云容器实例

最佳实践

文档版本 01

发布日期 2020-09-22

华为技术有限公司

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，华为公司对本文档内容不做任何明示或默示的声明或保证。

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。

目 录

1 Dockerfile 参数在云容器实例中如何使用

如果您了解容器引擎的使用，Dockerfile文件的一些配置如何对应到云容器实例中去使用呢?

下面通过一个例子来说明他们之间的关系，这样您就可以更好的了解和熟悉云容器实例。

```
FROM ubuntu:16.04

ENV VERSION 1.0

VOLUME /var/lib/app

EXPOSE 80

ENTRYPOINT ["./entrypoint.sh"]
CMD ["start"]
```

上面是一个Dockerfile文件，包含一些常见的参数ENV、VOLUME、EXPOSE、ENTRYPOINT、CMD，这些参数在云容器实例中可以按如下方法配置。

- ENV为环境变量，在云容器实例中创建负载的时候，可以在高级配置中设置，如下所示。

- VOLUME为定义容器卷，通常配合**docker run -v 宿主机路径:容器卷路径**一起使用。

  云容器实例中支持将云硬盘挂载到容器中，只需在创建负载时添加云硬盘卷，并配置大小、挂载路径（也就是容器卷的路径）即可。

- ENTRYPOINT与CMD对应云容器实例中高级配置的启动命令，详细内容请参见**容器启动命令**。

- EXPOSE即暴露某个端口，通常在启动容器时配合**docker run -p <宿主端口>:<容器端口>**一起使用，云容器实例中容器如果要对外暴露端口，只需在创建负载的时候配置**负载访问端口:容器端口**的映射，这样就可以通过**负载请求域名:负载访问端口**访问到容器。

2 使用多种方法创建工作负载

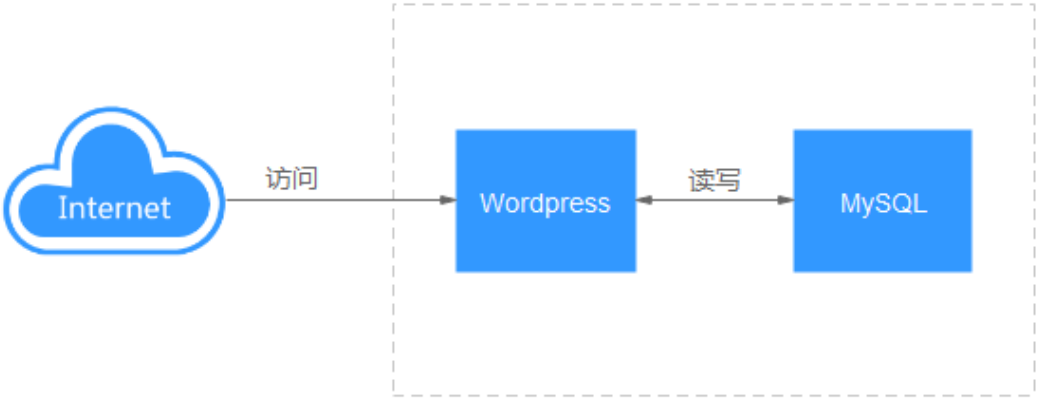
2.1 概述

在云容器实例中，您可以使用多种方法创建负载，包括使用云容器实例的Console控制台界面、调用API部署应用，那这些方式的使用有什么不同的地方呢？这些方法又与直接运行Docker run命令运行容器有什么区别呢？

本文将通过运行一个Wordpress + MySQL的博客为例，比较这几种方法之间的异同，以利于您挑选合适的使用方法。

WordPress是使用PHP语言开发的博客平台。用户可以在支持PHP和MySQL数据库的服务上架设属于自己的网站，也可以把WordPress当作一个内容管理系统来使用。更多WordPress信息可以通过官方网站了解：**https://wordpress.org/**。

WordPress需配合MySQL一起使用，WordPress运行内容管理程序，MySQL作为数据库存储数据。在容器中运行通常会将WordPress和MySQL分别运行两个容器中，如下图所示。



2.2 使用 Docker run 运行容器

镜像准备

WordPress和MySQL的镜像都是通用镜像，可以直接从镜像中心获取。

您可以在安装了容器引擎的机器上使用**docker pull**命令即可下载镜像，如下所示。

```
docker pull mysql:5.7
docker pull wordpress
```

