容器镜像服务

最佳实践

文档版本 01

发布日期 2023-08-03

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。

安全声明

漏洞处理流程

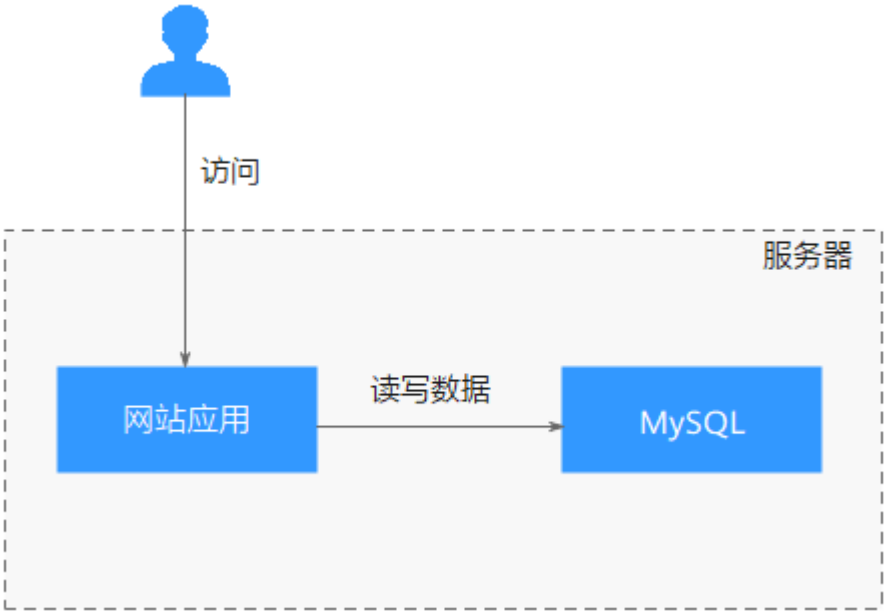
华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 编写高效的 Dockerfile

本章基于容器镜像服务实践所编写，将一个单体应用进行容器改造为例，展示如何写出可读性更好的Dockerfile，从而提升镜像构建速度，构建层数更少、体积更小的镜像。

下面是一个常见企业门户网站架构，由一个Web Server和一个数据库组成，Web Server提供Web服务，数据库保存用户数据。通常情况下，这样一个门户网站安装在一台服务器上。



如果把应用运行在一个Docker容器中，那么很可能写出下面这样的Dockerfile来。

```
FROM ubuntu

ADD . /app

RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y nodejs ssh mysql
RUN cd /app && npm install

# this should start three processes, mysql and ssh
# in the background and node app in foreground
```



```
# isn't it beautifully terrible? <3
CMD mysql & sshd & npm start
```

但是这样Dockerfile有很多问题，这里CMD命令是错误的，只是为了说明问题而写。

下面的内容中将展示对这个Dockerfile进行改造，说明如何写出更好的Dockerfile，共有如下几种处理方法。

- **一个容器只运行一个进程**
- **不要在构建中升级版本**
- **将变化频率一样的RUN指令合一**
- **使用特定的标签**
- **删除多余文件**
- **选择合适的基础镜像**
- **设置WORKDIR和CMD**
- **使用ENTRYPOINT（可选）**
- **ENTRYPOINT脚本中使用exec**
- **优先使用COPY**
- **合理调整COPY与RUN的顺序**
- **设置默认的环境变量、映射端口和数据库逻辑卷**
- **使用EXPOSE暴露端口**
- **使用VOLUME管理数据库逻辑卷**
- **使用LABEL设置镜像元数据**
- **添加HEALTHCHECK**
- **编写.dockerignore文件**

