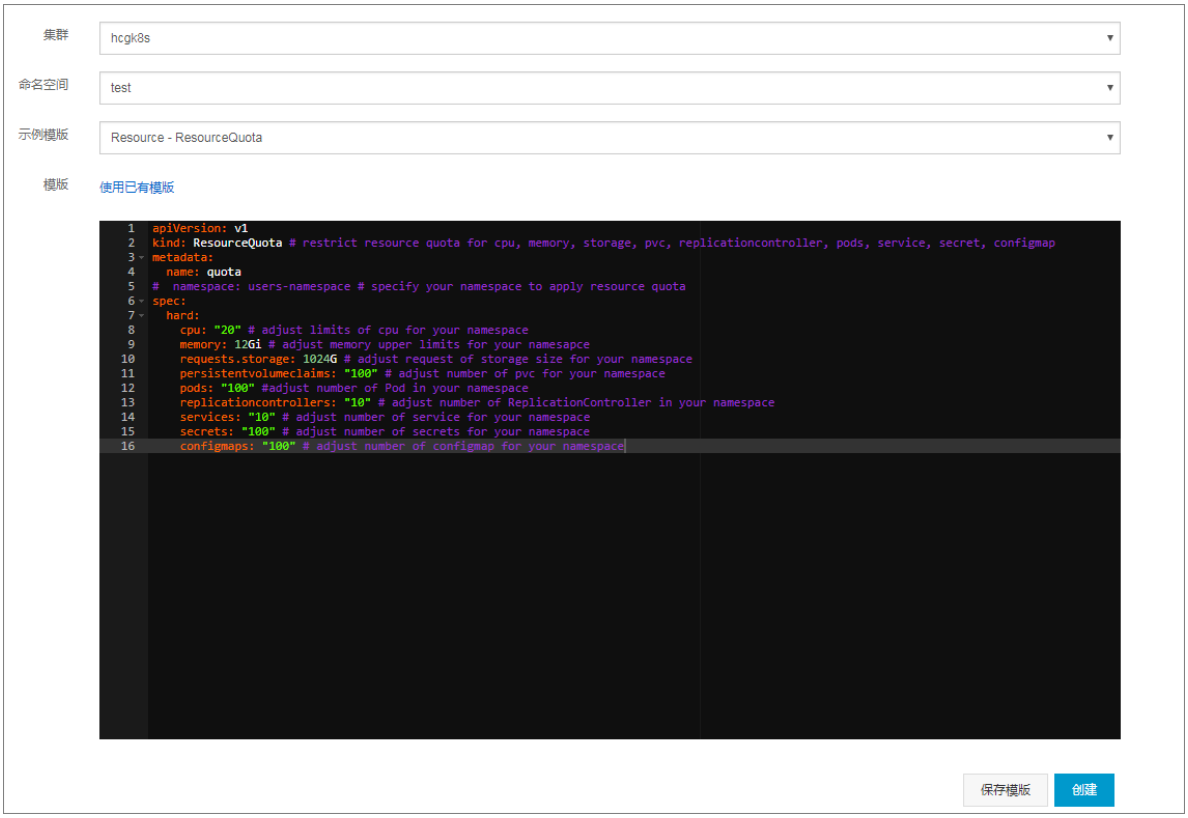
命名空间的一个重要的作用是充当一个虚拟的集群，用于多种工作用途，满足多用户的使用需求，因此，为命名空间配置资源额度是一种最佳实践。

您可为命名空间配置包括 CPU、内存、Pod 数量等资源的额度，更多信息请参见 *Resource Quotas*。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏的**应用** > **部署**，然后单击右上角的**使用模板创建**。

3. 在使用模板创建页面中，选择所需的集群和示例命名空间**test**，使用自定义或示例模板Resource-ResourceQuota。



您需要根据自己集群的资源情况，以及对该命名空间的资源规划，对 ResourceQuota 模板进行配置，本例的模板如下。

```
apiVersion: v1
kind: ResourceQuota # restrict resource quota for cpu, memory
, storage, pvc, replicationcontroller, pods, service, secret,
configmap
metadata:
  name: quota
  namespace: test # specify your namespace to apply resource quota
spec:
  hard:
    cpu: "20" # adjust limits of cpu for your namespace
    memory: 12Gi # adjust memory upper limits for your namesapce
    requests.storage: 1024G # adjust request of storage size for
your namespace
    persistentvolumeclaims: "100" # adjust number of pvc for your
namespace
    pods: "100" #adjust number of Pod in your namespace
    replicationcontrollers: "10" # adjust number of Replicatio
nController in your namespace
    services: "10" # adjust number of service for your namespace
    secrets: "100" # adjust number of secrets for your namespace
```

```
    configmaps: "100" # adjust number of configmap for your
namespace
```

4. 您已成功为该命名空间配置了资源额度，连接到 Master 节点 SSH 连接地址，执行以下命令，查看该命名空间的资源限额和使用情况。

```
# kubectl describe quota quota --namespace=test
Name:                   quota
Namespace:              test
Resource                Used  Hard
--------                ----  ----
configmaps              0     100
cpu                     0     20
memory                  0     12Gi
persistentvolumeclaims  0     100
pods                    0     100
replicationcontrollers  0     10
requests.storage        0     1024G
secrets                 1     100
services                0     10
```

8.3 编辑命名空间

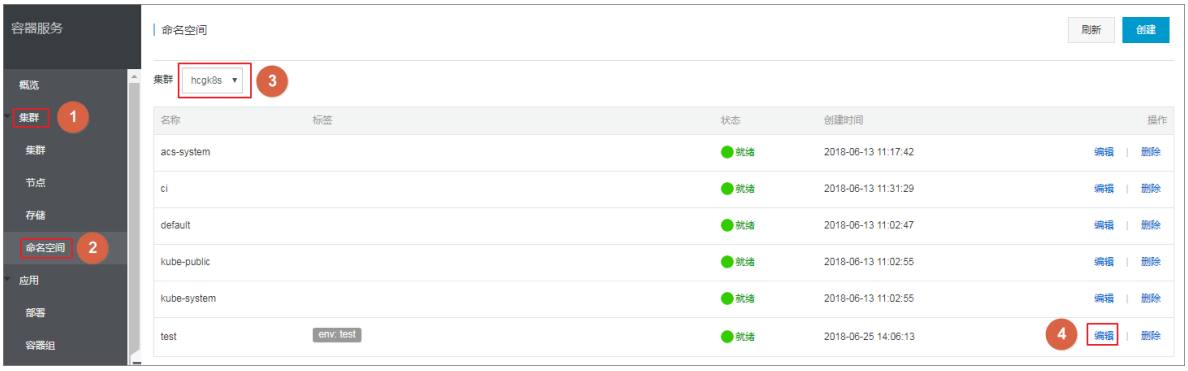
您可对命名空间进行编辑，对命名空间的标签进行增、删、改等操作。

前提条件

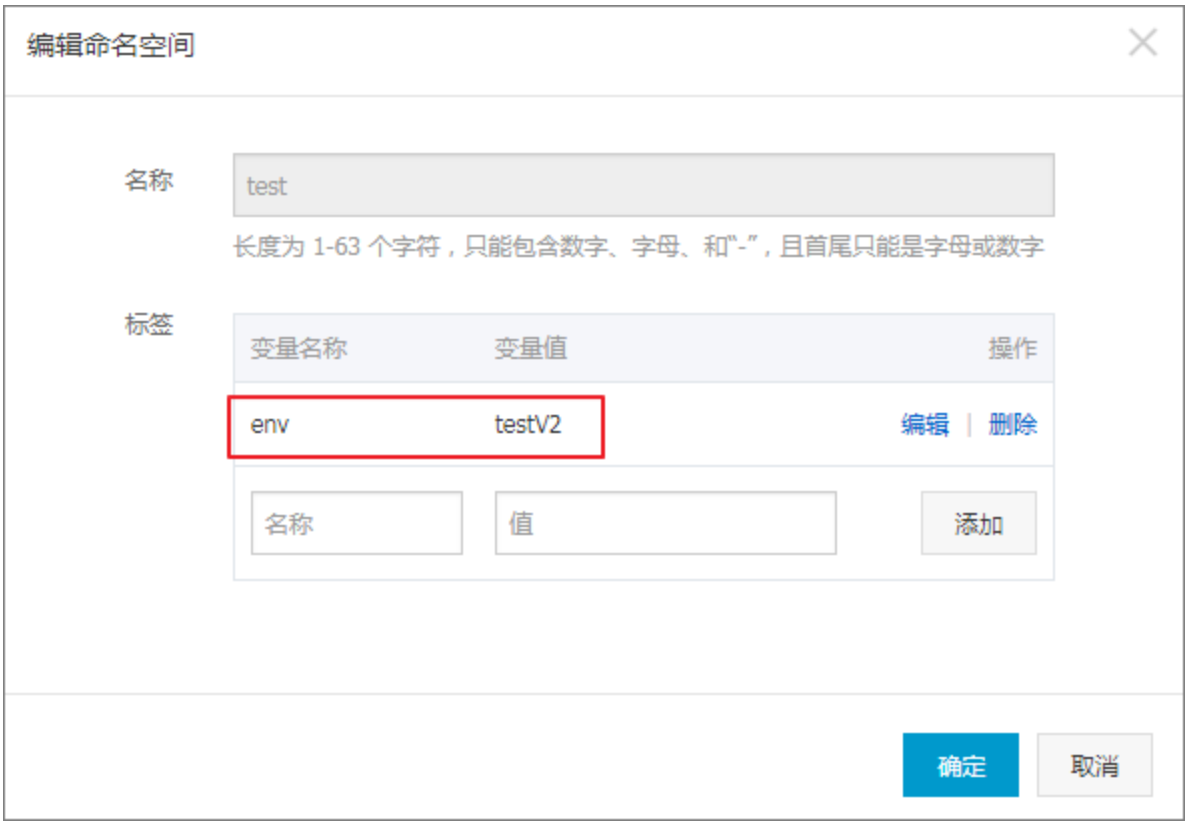
- 您已成功创建一个 Kubernetes 集群。
- 您已成功创建一个示例的命名空间**test**，参见创建命名空间。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏的**集群** > **命名空间**，进入命名空间列表页面。
3. 选择所需的集群，选择所需的命名空间，单击操作列的**编辑**。