下载完成后，执行**docker images**命令可以看到本地已经存在两个镜像，如下图所示。

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
wordpress           latest              6a837ea4bd22        6 days ago          408MB
mysql               5.7                 0d16d0a97dd1        5 weeks ago         372MB
```

运行容器

使用容器引擎可以直接运行Wordpress和MySQL，且可以使用**--link**参数将两个容器连接，在不改动代码的情况下让Wordpress的容器访问MySQL的容器。

执行下面的命令运行MySQL。

```
docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=******** -d mysql:5.7
```

参数解释如下：

- --name指定容器的名称为some-mysql。
- -e指定容器的环境变量，如这里指定环境变量MYSQL_ROOT_PASSWORD的值为**********，请替换为您设置的密码。
- -d表示在后台运行。

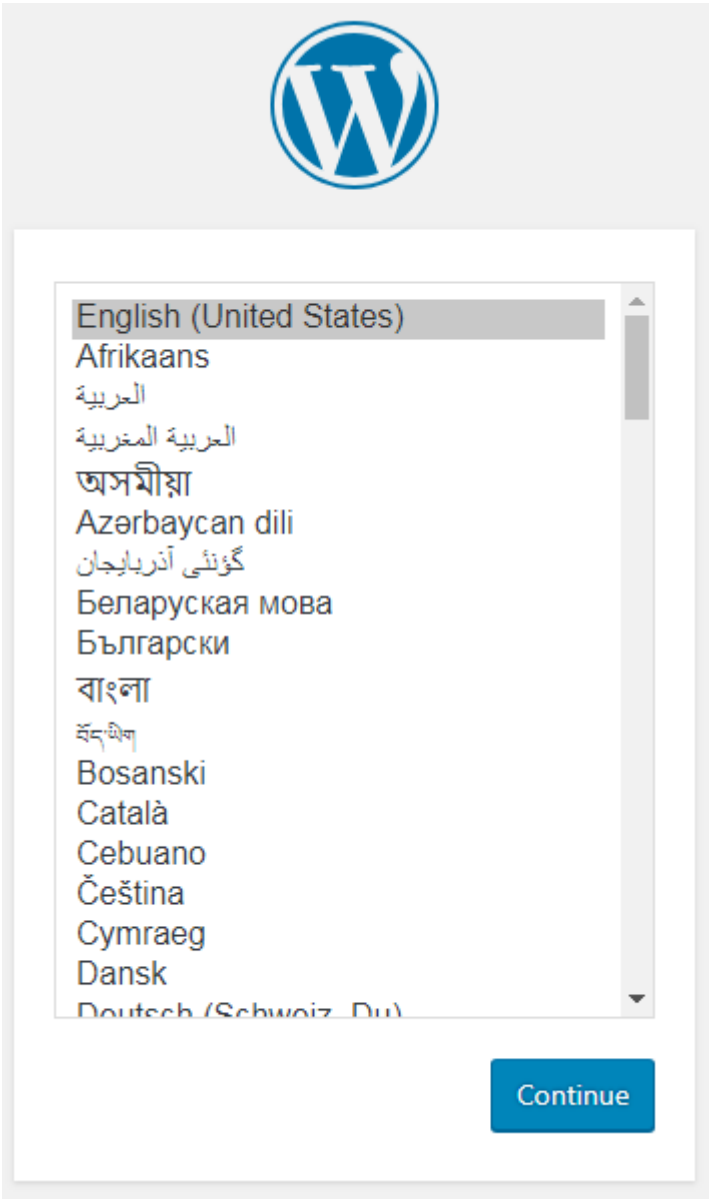
执行下面的命令运行Wordpress。