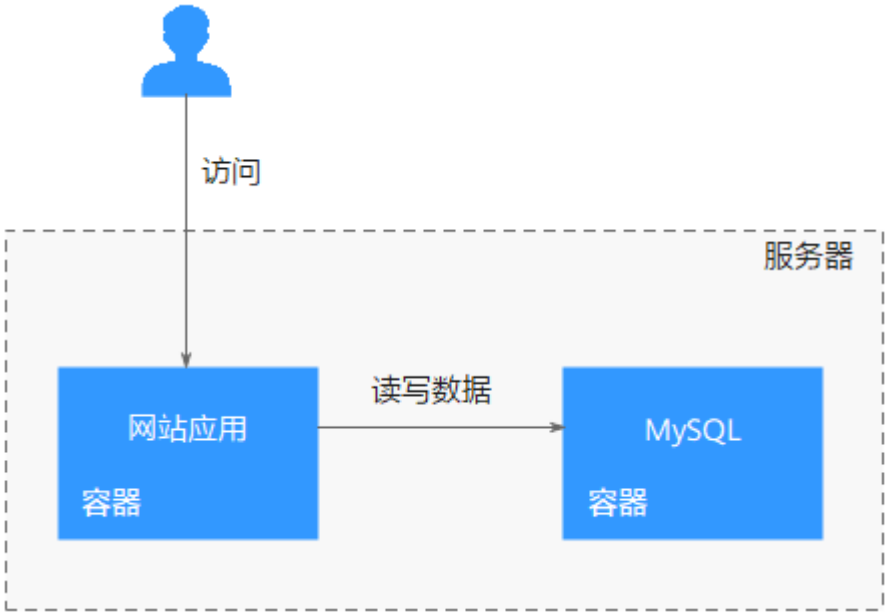
一个容器只运行一个进程

从技术角度讲，Docker容器中可以运行多个进程，您可以将数据库、前端、后端、ssh等都运行在同一个Docker容器中。但是，这样跟未使用容器前没有太大区别，且这样容器的构建时间非常长（一处修改就要构建全部），镜像体积大，横向扩展时非常浪费资源（不同的应用需要运行的容器数并不相同）。

通常所说的容器化改造是对应用整体微服务进行架构改造，改造后，再容器化。这样做可以带来如下好处：

- 单独扩展：拆分为微服务后，可单独增加或缩减每个微服务的实例数量。
- 提升开发速度：各微服务之间解耦，某个微服务的代码开发不影响其他微服务。
- 通过隔离确保安全：整体应用中，若存在安全漏洞，一旦被攻击，所有功能的权限都可能会被窃取。微服务架构中，若攻击了某个服务，只可获得该服务的访问权限，无法入侵其他服务。
- 提升稳定性：如果其中一个微服务崩溃，其他微服务还可以持续正常运行。

因此，上述企业门户网站可以进行如下改造，Web应用和MySQL运行在不同容器中。



MySQL运行在独立的镜像中，这样的好处就是，可以对它们分别进行修改，且不会牵一发而动全身。如下面这个例子所示，可以删除MySQL，只安装node.js。

```
FROM ubuntu

ADD . /app

RUN apt-get update
RUN apt-get upgrade -y

RUN apt-get install -y nodejs
RUN cd /app && npm install

CMD npm start
```

不要在构建中升级版本

为了降低复杂性、减少依赖、减小文件大小、节约构建时间，您应该避免安装任何不必要的包。例如，不要在数据库镜像中包含一个文本编辑器。

如果基础镜像中的某个包过时了，但您不知道具体是哪一个包，您应该联系它的维护者。如果您确定某个特定的包，比如foo需要升级，使用apt-get install -y foo就行，该指令会自动升级foo包。

apt-get upgrade会使得镜像构建过程非常不稳定，在构建时不确定哪些包会被安装，此时可能会产生不一致的镜像。因此通常会删掉apt-get upgrade。

删掉apt-get upgrade后，Dockerfile如下：

```
FROM ubuntu

ADD . /app

RUN apt-get update

RUN apt-get install -y nodejs
RUN cd /app && npm install

CMD npm start
```


