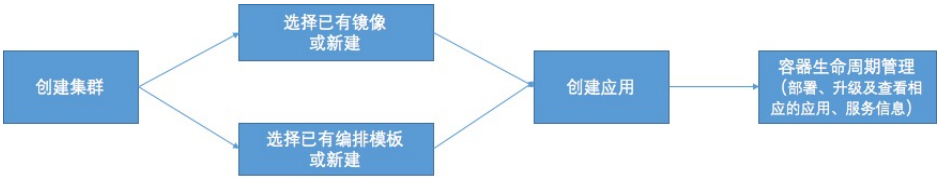
容器服务

用户指南

用户指南

概述

完整的容器服务使用流程包含以下 4 个步骤：



第一步：创建集群，选择集群的网络环境，设置集群的节点个数和配置信息。

第二步：选择镜像或编排模板（如果您的应用由多个镜像组成，可以选择一个编排模板）。

第三步：创建应用并部署。

第四步：查看部署后应用的状态和相应的服务、容器信息。

集群

一个集群指容器运行所需要的云资源组合，关联了若干服务器节点、负载均衡、专有网络等云资源。

节点

一台服务器（可以是虚拟机实例或者物理服务器）已经安装了 Docker Engine，可以用于部署和管理容器；容器服务的 Agent 程序会安装到节点上并注册到一个集群上。集群中的节点数量可以伸缩。

容器

一个通过 Docker 镜像部署的实例，一个节点可运行多个容器。

镜像

Docker 镜像是容器应用打包的标准格式，在部署容器化应用时可以指定镜像，镜像可以来自于 Docker

Hub，阿里云容器 Hub，或者用户的私有 Registry。镜像 ID 可以由镜像所在仓库 URI 和镜像 Tag（缺省为 latest）唯一确认。

编排模板

编排模板包含了一组容器服务的定义和其相互关联，可以用于多容器应用的部署和管理。容器服务支持 Docker Compose 模板规范并有所扩展。

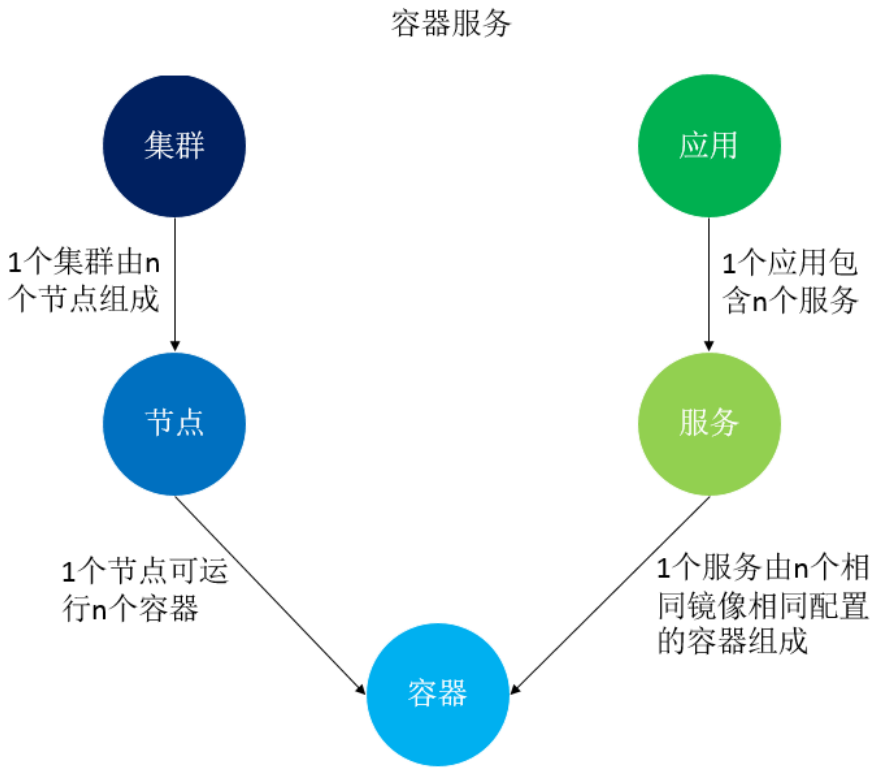
应用

一个应用可通过单个镜像或一个编排模板创建，每个应用可包含一个或多个服务。

服务

一组基于相同镜像和配置定义的容器，作为一个可伸缩的微服务。

关联关系



集群管理

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。

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。**

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。**

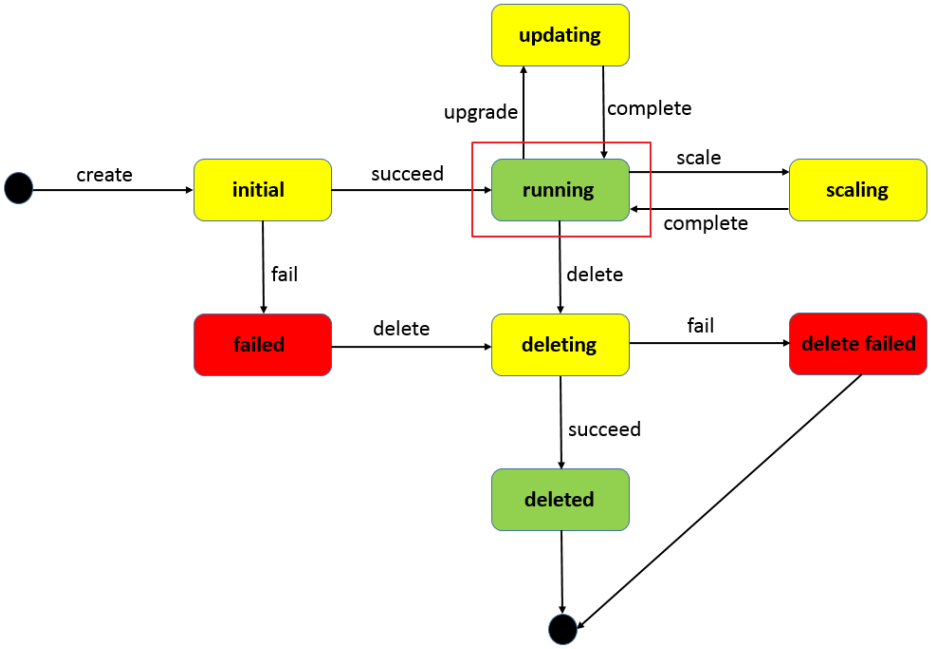
集群管理

支持集群扩容、删除、清理、连接等操作。

集群状态说明

状态	说明
创建中（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
升级中（Updating）	集群升级服务端版本
集群规模变更中（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（DeleteFailed）	集群删除失败
已删除（Deleted，该状态用户不可见）	集群删除成功

集群状态流转



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。

注意：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“就绪”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。**

限制说明

- 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。

操作流程

登录 **容器服务管理控制台**。

单击左侧导航中的 **集群**，单击右上角的 **创建集群**。

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 1~64 个字符，包括数字，中文字符，英文字符和连字符（-）。
  注意： 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。
  注意：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。

设置集群的网络类型。

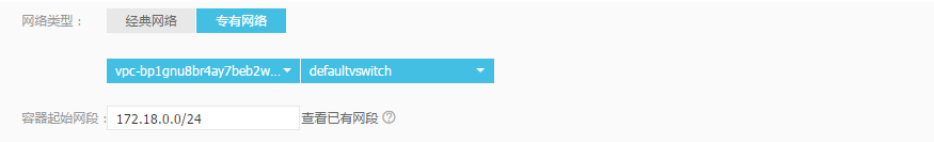
可以选择集群归属的不同网络类型：经典网络或专有网络 VPC。相应的云服务器等云资源会在相应的网络环境下管理。

选择经典网络不需要额外配置。

经典网络是相对于专有网络 VPC 的说法，是由阿里云统一规划的公共基础网络。经典网络的网络地址和网络拓扑由阿里云分配，您无需进行任何特殊配置即可使用。经典网络是阿

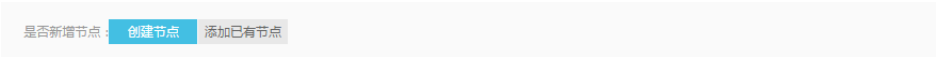
里云的默认网络环境。

选择专有网络 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。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。

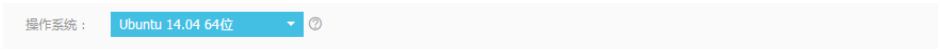
添加节点。



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

设置节点的操作系统。



目前支持的操作系统包括 Ubuntu 14.04 64 位和 CentOS 7.0 64 位。

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。

注意：

- 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。

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。

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。

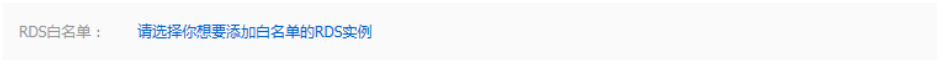
将节点 IP 添加到 RDS 实例的白名单。

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。

注意：

- 该设置仅在您选择 **创建节点** 时可用。
- 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。

单击 **请选择您想要添加白名单的RDS实例。**



在弹出的对话框中选择所需的 RDS 实例并单击 **确定。**

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。

您可以创建集群用以使用 GPU 计算型 GN4 规格族 ECS 实例。

注意：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“就绪”状态）。有关如何向集群中添加已有云服务器，参见 添加已有云服务器。

前提条件

您需要先 申请 使用 GPU 计算型 GN4 规格族 ECS 实例。

遇到没有资源的情况怎么办？

如果您已经通过了使用 GPU 计算型 GN4 规格族 ECS 实例的申请，但是在选择 **实例规格** 时，未找到 GPU 计算型 GN4 规格族 ECS 实例（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 和华东 2 地域创建 GPU 计算型 GN4 规格族 ECS 实例集群。
- 目前 GPU 计算型 GN4 规格族 ECS 实例只支持专有网络（VPC）。
- 为了保证您的 GPU 计算型 GN4 规格族 ECS 实例集群的使用效果，建议集群中统一使用 GPU 计算型 GN4 规格族 ECS 实例，不要添加其它规格族的 ECS 实例到 GPU 计算型 GN4 规格族实例集群。
- 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。

操作流程

登录 **容器服务管理控制台**。

单击左侧导航中的 **集群**，单击右上角的 **创建集群**。

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 1~64 个字符，包括数字，中文字符，英文字符和连字符（-）。

注意： 集群名称在同一个用户和同一个地域下必须唯一。

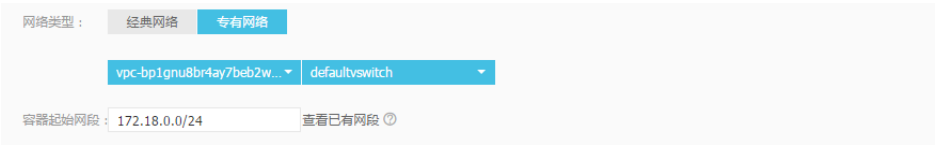
地域： 所创建集群将要部署到的地域。选择 **华南 1** 或 **华东 2**。

注意：目前，仅支持在华南 1 和华东 2 地域创建 GPU 计算型 GN4 规格族 ECS 实例集群。

可用区：集群的可用区。

注意：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。

设置集群的网络类型为 **专有网络** 并配置相关信息。

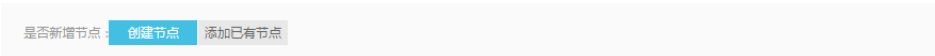


专有网络 VPC 支持您基于阿里云构建一个隔离的网络环境，您可以完全掌控自己的虚拟网络，包括自由 IP 地址范围、划分网段、配置路由表和网关等。

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。

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 添加已有云服务器。

创建节点

设置节点的操作系统。

目前支持的操作系统包括 Ubuntu 14.04 64 位和 CentOS 7.0 64 位。

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）**。

  注意：如果您已经通过了 GPU 计算型 GN4 规格族 ECS 实例的使用申请，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。

注意：

- 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。

注意：为了保证您的 GPU 计算型 GN4 规格族 ECS 实例集群的使用效果，建议集群

> 中统一使用 GPU 计算型 GN4 规格族 ECS 实例，不要添加其它规格族的 ECS 实例到 GPU 计算型 GN4 规格族 ECS 实例集群。

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。

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。

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。

您可以将已购买的云服务器添加到指定集群。如果之前没有创建过集群，您可以先创建一个零节点的集群。有关创建集群的详细信息，参见 创建集群。

注意：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。

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（经典网络或专有网络）。

添加已有云服务器时，请确保您的云服务器有公网 IP（经典网络）或 EIP （专有网络）；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 Ubunt14.04 和 Centos7 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。

容器服务的云服务器内核版本要求为 Linux 3.18+。进行手动添加时，如果您的内核版本太低，容器服务会提示您内核版本太低，添加云服务器会失败，请升级您的内核。为了保证您的数据不丢失，升级内核之前您可以创建快照，有关如何创建快照，参见 **创建快照**。

操作流程

登录 **容器服务管理控制台**。

单击左侧导航栏中的 **集群**。

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。

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

注意：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**。

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。

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**。

单击 **完成**。

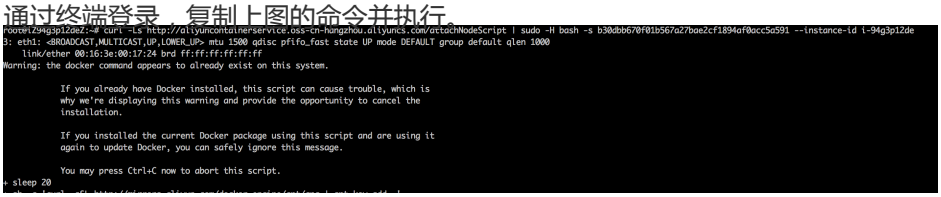
手动添加，您需要在云服务器上执行脚本。

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。

设置实例信息并单击 **下一步**。

页面显示专属这台云服务器的脚本命令。

通过终端登录，复制上图的命令并执行。



等待脚本执行成功，显示 success。该云服务器即添加成功。

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，亦或是其他公有云上的机器，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。

注意：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。

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。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

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**。如下图所示。

填写集群信息。

注意：

- **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。

单击 **创建集群**。

添加机器

操作流程

单击左侧导航栏中的 **集群**。

单击 **更多** 并在弹出菜单中单击 **添加已有实例**，如下图所示。

系统会生成一条 shell 命令，如下图所示。

注意：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。

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，如下图所示。

部署应用

本地集群可以和普通集群一样部署应用。

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。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

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。

注意：搜索不区分大小写。

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。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择要删除的集群并单击 **删除**，如下图所示。

在弹出的对话框中，单击 **确定**。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

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**。

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。

单击 **集群扩容**。

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。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择要清理的集群，单击 **管理**， 如下图所示。

在集群管理页面中，单击 **清理磁盘**，如下图所示。

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。

操作流程

操作步骤

获取访问地址。

i. 登录 容器服务管理控制台。
ii. 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**。

您可以查看集群的连接信息，如下图所示。

下载和保存证书。

要通过上面的服务地址访问 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** 开始下载证书。下载到的文件为 certFile.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/certFile.zip .
unzip certFile.zip
```

certFile.zip 文件包含 ca.pem，cert.pem 和 key.pem。

当您 通过镜像创建 Nginx 并填写 web 路由规则时，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 toolchainx.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.toolchainx.com，方便您使用自己的根域名对集群应用进行访问。

下面的示例介绍了该操作的完整流程。

注意：为了保证下面的示例能够工作，请先将 Agent 升级到最新版本。

操作流程

绑定一个负载均衡实例。

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择要配置的集群（本示例为routing-test-online），单击 **管理**。如下图所示。

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，单击 **前往 SLB 控制台** 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。
注意：支持绑定内网的负载均衡实例。

设置域名。

单击 **域名设置**，填写您自己购买的根域名，本示例中为toolchainx.com，如下图所示。

单击 **设置**。

将域名解析到绑定的负载均衡实例。

单击 **负载均衡** 并单击 **前往 SLB 控制台**。

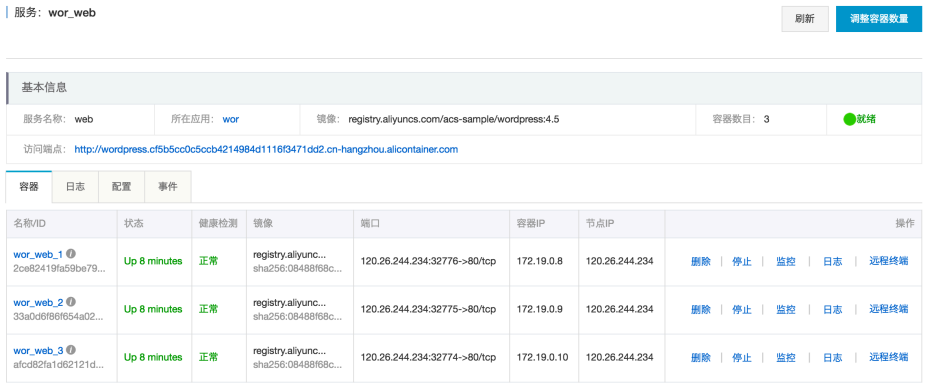
找到绑定的负载均衡实例的 IP 地址，如下图所示。

登录阿里云云解析产品页面，添加 A 记录，将*.toolchainx.com解析到负载均衡的 VIP 地址。如下图所示。

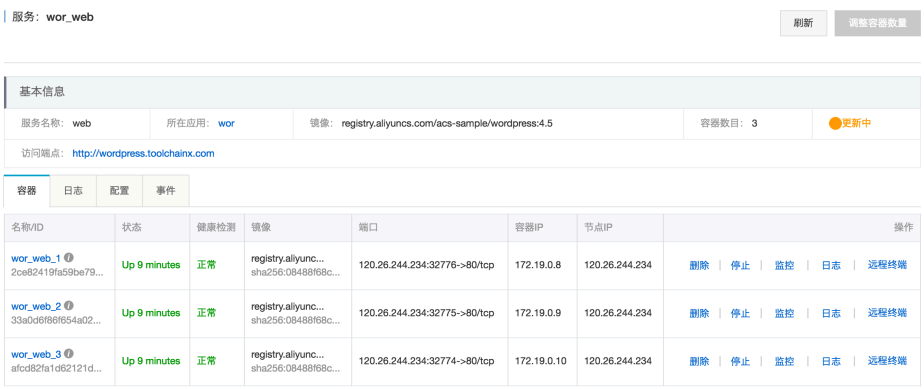
重新部署 web 服务。

重新部署应用wor，应用下的服务web访问端点发生了变化，如下图所示。

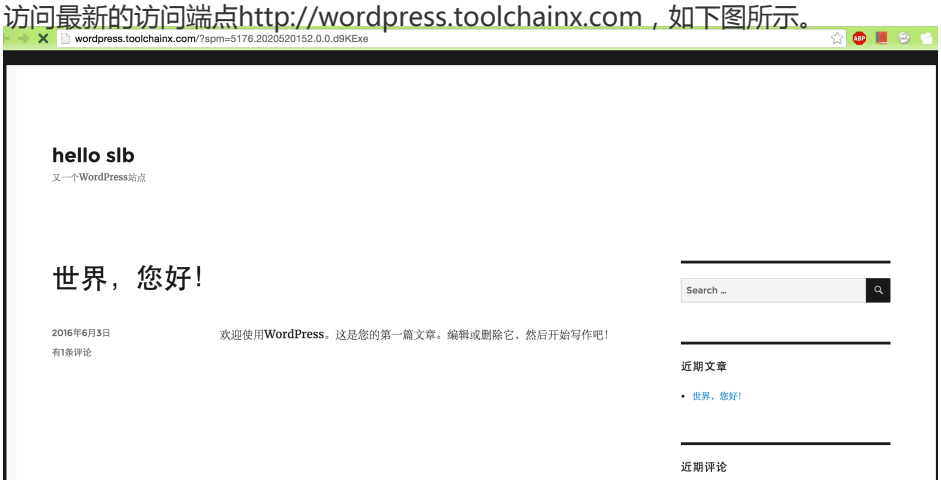
设置根域名之前的访问端点：



设置根域名之后的访问端点：



访问最新的访问端点http://wordpress.toolchainx.com，如下图所示。



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。

注意：

- 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

操作流程

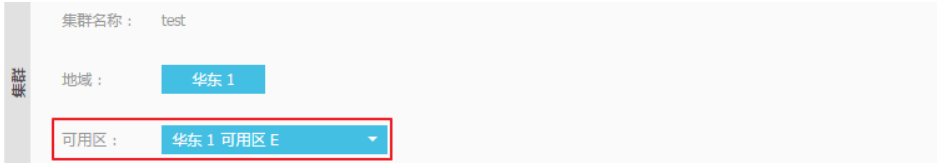
登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择要扩容的集群，单击 **更多** 并在下拉菜单中单击 **集群扩容**。如下图所示。

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。



单击 **集群扩容** 将新节点添加到集群中。

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。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择要扩容的集群，单击 **更多** > **添加已有节点**。

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。

重复以上步骤，为集群添加位于不同可用区的节点。

您可以移除或重置集群中的节点。

移除节点

移除节点可以将机器从集群中摘除。移除后将不能在节点信息内看到该机器信息。

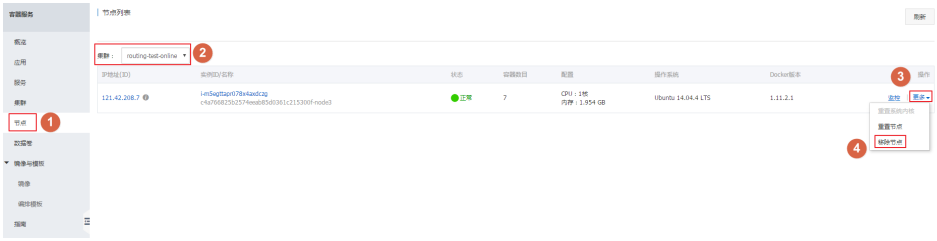
操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **节点**。

选择要移除的节点所在的集群。

选择要移除的节点，单击 **更多**，在下拉菜单中单击 **移除节点**。



在弹出的确认对话框中，选择是否 **迁移容器** 并单击 **确定**。

有关迁移容器的注意事项，参见下面的 **迁移容器**。

重置节点

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。

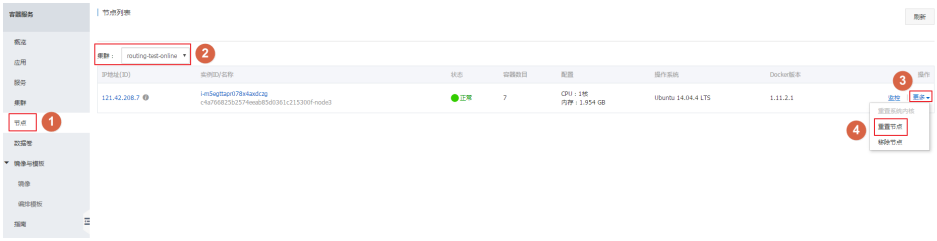
操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **节点**。

选择要重置的节点所在的集群。

选择要重置的节点，单击 **更多**，在下拉菜单中单击 **重置节点**。



在弹出的确认对话框中，选择是否 **迁移容器**，填写该实例的登录密码并单击 **确定**。

有关迁移容器的注意事项，参见下面的 **迁移容器**。

迁移容器

- 在移除节点和重置节点时，均可以选择迁移容器。
- 迁移容器会将该节点上的所有容器迁移到集群内其他机器上。
- 迁移容器会丢失本地数据卷中的数据，迁移容器前请先做好数据备份。同时，如果迁移容器失败，不会继续重置节点，需要您手动重新重置节点，再次重置时可以不勾选迁移容器，从而强行重置节点。
- 您可以单击 **集群** > **管理** > **集群事件** 查看迁移容器过程中的日志信息。

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。

操作流程

登录 **容器服务管理控制台**。

单击左侧导航栏中的 **集群**。

选择要升级 Agent 的集群，单击 **更多** > **升级 Agent**。

在弹出的对话框中，单击 **确定**。

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。

您可以在集群列表页面查看集群的 Docker 版本，如下图所示。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

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**。

如果目前系统的 Agent 版本太旧的话，需要先升级 Agent。单击 **升级Agent**，根据提示进行相应的操作即可，如下图所示。

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。

由于打快照需要一点时间，您需要耐心等待一会。完成打快照之后，系统自动进入升级 Docker Engine 的流程。

如果打快照失败，**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** 放弃升级系统服务。

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。

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

注意：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。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

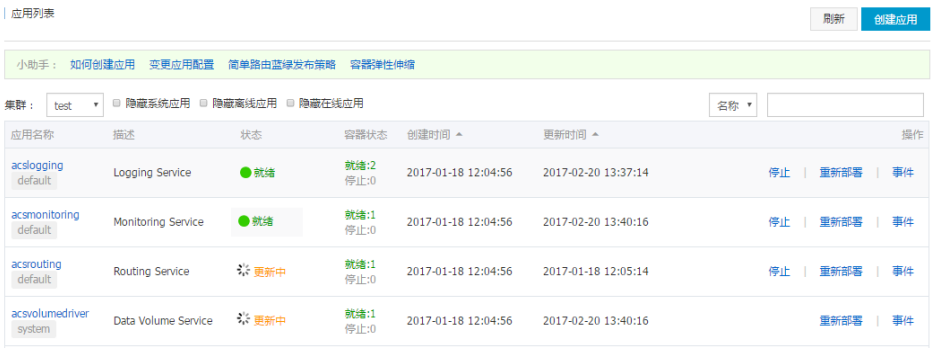
选择您要升级系统服务的集群，单击 **更多** 并在下拉菜单中单击 **升级系统服务**。如下图所示。

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。



安全组

查看安全组规则

操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择所需集群并单击右侧的 **管理**。

单击安全组的 ID，跳转到云服务器 ECS 管理控制台上该安全组的详情页面。

单击左侧导航栏中的 **安全组规则**。您可以查看安全组的规则。

安全组规则

对于 2017 年 2 月 28 日之后创建的容器服务集群，默认创建的安全组已经做了加固。开放的规则如下。

VPC 安全组：

经典网络安全组（公网入方向和内网入方向）：

注意：

- 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，方便排查问题。有些工具也依赖 ICMP。
- 容器服务依赖 22 端口和 2376 端口对机器初始化，请务必保留这两条规则。

对于 2017 年 2 月 28 日之前创建的集群，安全组规则开的比较大。以经典网络安全组规则为例。

如果希望收紧规则，可以参考安全组的配置进行如下修改（使用上图中的 **增加安全组规则** 和 **删除**）。

在内网入方向和公网入方向添加允许 ICMP 规则。

如果直接访问 VM 的 80 端口和 443 端口或者其它端口，增加内网和公网规则放开此端口。

> **注意**：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。通过负载均衡访问的端口不需要放开。

删除地址段0.0.0.0 端口 -1/-1 的公网入规则和内网入规则。

安全配置原则

每个集群一个安全组。

容器服务每个集群都管理了一个安全组。您可以在这个安全组上配置规则。

最小权限原则。

为了您集群的安全性，安全组应该对外开放最小的权限。

经典网络安全组规则区分公网和内网。

按照最小权限原则，只在需要的网卡类型上增加规则。默认情况下，安全组内的 ECS 实例都可以相互通信，所以如果要增加内网入方向的规则，您需要明确为什么要添加，是否需要给安全组外的 ECS 访问。

容器服务创建的安全组添加了一些默认规则。

为了方便用户操作 ECS 实例，容器服务创建的安全组添加了一些默认规则，开放了诸如 80/443 等端口。如果不需要，您可以删除这些规则。

> **注意**：不要屏蔽 22 和 2376 端口。容器服务需要通过这两个端口初始化机器。

尽量使用容器内部网络进行通信，不将通信暴露到宿主机上。

授权其它 ECS 实例访问安全组时，授权给安全组，而非单个 IP。

要授权其它 ECS 实例访问当前安全组，先创建一个新安全组，把要访问当前安全组的 ECS 实例加入

新安全组，再授权新安全组访问当前安全组。

优先使用 VPC 网络。如非必要，节点不要绑定 EIP， VPC 网络的隔离性更好。

VPC 内网出/入方向里要放开容器的网段。

如果不放开，会导致容器之间网络不通。

镜像与模板管理

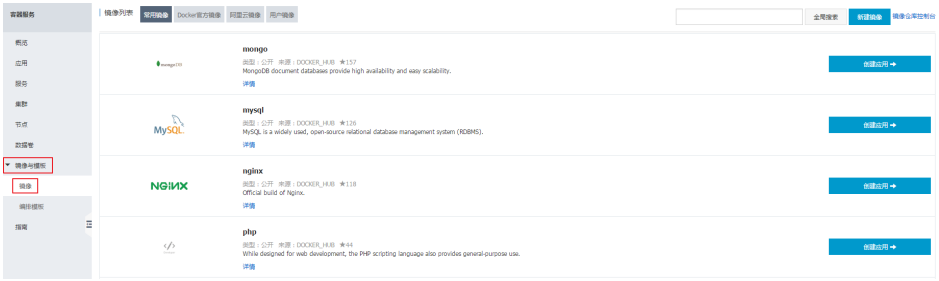
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** 并单击 **镜像**。

您可以查看镜像的种类或单击 **全局搜索** 通过镜像的名称前缀搜索镜像。

- **常用镜像**： 容器服务推荐的一些常用镜像。
- **Docker 官方镜像**： Docker Hub 提供的官方镜像。
- **阿里云镜像**： 阿里云容器 Hub 提供的镜像，包含公开镜像和私有镜像。
- **用户镜像**： 用户创建的镜像。

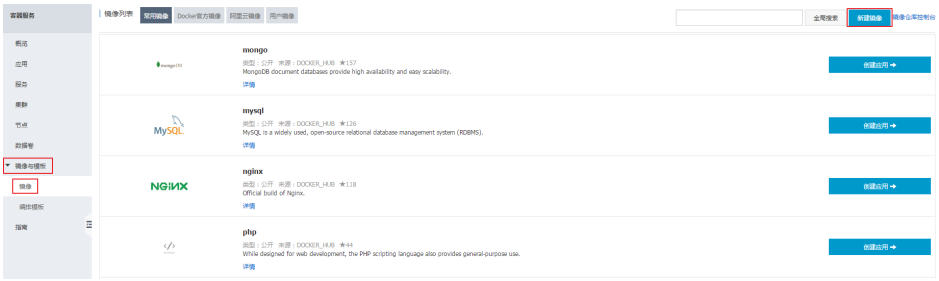


操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** 并单击 **镜像**。

单击 **新建镜像**。



页面跳转到容器 Hub 服务的 **管理中心**。

如果您是第一次访问，系统会提示您进行初始化设置。设置您的 Docker 登录密码并单击 **确定**。

然后，您可以申请一个仓库的命名空间，并单击右上角的 **创建镜像仓库** 进行镜像仓库的创建。

您可以选择通过命令行上传自己的镜像（本地仓库），也可以选择第三方代码仓库进行自动构建（GitHub\Bitbucket）。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** 并单击 **编排模板**。

您可以查看模板分类或单击 **搜索** 通过模板的名称前缀搜索模板。

- **示例编排**：显示容器服务推荐的常用编排模板。
- **我的编排**：显示您自己创建的编排模板。

操作流程

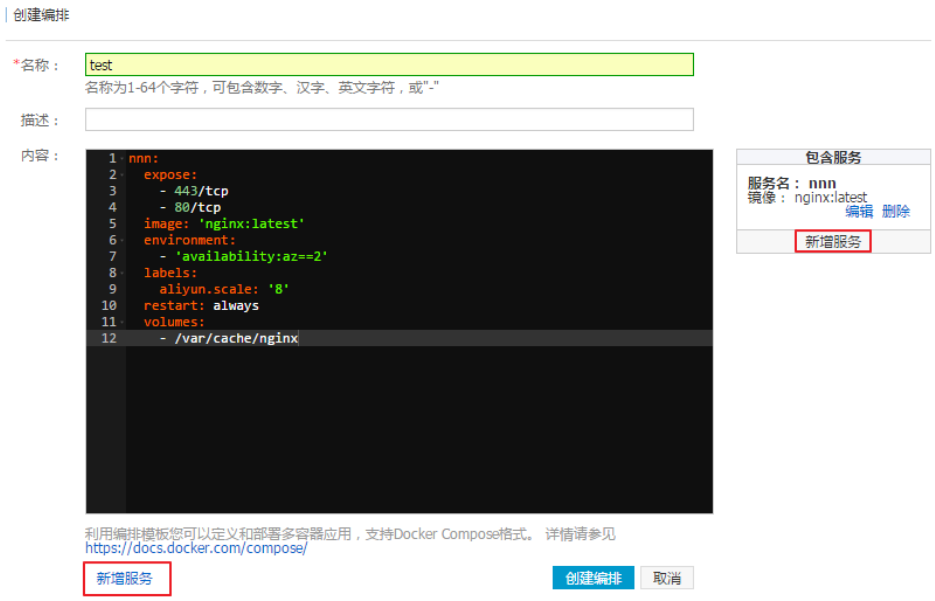
登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** > **编排模板**。

单击右上角的 **创建**。

在 **创建编排** 页面，填写模板信息。

- **名称**：模板的名称。
- **描述**：模板的相关信息。
- **内容**：Docker Compose 的 yml 文件。有关文件的详细信息，参见 Compose File 的详细说明。



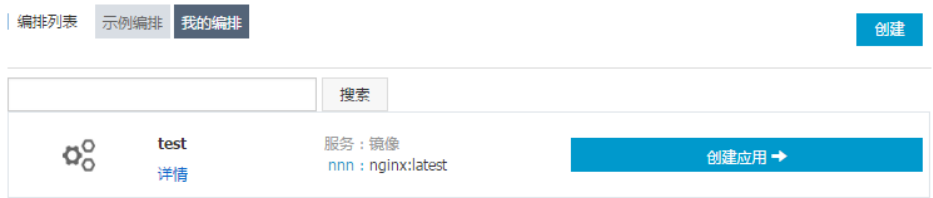
编排模板中包含的服务会显示在页面的右侧。您可以单击 **编辑** 在弹出的 **新建服务** 对话框中通过参数修改编排模板或者单击 **删除** 删除所选服务。

此外，您还可以单击 **新增服务**，选择所需的镜像，添加服务到编排模板中。

设置完成后，单击 **创建编排**。

后续操作

您可以在 **编排列表** 页面中 **我的编排** 下查看所创建的编排模板。



您可以单击 **详情** 查看编排模板的详细信息或者单击 **创建应用** 使用该编排模板创建应用。

您只能编辑 **编排列表** 页面中 **我的编排** 下的编排模板。如果您想编辑 **示例编排**，可以先将示例编排另存一下，然后再编辑。

有关如何另存编排模板，参见 另存编排模板。

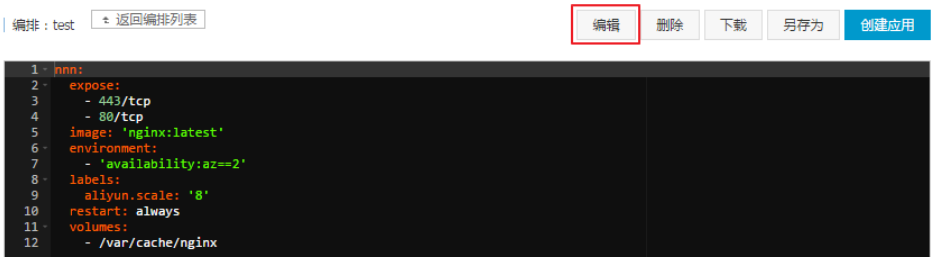
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** > **编排模板**。

单击 **我的编排**，选择一个模板并单击 **详情**。

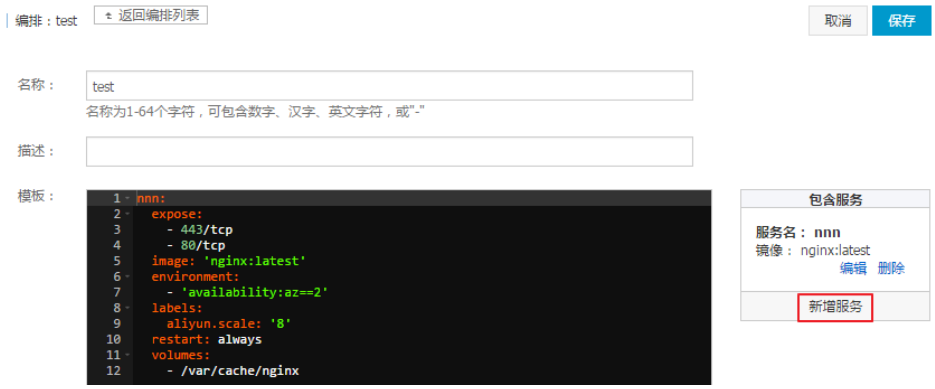
单击右上角的 **编辑**。



编辑模板内容。

您可以直接在模板中进行修改，或者在右侧选择所需的服务，单击 **编辑** 在弹出的对话框中通过参数进行修改或者单击 **删除** 删除所选的服务。

此外，您还可以单击 **新增服务**，在弹出的对话框中选择所需的镜像，将服务添加到编排模板中。



修改完成后，单击右上角的 **保存** 保存模板。

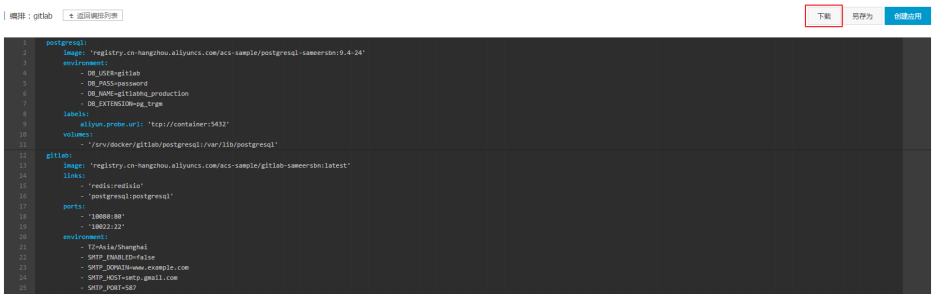
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** 并单击 **编排模板**。

选择一个模板并单击 **详情**。

单击右上角的 **下载**，会立即下载后缀为yml格式的模板文件。



注意： 您仅能删除 **编排列表** 页面中 **我的编排** 下的编排模板。

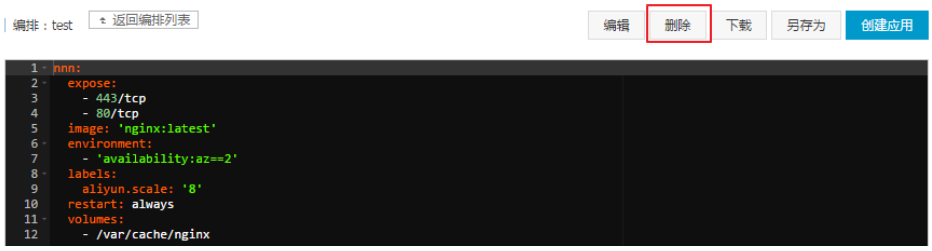
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **镜像与模板** 并单击 **编排模板**。

单击 **我的编排**，选择一个模板并单击 **详情**。

单击右上角的 **删除**。



在弹出的确认对话框中，单击 **确定**。

您可以将示例编排模板或者您自己的编排模板另存为一个新的编排模板。

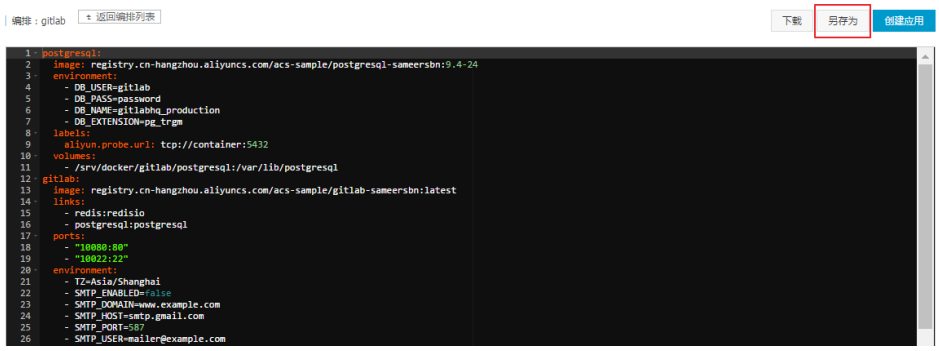
操作流程

登录 容器服务管理控制台。

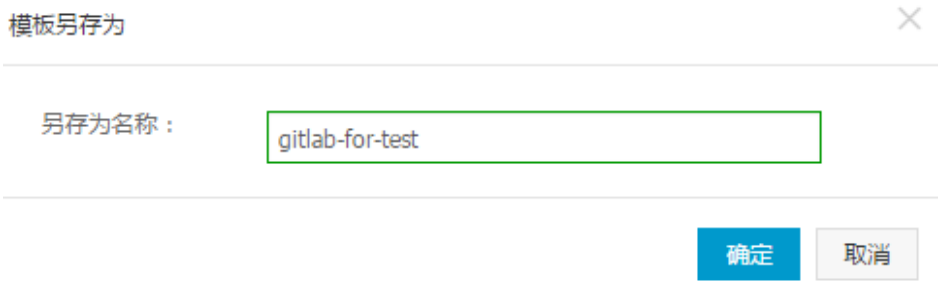
单击左侧导航栏中的 **镜像与模板** > **编排模板**。

选择一个模板并单击 **详情**。

单击右上角的 **另存为**。



填写新的编排模板名称并单击 **确定**。



另存的编排模板会显示在 **编排列表** 页面中 **我的编排** 下。

服务编排文档

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容器应用。

编排模板文件使用的标签兼容大部分 Docker Compose 1.5.x 到 1.7.x 版本实现的标签。有关具体兼容的标签，参见 标签说明。

编排模板文件也支持 Compose V1 和 V2 两种不同版本的模板格式。更多详细信息，参见 文档。

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 标签说明。

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
```

