云容器实例(CCI) 2.0

快速入门

文档版本 01

发布日期 2025-07-15

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 使用 CCI 部署静态 Web 应用

本节通过在云容器实例上部署一个名称为2048的静态Web游戏应用为例，帮助您学习如何使用云容器实例。

您将按以下流程学习如何使用云容器实例。

操作流程

操作步骤	说明
准备工作	• 您需要注册华为账号。 • 如需使用ccictl工具方式部署静态Web应用可提前下载相关工具。
步骤一：构建镜像并上传至SWR镜像仓库	将应用构建镜像并上传镜像仓库，便于在云容器实例创建负载时，拉取上传的镜像。
步骤二：创建命名空间	您需要在CCI服务中创建一个命名空间，便于项目管理。
步骤三：创建容器组	配置基本信息和访问信息。
步骤四：访问容器组	通过公网IP去访问创建好的负载。
步骤五：清理资源	如果您在完成实践后不需要继续使用CCI，请及时清理资源以免产生额外扣费。

准备工作

- 在开始操作前，请您先注册华为账号，详情请参见**注册华为账号并开通华为云**。
- 您可以通过控制台或使用ccictl工具部署静态Web应用，如果使用ccictl工具，请下载并配置ccictl工具，具体操作步骤请参考**ccictl配置指南**。

步骤一：构建镜像并上传至 SWR 镜像仓库

要将已有的应用部署在云容器实例上运行，首先，需要将应用构建镜像并上传镜像仓库，再在云容器实例创建负载时，拉取上传的镜像。

安装容器引擎

上传镜像前，您需要安装容器引擎，如果您已经安装了容器引擎，请确保容器引擎为**1.11.2**及以上版本。

步骤1 参考**购买弹性云服务器**创建一台带有公网IP地址的Linux弹性云服务器。

作为演示，弹性云服务器和公网IP的规格不需要太高，例如弹性云服务器的规格为“1vCPUs | 2GiB”、公网IP带宽为“1 Mbit/s”即可，操作系统请选择“CentOS 8.2”。

说明

您也可以使用其他机器安装容器引擎，不创建弹性云服务器。

步骤2 返回弹性云服务器列表，单击“远程登录”登录购买的弹性云服务器。

步骤3 使用如下命令快速安装容器引擎。

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
sudo systemctl daemon-reload
sudo systemctl restart docker
```

----结束

构建镜像

此处以如何使用Dockerfile，并使用nginx为基础镜像构建镜像**2048**为例。在构建镜像之前，需要先创建Dockerfile文件。

步骤1 从镜像仓库拉取nginx镜像，作为基础镜像：

```
docker pull nginx
```

步骤2 下载2048静态页面应用：

```
git clone https://gitee.com/jorgensen/2048.git
```

步骤3 构建Dockerfile。

1. 输入命令：
   ```
   vi Dockerfile
   ```
2. 编辑Dockerfile文件内容：
   ```
   FROM nginx

   MAINTAINER Allen.Li@gmail.com
   COPY 2048 /usr/share/nginx/html

   EXPOSE 80

   CMD ["nginx", "-g", "daemon off;"]
   ```
   - **nginx**为基础镜像，基础镜像可根据创建的应用类型进行选择，例如创建的是Java应用，基础镜像可选择Java镜像。
   - **/usr/share/nginx/html**为nginx存放Web静态页面的路径。
   - **80**为容器端口。

   说明

   Dockerfile内容格式，详情请参考**Dockerfile reference**。

步骤4 构建镜像2048。

1. 输入命令：
```
docker build -t='2048' .
```
创建镜像成功，如下图：

图 1-1 成功构建镜像

```
[root@ecs-8b9a ~]# docker build -t='2048' .
Sending build context to Docker daemon  1.343MB
Step 1/5 : FROM nginx
 ---> dd34e67e3371
Step 2/5 : MAINTAINER Allen.Li@gmail.com
 ---> Running in c10a4ae5b0f2
Removing intermediate container c10a4ae5b0f2
 ---> bae735c6b9e5
Step 3/5 : COPY . /usr/share/nginx/html
 ---> 50d8b973a076
Step 4/5 : EXPOSE 80
 ---> Running in ada8dcf0df3f
Removing intermediate container ada8dcf0df3f
 ---> 45d7fccc6ad4
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
 ---> Running in fcafc573a441
Removing intermediate container fcafc573a441
 ---> 71eecd989add
Successfully built 71eecd989add
Successfully tagged 2048:latest
```

2. 查询镜像：
```
docker images
```
显示如下图，说明镜像构建成功：

图 1-2 查询镜像