将变化频率一样的 RUN 指令合一

Docker镜像是分层的，类似于洋葱，它们都有很多层，为了修改内层，则需要将外面的层都删掉。Docker镜像有如下特性：

- Dockerfile中的每个指令都会创建一个新的镜像层。
- 镜像层将被缓存和复用。
- Dockerfile修改后，复制的文件变化了或者构建镜像时指定的变量不同了，对应的镜像层缓存就会失效。
- 某一层的镜像缓存失效之后，它之后的镜像层缓存都会失效。
- 镜像层是不可变的，如果在某一层中添加一个文件，然后在下一层中删除它，则镜像中依然会包含该文件，只是这个文件在Docker容器中不可见。

将变化频率一样的指令合并在一起，目的是为了更好的将镜像分层，避免带来不必要的成本。如本例中将node.js安装与npm模块安装放在一起的话，则每次修改源代码，都需要重新安装node.js，这显然不合适。

```
FROM ubuntu

ADD . /app

RUN apt-get update \
    && apt-get install -y nodejs \
    && cd /app \
    && npm install

CMD npm start
```

因此，正确的写法是这样的：

```
FROM ubuntu

RUN apt-get update && apt-get install -y nodejs
ADD . /app
RUN cd /app && npm install

CMD npm start
```

使用特定的标签

当镜像没有指定标签时，将默认使用latest标签。因此，FROM ubuntu指令等同于FROM ubuntu:latest。当镜像更新时，latest标签会指向不同的镜像，这时构建镜像有可能失败。

如下示例中使用16.04作为标签。

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y nodejs
ADD . /app
RUN cd /app && npm install

CMD npm start
```

删除多余文件

假设更新了apt-get源，下载解压并安装了一些软件包，它们都保存在“/var/lib/apt/lists/”目录中。

但是，运行应用时Docker镜像中并不需要这些文件。因此最好将它们删除，因为它会使Docker镜像变大。

示例Dockerfile中，删除“/var/lib/apt/lists/”目录中的文件。

```
FROM ubuntu:16.04

RUN apt-get update \
    && apt-get install -y nodejs \
    && rm -rf /var/lib/apt/lists/*

ADD . /app
RUN cd /app && npm install

CMD npm start
```

选择合适的基础镜像

在示例中，选择了ubuntu作为基础镜像。但是只需要运行node程序，没有必要使用一个通用的基础镜像，node镜像应该是更好的选择。

更好的选择是alpine版本的node镜像。alpine是一个极小化的Linux发行版，只有4MB，这让它非常适合作为基础镜像。

```
FROM node:7-alpine

ADD . /app
RUN cd /app && npm install

CMD npm start
```

设置 WORKDIR 和 CMD

WORKDIR指令可以设置默认目录，也就是运行RUN / CMD / ENTRYPOINT指令的地方。

CMD指令可以设置容器创建时执行的默认命令。另外，您应该将命令写在一个数组中，数组中每个元素为命令的每个单词。

```
FROM node:7-alpine

WORKDIR /app
ADD . /app
RUN npm install

CMD ["npm", "start"]
```

使用 ENTRYPOINT（可选）

ENTRYPOINT指令并不是必须的，因为它会增加复杂度。ENTRYPOINT是一个脚本，它会默认执行，并且将指定的命令作为其参数。它通常用于构建可执行的Docker镜像。

```
FROM node:7-alpine

WORKDIR /app
ADD . /app
RUN npm install

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


ENTRYPOINT 脚本中使用 exec

在前文的ENTRYPOINT脚本中，使用了exec命令运行node应用。不使用exec的话，则不能顺利地关闭容器，因为SIGTERM信号会被bash脚本进程吞没。exec命令启动的进程可以取代脚本进程，因此所有的信号都会正常工作。

