阿里云
容器服务

用户指南

文档版本：20181116

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. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，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、发行或公开发表阿里云网站、产品程序或内容。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。

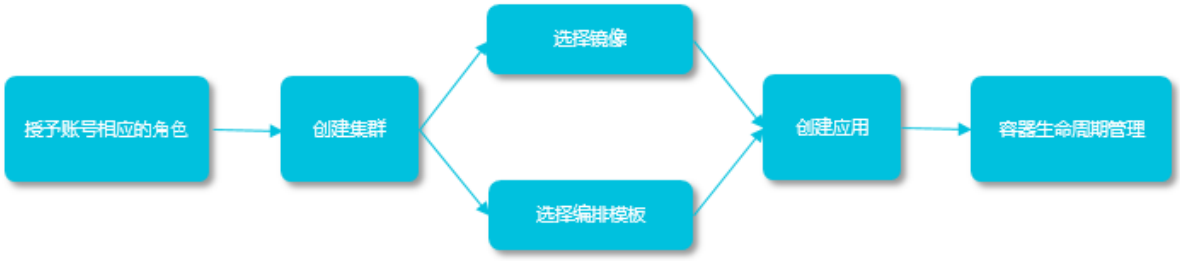
通用约定

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

1 使用流程

完整的容器服务使用流程包含以下步骤：



步骤 1：**授予账号相应的角色**。

详细信息参见角色授权。

步骤 2：**创建集群**。

您可以选择集群的网络环境，设置集群的节点个数和配置信息。

步骤 3：**通过镜像或编排模板创建应用**。

您可以使用已有的镜像或编排模板，或者新建镜像或者编排模板。

如果您的应用由多个镜像承载的服务组成，可以选择通过编排模板创建应用。

步骤 4：**查看部署后应用的状态和相应的服务、容器信息**。

2 集群管理

2.1 集群简介

一个集群（cluster）指容器运行所需要的云资源组合，关联了若干云服务器节点、负载均衡等云资源。

集群创建

您可以通过多种方式创建一个集群：

方法一：创建一个集群，并同时创建若干个云服务器。

您可以通过容器服务直接创建一个包含若干个新云服务器的集群。

详细信息参见创建集群。

通过此方式创建的云服务器皆为按量付费，如需要包年包月的云服务器，可以单独购买再通过接下来的 **方法二** 操作。

方法二：创建一个零节点的集群并添加已有的云服务器。

1. 创建一个零节点的集群。

   如果您已经在云服务器 ECS 上购买了若干个云服务器，可以在容器服务上创建一个零节点的集群。

   操作方式同 **方法一**，您只需要选择**添加已有节点**。

2. 添加已有的云服务器。

   您可以通过以下两种方法将已有云服务器添加到容器服务中。

   - 重置云服务器的镜像，将其自动加入集群。

     此种方式会重置云服务器的镜像和系统盘，需要谨慎。但是这种方式加入的服务器比较干净。

   - 在云服务器上执行脚本，将云服务器手动加入集群。

     此种方式适合于不希望重置云服务器的镜像。

   详细信息参见添加已有节点。

集群管理

支持集群扩容、删除、清理、连接等操作。更多详细的内容，请参考：

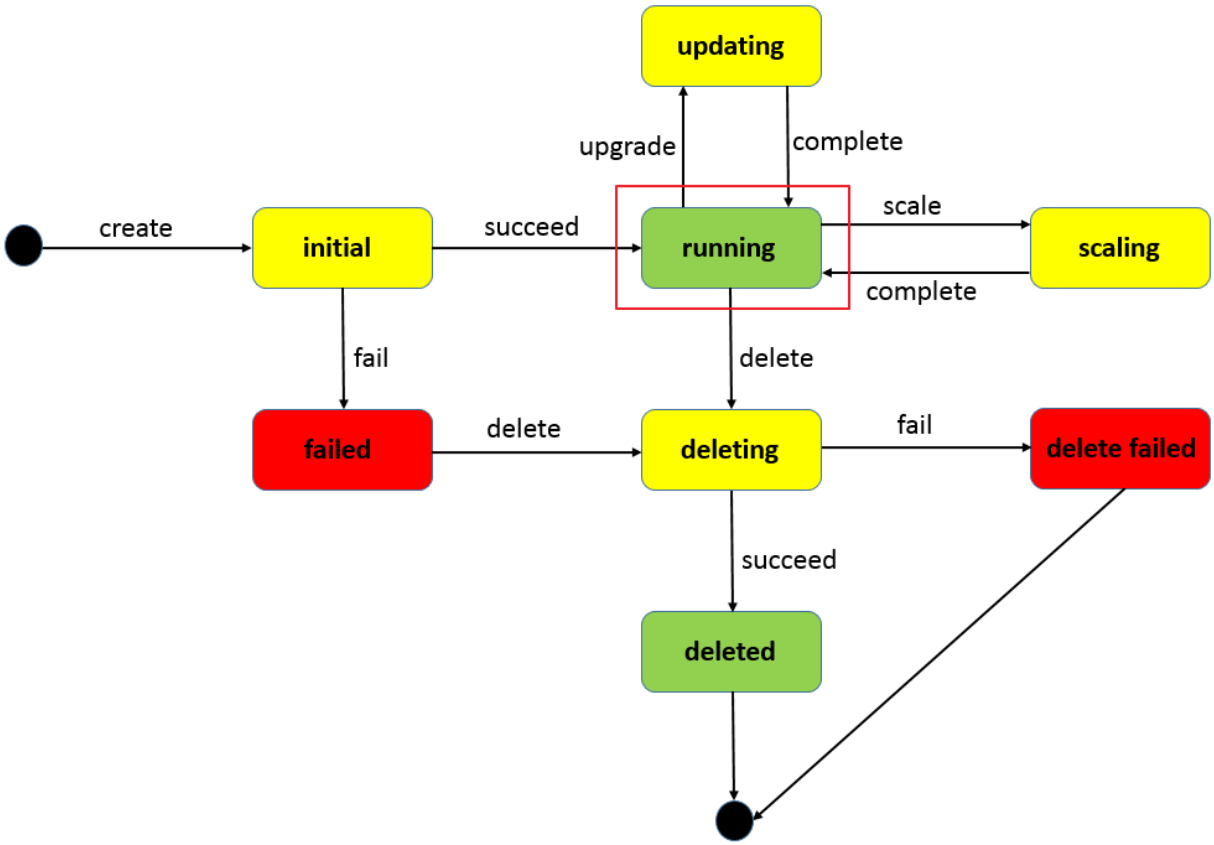
- 搜索集群
- 扩容集群
- 下载集群证书
- 清理集群磁盘
- 删除集群

2.2 集群的生命周期

表 2-1: 集群状态说明

状态	说明
待激活（inactive）	集群创建成功，但是集群中没有节点
初始化中（initial）	集群正在申请相应的云资源
运行中（running）	集群申请云资源成功
更新中（updating）	集群升级 Agent 版本
伸缩中（scaling）	变更集群的节点数量
创建失败（failed）	集群申请云资源失败
删除中（deleting）	集群删除中
删除失败（delete_failed）	集群删除失败
已删除（deleted，该状态用户不可见）	集群删除成功

图 2-1: 集群状态流转



2.3 创建集群

您可以在创建集群的时候同时指定云服务器的配置和个数，也可以创建一个零节点的集群，之后再绑定其他云服务器。

说明：

如果您选择创建一个零节点集群，创建完成后，集群会处于“待激活”状态，添加云服务器后就可以激活集群（变为“运行中”状态）。有关如何向集群中添加已有云服务器，参见 添加已有节点。

使用须知

容器服务在创建集群的过程中，会进行如下操作：

- 如果您勾选**自动创建负载均衡**，系统会创建负载均衡，并配置 80->9080 监听。
- 创建安全组，安全组的规则如下：

  VPC 网络入方向：

- 如果您已经开通了 RAM 服务，系统会创建 RAM 子账号。
- 如果您选择**创建节点**，系统会创建 ECS。同时为 ECS 分配公网 IP（如果是 VPC 网络，则会分配 EIP，同时会创建相应的路由规则）。
- 集群创建过程中，容器服务会使用您设置的**登录密码**配置 ECS 节点。

> **说明：**
> 容器服务不会保存该密码。

- VPC 节点配置失败时，容器服务会收集节点创建初始化的标准输出信息。您可以在集群的日志信息中查看。

限制说明

- 用户账户需有 100 元的余额并通过实名认证，否则无法创建按量付费的 ECS 实例和负载均衡。
- 随集群一同创建的负载均衡实例只支持按量付费的方式。
- 每个账号默认可以创建的云资源有一定的配额，如果超过配额创建集群会失败。请在创建集群前确认您的配额。如果您需要提高您的配额，请提交工单申请。
  - 每个账号默认最多可以创建 5 个集群（所有地域下），每个集群中最多可以添加 20 个节点。
  - 每个账号默认最多可以创建 100 个安全组。
  - 每个账号默认最多可以创建 60 个按量付费的负载均衡实例。
  - 每个账号默认最多可以创建 20 个EIP。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航中的**集群**，单击右上角的**创建集群**。

3. 设置集群的基本信息。

   - **集群名称**： 要创建的集群的名称。可以包含 1~63 个字符，包括数字、中文字符、英文字符和连字符（-）。

     说明：

     集群名称在同一个用户和同一个地域下必须唯一。

   - **地域**： 所创建集群将要部署到的地域。
   - **可用区**：集群的可用区。

     说明：

     您可以根据您的服务器分布情况，选择不同的地域和可用区。

4. 设置集群的网络类型。目前支持专有网络 VPC 网络类型。

   专有网络 VPC 需要配置相关信息。

专有网络 VPC 支持您基于阿里云构建一个隔离的网络环境，您可以完全掌控自己的虚拟网络，包括自由 IP 地址范围、划分网段、配置路由表和网关等。

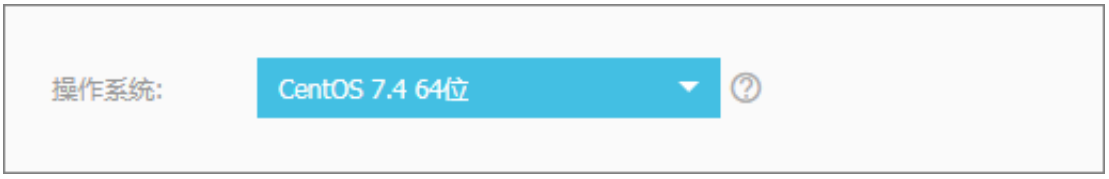
您指定一个 VPC、一个 VSwitchId 和容器的起始网段（Docker 容器所属的子网网段，为了便于 IP 管理，每个虚拟机的容器属于不同网段，容器子网网段不能和虚拟机网段冲突）。为了防止网络冲突等问题，建议您为容器集群建立属于自己的 VPC/VSwitchId。

5. 添加节点。

您可以在创建集群的同时创建若干个节点，或者创建一个零节点集群并添加已有云服务器。有关如何添加已有云服务器的详细信息，参见添加已有节点。

- 创建节点

1. 设置节点的操作系统。



目前支持的操作系统包括 Ubuntu 16.04 64 位和 CentOS 7.4 64 位。

2. 设置云服务器的实例规格。

您可选择不同的实例规格和数量，并指定数据盘的容量（云服务器默认带有 20G 大小的系统盘）和登录密码。集群创建过程中，容器服务会使用您设置的 登录密码 配置 ECS 节点，但是容器服务不会保存该密码。

> **说明：**
>
> - 如果您选择了数据盘，它会被挂载到 `/var/lib/docker` 目录，用于 Docker 镜像和容器的存储。
> - 从性能和管理考虑，建议您在宿主机挂载独立的数据盘，并利用 Docker 的 volume 对容器的持久化数据进行管理。

- 添加已有节点

  您可以单击下边的**选择已有实例**将已有的云服务器添加到集群中，或者直接单击**创建集群**等集群创建完成后再通过集群列表页面添加已有云服务器，参见 添加已有节点。

6. 配置 EIP。

当您将网络类型设置为 VPC 时，容器服务会默认给每一个专有网络下的云服务器配置一个 EIP。如果不需要，您可以勾选**不保留公网EIP**复选框，但是需要额外配置 SNAT 网关。

> **说明：**
>
> 每个账号最多可申请 20 个EIP。当您创建集群时选择使用 VPC 网络且选择系统自动创建 EIP 时，如果您账号下的 EIP 已达到配额，创建集群会失败。

7. 创建一个负载均衡实例。

目前创建集群会默认创建一个负载均衡实例。您可以通过这个负载均衡实例访问集群内的容器应用。所创建的负载均衡实例为按量付费实例。

8. 在 ECS 节点上安装云监控插件。

您可以选择在节点上安装云监控插件，从而在云监控控制台查看所创建 ECS 实例的监控信息。

9. 将节点 IP 添加到 RDS 实例的白名单。

您可以选择将所创建节点的 IP 添加到 RDS 实例的白名单中，方便 ECS 实例访问 RDS 实例。

说明：

- 建议您选择**创建节点**时进行该项配置。
- 如果您选择**添加已有节点**，对该项进行配置。在集群创建界面就添加已有 ECS 实例，配置可用；在创建零节点集群后添加 ECS 实例，该配置不生效。
- 您仅能将 ECS 实例的 IP 添加到位于同一地域的 RDS 实例的白名单中。

a. 单击**请选择您想要添加白名单的RDS实例**。

b. 在弹出的对话框中选择所需的 RDS 实例并单击**确定**。

10. 单击**创建集群**。

集群创建成功后，如果需要单独配置云服务器或负载均衡，可以去相应的控制台进行相关操作。

后续操作

您可以查看集群创建日志。在**集群列表**页面，选择所创建的集群并单击**查看日志**。

您可以在创建的集群中创建应用。有关创建应用的详细信息，参见创建应用。

参考文档

如果集群创建失败，您可以参考创建集群失败常见错误 进行问题排查。

2.4 集群参数配置说明

本文档旨在帮助您理解创建集群界面的参数含义，顺利进行参数配置，其中对一部分参数，会给出一些文档资源，供您了解更多信息。

集群名称

设置该集群的名称。

- 名称为 1-63 个字符，可包含数字、汉字、英文字符，或”-“。但不能以分隔符（-）开头。
- 您可以在创建成功后，在集群列表页面对目标集群的名称进行修改。

地域和可用区

容器服务授权创建 ECS 实例的地域和可用区，目前容器服务支持的地域和可用区属于 ECS 产品的子集。请参见 地域和可用区。

网络类型

选择 ECS 实例的网络类型，支持 VPC 网络，阿里云 VPC 支持您创建自定义的专有网络，不同的专有网络之间二层逻辑隔离，同时您可以灵活规划各个集群的网段，非常适合大规模容器集群的使用场景，提供了更高的安全性和灵活性。为了更好的保障系统安全和混合云业务的支持，从2018年1月1日起不再支持经典网络、非IO优化实例的新集群创建。

容器起始网段

该参数只在您选择 VPC 网络时，才会要求您进行配置。在规划时，需要保证容器的初始网段和 VPC 网段不能重叠。

- 每个专有网络只能指定 1 个网段，其中 172.16.0.0/12 是专有网络的默认网段。
- 在创建 VPC 网络的容器服务集群时，需要指定对应的容器网段，目前容器服务支持的容器网段是 192.168.1.0/24 和 172.[16-31].1.0/24

是否新增节点

容器服务提供两种新增节点的方式：创建节点和添加已有节点。创建节点授权容器服务在创建集群时自动创建 ECS 实例，自动将新建的 ECS 实例加入到集群中。添加节点支持将已有的 ECS 实例

添加到集群中，其中，可以在集群创建页面直接进行添加，或者创建好零节点集群后，在集群页面进行添加，详情参考 添加已有节点。

节点类型

默认为按量付费。等待创建节点成功后，您可以前往 ECS 控制台转换为包年包月实例。

操作系统

设置 ECS 实例中安装的操作系统，推荐使用 Ubuntu 16.04 64位 和 CentOS 7.4 64位。

实例系列

不同系列对应不同的实例规格族。ECS 实例是以一定的规格来为您提供相应的计算能力的。根据业务场景和使用场景，ECS 实例可以分为多种规格族。具体各个实例规格族的使用场景，请参见 实例规格族。

实例规格

ECS 实例规格定义了实例的 CPU 和内存的配置这两个基本属性。但是，ECS 实例只有同时配合磁盘、镜像和网络类型，才能唯一确定一台实例的具体服务形态。

实例数量

创建 ECS 实例的数量，单个集群的 ECS 实例数量不超过 20 台机器。不建议创建单节点集群，以提高集群的可用性。控制台默认设置为 2 个节点。

系统盘类型

选择安装系统的云盘类型，支持高效云盘和 SSD 云盘。根据您对 ECS 实例的系统性能要求，进行选择，两种云盘的性能指标对比，请参考 块存储性能。

数据盘配置

选择容器挂载的数据盘类型，需要勾选**挂载数据盘**，并选择数据盘容量，才算真正配置完毕。数据盘会被挂载到容器的 `/var/lib/docker` 目录，用于存储镜像和容器数据。

登录密码和确认密码

设置和确认 ECS 节点的登录密码。8-30 个字符，且同时包含三项（大、小写字母，数字和特殊符号），您在 ECS 控制台登录或 SSH 到 ECS 实例上进行管理时需要用到。

说明：

- 集群创建过程中，容器服务将使用该密码配置 ECS 节点，容器服务不会保存该密码。
- 密码为初始化使用，请妥善保管。

EIP

EIP 即是弹性公网 IP，用于公网访问。默认保留，如果选择不保留，集群在完成实例初始化后释放 EIP 。您可用 *NAT 网关*或自行 *Linux系统配置SNAT* 实现公网访问。

负载均衡

创建集群会默认创建一个公网负载均衡实例，计费类型为按量付费，用于流量分发控制服务，实现服务高可用。

云监控插件

勾选则在 ECS 节点上安装云监控插件，从而监控集群下 ECS 实例操作系统级别的性能指标。

RDS 白名单

您可以选择将所创建节点的 IP 添加到 RDS 实例的白名单中，方便 ECS 实例访问 RDS 实例。

- 建议您选择 **创建节点** 时进行该项配置。
- 如果选择在 **添加已有节点** 时进行该项配置，请在集群创建界面就添加已有 ECS 实例，该配置生效；如果在创建零节点集群后添加 ECS 实例，该配置不生效。
- 您仅能将 ECS 实例的 IP 添加到位于同一地域的 RDS 实例的白名单中。

默认安全组信息

容器服务设置了默认的安全组，容器服务只设置入方向的安全组规则，您可以在集群创建成功后，根据您的业务场景对安全组进行配置。具体请参见容器服务安全组规则

- 443端口和80端口可以根据自己的需求选择放开或者关闭。
- ICMP规则建议保留，用于节点间通信，方便排查问题。有些工具也依赖ICMP
- 务必确保放开所有您需要的端口，否则会导致服务不可访问。通过 SLB 访问的端口不需要放开。

2.5 创建GN4型GPU 云服务器集群

您可以创建集群，以使用 GN4 型 GPU 云服务器。

说明：

如果您选择创建一个零节点集群，创建完成后，集群会处于“待激活”状态，添加云服务器后就可以激活集群（变为“运行中”状态）。有关如何向集群中添加已有云服务器，参见 添加已有节点。

前提条件

目前，按量付费的 GN4 型 GPU 云服务器以白名单方式开放给用户使用。如果您需要使用按量付费的 GN4 型 GPU 云服务器，需要 提交 *ECS* 工单 进行申请。

遇到没有资源的情况怎么办？

如果您已经通过了使用 GN4 型 GPU 云服务器的申请，但是在选择实例规格 时，未找到 GN4 型 GPU 云服务器（32核 48GB（ecs.gn4.8xlarge）或 56核 96GB（ecs.gn4.14xlarge）），建议您采取以下措施：

- 更换地域
- 更换可用区

如果依然没有资源，建议您耐心等待一段时间再购买。实例资源是动态的，如果资源不足，阿里云会尽快补充资源，但是需要一定时间。您可以在晚些时候或者次日再尝试购买。

使用限制

- 目前，容器服务仅支持在华南 1 、华东 2 和华北 2 地域创建 GN4 型 GPU 云服务器集群。
- 目前，GN4 型 GPU 云服务器只支持专有网络（VPC）。
- 为了保证您的 GN4 型 GPU 云服务器集群的使用效果，建议集群中统一使用 GN4 型 GPU 云服务器，不要添加其它规格族的 ECS 实例到 GN4 型 GPU 云服务器集群。
- 默认情况下，您最多可以创建 5 个集群（所有地域下），每个集群中最多可以添加 20 个节点。如果您需要创建更多的集群或添加更多的节点，请提交 工单申请。
- 目前负载均衡只支持按量付费的方式，后续将提供更多选择。
- 用户账户需有 100 元的余额并通过实名认证，否则无法创建按量付费的 ECS 实例和负载均衡。

操作步骤

1. 登录 容器服务管理控制台。
2. 单击左侧导航中的集群，单击右上角的创建集群。

3. 设置集群的基本信息。

- **集群名称**：要创建的集群的名称。可以包含 1~63 个字符，包括数字、中文字符，英文字符和连字符（-）。

  说明：

  集群名称在同一个用户和同一个地域下必须唯一。

- **地域**：所创建集群将要部署到的地域。选择**华南 1**、**华东 2** 或、。

  说明：

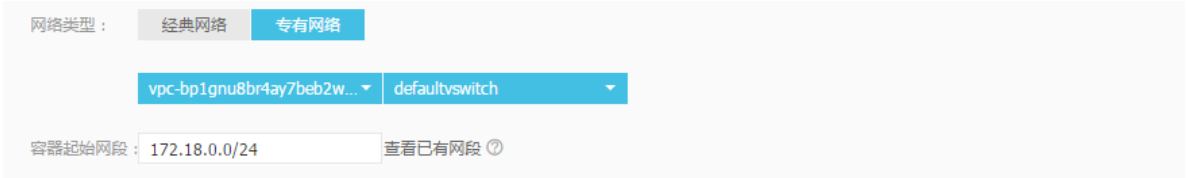
  目前，仅支持在华南 1、华东 2 和华北 2地域创建 GN4 型 GPU 云服务器集群。

- **可用区**：集群的可用区。

  说明：

  您可以根据您的服务器分布情况，选择不同的地域和可用区。

4. 设置集群的网络类型为专有网络并配置相关信息。



专有网络 VPC 支持您基于阿里云构建一个隔离的网络环境，您可以完全掌控自己的虚拟网络，包括自由 IP 地址范围、划分网段、配置路由表和网关等。

专有网络需要您指定一个 VPC、一个 VSwitchId 和容器的起始网段（Docker 容器所属的子网网段，为了便于 IP 管理，每个虚拟机的容器属于不同网段，容器子网网段不能和虚拟机网段冲突）。

为了防止网络冲突等问题，建议您为容器集群建立属于自己的 VPC/VSwitchId。

5. 添加节点。

您可以在创建集群的同时创建若干个节点，或者创建一个零节点集群并添加已有云服务器。有关如何添加已有云服务器的详细信息，参见添加已有节点。

- 创建节点

1. 设置节点的操作系统。

目前支持的操作系统包括 Ubuntu 16.04 64 位和 CentOS 7.4 64 位。

2. 设置云服务器的实例规格。

- 实例系列选择**系列III**。
- 实例规格选择32核 48GB（ecs.gn4.8xlarge） 或 56核 96GB（ecs.gn4.14xlarge）。

> **说明：**
> 如果您已经通过了 GN4 型 GPU 云服务器的使用申请，但是未找到这两种实例规格，说明目前这两种规格的实例没有资源，建议晚些时候或者次日再尝试购买。

您可选择实例的数量，并指定数据盘的容量（云服务器默认带有 20G 大小的系统盘）和登录密码。

> **说明：**
>
> - 如果您选择了数据盘，它会被挂载到*/var/lib/docker*目录，用于 Docker 镜像和容器的存储。
> - 从性能和管理考虑，建议您在宿主机挂载独立的数据盘，并利用 Docker 的 volume 对容器的持久化数据进行管理。

- 添加已有节点

  您可以单击下边的**选择已有实例**将已有的云服务器添加到集群中，或者直接单击**创建集群**等集群创建完成后再通过集群列表页面添加已有云服务器。

> **说明：**
>
> 为了保证您的 GN4 型 GPU 云服务器集群的使用效果，建议集群中统一使用 GN4 型 GPU 云服务器，不要添加其它规格族的 ECS 实例到 GN4 型 GPU 云服务器集群。

6. 配置 EIP。

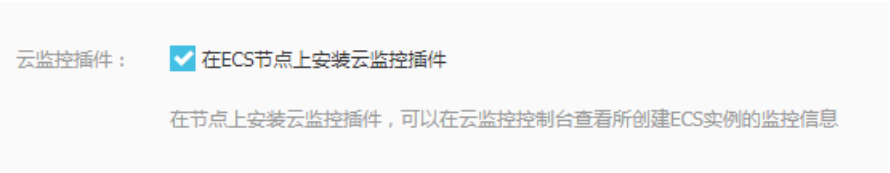
   当您将网络类型设置为 VPC 时，容器服务会默认给每一个专有网络下的云服务器配置一个EIP。如果不需要，您可以勾选**不配置公网EIP** 复选框，但是需要额外配置 SNAT 网关。

7. 创建一个负载均衡实例。

   目前创建集群会默认创建一个负载均衡实例。您可以通过这个负载均衡实例访问集群内的容器应用。所创建的负载均衡实例为按量付费实例。

8. 在 ECS 节点上安装云监控插件。

   您可以选择在节点上安装云监控插件，从而在云监控控制台查看所创建 ECS 实例的监控信息。



9. 单击**创建集群**。

   集群创建成功后，您可以在**集群列表**页面单击所创建集群的名称查看集群中节点的信息。

如果需要单独配置云服务器或负载均衡，可以去相应的控制台进行相关操作。

后续操作

您可以查看集群创建日志。在集群列表页面，选择所创建的集群并单击查看日志。

您可以在创建的集群中创建应用。有关创建应用的详细信息，参见创建应用。

2.6 管理本地集群

容器服务不仅可以管理 ECS 机器，也可以管理 ECS 之外的机器。无论是您本地的机器，还是 IDC 的机器，都可以在容器服务里进行管理。这些机器加入到容器服务之后，您可以利用容器服务的能力管理机器、部署应用、查看应用状态。

> **说明：**
> 本地集群功能目前以白名单方式开放。如果您需要本地集群，请提交 工单申请。

本地集群使用限制

现有的版本中，以下扩展服务暂不支持：

- 数据卷只支持 OSSFS，不支持 NAS 和云盘。
- 不支持负载均衡。如果您需要使用 Routing，可以将本地机器的 9080 端口加到本地负载均衡设备上。9080 端口是 Routing 的端口。
- 不支持自动采集日志到 Logstore 中。