```
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
2048         latest    71eecd989add   3 minutes ago    134MB
nginx        latest    dd34e67e3371   2 days ago       133MB
```

----结束

上传镜像

步骤1 连接容器镜像服务。

1. 登录管理控制台，展开“服务列表”，选择“容器服务 > 容器镜像服务”。
2. 在左侧菜单栏选择“我的镜像”，单击右侧“客户端上传”，在弹出的页面中单击“生成临时登录指令”，单击 复制登录指令。

   说明

   此处生成的临时登录指令有效期为24小时，若需要长期有效的登录指令，请参见**获取长期有效登录指令**。
3. 在安装容器引擎的机器中执行上一步复制的登录指令。
   登录成功会显示“login succeeded”。

步骤2 上传镜像。

1. 在安装容器引擎的机器给2048镜像打标签。

   docker tag [*镜像名称:版本名称*] [*镜像仓库地址*]/[*组织名称*]/[*镜像名称:版本名称*]

   样例如下：

   docker tag 2048:latest {Image repository address}/cloud-develop/2048:latest

   其中：

   - {Image repository address}为容器镜像服务的镜像仓库地址。
   - cloud-develop为镜像的组织名称。
   - 2048:latest为镜像名称和版本号。

2. 上传镜像至镜像仓库。

   docker push [*镜像仓库地址*]/[*组织名称*]/[*镜像名称:版本名称*]

   样例如下：

   docker push {Image repository address}/cloud-develop/2048:latest

   终端显示如下信息，表明上传镜像成功。

   ```
   6d6b9812c8ae: Pushed
   695da0025de6: Pushed
   fe4c16cbf7a4: Pushed
   v1: digest: sha256:eb7e3bbd8e3040efa71d9c2cacfa12a8e39c6b2ccd15eac12bdc49e0b66cee63 size: 948
   ```

   返回容器镜像服务控制台，在“我的镜像”页面，执行刷新操作后可查看到对应的镜像信息。

----结束

步骤二：创建命名空间

- 通过控制台中创建命名空间webapp。

步骤1 登录**云容器实例 CCI2.0**控制台。

步骤2 左侧导航栏中选择“命名空间”。

步骤3 进入命名空间页面，单击右侧“创建命名空间”。

步骤4 填写命名空间名称。

说明

- 命名空间名称在云容器实例中需全局唯一。
- 请输入1到63个字符的字符串，可以包含小写英文字母、数字和中划线（-），并以小写英文字母或数字开头，小写英文字母或数字结尾。

步骤5 监控配置(可选)。

参数	说明
对接AOM(可选)	开启后可选择对接AOM实例。

步骤6 网络平面配置。

表 1-1 网络平面配置

参数	说明
启用IPv6	开启后支持ipv4/ipv6双栈协议网络。
虚拟私有云	选择云容器实例所在的虚拟私有云VPC，如果没有可选项可以单击右侧“新建虚拟私有云”创建。创建后不可修改。 建议使用网段：10.0.0.0/8~22，172.16.0.0/12~22，192.168.0.0/16~22。 **须知** • 此处VPC和子网的网段不能为10.247.0.0/16，10.247.0.0/16是云容器实例预留给负载访问的网段。如果您使用此网段，后续可能会造成IP冲突，导致负载无法创建或服务不可用；如果您不需要通过负载访问，而是直接访问Pod，则可以使用此网段。 • 命名空间创建完成后，在“命名空间 > 子网”中可查看到VPC和子网信息。
子网	选择子网，如果没有可选项可以单击右侧“新建子网”创建。创建后子网不可修改。 • 创建的namespace会在设置的子网中预热部分IP，默认个数为10个。 • 在**高级设置**中可以设置预热的个数。 • 创建namespace后，由于预热了部分IP，会影响设置的subnet和VPC的删除，需要删除namespace之后才能正常删除对应的subnet和VPC。 **说明** 您需要关注子网的可用IP数，确保有足够数量的可用IP，如果没有可用IP，则会导致负载创建失败。
安全组	选择已有安全组作为该命名空间下安全组，如果没有可选项可以单击右侧“创建安全组”创建。创建后不可修改。

步骤7 高级设置（可选）。

每个命名空间下都提供了一个IP池，申请IP需要一段时间，如果需要快速创建负载，减少IP的申请时间，可通过自定义资源池大小来实现。

例如，某业务线日常的负载数为200，当达到流量高峰时，IP资源池会自动扩容，瞬间将IP资源池扩容到65535（IP资源池大小），同时会在回收间隔23h（IP资源池回收间隔）之后，进行回收超过资源池大小的部分即（65535-200=65335）个。

表 1-2 高级设置（**可选**）

参数	说明
预热 IP 资源池大小（个）	• 为每个命名空间预热一个 IP 池，用来加速容器组创建。 • 预热IP资源池的大小不能超过65535个。 • 使用通用型pod规格时，建议根据业务配置合理的预热IP资源池大小，以加快负载启动速度。 • 请合理配置预热IP数量，如配置IP数量过大，可能出现预热IP资源池大于子网可用IP数，导致子网IP被耗尽，进而影响用户使用其他服务。
预热 IP 资源池回收间隔（h）	IP 资源池弹性扩容出来的空闲 IP 资源，在一定时间内可进行回收。 **说明** 网卡回收机制： • 回收时间要求：根据network上配置的 yangtse.io/warm-pool-recycle-interval（单位小时）时间，控制该网卡是否能够完成回收。如 yangtse.io/warm-pool-recycle-interval 配置为 24，则回收的网卡必须已经申请了超过24小时后才会被回收。 • 回收速率：内部维持一个合理的速率，按照每次最多50个网卡的速率进行回收，避免回收过快或频繁回收导致网卡的重复创建和删除动作。

步骤8 单击“确定”。

创建完成后，可以在命名空间详情中看到VPC、子网等信息。

----结束

- 通过ccictl工具创建命名空间webapp和对应的网络，完成工具配置后执行以下步骤：

步骤1 创建命名空间：