优先使用 COPY

COPY指令非常简单，仅用于将文件拷贝到镜像中。ADD相对来讲复杂一些，可以用于下载远程文件以及解压压缩包。

```
FROM node:7-alpine

WORKDIR /app

COPY . /app
RUN npm install

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

合理调整 COPY 与 RUN 的顺序

将变化最少的部分放在Dockerfile的前面，这样可以充分利用镜像缓存。

示例中，源代码会经常变化，则每次构建镜像时都需要重新安装NPM模块，这显然不是希望看到的。因此可以先拷贝package.json，然后安装NPM模块，最后才拷贝其余的源代码。这样的话，即使源代码变化，也不需要重新安装NPM模块。

```
FROM node:7-alpine

WORKDIR /app

COPY package.json /app
RUN npm install
COPY . /app

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

设置默认的环境变量、映射端口和数据库逻辑卷

运行Docker容器时很可能需要一些环境变量。在Dockerfile设置默认的环境变量是一种很好的方式。另外，应该在Dockerfile中设置映射端口和数据库逻辑卷。示例如下：

```
FROM node:7-alpine

ENV PROJECT_DIR=/app

WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

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

ENV指令指定的环境变量在容器中可以使用。如果您只是需要指定构建镜像时的变量，您可以使用ARG指令。

使用 EXPOSE 暴露端口

EXPOSE指令用于指定容器将要监听的端口。因此，您应该为您的应用程序使用常见的端口。例如，提供Apache web服务的镜像应该使用EXPOSE 80，而提供MongoDB服务的镜像使用EXPOSE 27017。

对于外部访问，用户可以在执行docker run时使用一个标志来指示如何将指定的端口映射到所选择的端口。

```
FROM node:7-alpine

ENV PROJECT_DIR=/app

WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENV APP_PORT=3000
EXPOSE $APP_PORT

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

使用 VOLUME 管理数据库逻辑卷

VOLUME指令用于暴露任何数据库存储文件、配置文件或容器创建的文件和目录。强烈建议使用VOLUME来管理镜像中的可变部分和用户可以改变的部分。

下面示例中填写一个媒体目录。

```
FROM node:7-alpine

ENV PROJECT_DIR=/app

WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENV MEDIA_DIR=/media \
    APP_PORT=3000

VOLUME $MEDIA_DIR
EXPOSE $APP_PORT

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

使用 LABEL 设置镜像元数据

您可以给镜像添加标签来帮助组织镜像、记录许可信息、辅助自动化构建等。每个标签一行，由LABEL开头加上一个或多个标签对。

须知

如果您的字符串中包含空格，必须将字符串放入引号中或者对空格使用转义。如果字符串内容本身就包含引号，必须对引号使用转义。

```
FROM node:7-alpine
LABEL com.example.version="0.0.1-beta"
```

添加 HEALTHCHECK

运行容器时，可以指定--restart always选项。这样的话，容器崩溃时，docker daemon会重启容器。对于需要长时间运行的容器，这个选项非常有用。但是，如果容器的确在运行，但是不可用怎么办？使用HEALTHCHECK指令可以让Docker周期性的检查容器的健康状况。只需要指定一个命令，如果一切正常的话返回0，否则返回1。当请求失败时，curl --fail命令返回非0状态。示例如下：

```
FROM node:7-alpine
LABEL com.example.version="0.0.1-beta"

ENV PROJECT_DIR=/app
WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENV MEDIA_DIR=/media \
    APP_PORT=3000

VOLUME $MEDIA_DIR
EXPOSE $APP_PORT
HEALTHCHECK CMD curl --fail http://localhost:$APP_PORT || exit 1

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

编写.dockerignore 文件

.dockerignore的作用和语法类似于.gitignore，可以忽略一些不需要的文件，这样可以有效加快镜像构建时间，同时减少Docker镜像的大小。

构建镜像时，Docker需要先准备context，将所有需要的文件收集到进程中。默认的context包含Dockerfile目录中的所有文件，但是实际上，并不需要.git目录等内容。