创建本地集群

默认创建的集群只能添加 ECS 机器。如果您需要添加本地机器，需要先创建一个本地集群。只有本地集群才可以添加本地机器。

1. 登录 容器服务管理控制台。
2. 单击左侧导航栏中的集群。
3. 单击页面右上角的创建集群下拉按钮并单击创建本地集群。

4. 填写集群信息。

> **说明：**
>
> - 容器网段不能与您的宿主机网段冲突。
> - 创建的本地集群为零节点集群。

5. 单击创建集群。

添加机器

1. 单击左侧导航栏中的集群。
2. 单击更多并在弹出菜单中单击添加已有实例。

系统会生成一条 shell 命令。

> **说明：**
>
> 其中，`eth0` 是容器之间通信所用的宿主机网卡，您需要根据情况判断网卡的名称。

3. 复制 shell 命令并在机器上执行。如果命令运行成功，则机器被添加进集群。

部署应用

本地集群可以和普通集群一样部署应用。

2.7 添加已有节点

您可以将已购买的云服务器添加到指定集群。

> **说明：**
> 默认情况下，每个集群中最多可添加 20 个节点。如果您需要添加更多节点，请提交 工单申请。

您可以通过以下两种方法之一添加已有云服务器：

- **自动添加**：通过此方法添加实例会重置镜像和系统盘。您可以选择一次添加一个或多个云服务器。
- **手动添加**：在云服务器上执行脚本。您一次仅能选择添加一个云服务器。

前提条件

如果之前没有创建过集群，您需要先创建一个集群。有关创建集群的详细信息，参见创建集群。

使用说明

- 添加的云服务器必须与集群在同一地域并使用相同类型的网络（专有网络）。
- 添加已有云服务器时，请确保您的云服务器有 EIP （专有网络），或者相应 VPC 已经配置了 NAT网关。总之，需要确保相应节点能正常访问公网，否则，添加云服务器会失败。
- 容器服务不支持添加不同账号下的云服务器。
- 如果您选择**手动添加**，请注意以下事项：
  - 如果您的云服务器中已经安装了 Docker，手动添加的时候可能会失败。建议在添加云服务器之前执行清理命令。命令如下所示：

    Ubuntu：`apt-get remove -y docker-engine`，`rm -fr /etc/docker/ /var/lib/docker /etc/default/docker`

    CentOS：`yum remove -y docker-engine`，`rm -fr /etc/docker /var/lib/docker`
  - 容器服务的节点对系统有要求，推荐您使用 Ubuntu 16.04 和 CentOS 7 64位系统。我们对这两个系统进行了非常严格的稳定性和兼容性测试。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择所需的集群，单击**更多**并在弹出菜单中单击**添加已有节点**，如下图所示。

4. 添加 ECS 实例。

   页面显示的实例列表，是根据集群所定义的地域和网络类型，从您的所有云服务器列表中筛选后同步过来的。

   您可以通过以下两种方法之一添加实例。

   - 自动添加

     说明：

     通过此方法添加实例会重置镜像和系统盘。请谨慎使用。添加之前请创建快照进行数据备份。有关如何创建快照，参见创建快照。

     1. 选择要添加的实例，单击下一步。

        您可以同时添加一个或多个实例。

     2. 设置实例信息，单击下一步并在弹出的确认对话框中单击确定。

     3. 单击完成。

   - 手动添加，您需要在云服务器上执行脚本。

     1. 选择手动添加，选择一个 ECS 实例并单击下一步。

        您一次只能添加一个实例。

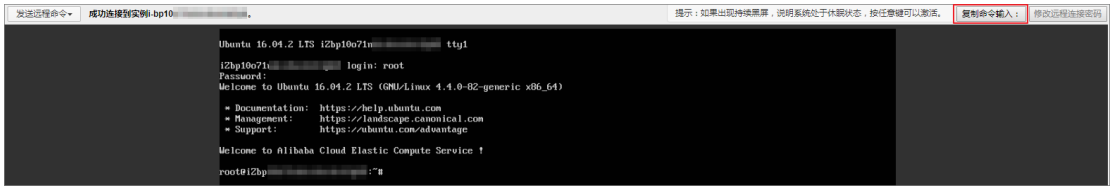
     2. 确认实例信息并单击下一步。

     3. 页面显示专属这台云服务器的脚本命令。单击登录云服务器**xxxxxxx**。

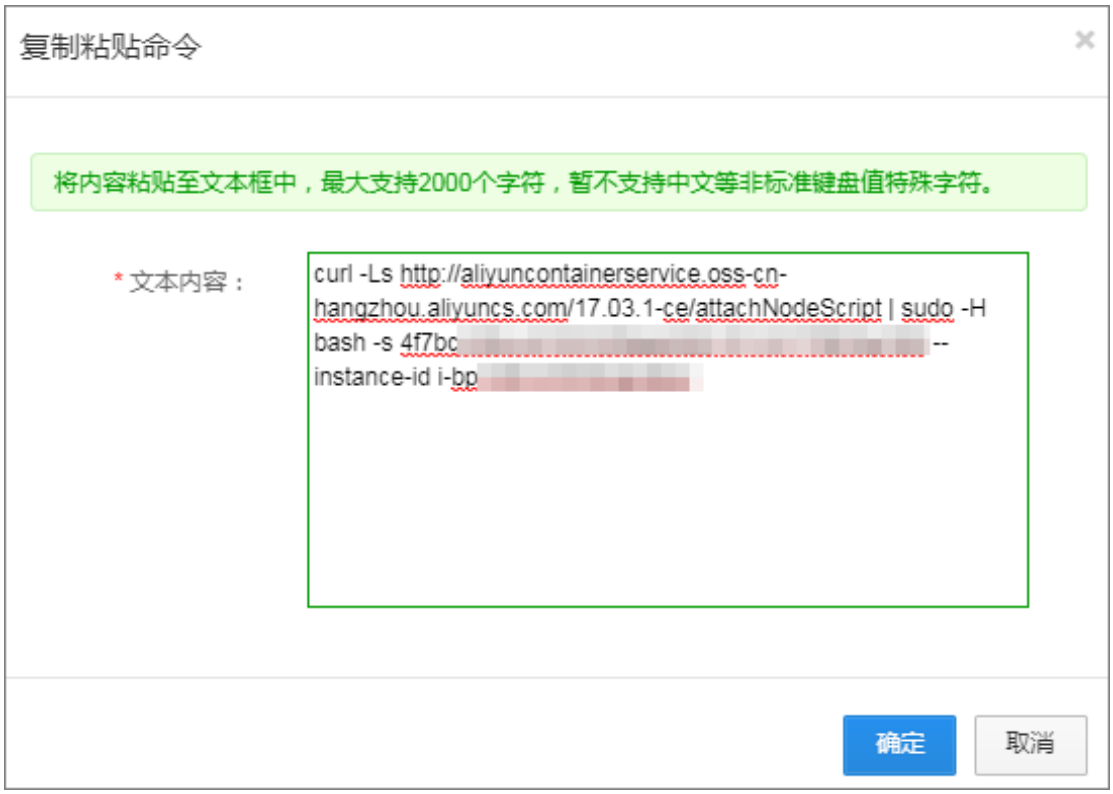
4. 弹出的对话框中显示远程连接密码，复制并妥善保存该密码，并单击关闭。

5. 在弹出的对话框中，输入远程连接密码并单击确定。

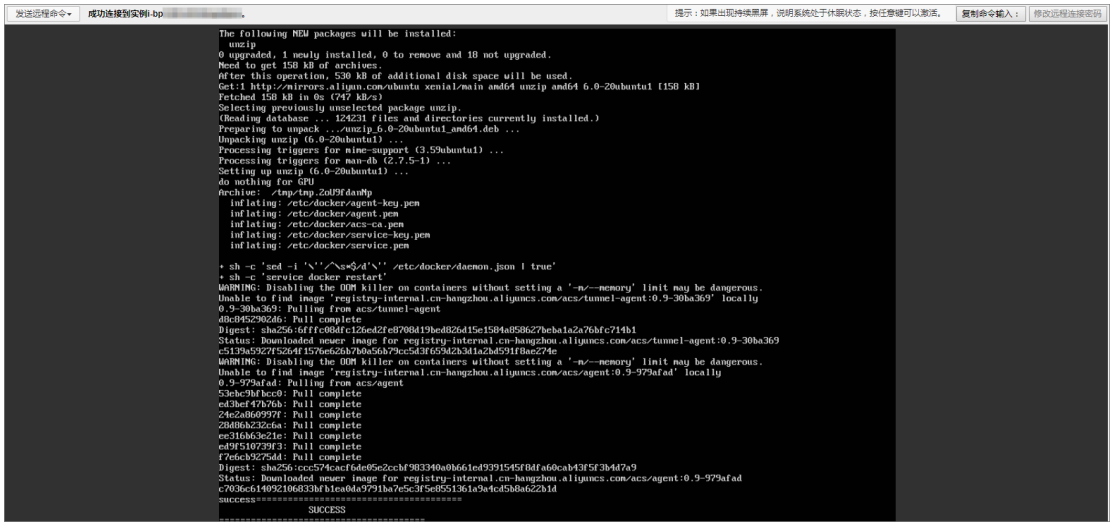
6. 输入登录云服务器的账号（root）和密码并回车，登录到云服务器。



7. 单击**复制命令输入**，将上面的脚本命令黏贴到弹出的对话框中，单击**确定**并回车。

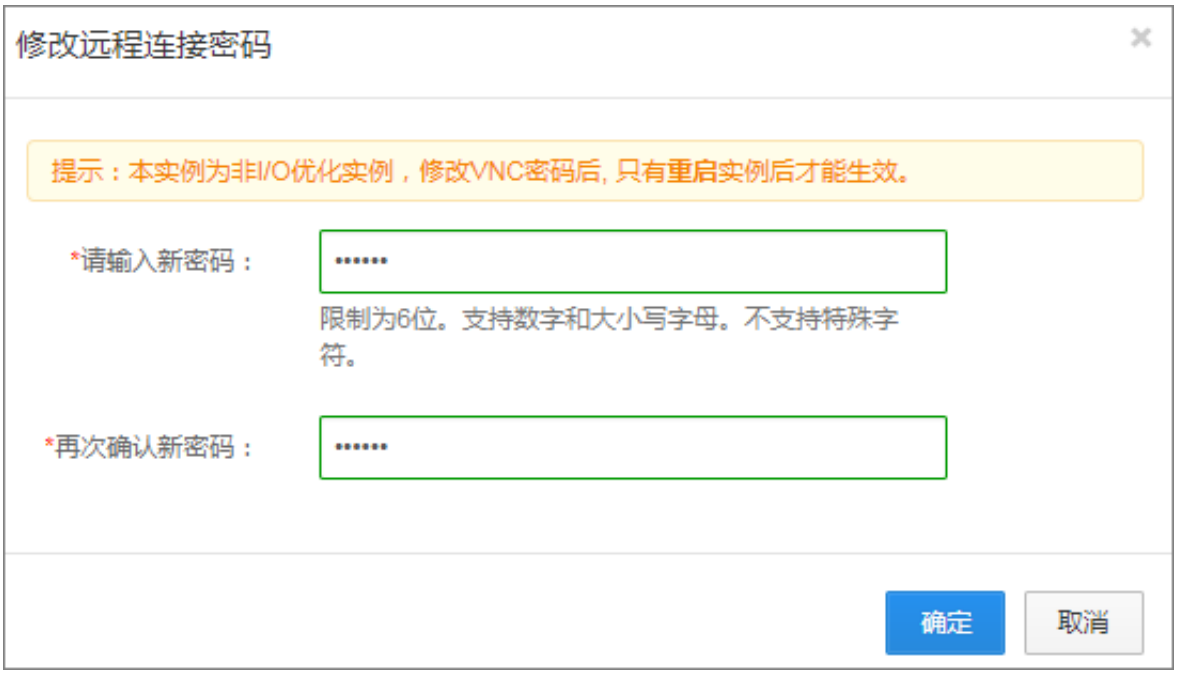


系统开始执行脚本。等待脚本执行成功，显示 success。该云服务器即添加成功。



相关操作

您可以在脚本执行页面修改您的云服务器远程连接密码。单击**修改远程连接密码**，在弹出的对话框中输入新密码并单击**确定**。



2.8 跨可用区节点管理

为了提高应用的高可用性，在创建集群的时候，可以选择将多个节点分布在不同的可用区。

在创建集群的时候，您可以先创建一个节点的集群或者直接创建零节点的集群，待集群创建完成后，通过集群扩容或者添加已有 ECS 实例的方式来增加不同可用区的节点。

说明：

- 通过集群扩容添加的节点为按量付费节点。
- 通过添加已有实例添加的节点可以是按量付费节点也可以是包年包月的节点。

通过集群扩容添加不同可用区的节点

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。
3. 选择要扩容的集群，单击更多 > 集群扩容。如下图所示。

4. 在弹出的对话框中，设置新节点的规格。

   您可以通过设置可用区创建分布在不同可用区的节点。

5. 单击集群扩容将新节点添加到集群中。
6. 重复以上步骤，创建位于不同可用区的节点并添加到集群中。

通过添加已有实例添加不同可用区的节点

前提条件

使用本方法来添加节点，您需要首先通过 ECS 的售卖页面自行购买 ECS 实例。购买的过程中可以为实例选择不同的可用区。

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。
3. 选择要添加已有节点的集群，单击更多 > 添加已有节点。如下图所示。

4. 选择位于不同可用区的 ECS 实例并自动或手动将其添加到集群中。

   有关添加已有实例的详细信息，参见添加已有节点。

5. 重复以上步骤，为集群添加位于不同可用区的节点。

2.9 为集群绑定和解绑负载均衡

您可以在创建集群时自动创建一个按量付费的负载均衡实例，或者在集群创建完成后为集群绑定一个包年包月或按量付费的负载均衡实例。

容器服务集群支持绑定公网负载均衡实例和私网负载均衡实例。

使用限制

- 您只能为集群绑定位于同一地域下的负载均衡实例。
- 不支持绑定跨账号的负载均衡实例。
- VPC 集群支持绑定公网负载均衡实例和 VPC 私网负载均衡实例。
- 一个集群仅支持绑定一个负载均衡实例。
- 两个集群不能共用一个负载均衡实例。

前提条件

您已经在 负载均衡管理控制台 创建了负载均衡实例并且为负载均衡实例配置了监听后端服务器的 TCP 9080 端口。

有关如何创建，参见 创建负载均衡实例。

绑定负载均衡实例

1. 登录 容器服务管理控制台。
2. 选择要配置的集群并单击管理。

3. 单击左侧导航栏中的负载均衡 > 绑定**SLB** 。

4. 从下拉列表中选择您要绑定的负载均衡实例并单击确定。

> **说明：**
> 如果您所选择的负载均衡实例已经绑定了后端服务器，系统会提示您“该负载均衡实例已经绑定了后端服务器”，您需要选择其它未绑定后端服务器的负载均衡实例。

重新绑定负载均衡实例

您可以根据您的需要更换集群绑定的负载均衡实例。

1. 登录 容器服务管理控制台。
2. 单击左侧导航栏中的集群。
3. 选择要配置的集群并单击管理。

4. 单击左侧导航栏中的负载均衡 > 重新绑定**SLB** 。

5. 从下拉列表中选择您要绑定的负载均衡实例并单击确定。

解绑负载均衡实例

如果您不再需要负载均衡实例，您可以通过容器服务管理控制台进行解绑。

1. 登录 容器服务管理控制台。
2. 单击左侧导航栏中的集群。
3. 选择要配置的集群并单击管理。

4. 单击左侧导航栏中的负载均衡 > 解绑SLB 。

2.10 设置集群根域名

背景信息

当您通过镜像创建 *Nginx* 并进行简单路由配置时，您只需要填写域名的前缀 `nginx`，即可获得 `$cluster_id.$region_id.alicontainer.com` 格式的域名。您可以通过设置集群根域名（本示例使用 `51ili.com`）来替换该域名。当您重新部署服务 `nginx` 时，域名从 `nginx.c2818a77aac20428488694c0cd1600e6e.cn-shenzhen.alicontainer.com` 变为 `nginx.51ili.com`，方便您使用自己的根域名对集群应用进行访问。

说明：

为了保证下面的示例能够工作，请先将 Agent 升级到最新版本。

操作步骤

1. 绑定一个负载均衡实例。

   a) 登录 容器服务管理控制台。

   b) 在Swarm菜单下，单击左侧导航栏中的集群。

   c) 选择要配置的集群（本示例为`routing-test-online`），单击管理。

   d) 单击左侧导航栏中的负载均衡。

   如果集群未绑定负载均衡实例，登录阿里云 负载均衡管理控制台，并创建一个负载均衡实例，然后回到本页面进行绑定。

   > **说明：**
   > 有关负载均衡在容器服务中的使用限制以及如何为集群绑定和解绑负载均衡，参见为集群绑定和解绑负载均衡。

2. 设置域名。

   a) 单击 域名设置，填写您自己购买的根域名，本示例中为51ili.com。

   b) 单击设置。

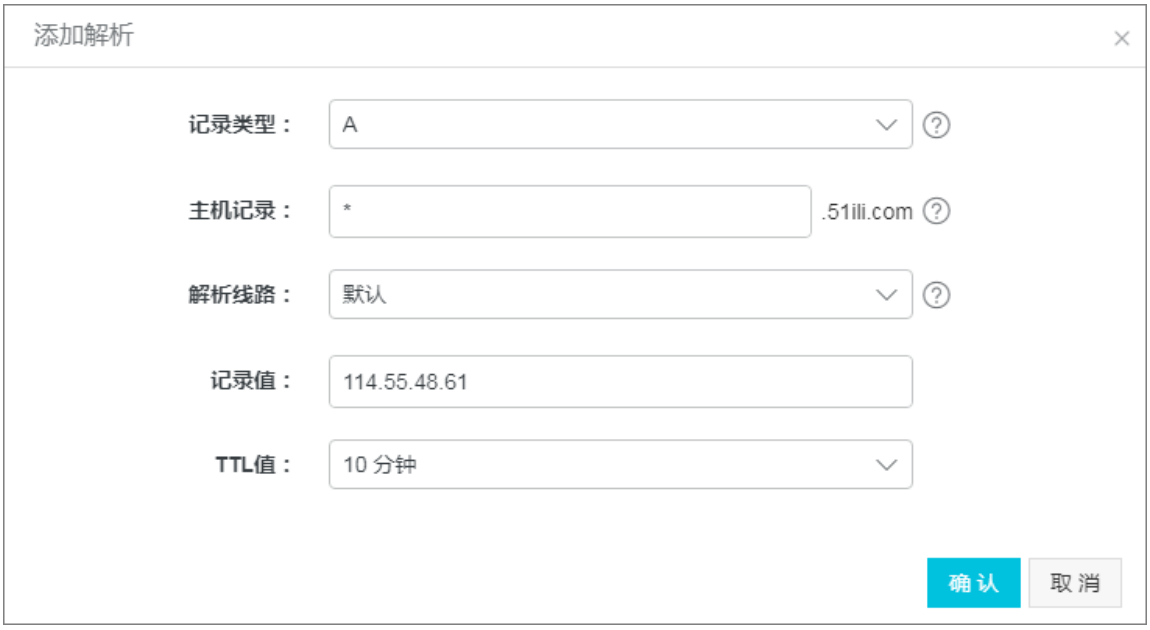
3. 将域名解析到绑定的负载均衡实例。

a) 在 SLB 控制台中，查找集群绑定的负载均衡实例。

b) 查看实例详情，找到绑定的负载均衡实例的服务地址。

c) 登录阿里云云解析 DNS 服务管理控制台，添加域名解析。

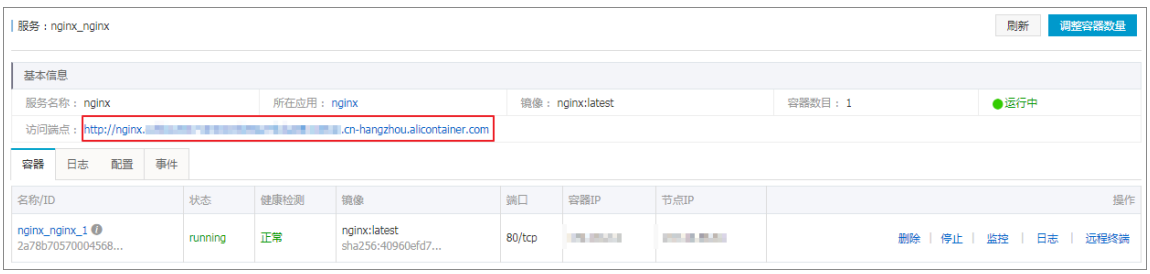
- 添加域名。若已有域名，跳过此步。
- 添加域名解析。其中记录值填写绑定的负载均衡实例的服务地址。



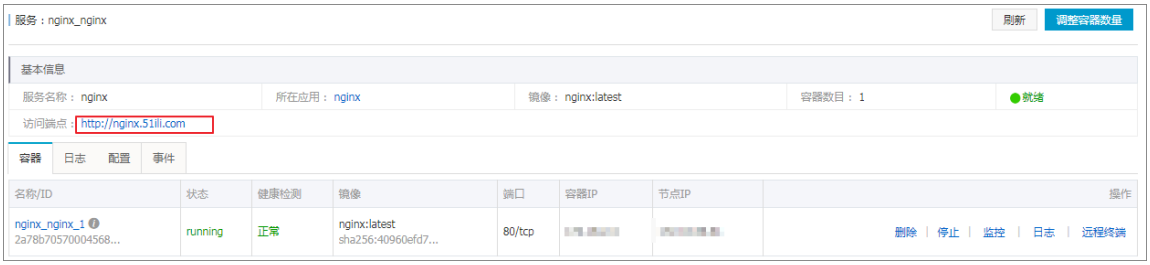
4. 重新部署 web 服务。

a) 重新部署应用，应用下的服务web访问端点发生了变化。

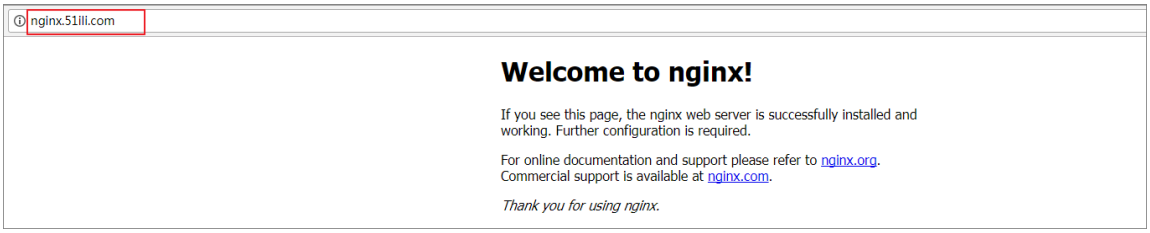
设置根域名之前的访问端点：



设置根域名之后的访问端点：



b) 访问最新的访问端点http://nginx.51ili.com。



2.11 下载集群证书

背景信息

您可以使用下载的证书通过 Docker Swarm API 或 Docker Client 连接集群暴露出来的 Endpoint，参见通过 *Docker* 工具连接集群。

操作步骤

1. 获取访问地址。

a) 登录 容器服务管理控制台。

b) 在 Swarm 菜单下，单击左侧导航栏中的集群，在集群列表中选择一个集群并单击管理。

c) 您可以查看集群的连接信息，如下图所示。

2. 下载和保存证书。

   要通过上面的访问地址访问 Docker 集群，您还需要配置 TLS 证书。

   在集群管理页面，单击**下载证书** 开始下载 TLS 证书。下载到的文件为 `certFiles.zip` 。在下面的例子中，下载的证书存放在 *~/.acs/certs/ClusterName/* 目录下。其中，`ClusterName` 是您集群的名字。您也可以使用其他目录，但是为了便于管理，推荐您将文件存放在 *~/.acs/certs/ClusterName/* 目录下。

   ```
   mkdir ~/.acs/certs/ClusterName/  #替换成真正的集群名字
    cd ~/.acs/certs/ClusterName/
    cp /path/to/certFiles.zip .
    unzip certFiles.zip
   ```

   certFiles.zip 文件包含*ca.pem*、*cert.pem* 和 *key.pem*。

2.12 扩容集群

前提条件

一个集群最多可以包含 10 个节点。

背景信息

您可以根据您的业务需求进行集群扩容。

> **说明：**
> 通过集群扩容添加的节点为按量付费节点。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择需要扩容的集群，单击更多并在下拉菜单中单击集群扩容。

4. 在弹出的对话框中，设置新节点的规格。

   您可以选择增加服务器节点的个数和相应的规格。

5. 单击集群扩容。

2.13 迁移集群

对于创建时间较早的Swarm集群，可以通过迁移集群保证集群的性能和稳定性。

背景信息

- 迁移集群的最晚时间会通过短信，站内信或邮件的方式告知，您需要在最晚迁移时间内完成Swarm集群的迁移，如果晚于该时间，系统将自动迁移集群。
- 迁移集群将重建集群节点到容器服务器的连接，不会影响已部署在集群中的应用，也不会新增或修改任何数据。但整个迁移过程中仍可能存在不可预知风险，请务必在业务低峰期操作。

操作步骤

1. 登录容器服务管理控制台。
2. 在Swarm菜单下，单击左侧导航栏中的集群，进入集群列表页面。
3. 选择需要迁移的集群，单击操作列的集群迁移。

4. 在提示对话框中单击确定。

> **说明：**
>
> 迁移集群过程中：
>
> - 无法在控制台进行信息查询、部署、升级等操作。
> - 无法通过集群接入点 API 连接到集群。
> - 集群中的数据、应用状态保持不变，部署在集群上的应用仍可以正常访问。
> - 迁移过程大约需要三分钟。

在集群列表页面，集群状态列显示迁移中。

预期结果

迁移集群完成后，在集群列表页面，集群状态列显示运行中：

> **说明：**
>
> - 集群 ID、接入点地址以及其他属性均保持不变。
> - 请您务必确认业务正常运行。
> - 迁移过程中，如有任何问题，请提交工单，并请在工单中附上：集群ID以及部署的应用是否正常。

2.14 搜索集群

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 在搜索框中输入要搜索的集群的名称或关键字。名称中带有该关键字的集群将显示在集群列表中。如下图所示。

> **说明：**
>
> 搜索不区分大小写。

2.15 删除集群

背景信息

您可以从容器服务中删除集群。删除集群时，会将关联的云服务器和负载均衡等云资源一起删除。请谨慎使用该操作。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择要删除的集群并单击删除，如下图所示。

4. 在弹出的对话框中，单击确定。

2.16 清理集群磁盘

背景信息

清理磁盘操作会清理用户集群内每台服务器上的脏数据。脏数据限于：

- 已下载到本地但未使用的 Docker 镜像。
- 曾经挂载到容器，但容器销毁后未清理的数据卷（volumn）目录。

操作步骤

1. 登录容器服务管理控制台。

2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择要清理的集群，单击管理， 如下图所示。

4. 在集群管理页面中，单击清理磁盘，如下图所示。

2.17 登录镜像仓库

前提条件

- 您需要准备一个可用的镜像仓库。本例中使用阿里云容器镜像服务，在上面构建了一个可用的仓库。
- 设置仓库的独立登录密码，本示例中使用阿里云容器镜像服务，在 容器镜像服务控制台 上设置与修改 Registry 的登录密码。注意首次修改 Registry 登录密码即是设置密码。