容器服务编排模板文件使用的标签兼容大部分 Docker Compose 1.5.x 到 1.7.x 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。
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。

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"
```

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。

注意：

- 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"
```

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 个容器更新成功且健康检查（配合使用 probe 标签做健康检查）显示为健康后，再去做第 n+1 个容器的更新，以此来最小化停止服务时间。

根据您所使用的 Agent 版本的不同，第 n 个容器健康检查失败包含以下两种情况：

- **使用的不是最新版本的 Agent：**如果健康检查失败且超时 30s，容器服务会停止更新容器并等待健康检查成功，然后再更新第 n+1 个容器。
- **使用最新版本的 Agent：**如果健康检查失败，服务会回滚到老版本。

示例：

部署 WordPress 服务，通过scale标签指定部署 2 个容器，通过probe标签指定检查的 URL 为 http://container/license.txt。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.probe.url: http://container/license.txt
aliyun.probe.initial_delay_seconds: '10'
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。

注意：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。
  注意：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 3 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: "3"
aliyun.probe.url: http://container:80/
aliyun.probe.timeout_seconds: "10"
aliyun.probe.initial_delay_seconds: "3"
aliyun.rolling_updates: 'true'
aliyun.rolling_updates.parallelism: "2"
```

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:

注意： 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
```

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 > 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
```

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wordpress.<cluster_id>.<region_id>.alicontainer.com
- $context_path: 请求的路径，即可以根据请求的路径来选择区分不同的后端容器

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。

注意：只有当您已经设置了 aliyun.routing.port_$container_port 时，该设置才能起作用。

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
```

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 vm 的端口，可从标签 ports 的主机:容器端口映射中获取到，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，转发到后端主机 vm 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负载均衡实例的名称 $slb_name 或者 $slb_id，以及要暴露的端口，使用的协议 $scheme（可能的值有 tcp、http、https、udp），映射的容器端口 $container_port，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 vm 机器。
- 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 vm 后端，多个服务不能共享使用同一个负载均衡实例。
- 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 vm 主机上只有一个实例。
- 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官方控制台创建监听的端口。
- 请自行登录负载均衡官方控制台对使用在容器服务中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 vm 服务器, 只需要配置好相应的标签,，就会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。

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 开启日志服务。

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
```

设置该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，slave 服务设置了 affinity.service!=master 的部署约束。使得 slave 服务一定会选择没有部署 master 服务的节点，这样当一个节点失效时，服务可用性不受影响。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。

```
master:
image: mysql:5.6
environment:
- MYSQL_USER=user
- MYSQL_PASS=test
- REPLICATION_MASTER=true
- REPLICATION_USER=repl
- REPLICATION_PASS=repl
ports:
- 3306
slave:
image: mysql:5.6
environment:
- MYSQL_USER=user
- MYSQL_PASS=test
- REPLICATION_SLAVE=true
- affinity:service!=master
ports:
- 3306
links:
- master:mysql
```

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
```

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"
```

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
```

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 文档。

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：

```
redis:
image: redis
labels:
- com.docker.swarm.reschedule-policies=["on-node-failure"]
```

注意：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。

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。

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
```

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 镜像仓库 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	说明同 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	说明同 detach 标签。
tty	说明同 detach 标签。
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。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。

应用管理

操作流程

登录 容器服务管理控制台。

单击左侧导航中的 **应用**，单击右上角的 **创建应用**。如下图所示。

设置应用的基本信息。

- **应用名称**：所创建应用的名称。名称可以包含 1~64 个字符，包括数字，中文字符，英文字母和连字符（-）。
- **应用版本**：所创建应用的版本。默认为 1.0。
- **部署集群**：所创建应用将要部署到的集群。
- **应用描述**：应用信息。可以不填写。该信息不能超过 1,024 个字符。该信息将显示在 **应用列表** 页面。
- **检查最新 Docker 镜像**：检查并使用最新版本的镜像。

单击 **使用镜像创建** 或 **使用编排模板创建**。

单击 **使用镜像创建**。根据您的需要设置以下参数。

设置 **镜像名称** 和 **镜像版本**。
您可以选择容器服务提供的镜像（单击 **选择镜像**。单击您所需的镜像并单击 **确定**），也可以直接填写自己的镜像信息。容器服务会默认使用镜像的最新版本。如果您需要使用镜像的其它版本，单击 **选择镜像版本**，单击所需版本并单击 **确定**。

设置 **容器数量**。

设置应用的 **网络模式**。
容器服务目前支持 **默认**和 **host** 网络模式。如果您不进行设置，容器服务会默认使用 **默认** 网络模式。

设置 **Restart**，即容器是否异常自动重启。

设置容器的启动命令（**Command** 和 **Entrypoint**）。
如果设置了会覆盖镜像的配置。

设置容器的资源限制（**CPU 限制** 和 **内存限制**）。
有关容器资源限制的详细信息，参见 **限制容器资源**。

设置 **端口映射**、**Web 路由规则**、和 **自定义 SLB**。

设置容器的 **数据卷**。

设置 **环境变量**。

设置容器的 **labels**。
有关容器服务支持的标签，参见 **标签说明**。

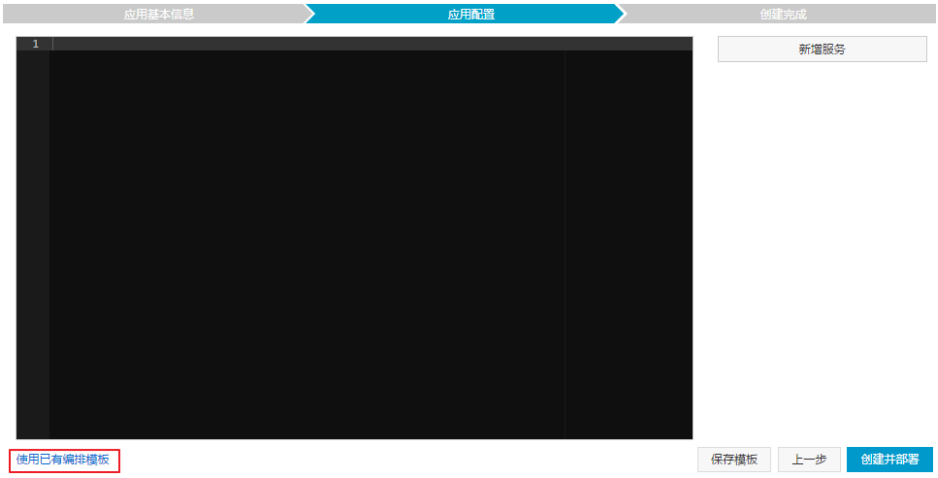
设置是否允许容器进行 **平滑升级**。有关平滑升级的相关信息，参见 **标签说明** 中的 rolling_updates。

设置容器的 **可用区调度**。
您可以选择 **跨可用区** 将容器部署在两个不同的可用区；如果您选择了此选项，但是当前集群没有两个可用区，或机器资源不够导致无法分布在两个可用区，容器创建会失败。您也可以选择 **尽量跨可用区**，容器服务会尽可能地将容器部署在两个不同的可用区中；无法满足时依然可以成功创建。
如果您不进行此项设置，容器服务会默认将容器部署在同一个可用区。
有关可用区调度的详细信息，参见 **高可用性调度**。

设置容器的 **自动伸缩** 规则。
有关容器自动伸缩的详细信息，参见 **容器自动伸缩**。

设置完毕后，单击 **创建**。

单击 **使用编排模板创建**。

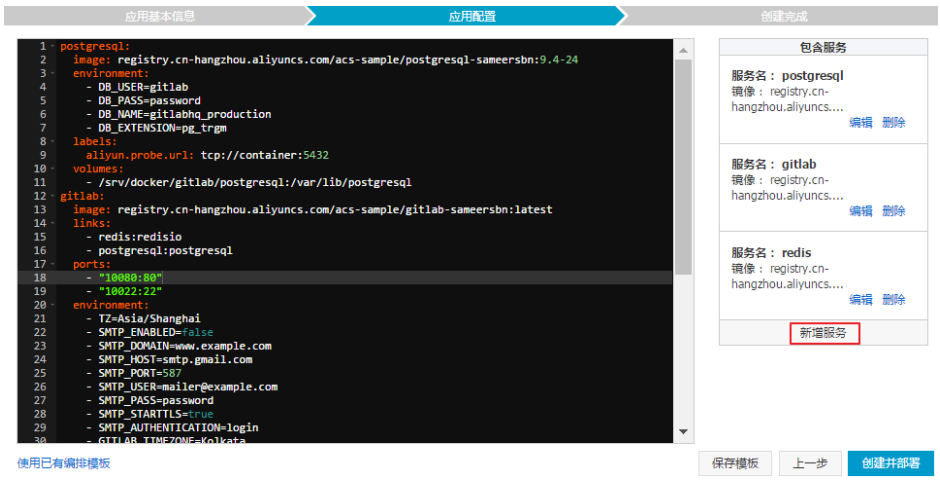


单击 **使用已有编排模板**。

选择一个模板并单击 **选择**。
编排模板的内容要求符合 Docker Compose 的格式。

编辑编排模板。

您可以根据自己的需要编辑编排模板。您可以直接在编排模板中进行修改，或者在页面右侧选择需要修改的服务，单击 **编辑** 进行修改或者单击 **删除** 删除所选的服务。



此外，您还可以单击服务列表下面的 **新增服务** 添加服务。选择你所需要的镜像，进行参数设置并单击 **确定**。

设置完毕后，单击 **创建并部署**。

Docker 容器的一大优势就是可以限制资源，包括 CPU、内存、IO 等。该设置无法通过容器服务管理控制台进行操作，您需要在编排模板中进行设置。

限制 CPU

一个 CPU 核等于 100 CPU 资源。如果机器配置是 4 核，则总共可用的 CPU 资源为 400。在编排模板中，可以通过 cpu_shares 参数指定。cpu_shares: 50表示使用 0.5 个核。

限制内存

您可以使用 mem_limit 参数限制内存，单位为 byte，最小内存为 4MB。如果设置了内存限制，当容器申请的内存超过限制时，容器会因为 OOM 而停止运行。

下面的编排模板演示了如何限制 CPU 和内存。

```
n1:
expose:
- 443/tcp
- 80/tcp
image: 'nginx:latest'
cpu_shares: 50 #0.5核
mem_limit: 500000000 #500MB
labels:
aliyun.scale: '1'
restart: always
volumes:
- /var/cache/nginx
```

资源调度

为了保证容器能获得足量的指定资源，比如上述例子里的 0.5 核 CPU 和 500MB 内存，容器服务会为容器预留资源。比如，一台 4 核的机器，最多会调度 8 个 cpu_shares=50 的容器。但是，如果创建容器时未指定 cpu_shars 和 mem_limit，则默认不占资源。

限制其他资源

其他资源限制请参考 Docker Compose 的说明。

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。当某个可用区发生故障时，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：服务至少分布在三个可用区中；如果当前集群没有三个可用区，或机器资源不够导致无法分布在三个可用区，容器创建会失败。
- availability:az==~3：服务尽可能分布在三个可用区中；无法满足时依然可以成功创建。

注意：部署约束只对新创建容器生效，对老容器变更配置时不起作用。

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
```

如果您需要将某个服务部署在指定的节点上，可以使用 constraint 关键字来实现这个功能。

注意：部署约束只对新创建容器生效，对老容器变更配置时不起作用。

在下面的示例中，服务指定部署在 node-1 上。

```
web:
image: 'nginx:latest'
restart: always
environment:
- 'constraint:aliyun.node_index==1'
ports:
- 80
labels:
aliyun.scale: 2
```

容器服务支持以下表达式：

表达式	说明
constraint:aliyun.node_index==1	指定部署到 node1。
constraint:aliyun.node_index!=1	不部署到 node1。
constraint:aliyun.node_index==(1|2|3)	指定部署到 node1 或者 node2 或者 node3。
constraint:aliyun.node_index!=(1|2|3)	部署到除 node1、node2、node3 的其他机器上。
affinity:image==~redis	尽量部署到有 Redis 镜像的机器上。
affinity:service!=~redis	尽量不部署到有 Redis 容器的机器上。

如果您希望一个应用在指定的某几个节点上部署，推荐您使用用户标签和 constraint 关键字来进行部署设置。

注意：

- 部署约束只对新创建容器生效，对老容器变更配置时不起作用。
- 使用用户标签部署应用后，如果您删除了用户标签，不会影响到已经部署的应用，但是会影响下次新的部署。请谨慎删除用户标签。

操作步骤

为节点添加用户标签。

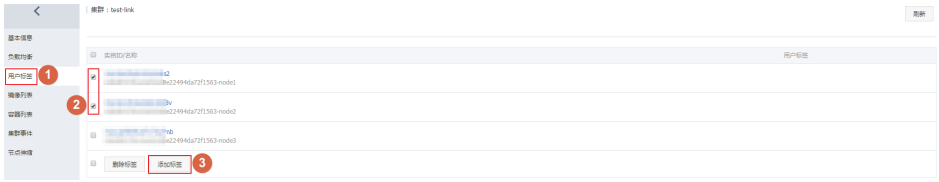
登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

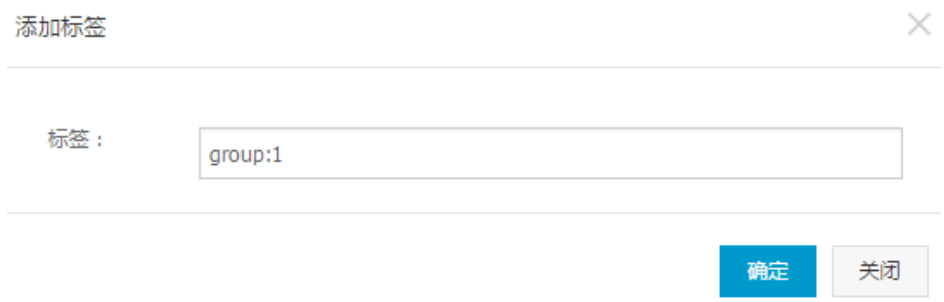
选择所需集群并单击右侧的 **管理**。

单击左侧导航栏中的 **用户标签**。

勾选您要部署应用的节点并单击 **添加标签**。



输入您自定义的标签键和标签值并单击 **确定**，为所选节点添加用户标签。



创建应用，选择 **使用编排模板创建** 并在编排模板中配置 constraint 关键字，如下所示。

有关如何创建应用，参见 **创建应用**。

```
environment:
- constraint:group==1 #表示在所有带有 “group:1” 标签的节点上部署
```

删除用户标签

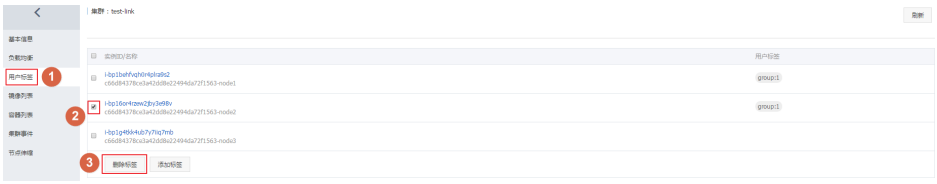
登录 容器服务管理控制台。

单击左侧导航栏中的 **集群**。

选择所需集群并单击右侧的 **管理**。

单击左侧导航栏中的 **用户标签**。

勾选要删除用户标签的节点并单击 **删除标签**。



在弹出的确认对话框中，单击 **确定**。

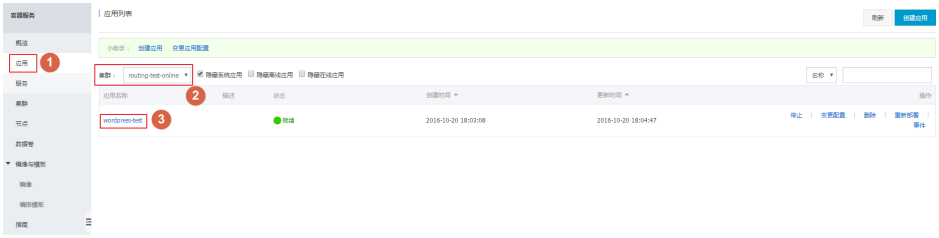
操作流程

登录 容器服务管理控制台。

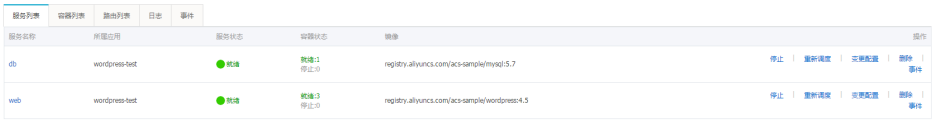
单击左侧导航中的 **应用**。

选择所要查看的应用所在的集群。

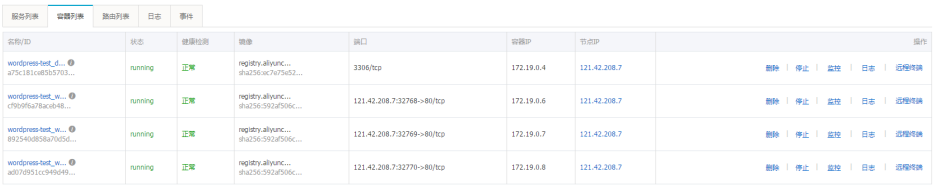
单击所要查看应用的名称。



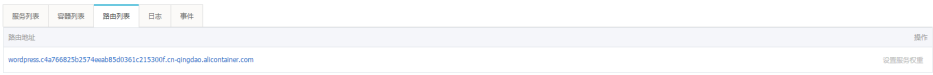
单击 **服务列表** 查看该应用的服务列表。



单击 **容器列表** 查看该应用的容器列表。

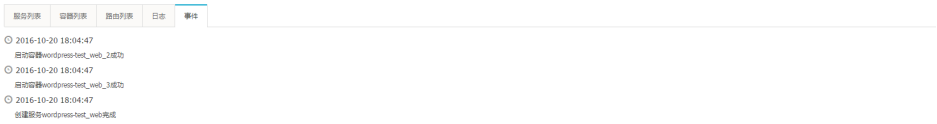


单击 **路由列表** 查看该应用的路由地址。



单击 **日志** 查看该应用的日志信息。

单击 **事件** 查看该应用的事件信息。



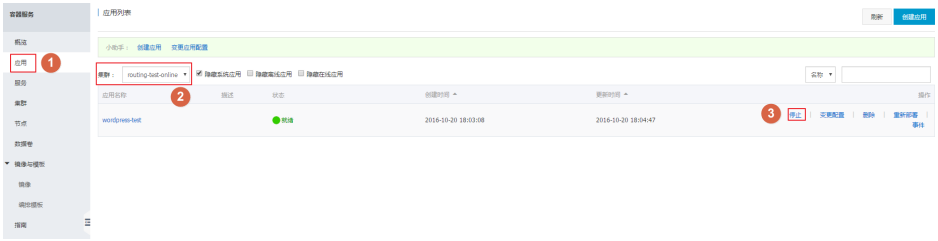
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择目标应用所在的集群。

选择目标应用并单击 **启动** 或 **停止**。



在弹出的对话框中，单击 **确定**。

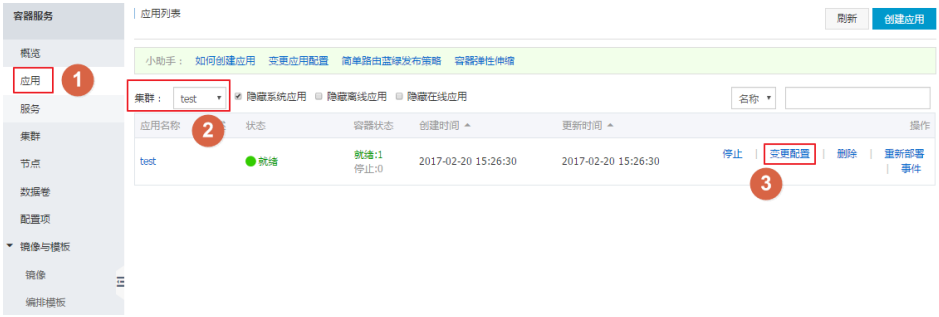
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择目标应用所在的集群。

选择目标应用并单击 **变更配置**。



在弹出的对话框中，修改配置。

> **注意**：您必须修改 **应用版本**；否则，**确定** 按钮不可用。

重新调度：默认情况下，您变更应用配置时，为了保证原服务容器本地数据卷不丢失，容器服务会在原机器上重启或重新创建容器。如果您希望将容器调度到其它机器上，您可以选择 **重新调度**，容器服务会根据您 **模板** 中的调度设置将容器调度到其它机器上。更多详情参见 变更配置常见问题。

> **注意**：选择该选项将容器调度到其它机器上，容器之前的本地数据卷数据将会丢失。请谨慎操作。

使用已有编排模板：您可以单击 **使用已有编排模板**，选择所需的编排模板进行配置变更。

> **注意**：新的模板会覆盖当前编辑模板。

后续操作

如果您变更配置后，发现应用没有更新，可以尝试重新部署应用。有关重新部署应用的详细信息，参见 **重新部署应用**。

应用部署之后您可以根据您的需求对应用进行重新部署。重新部署会重新拉取应用使用的镜像，因此如果您部署应用之后更新了镜像，重新部署会使用新的镜像进行应用部署。

注意：重新部署不会更新 volume，宿主机上的老 volume 仍会继续使用。因此，如果您挂载了 volume 并在新镜像中对 volume 设置进行了修改，重新部署后新设置不会生效。

在以下情况下，您会用到重新部署功能：

- 部署应用之后，您更新了镜像的内容，需要按照新的镜像部署应用。
- 您停止或删除了某些容器，希望可以启动或重新创建这些容器。重新部署时，容器服务会重新启动已经停止的容器并重新创建已经删除的容器。

操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择应用所在的集群。

选择要进行重新部署的应用，单击右侧的 **重新部署**。

在弹出的对话框中，单击 **确定**。

查看重新部署是否成功

您可以通过查看镜像的 sha256 确定重新部署后容器的镜像是否为最新镜像，从而确定重新部署是否成功。

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择应用所在的集群。

单击应用的名称。

单击 **容器列表** 并查看镜像的 sha256。

如果容器的镜像为新镜像，则重新部署成功。

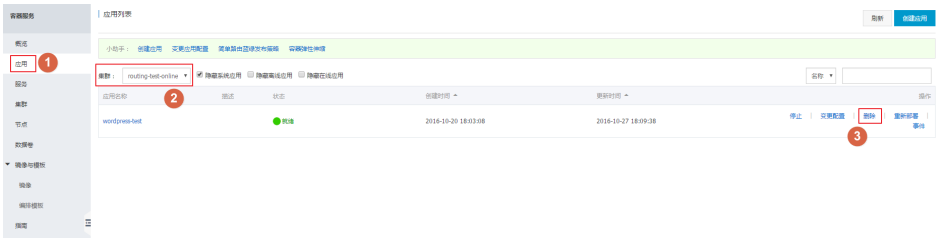
操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择所要删除的应用所在的集群。

选择所要删除的应用并单击 **删除**。



在弹出的对话框中，单击 **确定**。

容器服务抽象出离线计算的基本模型，推出了基于 Docker 容器的离线计算功能。

其核心功能包括：

- 作业编排

- 作业调度与生命周期管理
- 存储与日志等功能的集成

基本概念

下表中列出了离线应用与在线应用的概念对比。

概念	离线应用	在线应用
容器	任务执行单元	服务的执行单元
运行历史	任务出错重试的执行历史	无
服务（任务）	一个特定的功能，可以分割成若干个容器来执行	一组功能相同的容器
应用（作业）	若干个任务的组合	若干个服务的组合