4. 在弹出的对话框中，对命名空间的标签进行修改，如将标签修改为`env:test-V2`



5. 单击**确定**，返回命名空间列表，该命名空间的标签发生变化。

8.4 删除命名空间

您可删除不再需要的命名空间。

前提条件

- 您已成功创建一个 Kubernetes 集群。
- 您已成功创建一个示例的命名空间**test**，参见创建命名空间。

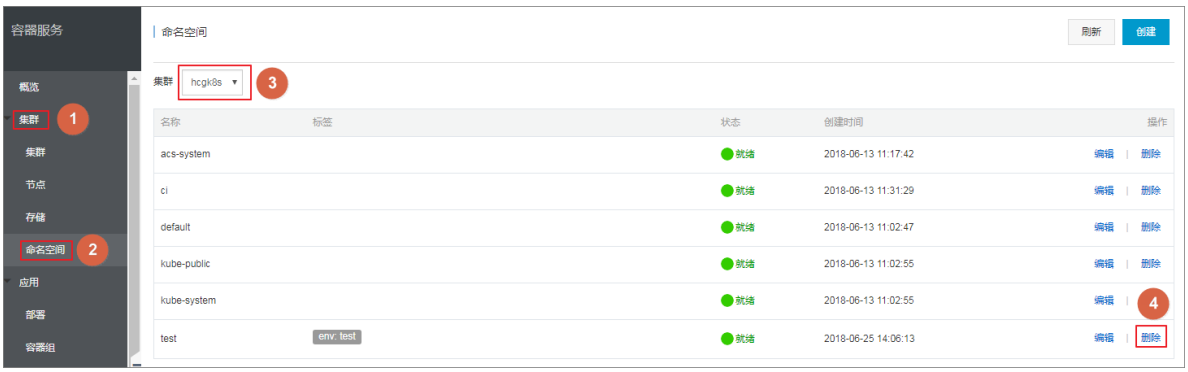
背景信息

说明：

删除命名空间会导致其下所有的资源对象一起被删除，请慎重操作。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏的**集群** > **命名空间**，进入命名空间列表页面。
3. 选择所需的集群，选择所需的命名空间，单击操作列的**删除**。



4. 在弹出的提示框中，单击**确定**。

5. 返回命名空间列表，您可看到该命名空间已被删除，其下的资源对象也会被删除。

9 应用

9.1 使用镜像创建应用

您可以在容器服务控制台使用镜像在Kubernetes集群中创建应用。

前提条件

您必须先部署一个Kubernetes集群。

使用镜像创建应用前，您需要先将镜像上传到DTR。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **部署**进入部署列表页面。
3. 单击页面右上角的**使用镜像创建**。
4. 在创建应用的页面中，根据页面提示进行配置。

表 9-1: 应用参数配置表

类别	配置	说明
应用基本信息配置	**应用名称**	所创建应用的名称。名称可以包含1~64个字符，包括数字、英文字母和连字符（-）
	部署集群	所创建应用将要部署到的集群。
	命名空间	所创建应用的命名空间。如果您不设置**命名空间**，系统会默认使用default命名空间。
应用配置	**镜像名称**	填写所使用的镜像信息。
	镜像版本	填写镜像的版本。若不指定，默认为latest。
	容器数量	可指定容器的数量，本示例是单容器Pod，若指定多个容器，会启动相同数量的Pod。
	资源限制	可指定该应用所能使用的资源上限，包括CPU和内存两种资源，防止占用过多资源。 **说明**： 其中，CPU资源的单位为millicores，即一个核的千分之一；内存的单位为Bytes，可以为Gi、Mi或Ki

类别	配置	说明
	所需资源	即为该应用预留资源额度，包括CPU和内存两种资源，防止因资源不足而被其他服务或进程争抢资源，导致应用不可用。
	数据卷	支持配置hostpath类型的数据卷。hostpath类型的数据卷会将主机文件系统中的文件或目录挂载到pod 中。更多信息参见 *volumes* 关于hostpath的信息。
	环境变量	支持通过键值对的形式为Pod配置环境变量。用于给Pod添加环境标志或传递配置等，具体请参见 *Pod variable*。
	Command	为容器设置启动命令，若不配置，会使用镜像默认的设置；若配置，则会覆盖镜像里的默认设置
	Args	为容器设置启动参数，若配置，则会覆盖镜像里的默认设置， **说明**： 并且如果只配置Arguments，容器启动时，默认的Command会执行新的Arguments。Command和Arguments 在Pod创建之后不可更改。
	容器启动项	勾选stdin表示为该容器开启标准输入；勾选tty表示为该容器分配一个虚拟终端，以便于向容器发送信号。通常这两个选项是一起使用的，表示将终端 (tty) 绑定到容器的标准输入（stdin）上，比如一个交互式的程序从用户获取标准输入，并显示到终端中。
	自动伸缩	您可勾选是否开启 自动伸缩，为了满足应用在不同负载下的需求，容器服务支持服容器的弹性伸缩，即根据容器CPU 和内存资源占用情况自动调整容器数量。 • 指标：支持CPU和内存，需要和设置的所需资源类型相同。 • 触发条件：资源使用率的百分比，超过该使用量，容器开始扩容。 • 最大容器数量：该Deployment可扩容的容器数量上限。 • 最小容器数量：该Deployment可缩容的容器数量下限。
访问设置	**服务**	您可以从下面 2 种服务类型中进行选择。

类别	配置	说明
		• 不创建：即不为该应用建服务。 • 虚拟集群 IP：即 ClusterIP，指通过集群的内部 IP 暴露服务，选择该项，服务只能够在集群内部可以访问。 • 节点端口：即 NodePort，通过每个 Node 上的 IP 和静态端口（NodePort）暴露服务。NodePort 服务会路由到 ClusterIP 服务，这个 ClusterIP 服务会自动创建。通过请求 `<NodeIP>:<NodePort>`，可以从集群的外部访问一个 NodePort 服务。
	名称	默认会生成一个应用名后缀 svc 的服务名称，也可更改服务的名称。您可自主设置。
	端口映射	您需要添加服务端口和容器端口，若您选择节点端口的服务类型，还需要自己设置节点端口，防止端口出现冲突。支持 TCP/UDP 协议。

5. 完成应用配置后，单击**创建**。

   创建成功后，进入创建完成页面，您可以单击**Kubernetes控制台**进入Kubernetes Dashboard，查看应用详情。

9.2 使用编排模板创建应用

您可以在容器服务控制台使用编排模板在Kubernetes集群中创建应用。

在容器服务kubernetes模板编排中，您需要自己定义一个应用运行所需的资源对象，通过标签选择器等机制，将资源对象组合成一个完整的应用。

本例演示如何分步创建nginx应用。首先通过部署deployment创建后端pod资源对象，然后部署service将其绑定到后端pod上，形成一个完整的nginx应用。

前提条件

您必须先部署一个Kubernetes集群。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **部署**进入部署列表页面。
3. 单击页面右上角的**使用模板创建**。
4. 对模板进行相关配置，完成配置后单击**创建**。

表 9-2: 通过编排模板创建应用

配置	说明
集群	选择目标集群。资源对象将部署在该集群内。
命名空间	选择资源对象所属的命名空间，默认为default。除节点、持久化存储卷等底层计算资源以外，大多数资源对象需要作用于命名空间。
资源类型	容器服务提供了多种资源类型的Kubernetes yaml示例模板，让您快速部署资源对象。您可以根据Kubernetes Yaml编排的格式要求自主编写，来描述您想定义的资源类型。 您也可以选择**自定义**并手动编写所需的编排模板。您还可以单击**使用已有模板**在弹出的对话框中选择一个模板。

下面是一个nginx应用的deployment示例编排，通过该编排模板，即可快速创建一个属于nginx应用的deployment 。

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
    name: nginx-deployment
    labels:
      app: nginx
spec:
    replicas: 2
    selector:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
          image: DTR_ip/username/nginx:1.7.9
          ports:
          - containerPort: 80
```

其中，*DTR_ip*是您的DTR的IP地址；*username*是您的用户名。

单击**创建**后，会提示部署状态信息。成功后，单击**Kubernetes控制台**前往控制台查看部署进度。

5. 参见上面的操作，使用模板部署一个Service资源对象。

使用下面的编排模板部署Service。

```
 apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
 kind: Service
 metadata:
   name: my-service1  #specify your service name
   labels:
     app: nginx
 spec:
   selector:
     app: nginx   #change label selector to match your backend pod
   ports:
   - protocol: TCP
     name: http
     port: 30080   #choose an unique port on each node to avoid port
 conflict
     targetPort: 80
   type: LoadBalancer
```

6. 部署成功后，单击**Kubernetes控制台**前往控制台查看部署进度。

9.3 通过Kubernetes Dashboard创建应用

您可以通过Kubernetes Dashboard创建应用。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **集群**进入Kubernetes集群列表页面。
3. 选择所需的集群并单击操作列的**控制台**，进入Kubernetes Dashboard。
4. 在Kubernetes Dashboard中，单击页面右上角的**创建**。
5. 您可以通过以下三种方式创建应用。
   - **使用文本创建**

     编写您自己的YAML或JSON文本，单击**上传**创建应用。
   - **使用文件创建**

     通过导入已有的YAML或JSON配置文件创建应用。单击，在弹出的对话框中选择所需的文件并单击**打开**。单击**上传**创建应用。

- **使用配置创建**

  填写应用详情：通过指定下边的设置创建应用。

  表 9-3: 使用配置创建应用

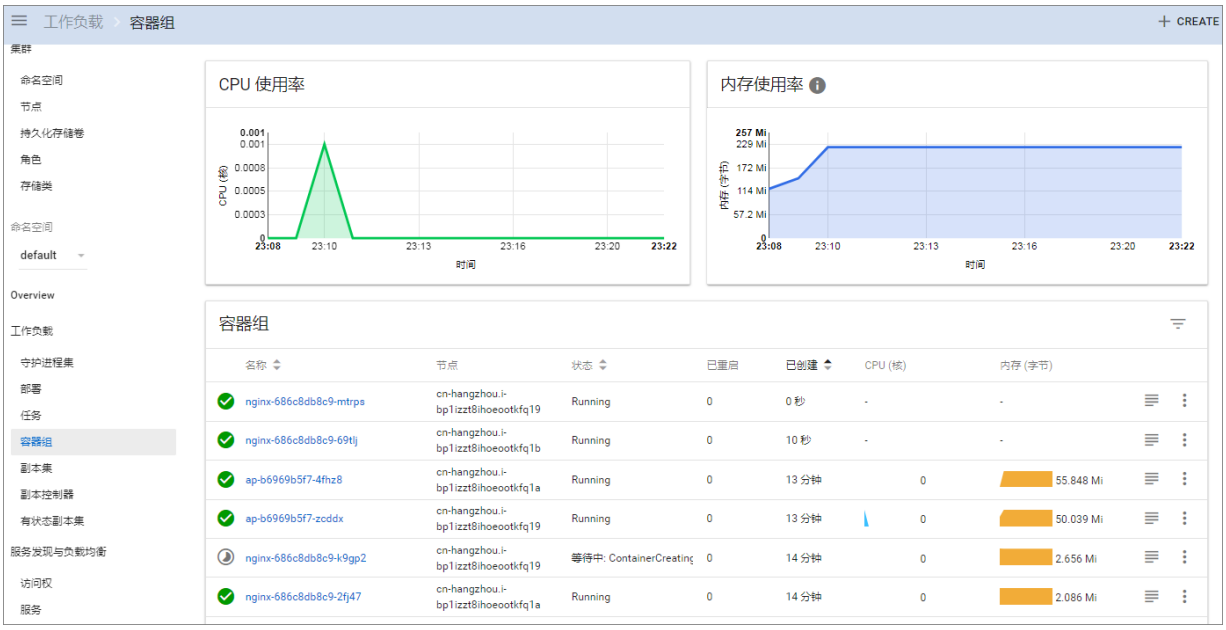
配置	说明
应用名称	所创建应用的名称。本示例中为nginx。
容器镜像	所要使用的镜像的URL。本示例使用的是Docker *Nginx*。
容器组数量	创建的应用的pod个数。
服务	可设置为`External`或`Internal`。`External`创建一个可以从集群外部访问的服务；`Internal`创建一个集群内部可以访问的服务。 此设置将流量负载均衡到三个 Pod。

6. 单击**部署**部署这些容器和服务。

您也可以单击**显示高级选项**展开高级选项进一步配置相关参数。

单击**部署**后，您可以查看应用的服务或查看应用的容器。

单击左侧导航栏中的**容器**，您可以通过左侧的图标查看每个Kubernetes对象的状态。 表示对象仍然处于部署状态。 表示对象已经完成部署。



9.4 通过命令创建应用

您可以通过命令创建应用或者查看应用的容器。

前提条件

在本地使用命令前，您需要先设置通过*kubectl*连接*Kubernetes*集群。

通过命令创建应用

可以通过运行以下语句来运行简单的容器（本示例中为Nginx Web服务器）：