背景信息

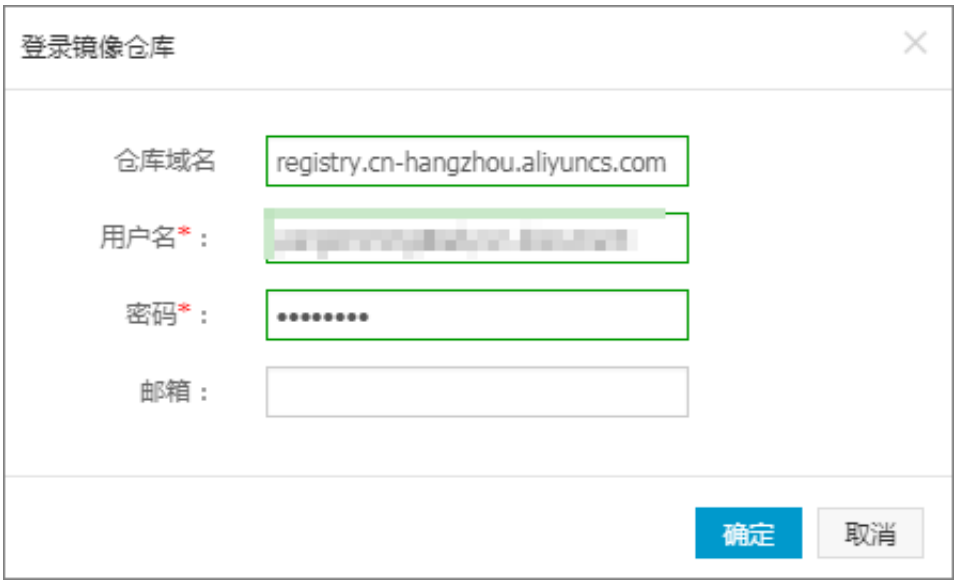
您可以在目标集群上登录到镜像仓库，从而提供相关的集群登录信息，方便您使用集群管理工具进行管理。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择要配置的集群并单击管理。

4. 在集群的详情页面，单击登录镜像仓库。

5. 在弹出的对话框中，对相关参数进行配置。



- 仓库域名：镜像仓库的 hub 域名。以一个镜像地址 `registry.cn-hangzhou.aliyuncs.com/acs/agent:0.8` 为例，`registry.cn-hangzhou.aliyuncs.com` 即是仓库域名。
- 用户名：您当前登录的阿里云账号全称。
- 密码：Registry 的独立登录密码。Registry 的登录密码是在容器镜像服务的控制台上设置与修改的。
- 邮箱：可选。

6. 单击确定后，若无报错信息，说明已经成功登录到镜像仓库。

2.18 升级 Agent

背景信息

> **说明：**
> 升级期间用户应用不受影响，但是无法通过Web界面对集群进行管理操作，也不能用Docker client连接集群的访问端口，时间大约2分钟。

集群内的每一台服务器都会安装容器服务的 Agent，用于接收容器服务控制系统下放的指令。

容器服务会定期的增加新的功能，如果您需要最新的功能，可以升级集群的 Agent。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择要升级 Agent 的集群，单击更多 > 升级 **Agent**。

4. 在弹出的对话框中，单击确定。

2.19 升级 Docker Daemon

背景信息

集群内的每一台服务器都会安装标准的 Docker Daemon 用于管理容器。

> 说明：
>
> - 集群升级需要机器可以公网访问，以便下载升级所需的软件包。
> - 集群升级 Docker 过程中，可能会有升级失败的情况，为了您的数据安全，强烈推荐您选择先打快照然后再升级的方式。
> - 集群升级 Docker 过程中，集群上部署的服务会中断，同时无法进行集群和应用的操作，请您在升级之前安排好相关事宜。升级时间大约3~30分钟， 升级完成后集群会变成运行中状态。

您可以在集群列表页面查看集群的 Docker 版本，如下图所示。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的集群。
3. 选择您要升级 Docker Daemon 的集群，单击 Docker 版本列下的升级 或者单击更多 > 升级**Docker**。

4. 如果目前系统的 Agent 不是最新版本，需要先升级 Agent。单击升级**Agent**，根据提示进行相应的操作即可。

5. 若 Agent 已是最新版本，则可以直接升级 Docker。

您可以使用以下方法之一升级 Docker：

- 直接升级

单击**直接升级**，进入升级 Docker Engine 的流程。

- 备份快照并升级

  建议您通过备份快照升级 Docker（以便于在升级过程中出问题后，可以通过快照进行恢复）。

  单击**备份快照并升级**，此时系统会调用 ECS OpenAPI 对集群内的节点打快照。

  由于备份快照需要一点时间，您需要耐心等待一会。完成快照备份后，系统自动进入升级 Docker Engine 的流程。

  如果备份快照失败，**继续升级** 和 **放弃升级**可用。您可以单击**继续升级**进入升级 Docker Engine 的流程，或者单击**放弃升级**放弃升级 Docker Engine。

后续操作

此时，返回**集群列表**页面。您可以看到刚才操作的集群处于**Docker-Engine**升级中的状态。由于升级 Docker Engine 会进行相应的容器数据备份等工作，所以比较耗时，请耐心等待一会。

2.20 升级系统服务

背景信息

集群的系统服务用来解决应用需要的通用服务，例如日志服务 `acslogging`，路由服务 `acsrouting`，volume 服务 `acsvolumedriver`。下面介绍这些服务的升级操作流程。

> **说明：**
> 集群的系统服务在升级的过程中会导致您的应用或者服务短暂不可访问或者不能正常工作。请谨慎升级。建议选择访问低谷或者维护时间进行升级。

操作步骤

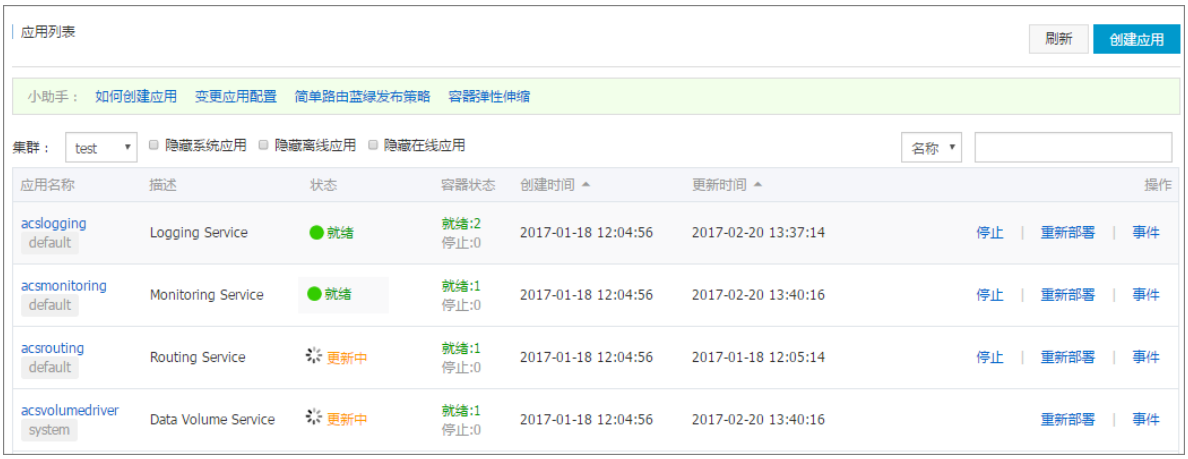
1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的**集群**。
3. 选择您要升级系统服务的集群，单击**更多**并在下拉菜单中单击**升级系统服务**。如下图所示。

4. 在弹出的对话框中，选择要升级的系统服务并单击**升级**，如下图所示。

   例如，本示例中选择的是**路由服务**（对应 acsrouting，注意升级会短暂影响用户应用的访问），**volume 服务**（对应 acsvolumedriver，注意升级可能会短暂影响用户相关联应用的功能）。

此时，单击左侧导航栏中的**应用**，您会发现系统服务正在升级中，如下图所示。

更新完成后，被影响的服务会恢复正常。


应用列表
刷新
创建应用
小助手： 如何创建应用 变更应用配置 简单路由蓝绿发布策略 容器弹性伸缩
集群： test
隐藏系统应用
隐藏离线应用
隐藏在线应用
名称
应用名称 描述 状态 容器状态 创建时间 更新时间 操作
acslogging default Logging Service 就绪 就绪:2 停止:0 2017-01-18 12:04:56 2017-02-20 13:37:14 停止 | 重新部署 | 事件
acsmonitoring default Monitoring Service 就绪 就绪:1 停止:0 2017-01-18 12:04:56 2017-02-20 13:40:16 停止 | 重新部署 | 事件
acsrouting default Routing Service 更新中 就绪:1 停止:0 2017-01-18 12:04:56 2017-01-18 12:05:14 停止 | 重新部署 | 事件
acsvolumedriver system Data Volume Service 更新中 就绪:1 停止:0 2017-01-18 12:04:56 2017-02-20 13:40:16 重新部署 | 事件

3 节点管理

3.1 移除节点

背景信息

您可以移除集群中的节点。移除节点可以将机器从集群中摘除。移除后将不能在节点列表内看到该机器的信息。

> 说明：
>
> - 在移除节点之前请先做好备份工作。
> - 移除节点仅仅是从集群中移除 ECS，并不会释放 ECS。如果需要释放 ECS，需要到 ECS 管理控制台自行释放。

操作步骤

1. 登录容器服务管理控制台。
2. 在 swarm 菜单下，单击左侧导航栏中的节点。
3. 选择要移除的节点所在的集群。
4. 选择要移除的节点，单击更多 > 移除节点。

5. 在弹出的确认对话框中，单击确定。

3.2 重置节点

背景信息

重置节点会替换该机器系统盘，替换后原机器系统盘数据会丢失，重置后的机器会重新加入到集群中。

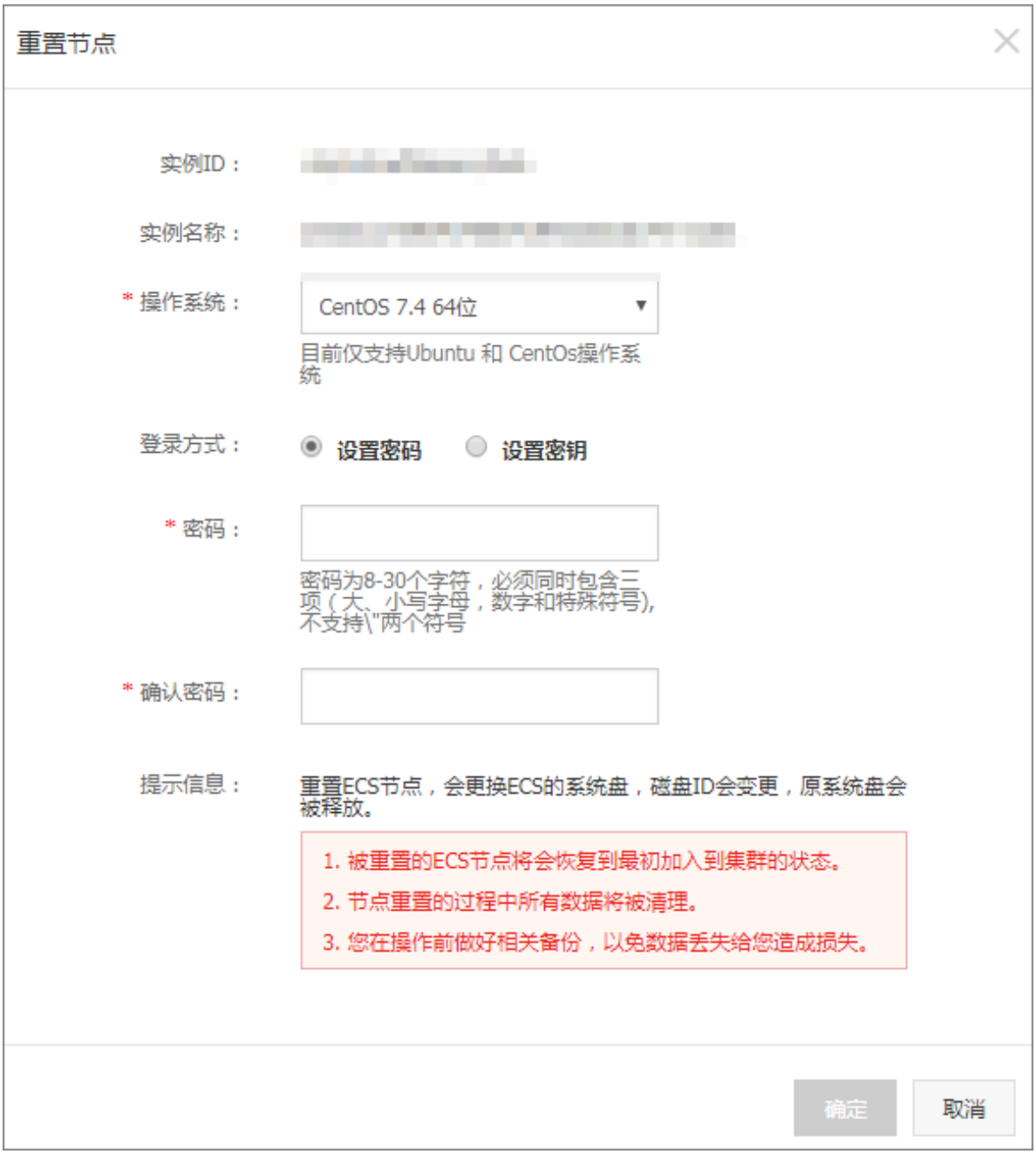
> 说明：

- 重置 ECS 节点，会更换 ECS 的系统盘，磁盘 ID 会变更，原系统盘会被释放。
- 被重置的 ECS 节点将会恢复到最初加入到集群的状态。
- 节点重置的过程中所有数据将被清理。
- 您在操作前做好相关备份，以免数据丢失给您造成损失。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的节点。
3. 选择要重置的节点所在的集群。
4. 选择要重置的节点，单击更多 > 重置节点。

5. 在弹出的确认对话框中，您可重置节点的操作系统，填写该实例的登录密码并单击确定。
   - 重置操作系统：目前仅支持Ubuntu和CentOS操作系统。
   - 重置登录方式：支持密码和密钥对的登录方式。



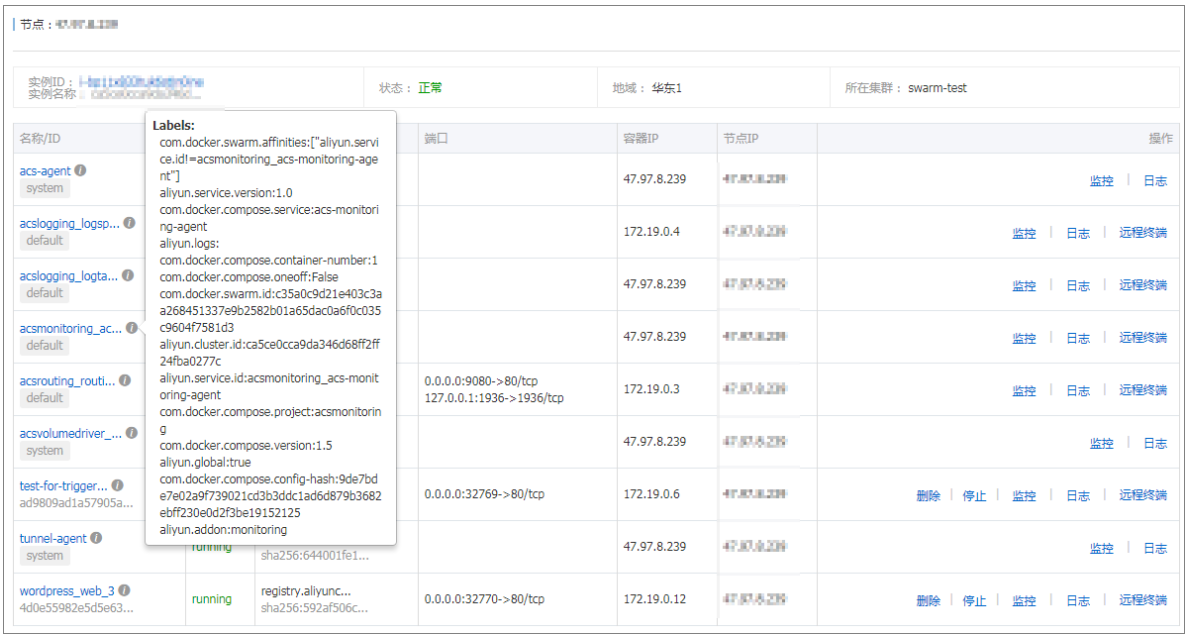
3.3 查看节点上运行的容器

您可以通过节点列表页面查看运行在某个节点上的容器。

操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的节点。
3. 选择节点所在的集群。
4. 选择所需的节点，单击节点的 ID。

您可以看到运行在该节点上的容器的列表。



后续操作

通过该列表，您可以查看容器的标签（Labels），查看容器的镜像并查看镜像的 sha256，查看容器的日志和监控信息，并进行容器相关操作（包括：启停容器、删除容器、通过远程终端操作容器）。

3.4 更新节点证书

前提条件

1. 您已成功创建一个Swarm集群，参见创建集群。
2. 更新节点证书会重启节点Docker Daemon，请确保节点容器均已设置自动重启，

> 说明：
> 创建应用时，可设置容器的重启策略。使用镜像创建应用时，勾选**restart**为**always**；使用编排模板创建应用时，在编排模板中配置一条容器的重启策略`restart: always`。

3. 若节点证书将在60天内过期，会进行提示，节点有必要及时更新节点证书。

背景信息

每个集群节点上都有用于访问系统管控服务的证书，默认证书签发都会有有效期限，当证书有效期限即将到期时，我们需要重新手动续签节点证书，否则会影响该节点的服务。

操作步骤

1. 登录容器服务管理控制台。
2. 在Swarm菜单下，单击左侧导航栏中的节点，在集群列表框中选择目标集群，则可以看到集群节点证书过期信息。

> **说明：**
> 只有节点证书将在60天内过期，节点状态列才会展示证书到期时间。

3. 在节点列表中选择所需节点，单击右侧更多 > 更新证书，即可重新签发本节点证书。

> **说明：**
> 建议在更新节点证书前，先将集群Agent升级最新版本。

4. （可选）若单击更新证书后，提示需升级集群Agent，则说明当前集群Agent并不支持该功能，我们需要先升级集群 Agent到新版本，可参考升级 *Agent*。若未提示，则直接进入下一步。

5. 若未提示，或者成功升级集群Agent后，此时单击更新证书，确认后则可进行节点证书更新操作。

> 说明：
>
> - 节点证书更新完成后，大约1分钟后会自动重启节点Docker Daemon。
> - 为保证节点容器能自动拉起，请确保设置了自动重启策略。

6. 成功更新完集群节点证书后，可以看到节点证书信息已不再展示。

4 服务编排

4.1 简介

容器服务支持 *Docker Compose* 编排模板来描述多容器应用。

编排模板允许您描述一个完整的应用，该应用可以由许多个服务组成。例如：一个门户网站应用，由一个 Nginx 服务、一个 Web 服务和一个数据库服务组成。

一个服务可能会有多个容器实例，所有容器实例的配置保持一致。例如：上述应用中的 Web 服务，就可以根据访问量需要启动两个甚至更多的容器。

能力

容器服务支持通过编排模板文件，自动化地部署和管理一个应用。

编排模板文件使用的标签兼容大部分 *Docker Compose* V1 和 V2 版本实现的标签。有关具体兼容的标签，参见标签概览。

编排模板文件也支持 Compose V1 和 V2 两种版本的模板格式。更多详细信息，参见 *Docker Compose V1* 和 *Docker Compose V2*。

容器服务也在社区版本之上提供了很多扩展能力：

- 与社区的 Docker Compose 和 Swarm 不同，阿里云容器服务支持跨节点的容器连接（link），所以您可以直接将 Docker Compose 模板描述的应用部署到分布式集群上来提供高可用性和可伸缩性。
- 容器服务也在社区 Compose 模板描述的基础上提供了一系列扩展来简化 Web、微服务应用的部署和运维。更多详细信息，参见标签概览。

示例

下面是一个 WordPress 应用，包含了由 WordPress 镜像提供的 Web 服务和 MySQL 镜像提供的 db 服务。

```
web:
  image: wordpress:4.2
  ports:
    - "80"
  environment:
    - WORDPRESS_AUTH_KEY=changeme
    - WORDPRESS_SECURE_AUTH_KEY=changeme
    - WORDPRESS_LOGGED_IN_KEY=changeme
    - WORDPRESS_NONCE_KEY=changeme
```

```
    - WORDPRESS_AUTH_SALT=changeme
    - WORDPRESS_SECURE_AUTH_SALT=changeme
    - WORDPRESS_LOGGED_IN_SALT=changeme
    - WORDPRESS_NONCE_SALT=changeme
  restart: always
  links:
    - db:mysql
  labels:
    aliyun.log_store_wordpress: stdout
    aliyun.probe.url: http://container/license.txt
    aliyun.probe.initial_delay_seconds: "10"
    aliyun.routing.port_80: wordpress;http://www.example.com;https://
www.nice.com
    aliyun.scale: "3"
db:
  image: mysql:5.6
  environment:
    MYSQL_ROOT_PASSWORD: password
  restart: always
  labels:
    aliyun.log_store_mysql: stdout
```

4.2 标签概览

容器服务编排模板文件使用的标签兼容大部分 *Docker Compose* V1 和 V2 版本实现的标签，并在社区版本的基础上提供了很多扩展能力。

扩展能力的标签

容器服务扩展了编排模板的部署和生命周期管理能力，所有扩展能力都被描述在 `labels` 标签下面，作为子标签使用。

标签	说明
probe	设置服务的健康性检查。
rolling_updates	设置服务滚动更新。
parallelism	设置 `rolling_updates` 每次并行更新的容器数量。注意：此标签必须和 `rolling_updates` 配合使用，单独使用无效。
depends	设置服务的依赖关系。
scale	设置该服务的容器数量，横向扩展服务。
routing	设置该服务的访问域名。
routing.session_sticky	设置 routing 在做请求路由的时候，是否保持 session sticky，即会话保持。注意：此标签必须和 `routing` 配合使用，单独使用无效。

标签	说明
lb	通过自定义阿里云负载均衡 nat 映射的方式来暴露服务端口到公网或者内网。
日志	和阿里云日志服务集成，采集容器日志并且发送到阿里云日志服务。
global	设置该服务为全局服务。

功能增强的标签

容器服务提供服务部署约束*#affinity:service#* 标签用来设置该服务的部署约束条件。

额外支持的标签

标签	说明
external	设置该服务直接链接到外部地址。
dns_options	设置 DNS 选项，和`docker run` 命令中的**`--dns-opt`**参数语义一致。
oom_kill_disable	设置是否禁止 OOM Killer, 和`docker run` 命令中的**`--oom-kill-disable`** 参数语义一致。

变量替换

容器服务支持参数化的 Docker Compose 模板。模板中可以包含环境变量作为参数，当模板部署时会提示输入参数值，并在部署时对模板进行变量替换。

更多详细信息，参见变量替换。

容器重新调度

容器服务支持对 Docker 容器的重新调度：当一个节点失效时，容器可以被自动调度到其他可用节点自动运行。

更多详细信息，参见容器重新调度。

高可用性调度

为了使应用有更高的可用性，容器服务支持将同一个服务的容器调度在不同的可用区（availability zone）里。当某个可用区故障时，应用依然能够提供服务。

更多详细信息，参见高可用性调度。

不支持的 **Docker Compose** 标签

容器服务暂不支持 Docker Compose 的部分标签。有关容器服务暂不支持的标签，参见不支持的 *Docker Compose* 标签。

4.3 gpu

申请 GPU 资源，将容器调度到满足可用 GPU 资源个数的机器上并将 GPU 资源分配给容器。

标签格式：

`aliyun.gpu: "1"`

`aliyun.gpu` 指定申请的 GPU 资源的个数。容器服务调度器会寻找满足可用 GPU 资源个数的机器，将容器部署到该机器上，将 GPU 资源分配给容器并将主机上的 GPU 卡映射到容器内。容器所分配到的 GPU 资源对于您是透明的。具体来说：

例如，如果您申请了一个 GPU 资源，主机上只有一个 */dev/nvidia1*可用，容器服务会将主机上的 */dev/nvidia1*映射为容器里的 */dev/nvidia0*。这样会让您的程序和具体的设备号解耦。

示例：

```
serving:
    image: inception-serving:gpu
    labels:
       aliyun.gpu: "1"
```

4.4 probe

设置服务的健康性检查。

- 通过 URL 进行检查，支持 HTTP 协议、TCP 协议。
- 通过 shell 脚本检查。

健康检查会从容器宿主机上发起，每隔一定时间（默认两秒）向容器发起请求或在容器上执行 shell 脚本命令。

检查成功的判断条件为：HTTP 请求的返回码为 2XX/3XX；TCP 端口可建立连接；shell 脚本运行返回值为 0。

检查的字段解释：

- `aliyun.probe.url`：HTTP、TCP 请求的 URL。请注意您不需要填写自己的域名或者 IP 地址，只需要加上 `container` 这个单词，该 URL 最终会被解析成容器相应的 IP 去进行健康检查，检查结果返回 2XX 或者 3XX 才认为服务是健康的。
  - 例如，容器通过 8080 端口提供 HTTP 服务，并提供了 `/ping` 作为健康检查的 URL，则探测 URL 的格式为 `http://container:8080/ping`，容器服务会自动通过 HTTP GET 请求检查 URL 的返回结果，如果返回结果的返回码为 2XX 或 3XX，则说明健康检查成功。
  - 例如，MySQL 容器侦听 3306 端口，探测 URL 的格式为 `tcp://container:3306`，服务会检查容器 3306 端口是否打开，如果打开则说明健康检查成功。
- `aliyun.probe.cmd`：健康检查执行的检查 Shell 命令，`/check.sh`；容器服务会定期在容器内执行该命令，当 shell 脚本返回值为 0 时表明健康检查成功。
- `aliyun.probe.timeout_seconds`：健康检查的超时时间。
- `aliyun.probe.initial_delay_seconds`：在容器启动后延迟几秒开始健康检查。

 说明：

- 一个服务中只能包含 `aliyun.probe.url` 和 `aliyun.probe.cmd` 其中之一。
- 如果服务不包含 `aliyun.probe.url` 或 `aliyun.probe.cmd`，则容器缺省为健康状态，且其他 `aliyun.probe.xxx`标签会被忽略。

示例：

利用 URL 检测容器健康状态。

```
os:
  image: my_nginx
  labels:
    aliyun.probe.url: http://container/ping
    aliyun.probe.timeout_seconds: "10"
    aliyun.probe.initial_delay_seconds: "3"
```

利用 shell 脚本检测容器健康状态。

```
os:
  image: my_app
  labels:
    aliyun.probe.cmd: health_check.sh
```

```
    aliyun.probe.initial_delay_seconds: "3"
```

4.5 rolling_updates

更新某个服务时，如果该服务包括超过一个以上容器（使用 `scale` 标签定义），在第 n 个容器更新成功后，再去做第 n+1 个容器的更新，以此来最小化停止服务时间。