简言之，一个离线作业包含若干个任务，每个任务可以由若干个容器来执行，每个容器可以有多个运行历史；而一个在线应用包含若干个服务，每个服务可以有若干个容器同时服务。

基于 Docker Compose 的作业编排

和在线应用一样，您可以使用 Docker Compose 来描述和编排作业。Docker Compose 支持 Docker 的绝大部分功能，比如：

- CPU、内存等资源限制
- 数据卷（Volume）
- 环境变量与标签
- 网络模型、端口暴露

除此之外，阿里云容器服务还扩展了以下功能：

- 容器数量：每个任务分成多少个容器
- 重试次数：每个容器重试多少次
- 移除容器：容器运行完后是否删除，可选策略包括 remove-finished（删除完成的容器）、remove-failed（删除失败的容器）、remove-all（删除全部容器）、remove-none（不删除）。
- DAG 模型的任务依赖：同一个作业的任务之间可以有依赖关系，被依赖的任务会先执行。

离线作业的 Docker Compose 示例：

```
version: "2"
labels:
aliyun.project_type: "batch"
services:
s1:
image: registry.aliyuncs.com/jimmycmh/testret:latest
restart: no
```

```
cpu_shares: 10
mem_limit: 100000000
labels:
aliyun.scale: "10"
aliyun.retry_count: "20"
aliyun.remove_containers: "remove-all"
s2:
image: registry.aliyuncs.com/jimmycmh/testret:latest
cpu_shares: 50
mem_limit: 100000000
labels:
aliyun.scale: "4"
aliyun.retry_count: "20"
aliyun.remove_containers: "remove-finished"
aliyun.depends: "s1"
```

注意：

- 该功能只支持 Docker Compose 2.0。
- 您需要在作业级别添加标签 aliyun.project_type: "batch"。如果您未添加该标签或标签值不为 batch，则认为该应用为在线应用。
- 无论您将 restart 设置为什么值，都会被修改为 no。
- 您可以用 aliyun.depends 标签指定依赖关系。可以依赖多个任务，用逗号（,）分隔。
- aliyun.retry_count 的默认值为 3。
- aliyun.remove_containers 的默认值为 remove-finished。

作业生命周期管理

容器状态由容器的运行及退出状态决定；任务状态由该任务中所有容器的状态决定；作业状态由该作业的所有任务决定。

容器状态

- 运行中（Running）：容器在运行。
- 完成（Finished）：容器退出且 ExitCode==0。
- 失败（Failed）：容器退出且 ExitCode!=0。

任务状态

- 运行中（Running）：有容器在运行。
- 完成（Finished）：所有容器都完成了。
- 失败（Failed）：有容器失败次数超过给定值。

作业状态

- 运行中（Running）：有任务在运行。

- 完成（Finished）：所有任务都完成了。
- 失败（Failed）：有任务失败了。

上述状态都可以通过 API 获取，方便您自动化运维。

共享存储

容器之间、任务之间会有数据共享和交换，共享存储可以解决这一问题。比如在 Hadoop上跑 MR 作业，是通过 HDFS 来交换数据的。在容器服务中，可以使用三类共享存储，其特性及应用场景对比如下所示。

存储	优点	缺点	适用范围
OSSFS 数据卷	跨主机共享	读写、ls 性能低；修改文件会导致文件重写	共享配置文件；附件上传
阿里云 NAS 数据卷	跨主机共享；按需扩容；高性能、高可靠性；挂载速度高	成本略高	需要共享数据的重 IO 应用，如文件服务器等；需要快速迁移的重 IO 应用，如数据库等
您自己集成成三方存储，如 Portworx	将集群内的云盘虚拟成共享的大磁盘；性能高；snapshot、多拷贝	需要一定运维能力	同 NAS

具体使用数据卷的帮助，可以参考以下文档：

- 在阿里云容器服务上使用 OSS 数据卷（volume）
- 在阿里云容器服务中使用 NAS（NFS）数据卷
- 用 OSS 数据卷实现 WordPress 附件共享

集成日志和监控服务

日志和监控是分析离线作业的重要工具。阿里云容器服务集成了阿里云日志服务与云监控功能，只要在编排模板中添加一个标签，就可以将日志收集到日志服务，将容器的 CPU、内存等数据收集到云监控。具体使用方便请参考下面的文档。

- 集成日志服务
- 容器监控服务

操作步骤

登录 **容器服务管理控制台** 并创建一个集群。

有关如何创建集群的详细信息，参见 **创建集群**。

单击左侧导航栏中的 **应用** 并单击右上角的 **创建应用**。

设置应用的基本信息，单击 **使用编排模板创建**。

填入上文中的编排模板并单击 **创建并部署**。

单击 **应用** 并单击创建的应用的名称，可以查看应用的运行状态，如下图所示。

定时任务是常见需求。普遍的做法是，选择一台或几台机器，通过 crontab 实现定时任务。但是对于大规模或大量的定时任务，这种做法的缺点非常多，比如：

- 可靠性低，一台机器宕机，该机器上的定时任务就无法执行了。
- 没有调度功能，机器之间的负载可能不均衡。
- 没有重试机制，任务可能运行失败。
- 无法运行大规模分布式任务。

容器服务在离线任务的基础上，增加了定时任务的功能，通过简单的描述，解决了上述问题。关于离线任务的详细信息，参见 运行离线任务。

注意：只有10月25号之后升级了 Agent 版本或新创建的集群才能使用该功能。

基于 Docker Compose 的定时任务描述

同离线任务一样，定时任务也是基于 Docker Compose 的，您只需要在应用模板里添加 aliyun.schedule 标签即可实现定时功能。如下面的例子所示。

```
version: "2"
labels:
aliyun.project_type: "batch"
aliyun.schedule: "0-59/30 * * * * *"
services:
s1:
```

```
image: registry.aliyuncs.com/jimmycmh/busybox:latest
labels:
aliyun.scale: "5"
aliyun.retry_count: "3"
aliyun.remove_containers: "remove-all"
command: date
```

说明：

aliyun.schedule: "0-59/30 * * * * *"表示每 30 秒执行一次该任务；schedule 的格式和 crontab 完全相同（但要注意 schedule 的格式为 秒 分 时 天 月 星期，比 Linux 上的 crontab 多了秒这一项），使用的时间为北京时间。

因为定时功能只适用于离线任务，所以只要您添加了 aliyun.schedule 标签，系统会自动添加 aliyun.project_type: "batch" 标签，因此上述例子中的 aliyun.project_type: "batch" 可以省略。

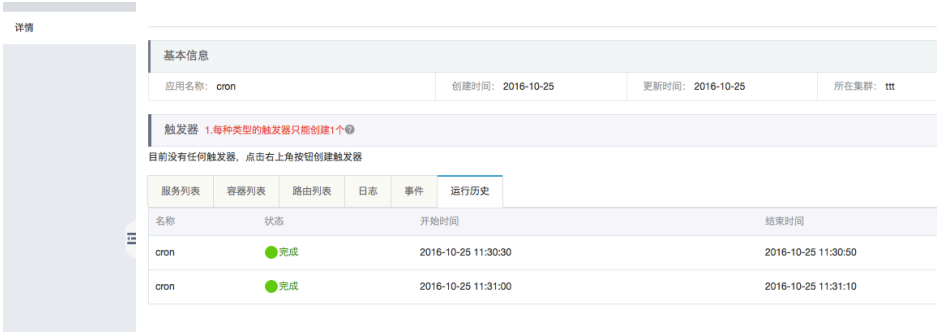
另外，离线任务中所有的功能，在定时任务中依然可用，比如 scale、retry_count、remove_containers 等。有关标签的具体含义，参见 运行离线任务。

执行过程

定时任务被创建后，应用处于“等待”状态。当任务指定的时间到达时，任务会被启动运行，其后的状态变化和离线应用相同；下一个执行时间到达时，应用状态会重复这一过程。

同一个定时任务同一时刻只会有一个实例在执行，如果任务的执行时间大于其执行周期（比如上述任务的执行时间大于 30 秒），则下一次执行会进入执行队列；如果执行队列长度大于 3，则会丢弃该次执行。

您可以在应用详情页面单击 **运行历史** 查看运行历史及结果，如下图所示。运行历史列表只保留最后 10 次的运行历史。



高可用性

定时任务控制器采用主-从备模式。主控制器故障时，控制功能将切换至备用控制器。

如果任务的执行时刻正好在主从切换期间，则会延迟至切换完成后执行。如果主从切换期间同一个任务有多次

执行，切换完成后只会执行一次；因此，为了保证任务不丢失，请不要设计重复周期小于一分钟的任务。

应用名称	中文名称	包含的服务	简介
acsrouting	路由服务	routing	提供 7 层协议的请求路由服务。组件包括负载均衡和一个 HAProxy 容器。域名配置正确后，即可将请求发送到指定的容器内。使用方法参见 路由服务。
acslogging	日志服务	logtail，logspout	与阿里云日志服务相结合，将容器内应用程序打印的日志上传到阿里云日志服务中进行保存，方便您查找和分析。使用方法参见 日志服务。
acsmonitoring	监控服务	acs-monitoring-agent	与阿里云云监控进行集成，并与目前比较流行的第三方开源监控框架做集成，方便您查询监控信息并配置监控报警。使用方法参见 监控服务。
acsvolumedriver	数据卷服务	volumedriver	与阿里云的存储服务 OSS 以及 NAS 做集成，方便您以 volume 卷（数据卷）的方式来使用共享式存储，告别有状态的容器运维。使用方法参见 数据卷服务。

多套环境

应用分为代码和配置两个部分，当应用容器化之后，通常通过容器环境变量的方式传递配置，从而实现同一个镜像使用不同的配置部署多套应用。

使用限制

- 关联配置文件时，仅能关联与应用处于同一地域中的配置文件。

- 目前，在创建应用时关联配置文件的场景仅适用于通过编排模板创建应用。

应用场景

创建应用

登录 容器服务管理控制台。

单击左侧导航栏中的 **配置项**，选择需要创建配置项的集群并单击 **创建配置项**。

填写配置文件的信息并单击 **确定**。

- **配置文件名**：可包含 1~128 个字符。
- **描述**：最多可包含 128 个字符。
- **配置项**：您最多可以设置 50 个配置项。

本示例中设置了 size 变量。

单击左侧导航栏中的 **应用**，选择所创建配置项所在的集群并单击 **创建应用**。

填写应用的基本信息并单击 **使用编排模板创建**。

填写如下所示的编排模板并单击 **创建并部署**。

其中，size 即为动态的变量，这个变量会被配置项中的数值覆盖。

```
busybox:
image: 'busybox'
command: 'top -b'
labels:
aliyun.scale: $size
```

在弹出的对话框中，选择要关联的配置文件，单击 **替换变量** 并单击 **确定**。

更新应用

如果创建应用时关联了配置文件，您可以通过修改配置项并重新部署的方式更新应用。

登录 **容器服务管理控制台**。

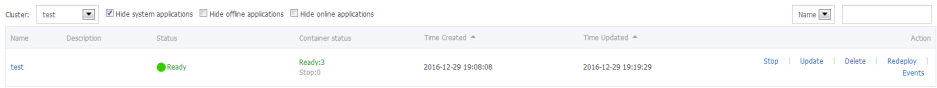
单击左侧导航栏中的 **配置项**，选择需要修改的配置项所在的集群，选择需要修改的配置项并单击 **修改**。

在弹出的确认对话框中，单击 **确定**。

选择要修改的变量并单击 **编辑**（单击后变为 **保存**）。修改变量值，单击 **保存**并单击 **确定**。

单击左侧导航栏中的 **应用**，选择应用所在的集群，选择应用并单击 **重新部署**。

更新完成后，容器数变为 3 个。



触发更新

如果创建应用时关联了配置文件，您可以通过触发器触发的方式进行重新部署。

登录 **容器服务管理控制台**。

单击左侧导航栏中的 **配置项**，选择需要修改的配置项所在的集群，选择需要修改的配置项并单击 **修改**。

在弹出的确认对话框中，单击 **确定**。

选择要修改的变量并单击 **编辑**（单击后变为 **保存**）。修改变量值，单击 **保存** 并单击 **确定**。

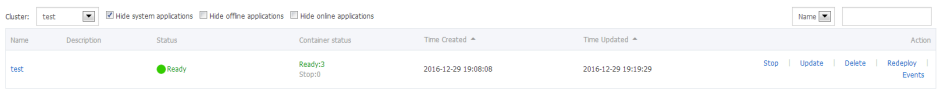
创建重新部署触发器。

有关如何创建触发器的详细信息，参见 **触发器**。

触发重新部署触发器。

```
curl "https://cs.console.aliyun.com/hook/trigger?triggerUrl=Y2FkZDgyMTUwNzdjYjQ3YThiZWEwMjAyODU1MjBlNDA5fGJ1c3lib3h8cmVkZXBsb3l8MTk0dHVzc29kNGJvcHw=&secret=517a45506f51317652744d774f58416a074b1b75299b83dd91b66150bac7ac15"
```

更新完成后，容器数变为 3 个。



服务管理

一个应用由一个或多个服务组成，您可以选择变更应用的配置来升级应用，也可以选择单独变更某个服务的配

置来独立升级。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

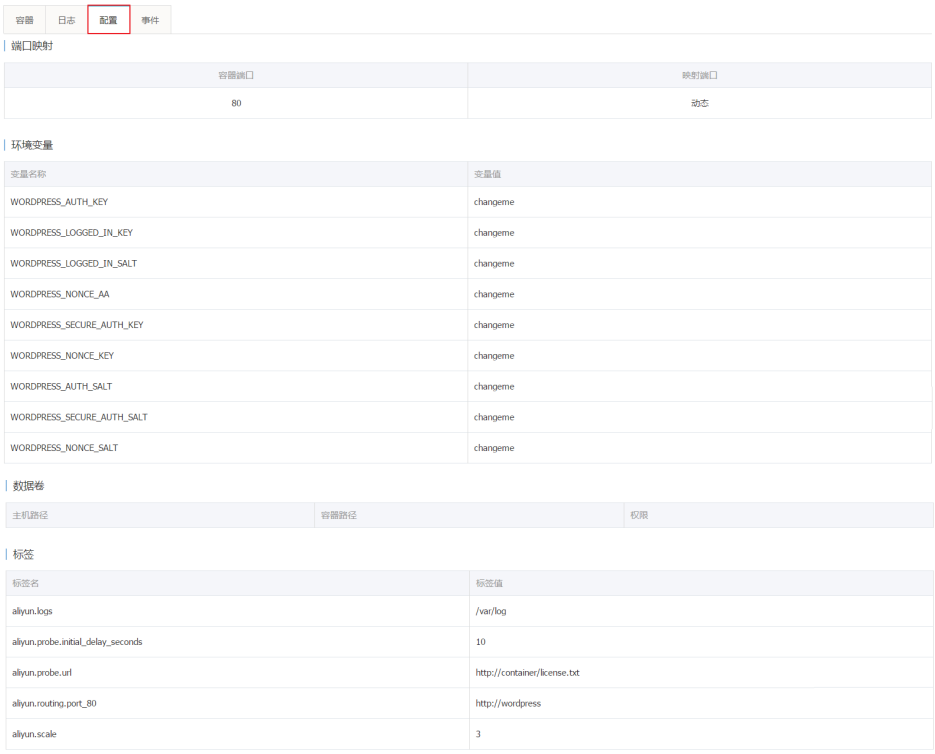
选择所要查看的服务所在的集群。

单击所要查看的服务的名称。如下图所示。

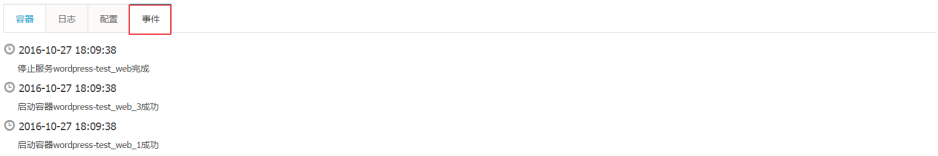
您可以在服务详情页面查看该服务的所有容器。

单击 **日志** 查看服务级别的日志信息。

单击 **配置** 查看服务的配置信息。



单击 **事件** 查看服务的事件信息。



操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

选择目标服务所在的集群。

选择目标服务并单击 **启动** 或 **停止**。

在弹出的对话框中，单击 **确定**。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

选择目标服务所在的集群。

选择目标服务并单击 **变更配置**。如下图所示。

在弹出的对话框中，修改服务的配置。

单击 **确定**。

您可以重新平衡各个节点运行容器数量，将负载较重节点的容器迁移到新加入的节点和负载较轻的节点上，实现集群负载重新的平衡。

操作流程

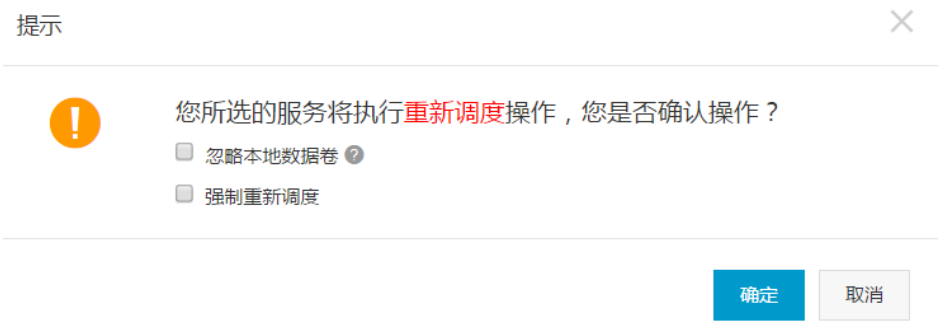
登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

选择目标服务所在的集群。

选择目标服务并单击 **重新调度**。如下图所示。

在弹出的对话框中，选择 **忽略本地数据卷** 和 **强制重新调度** 并单击 **确定**。



- **忽略本地数据卷**：对于有本地数据卷的容器，重新调度可能会将容器迁移到其他机器造成数据丢失。如果您需要忽略本地数据卷，请选择此参数；否则，对于有本地数据卷的容器将不进行重新调度。
- **强制重新调度**：目前为了保证线上服务的稳定性，默认只有在机器内存使用率超过 60%，CPU 使用率超过 40% 时才会进行重新调度。如果不想受限于这个限制，请选择此参数，容器服务将忽略使用率限制强制重新调度。
  注意：已使用内存和 CPU 数值以容器配置为准，因此不一定是机器的实际使用情况。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

选择所要删除的服务所在的集群。

选择所要删除的服务并单击 **删除**。

在弹出的对话框中，单击 **确定**。

网络管理

容器服务为容器创建全局的网络，集群中的容器都可以通过容器的 eth0 的网络接口访问其它容器。

例如：分别在两台机器上创建容器，并打印出它们的 IP 地址。

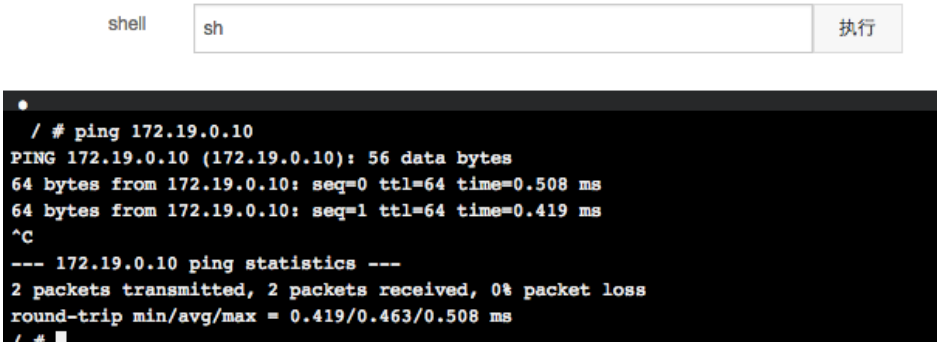
```
    cross-host-network-test1:
image: busybox
command: sh -c 'ifconfig eth0; sleep 100000'
tty: true
```

```
environment:
- 'constraint:aliyun.node_index==(1)'

cross-host-network-test2:
image: busybox
command: sh -c 'ifconfig eth0; sleep 100000'
tty: true
environment:
- 'constraint:aliyun.node_index==(2)'
```

可以看到这两个服务的容器分布在不同的节点上，如下图所示。

通过容器服务管理控制台或者通过容器 2 输出的ifconfig eth0日志，您可以看到容器 2 的 IP 地址为 172.19.0.10。您可以通过连接远程终端在容器 1 内访问容器 2 的 IP 地址。如下图所示。



数据卷指南

Docker 的特性决定了容器本身是非持久化的，容器被删除后其中的数据也会被一并删除。虽然 Docker 提供的数据卷（Volume）可以通过挂载宿主机上的目录来实现持久存储，但在集群环境中，宿主机上的数据卷有很大的局限性。

- 容器在机器间迁移时，数据无法迁移
- 不同机器之间不能共享数据卷

为了解决这些问题，阿里云容器服务提供第三方数据卷，将各种云存储包装成数据卷，可以直接挂载在容器上

，并在容器重启、迁移时自动重新挂载。目前支持 OSSFS 和 NAS 两种存储。

注意： 目前阿里云 NAS 开放了杭州、上海、北京和深圳地域，只有位于这些地域的集群才可以创建 NAS 数据卷。

前提条件

您的集群必须满足以下两个条件，才可以开通数据卷功能：

- 集群 Agent 的版本为 0.6 或更高。
- 集群里部署了 acsvolumedriver 应用。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **数据卷**，开通数据卷功能。

OSSFS 是阿里云官方提供的基于 FUSE 的文件系统（项目主页见 https://github.com/aliyun/ossfs）。OSSFS 数据卷可以将 OSS 的 Bucket 包装成数据卷。

由于数据需要经过网络同步到云端，OSSFS 在性能和功能上与本地文件系统有差距。请不要把数据库等重 IO 应用、日志等需要不断改写文件的应用运行在 OSSFS 上。

OSSFS 和本地文件系统具体差异如下所示：

- 随机或者追加写文件会导致整个文件的重写。
- 因为需要远程访问 OSS 服务器，元数据操作（例如 list directory）性能较差。
- 文件/文件夹的重命名操作不是原子的。
- 多个客户端挂载同一个 OSS Bucket 时，需要您自行协调各个客户端的行为。例如，避免多个客户端写同一个文件等等。
- 不支持硬链接（hard link）。

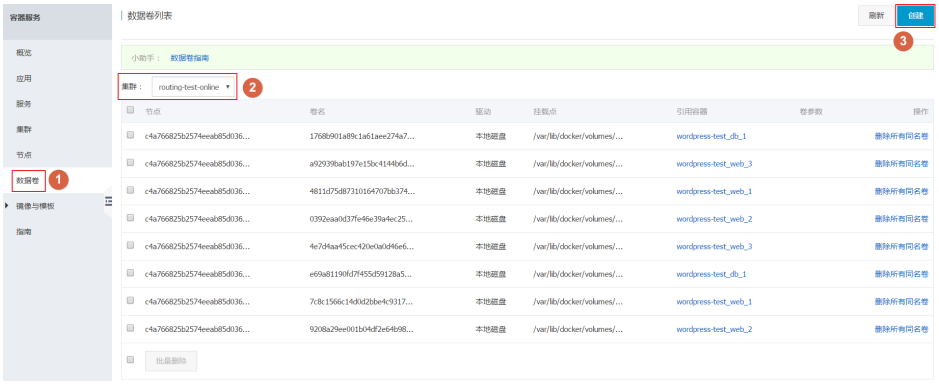
OSSFS 比较适合多容器之间共享配置文件，或者附件上传等没有改写操作的场景。

操作流程

登录 容器服务管理控制台。

单击左侧导航栏中的 **数据卷**。

选择需要创建数据卷的集群并单击页面右上角的 **创建**。如下图所示。



在弹出的对话框中，设置数据卷参数并单击 **创建**。容器服务会在集群的所有节点上创建名称相同的数据卷。

- **数据卷名**：数据卷的 ID。数据卷名在集群内必须唯一。
- **AccessKey ID**、**AccessKey Secret**：访问 OSS 所需的 AccessKey。您可以从 **阿里云账号 AccessKey 控制台** 获取。
- **访问域名**：如果 Bucket 和 ECS 实例位于同一个地域（Region），选择 **内网域名**；否则，选择 **外网**

域名。

- **文件缓存**：如果需要在不同机器间同步同一个文件的修改（比如在机器 A 中修改文件，在机器 B 中读取修改后的内容），请关闭文件缓存。
  注意：关闭文件缓存将导致 ls 文件夹变得很缓慢，尤其是同一个文件夹下文件比较多时。因此，没有上述需求时，请打开文件缓存，提高 ls 的速度。

阿里云 NAS 是面向阿里云 ECS 实例的文件存储服务，提供标准的文件访问协议，您无需对现有应用做任何修改，即可使用具备无限容量及性能扩展、单一命名空间、多共享、高可靠和高可用等特性的分布式文件系统。

使用限制

目前阿里云 NAS 开放了杭州、上海、北京和深圳地域，只有位于这些地域的集群才可以创建 NAS 数据卷。

操作步骤

本示例以位于华东 1 地域的 VPC 容器服务集群为例进行讲解。

步骤 1 创建 NAS 文件系统

登录 **文件存储管理控制台**，创建一个 NAS 文件系统，参见 **创建文件系统**。

> **注意**：创建的 NAS 文件系统需要和您的集群位于同一地域。

本示例创建了一个位于华东 1 地域的 NAS 文件系统。

步骤 2 添加容器服务集群的挂载点

登录 **文件存储管理控制台**，参见 **添加挂载点** 添加一个挂载点。

如果您的容器集群为经典网络类型，添加一个经典网络挂载点。

如果您的容器集群为 VPC 类型，添加一个 VPC 挂载点。

本示例要添加一个 VPC 挂载点。

> **注意**：**VPC网络** 处选择您的容器集群所在的 VPC。否则，创建数据卷的时候会出错。

步骤 3 添加集群 ECS 实例内网 IP 到 NAS 文件系统白名单

为了使集群内的 ECS 实例可以访问 NAS 文件系统，ECS 实例的内网 IP 需要添加进 NAS 文件系统的白名单。

对于 2017 年 2 月份之后创建的集群，创建 NAS 数据卷时，会自动将集群中 ECS 实例的内网 IP 添加到 NAS 文件系统的白名单中，您不需要进行任何操作。

创建 NAS 数据卷之后，您再进行集群扩容（通过 **添加已有云服务器** 或 **扩容集群**）时，容器服务会自动为新添加的或者扩容进来的 ECS 实例创建 NAS 数据卷并自动将新添加的 ECS 实例的内网 IP 添加进 NAS 文件系统的白名单。

对于 2017 年 2 月份之前创建的集群，您可以通过以下两种方法将集群中 ECS 实例的内网 IP 添加到 NAS 文件系统。

手动添加白名单。

登录 **文件存储管理控制台**，创建权限组并添加权限组规则将集群 ECS 实例的内网 IP 添加到白名单。具体操作参见 NAS 使用文档 **使用权限组进行访问控制** 。

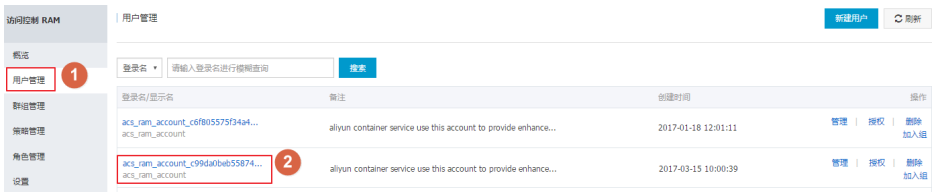
通过此中方法添加白名单并创建 NAS 数据卷后，再进行集群扩容（通过 **添加已有云服务器** 或 **扩容集群**）时，容器服务会自动为新添加的或者扩容进来的 ECS 实例创建 NAS 数据卷，但是使用这些数据卷前，您必须且只能通过手动方法将新添加的 ECS 实例的内网 IP 添加到 NAS 文件系统的白名单。

在 RAM 中进行授权。授权后会实现自动添加白名单，且以后再进行集群扩容（通过 **添加已有云服务器** 或 **扩容集群**）时，会自动将新添加的 ECS 实例的内网 IP 到 NAS 文件系统的白名单。

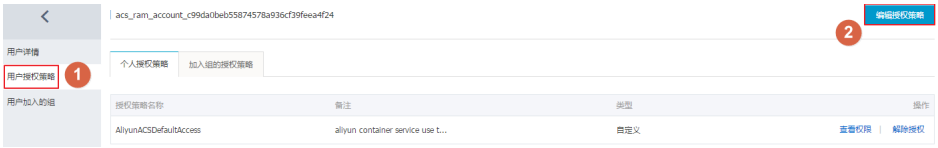
登录 RAM 管理控制台。

单击左侧导航栏中的 **用户管理**。

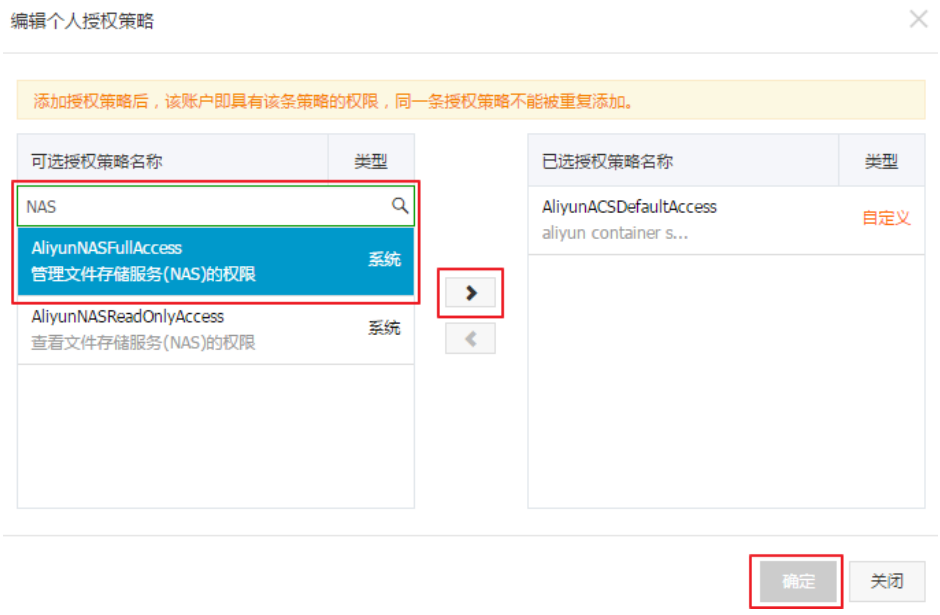
找到名为 acs_ram_account_[cluster_id] 的用户组并单击。