```
docker run --name some-wordpress --link some-mysql:mysql -p 8080:80 -e
WORDPRESS_DB_PASSWORD=******** -d wordpress
```

参数解释如下：

- --name指定容器的名称为some-wordpress 。
- --link指定some-wordpress容器链接some-mysql容器，并将some-mysql命名为mysql。这里--link只是提供了一种方便，不使用--link的话，可以指定some-wordpress的环境变量WORDPRESS_DB_HOST访问mysql的IP与端口。
- -p指定端口映射，如这里将容器的80端口映射到主机的8080端口。
- -e指定容器的环境变量，如这里指定环境变量WORDPRESS_DB_PASSWORD的值为**********，请替换为您设置的密码。Wordpress的环境变量WORDPRESS_DB_PASSWORD必须与MySQL的环境变量MYSQL_ROOT_PASSWORD值相同，这是因为Wordpress需要密码访问MySQL数据库。
- -d表示在后台运行。

Wordpress运行之后，就可以在本机通过http://127.0.0.1:8080访问Wordpress博客了，如下所示。



2.3 使用控制台创建负载

使用Docker run运行容器章节使用docker run命令运行了Wordpress博客，但是在很多场景下使用容器引擎并不方便，如应用弹性伸缩、滚动升级等。

云容器实例提供无服务器容器引擎，让您不需要管理集群和服务器，只需要三步简单配置，即可畅享容器的敏捷和高性能。云容器实例支持创建无状态负载（Deployment）和有状态负载（StatefulSet），并基于Kubernetes的负载模型增强了容器安全隔离、负载快速部署、弹性负载均衡、弹性扩缩容、蓝绿发布等重要能力。

创建命名空间

步骤1 登录云容器实例管理控制台，左侧导航栏中选择**命名空间**。

步骤2 在对应类型的命名空间下单击“创建”。

步骤3 填写命名空间名称。

步骤4 设置VPC。

选择使用已有VPC或新建VPC，新建VPC需要填写VPC网段，建议使用网段：10.0.0.0/8~24，172.16.0.0/12~24，192.168.0.0/16~24。

步骤5 设置子网网段。

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。

步骤6 单击“创建”。

----结束

创建 MySQL 负载

步骤1 登录云容器实例管理控制台，左侧导航栏中选择**工作负载 > 无状态（Deployment）**，在右侧页面单击“创建负载”。

步骤2 添加基本信息。

- **负载名称**：mysql。
- **命名空间**：选择**创建命名空间**创建的命名空间。
- **Pod数量**：本例中修改Pod数量为1。
- **Pod规格**：选择通用计算型，CPU 0.5核，内存 1GB。

- **容器配置**

  a. 在开源镜像中心搜索并选择mysql镜像。

b. 配置镜像参数，选择镜像版本为5.7，CPU和内存配置为0.5核和1G。

c. 在高级配置中，添加容器的环境变量MYSQL_ROOT_PASSWORD，并填入变量，变量值为MySQL数据库的密码（需自行设置）。

步骤3 单击“下一步”，配置负载信息，负载访问选择内网访问（可以被云容器实例中其他负载通过“服务名称:端口”方法），将“服务名称”定义为mysql，并指定负载访问端口3306映射到容器的3306端口（mysql镜像的默认访问端口）。

这样在云容器实例内部，通过mysql:3306就可以访问MySQL负载。

步骤4 配置完成后，单击“下一步”，确认规格后单击“提交”。

在负载列表中，待负载状态为“运行中”，负载创建成功。