示例：

部署 WordPress 服务，通过`scale`标签指定部署 2 个容器，使用`rolling_updates`标签可以使 WordPress 对外停止服务的时间最小化。

```
web:
  image: wordpress
  ports:
    - 80
  restart: always
  links:
    - 'db:mysql'
  labels:
    aliyun.logs: /var/log
    aliyun.routing.port_80: http://wordpress
    aliyun.rolling_updates: 'true'
    aliyun.scale: '2'
db:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: example
  restart: always
  labels:
    aliyun.logs: /var/log/mysql
```

parallelism

您可以使用 `parallelism` 标签定义 `rolling_updates` 每次并行更新的容器数量。

 说明：

此标签必须和 `rolling_update` 配合使用，单独使用无效。

取值：

- 默认值为 1，即每次只更新一个容器。
- 当其值大于 1 的时候，`rolling_updates` 过程中，每次会以 `parallelism` 定义的值来并行更新相应个数的容器，实现批量更新。
- 当定义值无效时，默认为 1。

 说明：

为了确保始终有容器在提供服务，建议 parallelism 定义的值小于服务包含的容器数。

示例：

下面的示例部署 Nginx 服务，通过 scale 标签部署 4 个容器，使用 rolling_updates 和 parallelism 标签定义每次以 2 个容器为单位来进行批量更新。

```
web:
  image: nginx:latest
  restart: always
  environment:
      - "reschedule:on-node-failure"
  ports:
      - 80
  labels:
    aliyun.scale: "4"
    aliyun.rolling_updates: 'true'
    aliyun.rolling_updates.parallelism: "2"
```

4.6 depends

设置服务的依赖关系。

设置之后，容器服务可以控制容器的启动顺序，一个接一个的启动容器。

示例:

 说明：

多个依赖使用逗号 (,) 分隔。

```
web:
  image: wordpress:4.2
  ports:
    - 80
  links:
    - db:mysql
  labels:
    aliyun.depends: db,redis
db:
  image: mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
redis:
  image: redis
```

4.7 scale

设置该服务的容器数量，横向扩展服务。

目前，Docker Compose 只能在每一个服务中启动一个容器，如果需要扩展容器数量，需要在启动后手动进行设置。

现在通过scale的扩展标签，支持您在容器启动的时候进行扩展。

此外，在容器被删除之后，您可以在容器服务管理控制台对应用进行重新部署（单击左侧导航栏中的应用，选择目标应用并单击右侧的重新部署），容器服务会重启或新建容器使容器恢复到指定数量。

示例：

```
web:
  image: wordpress:4.2
  ports:
    - 80
  links:
    - db:mysql
  labels:
    aliyun.scale: "3"
db:
  image: mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
```

4.8 routing

设置该服务的访问域名。

格式：

```
aliyun.routing.port_$container_port: [http://]$domain|$domain_prefix[:
$context_path]
```

名词解释：

- $container_port: 容器端口，注意 该处不是主机的端口。
- $domain: 域名，需要用户填写自己的域名。
- $domain_prefix: 域名前缀，如果填写域名前缀，容器服务会提供给您一个测试用的域名，域名后缀是.<cluster_id>.<region_id>.alicontainer.com。
- $context_path: 请求的服务路径，即可以根据请求的路径来选择区分不同的服务。

绑定域名的选择：

- 如果使用 HTTP 协议暴露服务，可以使用容器服务提供内部域名（顶级域为 alicontainer.com），供您测试使用，也可以使用您提供的域名。

- 如果使用 HTTPS 协议，那么仅支持配置您提供的域名，例如 `www.example.com`。您需要修改 DNS 设置将域名指定到容器集群提供的负载均衡服务上。

标签声明的格式要求：

- 容器服务为每一个集群分配了子域名，绑定内部域名只需要给出域名的前缀，域名前缀仅表示域名的一级，不能使用点号 (.) 进行分隔。
- 如果您不指定 `scheme`，则默认使用 HTTP 协议。
- 域名的长度不能超过 128 个字符，context root 的长度不能超过 128 个字符。
- 绑定多个域名到服务时，域名之间用分号 (;) 隔开。
- 一个后端服务可以有多个端口，该端口指的是容器暴露的端口，一个端口只能使用一条 label 进行声明，带有多个端口的服务需要声明多个 label。

示例：

使用 routing 标签。

将容器服务提供的内部域名 `wordpress.<cluster_id>.<region_id>.alicontainer.com` 绑定到 Web 服务的 80 端口，并且将您提供的自有域名 `http://wp.sample.com/context` 绑定到 Web 服务的 80 端口。

```
web:
  image: wordpress:4.2
  links:
    - db:mysql
  labels:
    aliyun.routing.port_80: wordpress;http://wp.sample.com/context
db:
  image: mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
```

最终您得到的内部域名为 `wordpress.cd3dfe269056e4543acbec5e19b01c074.cn-beijing.alicontainer.com`。

Web 服务运行之后，您可以通过 `http://wordpress.cd3dfe269056e4543acbec5e19b01c074.cn-beijing.alicontainer.com` 或者 `http://wp.sample.com/context` 访问相应的 Web 服务。

如果您需要支持 HTTPS 服务，需要自行通过阿里云官网负载均衡管理控制台上传 HTTPS 证书，并绑定相应的集群对外访问负载均衡端点。

routing.session_sticky

设置 routing 在做请求路由的时候，是否保持 session sticky，即会话保持。其效果是，在某个会话时间内，请求一直路由到同一个后端的容器，而不是每次请求都随机路由到不同的容器。

说明：

- 只有当您已经设置了 `aliyun.routing.port_$container_port` 时，该设置才能起作用。
- 简单路由会话保持基于Cookie机制，默认Cookie最大过期时间8h，空闲过期时间30m。
- 简单路由默认已开启会话保持机制。

其设置方法如下：

- 开启会话保持

  ```
  aliyun.routing.session_sticky: true
  ```

- 关闭会话保持

  ```
  aliyun.routing.session_sticky: false
  ```

模板编排文件示例：

```
web:
  image: wordpress:4.2
  links:
    - db:mysql
  labels:
    aliyun.routing.port_80: wordpress;http://wp.sample.com/context
    aliyun.routing.session_sticky: true
db:
  image: mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
```

4.9 lb

通过自定义阿里云负载均衡 nat 映射的方式来暴露服务端口到公网或者到内网。需要升级到最新版本的 Agent 方能支持该扩展能力标签。

标签格式如下，带$的变量为占位符。

```
aliyun.lb.port_$container_port:$scheme://$[slb_name|slb_id]:$
slb_front_port
```

示例

```
web:
  image: wordpress:4.2
```

```
  ports:
    - 7777:80
    - 9999:9999
    - 8080:8080
    - 53:53/udp
  links:
    - db:mysql
  labels:
    aliyun.lb.port_80: http://slb_example_name:8080
    aliyun.lb.port_9999: tcp://slb_example_name:9999
    aliyun.lb.port_8080: https://14a7ba06d3b-cn-hangzhou-dg-a01:80
    aliyun.lb.port_53: udp://14a7ba06d3b-cn-hangzhou-dg-a01:53
db:
  image: mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
```

要使用好自定义负载均衡的 `lb` 标签，您需要理解请求路由过程中的 3 个端口，即负载均衡的前端端口，负载均衡的后端端口（也就是 ECS vm 的端口），最后就是容器的端口。以第一个`lb`标签`aliyun.lb.port_80`为例，从左往右看，在 key 中的 80 端口指的是容器要暴露的端口，后面的 8080 端口指的是负载均衡要暴露的前端端口。负载均衡的后端端口是 ECS 实例的端口，可从标签 `ports` 的主机:容器端口映射中获取，由此，您可以查到容器端口 80 对应的主机端口是 7777，因此确定了负载均衡转发的后端端口是 7777 端口。因此第一个标签说明了当向服务 Web 发起请求时，首先通过负载均衡前端的 8080 端口进入，转发到后端 ECS 实例的 7777 端口，然后再根据端口映射 `ports` 的声明，请求最终从容器端口 80 进入，交由容器内的 WordPress 进程提供服务。接下来的标签以此进行相同的解释。该标签配置的负载均衡均不经过集群内置的 routing 服务，请求的路由由您自己控制。

标签声明的格式要求

- 指明负载均衡实例时，可以使用负载均衡实例的名称或者负载均衡实例的 ID。
- 负载均衡实例名称的限制为 1~80 个字符，允许包含字母、数字、连字符（-）、正斜杠（/）、点号（.）、下划线（_）。
- 容器端口限制为 1~65535。
- 负载均衡前端端口的限制为 1~65535。

带有自定义负载均衡 nat 映射的服务部署限制

- 您需要自己创建负载均衡实例，对负载均衡实例命名，并创建对应监听端口，然后以扩展标签的方式提供映射的容器端口 `$container_port`，使用的协议 `$scheme`（可能的值有 `tcp`、`http`、`https`、`udp`，负载均衡实例的名称 `$slb_name` 或者 `$slb_id`，以及指定负载均衡实例的前端端口 `$slb_front_port`。

- 您必须指定服务要暴露端口的主机和容器端口的映射，通过 Dockerfile 标准的标签 `ports` 指定，注意必须指定主机端口，且与其他服务映射的主机端口不能冲突，需要主机的端口用于负载均衡绑定后端的 ECS 实例。
- 一个服务只能使用一个或者多个负载均衡实例进行服务端口的暴露，因多个服务会分布在不同的 ECS 实例后端，多个服务不能共享使用同一个负载均衡实例。
- 通过 lb 标签来配置使用负载均衡路由时，不能配置为集群默认的负载均衡实例。
- 部署了带有负载均衡 nat 映射的服务的主机使用相同的主机：容器端口映射，因此这些服务在每台 ECS 上只有一个实例。
- 支持的负载均衡协议 `$scheme` 包括 tcp、http、https、udp 协议。
- 您需要自行在阿里云负载均衡管理控制台创建监听的端口。
- 请自行登录负载均衡管理控制台对在容器服务中使用的负载均衡实例进行具体的配置修改，例如带宽限制等配置。
- `lb` 标签的价值在于您不需要自行绑定负载均衡后端的 ECS 实例，只需要配置好相应的标签，就会自动帮助您完成绑定后端的操作。因此，除了绑定负载均衡后端的操作，您对负载均衡的设置和修改需要自行在阿里云负载均衡管理控制台上完成。
- 容器服务会帮助您生成一个 RAM 子账户(需要您开通 RAM)，使用这个具有部分负载均衡权限（没有创建和删除负载均衡的权限）的账号帮助您管理在容器服务中使用的负载均衡实例，例如绑定集群中某些节点作为服务的后端。
- 在服务的整个生命周期内，`lb`标签会一直生效，除非服务被删除，或者`lb`标签删除之后重新部署了服务，在此期间，配置在`lb`标签内的 SLB 实例不能混用。

4.10 日志

和阿里云日志服务集成，采集容器日志并且发送到阿里云日志服务。

示例

```
mysql:
    image: mysql
    ports:
        - 80
    labels:
        aliyun.scale: "1"
    environment:
        - MYSQL_ROOT_PASSWORD=password
wordpress:
    image: registry.aliyuncs.com/jiangjizhong/wordpress
    ports:
        - 80
    labels:
```

```
        aliyun.routing.port_80: wordpress-with-log
        aliyun.log_store_dbstdout: stdout   #注意这里
    links:
        - mysql
```

更多详细信息，参见集成日志服务。

4.11 global

设置该服务为全局服务。

有一些服务需要在每一个节点部署，例如监控或是日志类的服务。并且在新的节点建立的时候就对这个节点进行服务的部署。

当一个服务被设置为 `global` 时，该服务会在集群中的每一个节点进行部署。当集群中有新增节点时，也会自动部署一个容器实例到新节点之上。

```
monitor:
  image: sample
  labels:
    aliyun.global: true
```

4.12 服务部署约束（affinity:service）

设置服务的部署约束条件。

容器服务支持 Docker Swarm 兼容的容器部署约束条件，您可以通过 *Docker Swarm Filter* 控制一个容器的部署。

但是在社区版 Docker Compose 中，却并没有相关的能力来控制服务直接的部署约束。

在容器服务中，您可以在 `environment` 中添加相关 `affinity:service`，来约束服务之间的亲和度（`Affinity`），达到控制服务部署策略的功能。支持服务之间的 `Soft affinity` 和 `Hard affinity`。

示例：

本示例中，`web` 服务设置了 `affinity:service!=db` 的部署约束。使得 `web` 服务一定会选择没有部署 `db` 服务的节点，这样当一个节点失效时，可提高服务可用性。当您的集群只有一个节点的时候，由于指定的是 hard anti-affinity，该部署会失败，因为部署没有办法满足所指定的强约束条件。

```
web:
  image: registry.aliyuncs.com/acs-sample/wordpress:4.5
  ports:
    - '80'
```

```
  environment:
    - affinity:service!=db
  restart: always
  links:
    - 'db:mysql'
  labels:
    aliyun.logs: /var/log
    aliyun.probe.url: http://container/license.txt
    aliyun.probe.initial_delay_seconds: '10'
    aliyun.routing.port_80: http://wordpress
    aliyun.scale: '2'
db:
  image: registry.aliyuncs.com/acs-sample/mysql:5.7
  environment:
    MYSQL_ROOT_PASSWORD: password
  restart: always
  labels:
    aliyun.logs: /var/log/mysql
```

4.13 external

设置该服务直接链接到外部地址。

扩展字段下有以下字段可以使用：

- `host`：设置链接的域名。
- `ports`：设置链接的端口。

示例:

不使用 `external`，直接启动一个 MySQL 容器。

```
web:
  image: wordpress:4.2
  ports:
    - 80
  links:
    - db:mysql
db:
  image: 10.32.161.160:5000/mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
```

通过 `external`，描述一个并没有部署在集群中的 RDS 服务，并提供给部署在集群中的 WordPress 使用。

```
wordpress:
  image: wordpress:4.2
  ports:
    - 80
  links:
    - db:mysql
  environment:
    - WORDPRESS_DB_USER=cloud
    - WORDPRESS_DB_PASSWORD=MYPASSWORD
```

```
    - WORDPRESS_DB_NAME=wordpress
db:
  external:
    host: rdsxxxx.mysql.rds.aliyuncs.com
    ports:
      - 3306
```

4.14 dns_options

设置 DNS 选项，和`docker run`命令中的 **`--dns-opt`** 参数语义一致。

```
wordpress:
  image: wordpress:4.2
  dns_options:
    - "use-vc"
```

4.15 oom_kill_disable

设置是否禁止 OOM Killer，和`docker run`命令中的**`--oom-kill-disable`**参数语义一致。

```
wordpress:
  image: wordpress:4.2
  oom-kill-disable: true
```

4.16 变量替换

容器服务支持参数化的 Docker Compose 模板。模板中可以包含环境变量作为参数，当模板部署时会提示输入参数值，并在部署时对模板进行变量替换。

比如，您可以定义参数 **`POSTGRES_VERSION`**。

```
db:
  image: "postgres:${POSTGRES_VERSION}"
```

当部署上面的 Compose 模板的时候，容器服务会提示您输入**`POSTGRES_VERSION`**参数值，比如9.3。容器服务会根据参数值对 Compose 模板进行变量替换。在本示例中，会部署一个`postgres:9.3`的容器。

容器服务完全兼容 Docker Compose 的语法，可以在模板中使用`$VARIABLE`或者`${VARIABLE}`格式的语法。

在 Compose 模板中可以使用`$$`来对需要包含`$`的字符串进行转义，这样容器服务不会错误地将其作为参数来进行处理。

关于 Compose 模板支持变量替换的详细信息，参见 *Variable substitution*。

4.17 容器重新调度

容器服务支持对 Docker 容器的重新调度：当一个节点失效时，容器可以被自动调度到其他可用节点自动运行。

缺省情况下，容器的重新调度策略是关闭的。根据需要，您可以用如下配置来让重调度策略生效。

容器服务提供兼容 Docker Swarm 的容器重新调度策略，可以通过环境变量方式或者 label 方式启动。

环境变量：

```
redis:
  image: redis
  environment:
    - reschedule:on-node-failure
```

Label：

```
web:
  image: nginx
  restart: always
  environment:
    - aaaaa=aaaaa
  labels:
    aliyun.scale: "3"
    com.docker.swarm.reschedule-policies: "[\"on-node-failure\"]"
```

说明：

如果重新调度容器之后，需要恢复 Docker 容器所需的持久化状态，需要配合支持数据迁移或共享的 Docker 文件卷。

4.18 高可用性调度

为了使应用有更高的可用性，容器服务支持将同一个服务的容器调度在不同的可用区（zone）里。当某个可用区故障时，应用依然能够提供服务。

您可以在编排文件中通过环境变量指定对可用区的选择，有以下两种格式。

- availability:az==3

  服务至少分布在 3 个可用区中；如果当前集群没有 3 个可用区，或机器资源不够导致无法分布在 3 个可用区，容器创建会失败。

- availability:az==~3

  服务尽可能分布在 3 个可用区中；无法满足时依然可以成功创建。

在下面的示例中，服务至少要部署在两个可用区中。

```
nnn:
  expose:
    - 443/tcp
    - 80/tcp
  image: 'nginx:latest'
  environment:
    - 'availability:az==2'
  labels:
    aliyun.scale: '8'
  restart: always
  volumes:
    - /var/cache/nginx
```

4.19 不支持的 Docker Compose 标签

标签	说明
build	`build` 标签用于使用当前目录中的 Dockerfile 文件和其他文档进行容器镜像构建。目前容器服务暂不提供构建镜像功能，推荐您将构建和部署的动作分开处理：您可以利用阿里云的容器镜像服务直接从代码源构建镜像，或者将本地构建的镜像推送到镜像仓库；您可以在编排模板中使用 `image` 标签引用镜像仓库（包括私有仓库）中的镜像。
dockerfile	同 `build` 标签。
env_file	容器服务暂不支持以文件方式指定环境变量，您可以通过 `environment` 标签添加环境变量。
mac_address	暂时不支持 Mac 地址的设置。
detach	容器服务的所有镜像都是以 detach 模式启动的，不允许您指定 attach 方式执行。
stdin_open	同 `detach` 标签。
tty	同 `detach` 标签。
extends	不支持。
networks	Compose version 2 中的网络允许服务的容器启动在自定义的网络中，容器服务的容器都是在同一个跨主机互通的容器网络，所以不支持您在 Compose version 2 中使用 `networks` 标签。

标签	说明
	关于容器服务的网络管理和服务发现，参见 跨主机互联的容器网络。

5 服务发现和负载均衡

5.1 概述

服务发现和负载均衡主要解决通信的可靠性问题。为了达到可靠性，容器服务引入了负载均衡机制。通信又可以分为对外暴露服务的通信和内部服务之间的通信。下面根据场景引导您使用不同的解决方案。

场景一

普通且简单的 7 层协议负载均衡，Web 服务的反向代理，推荐使用简单路由服务。更多详细信息，参见简单路由#支持 *HTTP/HTTPS)*，简单路由-域名配置，简单路由-*HTTP* 协议变为 *HTTPS* 协议。

场景二

4 层协议的负载均衡，负载均衡直接负载均衡到多个相同功能的容器，在将传统架构迁移到容器架构过程中非容器集群的服务访问容器集群中容器的服务，推荐使用负载均衡路由。

场景三

同一个集群内，服务间需要相互发现和相互进行通信，且需要负载均衡的能力，推荐使用集群内服务间路由和负载均衡。

场景四

同一个集群内，服务间需要相互发现和相互进行通信，但是不需要负载均衡的能力，推荐使用容器间的互相发现。

场景五

对负载均衡和服务发现有较高的定制需求，例如需要支持泛域名，自定义错误页面，支持记录访问日志，URL 参数值选择后端服务，自定义 HAProxy 配置文件等等，推荐使用自定义路由-使用手册。更多详细信息，参见自定义路由-简单示例。

5.2 简单路由-域名配置

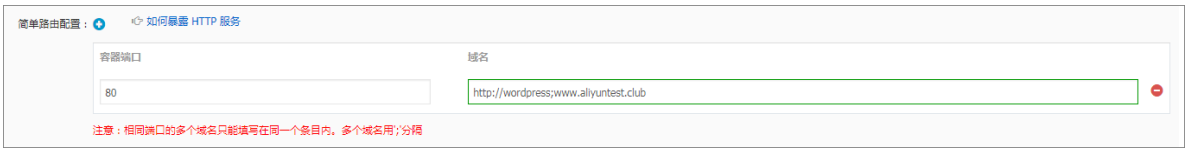
操作步骤

1. 登录容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的服务。

3. 选择要添加域名的服务所在的集群。
4. 选择要添加域名的服务（本示例中要添加域名的服务为 web，所属的应用为 wordpress）并单击**变更配置**。如下图所示。

5. 单击**简单路由配置**右侧的加号图标，输入要添加的域名（本示例中要添加的域名为 `www.aliyuntest.club`）并单击**确定**更新配置。如下图所示。

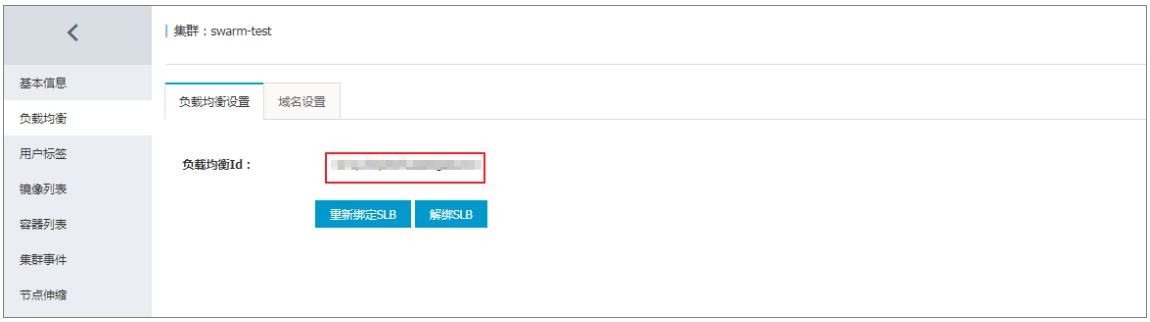
> **说明：**
> 同一个服务同一个端口的多个域名只能写在同一个条目内，并且域名和域名之间用分号（;）分隔。



此时，服务处于更新中。更新完毕变成就绪状态后，路由服务 `acsrouting_routing` 就已经将该域名配置好了。当有请求以域名 `www.aliyuntest.club` 访问服务 `wordpress-rds_web` 时，就能正确的解析并转发到相应的服务了。

6. 将域名解析到容器服务的集群上。容器服务在创建集群的时候，会给每一个集群分配一个负载均衡实例，该负载均衡实例是属于您自己的。
   a) 登录容器服务管理控制台。
   b) 在 Swarm 菜单下，单击左侧导航栏中的**集群**。
   c) 选择相应的集群，本示例为 `swarm-test` 并单击**管理**。

   d) 单击**负载均衡**，并查看负载均衡 ID。



7. 登录负载均衡控制台，在实例列表中找到目标实例 ID，进入实例详情页面，您可以查看负载均衡实例的服务地址。

8. 登录阿里云云解析 DNS 服务管理控制台，添加域名解析（本示例中为 www.aliyuntest.club）。

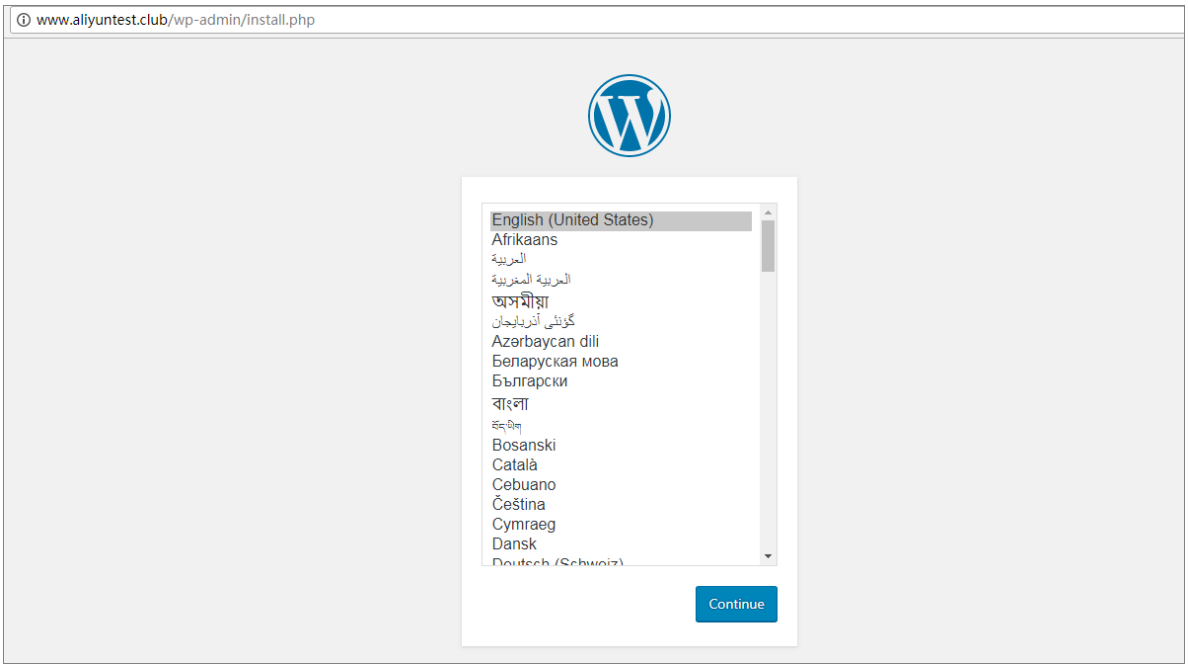
a) 添加域名。若已有域名，跳过此步。

b) 添加域名解析。

- 记录类型为 A
- 主机记录为 www。主机记录即是域名前缀，您也可以选择其他前缀。
- 解析线路为默认。
- 输入绑定的负载均衡实例的服务地址。
- 设置 TTL 值。