单击左侧导航栏的 **用户授权策略** 并单击右上角的 **编辑授权策略**。



在搜索框中输入 **NAS** 进行搜索，选择 **AliyunNASFullAccess**，添加到 **已选授权策略名称** 列表并单击 **确定**。



输入手机验证码并单击 **确定** 完成授权。

步骤 4 创建 NAS 数据卷

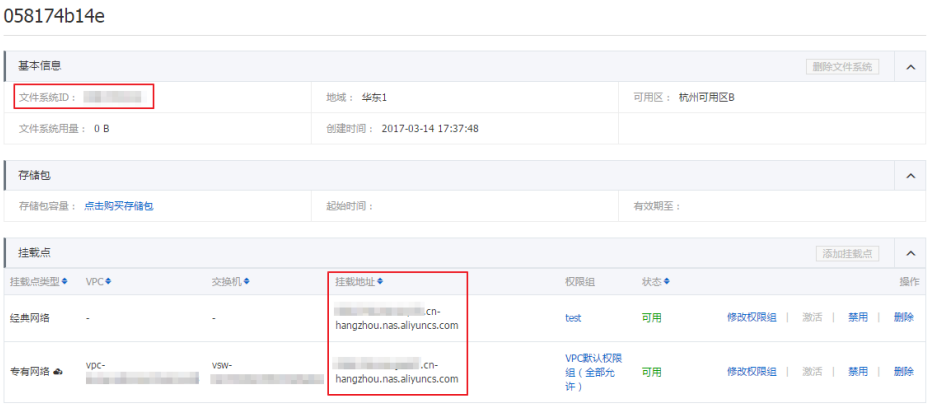
登录 容器服务管理控制台。

单击左侧导航栏中的 **数据卷**。

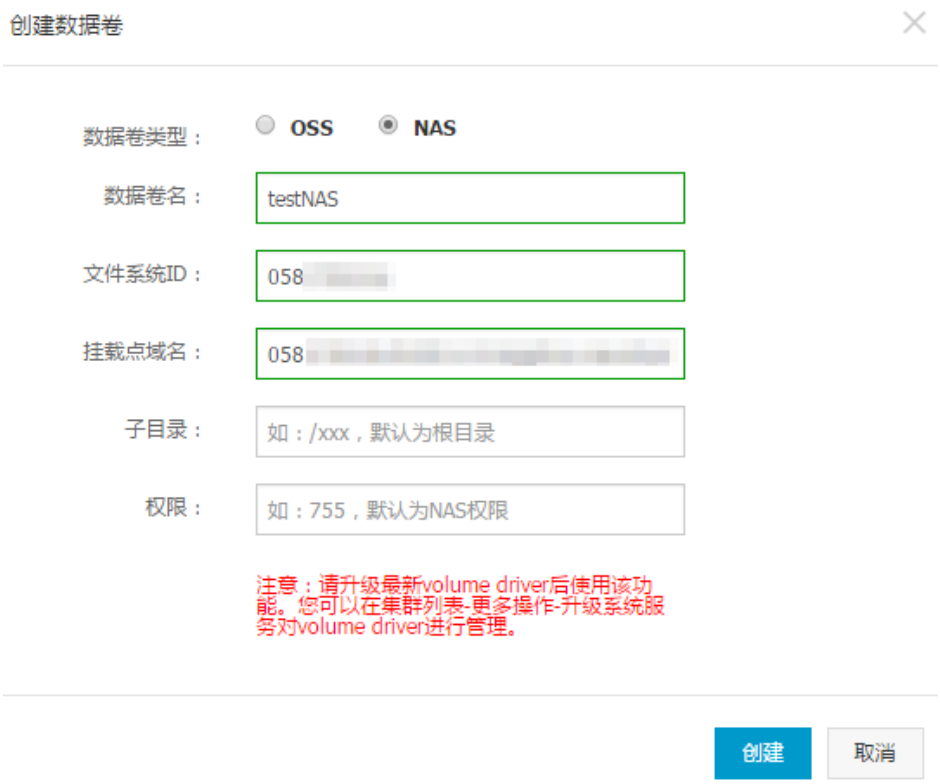
选择需要创建数据卷的集群并单击页面右上角的 **创建**。如下图所示。

在弹出的对话框中，设置数据卷参数并单击 **创建**。容器服务会在集群的所有节点上创建名称相同的 NAS 数据卷。

您可以登录 文件存储管理控制台，单击集群要挂载的 NAS 文件系统的 ID，查看文件系统的详细信息。



参见文件系统的详细信息，填写数据卷的配置。

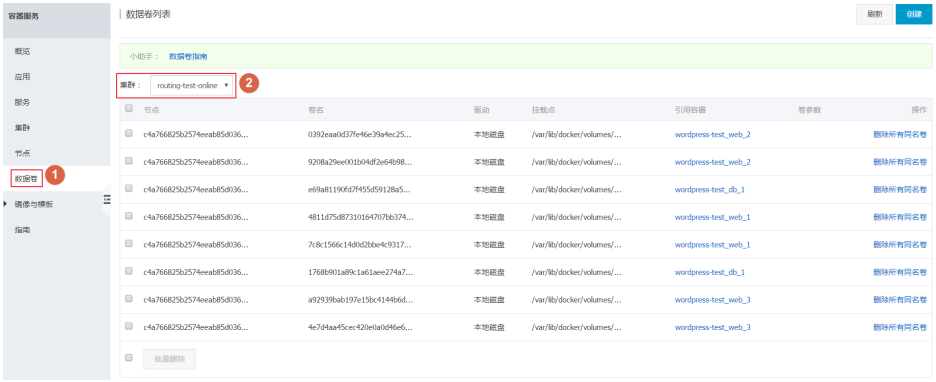


- **数据卷名**：数据卷的名称。数据卷名称在集群内必须唯一。
- **文件系统 ID**：NAS 文件系统的 ID。
- **挂载点域名**：集群在 NAS 文件系统中挂载点的挂载地址。
- **子目录**：NAS 路径下的子目录，以 / 开头，设定后数据卷将挂载到指定的子目录。
  - 如果 NAS 根目录下没有此子目录，会默认创建后再挂载。
  - 您可以不填此项，默认挂载到 NAS 根目录。
- **权限**：设置挂载目录的访问权限，例如：755、644、777 等。
  - 只有挂载到 NAS 子目录时才能设置权限，挂载到根目录时不能设置。
  - 您可以不填此项，默认权限为 NAS 文件原来的权限。

注意：使用 **子目录**、**权限** 选项时，请将 volume driver 升级到最新版本。

登录 **容器服务管理控制台** 并单击左侧导航栏中的 **数据卷**。

数据卷列表 页面列出了当前集群中所有的数据卷，包括本地数据卷和第三方数据卷。如下图所示。



对于本地数据卷，数据卷名称的格式为node_name/volume_name。

对于第三方数据卷，单击 **查看** 可以查看数据卷的参数。

创建第三方数据卷时，容器服务会在集群的所有节点上以同一数据卷名称创建该数据卷，便于容器在不同节点间迁移。您可以单击 **删除所有同名卷** 删除所有使用该名称的数据卷。

第三方数据卷使用方法跟本地数据卷相同。

您可以在创建应用时设置数据卷的相关信息，或者通过变更已有应用的配置来添加数据卷的设置。

通过镜像创建应用：

登录 **容器服务管理控制台** > 单击左侧导航栏中的 **应用** > 单击页面右上角的 **创建应用** > 填写应用的基本信息 > 单击 **使用镜像创建** > 单击 **数据卷** 中的加号图标 > 在 **主机路径或数据卷名** 处填写数据卷名称。如下图所示。

注意：有关如何使用镜像创建应用的详细信息，参见 **创建应用**。

通过编排文件创建应用：

登录 **容器服务管理控制台** > 单击左侧导航栏中的 **应用** > 单击页面右上角的 **创建应用** > 填写应用的基本信息 > 单击 **使用编排模板创建** > 单击 **使用已有编排模板** > 选择一个模板并单击 **选择** > 在 volumes 一节中，第一个冒号前填写数据卷名称。

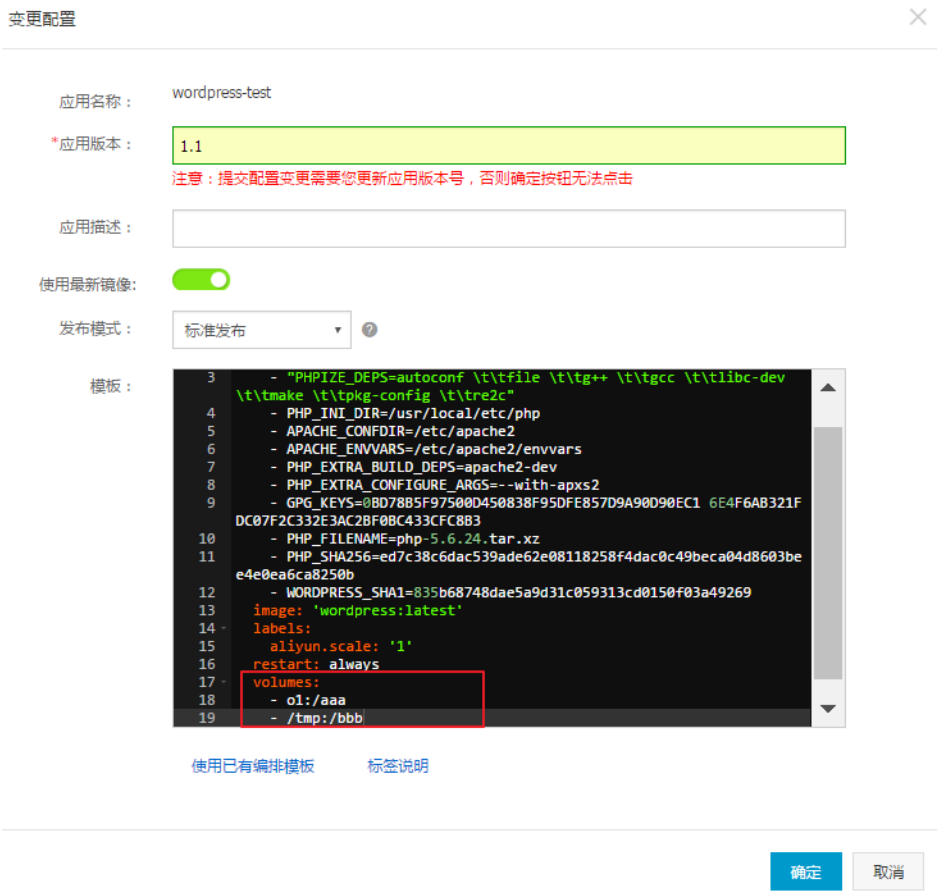
注意：有关如何使用编排模板创建应用的详细信息，参见 **创建应用**。

```
  volumes:
- o1:/aaa
- /tmp:/bbb
```

变更已有应用的配置

登录 **容器服务管理控制台** > 单击左侧导航栏中的 **应用** > 选择需要更新的应用 > 单击右侧的 **变更配置** > 在 **模板** 的 volumes 一节中，第一个冒号前填写数据卷名称。如下图所示。

注意：有关变更应用配置的详细信息，参见 **变更应用配置**。



如果用 数据卷名:镜像中已有目录 的方式使用第三方数据卷（如o1:/data，而镜像中有/data目录），启动容器会失败，系统会返回类似于 chown /mnt/acs_mnt/ossfs/XXXX: input/output error 的错误。

产生这个错误的原因是，对于命名数据卷，Docker 会把镜像中已有的文件复制到数据卷中，并用 chown 设置相应的用户权限，而 Linux 禁止对挂载点使用 chown。

您可以通过以下两种方法之一解决该问题：

升级 Docker 到 1.11 或以上版本，升级 Agent 到最新版本并在编排模板中指定 nocopy 选项。Docker 会跳过复制数据的过程，因此不会产生 chown 错误。

```
volumes:
- o1:/data:nocopy
- /tmp:/bbb
```

如果必须复制数据，您可以不使用数据卷名称，而使用挂载点路径进行设置，比如用 /mnt/acs_mnt/ossfs/XXXX:/data。但这种方式绕开了 volume driver，在机器重启时，无法保证在 OSSFS 挂载成功之后再启动容器，可能会导致容器挂载了一个本地数据卷。为了避免这种情况，您需要同时使用两个数据卷，其中一个数据卷使用数据卷名称进行设置，另外一个数据卷使用挂载点路径进行设置。使用数据卷名称进行设置的数据卷只起到和 volume driver 同步的功能，并不用于存储。

```
  volumes:
- o1:/nouse
- /mnt/acs_mnt/ossfs/XXXX:/data
- /tmp:/bbb
```

日志管理

查看应用级别的日志

登录 **容器服务管理控制台**。

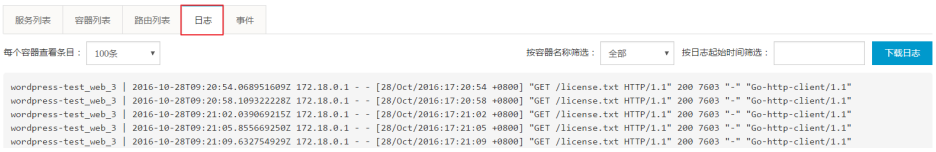
单击左侧导航栏中的 **应用**。

选择要查看的应用所在的集群。

单击要查看的应用的名称。如下图所示。

单击 **日志** 查看该应用的日志信息。

您可以选择显示的日志条目，也可以选择下载所有日志到本地。



查看服务级别的日志

登录 **容器服务管理控制台**。

单击左侧导航栏中的 **服务**。

选择要查看的服务所在的集群。

单击要查看的服务的名称。如下图所示。

单击 **日志** 查看该服务的日志信息。

您可以选择显示的日志条目，也可以选择下载所有日志到本地。

查看容器级别的日志

登录 **容器服务管理控制台**。

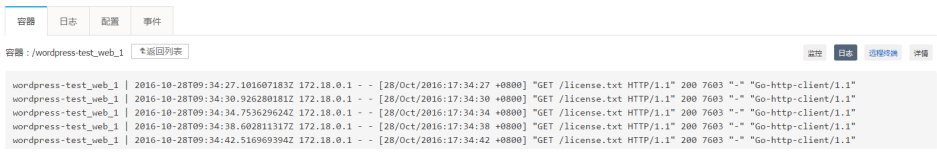
单击左侧导航栏中的 **服务**。

选择要查看的服务所在的集群。

单击要查看的服务的名称。如下图所示。

选择所要查看的容器并单击 **日志**。如下图所示。

您可以查看该容器的日志信息。



日志服务（Log Service，简称Log）是针对日志场景的平台化服务。无需开发就可以快速完成日志收集、分发、投递与查询， 适用于日志中转、监控、性能诊断、日志分析、审计等场景。容器服务提供了集成日志服务的能力，可以方便地将应用日志发送到日志服务里。

注意：阿里云日志服务已经开始收费，按照下面的方式进行配置，将会产生费用，收费标准参见 **日志服务计费规则**。请务必了解您的日志量，以免产生大量非预期的费用。

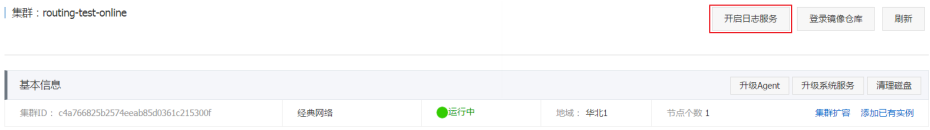
操作流程

登录 **容器服务管理控制台**。

单击左侧导航栏中的 **集群**。

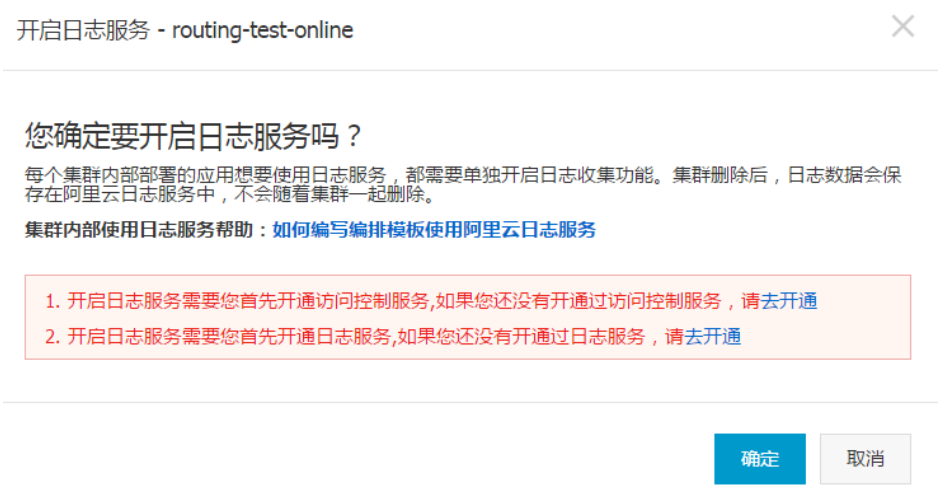
选择目标集群并单击 **管理**。

单击页面右上角的 **开启日志服务**。



在弹出的确认对话框中，单击 **确定**。

开通容器服务的日志服务之前，您需要先开通阿里云访问控制（RAM）和阿里云日志服务。如果您还未开通，请点击 **去开通** 开通访问控制（RAM）和阿里云日志服务。



查看 acslogging 服务安装结果

第一次启用日志服务时，容器服务会在您的机器安装日志服务所需的 Agent。您可以在应用列表中找到该应用。安装成功后，您就可以使用日志服务了。

同时，系统会在阿里云日志服务上创建一个对应的 project，您可以在 日志服务管理控制台 上进行查看。project 的名字里包含了容器服务集群的 ID。

在编排文件里使用日志服务

大多数的 Docker 应用会直接将日志写到 Stdout，现在您依然可以这样做（对于日志写到文件的场景，可以参考下边的 **使用日志文件**）。在开通日志管理功能后，Stdout 的日志可以自动收集并且发送到阿里云日志服务。

下面的例子创建了一个 WordPress 应用。该应用包含 WordPress 和 MySQL 两个服务，日志会收集到阿里云日志服务。

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
```

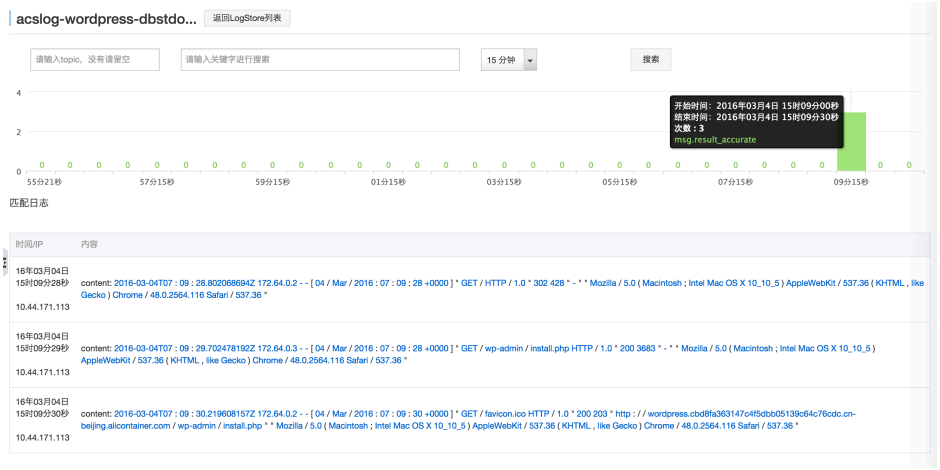
在上边的编排文件中，标签 aliyun.log_store_dbstdout: stdout表示将容器的标准写入 logstore acslog-wordpress-dbstdout里。这个标签的格式为 aliyun.log_store_{name}: {logpath}。其中 name 为阿里云日志服务 logstore 的名字，实际创建的 logstore 的名字为acslog-${app}-${name}。app 为应用名称；logpath为容器中日志的路径；stdout 是一个特殊的 logpath，表示标准输出。

用上面的编排文件，您可以在容器服务管理控制台上创建一个名为 wordpress 的应用。在应用启动完成后，可以在阿里云日志管理控制台上找到 logstore acslog-wordpress-dbstdout，其中存储了 wordpress 的日志。

在日志服务管理控制台上查看日志

使用上面的编排文件部署应用之后，您可以在阿里云日志服务控制台查看收集到的日志。登录日志服务管理控制台，找到集群对应的日志服务 project，单击进入。您可以看到编排文件里使用的logstore acs-wordpress-dbstdout。

在**日志索引** 列中单击 **查询** 查看日志。



使用文件日志

如果您不希望日志直接写到 stdout 中，而需要将日志直接写到文件中，比如/var/log/app.log，可以进行如下配置。

```
aliyun.log_store_name: /var/log/app.log
```

其中name为 logstore 的名字，/var/log/app.log为容器内日志的路径。

开启 timestamp

Docker 在收集日志的时候可以选择是否添加 timestamp。您可以在容器服务中通过aliyun.log.timestamp 标签进行配置。默认会添加 timestamp。

添加 timestamp

aliyun.log.timestamp: "true"

去除 timestamp

aliyun.log.timestamp: "false"

监控

为了满足应用在不同负载下的需求，容器服务支持服务的弹性伸缩，即根据服务的容器资源占用情况自动调整容器数量。

注意：如需使用容器自动伸缩，您需要先将集群的 Agent 升级到最新版本。有关如何升级 Agent的详细信息，参见 升级 Agent。

弹性伸缩策略：

- 当监测指标值超过所设定的上限，以用户设定的步长增加容器数量。
- 当监测指标值低于所设定的下限，以用户设定的步长减少容器数量。

服务监测指标：

- CPU 平均使用量
- 内存平均使用量

注意：容器服务在判断监测指标是否超出所设定的上下限时，使用的是采集周期（一分钟）内监测指标的平均值（即 CPU 平均使用量和内存平均使用量），而且只有当连续三个采集周期内的监测指标平均值均超出所设定的上下限时，容器服务才会触发扩容或伸缩操作，以避免因为监控数据抖动而引起频繁的扩容或缩容操作。

设置方法

使用镜像创建应用

在创建应用时，选择 **使用镜像创建**。

有关如何创建应用的详细信息，参见 创建应用。

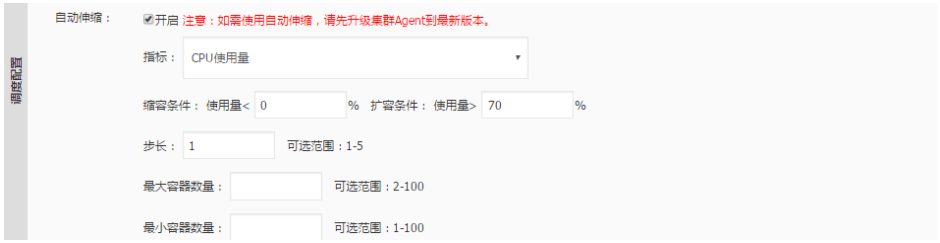
在页面最下边的 **调度配置** 中，勾选 **开启** 自动伸缩并设置自动伸缩参数。

约束规则：

- **扩容条件** 的可选范围是 50%~100%，**缩容条件** 的可选范围是 0%~50%。
- **扩容条件** 和 **缩容条件** 的差值不能小于30%。
- **步长** 的可选范围为 1~5，默认为 1。
- 设置 **最小容器数量** 和 **最大容器数量**。缩容时，如果容器数 ≤ **最小容器数量**，不会进行缩容操作；扩容时，如果容器数 ≥ **最大容器数量**，不会进行扩容操作。

注意：

- 建议不要同时设置基于 CPU 使用量和内存使用量的复合伸缩规则。
- 请谨慎设置伸缩策略。如果在您设置伸缩规则的时候，应用就满足所设置的伸缩条件而且伸缩后应用仍然满足伸缩条件，那么监控将会不断地触发伸缩。



使用编排模板创建应用

在创建应用时，选择 **使用编排模板创建**。

有关如何创建应用的详细信息，参见 创建应用。

单击 **新增服务**。

单击 **更多设置**，勾选 **开启** 自动伸缩并设置自动伸缩参数。

手动设置

在编排模板的labels配置中，添加相应的标签：

- 指定步长（默认值为 1）：aliyun.auto_scaling.step
- 最小容器数量（默认值为 1）：aliyun.auto_scaling.min_instances
- 最大容器数量（默认值为 10）：aliyun.auto_scaling.max_instances
- 以 CPU 使用量为指标
    - 指定上限： aliyun.auto_scaling.max_cpu
    - 指定下限：aliyun.auto_scaling.min_cpu
- 以内存使用量为指标
    - 指定上限：aliyun.auto_scaling.max_memory
    - 指定下限：aliyun.auto_scaling.min_memory

示例：

```
nginx:
image: 'nginx:latest'
restart: always
expose:
- 443/tcp
- 80/tcp
labels:
aliyun.scale: '1'
aliyun.auto_scaling.max_cpu: '70'
aliyun.auto_scaling.min_cpu: '40'
aliyun.auto_scaling.step: '1'
volumes:
- /var/cache/nginx
```

容器监控服务集成了阿里云云监控服务，为您提供涵盖容器、应用、集群、节点的监控与告警服务，满足了容器监控的基本需求。但是在很多业务场景中，您可能需要自定义监控以满足您系统和应用的监控需求，所以容器监控服务在基础监控能力以外，还提供了两种自定义监控方式来允许您通过自己编写数据采集脚本或者自己

暴露 HTTP 监控数据接口来上报自定义的监控数据。容器服务监控框架会按照每分钟采集一次的频率来执行脚本或者调用 HTTP 接口来采集数据。

前提条件

使用自定义监控功能之前，您必须将容器监控服务与第三方监控方案进行集成（详细信息参见 第三方监控方案集成）。

注意：目前，容器服务监控集成默认只支持 InfluxDB 和 Prometheus。

您的自定义监控数据会上报给您的 InfluxDB 或 Prometheus，然后再对接您的数据展示和分析服务。

使用自定义监控脚本上报监控数据

构建 Docker 镜像，在镜像中添加自定义的数据采集脚本。

采集脚本的输出数据必须遵守 InfluxDB 的数据格式协议，如下所示。

```
weather,location=us-midwest temperature=82 1465839830100400200
| -------------------- -------------- |
| | | |
| | | |
+-----------+--------+-+---------+-+---------+
|measurement|,tag_set| |field_set| |timestamp|
+-----------+--------+-+---------+-+---------+
```

更详细的数据格式协议，参见 InfluxDB 数据协议 Line Protocol。

登录 容器服务管理控制台，使用编排模板创建应用，使用 aliyun.monitoring.script 标签声明监控服务用于采集数据的脚本。

示例模板如下所示：

```
custom-script:
image: '您自己的镜像仓库地址'
labels:
aliyun.monitoring.script: "sh gather_mem.sh"
```

aliyun.monitoring.script 定义了监控服务执行应用容器内什么命令来收集监控数据。对应的 label 配置方式如下所示。

```
labels:
aliyun.monitoring.script: "执行脚本的命令"
```

打开 InfluxDB 的 web 端管理界面，查看以数据指标名称为表名的数据库表。

有关如何查看数据库表，参见 第三方监控方案集成 中的相关信息。

使用自定义 HTTP 监控数据接口采集数据

构建 Docker 镜像，在应用中对外暴露 HTTP 接口。

该接口输出监控数据。您可以自定义监控数据的格式，只需符合 JSON 语法即可。此外，由于系统不能分辨自定义的 HTTP 接口返回的 JSON 数据中哪些是数据指标字段，哪些是数据指标的元数据标签，所以您还需要另外一个配置项来指明 JSON 数据中哪些数据的属性是 tag。具体可参考 Telegraf JSON 数据格式。

登录 容器服务管理控制台，使用编排模板创建应用。您需要在模板中填加 aliyun.monitoring.http 标签来声明采集数据的接口，使用 aliyun.monitoring.tags: "您自己的 tag 属性名 1, 您自己的 tag 属性名 2, ..." 声明 HTTP 数据接口返回的数据字段中哪些的属性是 tag。

参考模板：

```
nodejsapp:
command: "bash /run.sh"
ports:
- "3000:3000"
image: '您自己的镜像仓库地址'
labels:
aliyun.monitoring.http: "http://container:3000/metrics/data"
aliyun.monitoring.tags: "tag1,tag2"
```

nodejsapp 应用对外暴露的数据接口 http://container:3000/metrics/data 返回的数据如下所示：