```
root@master # kubectl run -it nginx --image=registry.aliyuncs.com/
spacexnice/netdia:latest
```

此命令将为该容器创建一个服务入口，指定`--type=LoadBalancer`将会为您创建一个阿里云负载均衡路由到该 Nginx 容器。

```
root@master # kubectl expose deployment nginx --port=80 --target-port=
80 --type=LoadBalancer
```

通过命令查看容器

运行如下命令列出所有default命名空间里正在运行的容器。

```
root@master # kubectl get pods
NAME                                    READY      STATUS     RESTARTS
AGE
```

```
nginx-2721357637-dvwq3                    1/1       Running   1
9h
```

9.5 指定节点调度

您可通过为节点设置节点标签，然后通过配置nodeSelector对 pod 调度进行强制约束，将 pod 调度到指定的 Node 节点上。关于 nodeSelector 的详细实现原理，请参见 *nodeselector*。

前提条件

您已经成功部署一个 Kubernetes 集群。

背景信息

出于业务场景需要，如您需要将管控服务部署到 Master 节点；或者将某些服务部署到具有 SSD 盘的机器上。您可以采用这种方式实现指定节点调度。

操作步骤

1. 为节点添加标签
   a. 登录容器服务管理控制台。
   b. 单击左侧导航栏中的**集群** > **节点**进入节点列表页面。
   c. 选择所需的集群，在页面右上角单击**标签管理**。
   d. 在节点列表中，选择所需节点，然后单击 **添加标签**。本例中选择一个 worker 节点。
   e. 在弹出的添加标签对话框中，输入标签的名称和值，然后单击 **确定**。

      您可以在标签管理页面，看到该节点具有 `group:worker`标签。

      您也可以通过命令 `kubectl label nodes <node-name> <label-key>=<label-value>` 为节点添加标签。
2. 将 pod 部署到指定节点
   a. 登录容器服务管理控制台。
   b. 在 Kubernetes 菜单下，单击左侧导航栏中的**应用** > **部署**，进入部署列表页面。
   c. 单击页面右上角的 **使用模板创建**。
   d. 对模板进行相关配置，部署一个 pod，完成配置后，单击 **创建**。

      表 9-4: 模板配置

配置	说明
集群	选择所需的集群。

配置	说明
命名空间	选择资源对象所属的命名空间，本例中是default。

示例模板	本示例选择自定义模板。编排模板如下
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  labels:`
	`  name: hello-pod`
	`  name: hello-pod`
	`spec:`
	`  containers:`
	`  - image: nginx`
	`  imagePullPolicy: IfNotPresent`
	`  name: hello-pod`
	`  ports:`
	`  - containerPort: 8080`
	`  protocol: TCP`
	`  resources: {}`
	`  securityContext:`
	`  capabilities: {}`
	`  privileged: false`
	`  terminationMessagePath: /dev/termination-log`
	`  dnsPolicy: ClusterFirst`
	`  restartPolicy: Always`
	`  nodeSelector:`
	`  group: worker  ##注意与前面配置的节点标签一致`
	`status: {}`

e. 单击 **创建** 后，会提示部署状态信息。 成功后，单击 **Kubernetes控制台**。

f. 您可单击 pod 名称，进入详情页，了解 pod 详情。

   您可看到 pod 的标签、所处的节点 ID 等信息，表明该 pod 已经成功部署到具有 group: worker 标签的指定节点上。

9.6 创建服务

您可以通过容器服务管理控制台创建服务。

背景信息

Kubernetes 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来实现的。

在Kubernetes中，pod虽然拥有独立的IP， 但pod会快速地创建和删除，因此，通过pod直接对外界提供服务不符合高可用的设计准则。通过service这个抽象，Service能够解耦frontend（前端）和backend（后端） 的关联，frontend不用关心backend的具体实现，从而实现松耦合的微服务设计。

更多详细的原理，请参见 *Kubernetes service*。

操作步骤

1. 创建 Deployment。

   a. 登录容器服务管理控制台。

   b. 单击左侧导航栏中的**应用** > **部署**进入部署列表页面。

   c. 单击页面右上角的**使用模板创建**。

   d. 对模板进行相关配置，完成配置后单击**创建**。

表 9-5: 通过模板创建Deployment

配置	说明
集群	选择目标集群。资源对象将部署在该集群内。
命名空间	选择资源对象所属的命名空间，默认为default。除节点、持久化存储卷等底层计算资源以外，大多数资源对象需要作用于命名空间。

示例模板	容器服务提供了多种资源类型的Kubernetes yaml示例模板，让您快速部署资源对象。您可以根据Kubernetes Yaml编排的格式要求自主编写，来描述您想定义的资源类型。
	您也可以选择**自定义**并手动编写所需的编排模板。您还可以单击**使用已有模板**在弹出的对话框中选择一个模板。
	下面是一个nginx应用的deployment示例编排，通过该编排模板，即可快速创建一个属于nginx应用的deployment 。
	`apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1`