----结束

创建 Wordpress 负载

步骤1 登录云容器实例管理控制台，左侧导航栏中选择**工作负载 > 无状态（Deployment）**，在右侧页面单击“创建负载”。

步骤2 添加基本信息。

- **负载名称**：wordpress。

- **命名空间**：选择**创建命名空间**创建的命名空间。
- **Pod数量**：本例中修改Pod数量为2。
- **Pod规格**：选择通用计算型，CPU 0.5核，内存 1GB。

- 容器配置：

  a. 在开源镜像中心搜索并选择wordpress镜像。

  b. 配置镜像参数，选择镜像版本为php7.1，CPU和内存配置为0.5核和1G。

  c. 在高级配置中，设置环境变量，使WordPress可以访问MySQL数据库。

表 2-1 环境变量说明

变量名	变量/变量引用
WORDPRESS_DB_HOST	MySQL的访问地址。 示例：10.***.***.***:3306
WORDPRESS_DB_PASSWORD	MySQL数据库的密码，此处密码必须与**创建MySQL负载**设置MySQL的密码相同。

步骤3 单击“下一步”，配置负载信息。

负载访问选择公网访问，服务名称为“wordpress”，选择ELB实例（如果没有实例可以单击“新建增强型ELB实例”创建），选择ELB协议为“HTTP”，ELB端口号为9012，指定负载访问端口“8080”映射到容器的“80”端口（wordpress镜像的默认访问端口），HTTP路由设置为“/”（即通过“http://elb ip:外部端口”就可以访问wordpress）并映射到8080负载端口。

步骤4 配置完成后，单击“下一步”，确认规格后单击“提交”。

在负载列表中，待负载状态为“运行中”，负载创建成功。您可以单击负载名进入负载详情界面。

在“访问配置”处选择“公网访问”，查看访问地址，即ELB实例的“IP地址:端口”。

----结束

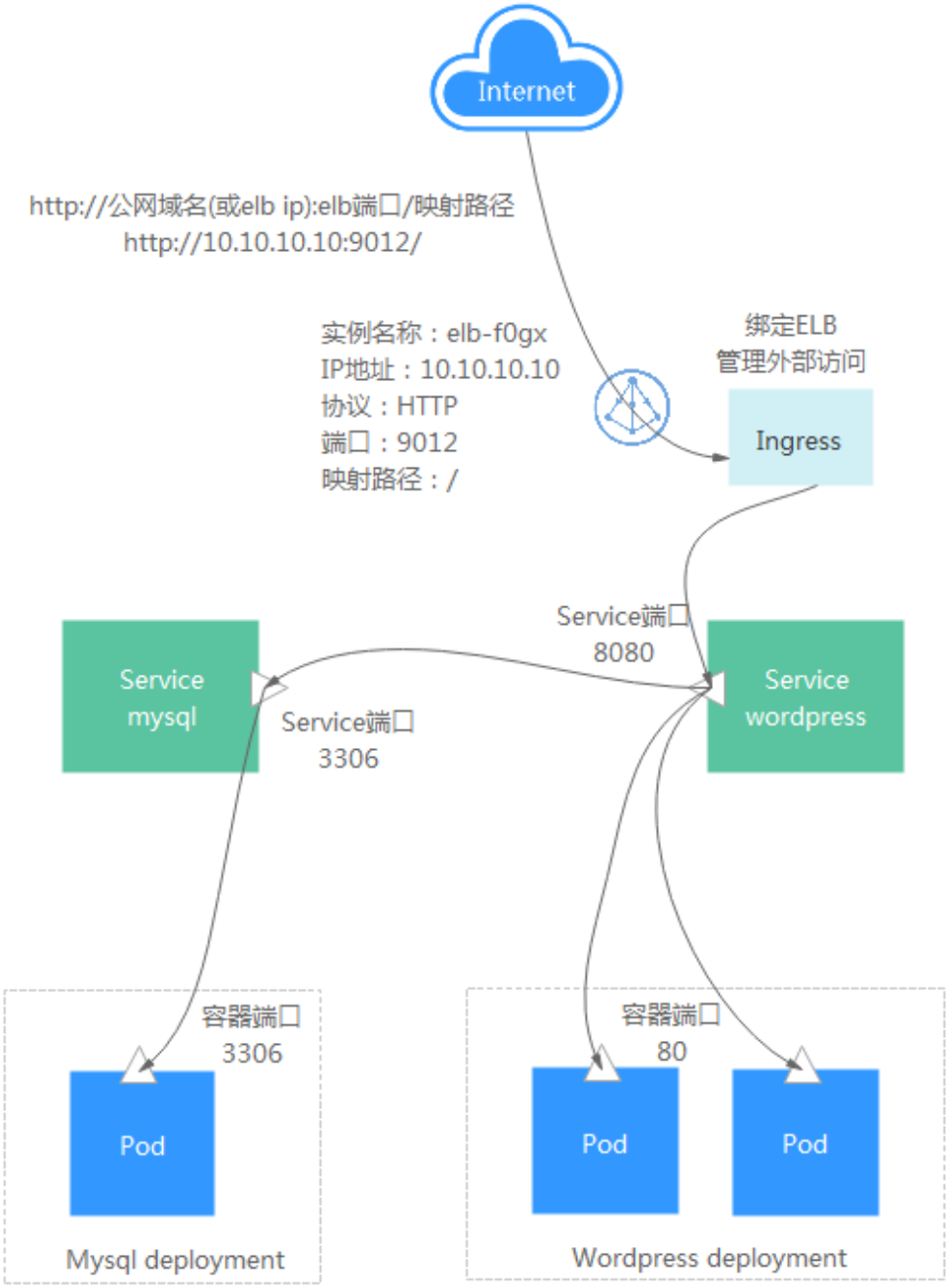
2.4 调用 API 创建负载

云容器实例原生支持Kubernetes API，相比从控制台创建负载，使用API的粒度更细一些。

Kubernetes中，运行容器的最小资源单位是Pod，一个Pod封装一个或多个容器、存储资源、一个独立的网络IP等。实际使用中很少直接创建Pod，而是使用Kubernetes中称为Controller的抽象层来管理Pod实例，例如Deployment和StatefulSet。另外在Kubernetes中使用Service定义一系列Pod以及访问这些Pod的策略的资源对象，使用Ingress管理外部访问的资源对象。 如果您对Kubernetes的资源不熟悉，请参见《**云容器实例开发指南**》了解各资源的关系。

对于Wordpress应用，可以按照下图调用API创建一系列资源。

- MySQL：创建一个Deployment部署mysql，创建一个Service定义mysql的访问策略。
- Wordpress：创建一个Deployment部署wordpress，创建Service和Ingress定义wordpress的访问策略。



Namespace

步骤1 调用**创建Namespace**接口创建命名空间，并指定使用命名空间的类型。

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "namespace-test",
    "annotations": {
      "namespace.kubernetes.io/flavor": "gpu-accelerated"
    }
  },
  "spec": {
    "finalizers": [
      "kubernetes"
```



```
    ]
  }
}
```