```
{
"tag1": "tag1value",
"tag2": "tag2value",
"field1": 1,
"field2": 2,
"field3": true,
"field4": 1.5
}
```

使用 aliyun.monitoring.tags: "tag1,tag2" 定义上报的 JSON 数据中，tag1 属性和 tag2 属性为上报数据的 tag。

打开 InfluxDB 的 web 端管理界面，查看以 httpjson_ 前缀加容器名称为表名的的数据库表。

例如，容器名称为 nodejsapp_nodejsapp_1，则 InfluxDB 中数据库表的表名为 httpjson_nodejsapp_nodejsapp_1。

有关如何查看数据库表，参见 **第三方监控方案集成** 中的相关信息。

查看服务器监控信息

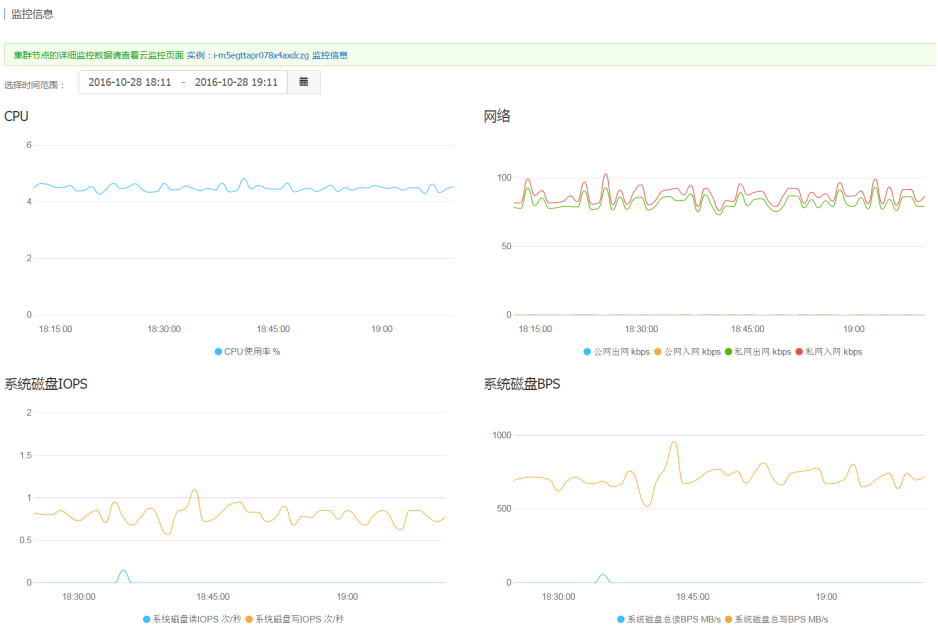
登录 **容器服务管理控制台**。

单击左侧导航栏中的 **集群**。

单击目标集群的名称。

选择所要查看的节点并单击 **监控**。

您可以查看该节点的监控信息。



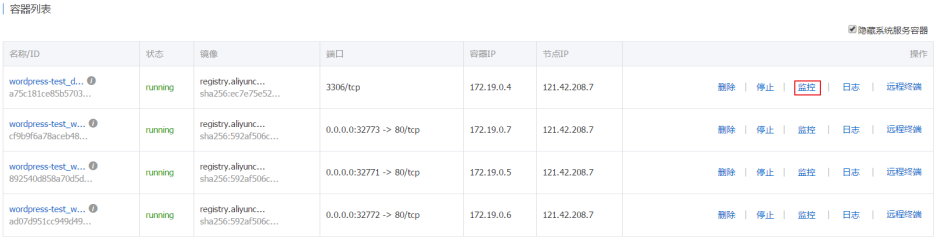
查看容器监控信息

登录 容器服务管理控制台。

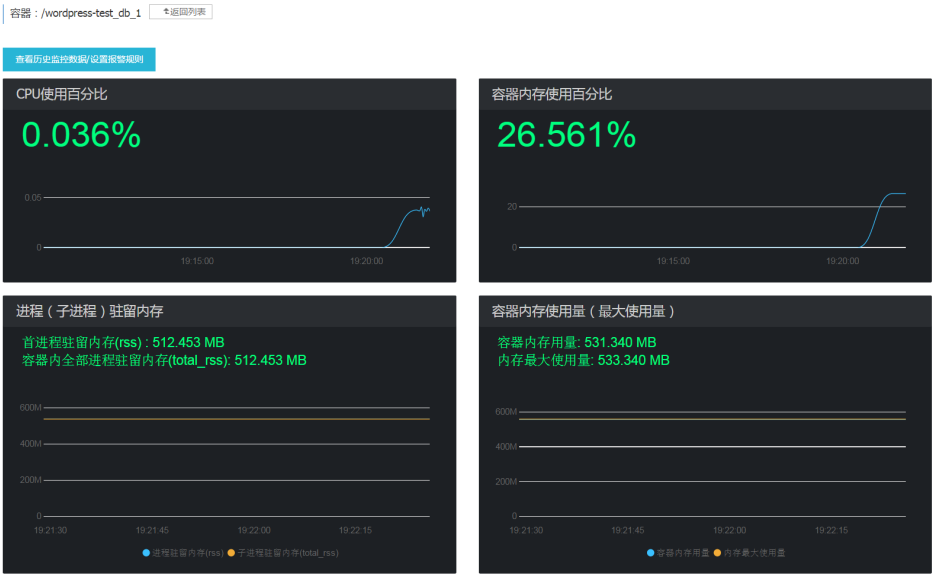
通过以下三种方法之一进入容器列表页面。

- 通过 **节点**。
  i. 单击左侧导航栏中的 **节点**。
  ii. 单击节点的 IP 地址。
- 通过 **应用**。
  i. 单击左侧导航栏中的 **应用**。
  ii. 选择所要查看的应用所在的集群。
  iii. 单击所要查看的应用的名称。
- 通过 **服务**。
  i. 单击左侧导航栏中的 **服务**。
  ii. 选择所要查看的服务所在的集群。
  iii. 单击所要查看的服务的名称。

在容器列表中，选择所要查看的容器并单击 **监控**。



您可以查看该容器的监控信息。



容器监控服务依托于阿里云云监控服务，为容器运维用户提供默认监控、报警规则配置等服务。容器服务的监控服务提供容器维度的监控数据展示以及报警功能。此外，容器服务还提供了与第三方开源监控方案集成的能力（详细信息参见 第三方监控方案集成）。

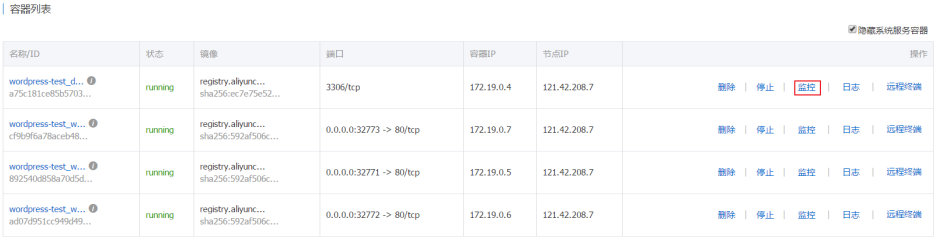
操作流程

登录 容器服务管理控制台。

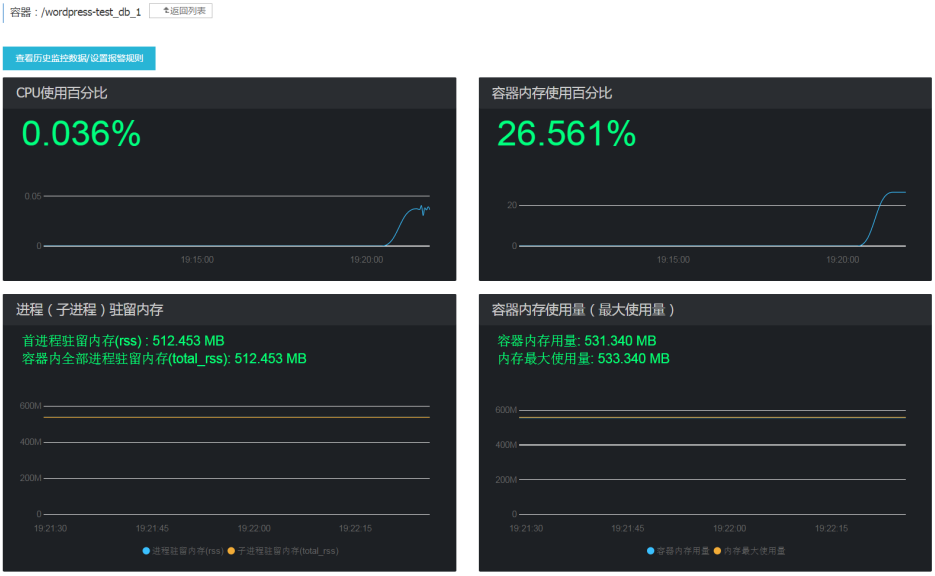
通过以下三种方法之一进入容器列表页面。

- 通过 **节点**。
  i. 单击左侧导航栏中的 **节点**。
  ii. 单击节点的 IP 地址。
- 通过 **应用**。
  i. 单击左侧导航栏中的 **应用**。
  ii. 选择所要查看的应用所在的集群。
  iii. 单击所要查看的应用的名称。
- 通过 **服务**。
  i. 单击左侧导航栏中的 **服务**。
  ii. 选择所要查看的服务所在的集群。
  iii. 单击所要查看的服务的名称。

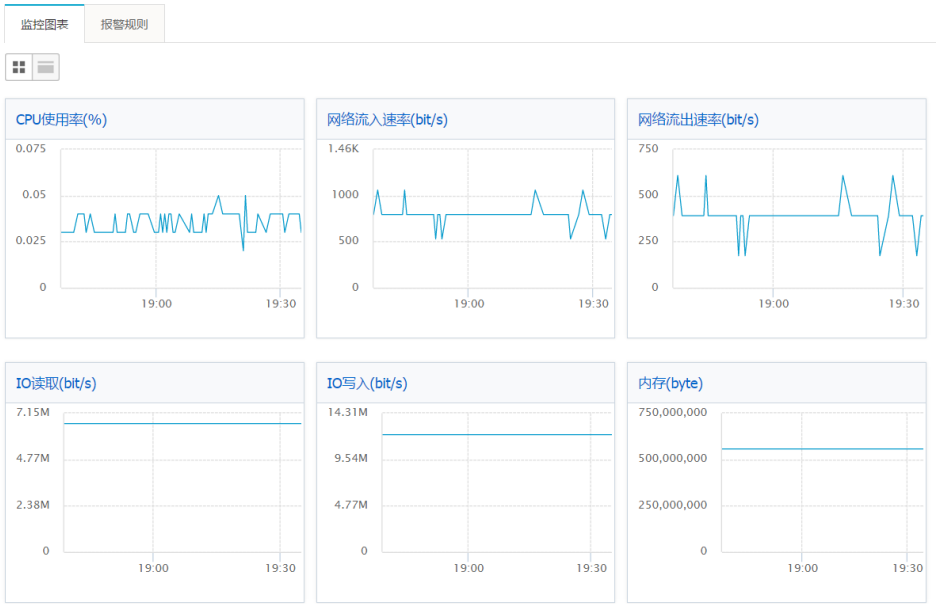
在容器列表中，选择所要查看的容器并单击 **监控**。



您可以查看该容器的实时监控信息。



单击 **查看历史监控数据/设置报警规则**，跳转到阿里云云监控管理控制台，查看该容器对应的历史监控数据。



设置报警规则。

在某些关键服务中，您可以根据自己业务的实际情况添加报警规则。容器监控服务会在监控指标达到告警阈值后短信通知云账号联系人。

i. 单击页面右上角的 **新建报警规则。**

ii. 基于您的实际业务需求设置报警规则并单击 **下一步。**

iii. 设置报警的联系人通知组并单击 **确定。**

iv. 页面显示已完成对实例的报警规则设置。单击 **关闭。**

v. 单击 **报警规则**，查看创建的报警规则以及其相关记录。

为了满足应用在不同负载下的需求，容器服务不仅提供了容器级别的弹性伸缩，还提供了节点级别的自动伸缩，即通过监测节点的资源占用情况自动调整节点数量。

节点伸缩采取的策略：

当监测指标值超过所设定的扩容条件，以用户设定的扩容步长增加节点数量。

当监测指标值低于所设定的缩容条件，以系统默认步长 1 减少节点数量。

自动伸缩的监测指标：

集群 CPU 平均使用量。

集群内存平均使用量。

使用说明

容器服务在判断监测指标是否超出所设定的上下限时，使用的是采集周期（一分钟）内监测指标的平均值（即 CPU 平均使用量和内存平均使用量），而且只有当连续三个采集周期内的监测指标平均值

均超出所设定的上下限时，容器服务才会触发扩容或伸缩操作，以避免因为监控数据抖动而引起频繁的扩容或缩容操作。

节点缩容只会对通过节点扩容创建出来的节点进行，您手动创建或者添加的节点不受影响。如果您希望这些手动添加的节点也可以自动缩容，需要为这些节点添加以下标签。

aliyun.reschedule==true

节点缩容的时候，系统会删除集群里的 ECS 实例，您需要提前做好数据备份。

不要调度有状态服务到可缩容的节点上。相关信息，参见 Docker Compose 中的 constraint。

扩容出来的 ECS 实例不会影响已经部署的容器。新部署的容器会根据容器的部署规则进行部署。

- 节点缩容时，容器服务会把删除掉的 ECS 实例上的容器迁移到其它 ECS 实例上。

操作步骤

登录 **容器服务管理控制台**。

单击左侧导航栏中的 **集群**。

在 **集群列表** 页面，选择要设置的集群，点击 **管理**。

单击左侧导航栏中的 **节点伸缩**，单击 **请新建自动伸缩规则**。

配置伸缩规则，并单击 **下一步**。

约束规则：

- **扩容条件** 的可选范围是 50%~100%，**缩容条件** 的可选范围是 0%~50%。
- **扩容条件** 和 **缩容条件** 的差值不能小于30%。
- **扩容步长** 的可选范围是 1~5， **缩容步长** 目前默认是 1，不支持配置。
- 设置 **集群最小节点数** 和 **集群最大节点数**。缩容时，如果节点数 ≤ **集群最小节点数**，不会进行缩容操作；扩容时，如果节点数 ≥ **集群最大节点数**，不会进行扩容操作。

注意：

- 建议不要同时设置基于 CPU 使用量和内存使用量的复合伸缩规则。
- 请谨慎设置伸缩策略。如果在您设置伸缩规则的时候，集群就满足所设置的伸缩条件而且伸缩后集群仍然满足伸缩条件，那么监控将会不断地触发伸缩。

配置实例规格，并单击 **确认配置**。

有关实例规格配置的信息，参见 **创建集群**。

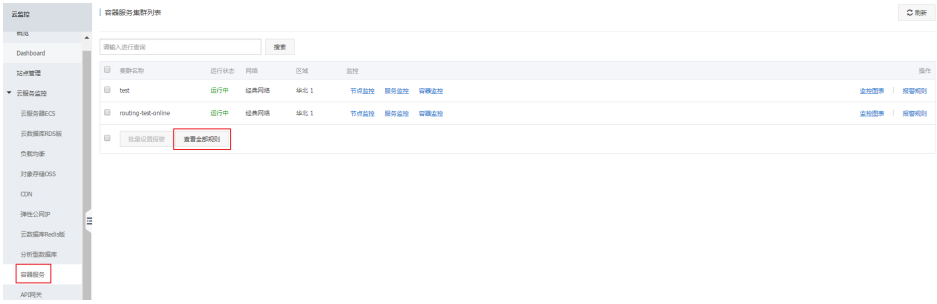
查看监控指标

单击左侧导航栏中的 **集群**。

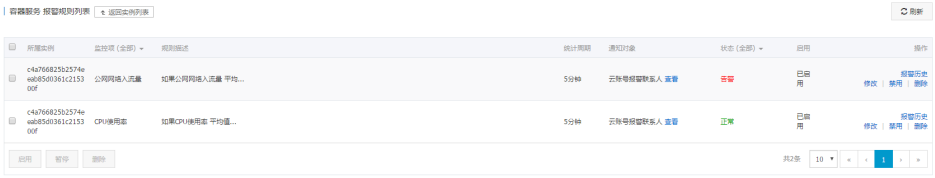
在 **集群列表** 页面，选择要设置的集群，单击 **监控**。

进入云监控管理控制台，您可以看到集群的监控信息，如下图所示。

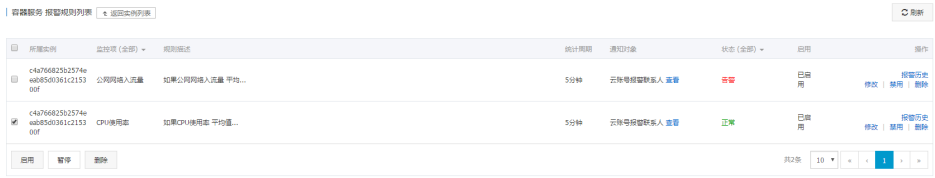
单击左侧导航栏中的 **容器服务**，查看集群列表。



单击 **查看全部规则**，可以看到弹性伸缩自动设置的报警规则。



选中一个报警规则，您可以修改报警条件及通知人（支持短信，邮件等通知方式），也可以禁用报警规则。



容器服务提供了与第三方开源监控方案集成的能力。

注意：目前，容器服务监控集成默认只支持 InfluxDB 和 Prometheus。

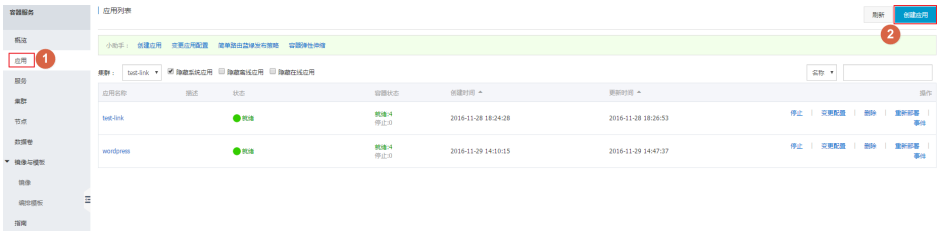
下面的示例以 InfluxDB 为例介绍如何进行容器服务的第三方监控方案集成。

操作流程

登录 容器服务管理控制台。

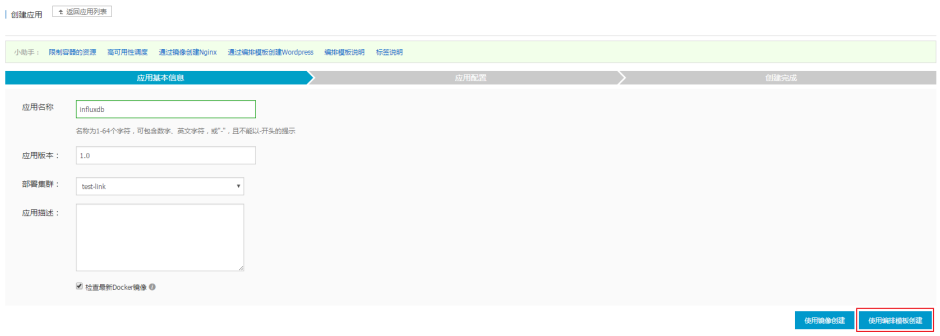
单击左侧导航栏中的 **应用**。

单击页面右上角的 **创建应用**。



填写应用的基本信息并单击 **使用编排模板创建**。

本示例创建名为 **influxdb** 的应用。

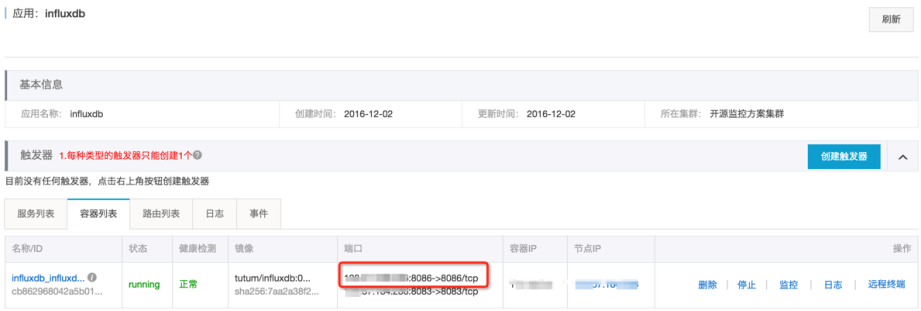


填写下面的编排模板并单击 **创建并部署**。

注意：在实际生产环境中，本示例中的模板需要做一些修改，其中 influxdb 的服务定义部分不要对宿主机暴露端口。

```
version: '2'
services:
#定义 influxdb
influxdb:
image: tutum/influxdb:0.9
ports:
- "8083:8083" #暴露 Web 界面端口
- "8086:8086" #暴露数据 API Web 接口端口
```

应用创建成功后，在应用列表页面，单击本示例所创建应用的名称 **influxdb**， 查看应用详情。单击 **容器列表**，查看当前应用对外暴露的节点 IP 和端口号，并复制该值（本示例中，复制 8086 端口对应的节点 IP 和端口号；该信息为 **influxdb** 对外暴露的数据上报地址），如下图所示。



单击左侧导航栏中的 **应用**，返回应用列表页面。单击 **更新配置**，在现有的模板中，添加以下内容来声明 InfluxDB 和容器监控服务的集成并单击 **确定**。

说明： 第三方开源监控集成目前只支持 InfluxDB 和 Prometheus，label 分别为 aliyun.monitoring.addon.influxdb 和aliyun.monitoring.addon.prometheus。标签取值的格式必须为 schema:hostIp:port。

```
labels
aliyun.monitoring.addon.influxdb:"http://刚才复制的节点 IP：端口号"
```

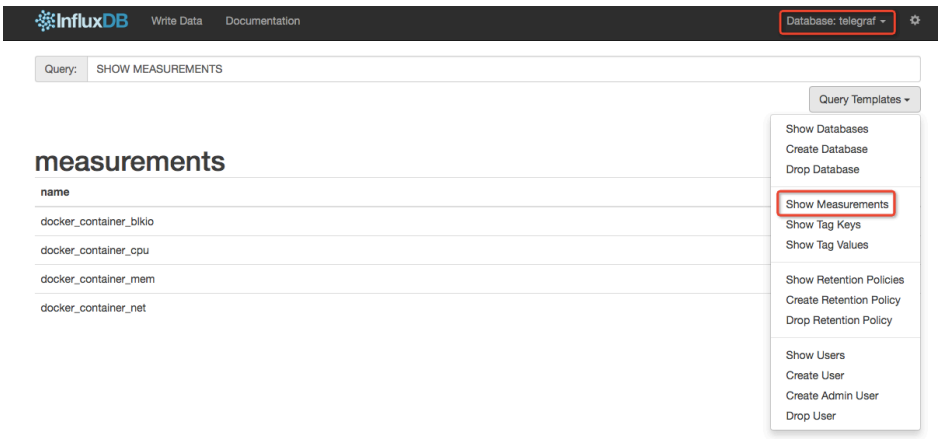
由于容器监控服务的 Agent 采用了 host 网络模式，容器服务无法使用 link 来识别 InfluxDB，所以您需要先创建 **influxdb**，再将 **influxdb** 对外暴露的数据上报地址添加到应用 labels 中，来通知数据采集客户端。完成以上步骤以后，监控服务会将采集到的容器运行状态数据自动写入 influxdb 中。

在应用列表页面，单击本示例所创建应用的名称 **influxdb** 并单击 **容器列表**。复制 **influxdb** 容器对外暴露的端口。

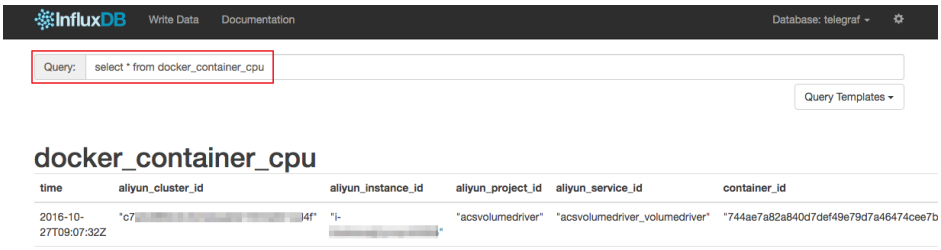
在浏览器中访问 InfluxDB 管理页面，查看容器监控服务写入的各项指标数据，如下图所示。

i. 选择 **telegraf**。
ii. 单击 **Query Templates** 并在下拉菜单中单击 **Show Measurements**。
iii. 按 Enter 键。

您可以查看数据库表，如下图所示。



查看某个表的数据详情，如下图所示。



后续操作

容器服务与 InfluxDB 集成之后，您可以根据自己的情况选择其他数据展示图表框架，比如 Grafana 等。

授权管理

简介

如果您使用容器服务创建了多个集群，您的组织里有多个用户需要使用这些集群。如果这些用户共享使用您的云账号秘钥，那么会存在以下的问题：

- 您的秘钥由多人共享，泄密风险高。
- 您无法限制用户的访问权限，容易出现误操作导致安全风险。

访问控制 RAM (Resource Access Management) 是阿里云提供的资源访问控制服务。通过 RAM，您可以集中管理您的用户，以及控制用户可以访问您名下哪些资源的权限。对于容器服务而言，90% 以上的用户主要分配的是集群维度的权限，主要是对集群维度的是否可增删的限制。因此为了更简单地使用权限控制，容器服务提供了两种鉴权策略，一种是资源授权，一种是细粒度 API 授权。对于 90% 的用户而言使用资源分配鉴权策

略即可满足需求。对于 RAM 有深入的了解希望在 API 级别进行鉴权的用户可以使用 API 细粒度鉴权。

使用场景

假设是一家 APP 开发的公司，主要使用阿里云的服务提供 APP 的后端服务与基础设施和中间件，公司中的成员有后端开发工程师，测试工程师，运维工程师、基础平台工程师。从职责分配上来讲：

- 后端开发工程师负责开发 APP 的后端服务。
- 测试工程师负责测试 APP 的后端服务和基础的测试环境维护。
- 运维工程师负责所有的环境的运维（集群的伸缩，监控等）。
- 基础平台工程师负责中间件的维护与基础设计的建设，包括 GitLab、Jenkins、Sentry等。

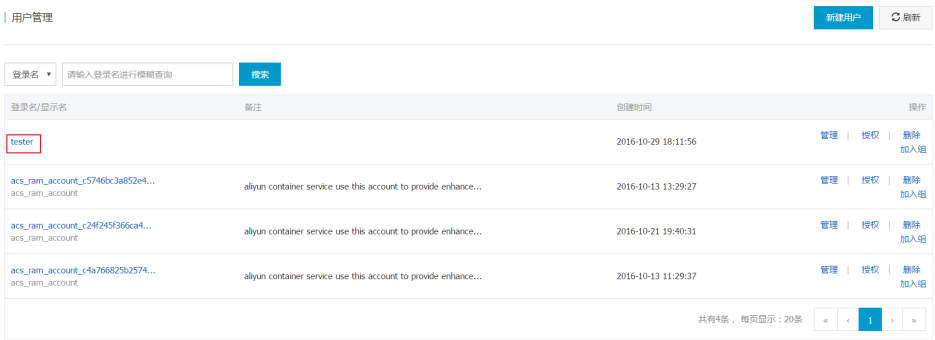
从职责划分上来看，不同的角色对于资源的需求不同，为了保证每个角色有足够的权限完成自己的任务，同时权限不越界。在这个场景中您使用资源分配鉴权即可。

操作步骤

注意：当您进入 RAM 控制台的时候，可能会发现一些以 acs_ 开头的 RAM 子账号，这些子账号是容器服务自动创建的子账号，请不要删除。

主账号授权

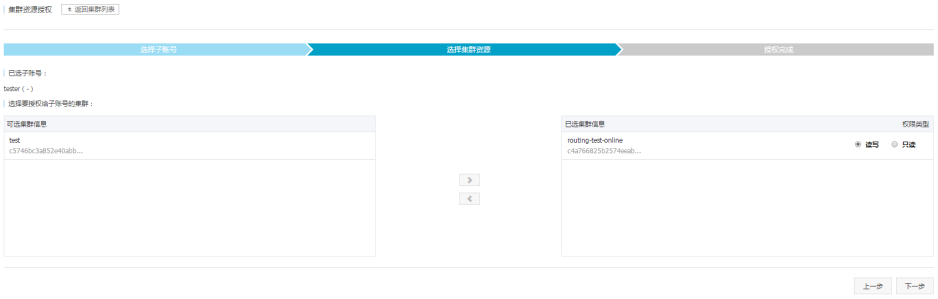
登录 **访问控制管理控制台**，创建一个子账号并开启控制台登录。



登录 **容器服务管理控制台**，单击左侧导航栏中的 **集群** 并单击 **子账号授权**。

选择相应的子账号并单击 **下一步**。

选择对应的集群资源和权限，单击 **下一步** 并在弹出的确认对话框中单击 **确定**。



单击 **完成** 完成授权。

使用子账号

使用子账号登录（要主账号进入 RAM 控制台并将控制台登录地址告知子用户）并进入 **容器服务管理控制台**。

此时，集群列表中显示被分配的集群。

深入理解鉴权策略

鉴权策略简单地来讲就是对云资源的使用权限的校验。鉴权分为两种，一种是资源分配鉴权，一种是 API 细粒度鉴权。

API 细粒度鉴权是基于访问控制 RAM （Resource Access Management）进行的鉴权，相对来讲更加复杂。

资源分配鉴权是容器服务在 API 细粒度鉴权之上的包装，主要将 API 细粒度鉴权无法实现的资源分配在容器服务中进行了实现，并将 API 细粒度鉴权的鉴权策略封装为只读和读写两个权限。只读权限可以创建应用、删除应用等等，但是无法对集群级别的增删改查进行操作。而读写权限则可以对集群进行完全的管理。

高级用法

API 细粒度鉴权可以实现 API 级别的鉴权。目前提供的后置鉴权条件如下所示。

鉴权 Action 名称	解释
GetClusterById	获取集群描述

GetClusterCerts	获取集群证书
CreateCluster	创建集群
DeleteCluster	删除集群
UpdateClusterSizeById	集群扩容

资源列表如下所示。

资源	解释
cluster	集群

参考示例：

```
{
"Statement": [
{
"Action": "cs:*",
"Effect": "Allow",
"Resource": "acs:cs:*:*:cluster/cc6b56877fd64407fb615dd09ff85303e"
},
{
"Action": "cs:*",
"Effect": "Allow",
"Resource": "acs:cs:*:*:cluster/cee52159dd72d4ead9c0ee1b1708b7065"
}
],
"Version": "1"
}
```

本示例表示授予集群 cc6b56877fd64407fb615dd09ff85303e 与 cee52159dd72d4ead9c0ee1b1708b7065 的全部权限。

更多的用法参见 RAM 的文档。

构建管理

容器镜像服务支持构建的源代码仓库包括阿里云 Code 仓库，Github 仓库，Bitbucket 仓库，本地直接推送镜像到容器镜像服务仓库。

操作流程

创建镜像仓库

登陆 容器镜像服务控制台。

单击左侧导航栏中的 **镜像列表**，单击右上角的 **创建镜像仓库**，如下图所示。

填写自己的仓库名称。

填写摘要（必填项）。

填写可选的描述信息。

选择仓库类型，公开或者私有，如下图所示。

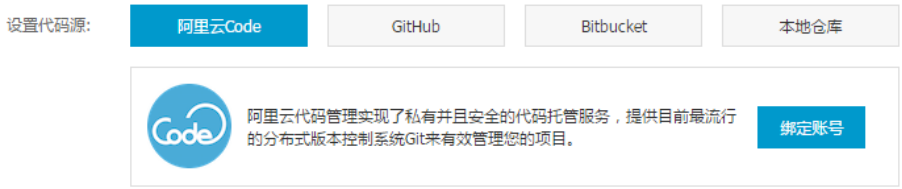
设置代码源

容器镜像服务支持构建的源代码仓库包括阿里云 Code 仓库，Github 仓库，Bitbucket 仓库以及本地仓库。本地仓库仅支持将本地已经构建好的镜像推送到容器镜像服务仓库，其他源代码仓库支持镜像的自动构建。

进入相应的源代码仓库获得正确的授权，即授权容器镜像服务拉取其他源代码仓库的权限。目前支持通过 Git 版本管理系统的方式进行拉取。

选择相应的代码命名空间和项目。

注意：项目下面必须要有用于镜像构建的 Dockerfile 以及构建上下文目录（即 Dockerfile 所在的目录）。



设置构建规则并单击 **创建镜像仓库**。

选择是否在代码仓库发生变更时自动触发构建镜像。

选择是否使用海外机器进行构建。

选择是否在构建过程中使用缓存。如果构建中执行的 RUN 命令会执行去其他网址拉取更新内容的操作，需要禁用缓存。

选择正确的源代码分支或者 Tag。

填写正确的 Dockerfile 文件所在的目录名称，默认为根目录 /。

填写正确的构建文件名称，默认为 Dockerfile。

填写构建时镜像的版本名称，一个镜像支持多个版本名称，用逗号（，）分隔。

点击 **添加一条构建规则**，支持通过指定多条构建规则，来一次性构建多个镜像。

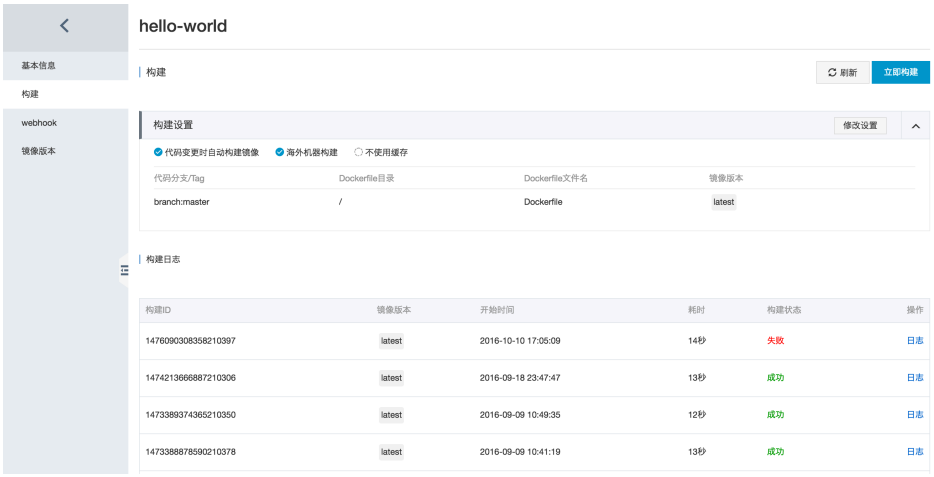
执行构建

回到 **镜像列表** 页面，找到创建成功的仓库，本示例中为 hello-world，单击 **管理**。

单击左侧导航栏中的 **构建**，进入构建选项页。

单击 **立即构建**，开始执行构建。

单击 **日志**，可以查看实时的构建日志。



有用户容易将 Dockerfile 文件内的指令理解为 shell 脚本，其实并不是，Dockerfile 仅支持少部分的指令，下面介绍这些指令和功能。

FROM

```
FROM <image>
```

或者

```
FROM <image>:<tag>
```

或者

```
FROM <image>@<digest>
```

指定构建依赖的基础镜像，FROM 指令必须作为 Dockerfile 中第一条没有被注释的指令。

MAINTAINER

```
MAINTAINER <name>
```

指定镜像的 Author 字段。

RUN

shell 形式，command 作为 /bin/sh -c 的参数进行执行，即为 shell 的子进程。

```
RUN <command>
```

exec 形式，直接执行。

```
RUN ["executable", "param1", "param2"]
```

RUN 指令是在当前镜像上执行命令，并且提交执行之后的结果，作为最新的一层 layer，并且后续的 Dockerfile 指令会在 RUN 指令执行完生成的最新镜像上继续执行。

CMD

exec 形式，直接执行，推荐使用该形式。

```
CMD ["executable","param1","param2"]
```

第二种形式，作为ENTRYPOINT指令的默认参数。

```
CMD ["param1","param2"]
```

第三种形式，shell形式，作为/bin/sh -c的参数进行执行，即为 shell 的子进程。

```
CMD command param1 param2
```

在一个 Dockerfile 文件中，只能有一个CMD指令，如果有多于一条CMD指令，那只有最后一条CMD指令会生效。

CMD指令的主要目的是提供容器运行时的默认值，这些默认值可以包括一个可执行文件名，加上执行时的一些参数，或者不包含可执行文件名，只提供参数，但是必须通过增加一个ENTRYPOINT指令来指定可执行文件名。

LABEL

```
LABEL <key>=<value> <key>=<value> <key>=<value> ...
```

LABEL指令给一个镜像增加元信息metadata。一个LABEL是一个键值对。如果LABEL值中需要包含空格或者换行符，使用双引号"或者反斜杠\。

下面是一些用例：

```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

EXPOSE

```
EXPOSE <port> [<port>...]
```

EXPOSE指令设置 Docker 容器在运行时监听指定网络端口。EXPOSE指令并不会使得容器所在的主机可以访问容器的端口。为了使主机可以访问容器端口，必须使用 -p 或者 -P 参数。

ENV

```
ENV <key> <value>
ENV <key>=<value> ...
```

ENV指令设置镜像的环境变量，可在实际启动容器时使用docker run --env <key>=<value>进行覆盖。

ADD

有两种形式：

```
ADD <src>... <dest>
```

第二种形式用于路径或者文件名包含空格的情况。

```
ADD ["<src>",... "<dest>"]
```

如果src是文件路径，则必须是相对于构建上下文context的相对路径，且不能引用构建上下文目录之外的内容。dest必须是绝对路径，或者是工作路径WORKDIR的相对路径。如果dest不存在，则将自动创建，如果dest不以/结尾，则将被认为是一个文件，而不是目录。

COPY

有两种形式：

```
```

```
COPY <src>... <dest>
```

第二种形式用于路径或者文件名包含空格的情况。

```
COPY ["<src>",... "<dest>"]
```

与ADD类似，区别在于src不能是网络链接 URL。

ENTRYPOINT

exec 形式，推荐使用该形式。

```
ENTRYPOINT ["executable", "param1", "param2"]
```

shell 形式，command作为/bin/sh -c的参数进行执行，即为 shell 的子进程。

```
ENTRYPOINT command param1 param2
```

ENTRYPOINT指令允许您指定容器启动时的启动进程。

VOLUME

```
VOLUME ["/data"]
```

VOLUME指令指定了一个挂载点，并给该挂载点命名，表明该挂载点的数据卷来自于主机的某个目录或者共享了其他容器的目录，该挂载点的内容不会随镜像的分发而分发。

USER

```
USER daemon
```

USER指令设置启动镜像时的用户或者UID，随后所有在Dockerfile文件内的RUN，CMD以及ENTRYPOINT指令都将该用户作为执行用户。

WORKDIR

```
WORKDIR /path/to/workdir
```

WORKDIR 指令设置工作目录，随后所有在Dockerfile文件内的RUN，CMD以及ENTRYPOINT指令都将该目

录作为当前目录，并执行相应的命令。

对 Docker 进行构建前需要安装必要的软件。

- docker-engine 下载页面。
- docker-toolbox（for MacOS and Windows 用户） 下载页面。

目前构建镜像有以下两种方式。

- 通过 Docker Hub 来自动构建。自动推送到阿里云镜像仓库，需要您将构建的 Dockerfile 及相关的上下文（context）上传到 GitHub 或者 Bitbucket 进行构建。同时支持持续集成，即您上传代码到 GitHub 或者 Bitbucket 之后会触发自动构建。
- 您在自己的机器上进行构建，然后推送到阿里云镜像仓库。