	`kind: Deployment`
	`metadata:`
	`  name: nginx-deployment`
	`  labels:`
	`  app: nginx`

配置	说明

	`spec:`
	`  replicas: 2`
	`  selector:`
	`  matchLabels:`
	`  app: nginx`
	`  template:`
	`  metadata:`
	`  labels:`
	`  app: nginx`
	`  spec:`
	`  containers:`
	`  - name: nginx`
	
	`  image: DTR_ip/username/nginx:1.7.9`
	`  ports:`
	`  - containerPort: 80`
	
	其中，*DTR_ip*是您的DTR的IP地址；*username*是您的用户名。

e. 单击**创建**后，会提示部署状态信息。成功后，单击**Kubernetes控制台**前往控制台查看部署进度。

2. 创建服务。

a. 登录容器服务管理控制台。

b. 单击左侧导航栏中的**应用** > **服务**进入服务列表页面。

c. 选择所需的集群和命名空间，单击页面右上角的**创建**。

d. 在弹出的创建服务对话框中，进行配置。

表 9-6: 通过web界面创建服务

名称	输入服务的名称，本例中为nginx-vc。
类型	选择服务类型，即服务访问的方式，包括： • 虚拟集群 IP ：即 ClusterIP，指通过集群的内部 IP 暴露服务，选择该值，服务只能够在集群内部可以访问，这也是默认的 ServiceType。 • 节点端口：即 NodePort，通过每个 Node 上的 IP 和静态端口（NodePort）暴露服务。NodePort 服务会路由到 ClusterIP 服务，这个 ClusterIP 服务会自动创建。通

	过请求`<NodeIP>:<NodePort>`，可以从集群的外部访问一个 NodePort 服务。 • **负载均衡**：即 LoadBalancer，指阿里云提供的负载均衡服务（SLB），可选择公网访问或内网访问。阿里云负载均衡服务可以路由到 NodePort 服务和 ClusterIP 服务。
关联部署	选择服务要绑定的后端对象，本例中是前面创建的 nginx-deployment-basic 。若不进行关联部署，则不会创建相关的 Endpoints 对象，您可自己进行绑定，参见 *services-without-selectors*。
端口映射	添加服务端口和容器端口，容器端口需要与后端的 pod 中暴露的容器端口一致。

e. 单击**创建**，nginx-svc 服务出现在服务列表中。

f. 您可查看服务的基本信息，在浏览器中访问 nginx-svc 的外部端点。

至此，您完成如何创建一个关联到后端的 deployment 的服务，最后成功访问 Nginx 的欢迎页面。

9.7 查看服务

除了运行容器，Kubernetes Dashboard还创建外部Service，用于预配负载均衡器，以便将流量带到集群中的容器。

通过容器服务控制台查看

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **服务**进入服务列表页面。
3. 选择所需的集群和命名空间，您可以查看该集群和命名空间下的服务。

通过Kubernetes Dashboard查看

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群**进入Kubernetes集群列表页面。
3. 选择所需的集群并单击右侧的**控制台**，进入Kubernetes Dashboard。
4. 在左侧导航栏中，单击**服务**查看所有服务。

您可以看到分配给服务的外部入口（IP地址）。您可以单击该IP地址访问负载均衡器后面运行的Nginx容器。

9.8 更新服务

您可以通过容器服务控制台或者Kubernetes Dashboard变更服务的配置。

通过容器服务控制台

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **服务**进入服务列表页面。
3. 选择所需的集群和命名空间，选择需要更新的服务并单击操作列的**更新**。
4. 修改服务的配置并单击**确定**。

通过Kubernetes Dashboard

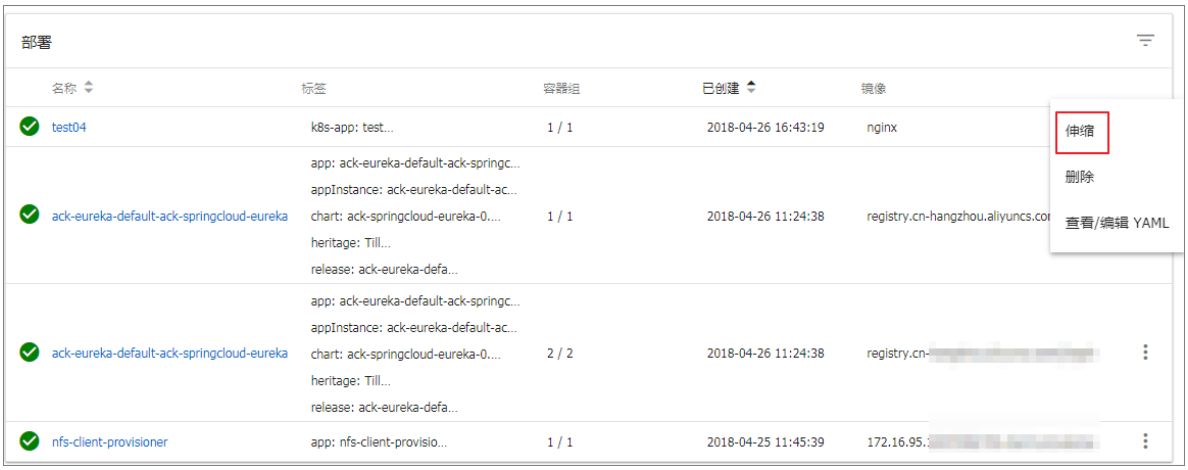
1. 登录容器服务管理控制台。
2. 单击**集群** > **集群**进入Kubernetes集群列表页面。
3. 选择所需的集群并单击操作列的**控制台**，进入Kubernetes Dashboard。
4. 在左侧导航栏中，单击**服务**查看所有服务。
5. 选择所需的服务，单击右侧的操作图标并单击**查看/编辑YAML**。
6. 更新服务的配置并单击**更新**。

9.9 服务伸缩

应用创建后，您可以根据自己的需求来进行服务扩容或缩容。

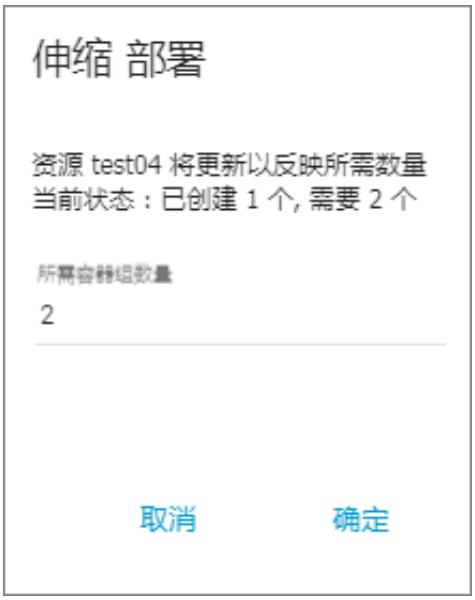
操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **集群**进入Kubernetes集群列表页面。
3. 选择所需的集群并单击操作列的**控制台**，进入Kubernetes Dashboard。
4. 在 Kubernetes Dashboard 中，单击左侧导航栏中的**部署**，查看部署的 deployment。
5. 选择所需的 deployment，单击右侧的操作图标并单击**伸缩**。



6. 在弹出的对话框中，将**所需容器组数**修改为 2，并单击**确定**。

   此操作会扩容一个 Pod，将副本数升到 2。



您可以通过左侧的图标查看每个 Kubernetes 对象的状态。 表示对象仍然处于部署状态。 表示对象已经完成部署。

应用部署完成后，您可以单击某个部署的名称查看正在运行的 Web 服务的详细信息。您可以查看部署中的副本集以及这些副本集所使用的CPU和Memory资源的相关信息。您也可以单击**查看日志**查看容器的日志。

9.10 删除服务

您可以通过容器服务控制台删除不再使用的服务。

操作步骤

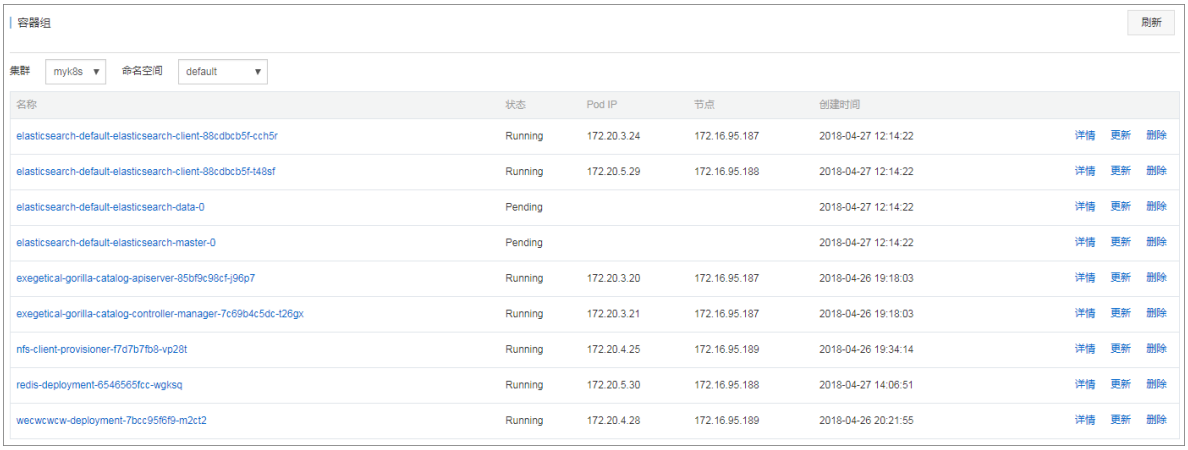