步骤2 调用**创建Network**接口创建网络，与VPC与子网关联。

```
{
  "apiVersion": "networking.cci.io/v1beta1",
  "kind": "Network",
  "metadata": {
    "annotations": {
      "network.alpha.kubernetes.io/default-security-group": "{{security-group-id}}",
      "network.alpha.kubernetes.io/domain-id": "{{domain-id}}",
      "network.alpha.kubernetes.io/project-id": "{{project-id}}"
    },
    "name": "test-network"
  },
  "spec": {
    "availableZone": "cnnorth-1a",
    "cidr": "192.168.0.0/24",
    "attachedVPC": "vpc-id",
    "networkID": "network-id",
    "networkType": "underlay_neutron",
    "subnetID": "subnet-id"
  }
}
```

----结束

MySQL

步骤1 调用**创建Deployment**接口部署MySQL。

- Deployment名称为mysql。
- 设置Pod的标签为app:mysql。
- 使用mysql:5.7镜像。
- 设置容器环境变量MYSQL_ROOT_PASSWORD为“********”，请替换为您设置的密码。

```
{
  "apiVersion": "apps/v1",
  "kind": "Deployment",
  "metadata": {
    "name": "mysql"
  },
  "spec": {
    "replicas": 1,
    "selector": {
      "matchLabels": {
        "app": "mysql"
      }
    },
    "template": {
      "metadata": {
        "labels": {
          "app": "mysql"
        }
      },
      "spec": {
        "containers": [
          {
            "image": "mysql:5.7",
            "name": "container-0",
            "resources": {
              "limits": {
                "cpu": "500m",
                "memory": "1024Mi"
```



```
                    },
                    "requests": {
                        "cpu": "500m",
                        "memory": "1024Mi"
                    }
                },
                "env": [
                    {
                        "name": "MYSQL_ROOT_PASSWORD",
                        "value": "********"
                    }
                ]
            }
        ],
        "imagePullSecrets": [
            {
                "name": "imagepull-secret"
            }
        ]
    }
  }
}
```

步骤2 调用**创建Service接口**创建一个Service，定义**步骤1**中创建的Pod的访问策略。

- Service名称为mysql。
- 选择标签为app:mysql的Pod，即关联**步骤1**中创建的Pod。
- 负载访问端口3306映射到容器的3306端口。
- Service的访问类型为NodePort，即通过Pod所在节点的端口访问。

```
{
  "apiVersion": "v1",
  "kind": "Service",
  "metadata": {
    "name": "mysql",
    "labels": {
      "app": "mysql"
    }
  },
  "spec": {
    "selector": {
      "app": "mysql"
    },
    "ports": [
      {
        "name": "service0",
        "targetPort": 3306,
        "port": 3306,
        "protocol": "TCP"
      }
    ],
    "type": "ClusterIP"
  }
}
```

----结束

Wordpress

步骤1 调用**创建Deployment**接口部署Wordpress。

- Deployment名称为wordpress。
- replicas值为2，表示创建2个pod。
- 设置Pod的标签为app:wordpress。

- 使用wordpress:latest镜像。
- 设置容器环境变量WORDPRESS_DB_PASSWORD为“********”，请替换为您设置的密码。此处的密码必须与MySQL的MYSQL_ROOT_PASSWORD一致。

```
{
   "apiVersion": "apps/v1",
   "kind": "Deployment",
   "metadata": {
      "name": "wordpress"
   },
   "spec": {
      "replicas": 2,
      "selector": {
         "matchLabels": {
            "app": "wordpress"
         }
      },
      "template": {
         "metadata": {
            "labels": {
               "app": "wordpress"
            }
         },
         "spec": {
            "containers": [
               {
                  "image": "wordpress:latest",
                  "name": "container-0",
                  "resources": {
                     "limits": {
                        "cpu": "500m",
                        "memory": "1024Mi"
                     },
                     "requests": {
                        "cpu": "500m",
                        "memory": "1024Mi"
                     }
                  },
                  "env": [
                     {
                        "name": "WORDPRESS_DB_PASSWORD",
                        "value": "********"
                     }
                  ]
               }
            ],
            "imagePullSecrets": [
               {
                  "name": "imagepull-secret"
               }
            ]
         }
      }
   }
}
```