9. 重新部署 wordpress 应用，然后进入应用的路由列表，发现域名解析已经生效。

10. 访问页面 `www.aliyuntest.club`。



5.3 简单路由-HTTP 协议变为 HTTPS 协议

前提条件

如果您还没有配置成功 HTTP 协议的域名访问，请先了解配置 HTTP 的域名访问。更多详细信息，参见简单路由-域名配置。

操作步骤

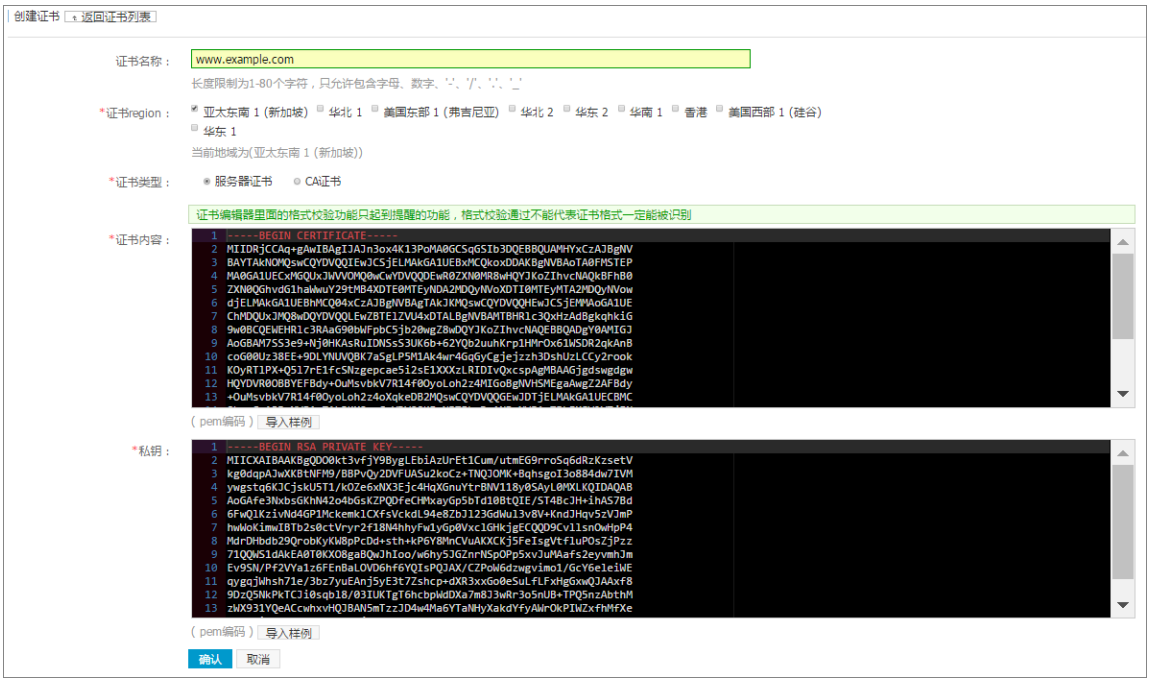
1. HTTPS 协议是在负载均衡这一层进行支持的。为了支持 HTTPS 协议，您需要创建负载均衡证书。

   a) 登录 负载均衡管理控制台。

   b) 单击左侧导航栏中的证书管理并单击页面右上角的创建证书。

   c) 输入证书的相关信息。

   更多详细信息，参见证书要求 和生成证书 相关的文档，如下图所示。



2. 证书创建成功后，找到创建集群时分配的负载均衡实例。

   容器服务在创建集群的时候，给每一个集群分配了一个负载均衡实例，该负载均衡实例是属于您自己的。

   a) 登录容器服务管理控制台。

   b) 单击左侧导航栏中的集群，选择相应的集群，本示例中为routing-test-online，单击管理。

c) 单击**负载均衡** ，并查看负载均衡 ID。

前往**产品与服务**中查找，并进入负载均衡管理控制台。根据 ID 查看对应负载均衡实例的服务地址。

3. 单击左侧导航栏中的**监听**并单击**添加监听**。在添加监听页面，填写端口信息，如下所示。

	协议	端口
前端协议(端口)	HTTPS	443
后端协议(端口)	HTTP	9080

a) 前端协议，选择 `HTTPS`。

b) 端口使用 443 端口，后端端口使用 9080 端口（该端口为路由服务**acsrouting_routing** 在每一台 ECS 主机上暴露的端口，所有的 HTTP 请求会在路由服务 **acsrouting_routing**上根据 HTTP 协议的 `HOST` header 转发到相应的提供各种服务的容器内）。

c) 选择前面步骤创建的证书 `www.example.com`。

d) 根据需要设置其它选项。

e) 单击下一步。

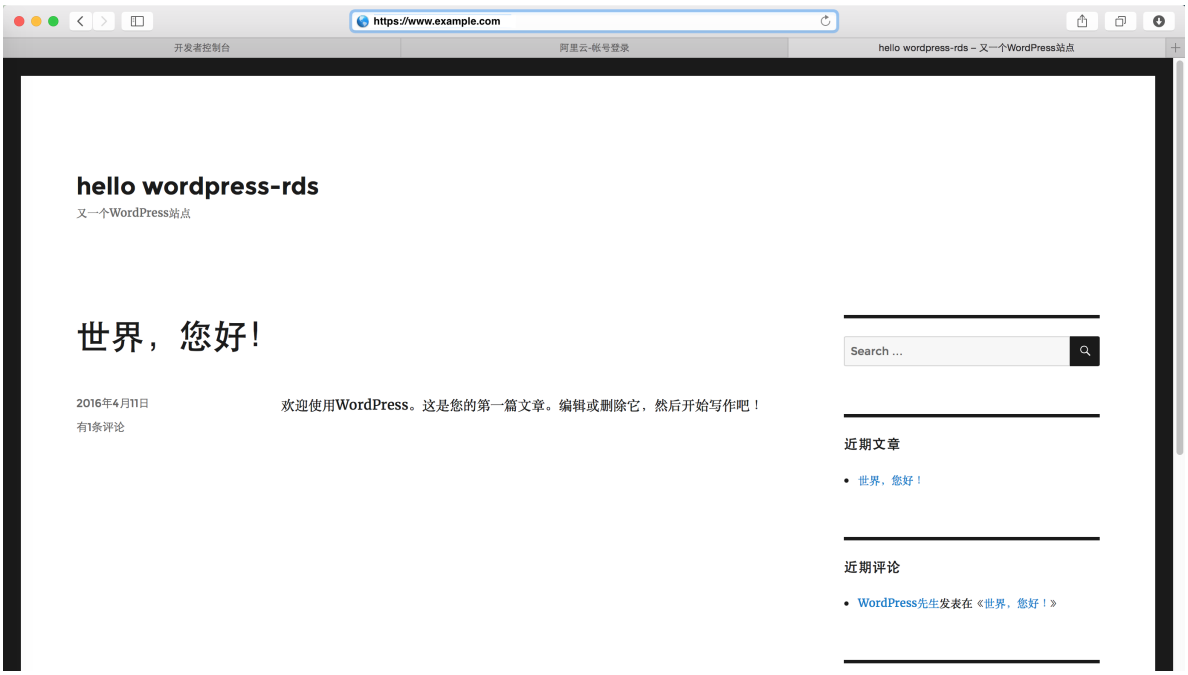
4. 完成健康检查配置标签页中的配置，如下所示。单击确认。

   您可以选择开启或关闭健康检查。如果您选择开启健康检查，您需要在域名中填写您自己的域名或者在检查路径中填写 `/haproxy-monitor`。否则，健康检查会报异常。


添加监听
1.基本配置
2.健康检查配置
3.配置成功
是否开启健康检查:
已开启
域名：
长度1-80个字符
只能使用字母、数字、'-'、'.'，默认使用各后端服务器的内网IP为域名
检查端口：
端口输入范围为1-65535。
默认使用后端服务器的端口进行健康检查
检查路径：
/haproxy-monitor
用于健康检查页面文件的URI，建议对静态页面进行检查。长度限制为1-80个字符，只能使用字母、数字、'-'、'/'、'.'、'%'、'?'、'#'、'&'、'='这些字符。
响应超时时间：*
5
秒
每次健康检查响应的最大超时时间;输入范围1-300秒,默认为5秒
健康检查间隔：*
2
秒
进行健康检查的时间间隔; 输入范围1-50秒,默认为2秒
不健康阈值：*
2 3 4 5 6 7 8 9 10
表示云服务器从成功到失败的连续健康检查失败次数。
健康阈值：*
2 3 4 5 6 7 8 9 10
表示云服务器从失败到成功的连续健康检查成功次数。
正常状态码:
http_2xx
http_3xx
http_4xx
http_5xx
健康检查正常的http状态码
上一步
确认
取消

5. 配置成功后，单击确认。

6. 访问页面`https://www.example.com`。



后续操作

完成以上配置后，如果您需要设置访问 `http://www.example.com`,直接跳转到`https://www.example.com`，请参考简单路由-*HTTP* 强制跳转到 *HTTPS* 进行设置。

5.4 简单路由-HTTP 强制跳转到 HTTPS

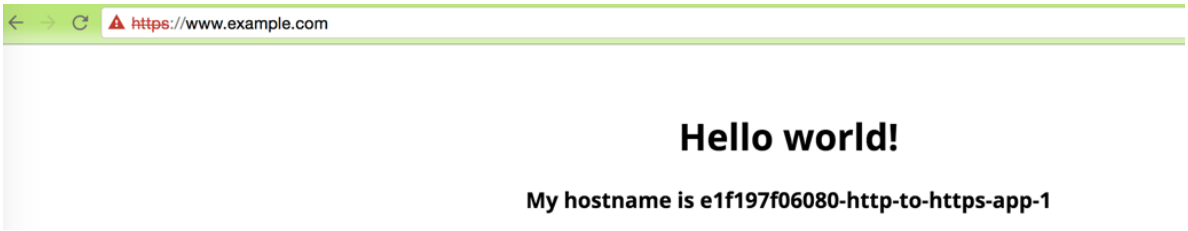
步骤1 实现 https 协议访问 helloworld 应用

1. 您可以使用编排模板创建 hello world 应用。

   应用模板模板示例如下。

```
app:
     ports:
         - 80/tcp
     image: 'registry.cn-hangzhou.aliyuncs.com/linhuatest/hello-
world:latest'
     labels:
         # 此处只是 http/https/ws/wss 协议
         aliyun.routing.port_80: "http://www.example.com"
     restart: always
```

2. 配置好负载均衡之后，参见简单路由-*HTTP* 协议变为 *HTTPS* 协议，访问 HTTPS 协议的网站如下所示。



步骤2 配置 nginx 容器实现强制跳转到 HTTPS

1. 您可以配置使 HTTP 协议请求强制跳转到 HTTPS 协议。

   本例中，创建一个 nginx 容器，监听 http 请求，将所有的 http 请求通过 rewrite 重写到 https上，从而将请求重定向至 https 协议，实现基于 `www.example.com` 域名的强制跳转。

   下面的示例配置了 nginx 容器并将 rewrite 规则写到配置文件中，即如果收到请求 `http://www.example.com`，则返回 `301` 且自动跳转到 `https://www.example.com`。

   - 登录集群中的每台机器，创建 nginx.conf 配置文件，该配置文件将以 volume 形式挂载到容器 nginx 中。

```
cd /                              ## 返回根目录
mkdir ngx                         ## 创建 ngx 目录
vim nginx.conf                    ## 创建 nginx.conf 配置文件
```

   - 在*/ngx/nginx.conf*配置文件中输入如下的配置代码。

```
user  nginx;
error_log  /var/log/nginx/error.log warn;
```

```
pid        /var/run/nginx.pid;
events {
  worker_connections  65535;
}
http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;
  log_format  main  '$remote_addr - $remote_user [$time_local] "$
request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';
  access_log  /var/log/nginx/access.log  main;
  keepalive_timeout  65;
  gzip  on;
  server {
      listen       80;
      server_name  localhost;
      return 301 https://$host$request_uri;
  }
}
```

2. 使用编排模板创建 nginx 应用。

   nginx 应用的编排模板如下所示。

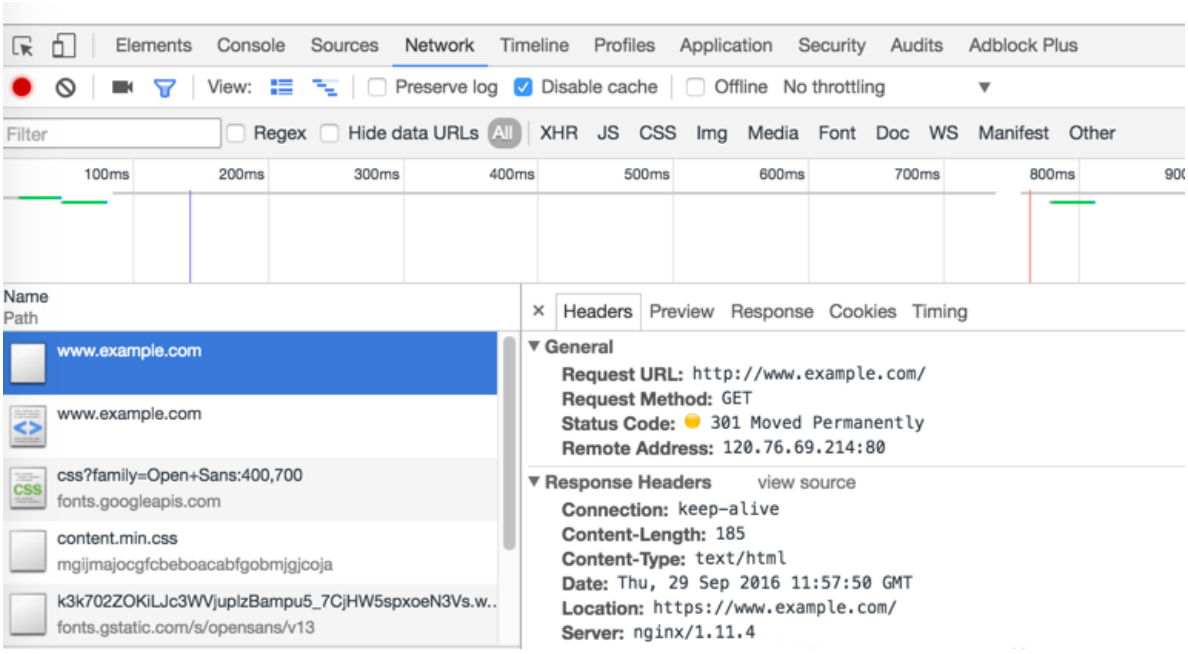
```
nginx:
     ports:
       - 80:80/tcp # 映射到主机的 80 端口
     image: 'nginx:latest'
     labels:
       aliyun.global: true # 每台机器均部署一个 nginx 容器，达到高可用目的
     volumes:
       - /ngx/nginx.conf:/etc/nginx/nginx.conf
     restart: always
```

3. 配置集群负载均衡的监听规则。

   如下图所示（其中，前端 80 端口 > 后端 80 端口，即负载均衡的前端端口 > 后端 ECS 实例的端口 80）。

4. 验证 HTTP 强制跳转到 HTTPS

   当您访问 `http://www.example.com` 时，会自动跳转到 `https://www.example.com`。返回的 HTTP 协议内容如下图所示，即完成了正确跳转到 `https://www.example.com`。



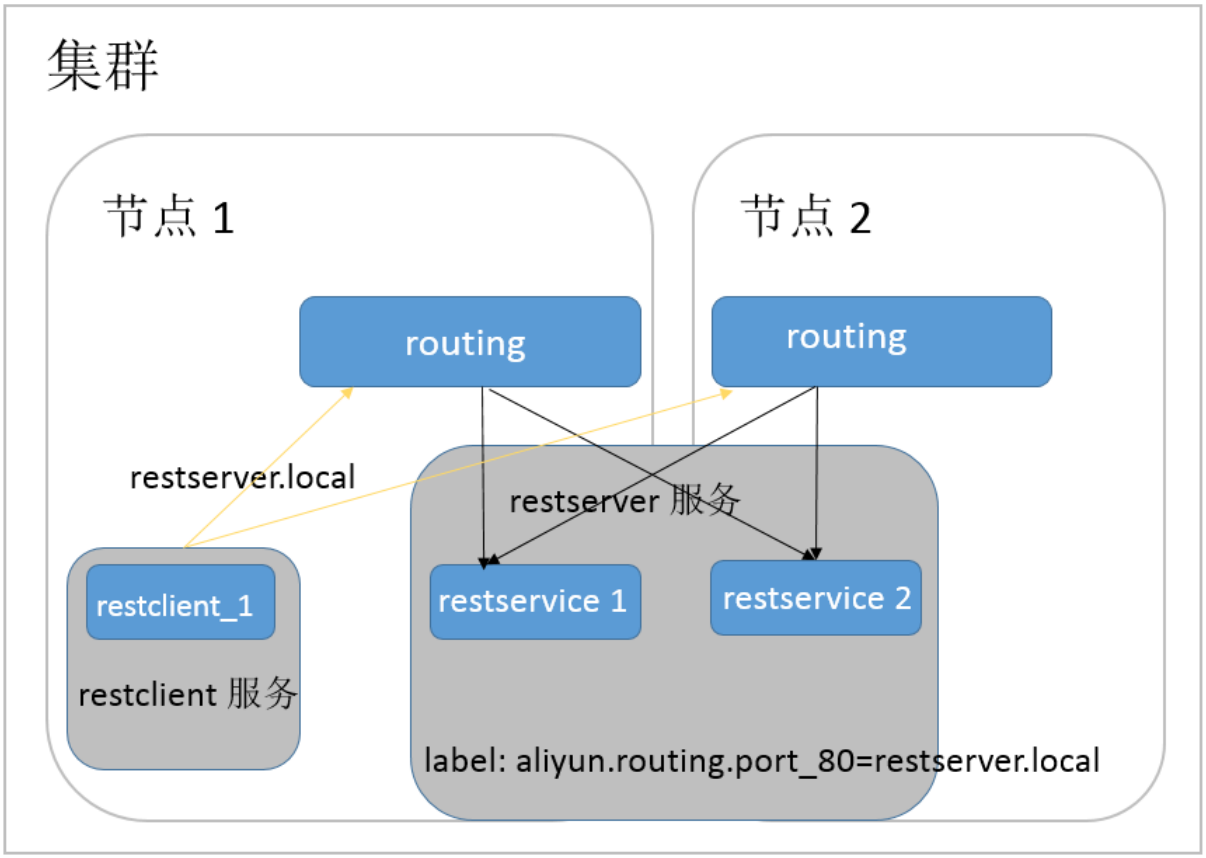
5.5 集群内服务间路由和负载均衡

在容器服务上可以通过简单路由#支持 *HTTP/HTTPS)* 将基于域名的 HTTP 服务暴露出去，而且能够配合健康检查自动的负载均衡和服务发现，当其中一个容器出现问题之后，routing 会自动将健康检查失败的容器从后端摘除，所以能做到自动的服务发现。然而这个是将服务暴露到外网环境。

那么集群内服务间如何通过这种方式做到自动的服务发现和负载均衡呢？阿里云容器服务中的 routing 容器具备负载均衡的功能，您只需要使用以 `.local` 结尾的域名，让容器仅能被集群内的其他容器所访问，然后配合 external_links 标签，从而实现集群内服务间发现和负载均衡。

实现原理

1. 利用了 Docker 1.10 之后支持在容器中做别名的方式，在依赖负载于 restserver.local 的 restservice 容器中， `restserver.local` 域名实际解析的是 routing 容器的地址，restclient 服务发起请求时，首先将 HTTP 请求转发到 routing 容器，并带上 `HOST` 为 `restserver.local` 的请求头。
2. routing 容器会对配置 `aliyun.routing.port_xxx: restserver.local` 标签的容器健康状态进行监听，并将其挂载到 HAProxy 的后端，HAProxy 接收到带有 `restserver.local` HOST 头的 HTTP 请求后，就能转发到对应的容器。



优势

- 相对于使用 link 或者 hostname 的基于 DNS 的方式，首先不同客户端对 DNS 缓存的处理不一致会导致服务发现的延迟性，其次 DNS 的方案也只有 round robin，不能满足微服务的场景需求。
- 而相对于其他的微服务服务发现的解决方案，本方案提供了一个实现无关的服务发现和负载均衡机制，无需 server 端和 client 应用做任何修改即可使用。
- 服务生命周期是解耦的，每个微服务可以采用一个 docker-compose 模板独立部署，更新。相互之间只是通过一个虚拟域名实现动态绑定即可。

编排示例

在下面的编排示例中，为 restserver 服务增加 `aliyun.routing.port_80: restserver.local`标签，确保只有集群内的容器可以访问这个域名。然后为 restclient 服务配置 `external_links`，指向 restserver.local 这个域名。reclient 服务便可以这个域名访问到 restserver 服务，并且能够配合健康检查做到自动的服务发现。

```
restserver: # 模拟 rest 服务
  image: nginx
  labels:
```

```
    aliyun.routing.port_80: restserver.local # 使用 local 的域名，只有集
群内的容器可以访问这个域名
    aliyun.scale: "2" # 扩展出两个实例，模拟负载均衡
    aliyun.probe.url: "http://container:80" # 定义容器的健康检查策略是
http，端口是 80
    aliyun.probe.initial_delay_seconds: "2" # 健康检查在容器起来之后两秒之
后再检查
    aliyun.probe.timeout_seconds: "2" # 健康检查超时时间，如果两秒还没返回认
为不健康
restclient: # 模拟 rest 服务消费者
  image: registry.aliyuncs.com/acs-sample/alpine:3.3
  command: "sh -c 'apk update; apk add curl; while true; do curl --
head restserver.local; sleep 1; done'" #访问 rest 服务，测试负载均衡
  tty: true
  external_links:
    - "restserver.local" #指定 link 的服务的域名。请确保您设置了 external_l
inks，否则访问会失败。
```

然后，通过如下的 restclient 服务的日志，您可以看到 restclient 的 curl 的 http 请求被路由到不同 rest 服务的容器上，容器 ID 分别为 `053cb232fdfbcb5405ff791650a0746ab77f26cce74fea2320075c2af55c975f` 和 `b8c36abca525ac7fb02d2a9fcaba8d36641447a774ea956cd93068419f17ee3f`。

```
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066803626Z
Server: nginx/1.11.1
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066814507Z
Date: Fri, 01 Jul 2016 06:43:49 GMT
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066821392Z
Content-Type: text/html
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066829291Z
Content-Length: 612
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066835259Z
Last-Modified: Tue, 31 May 2016 14:40:22 GMT
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066841201Z
ETag: "574da256-264"
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066847245Z
Accept-Ranges: bytes
internal-loadbalance_restclient_1 | 2016-07-01T06:43:49.066853137Z
 Set-Cookie: CONTAINERID=053cb232fdfbcb5405ff791650a0746ab77f26cc
e74fea2320075c2af55c975f; path=/
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.080502413Z
HTTP/1.1 200 OK
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082548154Z
Server: nginx/1.11.1
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082559109Z
Date: Fri, 01 Jul 2016 06:43:50 GMT
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082589299Z
Content-Type: text/html
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082596541Z
Content-Length: 612
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082602580Z
Last-Modified: Tue, 31 May 2016 14:40:22 GMT
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082608807Z
ETag: "574da256-264"
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082614780Z
Accept-Ranges: bytes
```

```
internal-loadbalance_restclient_1 | 2016-07-01T06:43:50.082621152Z
 Set-Cookie: CONTAINERID=b8c36abca525ac7fb02d2a9fcaba8d36641447a7
74ea956cd93068419f17ee3f; path=/
```

5.6 负载均衡路由

暴露 HTTP 协议或者 HTTPS 协议的服务

推荐使用简单路由服务（即 routing）的方式来暴露 HTTP 服务或者 HTTPS 协议的服务，如果您希望搭建自己的路由链路，可以开通新的内网或者公网负载均衡实例路由到 VM 的端口（通过阿里云扩展标签*lb* 来实现），并设置主机和容器的映射关系来进行请求的路由。

适用场景：

7 层协议负载均衡，自定义各服务的路由，在将传统架构迁移到容器架构过程中非容器集群的服务访问容器集群中的服务。

暴露 TCP 协议或者 UDP 协议的服务

目前如果要暴露 TCP 协议的服务，需要您自行设置负载均衡实例或者公网 IP，并配置好主机端口与容器端口的映射（通过阿里云扩展标签*lb* 来实现）。

> 说明：
> 如果要使用负载均衡实例，您需要购买一个新的负载均衡实例。多个服务不能共享使用同一个负载均衡实例，同时不能共享使用集群默认负载均衡实例。

适用场景：

4 层协议的负载均衡，自定义各服务的路由，在将传统架构迁移到容器架构过程中非容器集群的服务访问容器集群中的服务。

示例：

通过自定义负载均衡的方式来将容器集群内的 Redis 服务暴露给容器集群外的 Python 应用。

1. 首先在负载均衡管理控制台（单击页面右上角的**创建负载均衡**）购买创建一个用于路由的负载均衡实例。

   本示例中选择的是公网实例，您可以根据自己的需要选择公网或者私网。

   > 说明：

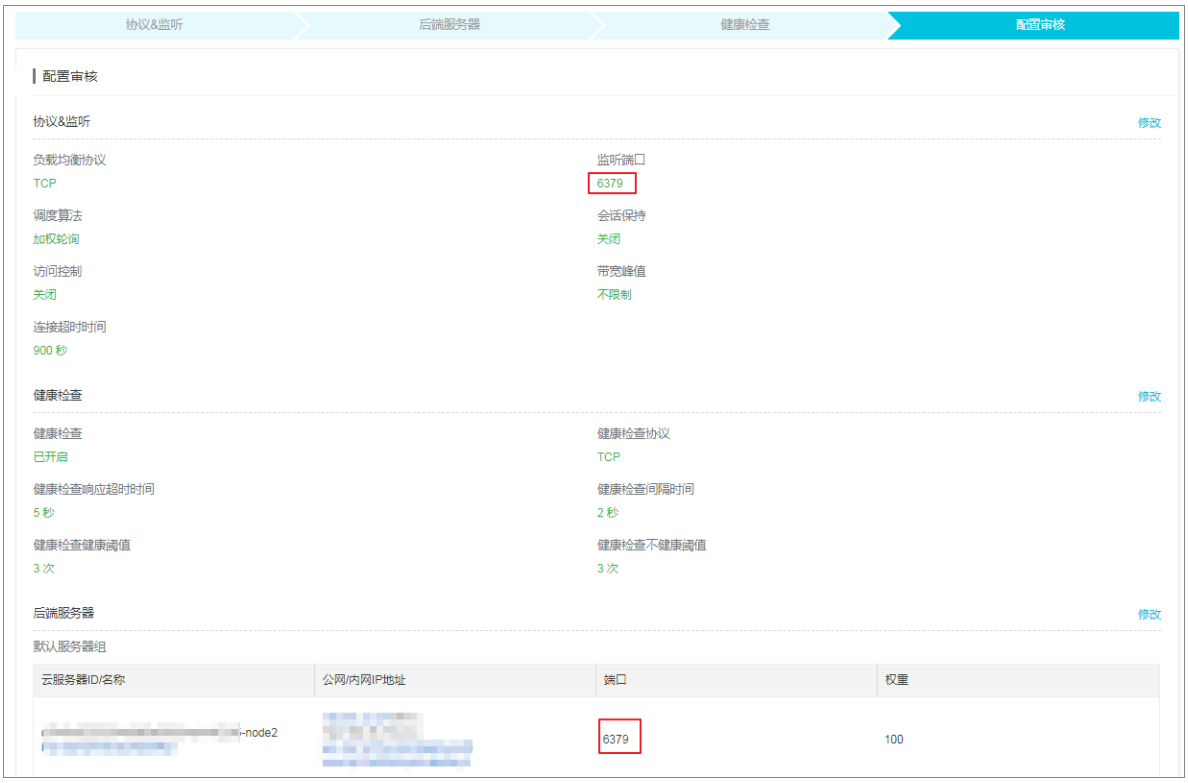
由于负载均衡不支持跨地域（Region）部署，因此应选择与您所使用容器服务集群相同的地域。

2. 返回负载均衡管理控制台，将购买创建的负载均衡实例命名为 slb_redis_app。容器服务能通过该名称来引用这个负载均衡实例。您也可以通过负载均衡实例 ID 来引用。

   单击左侧导航栏中的**实例 > 实例管理**，选择实例所在的地域，选择所需实例，然后编辑实例的名称并单击**确定**。

3. 创建监听端口。

   单击实例右侧的监听配置向导，在负载均衡业务配置向导页面配置监听规则，创建协议为 TCP，添加后端服务器，端口映射为 6379:6379。最后的配置如下图所示。



4. 登录容器服务管理控制台，选择一个已有集群，创建一个名称为 redis-demo 的应用，单击使用镜像创建。

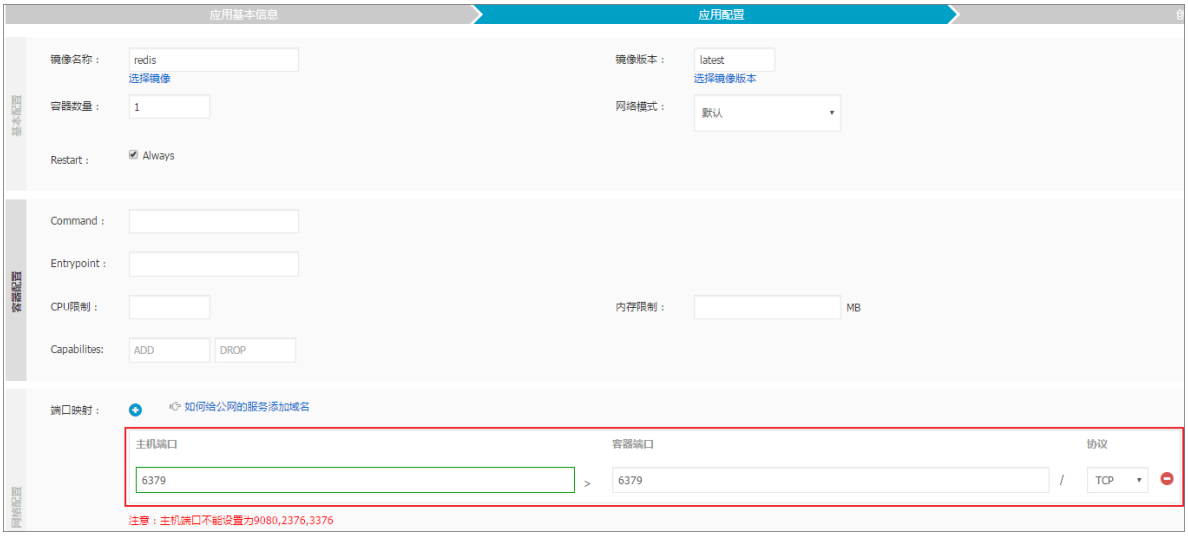
   有关如何创建应用，参见创建应用。

> **说明：**
> 由于负载均衡不支持跨地域（Region）部署，因此您所使用的容器服务集群需要和上边创建的负载均衡实例处于相同的地域。

5. 选择 Redis 镜像并设置端口映射。

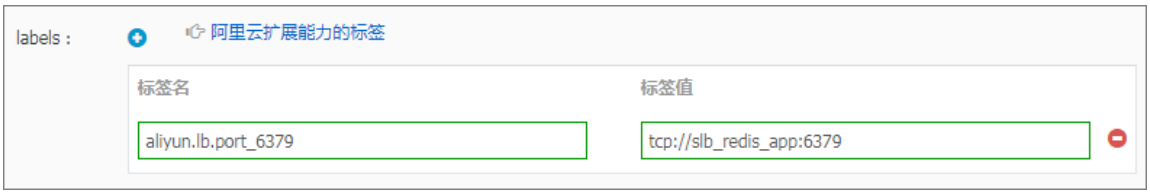
> 说明：
>
> 此处 Redis 镜像只是开通了容器的 6379 端口，为了使创建的负载均衡路由到这个容器端口，您必须指定 Redis 镜像的主机端口:容器端口的映射。

在端口映射中，指定主机端口为 6379，主机端口 6379 即为负载均衡实例绑定的后端主机端口，选择使用的协议为 TCP.