1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **服务**进入服务列表页面。
3. 选择所需的集群和命名空间，选择需要更新的服务并单击操作列的**删除**。
4. 在弹出的对话框中单击**确认**。

9.11 查看容器

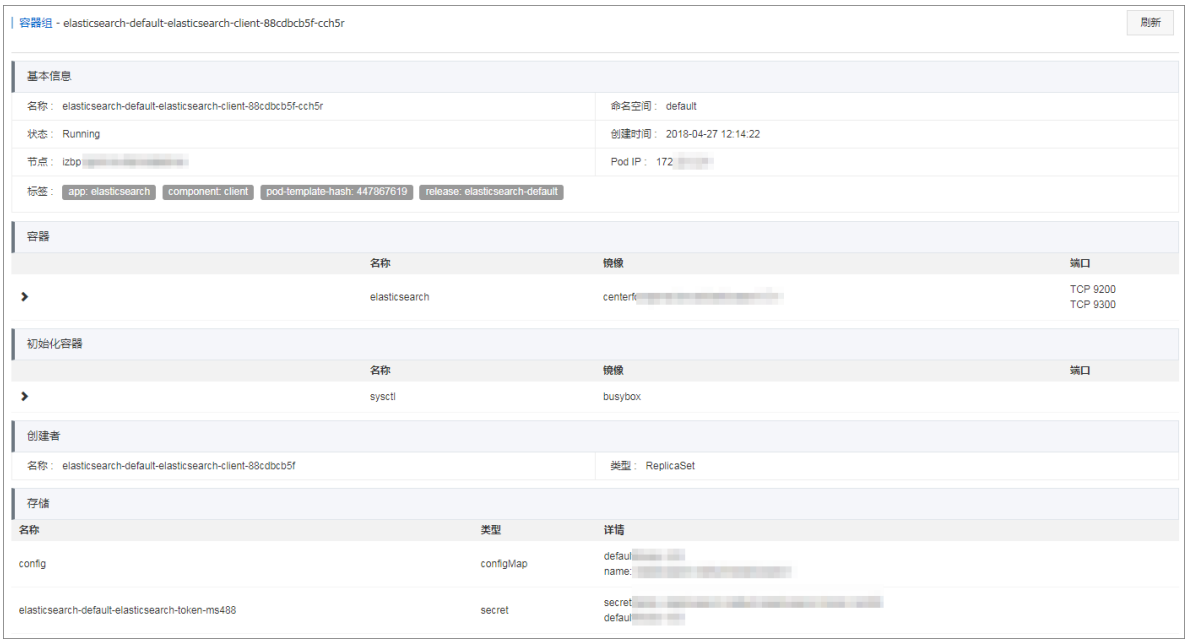
您可以通过容器服务控制台或者Kubernetes Dashboard查看容器。

通过容器服务控制台

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **容器组**进入容器组页面。
3. 选择所需的集群和命名空间，即可查看该集群和命名空间下的容器。



您可以选择所需的容器，单击操作列的**详情**查看容器的详细信息。



您可以选择所需的容器，单击右侧的**删除**删除该容器。

通过Kubernetes Dashboard

1. 登录容器服务管理控制台。
2. 在**Kubernetes**菜单下，单击左侧导航栏中的**集群** > **集群**进入集群列表页面。
3. 选择所需的集群并单击右侧的**控制台**，进入Kubernetes Dashboard。
4. 单击左侧导航栏中的**容器组**，查看集群中的容器。

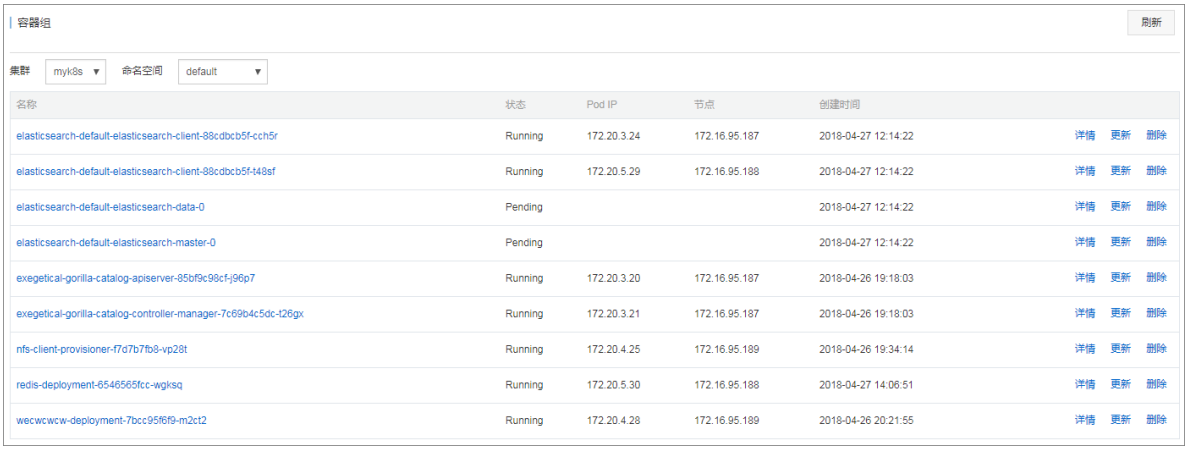
   或者，您可以单击左侧导航栏中的**服务**，选择所需的服务并单击服务的名称，查看该服务下的容器。
5. 您可以通过左侧的图标查看每个Kubernetes对象的状态。 表示对象仍然处于部署状态。 表示对象已经完成部署。
6. 选择所需的容器，单击容器的名称，您可以查看容器的详细信息以及容器使用的CPU和Memory。
7. 选择所需的容器，单击右侧的**查看日志**查看容器的日志信息。
8. 您可以单击右侧的**删除**删除该容器。

9.12 变更容器配置

您可以通过容器服务管理控制台或者Kubernetes Dashboard变更容器的配置。

通过容器服务控制台

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **容器组**进入容器组页面。
3. 选择所需的集群和命名空间，即可查看该集群和命名空间下的容器。



4. 您可以选择所需的容器，单击操作列的**更新**。
5. 在弹出的对话框中，更新容器的配置并单击**更新**。

通过Kubernetes Dashboard

1. 登录容器服务管理控制台。
2. 在**Kubernetes**菜单下，单击左侧导航栏中的**集群** > **集群**进入集群列表页面。
3. 选择所需的集群并单击右侧的**控制台**，进入Kubernetes Dashboard。
4. 单击左侧导航栏中的**容器组**，查看集群中的容器。

   或者，您可以单击左侧导航栏中的**服务**，选择所需的服务并单击服务的名称，查看该服务下的容器。
5. 选择所需的容器，单击右侧的操作图标并单击**查看/编辑YAML**。
6. 更新服务的配置并单击**更新**。

10 配置项

阿里云容器服务Kubernetes集群提供配置项功能，来存储配置文件。

10.1 创建配置项

在容器服务管理控制台上，您可以通过配置项菜单或使用模板来创建配置项。

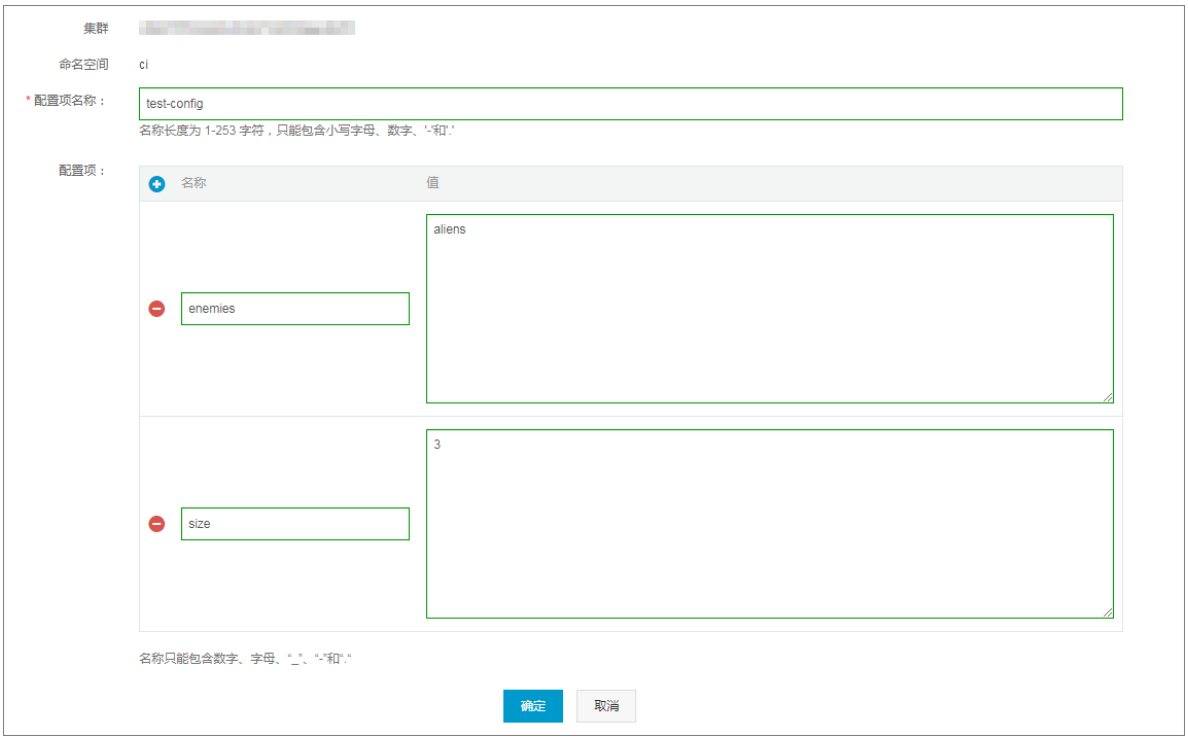
通过配置项菜单创建

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **配置项**进入配置项列表页面。
3. 选择需要创建配置项的集群和命名空间，然后单击**创建**。
4. 填写配置文件的信息并单击**确定**。

表 10-1: 通过Web界面创建配置项

配置	说明
集群	列出已选择的集群ID
命名空间	列出已选择的命名空间。配置项（ConfigMap）是kubernetes资源对象，需要作用于命名空间。
配置名称	指定配置项的文件名，名称可以包含小写字母、数字、连字符（-）或者点号（.），名称不能为空。其他资源对象需要引用配置文件名来获取配置信息。
配置项	填写**变量名称**和**变量值**后，需要单击右侧的**添加**。您也可以单击**编辑配置文件**在弹出的对话框里编写配置项并单击**确定**。

本示例中设置了enemies和lives变量，分别用于传递aliens和3这两个参数。



5. 配置完成后，单击**确定**。您可以在配置文件列表中看到 test-config 配置文件。

通过模板创建

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **部署**进入应用列表页面。
3. 单击页面右上角的**使用模板创建**。
4. 在使用模版部署的页面，设置配置文件的信息并单击**创建**。

表 10-2: 通过编排模板创建配置项

配置	说明
集群	选择需要创建配置项的集群。
命名空间	选择该配置项所属的命名空间。配置项（ConfigMap）是kubernetes资源对象，需要作用于命名空间。
编排模板	您可以选择自定义，根据Kubernetes yaml语法规则编写ConfigMap文件，或者选择示例模板`Resource-ConfigMap`。该示例模板的ConfigMap名称为 aliyun-config，包含两个变量文件 *`game.properties`*和*`ui.properties`*，您可以在此基础上进行修改。

部署成功后，您可以在配置文件列表下看到*aliyun-config*配置文件。

10.2 在 Pod 中使用配置项

您可以在Pod中使用配置项。

背景信息

在Pod中使用配置项有多种使用场景，主要包括：

- 使用配置项定义Pod环境变量
- 通过配置项设置命令行参数
- 在数据卷中使用配置项

更多关于配置项的信息，可以参见 *Configure a Pod to Use a ConfigMap*。

使用限制

您在Pod里使用配置项时，需要两者处于同一集群和命名空间中。

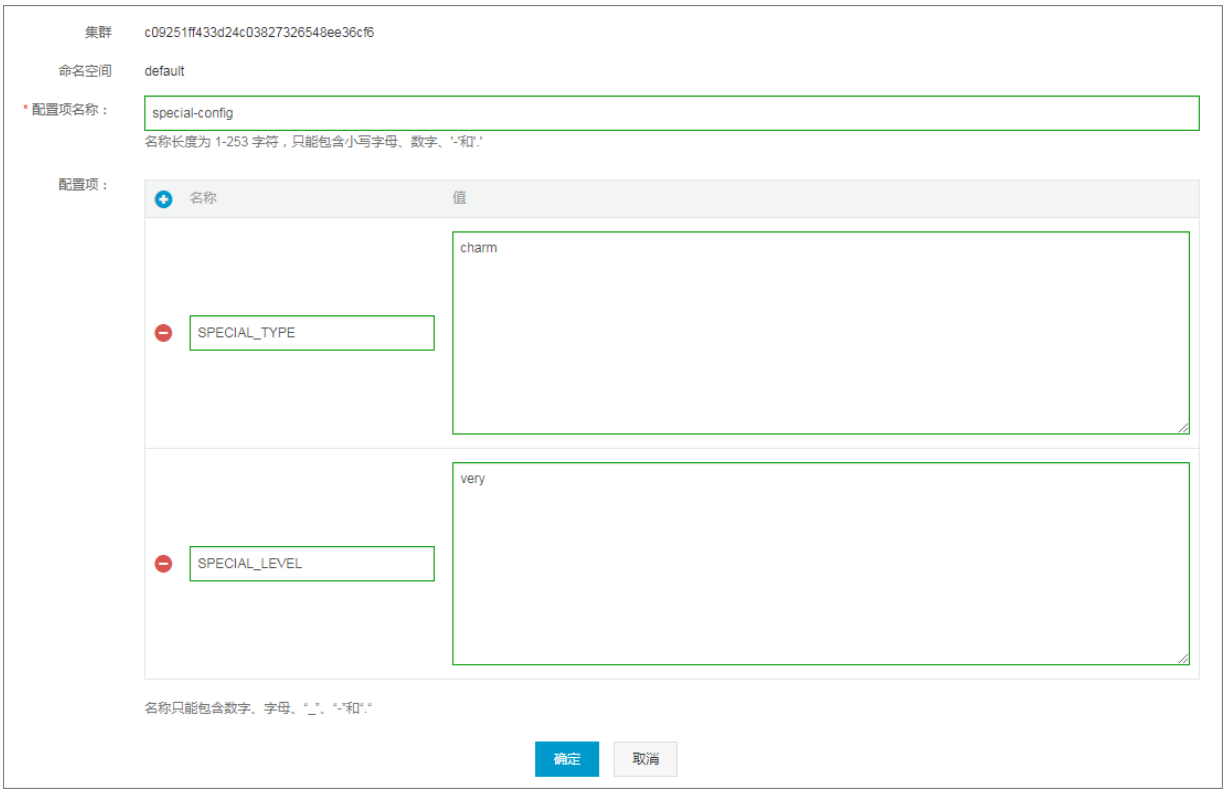
创建配置项

本示例创建配置项 special_config，包含`SPECIAL_LEVEL: very`和`SPECIAL_TYPE: charm`两个键值对。

您可以使用如下yaml示例模板创建配置项。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  SPECIAL_LEVEL: very
  SPECIAL_TYPE: charm
```