步骤2 调用**创建Service接口**创建一个Service，定义**步骤1**中创建的Pod的访问策略。

- Service名称为wordpress。
- 选择标签为app:wordpress的Pod，即关联**步骤1**中创建的Pod。
- 负载访问端口8080映射到容器的80端口，80端口为wordpress镜像的默认对外暴露的端口。
- Service的访问类型为NodePort，即通过Pod所在节点的端口访问。

```
{
   "apiVersion": "v1",
   "kind": "Service",
```



```
  "metadata": {
    "name": "wordpress",
    "labels": {
      "app": "wordpress"
    }
  },
  "spec": {
    "selector": {
      "app": "wordpress"
    },
    "ports": [
      {
        "name": "service0",
        "targetPort": 80,
        "port": 8080,
        "protocol": "TCP"
      }
    ],
    "type": "ClusterIP"
  }
}
```

步骤3 调用**创建Ingress**接口创建一个Ingress，定义wordpress的外部访问策略，即关联ELB实例（ELB实例需要与Wordpress负载在同一个VPC内）。

- metadata.annotations.kubernetes.io/elb.id：必须为data。
- metadata.annotations.kubernetes.io/elb.ip：ELB实例的IP地址。
- metadata.annotations.kubernetes.io/elb.port：ELB实例的端口。
- spec.rules：访问服务的规则集合。path列表，每个path（比如：/）都关联一个backend（比如“wordpress:8080”）。backend是一个service:port的组合。Ingress的流量被转发到它所匹配的backend。

这里配置完后，访问ELB的IP:端口的流量就会流向wordpress:8080这个Service，由于Service是关联了wordpress的Pod，所以最终访问的就是**步骤1**中部署的wordpress容器。

```
{
  "apiVersion": "extensions/v1beta1",
  "kind": "Ingress",
  "metadata": {
    "name": "wordpress",
    "labels": {
      "app": "wordpress",
      "isExternal": "true",
      "zone": "data"
    },
    "annotations": {
      "kubernetes.io/elb.id": "2d48d034-6046-48db-8bb2-53c67e8148b5",
      "kubernetes.io/elb.ip": "10.10.10.10",
      "kubernetes.io/elb.port": "9012"
    }
  },
  "spec": {
    "rules": [
      {
        "http": {
          "paths": [
            {
              "path": "/",
              "backend": {
                "serviceName": "wordpress",
                "servicePort": 8080
              }
            }
          ]
        }
```



```
        }
      ]
    }
}
```

----结束

3 使用 Tensorflow 训练神经网络

本文主要演示在云容器实例中创建GPU类型的负载，以tensorflow的图像分类为示例，演示在容器中直接使用GPU训练一个简单的神经网络。

使用容器化的方式做此类人工智能训练与推理有如下优势：

- 容器化消除环境差异，不需要自己安装各种软件和配套版本，如python，tensorflow，cuda toolkit等软件。
- GPU驱动免安装。
- 低成本，按秒计费。
- serverless带来的免VM运维。

镜像制作

tensorflow社区有tensoflow的基础镜像，已经装好了基础的tensorflow库，它分支持GPU和支持CPU两个版本，在镜像中心即可下载。

- GPU版本地址为 tensorflow/tensorflow:1.15.0-gpu
- CPU版本地址为 tensorflow/tensorflow:1.13.0

本文采用tensorflow官网中一个已经训练好的模型，对图片进行分类，模型名称Inception-v3。Inception-v3是在2012年ImageNet视觉识别挑战赛上训练出的模型，它将一个非常大的图片集进行了1000个种类的图片分类。Github有使用Inception-v3进行图片分类的代码。

训练模型的代码，均在工程**https://gpu-demo.obs.cn-north-1.myhuaweicloud.com/gpu-demo.zip**中，您需要将代码下载解压，并将代码工程打入镜像中。下面附上制作镜像的Dockerfile文件内容：

```
FROM tensorflow/tensorflow:1.15.0-gpu
ADD gpu-demo /home/project/gpu-demo
```

其中ADD将gpu-demo工程拷贝到镜像的/home/project目录下，可以根据自己需要修改。

执行**docker build -t tensorflow/tensorflow:v1 .** 命令制作镜像（.表示当前目录，即Dockerfile文件所在目录）。

镜像制作好后需要上传到容器镜像服务，具体步骤请参见**https://support.huaweicloud.com/usermanual-swr/swr_01_0009.html**。