6. 为了配置自定义负载均衡，需要让 Redis 服务知道使用的负载均衡实例的信息。您可以通过向服务注入一个标签来实现或者通过设置负载均衡路由配置。

   - 向服务注入一个标签。本示例中，标签为 `aliyun.lb.port_6379: tcp://slb_redis_app:6379`。



标签格式如下，带 $ 的变量为占位符。

```
aliyun.lb.port_$container_port:$scheme://$[slb_name|slb_id]:$
front_port
```

- `$container_port` 表示容器要暴露的端口。
- `$scheme`表示负载均衡实例监听端口支持的协议，可能的值为 `tcp`、`http`、`https`、`udp`。
- `$[slb_name|slb_id]` 表示可以填写负载均衡实例的名称或者 ID。
- `$front_port` 表示负载均衡实例要暴露的前端端口。

更多详细信息，参见阿里云扩展标签*lb*。

- 在**创建应用**页面，单击**负载均衡路由配置**右侧的加号图标，设置要配置的负载均衡实例的信息，如下图所示。

  该设置对应的标签内容为 `6379: tcp://slb_redis_app:6379`。

本示例中，路由到的容器端口为 6379，引用前面创建的负载均衡实例名称 `slb_redis_app`，与上面主机端口:容器端口的映射设置的 TCP 协议相呼应，本示例设置监听端口的协议为 TCP 协议，同时设置负载均衡的前端端口为 6379。

> **说明：**
> 本示例中，同时将负载均衡实例的前端端口、后端端口（即主机的端口）和容器端口均设置为 6379，您可以根据自己的需要设置不同的前端端口和主机端口。

7. 单击**创建**，Redis 应用即开始创建了。Redis 应用在创建的过程中会自动将名称为 slb_redis_app 的负载均衡实例绑定到部署了 redis 镜像的后端主机。

8. 当应用处于就绪状态后，登录负载均衡管理控制台，查看名为 `slb_redis_app` 的负载均衡实例的状态。

   单击实例ID，在实例详情页面，单击**默认服务器**。

   由健康状态可见，负载均衡已经正确地绑定到了 Redis 的后端。

9. 您可以在负载均衡管理控制台 的**实例管理**页面查看负载均衡实例的 IP 地址，并使用命令行工具 `telnet $Server_Load_Balancer_IP_address 6379` 来检查端口的可访问性。

10. 为了测试以上配置，在本地运行一个简单的 Python 应用来通过 slb_redis_app 负载均衡实例访问容器集群内的 Redis。

> 说明：
>
> Redis 主机地址是负载均衡的 IP 地址。

app.py

```
from flask import Flask
from redis import Redis
app = Flask(__name__)
redis = Redis(host='$Server_Load_Balancer_IP_address', port=6379)
@app.route('/')
def hello():
redis.incr('hits')
return 'Hello World! I have been seen %s times.' % redis.get('hits')
if __name__ == "__main__":
app.run(host="0.0.0.0", debug=True)
```

requirements.txt

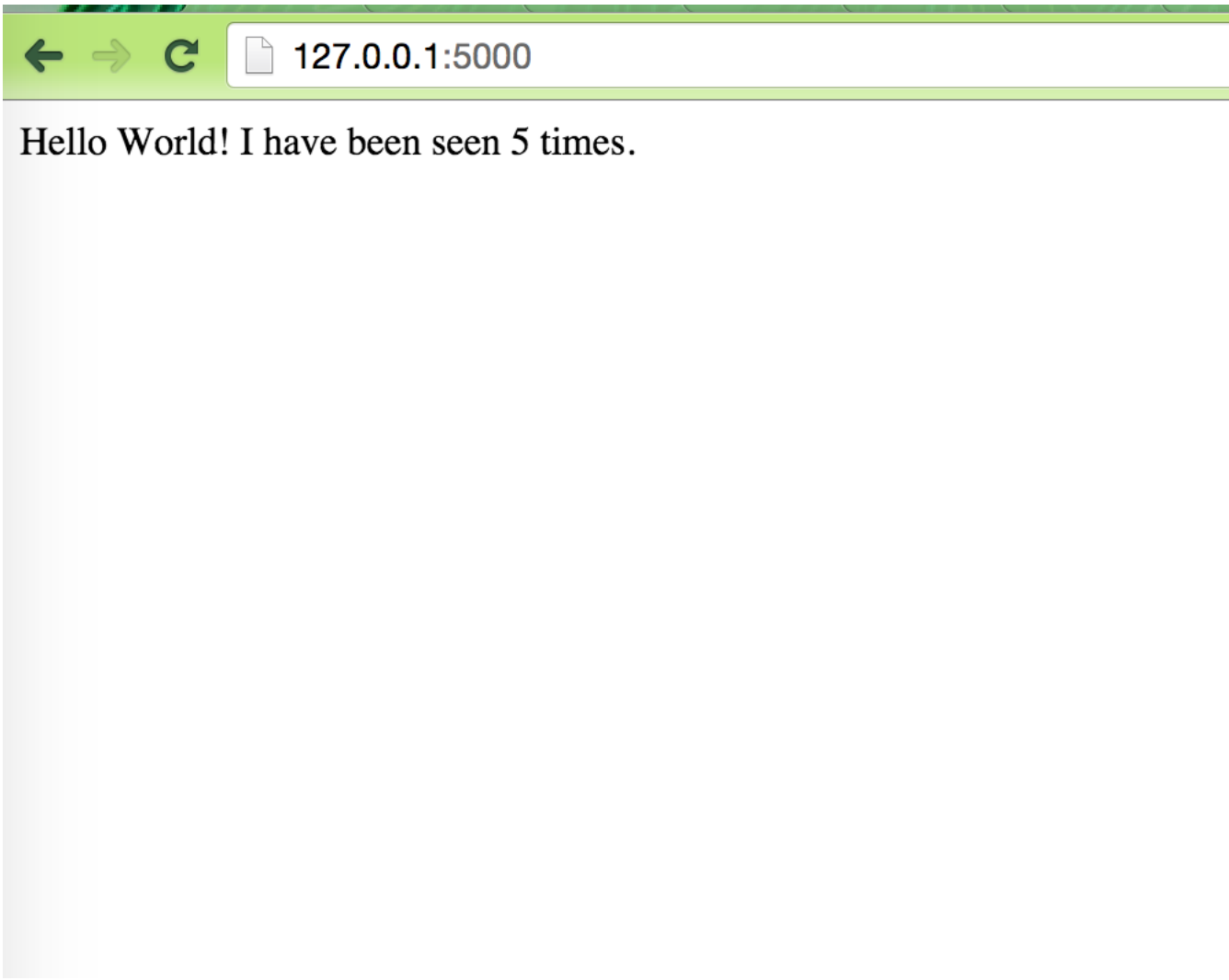
```
flask
redis
```

shell

```
$ pip install -r requirements.txt
$ python app.py
Running on http://0.0.0.0:5000/  ## Press CTRL+C to quit
Restarting with stat
Debugger is active!
```

```
Debugger pin code: 243-626-653
```

访问结果如下图所示。



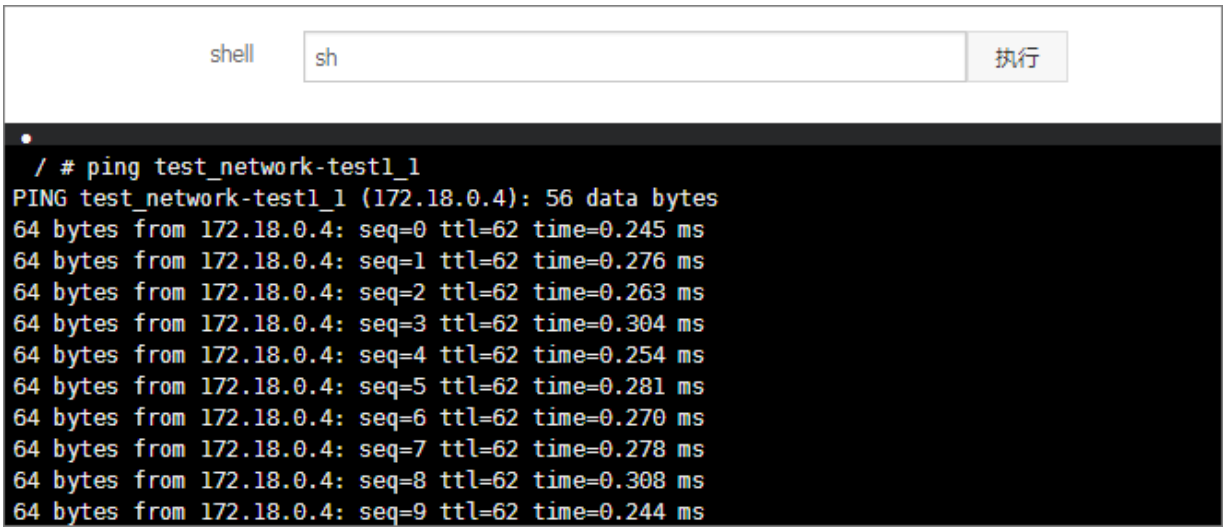
5.7 容器间的互相发现

容器服务为集群内的服务和容器提供多种服务发现方式，可以通过容器名、link、hostname 等进行发现。

通过容器名

容器服务不仅可以通过容器的 IP 进行访问，还可以通过网络中其他容器的容器名进行访问，通过跨主机互联的容器网络 中的例子，您可以在 `test_network-test2_1` 的容器中通过 `test_network-test1_1` 的容器名进行访问。

如果在编排文件中不指定 `container_name` 的话，默认的容器名为 `{project-name}_{service-name}_{container-index}`。在连接管理终端后，您可以通过另外一个服务的容器名进行访问，来测试网络的连通性。如下图所示。



通过 link

容器服务支持编排模板服务间的 link，服务间的 link 可以将一个服务的容器 link 到另外一个服务的容器中，而容器中可以通过 link 进来的服务别名访问到依赖的容器，并且在依赖的容器的 IP 变化时可以动态的更新别名解析的 IP。具体的例子可以参考容器服务示例编排中的 WordPress 编排，其中 WordPress 中 Web 服务 link db:mysql 的服务到容器内，如下所示。容器内部就可以通过 MySQL 的域名访问到 db 服务的容器。

```
links:
    - 'db:mysql'
```

通过 hostname

如果在编排模板的服务中定义了 *hostname* 的配置，则在集群中便可以通过这个 hostname 访问到这个容器。

例如：

```
testhostname:
        image: busybox
        hostname: xxserver
        command: sleep 100000
```

```
        tty: true
```

那么，集群中就可以通过 xxserver 解析并访问到这个服务的容器，具体请参见跨主机互联的容器网络 的编排示例。当这个服务在有多个容器时，通过这个域名访问还可以做到一定的负载均衡的作用。

另外，如果服务没有配置 hostname 的话，容器服务会把容器的容器名作为容器内部的 hostname；如果有应用需要在容器内知道自己的容器名，用于服务的注册，比如 Eureka Client，需要注册一个可被访问的地址到 Eureka Server，容器内的进程可以获取到容器名用于服务注册，并让其他的服务调用者通过容器名互相访问。

5.8 自定义路由-使用手册

acs/proxy

自定义代理镜像，通过 FROM dockercloud/haproxy 的方式继承自镜像 dockercloud/haproxy，动态感知容器的状态，做到后端容器负载均衡代理和服务发现。特点是将 HAProxy 负载均衡软件的所有配置都参数化了，方便您自定义自己的需求和配置。

该镜像主要用于 Alibaba Cloud 容器服务的默认路由服务不能满足您需求的场景，方便您对 HAProxy 进行自定义配置。

文档中会提到 acs/proxy 和 HAProxy，均指代该镜像或者镜像中的软件 HAProxy。

动态负载均衡代理和服务发现的原理

- 镜像 acs/proxy 通过容器自身环境变量确定负载均衡的全局（GLOBAL）和默认（DEFAULT）配置。
- 镜像 acs/proxy 侦听集群中的事件，例如容器状态的变化，发生变化后重新获取集群中相关容器的信息，确定最新的负载均衡配置。
- 镜像 acs/proxy 根据最新的负载均衡配置去重新加载（reload）该配置，使得该配置生效。

如何确定负载均衡的后端容器

- 根据 acs/proxy 的环境变量 ADDITIONAL_SERVICES 来确定范围。
  - ADDITIONAL_SERVICES: "*" 表示范围为整个集群。
  - ADDITIONAL_SERVICES: "project_name1:service_name1,project_name2:service_name2"表示范围为当前应用和指定应用的指定服务。
  - ADDITIONAL_SERVICES 不设置或者为空表示范围为当前应用的容器。

- 根据每个容器的标签来确定是否加入 `acs/proxy` 的后端。
  - `aliyun.proxy.VIRTUAL_HOST: "www.toolchainx.com"` 表示加入后端，且域名为 `www.toolchainx.com`。
  - `aliyun.proxy.required: "true"` 表示加入后端，且作为默认的后端。

如何在前端绑定负载均衡

使用自定义负载均衡标签，例如 `aliyun.lb.port_80: 'tcp://proxy:80'`。

说明：

- 请不要将集群的负载均衡实例（即简单路由的负载均衡）用来做自定义负载均衡。
- 任何两个不同的服务均不能共享使用同一个负载均衡，否则会导致负载均衡后端机器被删除，服务不可用。

关于自定义负载均衡标签的使用方法，参见 标签。

示例模板

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
          -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  使用自定义负载均衡，前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
appone:
    expose:  # 被代理的服务一定要使用 expose 或者 ports 告诉 proxy 容器暴露哪个端口
        -  80/tcp
    image:  'nginx:latest'
    labels:
        # 此处支持 http/https/ws/wss 协议
        # 必须使用您自己的域名而不是容器服务提供的测试域名
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
    restart:  always
```

配置说明

通过 **acs/proxy** 镜像的环境变量设置全局（GLOBAL）和默认（DEFAULT）配置

说明：

您需要重新部署 HAProxy 服务才能使此处的配置变更生效。该部分的配置针对的是镜像 `acs/proxy` 所在服务的环境变量配置。

环境变量	默认值	描述
ADDITIONAL_SERVICES		需要进行负载均衡的附加服务列表（比如：`prj1:web,prj2:sql`）。服务发现将基于 com.docker.compose.[project|service] 容器标签。该环境变量仅适用于 compose V2，且必须确保容器可以解析和访问依赖的服务所在的网络。
BALANCE	roundrobin	要使用的负载均衡算法。可能的取值包括 `roundrobin`、`static-rr`、`source` 和 `leastconn`。更多详细信息，参见 *HAProxy#balance*。
CA_CERT_FILE		CA 证书文件的路径。通过使用该环境变量，您可以直接从数据卷挂载 CA 证书文件，而不需要通过环境变量传递证书文件内容。如果您设置了该变量，系统会忽略 `CA_CERT` 环境变量。可设置为 `/cacerts/cert0.pem`。
CA_CERT		HAProxy 用于验证客户端的 CA 证书。该环境变量的格式与 `DEFAULT_SSL_CERT` 相同。
CERT_FOLDER		证书的路径。通过使用该变量，您可以直接从数据卷挂载 CA 证书文件，而不需要通过环境变量传递证书文件的内容。如果您设置了该变量，系统会忽略所依赖服务的 `DEFAULT_SSL_CERT` 和

环境变量	默认值	描述
		`SSL_CERT` 环境变量。可设置为 `/certs/`。
DEFAULT_SSL_CERT		默认的 SSL 证书。该 pem 文件包含私钥和公钥（其中，私钥在前，公钥在后），使用 `\n`（两个字符）作为行分隔符。内容应该写在同一行，参见 *SSL Termination*。
EXTRA_BIND_SETTINGS		额外设置，由逗号进行分隔的字符串（`<port>:<setting>`）。每一部分会被附加到配置文件中对应的端口绑定节点。如果需要使用逗号，需要使用 `\,` 进行转义。可设置为 `443: accept-proxy, 80:name http`。
EXTRA_DEFAULT_SETTINGS		额外设置，由逗号进行分隔的字符串。每一部分会被附加到配置文件中的 DEFAULT 节点。如果需要使用逗号，需要使用 `\,` 进行转义。
EXTRA*FRONTEND_SETTINGS*\<PORT>		额外设置，由逗号进行分隔的字符串。每一部分会按照环境变量名称中指定的端口号被附加到前端节点。如果需要使用逗号，需要使用 `\,` 进行转义。可设置为 `EXTRA_FRONTEND_SETTINGS_80=balance source, maxconn 2000`。
EXTRA_GLOBAL_SETTINGS		额外设置，由逗号进行分隔的字符串。每一部分会被附加到配置文件中的 GLOBAL 节点。如果需要使用逗号，需要使用 `\,` 进行转义。可设置为 `tune.ssl.cachesize 20000`

环境变量	默认值	描述
		, tune.ssl.default-dh-param 2048。
EXTRA_ROUTE_SETTINGS		该字符串会在健康检查结束后被附加到每一个后端路由上。依赖的服务中的设置会覆盖该设置。可设置为 send-proxy 。
EXTRA_SSL_CERTS		多余证书的名称列表，由逗号分隔，例如 CERT1, CERT2, CERT3。您还需要将每一个证书指定为单独的环境变量，例如 CERT1="<cert-body1>", CERT2="<cert-body2>", CERT3="<cert-body3>"。
HEALTH_CHECK	check	在每一个路由上设置健康检查。可设置为 “check inter 2000 rise 2 fall 3”。更多详细信息，参见 *HAProxy#check*。
HTTP_BASIC_AUTH		HTTP 基本认证的身份信息列表，格式为 <user>:<pass>，由逗号分隔。该身份信息适用于所有后端路由。如果需要使用逗号，需要使用 \, 进行转义。注意：生产环境中，请不要使用该环境变量进行认证。
MAXCONN	4096	设置每个进程的最大同时连接数。
MODE	http	HAProxy 的负载均衡模式。可能的值包括 http、tcp 和 health。
MONITOR_PORT		要添加 MONTIOR_URI 的端口号。该变量同 MONTIOR_URI 配合使用。可设置为 80。
MONITOR_URI		为获取 HAProxy 的健康状态所需要拦截的具体 URI。更多详

环境变量	默认值	描述
		细信息，参见 *Monitor URI*。可设置为 `/ping`。
OPTION	redispatch	`default` 节点中的 HAProxy `option` 条目列表，由逗号分隔。
RSYSLOG_DESTINATION	127.0.0.1	HAProxy 日志将要发送到的 rsyslog 目的地址。
SKIP_FORWARDED_PROTO		如果设置了该环境变量，HAProxy 将不会添加 X-Forwarded-For http请求头。当 HAProxy 与其它负载均衡一同使用时，可以使用该环境变量。
SSL_BIND_CIPHERS		设置 SSL 服务器所要使用的 SSL 密钥套件。该环境变量设置了 HAProxy `ssl-default-bind-ciphers` 的配置。
SSL_BIND_OPTIONS	no-sslv3	设置 SSL 服务器所使用的 SSL 绑定选项。该环境变量设置了 HAProxy `ssl-default-bind-options` 的配置。设置为默认值则只允许在 SSL 服务器上使用 TLSv1.0+。
STATS_AUTH	stats:stats	访问 Haproxy stats 统计页面所需要的用户名和密码。
STATS_PORT	1936	HAProxy stats 统计页面暴露的端口号。如果开放了该端口，则可以通过 `http://<host-ip>:<STATS_PORT>/`访问 stats统计页面。
TIMEOUT	connect 5000, client 50000, server 50000	`default` 节点中 HAProxy `timeout` 条目列表，由逗号分隔。

被代理的后端服务通过相应服务镜像的标签进行某一后端服务的配置

即通过将标签写到后端服务的镜像上来进行配置。该部分的配置写在被代理的服务的模版部分

此处的设置可以覆盖 HAProxy 的设置。HAProxy 的设置仅适用于依赖的服务。如果在阿里云容器服务上运行，当服务重新部署，加入或者退出 HAProxy 服务时，HAProxy 服务会自动进行更新来应用这些变更。

标签	描述
aliyun.proxy.APPSESSION	会话保持选项。可设置为 `JSESSIONID len 52 timeout 3h`。更多详细信息，参见 *HAProxy:appsession*。
aliyun.proxy.BALANCE	要使用的负载均衡算法。可设置为 `roundrobin`、`static-rr`、`source` 和 `leastconn`。更多详细信息，参见 *HAProxy:balance*。
aliyun.proxy.COOKIE	会话保持选项。可设置为 `SRV insert indirect nocache`。更多详细信息，参见 *HAProxy:cookie*。
aliyun.proxy.DEFAULT_SSL_CERT	与 `SSL_CERT` 类似。但是该变量将 pem 文件保存在 `/certs/cert0.pem` 下作为默认的 SSL 证书。如果在依赖的服务或者HAProxy 中设置了多个 `DEFAULT_SSL_CERT`，会导致未定义的行为。
aliyun.proxy.EXCLUDE_PORTS	端口号，由逗号分隔（例如：3306, 3307）。默认情况下，HAProxy 将应用服务暴露的所有端口添加到后端路由。您可以指定您不希望进行路由的端口，比如数据库端口。
aliyun.proxy.EXTRA_ROUTE_SETTINGS	该字符串会在健康检查结束后被附加到每一个后端路由上。可设置为 “send-proxy”。
aliyun.proxy.EXTRA_SETTINGS	额外设置，由逗号分隔。每一部分会被附加到配置文件中相关的后端节点或监听会话。如果需要使用逗号，使用 `\,` 进行转义。可设置为 `balance source`。
aliyun.proxy.FORCE_SSL	如果设置了该环境变量且启用了 SSL termination，HAProxy 会将 HTTP 请求重定向为 HTTPS 请求。

标签	描述
aliyun.proxy.GZIP_COMPRESSION_TYPE	启用 gzip 压缩。该环境变量的值为将要压缩的 MIME 类型列表。可设置为 `text/html text/plain text/css`。
aliyun.proxy.HEALTH_CHECK	在每一个后端路由上设置健康检查。可设置为“check inter 2000 rise 2 fall 3”。更多详细信息，参见 *HAProxy:check*。
aliyun.proxy.HSTS_MAX_AGE	启用 HSTS。该环境变量的值为一个整数，代表 HSTS 的有效期，单位为秒。可设置为 `31536000`。
aliyun.proxy.HTTP_CHECK	启用 HTTP 协议来检查服务器的健康情况。可设置为“OPTIONS * HTTP/1.1\r\nHost:\ www”。更多详细信息，参见 *HAProxy:httpchk*。
aliyun.proxy.OPTION	HAProxy `option` 条目列表，由逗号分隔。此处设置的 `option` 会被添加到相关的后端节点或监听节点，并会覆盖 HAProxy 容器中的 OPTION 设置。
aliyun.proxy.SSL_CERT	SSL 证书。该 pem 文件包含私钥和公钥（其中，私钥在前，公钥在后），使用 `\n`（两个字符）作为行分隔符。
aliyun.proxy.TCP_PORTS	由逗号分隔的端口（比如：9000, 9001, 2222/ssl）。`TCP_PORTS` 中列出的端口将在 TCP 模式下被负载均衡。以 `/ssl` 结尾的端口表示该端口需要 SSL termination。
aliyun.proxy.VIRTUAL_HOST_WEIGHT	虚拟主机的权重，同 `aliyun.proxy.VIRTUAL_HOST` 配合使用。该值为一个整数，默认值为 0。该设置会影响虚拟主机的 ACL 规则的顺序。 虚拟主机的权重越高，ACL 规则的优先级越高，即值越大，越先被路由到。
aliyun.proxy.VIRTUAL_HOST	指定虚拟主机和虚拟路径。格式为 `[scheme://]domain[:port][/path], ...`。可以在 `domain` 和 `path` 部分使用通配符 `*`。

有关以上内容的更多信息，参见*HAProxy* 配置手册。

虚拟主机和虚拟路径

您可以在 `VIRTUAL_HOST` 环境变量中同时指定虚拟主机和虚拟路径。该变量的值由 URL 组成，多个 URL 之前用逗号分隔，格式为 `[scheme://]domain[:port][/path]`。

条目	默认值	描述
scheme	http	可能的值包括 `http`、`https` 和 `wss`。
domain		虚拟主机。可以使用通配符 `*`。
port	80/433	虚拟主机的端口号。当 `scheme` 设置为 `https` 或 `wss` 时，默认的端口为 `443`。
/path		虚拟路径。以 `/` 开头，可以使用通配符 `*`。

匹配示例

虚拟主机	匹配	不匹配
`http://domain.com`	`domain.com`	`www.domain.com`
`domain.com`	`domain.com`	`www.domain.com`
`domain.com:90`	`domain.com:90`	`domain.com`
`https://domain.com`	`https://domain.com`	`domain.com`
`https://domain.com:444`	`https://domain.com:444`	`https://domain.com`
`\*.domain.com`	`www.domain.com`	`domain.com`
`\*domain.com`	`www.domain.com`、`domain.com`、`anotherdomain.com`	`www.abc.com`
`www.e\*e.com`	`www.domain.com`、`www.exxe.com`	`www.axxa.com`
`www.domain.\*`	`www.domain.com`、`www.domain.org`	`domain.com`
`\*`	any website with HTTP	
`https://\*`	any website with HTTPS	
`\*/path`	`domain.com/path`、`domain.org/path?u=user`	`domain.com/path/`

虚拟主机	匹配	不匹配
*/path/	domain.com/path/、domain.org/path/?u=user	domain.com/path、domain.com/path/abc
/path/	domain.com/path/、domain.org/path/abc	domain.com/abc/path/
//path/*	domain.com/path/、domain.org/abc/path/、domain.net/abc/path/123	domain.com/path
/.js	domain.com/abc.js、domain.org/path/abc.js	domain.com/abc.css
/.do/	domain.com/abc.do/、domain.org/path/abc.do/	domain.com/abc.do
/path/.php	domain.com/path/abc.php	domain/abc.php、domain.com/root/abc.php
.domain.com/.jpg	www.domain.com/abc.jpg、abc.domain.com/123.jpg	domain.com/abc.jpg
*/path, */path/	domain.com/path、domain.org/path/	
domain.com:90、https://domain.com	domain.com:90、https://domain.com	

说明：

- 基于 VIRTUAL_HOST 所生成的 ACL 规则的顺序是随机的。在 HAProxy 中，当范围较窄的规则（比如 web.domain.com）被放置在范围较宽的规则（比如 *.domain.com）之后时，系统永远不会达到范围较窄的规则。因此，如果您所设置的虚拟主机包含相互重叠的范围时，您需要使用 VIRTUAL_HOST_WEIGHT 来手动设置 ACL 规则的顺序，为范围较窄的虚拟主机设置高于范围较宽的虚拟主机的权重。
- 所有带有相同 VIRTUAL_HOST 环境变量设置的服务将被看做并合并为一个服务。这对于一些测试场景比较适用。

SSL termination

acs/proxy 支持证书的 SSL ternimation。您只需要为每一个需要使用 SSL ternimation 的应用设置 SSL_CERT 和 VIRTUAL_HOST 即可。然后，HAProxy 会读取依赖环境中的证书并开启 SSL ternimation。

说明：

当环境变量的值中包含等号（=）时（这种情况在 SSL_CERT中很普遍），Docker 会跳过该环境变量。因此，您只能在 Docker 1.7.0 或更高版本上使用多 SSL ternimation。

在以下情况下会启用 SSL ternimation：

- 至少设置了一个 SSL 证书。
- 没有设置 VIRTUAL_HOST 或者设置为使用 HTTPS 协议。

您可以通过以下方法设置 SSL 证书：

- 在 acs/proxy 中设置 DEFAULT_SSL_CERT。
- 在依赖于 acs/proxy 的应用服务中设置 aliyun.proxy.SSL_CERT 和/或 DEFAULT_SSL_CERT。

aliyun.proxy.SSL_CERT 和 DEFAULT_SSL_CERT 的区别在于 SSL_CERT 指定的多个证书保存为 cert1.pem, cert2.pem, ...，而 DEFAULT_SSL_CERT 指定的证书通常保存为 cert0.pem。在这种情况下，当没有设置 SNI 匹配时，HAProxy 会将 cert0.pem 用作默认证书。然而，当提供了多个 DEFAULT_SSL_CERTIFICATE 时，仅有其中一个证书可以保存为 cert0.pem，其它的证书将被弃用。

PEM 文件

acs/proxy 或依赖的应用服务所指定的证书为一个 pem 文件。该文件包含私钥和公钥（其中，私钥必须放在公钥和其它许可证书的前面）。您可以运行下面的脚本生成一个自签名证书。

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out ca.pem -days
1080 -nodes -subj '/CN=*/O=My Company Name LTD./C=US'
cp key.pem cert.pem
cat ca.pem >> cert.pem
```