您也可以登录容服务管理控制台，在左侧导航栏中单击**应用** > **配置项**，然后单击**创建**进行创建。



使用配置项定义 pod 环境变量

使用配置项的数据定义 pod 环境变量

您可以登录容器服务管理控制台，单击左侧导航栏中的**应用** > **部署**。然后单击**使用模板创建**，编辑 pod 类型的 kubernetes yaml 模板，然后进行部署；或者在 Kubernetes Dashboard 中**上传YAML或JSON文件**。

您可以使用如下示例模板创建一个pod， 在其中定义环境变量，使用**`valueFrom`**引用SPECIAL_LEVEL的 value值，从而定义 pod 的环境变量。

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-1
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:                              ##使用valueFrom来指定
env引用配置项的value值
            configMapKeyRef:
              name: special-config                ##引用的配置文件名称
              key: SPECIAL_LEVEL                  ##引用的配置项key
```

```
  restartPolicy: Never
```

同理，如果您需要将多个配置项的 value 值定义为 pod 的环境变量值，您只需要在 pod 中添加多个 env 参数即可。

将配置项的所有 key/values 配置为 pod 的环境变量

如果您想在一个 pod 中将配置项的所有 key/values 键值对配置为 pod 的环境变量，可以使用 envFrom 参数，配置项中的 key 会成为 Pod 中的环境变量名称。

以下是一个 pod 的示例编排。

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-2
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:                ##引用 sepcial-config 配置文件的所有 key/
values 键值对
      - configMapRef:
        name: special-config
  restartPolicy: Never
```

通过配置项设置命令行参数

您可以使用配置项设置容器中的命令或者参数值，使用环境变量替换语法*$(VAR_NAME)*来进行。如下面的编排示例所示。

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-3
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY) $(
SPECIAL_TYPE_KEY)" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_TYPE
```

```
  restartPolicy: Never
```

运行这个 pod 后，会输出如下结果。

```
very charm
```

在数据卷中使用配置项

配置项也可以在数据卷里面被使用。如下面的编排示例所示，在 volumes 下指定配置项名称，会将 key/values 的数据存储到 mountPath 路径下（本例中是 /etc/config）。最终生成以 key 为文件名，values 为文件内容的配置文件。

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-4
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]   ##列出该目录下的
文件名
      volumeMounts:
        - name: config-volume
          mountPath: /etc/config
   volumes:
     - name: config-volume
       configMap:
         name: special-config
  restartPolicy: Never
```

运行 pod 后，会输出配置项的 key。

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

10.3 修改配置项

您可以修改配置项的配置。

注意事项

修改配置文件会影响使用该配置文件的应用。

通过配置项菜单修改

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **配置项**进入配置项列表页面。
3. 选择所需的集群，选择所需的配置项并单击操作列的**修改**。

4. 在弹出的确认对话框中，单击**确定**。
5. 选择需要修改的配置项并单击操作列的**编辑**，修改配置后单击**保存**。
6. 完成配置项修改后，单击**确定**。

通过 Kubernetes Dashboard 进行修改

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **集群**，选择目标集群，单击操作列的 **控制台**。
3. 在 Kubernetes Dashboard 页面，单击左侧导航栏中的 **配置与存储** > **配置字典**，选择目标字典，单击右侧的**操作** > **查看/编辑 YAML**。
4. 弹出修改配置集对话框，对配置变量进行修改后，单击 **更新**。

10.4 删除配置项

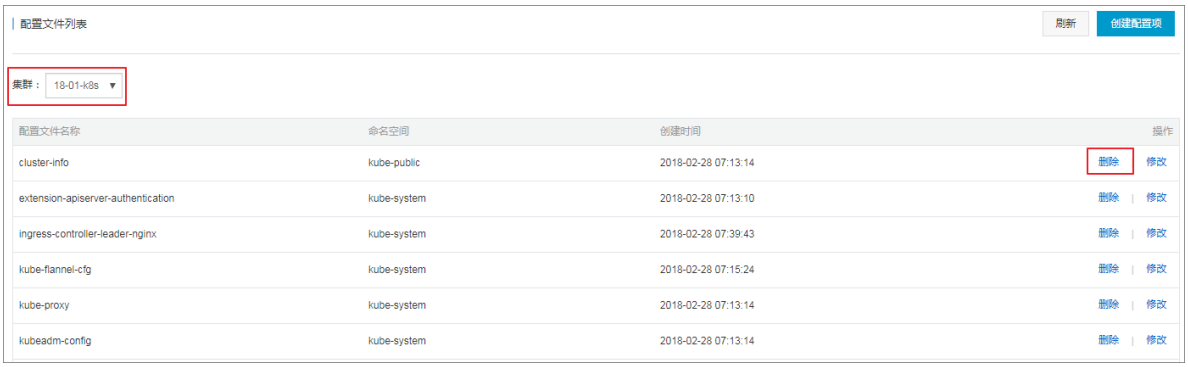
您可以删除不再使用的配置项。

注意事项

删除配置文件会影响使用该配置文件的应用。

通过配置项菜单删除

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **配置项**进入配置项列表页面。
3. 选择所需的集群，选择所需的配置项并单击操作列的**删除**。



通过 Kubernetes Dashboard 删除

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**集群** > **集群**，选择目标集群，单击操作列的 **控制台**。
3. 在 Kubernetes Dashboard 页面，单击左侧导航栏中的 **配置与存储** > **配置字典**，选择目标配置字典，单击右侧的**操作** > **删除**。
4. 在弹出的对话框中，单击 **删除**。

11 保密字典

您可是使用保密字典（secret）保存一些敏感数据。

11.1 创建保密字典

您可通过控制台创建保密字典。

前提条件

您已成功创建一个 Kubernetes 集群。

背景信息

若您需要在 Kubernetes 集群中使用一些敏感的配置，比如密码、证书等信息时，建议使用保密字典（secret），即保密字典。

保密字典有多种类型，例如：

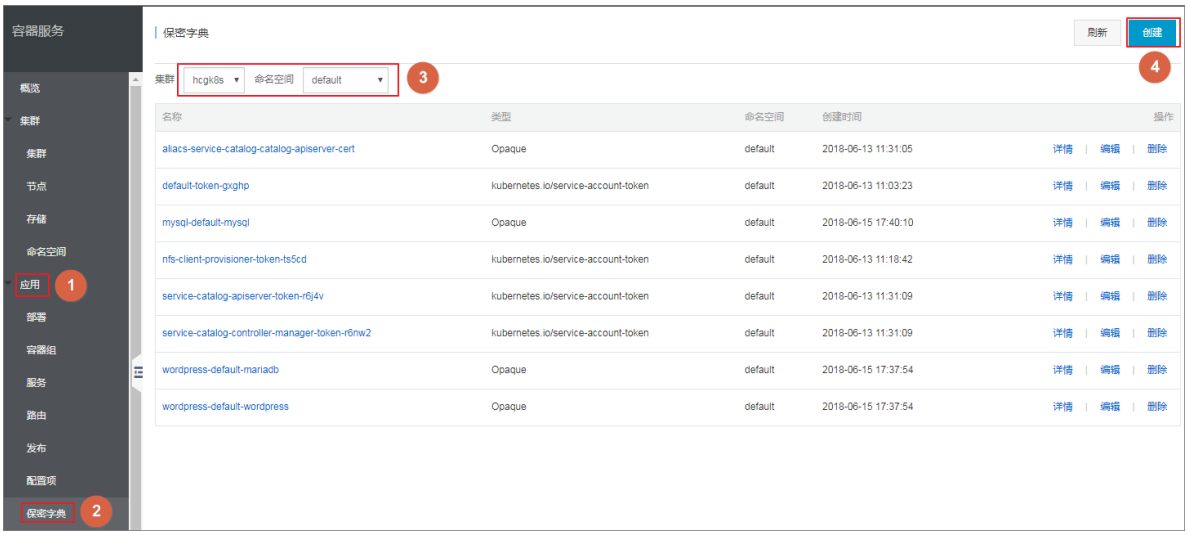
- Service Account：用来访问 Kubernetes API，由 Kubernetes 自动创建，并且会自动挂载到 Pod 的`/run/secrets/kubernetes.io/serviceaccount`目录中。
- Opaque：base64 编码格式的 Secret，用来存储密码、证书等敏感信息。

默认情况下，通过 Web 界面只能创建 Opaque 类型的保密字典。Opaque 类型的数据是一个 map 类型，要求value 是 base64 编码格式。阿里云容器服务提供一键创建保密字典的功能，自动将明文数据编码为 base64 格式。

您也可通过命令行手动创建保密字典，请参见 *kubernetes secret* 了解更多信息。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **保密字典**，进入保密字典页面。
3. 选择所需的集群和命名空间，单击右上角的**创建**。



4. 配置新的保密字典，单击**确定**。

说明：

若您输入保密字典的明文数据，请注意勾选**对数据值进行base64编码**

表 11-1: 配置保密字典

配置	说明
名称	输入保密字典的名称。名称长度为 1 -253 字符，只能包含小写字母、数字、’ - ‘和’ .’ 。
数据	配置保密字典的数据。单击**添加**，在弹出的对话框中，输入保密字典名称和值，即键值对。本例中创建的保密字典包含两个value，`username:admin`和 `passwrod : 1f2d1e2e67df`。

5. 默认返回保密字典页面，您可看到新建的保密字典出现在列表中。

11.2 查看密钥

您可通过阿里云容器服务 Web 界面查看已创建的密钥。

前提条件

- 您已成功创建一个 Kubernetes 集群。
- 您已创建一个密钥，参见创建保密字典。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **保密字典**，进入保密字典页面。
3. 选择所需的集群和命名空间，选择所需的密钥，并单击操作列的**详情**。

4. 进入密钥的详情页面，您可查看该密钥的基本信息，以及密钥包含的数据信息。

   单击详细信息中数据名称右侧的图标，可查看数据的明文。

11.3 编辑密钥

您可通过阿里云容器服务 Web 界面直接编辑已有密钥。

前提条件

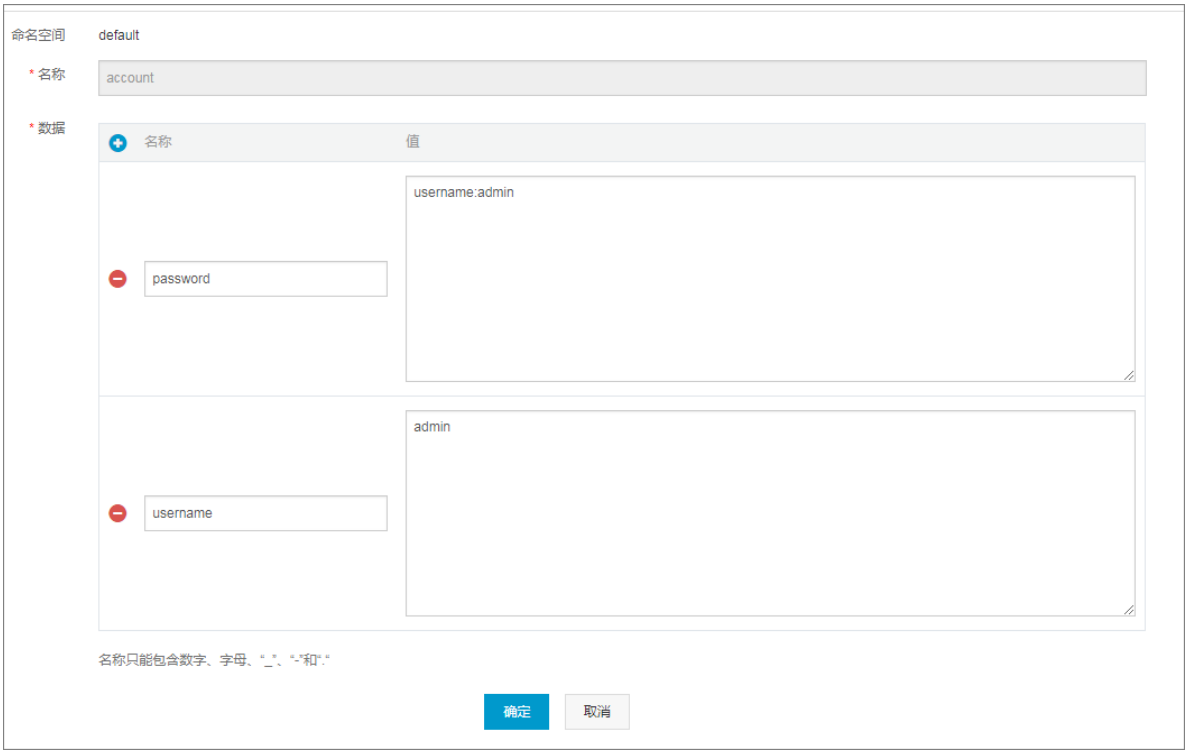
- 您已成功创建一个 Kubernetes 集群。
- 您已创建一个密钥，参见创建保密字典。