创建 Tensorflow 负载

步骤1 登录云容器实例管理控制台。

步骤2 创建GPU型命名空间，填写命名空间名称，设置好VPC和子网网段后，单击“创建”。

图 3-1 GPU 型命名空间

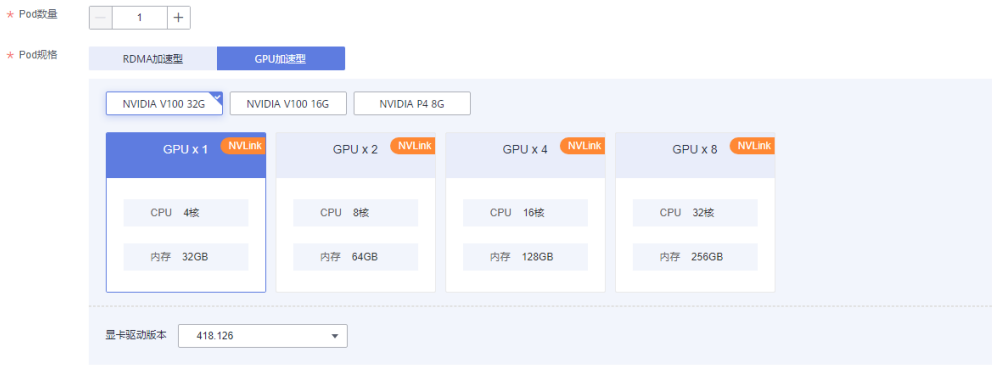
步骤3 左侧导航栏中选择“工作负载 > 无状态（Deployment）”，在右侧页面中单击“创建无状态负载”。

步骤4 配置负载信息。

1. 填写基本信息，选择**步骤2**创建的命名空间，Pod数量选择为“1”，选择Pod规格为“GPU加速型”，显卡的驱动版本选择“418.126”，如下所示。

   GPU Pod的详细规格和显卡驱动的说明请参见**Pod规格**。

   图 3-2 选择 GPU 容器规格



2. 选择需要的容器镜像，这里选择的上传到镜像容器仓库的tensorflow镜像。
3. 在容器设置下面的高级设置中，挂载一个NFS类型的文件存储卷，用于保存训练后的数据。

   图 3-3 挂载 NFS 存储

4. 在启动命令中输出执行命令和参数。

- 可执行命令：/bin/bash
- 参数1：-c
- 参数2：python /home/project/gpu-demo/cifar10/cifar10_multi_gpu_train.py --num_gpus=1 --data_dir=/home/project/gpu-demo/cifar10/data --max_steps=10000 --train_dir=/tmp/sfs0/train_data; while true; do sleep 10; done

  此处 **--train_dir** 表示训练结果存储路径，其前缀 **/tmp/sfs0** 需要与**步骤4.3**中设置的NFS“容器内挂载路径”路径保持一致，否则训练结果无法写入NFS中。

  --max_steps表示训练迭代的次数，这里指定了10000次迭代，完成模型训练大概耗时3分钟，如果不指定，默认是1000000次迭代，耗时会比较长。max_steps数值越大，训练时间越久，结果越精确。

该命令是训练图片分类模型，然后单击“下一步”。

图 3-4 设置容器启动命令

5. 配置负载访问信息。

   本例中选择“不启用”，单击“下一步”。

6. 单击“提交”，然后单击“返回工作负载列表”。

   在负载列表中，待负载状态为“运行中”时，负载创建成功。

----结束

使用已有模型分类图片

步骤1 单击负载名称，进入负载详情界面，选择“Pod列表>终端”Tab页。当黑色区域文本框中出现#号时，说明已登录。

图 3-5 Pod 终端访问

步骤2 进入到工程所在目录，执行python classify_image.py --model_dir=model命令，可以看到分类结果。

```
# cd /home/project/gpu-demo
# ls -l
total 96
-rw-r--r-- 1 root root  6874 Aug 30 08:09 airplane.jpg
drwxr-xr-x 3 root root  4096 Sep  4 07:54 cifar10
drwxr-xr-x 3 root root  4096 Aug 30 08:09 cifar10_estimator
-rw-r--r-- 1 root root 30836 Aug 30 08:09 dog.jpg
-rw-r--r-- 1 root root 43675 Aug 30 08:09 flower.jpg
drwxr-xr-x 4 root root  4096 Sep  4 02:14 inception
# cd inception
# python classify_image.py --model_dir=model --image_file=/home/project/gpu-demo/
airplane.jpg
...
2019-01-02 08:05:24.891201: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1084] Created
TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 15131 MB memory) -> physical GPU
(device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:0a.0, compute capability: 6.0)
airliner (score = 0.84250)
wing (score = 0.03228)
space shuttle (score = 0.02524)
warplane, military plane (score = 0.00691)
airship, dirigible (score = 0.00664)
```