在镜像中利用国内软件源加速软件下载和更新

使用已经配置了阿里云镜像软件源的容器镜像

- docker pull registry.cn-hangzhou.aliyuncs.com/acs/ubuntu
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/centos
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/debian
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/alpine
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/node
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/python
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/django
- docker pull registry.cn-hangzhou.aliyuncs.com/acs/ruby

您自行在镜像中配置阿里云镜像软件源

操作流程

访问 阿里云软件源镜像网站。

找到您需要的软件源对应的帮助页面，例如 ubuntu 软件源镜像配置帮助页。

查看帮助页，学习如何配置软件源镜像信息。

在原有容器镜像的基础上添加该软件源配置文件作为 Dockerfile 依赖的上下文（context）文件，通过 Dockerfile 制作新的镜像。

通过您本地或者 Docker Hub 的镜像自动构建机制进行镜像的构建。

将镜像推送到阿里云镜像仓库，示例 docker push registry.aliyuncs.com/sample/demo。

示例：

ubuntu:12.04,14:04 GitHub 地址

ubuntu:14:04 Dockerfile

```
FROM ubuntu:14.04
MAINTAINER Li Yi <denverdino@gmail.com>
RUN sed -i 's/archive.ubuntu.com/mirrors.aliyun.com/' /etc/apt/sources.list
```

centos:6, centos:7 GitHub 地址

- nodejs:4.2, nodejs:5.3 GitHub 地址
- pyhton:2.5, python:3.5 GitHub 地址

获取和更新包列表失败

如果您获取和更新包列表（例如 apt-get update）失败，尝试以下解决方法。

- 使用上面提到的方法采用国内的软件源镜像替换更新国外软件源镜像。
- 跳过更新，继续执行，例如使用命令 sudo apt-get update || true && sudo apt-get install python。

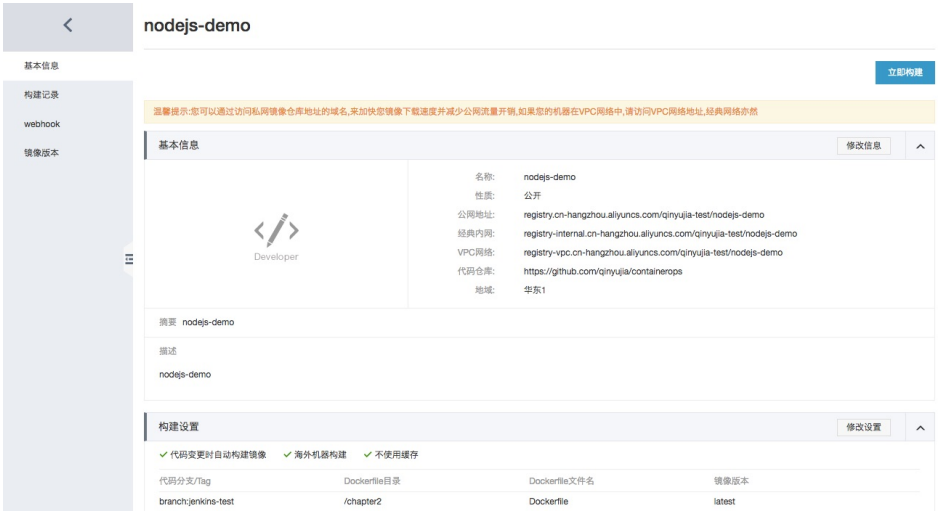
DevOps

本章节主要是介绍如何通过添加触发器和设置 Webhook 实现自动重新部署应用。

操作步骤

假设您已经有一个部署在阿里云容器服务上的 nodejs 应用。该应用代码托管在 GitHub 中，镜像仓库使用的是阿里云 Docker Hub，镜像仓库设置了代码变更时自动构建镜像的构建设置。

Docker Hub 中 的镜像如下所示。



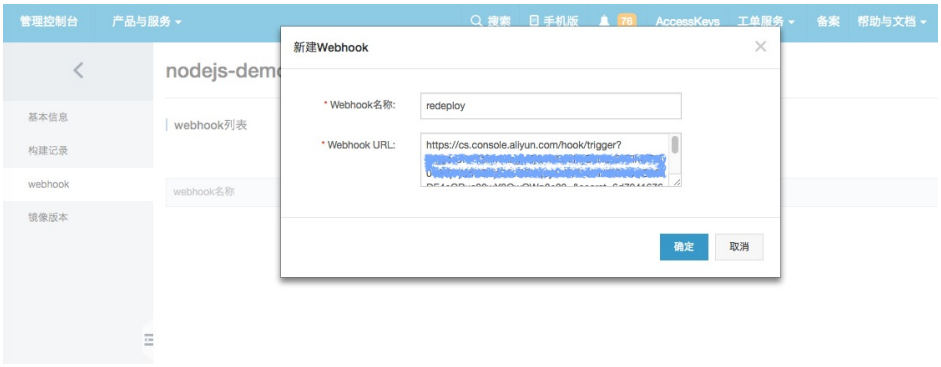
初始的编排模板如下所示。

```
nodejs-demo:
image: 'registry.cn-hangzhou.aliyuncs.com/qinyujia-test/nodejs-demo'
expose:
- '22'
- '3000'
restart: always
labels:
aliyun.routing.port_3000: nodejs-demo
```

应用如下所示。

为 nodejs 应用创建重新部署类型的触发器。

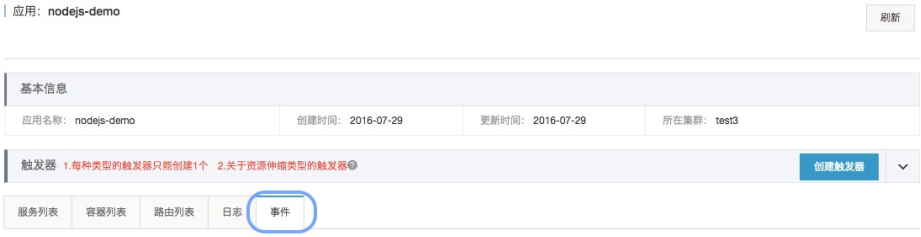
在 Docker Hub 中添加一条 Webhook 记录，将上一步创建的触发器链接填写到 Webhook URL 中。



此时在 GitHub 中提交代码变更，镜像就会自动重新构建，应用会自动重新部署。

确认更新。

通过查看应用的事件来确认更新。



通过 Webhook 的访问历史来确认更新。

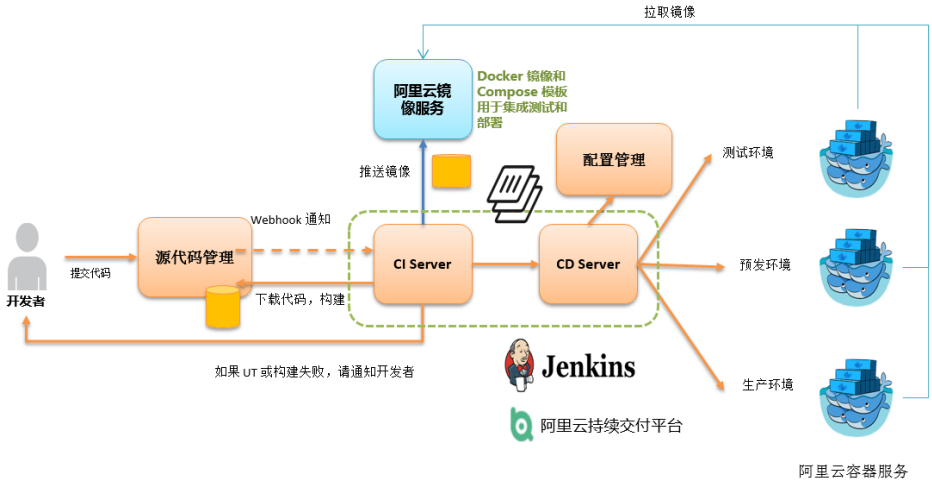
持续集成作为敏捷开发重要的一步，其目的在于让产品快速迭代的同时，尽可能保持高质量。每一次代码更新，都要通过自动化测试来检测代码和功能的正确性，只有通过自动测试的代码才能进行后续的交付和部署。本文主要介绍如何将时下最流行的持续集成工具之一的 Jenkins 结合阿里云容器服务，实现自动测试和镜像构建推送。

以下内容演示如何通过阿里云容器服务 Jenkins 实现自动测试和 Docker 镜像构建，实现高质量的持续集成。

背景信息

每次代码提交到 GitHub 上的 nodejs 的项目后，阿里云容器服务 Jenkins 都会自动触发单元测试，测试通过则继续镜像构建及推送到目标镜像仓库中，最后邮件通知结果。

大致流程如下图所示：



slave-nodejs 用于进行单元测试，构建镜像和推送镜像的 slave 节点。

Jenkins 相关介绍

Jenkins 是基于 Java 开发的一种开源持续集成工具，监控并触发持续重复的工作，具有开源，支持多平台和插件扩展，安装简单，界面化管理等特点。Jenkins 使用 job 来描述每一步工作。节点是用来执行项目的环境。Master 节点是 Jenkins job 的默认执行环境，也是 Jenkins 应用本身的安装环境。

master/slave

master/slave 相当于 server 和 agent 的概念。master 提供 web 接口让用户来管理 job 和 slave，job 可以运行在 master 本机或者被分配到 slave 上运行。一个 master 可以关联多个 slave 用来为不同的 job 或相同的 job 的不同配置来服务。

可以通过配置多个 slave 为不同项目准备单独的测试和构建环境。

注意：本文中提到的 Jenkins job 和项目均指的是 Jenkins 一个构建单元，执行单元。

使用容器服务部署 Jenkins 应用和 slave 节点

不同应用构建、测试所需要的依赖不同，最佳实践是使用包含相应的运行时依赖和工具的不同 Slave 容器执行测试和构建。通过阿里云容器服务提供的针对 Python、NodeJS、go 等不同环境的 Slave 镜像以及示例模板，用户可以简单快速地生成 Jenkins 应用以及各种 slave 节点，在 Jenkins 应用中配置节点信息，然后在构建项目中指定执行节点，从而实现整个持续集成流程。

注意：有关阿里云容器服务提供的开发 slave 节点镜像，参见 https://github.com/AliyunContainerService/jenkins-slaves。

创建 Jenkins 编排模版。

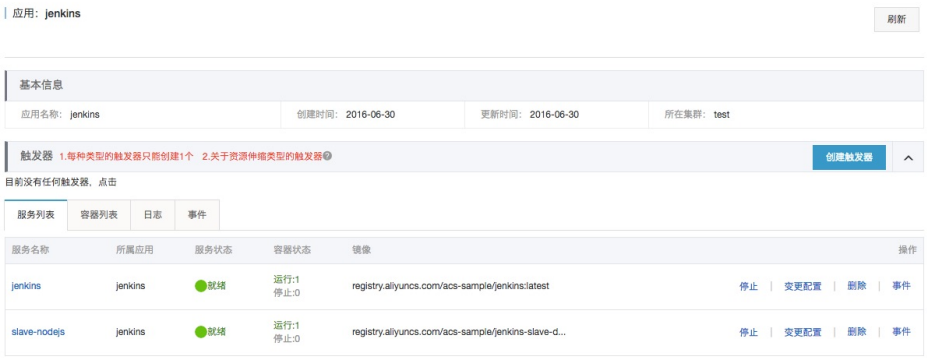
新建模版，以如下内容为基础，创建编排。

```
jenkins:
image: 'registry.aliyuncs.com/acs-sample/jenkins:1.651.3'
ports:
- '8080:8080'
- '50000:50000'
volumes:
- /var/lib/docker/jenkins:/var/jenkins_home
privileged: true
restart: always
labels:
aliyun.scale: '1'
aliyun.probe.url: 'tcp://container:8080'
aliyun.probe.initial_delay_seconds: '10'
aliyun.routing.port_8080: jenkins
links:
- slave-nodejs
slave-nodejs:
image: 'registry.aliyuncs.com/acs-sample/jenkins-slave-dind-nodejs'
restart: always
volumes:
- /var/run/docker.sock:/var/run/docker.sock
labels:
aliyun.scale: '1'
```

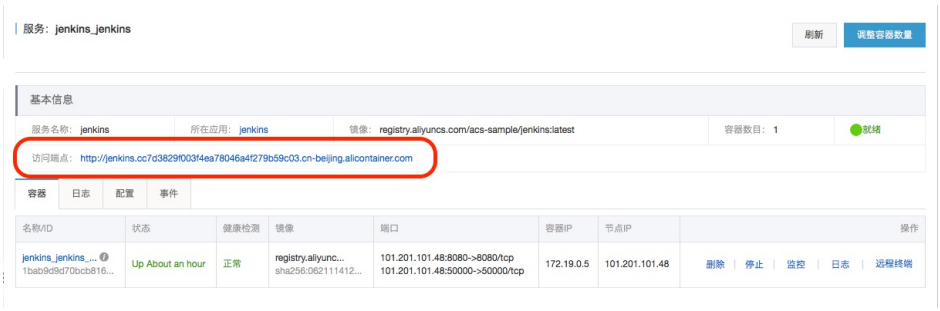
使用模板创建 Jenkins 应用和 slave 节点。

也可以直接使用阿里云容器服务提供的 Jenkins 示例模版创建 Jenkins 应用和 slave 节点。

创建成功后，Jenkins 应用和 slave 节点显示在服务列表中。

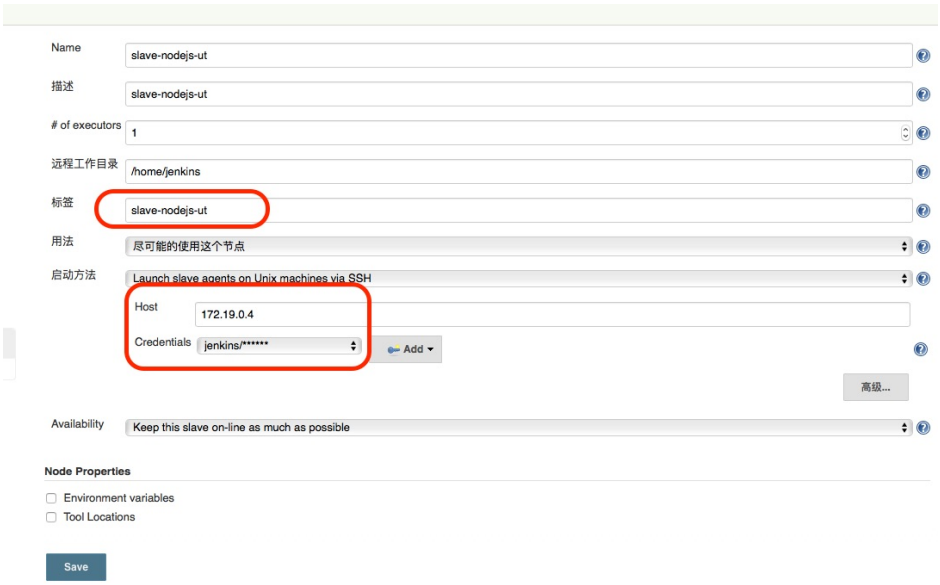


打开容器服务提供的访问端点，就可以使用刚刚部署的 Jenkins 应用。



实现自动测试及自动构建推送镜像

将 slave 容器配置成 Jenkins 应用的 slave 节点。 打开 Jenkins 应用，进入系统设置界面，选择管理节点，新建节点，配置相应参数。如下图所示。



注意：

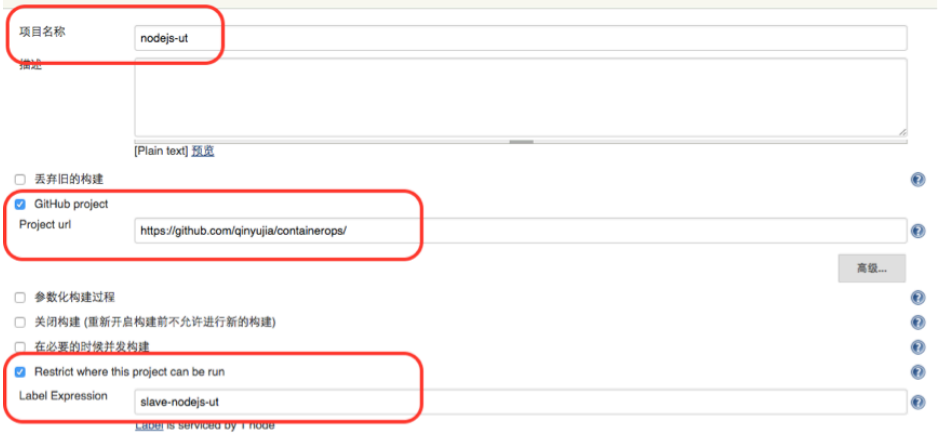
- 标签是 slave 的唯一标识。
- slave 容器和 Jenkins 容器同时运行在阿里云平台上，因此需要填写外网访问不到的容器节点 IP，隔离测试环境。

- 添加 Credential 的时候，使用创建 slave-nodejs 镜像的 Dockerfile 里的 jenkins 用户（初始密码为 jenkins）。镜像 Dockerfile 的地址为 https://github.com/AliyunContainerService/jenkins-slaves/tree/master/jenkins-slave-dind-nodejs。

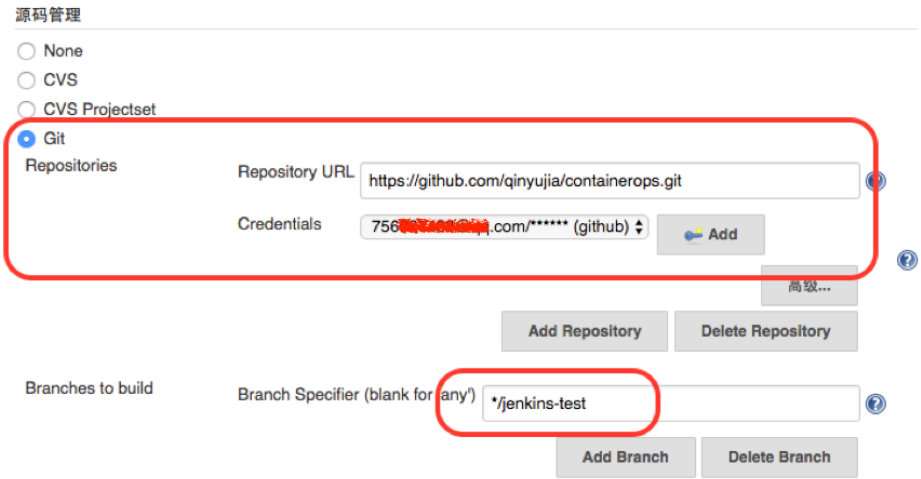
创建项目实现自动化测试。

i. 新建 Item，选择构建一个自由风格的软件项目。

ii. 填写项目名称，并选择项目运行节点。此示例中填写上述准备的 slave-nodejs-ut 节点。



iii. 配置源码管理和代码分支。此示例中源代码使用 GitHub 管理。

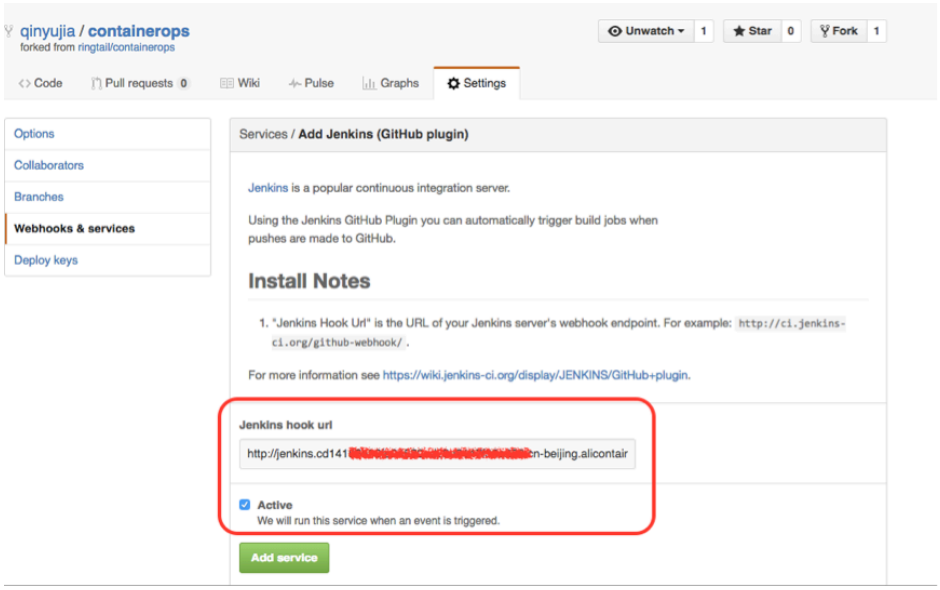


iv. 配置构建触发器，此示例采用结合 GitHub Webhooks & services 实现自动触发项目执行。

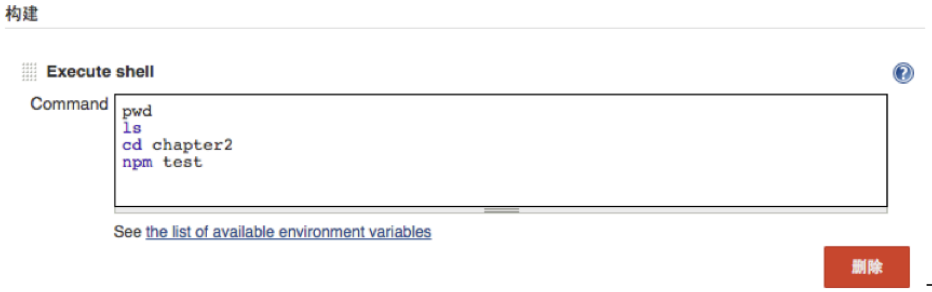
在 GitHub 中添加 Jenkins 的 service hook，完成自动触发实现。

在 GitHub 项目主页单击 **Settings**，单击左侧菜单栏中的 **Webhooks & services**，单击 **Add Service**，在下拉框中选择 **Jenkins(Git plugin)**。在 **Jenkins hook url** 对话框中填写 ${Jenkins IP}/github-webhook/，例如：

```
http://jenkins.cd**********.cn-beijing.alicontainer.com/github-webhook/
```



增加 Execute shell 类型的构建步骤，编写 shell 脚本执行测试。

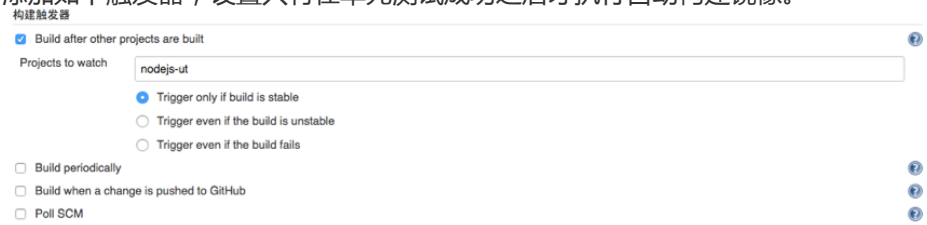


本示例中的命令如下所示：

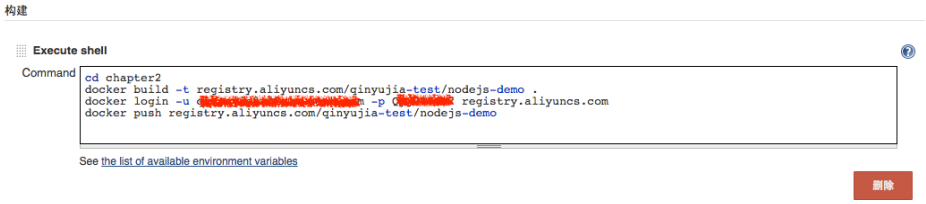
```
pwd
ls
cd chapter2
npm test
```

创建项目实现自动构建，推送镜像。

i. 新建 Item，选择构建一个自由风格的软件项目。
ii. 填写项目名称，并选择项目运行节点。此示例中填写上述准备的 slave-nodejs-ut 节点。
iii. 配置源码管理和代码分支。此示例中源代码使用 GitHub 管理。
iv. 添加如下触发器，设置只有在单元测试成功之后才执行自动构建镜像。



v. 编写构建镜像和推送镜像的 shell 脚本。



本示例的命令如下所示：

```
cd chapter2
docker build -t registry.aliyuncs.com/qinyujia-test/nodejs-demo .
docker login -u ${yourAccount} -p ${yourPassword} registry.aliyuncs.com
docker push registry.aliyuncs.com/qinyujia-test/nodejs-demo
```

自动重新部署应用

首次部署应用

使用编排模板，将 **创建项目实现自动构建，推送镜像** 中创建的镜像部署到容器服务中，创建 nodejs-demo 应用。

示例如下所示：

```
```
express:
image: 'registry.aliyuncs.com/qinyujia-test/nodejs-demo'
expose:
- '22'
- '3000'
restart: always
labels:
aliyun.routing.port_3000: express
```
```

自动重新部署

选择刚刚创建的应用 nodejs-demo，创建触发器。

在 **创建项目实现自动构建，推送镜像** 的 shell 脚本中添加一行，地址即为上文创建的触发器给出的触发器链接。

```
curl  'https://cs.console.aliyun.com/hook/trigger?triggerUrl=***==&secret=***'
```

把 **创建项目实现自动构建，推送镜像** 的示例中的命令改为：

```
 cd chapter2
docker build -t registry.aliyuncs.com/qinyujia-test/nodejs-demo .
docker login -u ${yourAccount} -p ${yourPassword} registry.aliyuncs.com
docker push registry.aliyuncs.com/qinyujia-test/nodejs-demo
curl  'https://cs.console.aliyun.com/hook/trigger?triggerUrl=***==&secret=***'
```

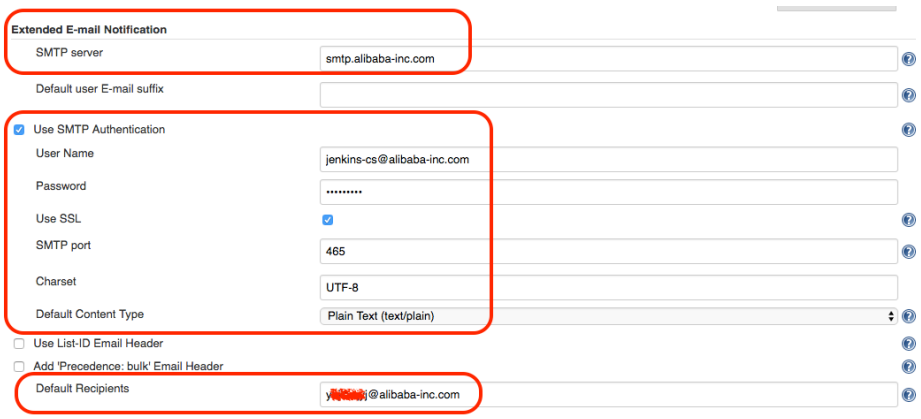
到此，镜像推送之后，Jenkins 会自动触发重新部署 nodejs-demo 应用。

配置邮件推送结果

如果希望单元测试或者镜像构建的结果能够通过邮件推送给相关开发人员或者项目执行发起者。可以通过如下配置来实现。

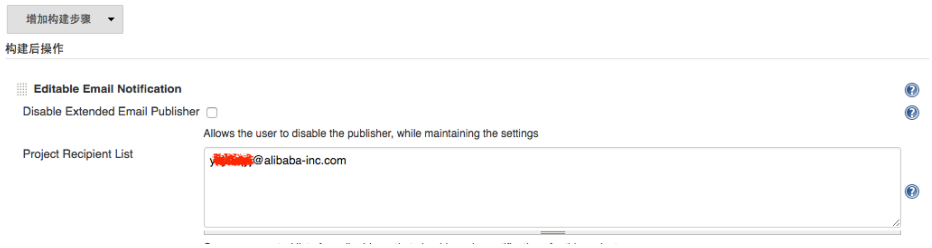
在 Jenkins 主页，选择系统管理，系统设置，配置 Jenkins 系统管理员邮箱。

安装 Extended Email Notification plugin，配置 SMTP server 等相关信息，并设置默认邮件接收人列表。如下图所示。

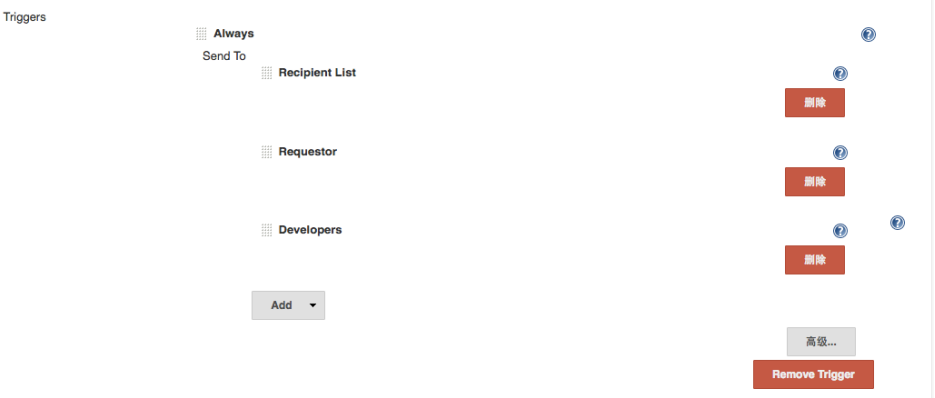


以上是 Jenkins 应用系统参数设置，下面是为需要用邮件推送结果的 Jenkins 项目进行相关配置。

在 Jenkins 项目中添加构建后操作步骤。选择 Editable Email Notification 类型，填写邮件接收人列表。



添加邮件发送触发器。



服务发现和负载均衡

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 变成 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。更多详细信息，参见 自定义路由使用说明，自定义路由使用示例。

适用场景

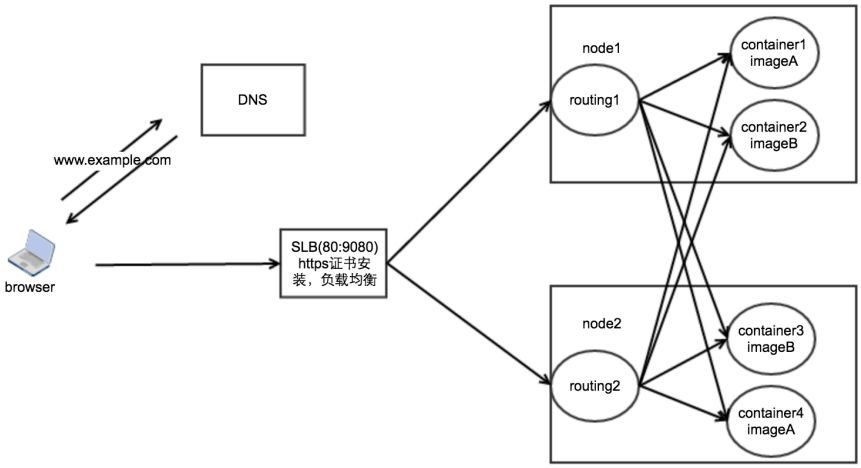
普通且简单的 7 层协议负载均衡，Web 路由服务，容器集群内服务之间 7 层协议互相访问的通信代理和负载均衡。

原理

如下图所示，当您新建一个集群的时候，会默认给这个集群分配一个负载均衡实例。该负载均衡实例会将集群中的所有节点加入作为后端，同时前端会暴露 80 端口，后端所有节点的机器会暴露 9080 端口。容器服务会启动一个路由应用 acsrouting，即阿里云容器服务路由应用（Alibaba Cloud Container Service Routing）。该路由应用只有一个服务，即路由服务。该服务是全局（global）的，即每个节点（下面说到的主机和节点都

是同一个意思，即 ECS 的 vm 实例）都部署了该服务（或者说镜像）的一个拷贝，也就是容器。每个节点都由这个容器用来路由 HTTP 服务或者 HTTPS 服务。

如图所示，HTTP 服务，负载均衡实例的前后端端口的映射为 80:9080，主机与路由容器之间的端口映射为 9080:80，即路由的容器暴露 80 端口，其它用作 Web 服务的容器可以暴露任意的端口。只要在容器启动的时候设置主机和容器端口的映射，routing 服务就能获取到相应的端口进行请求的路由。有关暴露 HTTP 服务的完整示例，参见 通过镜像创建 Nginx。