```
ccictl create namespace webapp
```

```
root@                                   :~# ccictl create namespace webapp
namespace.cci/                         /webapp created
```

步骤2 创建命名空间对应的网络，yaml示例如下：

```
apiVersion: yangtse/v2
kind: Network
metadata:
  annotations:
    yangtse.io/domain-id: <domain_id> # 账号ID
    yangtse.io/project-id: <project_id> # 项目ID
  name: cci-network
  namespace: webapp
spec:
  networkType: underlay_neutron
  securityGroups:
    - <security_group_id> # 安全组ID
  subnets:
    - subnetID: <subnet_id> # 子网ID
```

```
root@                                   :~# ccictl apply -f network.yaml
network.yangtse/cci-network created
```

----结束

步骤三：创建容器组

在创建容器组前，需要使用VPCEP将您命名空间的网络与华为云其他服务后端网络连通，请参考**购买云服务VPCEP**章节进行操作。创建VPCEP之后，您可参考以下方式创建容器组：

- 通过控制台创建容器组：
  a. 登录**云容器实例 CCI2.0**控制台。
  b. 左侧导航栏中选择“负载管理”，单击右侧“容器组”，再单击“YAML创建”。

  c. 添加基本信息，yaml示例如下：

```
apiVersion: cci/v2
kind: Pod
metadata:
  labels:
    app: webapp-2048
  name: webapp-2048
  namespace: webapp
spec:
  containers:
    - image: {Image repository address}/cloud-develop/2048:latest # 上传的镜像
      name: webapp-2048
      ports:
        - containerPort: 80
          protocol: TCP
      resources:
        limits:
          cpu: 500m
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
  dnsPolicy: Default
```

- 通过ccictl工具创建容器组。将上述yaml保存为pod.yaml并执行以下命令：

```
ccictl apply -f pod.yaml
```

```
root@                                   :~# ccictl apply -f pod.yaml
pod.cci/webapp-2048 created
```

步骤四：访问容器组

- 通过控制台配置服务访问容器组：
  a. 左侧导航栏中选择“服务管理”，再单击右侧“YAML创建”。

  b. 创建Load Balancer类型的服务，使用的ELB需包含公网IP，yaml示例如下：

```
kind: Service
apiVersion: cci/v2
metadata:
  name: service-2048
  namespace: webapp
  annotations:
    kubernetes.io/elb.class: elb
    kubernetes.io/elb.id: <elb_id> # 使用的ELB需要包含公网IP
spec:
  ports:
    - name: service-port
      protocol: TCP
      port: 80
      targetPort: 80
  selector:
    app: webapp-2048
  type: LoadBalancer
```

- 通过ccictl配置服务。保存上述yaml为service.yaml并执行命令：

```
ccictl apply -f service.yaml
```

```
root@                                   :~# ccictl apply -f service.yaml
service.cci/service-2048 created
```

容器组和服务创建成功后，您可以使用浏览器通过 http://<公网IP>/2048/index.html 访问您的Web应用。

步骤五：清理资源

- 使用云容器实例管理控制台，执行以下清理CCI服务资源：
  a. 登录**云容器实例 CCI2.0**控制台。
  b. 左侧导航栏中选择“负载管理 > 容器组”。
  c. 选择需要删除的容器组，在对应容器组右侧操作列单击“删除”。

> **说明**
>
> 如需删除CCI服务所绑定的ELB，请前往云容器实例 CCI2.0控制台先删除“网络管理”中所创建的服务，然后前往弹性负载均衡控制台删除弹性负载均衡。

- 使用ccictl，执行如下命令清理CCI服务资源：

```
ccictl delete -f service.yaml
ccictl delete -f pod.yaml
ccictl delete namespace webapp
```

```
root@                                   :~# ccictl apply -f service.yaml
service.cci/service-2048 created
root@                                   :~# ccictl delete -f service.yaml
service.cci "service-2048" deleted
root@                                   :~# ccictl delete -f pod.yaml
pod.cci "webapp-2048" deleted
root@                                   :~# ccictl delete namespace webapp
namespace.cci "webapp" deleted
```