这里通过**--image_file**指定了要分类的图片，图片如下。执行结果最后几行是分类的label和对应的打分，其中有一行显示airliner(score = 0.84250)，分数越高越准确，可见模型认为这个图片是一架客机。

图 3-6 airliner

也可以不指定要分类的图片，默认将使用下面这张图片分类。

图 3-7 熊猫

执行命令**python classify_image.py –model_dir=mode**

```
# python classify_image.py --model_dir=model
...
2019-01-02 08:02:33.271527: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1084] Created
TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 15131 MB memory) -> physical GPU
(device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:0a.0, compute capability:
6.0)
```



```
giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score =
0.89107)
indri, indris, Indri indri, Indri brevicaudatus (score = 0.00779)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score =
0.00296)
custard apple (score = 0.00147)
earthstar (score = 0.00117)
```

执行结果显示模型认为这是一只大熊猫。

----结束

使用训练的图片分类模型

tensorflow官网中给了一个深度卷积网络的模型代码和训练数据：CIFAR-10。这是个简化的图片分类模型，将图片分成以下10类：airplane, automobile, bird, cat, deer, dog, frog, horse, ship和truck。当然喂给模型的图，也就是训练数据，也是这10种类型的图片。

步骤1 单击负载名称，进入负载详情界面，选择“Pod列表>终端”Tab页，使用代码中提供的cifar10_eval.py校验模型的准确性，这里的checkpoint_dir指定使用刚刚训练出来的模型进行准确性校验。

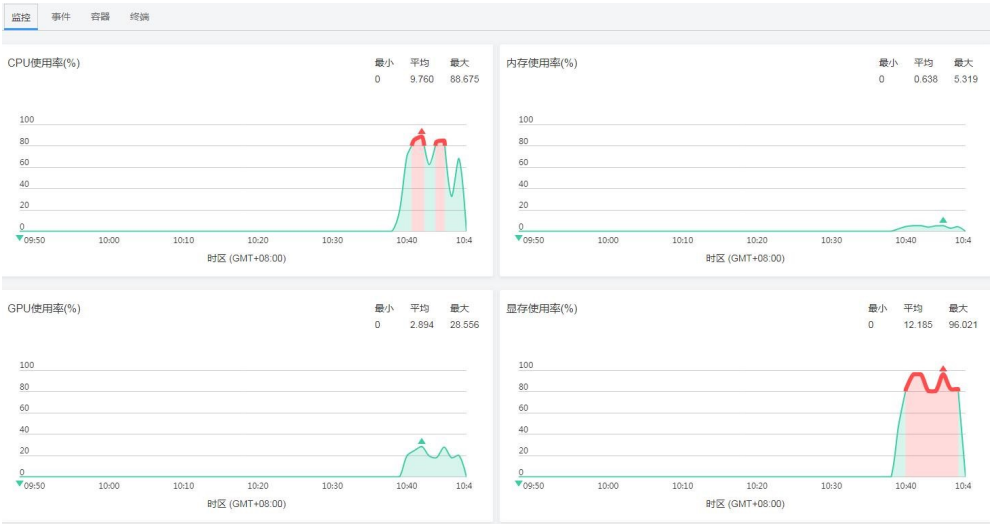
```
# cd /home/project/gpu-demo/cifar10
# python cifar10_eval.py --data_dir=data --checkpoint_dir=/tmp/sfs0/train_data --run_once
...
2019-01-02 08:25:43.914186: precision @1 = 0.817
```

步骤2 继续使用上面的飞机图片进行测试，这里的checkpoint_dir指定使用刚刚训练出来的模型进行图片分类，test_file指定用哪张图片测试。

```
# python label_image.py --checkpoint_dir=/tmp/sfs0/train_data --test_file=/home/project/gpu-demo/
airplane.jpg
   ...
2019-01-02 08:36:42.149700: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1084] Created
TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 15131 MB memory) -> physical GPU
(device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:0a.0, compute capability:
6.0)
airplane (score = 4.28143)
ship (score = 1.92319)
cat (score = 0.03095)
```

可见它准确识别出图中是架飞机。label_image.py是使用刚刚训练的模型来进行图片分类的代码。

同时，在“Pod列表>监控”Tab页中，可以看到各种资源的使用率。



----结束