操作步骤

1. 登录容器服务管理控制台。

2. 单击左侧导航栏中的**应用** > **保密字典**，进入保密字典页面。
3. 选择所需的集群和命名空间，选择所需的密钥，并单击操作列的**编辑**。

4. 在编辑密钥的页面，可对密钥的数据进行编辑。



5. 单击**确定**。

11.4 删除密钥

您可通过阿里云容器服务 Web 界面删除已有密钥。

前提条件

- 您已成功创建一个 Kubernetes 集群。
- 您已创建一个密钥，参见创建保密字典。

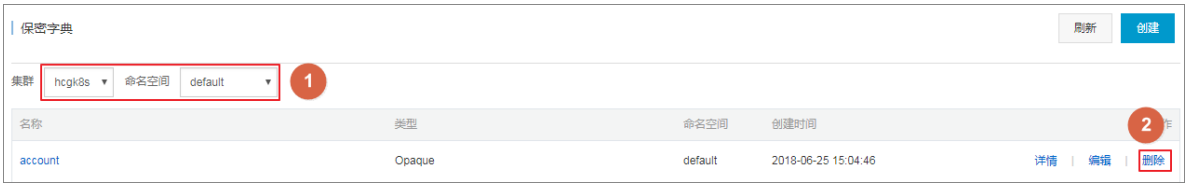
背景信息

说明：

集群创建过程中生成的密钥请勿删除。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**应用** > **保密字典**，进入保密字典页面。
3. 选择所需的集群和命名空间，选择所需的密钥，并单击操作列的**删除**。



4. 在弹出的提示框中，单击**确定**，完成删除。

12 负载均衡

Kubernetes 集群提供了多种访问容器应用的方式，支持通过阿里云的负载均衡服务（Server Load Balancer）或者 Ingress 方式来访问内部服务的和实现负载均衡。

12.1 通过负载均衡（Server Load Balancer）访问服务

您可以使用阿里云负载均衡来访问服务。

通过命令行操作

1. 通过命令行工具创建一个 Nginx 应用。

```
root@master # kubectl run nginx --image=registry.aliyuncs.com/acs/
netdia:latest
       root@master # kubectl get po
       NAME                                    READY     STATUS
RESTARTS   AGE
       nginx-2721357637-dvwq3                  1/1       Running   1
         6s
```

2. 为 Nginx 应用创建阿里云负载均衡服务，指定 `type=LoadBalancer` 来向外网用户暴露 Nginx 服务。

```
root@master # kubectl expose deployment nginx --port=80 --target-
port=80 --type=LoadBalancer
       root@master # kubectl get svc
       NAME                   CLUSTER-IP      EXTERNAL-IP      PORT(S
)                         AGE
       nginx                  172.19.10.209   101.37.192.20   80:
31891/TCP                   4s
```

3. 在浏览器中访问 `http://101.37.192.20`，来访问您的 Nginx 服务。

通过 Kubernetes Dashboard 操作

1. 将下面的 yml code 保存到 `nginx-svc.yml`文件中。

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: http-svc
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
```

```
  type: LoadBalancer
```

2. 登录容器服务管理控制台，单击目标集群右侧的**控制台**，进入 Kubernetes Dashboard 页面。
3. 单击**创建**，开始创建应用。
4. 单击**使用文件创建**。选择刚才保存的*nginx-svc.yml* 文件
5. 单击**上传**。

   这样会创建一个阿里云负载均衡实例指向创建的 Nginx 应用，服务的名称为 `http-svc`。
6. 在 Kubernetes Dashboard 上定位到 default 命名空间，选择**服务**。

   可以看到刚刚创建的 `http-svc` 的 Nginx 服务和机器的负载均衡地址 `http://114.55.79.24:80`。
7. 将该地址拷贝到浏览器中即可访问该服务。

更多信息

阿里云负载均衡还支持丰富的配置参数，包含健康检查、收费类型、负载均衡类型等参数。详细信息参见配置参数表 。

注释

阿里云可以通过注释的形式支持丰富的负载均衡功能。

使用已有的内网 SLB

需要指定两个annotation。注意修改成您自己的 Loadbalancer-id。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type:
intranet
    service.beta.kubernetes.io/alicloud-loadbalancer-id: your-
loadbalancer-id
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  sessionAffinity: None
```

```
  type: LoadBalancer
```

然后保存为 slb.svc 后，执行 `kubectl apply -f slb.svc`。

创建 HTTPS 类型的 Loadbalancer

先在阿里云控制台上创建一个证书并记录 cert-id，然后使用如下 annotation 创建一个 HTTPS 类型的 SLB。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-cert-id: your-
cert-id
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
https:443"
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

说明：

注释的内容是区分大小写的。

注释	描述	默认值
service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port	多个值之间由逗号分隔，比如：https:443,http:80	无
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	取值可以是 internet 或者 intranet	internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	负载均衡的网络类型，取值可以是 classic 或者 vpc	classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	取值可以是 paybytraffic 或者 paybybandwidth	paybybandwidth

注释	描述	默认值
service.beta.kubernetes.io/alicloud-loadbalancer-id	负载均衡实例的 ID。通过 loadbalancer-id 指定您已有的 SLB，已有 listener 会被覆盖，删除 service 时该 SLB 不会被删除。	无
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	通过 label 指定 SLB 后端挂哪些节点。	无
service.beta.kubernetes.io/alicloud-loadbalancer-region	负载均衡所在的地域	无
service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth	负载均衡的带宽	50
service.beta.kubernetes.io/alicloud-loadbalancer-cert-id	阿里云上的认证 ID。您需要先上传证书	“”
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag	取值是 on 或者 off	默认为 off。TCP 不需要改参数。因为 TCP 默认打开健康检查，用户不可设置。
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type	参见 *HealthCheck*	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	参见 *HealthCheck*	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	参见 *HealthCheck*	
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	参见 *HealthCheck*	
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	参见 *HealthCheck*	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	参见 *HealthCheck*	

注释	描述	默认值
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	参见 *HealthCheck*	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout	参见 *HealthCheck*	

12.2 Ingress监控配置

您可以通过开启 Ingress 默认 VTS 模块来查看 Ingress 监控数据。

通过命令行操作

1. 修改 Ingress ConfigMap 配置，增加配置项 `enable-vts-status: "true"`。

```
root@master # kubectl edit configmap nginx-configuration -n kube-
system
       configmap "nginx-configuration" edited
```

修改后 Ingress ConfigMap 内容如下。

```
apiVersion: v1
data:
  enable-vts-status: "true" # 开启VTS模块
  proxy-body-size: 20m
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"proxy-body-size":"20m"},"kind":"
ConfigMap","metadata":{"annotations":{},"labels":{"app":"ingress-
nginx"},"name":"nginx-configuration","namespace":"kube-system"}}
  creationTimestamp: 2018-03-20T07:10:18Z
  labels:
    app: ingress-nginx
  name: nginx-configuration
  namespace: kube-system
  selfLink: /api/v1/namespaces/kube-system/configmaps/nginx-
configuration
```

2. 验证 Ingress Nginx 正常开启 VTS 模块。

```
root@master # kubectl get pods --selector=app=ingress-nginx -n kube-
system
NAME                                        READY     STATUS
RESTARTS   AGE
nginx-ingress-controller-79877595c8-78gq8   1/1       Running   0
       1h
root@master # kubectl exec -it nginx-ingress-controller-79877595c8-
78gq8 -n kube-system -- cat /etc/nginx/nginx.conf | grep vhost_traf
fic_status_display
         vhost_traffic_status_display;
```

```
            vhost_traffic_status_display_format html;
```

3. 本地访问 Ingress Nginx 监控控制台。

> **说明**：
> 鉴于安全考虑，默认 VTS Port 并不对外开放，这里通过 port-forward 方式来访问。

```
root@master # kubectl port-forward nginx-ingress-controller-
79877595c8-78gq8 -n kube-system 18080
       Forwarding from 127.0.0.1:18080 -> 18080
       Handling connection for 18080
```

4. 通过 `http://localhost:18080/nginx_status` 来访问 VTS 监控控制台。

Nginx Vhost Traffic Status

Server main

Host	Version	Uptime	Connections				Requests				Shared memory			
			active	reading	writing	waiting	accepted	handled	Total	Req/s	name	maxSize	usedSize	usedNode
nginx-ingress-controller-79877595c8-78gq8	1.13.7	32m 41s	7	0	1	6	93566	93566	1428	1	vhost_traffic_status	10.0 MiB	2.4 KiB	1

Server zones

Zone	Requests			Responses						Traffic				Cache								
	Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s	Miss	Bypass	Expired	Stale	Updating	Revalidated	Hit	Scarce	Total
_	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0
*	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0

Upstreams

upstream-default-backend

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests			Responses						Traffic			
						Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
172.16.3.6:8080	up	0ms	1	0	0	0	0	0ms	0	0	0	0	0	0	0 B	0 B	0 B	0 B

update interval: 1 sec

JSON | GITHUB

通过 Kubernetes Dashboard 操作

1. 登录容器服务管理控制台。
2. 选择所需的集群并单击集群右侧的**控制台**，进入 Kubernetes Dashboard 页面。
3. 编辑 kube-system 命名空间下的配置字典 nginx-configuration，增加配置项 `enable-vts-status: "true"`。

   保存后 Ingress ConfigMap 内容如下。

```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "nginx-configuration",
    "namespace": "kube-system",
    "selfLink": "/api/v1/namespaces/kube-system/configmaps/nginx-
configuration",
    "creationTimestamp": "2018-03-20T07:10:18Z",
    "labels": {
      "app": "ingress-nginx"
```