设置方式

通过容器服务管理控制台进行设置

通过 服务 > 变更配置 进行设置

登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

选择所要暴露的服务所在的集群。

选择所要暴露的服务（本示例中为 spring-boot）并单击 **变更配置**。

在变更配置页面，配置主机和容器端口的映射，如下图所示。

主机端口为空，表示随机暴露一个主机的端口（暴露 HTTP/HTTPS 服务时，您可以不需要知道主机暴露的具体端口是什么，可以使用 overlay 网络或者 VPC 网络来直接访问容器的端口），容器端口为 8080。您使用 spring-boot 默认暴露 Web 服务的 8080 端口来提供HTTP 服务，使用的协议是 TCP 协议。

路由配置通过域名来暴露服务，须标明要暴露的端口，此处为 Web 服务 8080 端口。域名字段只填写了域名前缀，如果域名前缀为 XXX，获得的域名为 XXX.$cluster_id.$region_id.alicontainer.com 供测试使用。此处您获得的域名为 spring-boot.c0cffb4340aee47ccb26dea062cfb0b2e.cn-beijing.alicontainer.com。您也可以填写自己的域名，需要添加解析到相应的负载均衡实例 IP。关于配置路由的容器端口和 HTTP 服务的域名的详细信息，参见 routing 标签。

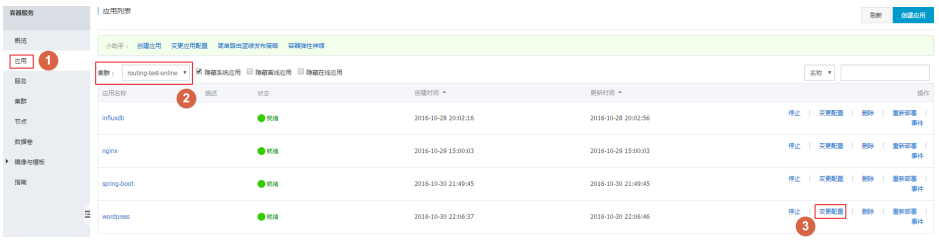
通过应用的模板编辑器进行设置

登录 容器服务管理控制台。

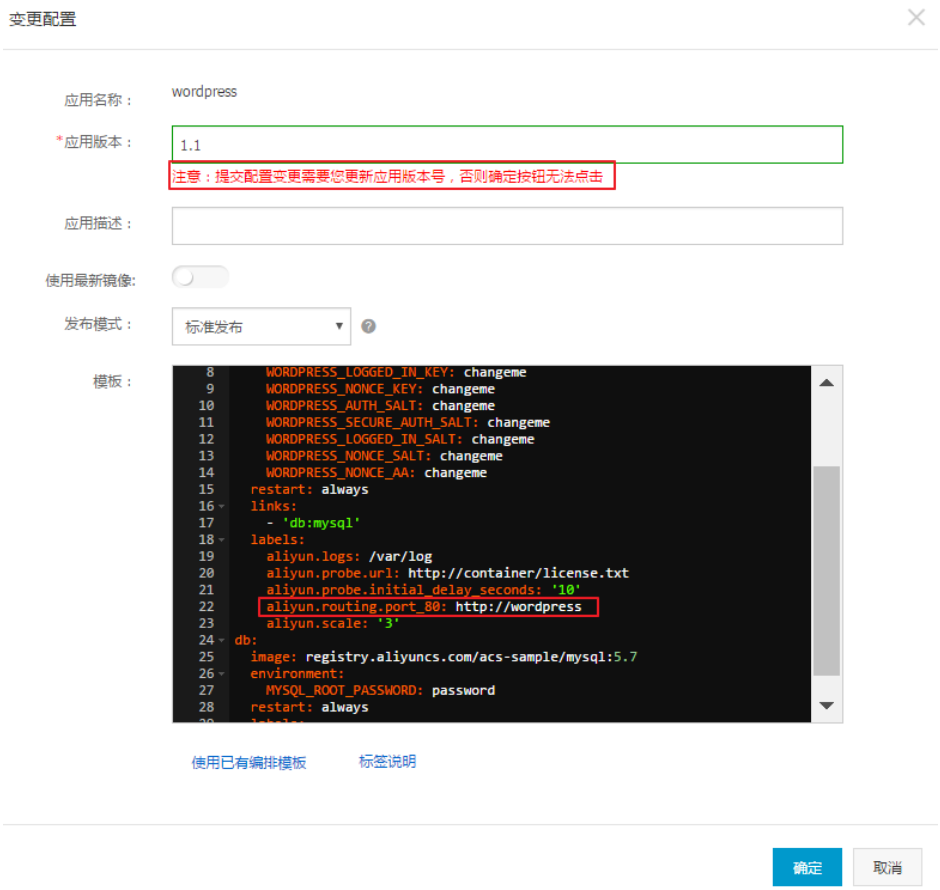
单击左侧导航栏中的 **应用**。

选择目标应用所在的集群。

选择目标应用（本示例中为 wordpress）并单击 **变更配置**。



在模板编辑器中，添加 routing 标签，定义相应的域名或者域名前缀。注意升级应用的版本，以及确定是否拉取最新的 Docker 镜像，最后单击 **确定** 更新域名，如下图所示。



通过客户端工具进行设置

- docker help run：查看使用的 “-p” 选项，路由配置在容器服务管理控制台进行。
- docker-compose：查看支持的 “ports” 选项，路由配置规则详情见 routing 标签。

登录 容器服务管理控制台。

单击左侧导航栏中的 **服务**。

选择要添加域名的服务所在的集群。

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-rds）并单击 **变更配置**。如下图所示。

单击 **Web 路由规则** 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.example.com ）并单击 **确定** 更新配置。如下图所示。

注意：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.example.com 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。

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。

i. 单击 **容器服务管理控制台** 左侧导航栏中的 **集群**。

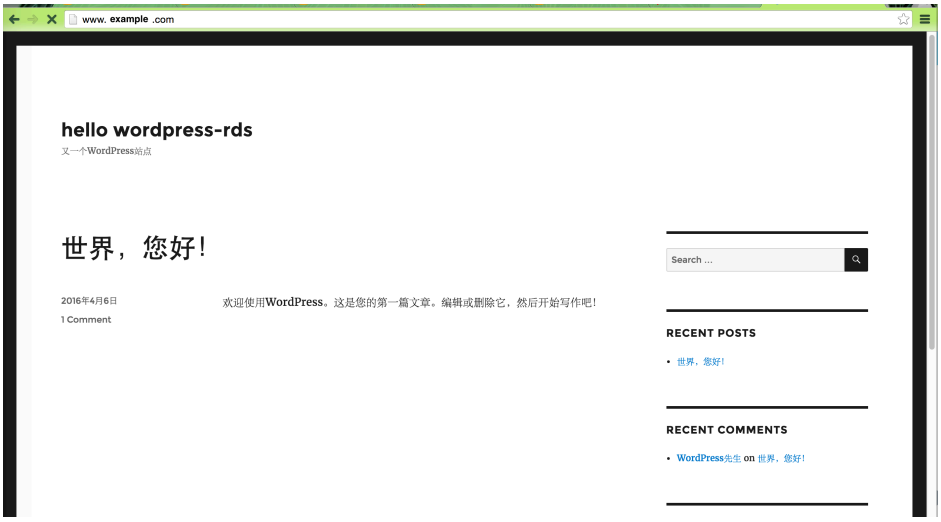
ii. 选择相应的集群，本示例为 routing-test-online 并单击 **管理**。

iii. 单击 **负载均衡** 并单击 **前往SLB控制台**。

您可以查看负载均衡实例的 **服务地址**。

以万网的域名解析为例，添加 A 记录，绑定域名 www.example.com 的地址到负载均衡实例的服务地址。

访问页面 www.example.com。



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 给暴露公网的服务添加域名。

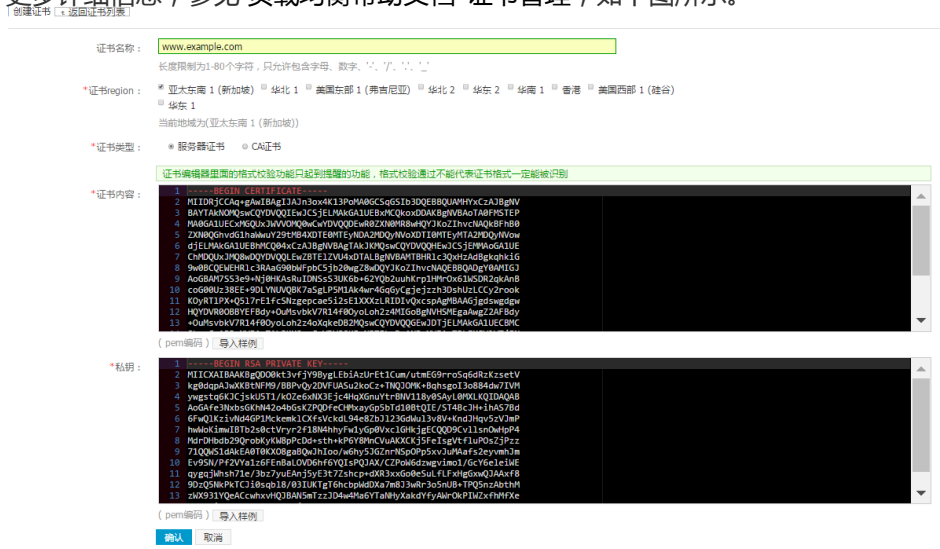
操作流程

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。

i. 登录 负载均衡管理控制台。
ii. 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**。

iii. 输入证书的相关信息。

更多详细信息，参见 负载均衡帮助文档-证书管理，如下图所示。



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。

i. 单击 **容器服务管理控制台** 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**。

ii. 单击 **负载均衡** 并单击 **前往SLB控制台**。

您可以查看负载均衡实例的服务地址。

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

i. 前端协议，选择 HTTPS。
ii. 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）。
iii. 选择前面步骤创建的证书 www.example.com。
iv. 根据需要设置其它选项。
v. 单击 **下一步**。

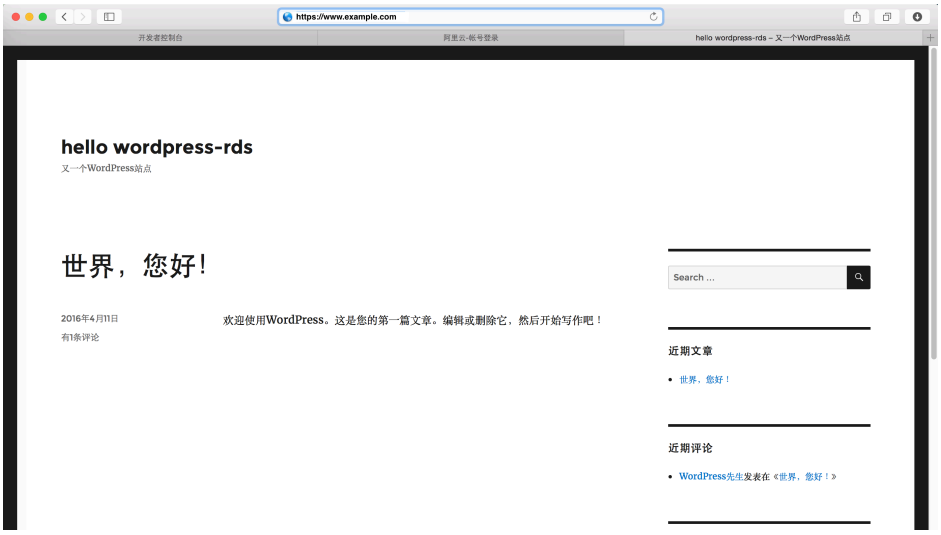
完成 **健康检查配置** 标签页中的配置， 如下所示。单击 **确认**。

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

配置成功后，单击 **确认**。

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 配置 HTTP 直接跳转到 HTTPS 进行设置。

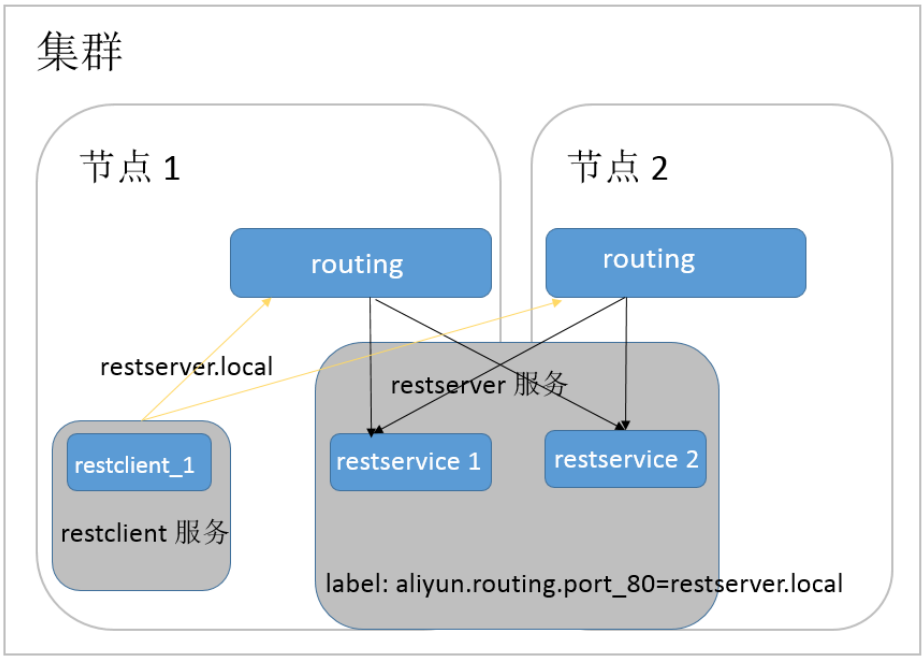
在容器服务上可以通过 acsrouting 将基于域名的 HTTP 服务暴露出去，而且能够配合健康检查自动的负载均衡和服务发现，当其中一个容器出现问题之后，routing 会自动将健康检查失败的容器从后端摘除，所以能做到自动的服务发现。

然而这个例子是将服务暴露到外网，那么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？容器服务引入了负载均衡功能，您只需要使用以 .local 结尾的域名，并在依赖的服务的 external_links 中增加这个域名， 依赖的服务便可以通过 .local 的域名访问到依赖的服务，并且能够配合健康检查做到自动的服务发现。

实现原理

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 的服务的容器中 restserver.local 的域名解析到的是 routing 服务的地址，这样可以将 HTTP 请求转发到 routing 的容器，并带上 HOST 为 restserver.local 的请求头。

routing 会对配置了 aliyun.routing_port_xxx: restserver.local 的服务监听其容器的健康状态并挂载到 HAProxy 的后端，HAProxy 接收到带有 restserver.local HOST 头的 HTTP 请求就能转发到对应了容器了。



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，对于微服务的场景是不够用的。
- 而相对于其他的微服务服务发现的解决方案，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。

编排实例

```
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_links，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 请求被路由到不同的 rest 服务的容器上了，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=053cb232fdfbcb5405ff791650a0746ab77f26cce74fea2320075c2af55c975f; 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=b8c36abca525ac7fb02d2a9fcaba8d36641447a774ea956cd93068419f17ee3f; path=/
```

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 label aliyun.lb.port_$container_port 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容器的服务。

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 label aliyun.lb.port_$container_port 来实现）。

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容器的服务。

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。

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。

注意： 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。

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。容器服务会通过该名称来引用这个负载均衡实例。

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

创建监听端口。

单击实例右侧的 **管理** > 单击右侧导航栏中的 **监听** > 单击 **添加监听** > 设置监听配置。创建协议为 TCP，端口映射为 6379:6379。如下图所示。

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

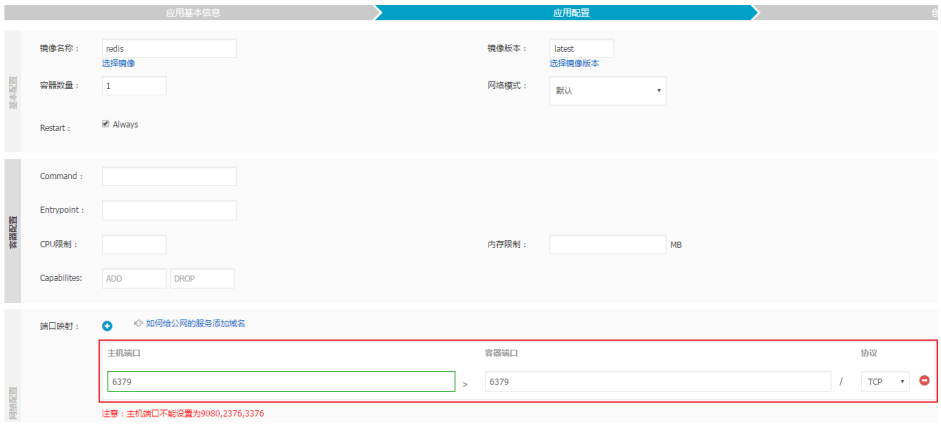
有关如何创建应用，参见 创建应用。

注意：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。

选择 Redis 镜像并设置 **端口映射**。

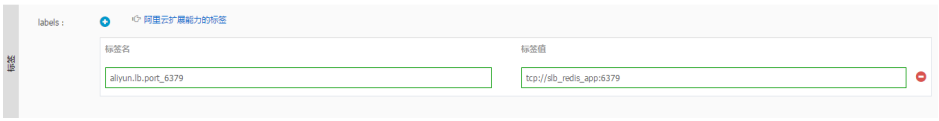
注意：此处 Redis 镜像只是开通了容器的 6379 端口，为了使创建的负载均衡路由到这个容器端口，您必须知道 Redis 镜像的主机:端口映射。

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.



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 **自定义 SLB**。

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。

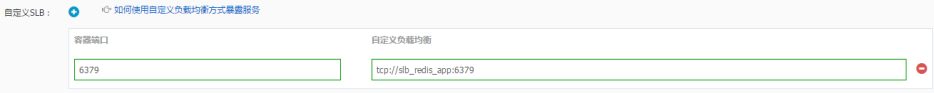
```
aliyun.lb.port_$container_port:$scheme://$[slb_name|slb_id]:$front_port
```

- $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 aliyun.lb.port_$container_port。

在 **创建应用** 页面，单击 **自定义 SLB** 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 aliyun.lb.port_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:容器端口映射设置的 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。

注意：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。

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。

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。

单击左侧导航栏中的 **实例管理** > 选择实例所在的地域 > 选择所需实例 > 单击实例右侧的 **管理** > 单击左侧导航栏中的 **服务器** > **后端服务器**。

由健康状态可见，负载均衡已经正确地绑定到了 Redis 的后端。

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。

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。

注意：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)
```

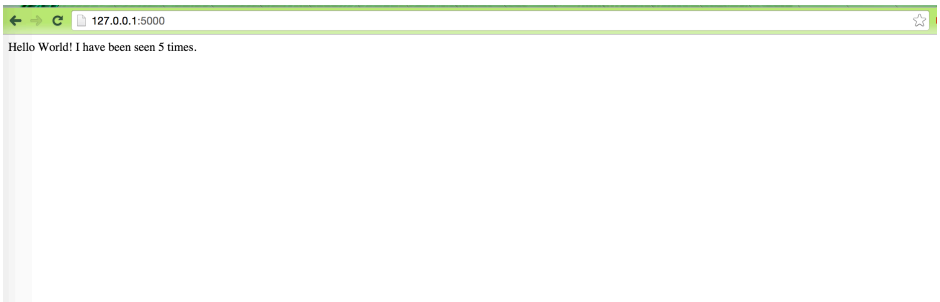
requirements.txt

```
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。

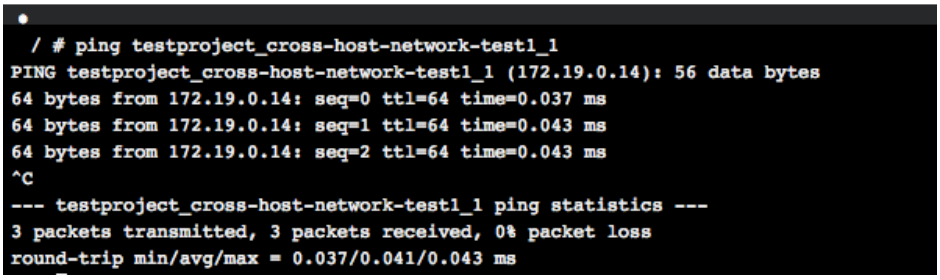


容器服务为集群内的服务和容器提供多种服务发现方式，可以通过容器名，link，hostname 等进行发现。

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 **容器网络互连** 中的例子，您可以在 cross-host-network-test2 的容器中通过 cross-host-network-test1 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}，所以在连接管理终端后，您可以通过另外一个服务的容器名进行访问。



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，容器内部就可以通过 MySQL 的域名访问到 db 服务的容器。

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，并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。

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'。

注意：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 lb 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

注意：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 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, 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。
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/
*/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	

注意：

- 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。

注意： 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 80 --expose 22 -e TCP_PORTS=22 your_app
docker --name app-2 --expose 80 --expose 22 -e TCP_PORTS=22 your_app
docker run --link app-1:app-2 --link app-2:app-2 -p 80:80 -p 22:22 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。

注意：

- 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

注意：

- 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_SSL_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_SSL_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
```

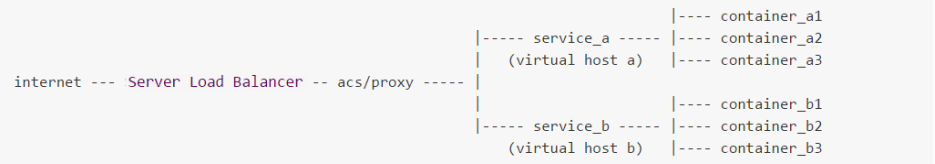
我希望将我的日志发送到 papertrailapp

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

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

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。

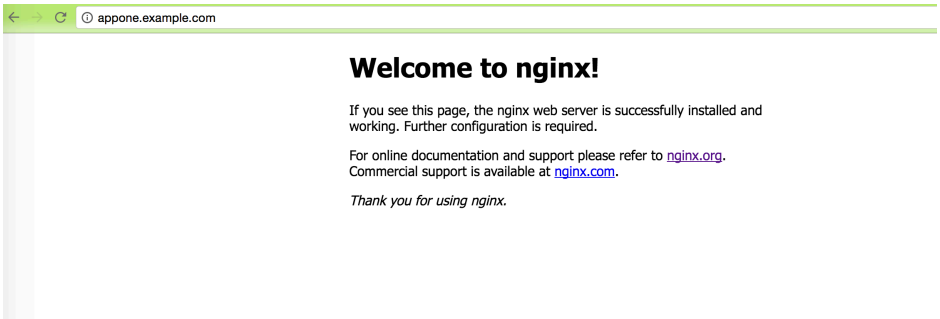
注意：任何两个不同的服务均不能共享使用同一个负载均衡，否则会导致负载均衡后端机器被删除，服务不可用。

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>请登陆容器服务控制台，在左侧边栏选择"服务"，在"服务列表"选择相应的"集群"，点击暴露到公网的"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
```

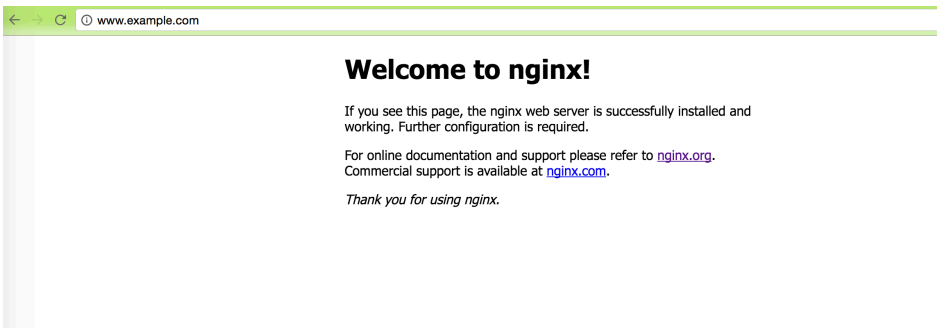
输入负载均衡的 VIP 地址之后，显示的 503 页面如下图所示。

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。



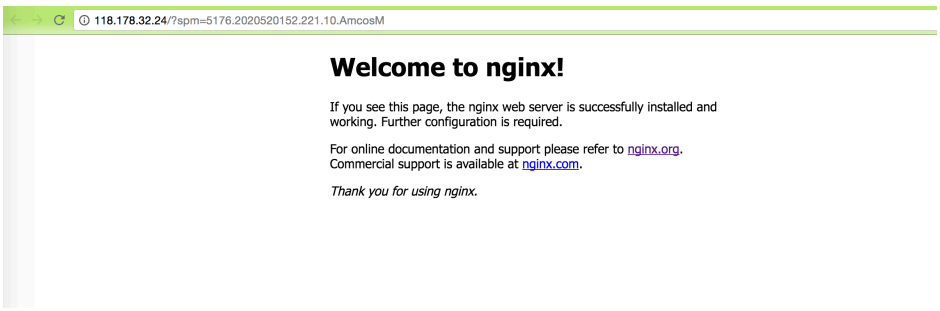
配置默认的后端

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。



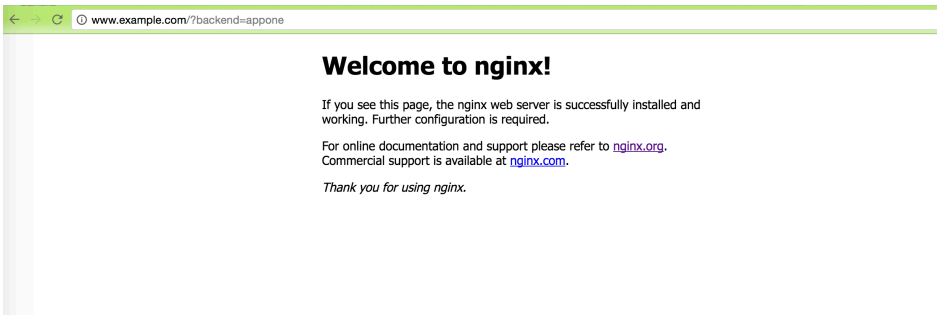
根据 URL 参数值选择后端

您也可以根据 URL 参数值的不同来代理不同的后端。

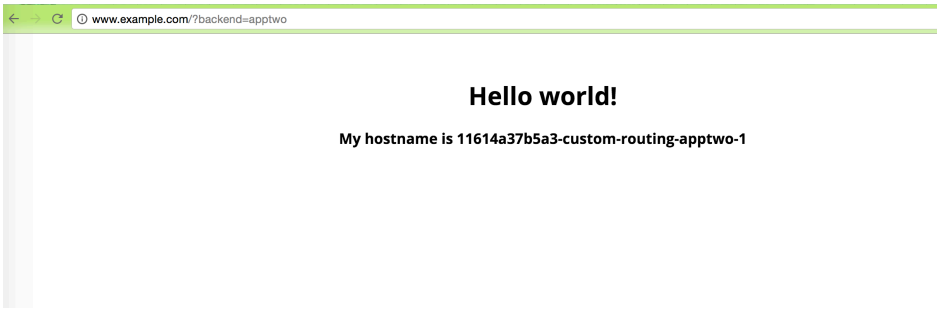
以下示例将通过 http://www.example.com?backend=appone 访问服务 appone，即 nginx 首页，通过 http://www.example.com?backend=apptwo 访问服务 apptwo，即 hello world 首页。应用模板代码如下所示。

```
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。

注意：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。

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。

注意：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
```