您获取 pem 文件之后，可以运行下面的命令将文件转化为一行内容。

```
awk 1 ORS='\\n' cert.pem
```

拷贝转换后的内容并将其设置为 aliyun.proxy.SSL_CERT 或 DEFAULT_SSL_CERT 的值。

亲和性和会话保持

您可以通过以下任一方法设置亲和性和会话保持。

- 在您的应用服务中设置 `aliyun.proxy.BALANCE=source`。当设置了 `source` 负载均衡方法时， HAProxy 会对客户端 IP 地址进行哈希并确保同一个 IP 总是连接到同一个后端服务容器上。
- 设置 `aliyun.proxy.APPSESSION=<value>`。使用应用会话来确定客户端应该连接到哪一个服务容器上。`<value>` 可以设置为 `JSESSIONID len 52 timeout 3h`。
- 设置 `aliyun.proxy.COOKIE=<value>`。使用应用 cookie 来确定客户端应该连接到哪一个后端服务容器上。`<value>`可以设置为 `SRV insert indirect nocache`。

更多详细信息，参见 *HAProxy:appsession* 和 *HAProxy:cookie*。

TCP 负载均衡

默认情况下，`acs/proxy` 运行在 `http` 模式下。如果您希望依赖的服务运行在 `tcp` 模式下，您可以指定 `TCP_PORTS` 环境变量。该变量的值为以逗号分隔的端口号（9000, 9001）。

例如，如果您运行以下命令：

```
docker --name app-1 --expose 9000 --expose 9001 -e TCP_PORTS="9000,
9001" your_app
docker --name app-2 --expose 9000 --expose 9001 -e TCP_PORTS="9000,
9001" your_app
docker run --link app-1:app-1 --link app-2:app-2 -p 9000:9000, 9001:
9001 acs/proxy
```

HAProxy 将分别在 9000 端口和 9001 端口上对 `app-1` 和 `app-2` 进行负载均衡。

此外，如果您所暴露的端口多于 `TCP_PORTS` 中所设置的端口，剩余的端口将会使用 `http` 模式进行负载均衡。

例如，如果您运行以下命令：

```
docker --name app-1 --expose 9000 --expose 9001 -e TCP_PORTS="9000,
9001" your_app
docker --name app-2 --expose 9000 --expose 9001 -e TCP_PORTS="9000,
9001" your_app
docker run --link app-1:app-1 --link app-2:app-2 -p 9000:9000, 9001:
9001 acs/proxy
```

HAProxy 会在 80 端口使用 `http` 模式进行负载均衡，在 22 端口使用 `tcp` 模式进行负载均衡。

通过这种方法，您可以同时使用 `tcp` 模式和 `http` 模式进行负载均衡。

如果您在 TCP_PORTS 中设置了以 /ssl 结尾的端口，比如 2222/ssl，HAProxy 会在 2222 端口上设置 SSL termination。

说明：

- 您可以同时设置 VIRTUAL_HOST 和 TCP_PORTS，以便为 http 模式提供更多的管控。
- tcp 端口上的负载均衡适用于所有的服务。因此，如果您依赖了两个或多个使用同一 TCP_PORTS 设置的不同服务，acs/proxy 会将这些服务看做同一个服务。

WebSocket 支持

您可以通过以下任一方法启用 websocket 支持：

- 鉴于 Websocket 使用 HTTP 协议进行启动，您可以使用虚拟主机来指定使用 ws 或 wss 协议。比如，-e VIRTUAL_HOST="ws://ws.domain.com, wss://wss.domain.com"。
- 鉴于 Websocket 本身为 TCP 连接，因此您可以尝试使用本文档前面所介绍的 TCP 负载均衡。

使用场景示例

说明：

一下示例中的acs/proxy镜像均是registry.aliyuncs.com/acs/proxy:0.5的简写形式，请做相应替换。

我的 webapp 容器暴露了 8080 端口（或任意其它端口），我希望 proxy 监听 80 端口

使用下面的命令：

```
docker run -d --expose 80 --name webapp dockercloud/hello-world
docker run -d --link webapp:webapp -p 80:80 acs/proxy
```

我的 webapp 容器暴露了 80 端口和 8083/8086 数据库端口，我希望 proxy 监听 80 端口并且不希望将数据库端口加入 acs/proxy

```
docker run -d -e EXCLUDE_PORTS=8803,8806 --expose 80 --expose 8033 --
expose 8086 --name webapp dockercloud/hello-world
docker run -d --link webapp:webapp -p 80:80 acs/proxy
```

我的容器暴露了 8080 端口（或任意其它端口），我希望 acs/proxy 监听 8080 端口

使用下面的命令：

```
docker run -d --expose 8080 --name webapp your_app
```

```
docker run -d --link webapp:webapp -p 8080:80 acs/proxy
```

我希望 acs/proxy 处理 SSL 连接并将我的普通 HTTP 请求转发到端口 8080（或任意其它端口）

使用下面的命令：

```
docker run -d -e SSL_CERT="YOUR_CERT_TEXT" --name webapp dockercloud/
hello-world
docker run -d --link webapp:webapp -p 443:443 -p 80:80 acs/proxy
```

或者

```
docker run -d --link webapp:webapp -p 443:443 -p 80:80 -e DEFAULT_SS
L_CERT="YOUR_CERT_TEXT" acs/proxy
```

`YOUR_CERT_TEXT` 中的证书包含私钥和公钥（其中，私钥在前，公钥在后）。您需要在证书的行与行之间加上 `\n`。假设您的证书保存在 `~/cert.pem` 中，您可以运行以下命令。

```
docker run -d --link webapp:webapp -p 443:443 -p 80:80 -e DEFAULT_SS
L_CERT="$(awk 1 ORS='\\n' ~/cert.pem)" acs/proxy
```

我希望 acs/proxy 终结 SSL 连接并将 HTTP 请求重定向为 HTTPS 请求

使用下面的命令：

```
docker run -d -e FORCE_SSL=yes -e SSL_CERT="YOUR_CERT_TEXT" --name
webapp dockercloud/hello-world
docker run -d --link webapp:webapp -p 443:443 acs/proxy
```

我希望从数据卷中加载我的 SSL 证书，而不是通过环境变量来传递证书内容

您可以使用 `CERT_FOLDER` 环境变量来指定证书挂载在容器的哪个文件夹中。使用以下命令。

```
docker run -d --name webapp dockercloud/hello-world
docker run -d --link webapp:webapp -e CERT_FOLDER="/certs/" -v $(pwd)/
cert1.pem:/certs/cert1.pem -p 443:443 acs/proxy
```

我希望通过域名设置虚拟主机路由

可以通过 proxy 读取依赖的容器的环境变量（`VIRTUAL_HOST`）来配置虚拟主机。

示例：

```
docker run -d -e VIRTUAL_HOST="www.webapp1.com, www.webapp1.org" --
name webapp1 dockercloud/hello-world
docker run -d -e VIRTUAL_HOST=www.webapp2.com --name webapp2 your/
webapp2
```

```
docker run -d --link webapp1:webapp1 --link webapp2:webapp2 -p 80:80
acs/proxy
```

在上面的例子中，当您访问 `http://www.webapp1.com` 或 `http://www.webapp1.org` 时，将显示运行于容器 `webapp1` 上的服务，`http://www.webapp2.com` 会连接到容器 `webapp2`。

如果您使用下面的命令：

```
docker run -d -e VIRTUAL_HOST=www.webapp1.com --name webapp1
dockercloud/hello-world
docker run -d -e VIRTUAL_HOST=www.webapp2.com --name webapp2-1
dockercloud/hello-world
docker run -d -e VIRTUAL_HOST=www.webapp2.com --name webapp2-2
dockercloud/hello-world
docker run -d --link webapp1:webapp1 --link webapp2-1:webapp2-1 --link
 webapp2-2:webapp2-2 -p 80:80 acs/proxy
```

当您访问 `http://www.webapp1.com` 时，将显示运行于容器 `webapp1` 上的服务， `http://www.webapp2.com` 将基于轮询调度算法（或者任何其它 `BALANCE` 指定的算法）连接到容器 `webapp2-1` 和 `webapp2-2` 上。

我希望我所有的 `*.node.io` 域名均指向我的服务

```
docker run -d -e VIRTUAL_HOST="*.node.io" --name webapp dockercloud/
hello-world
docker run -d --link webapp:webapp -p 80:80 acs/proxy
```

我希望 `web.domain.com` 连接到一个服务， `*.domain.com` 连接到另外一个服务

```
docker run -d -e VIRTUAL_HOST="web.domain.com" -e VIRTUAL_HOST_WEIGHT=
1 --name webapp dockercloud/hello-world
docker run -d -e VIRTUAL_HOST="*.domain.com" -e VIRTUAL_HOST_WEIGHT=0
 --name app dockercloud/hello-world
docker run -d --link webapp:webapp --link app:app -p 80:80 acs/proxy
```

我希望所有到 `/path` 路径的请求均指向我的服务

```
docker run -d -e VIRTUAL_HOST="*/path, */path/*" --name webapp
dockercloud/hello-world
```

```
docker run -d --link webapp:webapp -p 80:80 acs/proxy
```

我希望所有的静态 **html** 请求均指向我的服务

```
docker run -d -e VIRTUAL_HOST="*/*.htm, */*.html" --name webapp
dockercloud/hello-world docker run -d --link webapp:webapp -p 80:80
acs/proxy
```

我希望查看 **HAProxy stats**

```
docker run -d --link webapp:webapp -e STATS_AUTH="auth:auth" -e
STATS_PORT=1936 -p 80:80 -p 1936:1936 acs/proxy
```

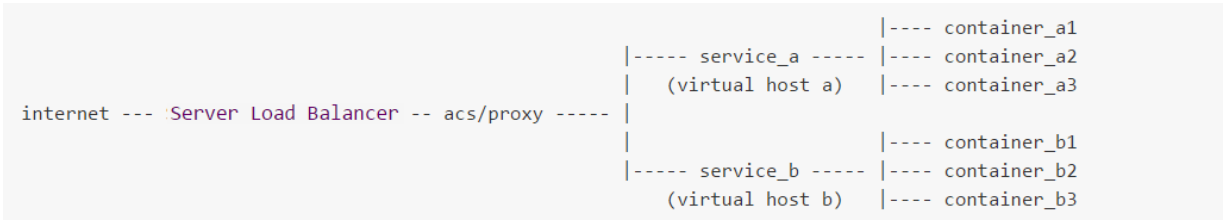
我希望将我的日志发送到 **papertrailapp**

用与您的 papertrailapp 账号相匹配的值替换 `<subdomain>` 和 `<port>`。

```
docker run -d --name web1 dockercloud/hello-world
docker run -d --name web2 dockercloud/hello-world
docker run -it --env RSYSLOG_DESTINATION='<subdomain>.papertrailapp.
com:<port>' -p 80:80 --link web1:web1 --link web2:web2 acs/proxy
```

使用虚拟主机的拓扑图

图 **5-1:** 阿里云容器服务代理拓扑图



手动重新加载 **HAProxy**

在大多数情况下，当依赖的服务发生变化时，`acs/proxy` 会自动进行配置。但是如果需要的话，您也可以按照下面的方法手动重新加载 `acs/proxy`。

```
docker exec <acs/proxy_container_id> /reload.sh
```

5.9 自定义路由-简单示例

此示例会部署一个*acs/proxy* 容器，对外通过负载均衡实例（使用*lb* 标签）暴露服务，同时在后端挂载一个 nginx，本示例只展示 nginx 的首页，但是会在基本示例的基础上，逐步增加其他功能。

说明：

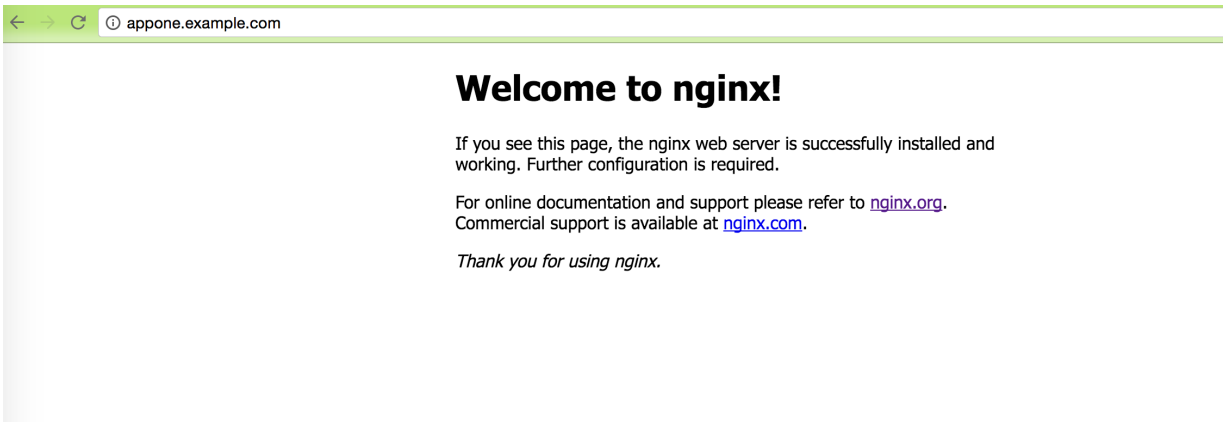
任何两个不同的服务均不能共享使用同一个负载均衡，否则会导致负载均衡后端机器被删除，服务不可用。

基本示例

compose 模板如下所示。

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
appone:
    expose:  # 被代理的服务一定要使用 expose 或者 ports 告诉 proxy 容器暴露哪个端口
        -  80/tcp
    image:  'nginx:latest'
    labels:
        #  此处支持 http/https/ws/wss 协议，必须使用用户自己的域名而不是容器服务提供的测试域名
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
    restart:  always
```

服务启动成功后，页面如下图所示。



开启会话保持

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
```

```
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        #  此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
        # 此处支持会话保持，使用 cookie 的方式，键为 CONTAINERID
        aliyun.proxy.COOKIE: "CONTAINERID insert indirect"
    restart:  always
```

自定义 503 页面

当直接输入负载均衡的 VIP 地址而不是域名时，访问的页面如下图所示，返回 503 错误页面。

如果希望在 503 页面增加一些提示信息，可以在容器所在的 vm 中增加文件夹 `/errors`，同时增加文件 `/errors/503.http`，文件内容如下：

```
HTTP/1.0 503 Service Unavailable
Cache-Control: no-cache
Connection: close
Content-Type: text/html;charset=UTF-8
<html><body><h1>503 Service Unavailable</h1>
<h3>No server is available to handle this request.</h3>
<h3>如果您看到这个页面，说明您访问服务出现了问题，请按如下步骤排查解决问题：</h3>
<li>如果您是应用的访问者，请寻求应用的维护者解决问题。</li>
<li>如果您是应用的维护者，继续查看下面的信息。</li>
<li>您现在使用的是简单路由服务，整个请求会从SLB -> acsrouting应用容器 -> 您的应用容器, 请按下列步骤排查。</li>
```

```
<li>请登录容器服务控制台，在左侧边栏选择"服务"，在"服务列表"选择相应的"集群"，单击暴露到公网的"服务"，查看服务的"访问端点"，仔细检查您的访问域名与在相应的服务中配置的域名是否一致。</li>
<li>按照<a href="https://help.aliyun.com/knowledge_detail/42660.html">简单路由服务链路排查</a>进行问题定位和排查。</li>
<li>查看<a href="https://help.aliyun.com/knowledge_detail/42658.html">路由常见问题文档</a>。</li>
<li>如果上面的方式还没有解决您的问题，请<a href="https://workorder.console.aliyun.com/#/ticket/add?productId=1254">提交工单</a>，联系技术人员协助解决，我们会竭诚为您服务。</li>
</body></html>
```

您可以修改为您需要展示的错误页面。修改 compose 模板如下：

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
        EXTRA_FRONTEND_SETTINGS_80: "errorfile 503 /usr/local/etc/haproxy/errors/503.http"
    volumes:
        - /errors/:/usr/local/etc/haproxy/errors/
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        #  配置 URL，支持指定 path，此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
    restart:  always
```

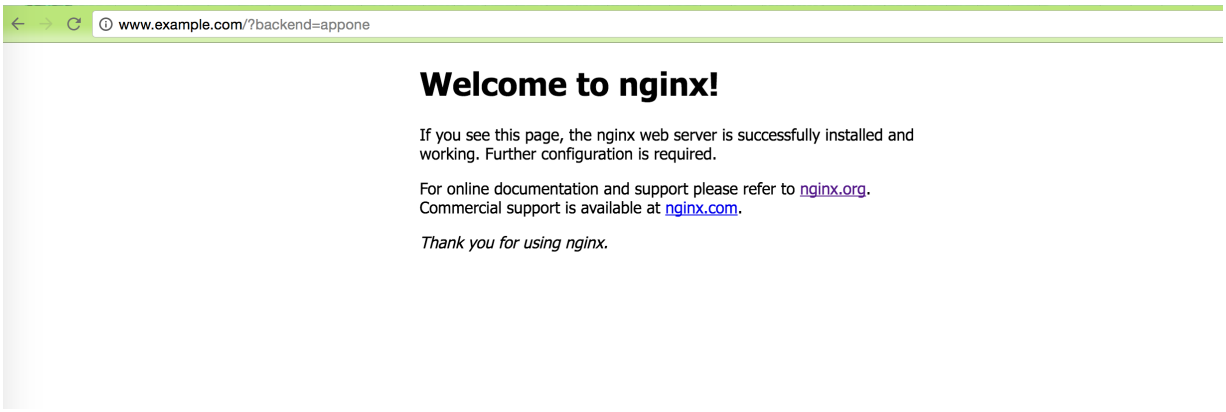
输入负载均衡的 VIP 地址之后，显示的 503 页面如下图所示。

支持泛域名

如果希望 nginx 的后端支持泛域名，即 `appone.example.com` 能访问到 nginx 首页，`*.example.com` 也能访问到 nginx 的首页，修改配置如下即可。

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
        EXTRA_FRONTEND_SETTINGS_80: "errorfile 503 /usr/local/etc/
haproxy/errors/503.http"
    volumes:
        - /errors/:/usr/local/etc/haproxy/errors/
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        #  配置 URL，支持指定 path，此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://*.example.com"
    restart:  always
```

绑定 host，输入域名 `www.example.com` 之后，显示 nginx 首页如下图所示。

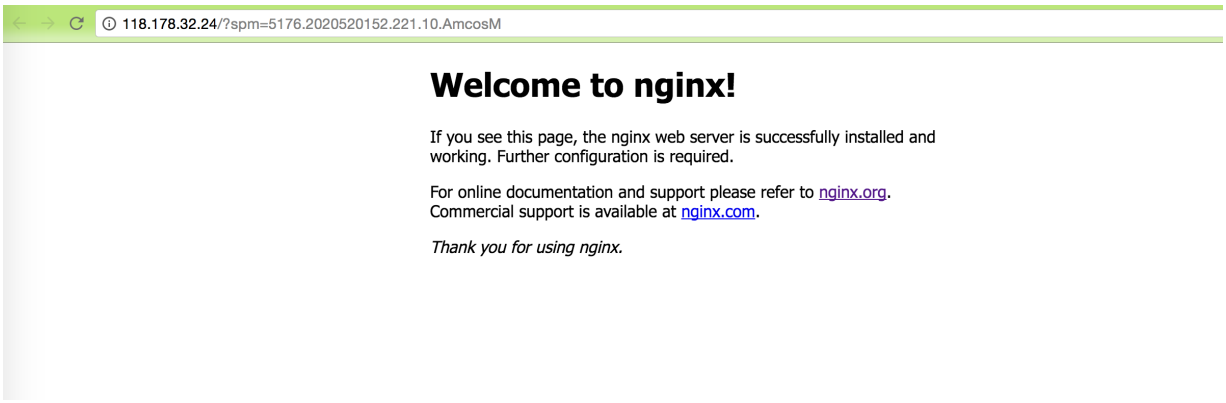


配置默认的后端

如果希望直接通过 IP 也能访问到后端的 nginx，把 URL 配置去掉，修改配置如下即可。

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*" 表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
        # 返回 503 时，指定错误页面
        EXTRA_FRONTEND_SETTINGS_80: "errorfile 503 /usr/local/etc/
haproxy/errors/503.http"
    volumes:
        # 从主机挂载错误页面到容器内部
        - /errors/:/usr/local/etc/haproxy/errors/
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        # 表明该服务需要通过 proxy 来代理
        aliyun.proxy.required: "true"
    restart:  always
```

输入负载均衡的 VIP 之后，显示 nginx 首页如下图所示。



根据 URL 参数值选择后端

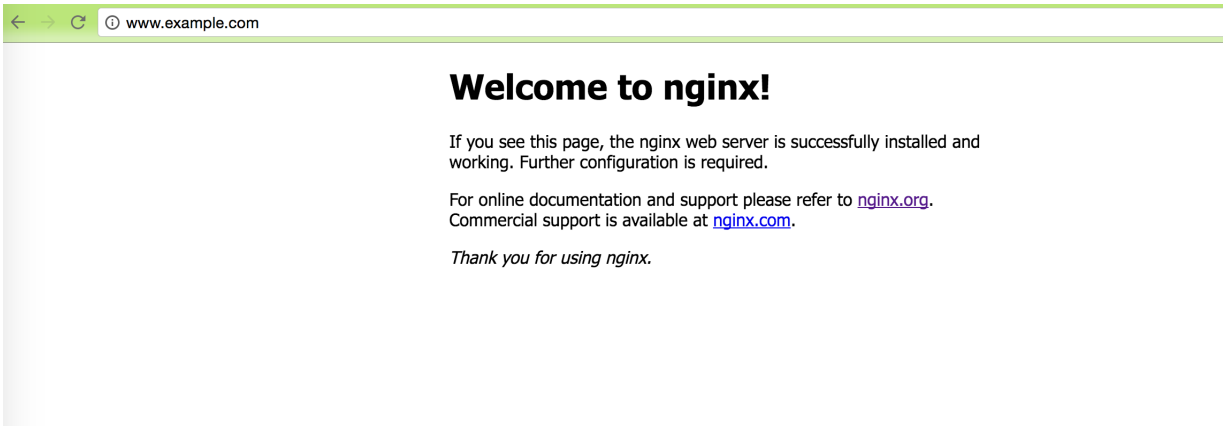
您也可以根据 URL 参数值的不同来代理不同的后端。

以下示例将通过 `http://www.example.com?backend=appone` 访问服务 appone，即 nginx 首页，通过 `http://www.example.com?backend=apptwo` 访问服务 apptwo，即 hello world 首页。应用模板代码如下所示。

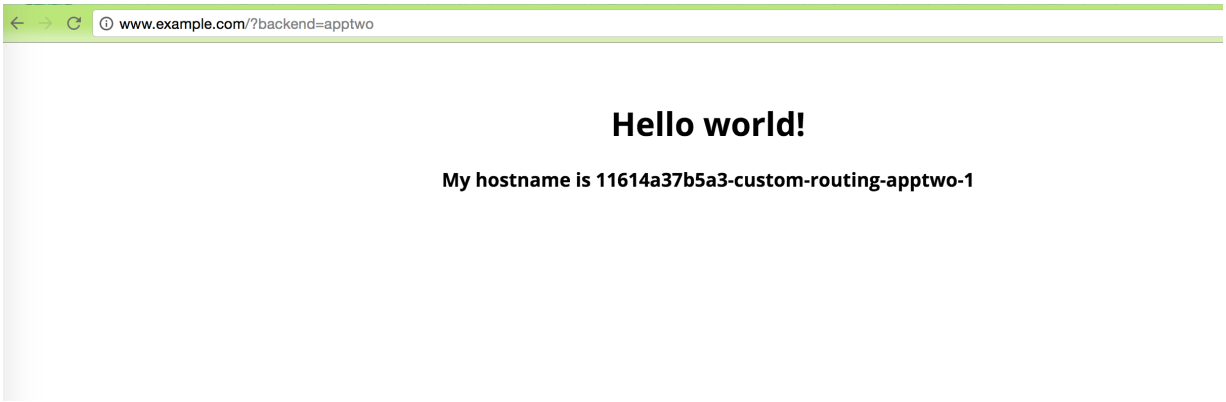
```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
          -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
        #  获取 URL 中参数名称为 backend 的参数值，同时将 HOST header 修改为要匹配的后端的域名
        EXTRA_FRONTEND_SETTINGS_80:  "  http-request set-header HOST %[urlp(backend)].example.com"
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        #   配置 URL，支持指定 path，此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
    restart:  always
apptwo:
    ports:
        -  80/tcp
    image:  'registry.cn-hangzhou.aliyuncs.com/linhuatest/hello-world:latest'
    labels:
```