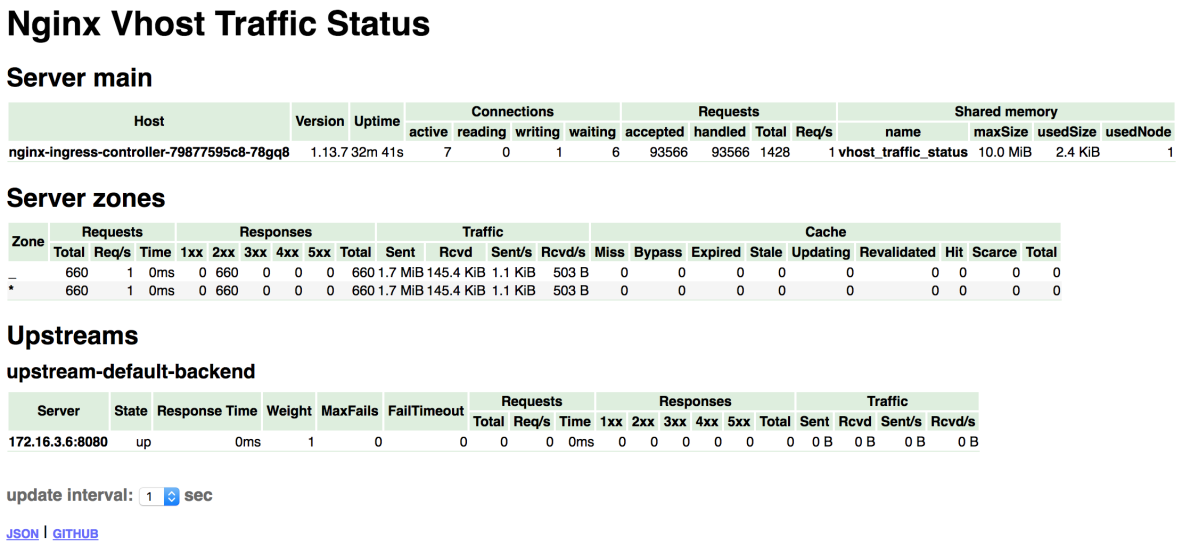
```
    },
    "annotations": {
      "kubectl.kubernetes.io/last-applied-configuration": "{\"
apiVersion\":\"v1\",\"data\":{\"proxy-body-size\":\"20m\"},\"kind\":
\"ConfigMap\",\"metadata\":{\"annotations\":{},\"labels\":{\"app\":
\"ingress-nginx\"},\"name\":\"nginx-configuration\",\"namespace\":\"
kube-system\"}}\n"
    }
  },
  "data": {
    "proxy-body-size": "20m",
    "enable-vts-status": "true"
  }
}
```

4. 本地访问 Ingress Nginx 监控控制台。

> **说明**：
>
> 鉴于安全考虑，默认 VTS Port 并不对外开放，这里通过 port-forward 方式来访问。

```
root@master # kubectl port-forward nginx-ingress-controller-
79877595c8-78gq8 -n kube-system 18080
       Forwarding from 127.0.0.1:18080 -> 18080
       Handling connection for 18080
```

5. 通过 `http://localhost:18080/nginx_status` 来访问 VTS 监控控制台。



Nginx Vhost Traffic Status

Server main

Host	Version	Uptime	Connections				Requests				Shared memory			
			active	reading	writing	waiting	accepted	handled	Total	Req/s	name	maxSize	usedSize	usedNode
nginx-ingress-controller-79877595c8-78gq8	1.13.7	32m 41s	7	0	1	6	93566	93566	1428	1	vhost_traffic_status	10.0 MiB	2.4 KiB	1

Server zones

Zone	Requests			Responses						Traffic				Cache								
	Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s	Miss	Bypass	Expired	Stale	Updating	Revalidated	Hit	Scarce	Total
_	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0
*	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0

Upstreams

upstream-default-backend

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests			Responses						Traffic			
						Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
172.16.3.6:8080	up	0ms	1	0	0	0	0	0ms	0	0	0	0	0	0	0 B	0 B	0 B	0 B

update interval: 1 sec

JSON | GITHUB

12.3 Ingress 支持

在 Kubernetes 集群中，Ingress是授权入站连接到达集群服务的规则集合，为您提供七层负载均衡能力。您可以给 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。

前置条件

为了测试复杂路由服务，本例中创建一个 nginx 的示例应用，您需要事先创建 nginx 的 deployment，然后创建多个 Service，用来观察路由的效果。实际测试请替换成自己的服务。

```
root@master # kubectl run nginx --image=registry.cn-hangzhou.aliyuncs.
com/acs/netdia:latest
root@master # kubectl expose deploy nginx --name=http-svc --port=80 --
target-port=80
root@master # kubectl expose deploy nginx --name=http-svc1 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc2 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc3 --port=80
 --target-port=80
```

简单的路由服务

通过以下命令创建一个简单的 Ingress，所有对 */svc* 路径的访问都会被路由到名为 http-svc 的服务。`nginx.ingress.kubernetes.io/rewrite-target: /`会将*/svc*路径重定向到后端服务能够识别的/路径上面。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
simple          *             101.37.192.211   80        11s
```

现在访问 `http://101.37.192.211/svc` 即可访问到 Nginx 服务。

基于域名的简单扇出路由

如果您有多个域名对外提供不同的服务，您可以生成如下的配置达到一个简单的基于域名的扇出效果。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout
```

```
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
  - host: foo.example.com
    http:
      paths:
      - path: /film
        backend:
          serviceName: http-svc3
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS          ADDRESS           PORTS     AGE
simple-fanout   *              101.37.192.211    80        11s
```

这时您可以通过 `http://foo.bar.com/foo` 访问到 `http-svc1` 服务；通过 `http://foo.bar.com/bar` 访问到 `http-svc2` 服务；通过 `http://foo.example.com/film` 访问到 `http-svc3` 服务。

说明：

- 如果是生产环境，您需要将您的这个域名指向上面返回的 ADDRESS `101.37.192.211`。
- 如果是测试环境测试，您可以修改 `hosts` 文件添加一条域名映射规则。

```
101.37.192.211 foo.bar.com
       101.37.192.211 foo.example.com
```

简单路由默认域名

如果您没有域名地址也没有关系，容器服务为 Ingress 服务绑定了一个默认域名，您可以通过这个域名来访问服务。域名的格式如下：`*.[cluster-id].[region-id].alicontainer.com`。您可以直接在控制台集群基本信息页获取到该地址。

您可以通过下面的配置借助该默认域名暴露两个服务。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: shared-dns
spec:
```

```
  rules:
  - host: foo.[cluster-id].[region-id].alicontainer.com  ##替换为您集群
默认的服务访问域名
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc1
          servicePort: 80
  - host: bar.[cluster-id].[region-id].alicontainer.com  ##替换为您集群
默认的服务访问域名
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc2
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
shared-dns   foo.[cluster-id].[region-id].alicontainer.com,bar.[
cluster-id].[region-id].alicontainer.com            47.95.160.171
80        40m
```

这时您可以通过 `http://foo.[cluster-id].[region-id].alicontainer.com/` 访问到 `http-svc1` 服务；通过 `http://bar.[cluster-id].[region-id].alicontainer.com` 访问到 `http-svc2` 服务。

配置安全的路由服务

支持多证书管理，为您的服务提供安全防护。

1. 准备您的服务证书。

   如果没有证书，可以通过下面的方法生成测试证书。

   > **说明**：
   > 域名与您的 Ingress 配置要一致。

   ```
   root@master # openssl req -x509 -nodes -days 365 -newkey rsa:2048 -
   keyout tls.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
   ```

   上面命令会生成一个证书文件 *tls.crt*、一个私钥文件*tls.key*。

然后用该证书和私钥创建一个名为*foo.bar* 的 Kubernetes Secret。创建 Ingress 时需要引用这个 Secret。

```
root@master # kubectl create secret tls foo.bar --key tls.key --cert
 tls.crt
```

2. 创建一个安全的 Ingress 服务。

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: tls-fanout
  annotations:
     nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: foo.bar
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
EOF
root@master # kubectl get ing
NAME            HOSTS         ADDRESS           PORTS     AGE
tls-fanout      *             101.37.192.211    80        11s
```

3. 按照 **基于域名的简单扇出路由** 中的注意事项，配置 `hosts` 文件或者设置域名来访问该 tls 服务。

   您可以通过 `http://foo.bar.com/foo` 访问到 `http-svc1` 服务；通过 `http://foo.bar.com/bar` 访问到 `http-svc2` 服务。

   您也可以通过 HTTP 的方式访问该 HTTPS 的服务。Ingress 默认对配置了 HTTPS 的 HTTP 访问重定向到 HTTPS 上面。所以访问 `http://foo.bar.com/foo` 会被自动重定向到 `https://foo.bar.com/foo`。

通过 Kubernetes Dashboard 部署 Ingress

1. 将下面的 yml code 保存到 `nginx-ingress.yml` 文件中。

```
apiVersion: extensions/v1beta1
kind: Ingress
```

```
metadata:
  name: simple
  annotations:
     nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
```

2. 登录容器服务管理控制台，在集群列表页面中，单击目标集群右侧的**控制台**，进入 Kubernetes Dashboard 页面。
3. 单击**创建**，开始创建应用。
4. 单击**使用文件创建**。选择刚才保存的 `nginx-ingress.yml` 文件。
5. 单击**上传**。

   这样就创建了一个 Ingress 的七层代理路由到 `http-svc` 服务上。
6. 在 Kubernetes Dashboard 上定位到 default 命名空间，选择**访问权**。

   可以看到您刚刚创建的 Ingress 资源及其访问地址 `http://118.178.174.161/svc`。
7. 打开浏览器输入该地址即可访问前面创建的 `http-svc` 服务。

13 创建配额

您可以通过为命名空间创建配额来限制命名空间可以使用的资源量。

操作步骤

1. 登录容器服务管理控制台。
2. 单击左侧导航栏中的**配额管理**。
3. 单击页面右上角的**创建**。
4. 选择所需的命名空间并填写配额信息。

   您可以通过配额限制命名空间所使用的以下资源。

资源类目	资源项	说明
计算资源限制	CPU	设置命名空间可以使用的最大CPU。
	内存	设置命名空间可以使用的最大内存。单位为Mi或Gi。
存储资源限制	存储空间	设置命名空间可以使用的最大存储空间。单位为Mi或Gi。
	挂载磁盘数目	设置命名空间最多可以挂在的磁盘的数量。
其他资源限制	configmap数目	设置命名空间中最多可以创建的configmap数量。
	pod数目	设置命名空间中最多可以创建的pod数量。
	service数目	设置命名空间中最多可以创建的service数量。
	loadbalancer数目	设置命名空间中最多可以创建的负载均衡的数量。

   此外，您还可以通过**高级**选项自定义您所需的资源限制。

5. 设置完成后，单击**确定**。