注意：

- 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 自定义路由镜像）：

```
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 **自定义路由镜像** 做负载均衡，做到了一定程度的高可用。

本示例使用的镜像是 acs/proxy 镜像。

注意：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/JF
mxloFrPtwEe2FGkLBfCDXXDrWgxyFhbuPQY\nBLNueUu94sffIxg+4u5Mriui7ftindOAf0d21PSM9gb/ZUypxIgAd3RHCe
/gtT0h\nVCn6FikXynXLDTODYWCthQHBwSZS88HNU+B0T9Yl65JiQ0mV+YF+h3D/c232E6Gp\nzK+8ehVB13s5hecUx
3dvdUQPBUhJYvzsPjChgsXSMDRexiN66kbhH6dJArsrYb8t\nEBWXfCZaTcF82wkAsUe/fhlGhh97h+66lh6OQQIDAQAB
AoIBAQC4d8ifNWRI9vIB\nbbAZRne7xMm5MCU2GI8q97Rgm+nAPl5bHinMVsaBnKgaj76EH+TQ+re1xyiSKwCH\nQ7
FidsQqYGwQjy9NncJATpAjQ4EPeLWQU2D9Ly+NjnhEKr/u0Ro6LhdA+hqt59dS\nXHvfEP/It5odN62yJzikDWBmk/hhK
0tu28dPYUuPoWswXWFMkaNttmfLgZlagiqr\nYp7rxAFqQurzctQ2VNwezekDHQoh8ounHGEniZ+fA6sFtYi83KTKWkv
Fom1chZQr\nxxPbbgANJJJlNgtkl6JZNxj6SYimmWvzmrrU25khKg/klP5EtQzIx6UFhURnuTKu\nzNgqcIABAoGBAOqU
OerveEUePvsAlta8CV/p2KKwenv+kUofQ4UpKFXfnHbQHQfr\nZHS29OQiPxqjVXYLu8gNfLRfKtUNyqV+TDrzJ1elW2R
Kc00GHAwPbXxijPhmJ2fW\neskn8tlDcyXpvoqWJG34896vo4IbcL0H/eUs0jJo6OJlCQBKXik+t3gxAoGBAM9k\nVOTV2
caKyrZ4ta0Q1LKqKfOkt0j+vKz167J5pSLjVKQSUxGMyLnGwiQdDtB4iy6L\nFcCB/S0HM0UWkJWhNYAL8kHry53bVdHt
QG0tuYFYvBJo7A+Nppsn9MtlVh8KbVu4\nhOz/3MWwbQNnvIVCGK/fSltS1GhTk4rKL7PjNwMRAoGBALK0n3bqXj6Rr
zs7FK6c\na6vlE4PFXFpv8jF8pcyhMThSdPlSzHsHCe2cn+3YZSie+/FFORZLqBAlXBUZP6Na\nFyrlqLgtofVCfppUKDPL4
QXccjaeZDDIBZyPUYPQzb05WE5t2WzqNqcUOUVaMEXh\n+7uGrM94espWXEgbX6aeP9lRAoGARlJQ7t8MXuQE5GZ
9w9cnKAXG/9RkSZ4Gv+cL\nKpNQyUmoE5IbFKJWFZgtkC1CLrIRD5EdqQ7ql/APFGgYUoQ9LdPfKzcW7cnHic0W\nw
W51rkQ2UU++a2+uhIHB4Y3U6+WPO0CP4gTICUhPTo5IQC8vS8M85UZqu41LRA5W\nqnpq1uECgYEAq+6KpHhlR+
5h3Y/m0n84yJ0YuCmrl7HFRzBMdOcaW3oaYL83rAaq\n6dJqpAVgeu3HP8AtiGVZRe78J+n4d2JGYSqgtP2lFFTdF9Hfh
cR2P9bUBNYtWols\nEs3iw53t8a4BndLGBwLPA3lklf7J5stYanRv6NqaRaLq4FQMxsW1A0Q=\n-----END RSA PRIVATE
KEY-----\n-----BEGIN CERTIFICATE-----
\nMIIDvDCCAqSgAwIBAgIBATANBgkqhkiG9w0BAQUFADBgMQswCQYDVQQGEwJDTjER\nMA8GA1UECBMIWmhla
mlhbmcxETAPBgNVBAcTCEhhbmd6aG91MRQwEgYDVQQKEwth\nbGliYWJhLmNvbTEVMBMGA1UECxMMd3d3LnJv
b3QuY29tMB4XDTE1MDIwOTA1MzQx\nOFoXDTE2MDIwOTA1MzQxOFowZjELMAkGA1UEBhMCQ04xETAPBgNVBA
gTCFpoZWpp\nYW5nMRQwEgYDVQQKEwthbGliYWJhLmNvbTEVMBMGA1UECxMMd3d3LnJvb3QuY29t\nMRcwFQY
DVQQDEw53d3cubGluaHVhLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP\nADCCAQoCggEBAL4JyoXqYVhylmAxI
gYko87Le50T9AycWeXCDscl5p3HdpLJa5FC\nAeak/zEoiRgS9y6+Gnb7OpCPyRZsZaBaz7cBHthRpCwXwg11w61oMch
YW7j0GASz\nbnlLveLH3yMYPuLuTK4rou37Yp3TgH9HdtT0jPYG/2VMqcSIAHd0Rwnv4LU9IVQp\n+hYpF8p1yw0zg2F
grYUBwcEmUvPBzVPgdE/WJeuSYkNJlfmBfodw/3Nt9hOhqcyv\nvHoVQdd7OYXnFMd3b3VEDwVISWL87D4woYLF0jA
0XsYjeupG4R+nSQK7K2G/LRAV\nl3wmWk3BfNsJALFHv34ZRoYfe4fuupYejkECAwEAAaN7MHkwCQYDVR0TBAIwAD
As\nBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYD\nVR0OBBYEFM6ESmkD
KrqnqMwBawkjeONKrRMQMB8GA1UdIwQYMBaAFFUrhN9ro+Nm\nrZnl4WQzDpgTbCBhMA0GCSqGSIb3DQEBBQU
AA4IBAQCQ2D9CRiv8brx3fnr/RZG6\nFYPEdxjY/CyfJrAbij0PdKjzZKk1O67chM1Oxs2JhJ6tMqg2sv50bGx4XmbSPmEe\
nYTJjIXMY+jCoJ/Zmk3Xgu4K1y1LvD25PahDVhRrPN8H4WjsYu51pQNshil5E/3iQ\n2JoV0r8QiAsPiiY5+mNCD1fm+Q
N1tyUabczi/DHafgWJxf2B3M66e3oUdtbzA2pf\nYHR8RVeSFrjaBqudO8ir+uYcRbRkroYmY5Vm+4Yp64oetrPpKUPWS
YaAZ0uRtpeL\nB5DpqXz9GEBb5m2Q4dKjs5Hm6vyFUORCzZcO4XexDhcgdLOH5qznmh9oMCk9QvZf\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/JF
mxloFrPtwEe2FGkLBfCDXXDrWgxyFhbuPQY\nBLNueUu94sffIxg+4u5Mriui7ftindOAf0d21PSM9gb/ZUypxIgAd3RHCe
/gtT0h\nVCn6FikXynXLDTODYWCthQHBwSZS88HNU+B0T9Yl65JiQ0mV+YF+h3D/c232E6Gp\nzK+8ehVB13s5hecUx
3dvdUQPBUhJYvzsPjChgsXSMDRexiN66kbhH6dJArsrYb8t\nEBWXfCZaTcF82wkAsUe/fhlGhh97h+66lh6OQQIDAQAB
AoIBAQC4d8ifNWRI9vIB\nbbAZRne7xMm5MCU2GI8q97Rgm+nAPl5bHinMVsaBnKgaj76EH+TQ+re1xyiSKwCH\nQ7
FidsQqYGwQjy9NncJATpAjQ4EPeLWQU2D9Ly+NjnhEKr/u0Ro6LhdA+hqt59dS\nXHvfEP/It5odN62yJzikDWBmk/hhK
```

```
0tu28dPYUuPoWswXWFMkaNttmfLgZlagiqr\nYp7rxAFqQurzctQ2VNwezekDHQoh8ounHGEniZ+fA6sFtYi83KTKWkv
Fom1chZQr\nxxPbbgANJJJlNgtkl6JZNxj6SYimmWvzmrrU25khKg/klP5EtQzIx6UFhURnuTKu\nzNgqcIABAoGBAOqU
OerveEUePvsAlta8CV/p2KKwenv+kUofQ4UpKFXfnHbQHQfr\nZHS29OQiPxqjVXYLu8gNfLRfKtUNyqV+TDrzJ1elW2R
Kc00GHAwPbXxijPhmJ2fW\neskn8tlDcyXpvoqWJG34896vo4IbcL0H/eUs0jJo6OJlCQBKXik+t3gxAoGBAM9k\nVOTV2
caKyrZ4ta0Q1LKqKfOkt0j+vKz167J5pSLjVKQSUxGMyLnGwiQdDtB4iy6L\nFcCB/S0HM0UWkJWhNYAL8kHry53bVdHt
QG0tuYFYvBJo7A+Nppsn9MtlVh8KbVu4\nhOz/3MWwbQNnvIVCGK/fSltS1GhTk4rKL7PjNwMRAoGBALK0n3bqXj6Rr
zs7FK6c\na6vlE4PFXFpv8jF8pcyhMThSdPlSzHsHCe2cn+3YZSie+/FFORZLqBAlXBUZP6Na\nFyrlqLgtofVCfppUKDPL4
QXccjaeZDDIBZyPUYPQzb05WE5t2WzqNqcUOUVaMEXh\n+7uGrM94espWXEgbX6aeP9lRAoGARlJQ7t8MXuQE5GZ
9w9cnKAXG/9RkSZ4Gv+cL\nKpNQyUmoE5IbFKJWFZgtkC1CLrIRD5EdqQ7ql/APFGgYUoQ9LdPfKzcW7cnHic0W\nw
W51rkQ2UU++a2+uhIHB4Y3U6+WPO0CP4gTICUhPTo5IQC8vS8M85UZqu41LRA5W\nqnpq1uECgYEAq+6KpHhlR+
5h3Y/m0n84yJ0YuCmrl7HFRzBMdOcaW3oaYL83rAaq\n6dJqpAVgeu3HP8AtiGVZRe78J+n4d2JGYSqgtP2lFFTdF9Hfh
cR2P9bUBNYtWols\nEs3iw53t8a4BndLGBwLPA3lklf7J5stYanRv6NqaRaLq4FQMxsW1A0Q=\n-----END RSA PRIVATE
KEY-----\n-----BEGIN CERTIFICATE-----
\nMIIDvDCCAqSgAwIBAgIBATANBgkqhkiG9w0BAQUFADBgMQswCQYDVQQGEwJDTjER\nMA8GA1UECBMIWmhla
mlhbmcxETAPBgNVBAcTCEhhbmd6aG91MRQwEgYDVQQKEwth\nbGliYWJhLmNvbTEVMBMGA1UECxMMd3d3LnJv
b3QuY29tMB4XDTE1MDIwOTA1MzQx\nOFoXDTE2MDIwOTA1MzQxOFowZjELMAkGA1UEBhMCQ04xETAPBgNVBA
gTCFpoZWpp\nYW5nMRQwEgYDVQQKEwthbGliYWJhLmNvbTEVMBMGA1UECxMMd3d3LnJvb3QuY29t\nMRcwFQY
DVQQDEw53d3cubGluaHVhLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP\nADCCAQoCggEBAL4JyoXqYVhylmAxI
gYko87Le50T9AycWeXCDscl5p3HdpLJa5FC\nAeak/zEoiRgS9y6+Gnb7OpCPyRZsZaBaz7cBHthRpCwXwg11w61oMch
YW7j0GASz\nbnlLveLH3yMYPuLuTK4rou37Yp3TgH9HdtT0jPYG/2VMqcSIAHd0Rwnv4LU9IVQp\n+hYpF8p1yw0zg2F
grYUBwcEmUvPBzVPgdE/WJeuSYkNJlfmBfodw/3Nt9hOhqcyv\nvHoVQdd7OYXnFMd3b3VEDwVISWL87D4woYLF0jA
0XsYjeupG4R+nSQK7K2G/LRAV\nl3wmWk3BfNsJALFHv34ZRoYfe4fuupYejkECAwEAAaN7MHkwCQYDVR0TBAIwAD
As\nBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYD\nVR0OBBYEFM6ESmkD
KrqnqMwBawkjeONKrRMQMB8GA1UdIwQYMBaAFFUrhN9ro+Nm\nrZnl4WQzDpgTbCBhMA0GCSqGSIb3DQEBBQU
AA4IBAQCQ2D9CRiv8brx3fnr/RZG6\nFYPEdxjY/CyfJrAbij0PdKjzZKk1O67chM1Oxs2JhJ6tMqg2sv50bGx4XmbSPmEe\
nYTJjIXMY+jCoJ/Zmk3Xgu4K1y1LvD25PahDVhRrPN8H4WjsYu51pQNshil5E/3iQ\n2JoV0r8QiAsPiiY5+mNCD1fm+Q
N1tyUabczi/DHafgWJxf2B3M66e3oUdtbzA2pf\nYHR8RVeSFrjaBqudO8ir+uYcRbRkroYmY5Vm+4Yp64oetrPpKUPWS
YaAZ0uRtpeL\nB5DpqXz9GEBb5m2Q4dKjs5Hm6vyFUORCzZcO4XexDhcgdLOH5qznmh9oMCk9QvZf\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 文档示例。

发布策略

背景信息

蓝绿发布是一种**零宕机**的应用更新策略。进行蓝绿发布时，应用的旧版本服务与新版本服务会同时并存，同一个应用不同版本的服务之间共享路由，通过调节路由权重的方式，可以实现不同版本服务之间的流量切换。验证无误后，可以通过发布确认的方式将应用的旧版本的服务删除；如果验证不通过，则进行发布回滚，应用的新版本会进行删除。

前置条件

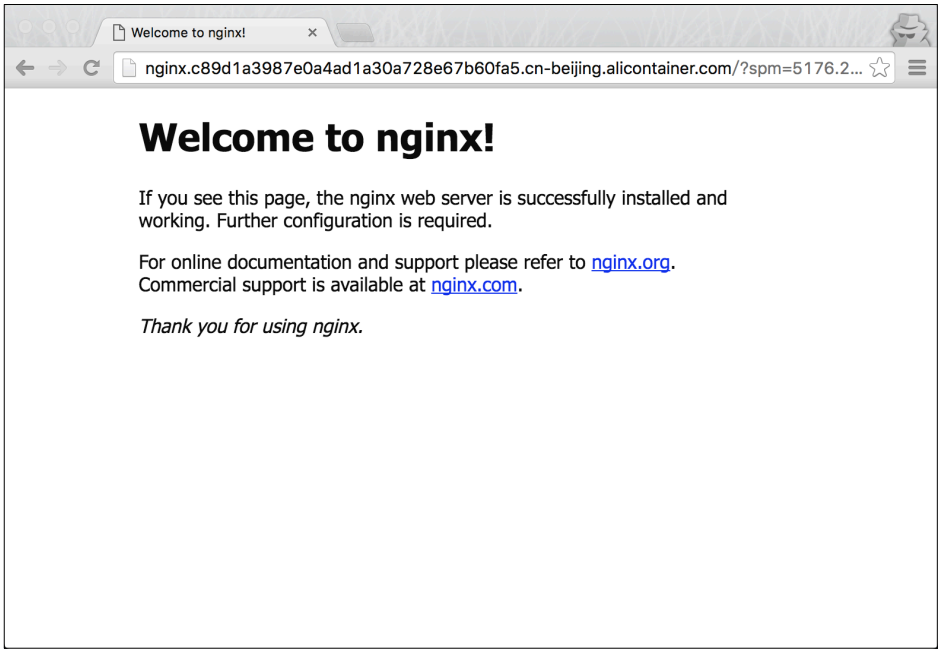
支持蓝绿发布需要将路由服务升级到最新的版本。详细信息参见 升级系统服务。

场景介绍

假设您要进行蓝绿发布的应用是一个 Nginx 的静态页面，初始的应用模板如下。

```
nginx-v1:
image: 'registry.aliyuncs.com/ringtail/nginx:1.0'
labels:
aliyun.routing.port_80: nginx
restart: always
```

部署后页面的效果如下。



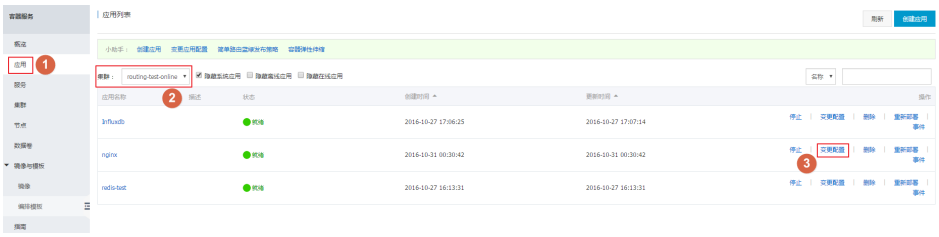
操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择目标应用所在的集群。

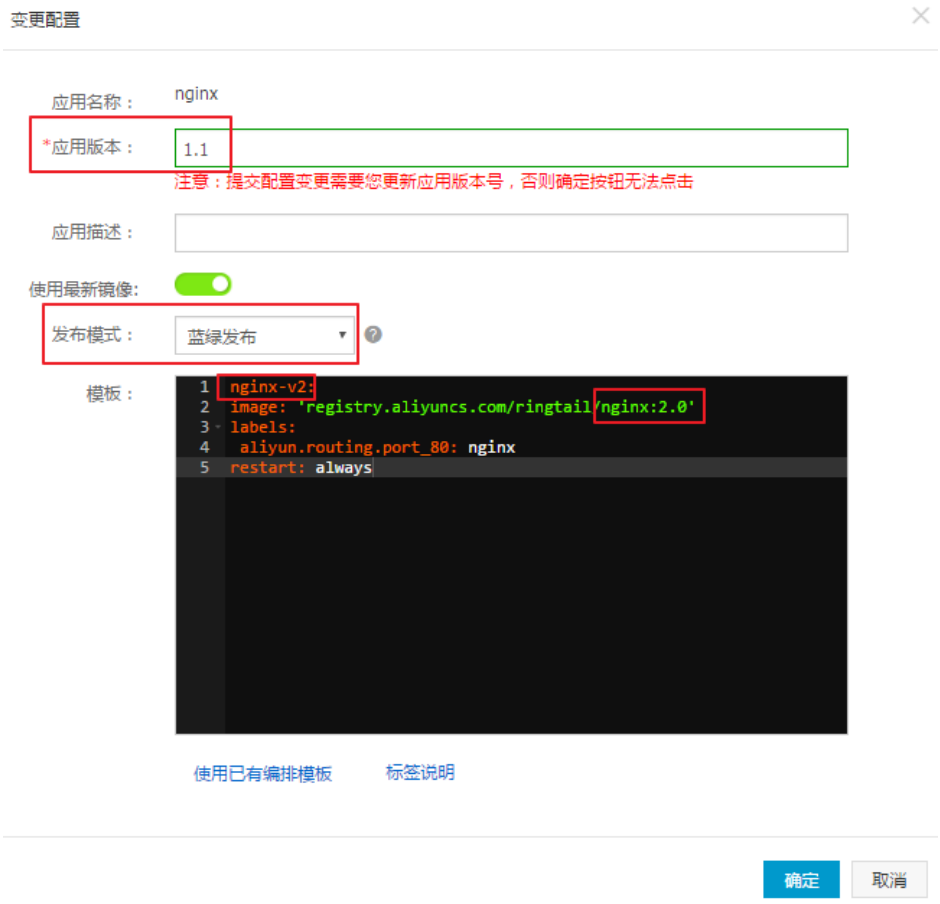
选择目标应用并单击 **变更配置**。



选择变更的发布模式与新版本服务的配置。

注意：

- 在蓝绿发布中，新版本与旧版本不能共用同一个名字。
- 在蓝绿发布的场景中，为了保证应用的零宕机切换，新版本的服务的路由权重默认为 0，需要通过路由管理页面进行调整，方可进行流量切换。



模板样例如下所示：

```
nginx-v2:
image: 'registry.aliyuncs.com/ringtail/nginx:2.0'
labels:
aliyun.routing.port_80: nginx
restart: always
```

单击 **确定**，发布变更。

在发布的过程中，会经历两个状态：

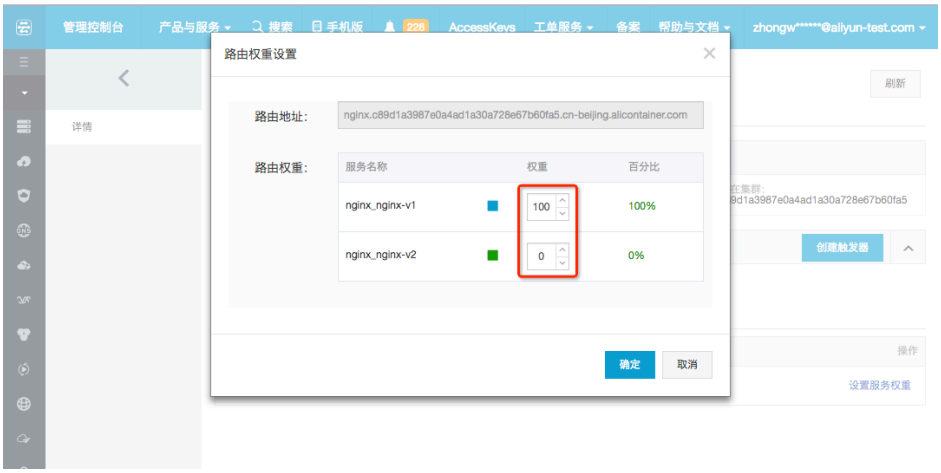
- 蓝绿发布中：表示新版本的服务尚未启动完成。

  蓝绿发布待确认：表示新版本的服务已经启动完成，此时需要进行发布确认或者发布回滚方可进行下一次发布。

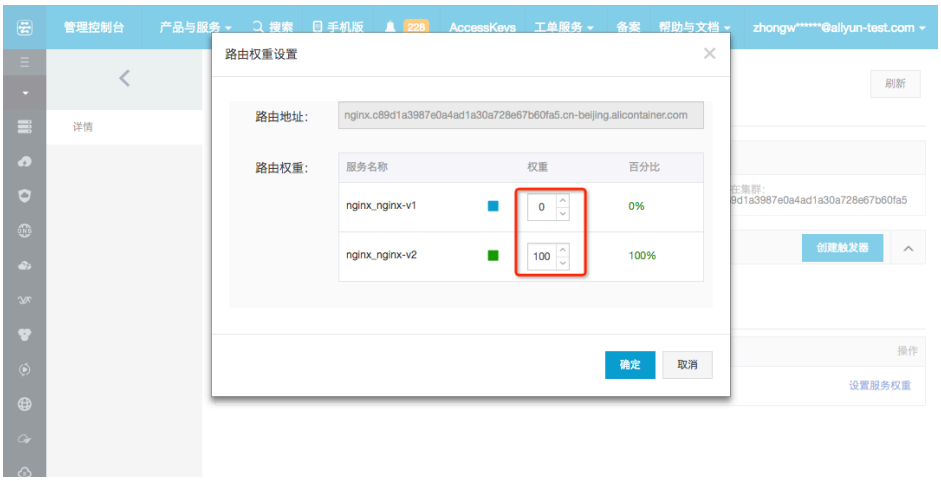
  进入应用的详情页面，可以看到新版本的应用和旧版本的应用并存。其中蓝色的表示旧版本的服务，绿色表示新版本的服务。如果一个服务在前后两个版本中都存在且没有变化，那么会出现黄色的标签，表示这个应用在蓝绿发布中不会出现任何变化。

单击 **路由列表** 并单击 **设置服务权重**。

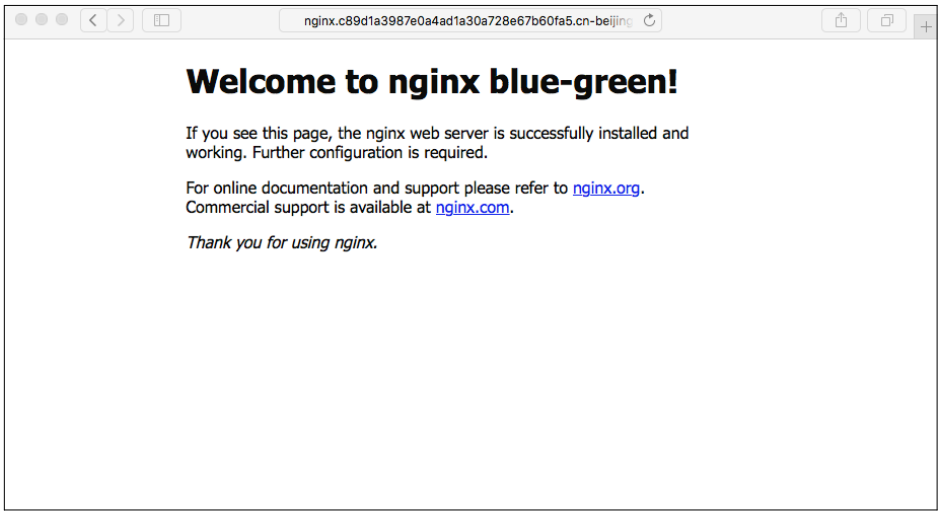
如下图所示，旧版本服务的权重为 100，新版本服务的权重为 0。



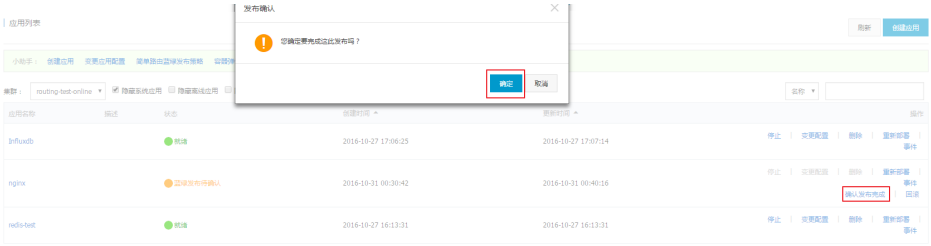
将旧版本服务的权重调整为 0，新版本服务的权重调整为 100。



由于默认路由服务是进行会话保持的，您可以打开一个新的浏览器窗口，访问新的版本，结果如下所示。



当整个发布流程验证完毕后，在应用列表页面，单击 **确认发布完成** 并在弹出的确认对话框中单击 **确认** 进行发布确认，方可进行下一次发布。



您可以看到应用的服务列表已经更新了，旧的服务已经完全下线删除了。

背景信息

蓝绿发布是一种**零宕机**的应用更新策略。进行蓝绿发布时，应用的旧版本服务与新版本服务会同时并存，同一个应用不同版本的服务之间共享负载均衡，通过调节负载均衡权重的方式，可以实现不同版本服务之间的流量

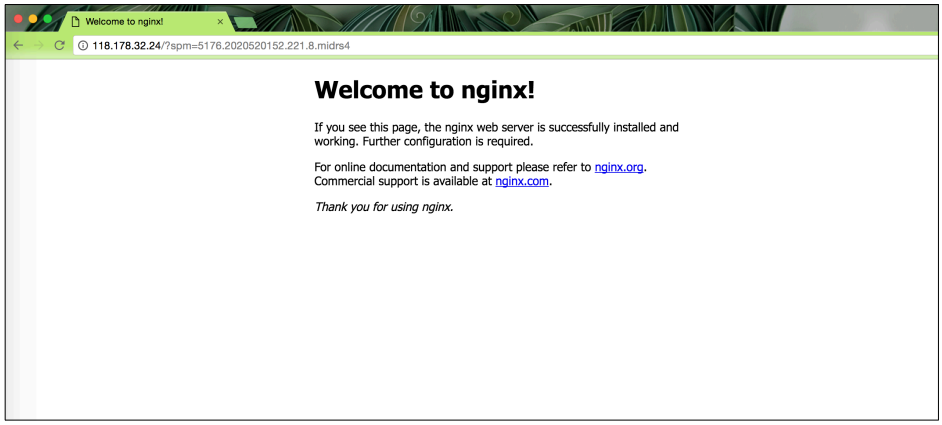
切换。验证无误后，可以通过发布确认的方式将应用的旧版本的服务删除；如果验证不通过，则进行发布回滚，应用的新版本会进行删除。

场景介绍

假设您要进行蓝绿发布的应用是一个 Nginx 的静态页面，初始的应用模板如下。

```
nginx-v1:
image: 'registry.aliyuncs.com/ringtail/nginx:1.0'
ports:
- 80:80/tcp
labels:
aliyun.lb.port_80: tcp://proxy_test:80
restart: always
```

部署后页面的效果如下。



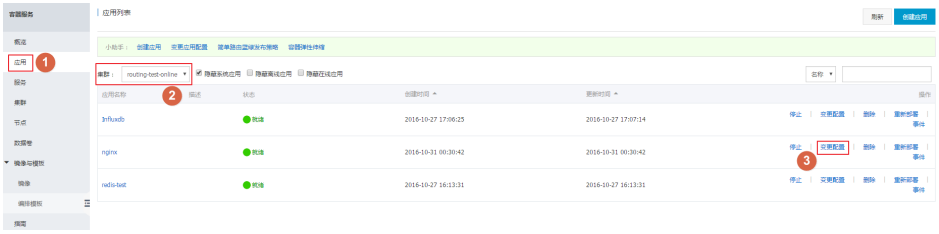
操作步骤

登录 容器服务管理控制台。

单击左侧导航栏中的 **应用**。

选择目标应用所在的集群。

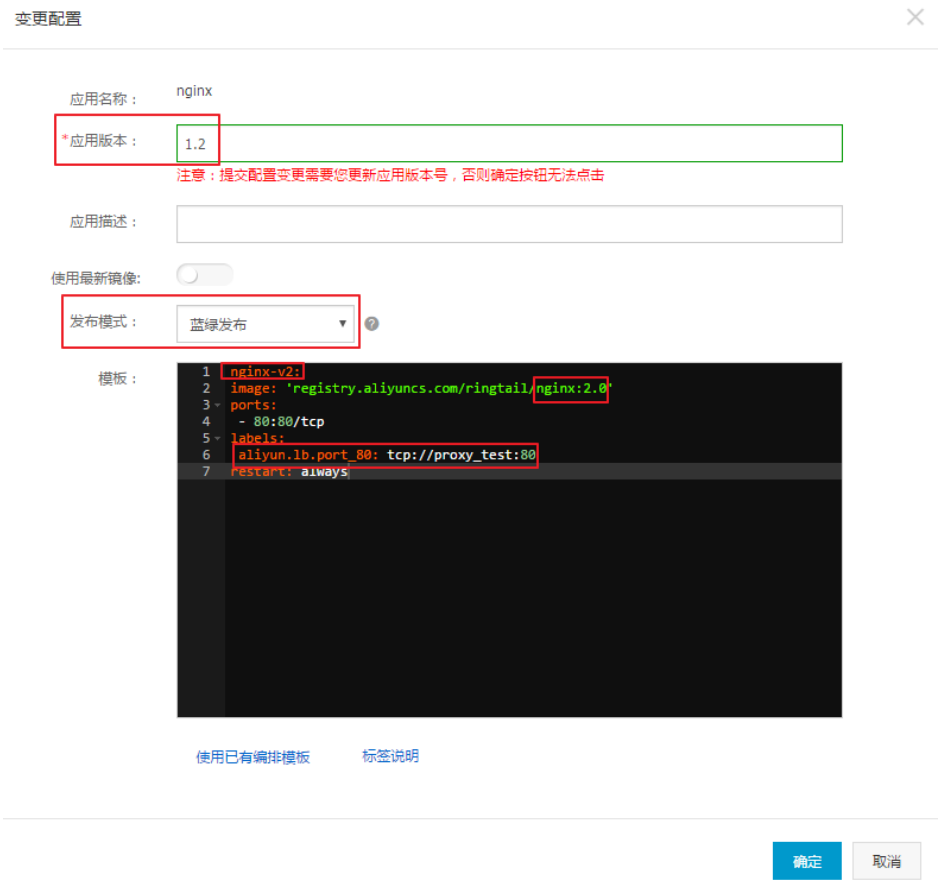
选择目标应用并单击 **变更配置**。



选择变更的发布模式与新版本服务的配置。

注意：

- 在蓝绿发布中，新版本与旧版本不能共用同一个名字。
- 在蓝绿发布的场景中，为了保证应用的零宕机切换，新版本的服务的路由权重默认为 0，需要通过路由管理页面进行调整，方可进行流量切换。



模板样例如下：

```
nginx-v2:
image: 'registry.aliyuncs.com/ringtail/nginx:2.0'
ports:
- 80:80/tcp
labels:
aliyun.lb.port_80: tcp://proxy_test:80
restart: always
```

单击**确定**，发布变更。

在发布的过程中，会经历两个状态：

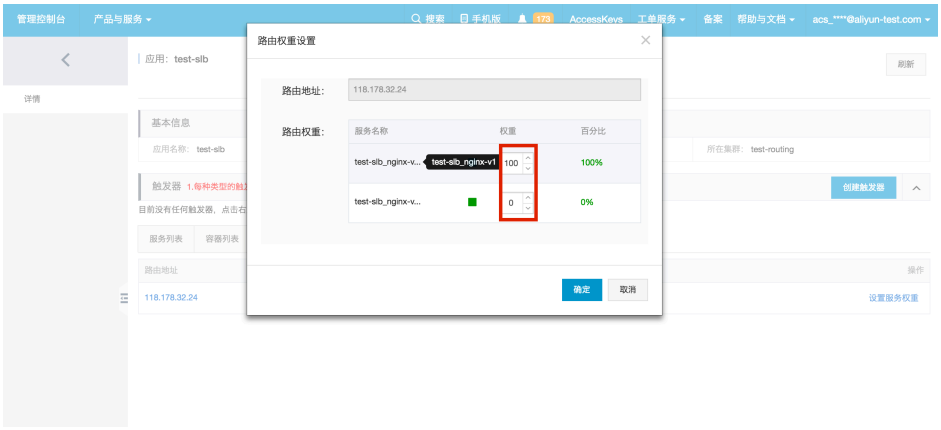
- 蓝绿发布中：表示新版本的服务尚未启动完成。

  蓝绿发布待确认：表示新版本的服务已经启动完成，此时需要进行发布确认或者发布回滚方可进行下一次发布。

  进入应用的详情页面，可以看到新版本的应用和旧版本的应用并存。其中蓝色的表示旧版本的服务，绿色表示新版本的服务。如果一个服务在前后两个版本中都存在且没有变化，那么会出现黄色的标签，表示这个应用在蓝绿发布中不会出现任何变化。

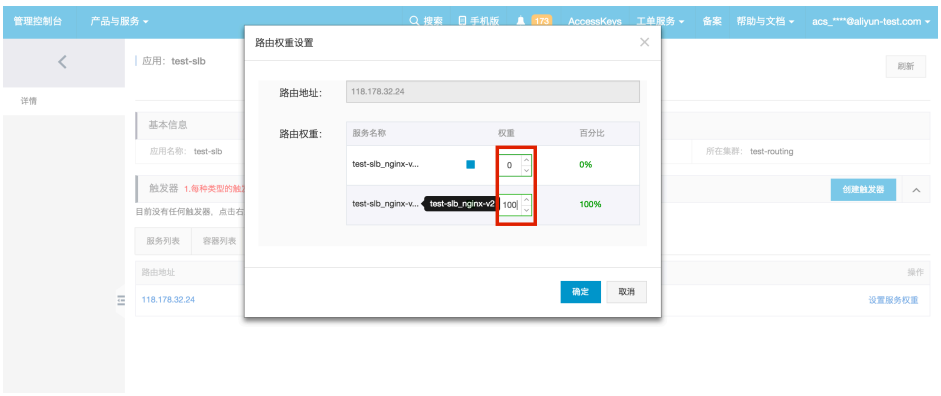
单击 **路由列表** 并单击**设置服务权重**。

如下图所示，旧版本服务的负载均衡权重为 100，新版本服务负载均衡的权重为 0。



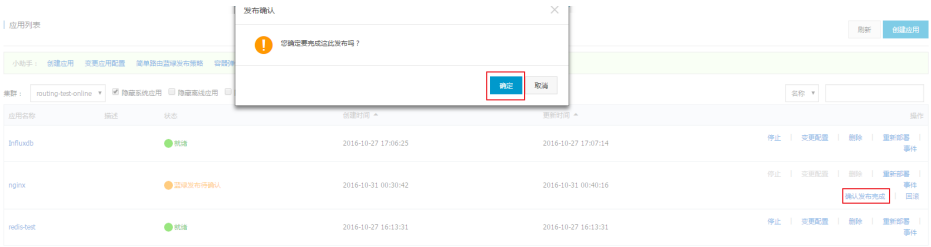
要做到**零宕机**升级，您需要先把新版本的服务的权重值调整为 100，此时新旧服务的权重各占 50%，测试新旧版本的服务都有稳定的流量。

注意：同时调整两个服务的权重，可能会导致部分请求失败，所以权重需要分为两步进行调整，一步只调整一个服务的权重。例如,先将新服务权重值从 0 调整为 100，待流量稳定后，再将然旧服务权重值从 100 设置为 0。下一步，将旧版本服务的权重调整为 0，新版本服务的权重调整为 100。



您可以打开一个新的浏览器窗口，访问新的版本，结果如下。

当整个发布流程验证完毕后，在应用列表页面，单击 **确认发布完成** 并在弹出的确认对话框中单击 **确认** 进行发布确认，方可进行下一次发布。



您可以看到应用的服务列表已经更新了，旧的服务已经完全下线删除了。


应用：nginx
刷新
基本信息
应用名称： nginx
创建时间： 2016-07-26
更新时间： 2016-07-26
所在集群： 北京测试发布
触发器 1.每种类型的触发器只能创建1个 2.关于资源伸缩类型的触发器
创建触发器
目前没有任何触发器，点击
服务列表
容器列表
路由列表
日志
事件
服务名称 所属应用 服务状态 容器状态 镜像 操作
nginx-v2 nginx 就绪 运行:1 停止:0 registry.aliyuncs.com/ringtail/nginx:2.0 停止 | 重新调度 | 变更配置 | 删除 | 事件