```
        #   配置 URL，支持指定 path，此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://apptwo.example.com"
    restart:  always
```

绑定 host，输入链接 `http://www.example.com?backend=appone`，显示服务 appone 的 nginx 首页，如下所示。



绑定 host，输入链接 `http://www.example.com?backend=apptwo`，显示服务 apptwo 的 hello world 首页，如下所示。



记录访问日志

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
```

```
        EXTRA_DEFAULT_SETTINGS: "log rsyslog local0,log global,option
httplog"
    links:
        - rsyslog:rsyslog
rsyslog:
    image: registry.cn-hangzhou.aliyuncs.com/linhuatest/rsyslog:latest
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        #  此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
    restart:  always
```

日志会直接打到 rsyslog 容器的标准输出中，通过 `docker logs $rsyslog_container_name` 即能看到自定义路由的访问日志。

服务间的负载均衡

以下模板创建一个负载均衡服务 `lb` 和一个应用服务 `appone`，整体对外提供域名为 `appone.example.com` 的服务。

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    hostname: proxy  # 指定该服务的域名为 proxy，即 proxy 会解析到所有部署了
该镜像的容器
    ports:
            -  '80:80'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
appone:
    ports:
        -  80/tcp
        -  443/tcp
    image:  'nginx:latest'
    labels:
        #  此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
```

```
    restart:  always
```

以下模板作为一个客户端，访问应用服务 appone，但是它的访问路径是请求访问负载均衡服务 lb，然后再反向代理到应用服务 appone 的。

```
restclient: # 模拟 rest 服务消费者
  image: registry.aliyuncs.com/acs-sample/alpine:3.3
  command: "sh -c 'apk update; apk add curl; while true; do curl --
head http://appone.example.com; sleep 1; done'" #访问 rest 服务，测试负载
均衡
  tty: true
  external_links:
    - "proxy:appone.example.com" #指定 link 的服务的域名，以及该域名的别名
```

在服务 restclient 的容器中，您会发现域名 appone.example.com 是解析到负载均衡服务 lb 的所有容器 IP 的。

```
/ # drill appone.example.com
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 60917
;; flags: qr rd ra ; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;; appone.example.com.  IN      A
;; ANSWER SECTION:
appone.example.com.     600     IN      A       172.18.3.4
appone.example.com.     600     IN      A       172.18.2.5
appone.example.com.     600     IN      A       172.18.1.5
;; AUTHORITY SECTION:
;; ADDITIONAL SECTION:
;; Query time: 0 msec
;; SERVER: 127.0.0.11
;; WHEN: Mon Sep 26 07:09:40 2016
;; MSG SIZE  rcvd: 138
```

配置监控页面示例

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
            -  '127.0.0.1:1935:1935' # 监控页面对公网暴露的端口，有安全风
险，请谨慎配置
    restart:  always
    labels:
        aliyun.custom_addon:  "proxy"
        aliyun.global:  "true"
        aliyun.lb.port_80: tcp://proxy_test:80
    environment:
        ADDITIONAL_SERVICES:  "*"
        STATS_AUTH: "admin:admin" # 监控时用于登录的账号和密码，可以自定义
        STATS_PORT: "1935" # 监控使用的端口，可以自定义
appone:
    expose:
        -  80/tcp
    image:  'nginx:latest'
    labels:
```

```
        aliyun.proxy.VIRTUAL_HOST:  "http://appone.example.com"
    restart:  always
```

登录到自定义路由镜像所在的每一台机器（每一台机器都可能接收请求，不管应用容器在哪台机器上），请求 acs/proxy健康检查页面。

说明：

按照应用模版的 STATS_AUTH 环境变量配置正确的用户名和密码。

```
root@c68a460635b8c405e83c052b7c2057c7b-node2:~# curl -Ss -u admin:
admin 'http://127.0.0.1:1935/' &> test.html
```

将页面 test.html 拷贝到有浏览器的机器，用浏览器打开本地文件 test.html，查看 stats 监控统计页面，显示为绿色，表示 acs/proxy 容器到后端容器的网络是连通的，在正常工作；显示为其他颜色则为异常。

5.10 自定义路由-支持 TCP 协议

阿里云容器服务在使用的过程中，针对 TCP 负载均衡的场景，会遇到这样的问题：如果一个应用的客户端镜像和服务端镜像均部署在同一个节点（ECS）上面，由于受负载均衡的限制，该应用的客户端不能通过负载均衡访问本机的服务端。本文档以常用的基于 TCP 协议的 redis 为例，通过自定义路由*acs/proxy* 来解决这一问题。

说明：

任何两个不同的服务均不能共享使用同一个负载均衡，否则会导致负载均衡后端机器被删除，服务不可用。

解法一：通过调度容器，避免客户端和服务端容器部署在同一个节点

示例应用模板（使用了*lb* 标签和 *swarm filter* 功能）：

```
redis-master:
    ports:
      - 6379:6379/tcp
    image:  'redis:alpine'
    labels:
        aliyun.lb.port_6379: tcp://proxy_test:6379
redis-client:
    image:  'redis:alpine'
    links:
      - redis-master
    environment:
      - 'affinity:aliyun.lb.port_6379!=tcp://proxy_test:6379'
    command: redis-cli -h 120.25.131.64
    stdin_open: true
```

```
    tty: true
```

说明：

- 如果发现调度不生效，在容器服务管理控制台，单击左侧导航栏的服务进入服务列表页面，选择您需要调度的服务，单击重新调度 > 在弹出的对话框中勾选强制重新调度，单击确定。
- 强制重新调度会丢弃已有容器的 volume，请做好相应的备份迁移工作。

解法二：容器集群内部客户端使用 **link** 访问服务端，集群外部使用负载均衡

示例应用模板（使用了*lb*标签）：

```
redis-master:
    ports:
      - 6379:6379/tcp
    image:  'redis:alpine'
    labels:
        aliyun.lb.port_6379: tcp://proxy_test:6379
redis-client:
    image:  'redis:alpine'
    links:
      - redis-master
    command: redis-cli -h redis-master
    stdin_open: true
    tty: true
```

解法三：容器集群内部客户端使用自定义路由（基于 **HAProxy**）作为代理访问服务端，集群外部使用负载均衡

示例应用模板（使用了*lb*标签和自定义路由-简单示例）：

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '6379:6379/tcp'
    restart:  always
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡，使用 lb 标签
        aliyun.lb.port_6379: tcp://proxy_test:6379
        # 告诉系统，自定义路由需要等待 master 和 slave 启动之后再启动，并且对
master 和 slave 有依赖
        aliyun.depends: redis-master,redis-slave
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
        EXTRA_DEFAULT_SETTINGS: "log rsyslog local0,log global,option
httplog"
        # 配置 HAProxy 工作于 tcp 模式
        MODE: "tcp"
```

```
    links:
        - rsyslog:rsyslog
rsyslog:
    image: registry.cn-hangzhou.aliyuncs.com/linhuatest/rsyslog:latest
redis-master:
    ports:
      - 6379/tcp
    image:  'redis:alpine'
    labels:
        # 告诉自定义路由需要暴露 6379 端口
        aliyun.proxy.TCP_PORTS:  "6379"
        # 告诉系统，该服务的路由需要添加到自定义路由服务中
        aliyun.proxy.required: "true"
redis-slave:
    ports:
      -  6379/tcp
    image:  'redis:alpine'
    links:
      - redis-master
    labels:
      # 告诉自定义路由需要暴露 6379 端口
      aliyun.proxy.TCP_PORTS:  "6379"
      # 告诉系统，该服务的路由需要添加到自定义路由服务中
      aliyun.proxy.required: "true"
      # 告诉系统，slave 需要等待 master 启动之后再启动，并且对 master 有依赖
      aliyun.depends: redis-master
    command: redis-server --slaveof redis-master 6379
redis-client:
    image:  'redis:alpine'
    links:
      - lb:www.example.com
    labels:
      aliyun.depends: lb
    command: redis-cli -h www.example.com
    stdin_open: true
    tty: true
```

该解决方案，做到了 redis 的主从架构，同时经过服务间的负载均衡 做负载均衡，做到了一定程度的高可用。

5.11 自定义路由-支持多 HTTPS 证书

本示例使用的镜像是*acs/proxy* 镜像。

说明：

任何两个不同的服务均不能共享使用同一个负载均衡，否则会导致负载均衡后端机器被删除，服务不可用。

```
lb:
    image:  registry.aliyuncs.com/acs/proxy:0.5
    ports:
            -  '80:80'
            - '443:443' # https 必须暴露这个端口
    restart:  always
```

```
    labels:
        # addon 使得 proxy 镜像有订阅注册中心的能力，动态加载服务的路由
        aliyun.custom_addon:  "proxy"
        # 每台 vm 部署一个该镜像的容器
        aliyun.global:  "true"
        #  前端绑定负载均衡
        aliyun.lb.port_80: tcp://proxy_test:80
        aliyun.lb.port_443:tcp://proxy_test:443
    environment:
        #  支持加载路由的后端容器的范围，"*"表示整个集群，默认为应用内的服务
        ADDITIONAL_SERVICES:  "*"
appone:
    expose:  # 被代理的服务一定要使用 expose 或者 ports 告诉 proxy 容器暴露哪
个端口
        -  80/tcp
    image:  'nginx:latest'
    labels:
        #   配置 URL，支持指定 path，此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "https://appone.example.com"
        #  配置 appone 的证书
        aliyun.proxy.SSL_CERT: "-----BEGIN RSA PRIVATE KEY-----\
nMIIEpQIBAAKCAQEAvgnKhephWHKWYDEiBiSjzst7nRP0DJxZ5cIOxyXmncd2kslr
\nkUIB5qT/MSiJGBL3Lr4advs6kI/JFmxloFrPtwEe2FGkLBfCDXXDrWgxyFhbuPQY
\nBLNueUu94sffIxg+4u5Mriui7ftindOAf0d21PSM9gb/ZUypxIgAd3RHCe/gtT0h
\nVCn6FikXynXLDTODYWCthQHBwSZS88HNU+B0T9Yl65JiQ0mV+YF+h3D/c232E6Gp
\nzK+8ehVB13s5hecUx3dvdUQPBUhJYvzsPjChgsXSMDRexiN66kbhH6dJArsrYb8t
\nEBWXfCZaTcF82wkAsUe/fhlGhh97h+66lh6OQQIDAQABAoIBAQC4d8ifNWRI9vIB
\nbbAZRne7xMm5MCU2GI8q97Rgm+nAPl5bHinMVsaBnKgaj76EH+TQ+re1xyiSKwCH
\nQ7FidsQqYGwQjy9NncJATpAjQ4EPeLWQU2D9Ly+NjnhEKr/u0Ro6LhdA+hqt59dS
\nXHvfEP/It5odN62yJzikDWBmk/hhK0tu28dPYUuPoWswXWFMkaNttmfLgZlagiqr
\nYp7rxAFqQurzctQ2VNwezekDHQoh8ounHGEniZ+fA6sFtYi83KTKWkvFom1chZQr
\nxxPbbgANJJJlNgtkl6JZNxj6SYimmWvzmrrU25khKg/klP5EtQzIx6UFhURnuTKu
\nzNgqcIABAoGBAOqUOerveEUePvsAlta8CV/p2KKwenv+kUofQ4UpKFXfnHbQHQfr
\nZHS29OQiPxqjVXYLu8gNfLRfKtUNyqV+TDrzJ1elW2RKc00GHAwPbXxijPhmJ2fW
\neskn8tlDcyXpvoqWJG34896vo4IbcL0H/eUs0jJo6OJlCQBKXik+t3gxAoGBAM9k
\nVOTV2caKyrZ4ta0Q1LKqKfOkt0j+vKz167J5pSLjVKQSUxGMyLnGwiQdDtB4iy6L
\nFcCB/S0HM0UWkJWhNYAL8kHry53bVdHtQG0tuYFYvBJo7A+Nppsn9MtlVh8KbVu4
\nhOz/3MWwbQNnvIVCGK/fSltS1GhTk4rKL7PjNwMRAoGBALK0n3bqXj6Rrzs7FK6c
\na6vlE4PFXFpv8jF8pcyhMThSdPlSzHsHCe2cn+3YZSie+/FFORZLqBAlXBUZP6Na
\nFyrlqLgtofVCfppUKDPL4QXccjaeZDDIBZyPUYPQzb05WE5t2WzqNqcUOUVaMEXh
\n+7uGrM94espWXEgbX6aeP9lRAoGARlJQ7t8MXuQE5GZ9w9cnKAXG/9RkSZ4Gv+cL
\nKpNQyUmoE5IbFKJWFZgtkC1CLrIRD5EdqQ7ql/APFGgYUoQ9LdPfKzcW7cnHic0W
\nwW51rkQ2UU++a2+uhIHB4Y3U6+WPO0CP4gTICUhPTo5IQC8vS8M85UZqu41LRA5W
\nqnpq1uECgYEAq+6KpHhlR+5h3Y/m0n84yJ0YuCmrl7HFRzBMdOcaW3oaYL83rAaq
\n6dJqpAVgeu3HP8AtiGVZRe78J+n4d2JGYSqgtP2lFFTdF9HfhcR2P9bUBNYtWols
\nEs3iw53t8a4BndLGBwLPA3lklf7J5stYanRv6NqaRaLq4FQMxsW1A0Q=\n-----
END RSA PRIVATE KEY-----\n-----BEGIN CERTIFICATE-----\nMIIDvDCCA
qSgAwIBAgIBATANBgkqhkiG9w0BAQUFADBgMQswCQYDVQQGEwJDTjER\nMA8GA1UEC
BMIWmhlamlhbmcxETAPBgNVBAcTCEhhbmd6aG91MRQwEgYDVQQKEwth\nbGliYWJhL
mNvbTEVMBMGA1UECxMMd3d3LnJvb3QuY29tMB4XDTE1MDIwOTA1MzQx\nOFoXDTE2M
DIwOTA1MzQxOFowZjELMAkGA1UEBhMCQ04xETAPBgNVBAgTCFpoZWpp\nYW5nMRQwE
gYDVQQKEwthbGliYWJhLmNvbTEVMBMGA1UECxMMd3d3LnJvb3QuY29t\nMRcwFQYDV
QQDEw53d3cubGluaHVhLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP\nADCCAQoCg
gEBAL4JyoXqYVhylmAxIgYko87Le50T9AycWeXCDscl5p3HdpLJa5FC\nAeak/
zEoiRgS9y6+Gnb7OpCPyRZsZaBaz7cBHthRpCwXwg11w61oMchYW7j0GASz\nbnlLveLH3
yMYPuLuTK4rou37Yp3TgH9HdtT0jPYG/2VMqcSIAHd0Rwnv4LU9IVQp\n+hYpF8p1yw0
zg2FgrYUBwcEmUvPBzVPgdE/WJeuSYkNJlfmBfodw/3Nt9hOhqcyv\nvHoVQdd7O
YXnFMd3b3VEDwVISWL87D4woYLF0jA0XsYjeupG4R+nSQK7K2G/LRAV\nl3wmWk3Bf
NsJALFHv34ZRoYfe4fuupYejkECAwEAAaN7MHkwCQYDVR0TBAIwADAs\nBglghkgBh
vhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYD\nVR0OBBYEF
M6ESmkDKrqnqMwBawkjeONKrRMQMB8GA1UdIwQYMBaAFFUrhN9ro+Nm\nrZnl4WQzD
```

```
pgTbCBhMA0GCSqGSIb3DQEBBQUAA4IBAQCQ2D9CRiv8brx3fnr/RZG6\nFYPEdxjY/
CyfJrAbij0PdKjzZKk1O67chM1Oxs2JhJ6tMqg2sv50bGx4XmbSPmEe\nYTJjIXMY+jCoJ
/Zmk3Xgu4K1y1LvD25PahDVhRrPN8H4WjsYu51pQNshil5E/3iQ\n2JoV0r8QiAsPiiY5
+mNCD1fm+QN1tyUabczi/DHafgWJxf2B3M66e3oUdtbzA2pf\nYHR8RVeSFrjaBqudO8i
r+uYcRbRkroYmY5Vm+4Yp64oetrPpKUPWSYaAZ0uRtpeL\nB5DpqXz9GEBb5m2Q4dK
js5Hm6vyFUORCzZcO4XexDhcgdLOH5qznmh9oMCk9QvZf\n-----END CERTIFICATE
-----\n"
    restart:  always
apptwo:
    expose:  # 被代理的服务一定要使用 expose 或者 ports 告诉 proxy 容器暴露哪
个端口
        -  80/tcp
    image:  'registry.cn-hangzhou.aliyuncs.com/linhuatest/hello-world:
latest'
    labels:
        #   配置 URL，支持指定 path，此处支持 http/https/ws/wss 协议
        aliyun.proxy.VIRTUAL_HOST:  "https://apptwo.example.com"
        #  配置 apptwo 的证书
        aliyun.proxy.SSL_CERT: "-----BEGIN RSA PRIVATE KEY-----\
nMIIEpQIBAAKCAQEAvgnKhephWHKWYDEiBiSjzst7nRP0DJxZ5cIOxyXmncd2kslr
\nkUIB5qT/MSiJGBL3Lr4advs6kI/JFmxloFrPtwEe2FGkLBfCDXXDrWgxyFhbuPQY
\nBLNueUu94sffIxg+4u5Mriui7ftindOAf0d21PSM9gb/ZUypxIgAd3RHCe/gtT0h
\nVCn6FikXynXLDTODYWCthQHBwSZS88HNU+B0T9Yl65JiQ0mV+YF+h3D/c232E6Gp
\nzK+8ehVB13s5hecUx3dvdUQPBUhJYvzsPjChgsXSMDRexiN66kbhH6dJArsrYb8t
\nEBWXfCZaTcF82wkAsUe/fhlGhh97h+66lh6OQQIDAQABAoIBAQC4d8ifNWRI9vIB
\nbbAZRne7xMm5MCU2GI8q97Rgm+nAPl5bHinMVsaBnKgaj76EH+TQ+re1xyiSKwCH
\nQ7FidsQqYGwQjy9NncJATpAjQ4EPeLWQU2D9Ly+NjnhEKr/u0Ro6LhdA+hqt59dS
\nXHvfEP/It5odN62yJzikDWBmk/hhK0tu28dPYUuPoWswXWFMkaNttmfLgZlagiqr
\nYp7rxAFqQurzctQ2VNwezekDHQoh8ounHGEniZ+fA6sFtYi83KTKWkvFom1chZQr
\nxxPbbgANJJJlNgtkl6JZNxj6SYimmWvzmrrU25khKg/klP5EtQzIx6UFhURnuTKu
\nzNgqcIABAoGBAOqUOerveEUePvsAlta8CV/p2KKwenv+kUofQ4UpKFXfnHbQHQfr
\nZHS29OQiPxqjVXYLu8gNfLRfKtUNyqV+TDrzJ1elW2RKc00GHAwPbXxijPhmJ2fW
\neskn8tlDcyXpvoqWJG34896vo4IbcL0H/eUs0jJo6OJlCQBKXik+t3gxAoGBAM9k
\nVOTV2caKyrZ4ta0Q1LKqKfOkt0j+vKz167J5pSLjVKQSUxGMyLnGwiQdDtB4iy6L
\nFcCB/S0HM0UWkJWhNYAL8kHry53bVdHtQG0tuYFYvBJo7A+Nppsn9MtlVh8KbVu4
\nhOz/3MWwbQNnvIVCGK/fSltS1GhTk4rKL7PjNwMRAoGBALK0n3bqXj6Rrzs7FK6c
\na6vlE4PFXFpv8jF8pcyhMThSdPlSzHsHCe2cn+3YZSie+/FFORZLqBAlXBUZP6Na
\nFyrlqLgtofVCfppUKDPL4QXccjaeZDDIBZyPUYPQzb05WE5t2WzqNqcUOUVaMEXh
\n+7uGrM94espWXEgbX6aeP9lRAoGARlJQ7t8MXuQE5GZ9w9cnKAXG/9RkSZ4Gv+cL
\nKpNQyUmoE5IbFKJWFZgtkC1CLrIRD5EdqQ7ql/APFGgYUoQ9LdPfKzcW7cnHic0W
\nwW51rkQ2UU++a2+uhIHB4Y3U6+WPO0CP4gTICUhPTo5IQC8vS8M85UZqu41LRA5W
\nqnpq1uECgYEAq+6KpHhlR+5h3Y/m0n84yJ0YuCmrl7HFRzBMdOcaW3oaYL83rAaq
\n6dJqpAVgeu3HP8AtiGVZRe78J+n4d2JGYSqgtP2lFFTdF9HfhcR2P9bUBNYtWols
\nEs3iw53t8a4BndLGBwLPA3lklf7J5stYanRv6NqaRaLq4FQMxsW1A0Q=\n-----
END RSA PRIVATE KEY-----\n-----BEGIN CERTIFICATE-----\nMIIDvDCCA
qSgAwIBAgIBATANBgkqhkiG9w0BAQUFADBgMQswCQYDVQQGEwJDTjER\nMA8GA1UEC
BMIWmhlamlhbmcxETAPBgNVBAcTCEhhbmd6aG91MRQwEgYDVQQKEwth\nbGliYWJhL
mNvbTEVMBMGA1UECxMMd3d3LnJvb3QuY29tMB4XDTE1MDIwOTA1MzQx\nOFoXDTE2M
DIwOTA1MzQxOFowZjELMAkGA1UEBhMCQ04xETAPBgNVBAgTCFpoZWpp\nYW5nMRQwE
gYDVQQKEwthbGliYWJhLmNvbTEVMBMGA1UECxMMd3d3LnJvb3QuY29t\nMRcwFQYDV
QQDEw53d3cubGluaHVhLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP\nADCCAQoCg
gEBAL4JyoXqYVhylmAxIgYko87Le50T9AycWeXCDscl5p3HdpLJa5FC\nAeak/
zEoiRgS9y6+Gnb7OpCPyRZsZaBaz7cBHthRpCwXwg11w61oMchYW7j0GASz\nbnlLveLH3
yMYPuLuTK4rou37Yp3TgH9HdtT0jPYG/2VMqcSIAHd0Rwnv4LU9IVQp\n+hYpF8p1yw0
zg2FgrYUBwcEmUvPBzVPgdE/WJeuSYkNJlfmBfodw/3Nt9hOhqcyv\nvHoVQdd7O
YXnFMd3b3VEDwVISWL87D4woYLF0jA0XsYjeupG4R+nSQK7K2G/LRAV\nl3wmWk3Bf
NsJALFHv34ZRoYfe4fuupYejkECAwEAAaN7MHkwCQYDVR0TBAIwADAs\nBglghkgBh
vhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYD\nVR0OBBYEF
M6ESmkDKrqnqMwBawkjeONKrRMQMB8GA1UdIwQYMBaAFFUrhN9ro+Nm\nrZnl4WQzD
pgTbCBhMA0GCSqGSIb3DQEBBQUAA4IBAQCQ2D9CRiv8brx3fnr/RZG6\nFYPEdxjY/
CyfJrAbij0PdKjzZKk1O67chM1Oxs2JhJ6tMqg2sv50bGx4XmbSPmEe\nYTJjIXMY+jCoJ
/Zmk3Xgu4K1y1LvD25PahDVhRrPN8H4WjsYu51pQNshil5E/3iQ\n2JoV0r8QiAsPiiY5
```

```
+mNCD1fm+QN1tyUabczi/DHafgWJxf2B3M66e3oUdtbzA2pf\nYHR8RVeSFrjaBqudO8i
r+uYcRbRkroYmY5Vm+4Yp64oetrPpKUPWSYaAZ0uRtpeL\nB5DpqXz9GEBb5m2Q4dK
js5Hm6vyFUORCzZcO4XexDhcgdLOH5qznmh9oMCk9QvZf\n-----END CERTIFICATE
-----\n"
    restart:  always
```

如上所示，有两个服务 appone 和 apptwo 分别通过 `aliyun.proxy.VIRTUAL_HOST` 来指定二者的域名，注意如果需要配置证书，一定要注明https协议。然后通过 `aliyun.proxy.SSL_CERT` 来指定各自的证书内容，证书内容配置的方法如下。

假设 *`key.pem`* 是私钥文件，*`ca.pem`*是公钥文件，在 bash 中执行如下命令（当前目录包含公钥文件和私钥文件）。

```
$ cp key.pem cert.pem
$ cat ca.pem >> cert.pem
$ awk 1 ORS='\\n' cert.pem
```

最后将 awk 指令的输出作为 label `aliyun.proxy.SSL_CERT` 值填入，注意用英文双引号（”“）分隔。其他内容，例如*lb* 标签等，参考上述模版和相应自定义路由-简单示例。