示例如下：

```
.git/
```


2 创建 JDK8 基础镜像并上传至 SWR

场景概述

在项目中，我们通常基于相同的基础镜像创建镜像。本章节将以JDK8基础镜像为例，介绍如何在CCE节点上创建JDK8基础镜像并上传至SWR。

操作步骤

步骤1 购买一个CCE集群。

1. 登录**CCE控制台**。
2. 在购买CCE集群页面配置集群参数，详细请参考**创建集群参数配置**。
3. 等待集群创建成功。创建成功后在集群管理下会显示一个运行中的集群，且集群节点数量为0。

步骤2 创建CCE节点。

集群创建成功后，您还需要在集群中创建运行工作负载的节点。CCE节点默认安装了Linux操作系统和Docker。我们可以用它创建基础镜像。

在下面的步骤中，我们将以Centos7.6为例，详细介绍如何创建JDK8基础镜像，并将它上传到SWR。

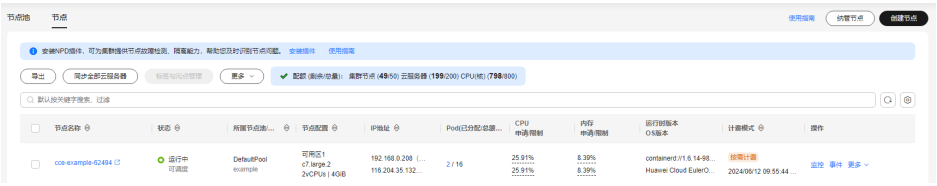
1. 登录**CCE控制台**。
2. 单击**1**中创建的集群，进入集群控制台。
3. 在左侧菜单栏选择节点管理，进入节点页签，单击右上角“创建节点”，在弹框中**配置节点的参数**。
4. 在网络配置中，选择“自动创建”1个弹性公网IP，带宽为5Mbit/s。

图 2-1 网络配置

5. 单击“下一步: 规格确认”。
6. 查看节点规格无误后，阅读页面上的使用说明，勾选“我已阅读并知晓上述使用说明和**《云容器引擎服务声明》**”，单击“提交”。

   等待节点创建成功。创建成功后在节点管理下会显示一个运行中的节点。

图 2-2 CCE 节点示例



步骤3 下载JDK软件包。

1. 节点创建成功后，单击节点名称，进入云服务器详情页。
2. 在云服务器详情页，单击右上角“远程登录”。
3. 选择一种登录方式，单击“登录”。
4. 以root用户登录弹性云服务器。
5. 新建一个目录image。

   mkdir image
6. 进入image目录。

   cd image
7. 下载JDK软件包。

   wget https://builds.openlogic.com/downloadJDK/openlogic-openjdk/8u352-b08/openlogic-openjdk-8u352-b08-linux-x64.tar.gz

步骤4 构建一个镜像。

1. 执行**vi dockerfile**命令，编写一个Dockerfile，并写入以下信息：

```
FROM centos                                                                    #使用centos作为基础镜像
RUN useradd -d /home/springboot -m springboot                                  #在工作目录下创建一个用户
ADD ./openlogic-openjdk-8u352-b08-linux-x64.tar.gz /home/springboot            #拷贝jdk软件包到镜像，并自动解压
RUN chown springboot:springboot /home/springboot/openlogic-openjdk-8u352-b08-linux-x64 -R
USER springboot                                                                #指定用户为springboot
ENV  JAVA_HOME=/home/springboot/openlogic-openjdk-8u352-b08-linux-x64           #设置环
```

```
境变量
ENV   PATH=$JAVA_HOME/bin:$PATH  \
    CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
WORKDIR /home/springboot/                                    #指定镜像的工作目录
```

2. 按**ESC**,输入**:wq**,保存Dockerfile，并退出编辑。
3. 执行下面的命令，构建一个镜像。

   docker build -t openjdk:8 .
4. 使用**docker images**命令，查看镜像是否构建成功。

   图 2-3 查看镜像是否创建成功

```
REPOSITORY                                                TAG       IMAGE ID       CREATED          SIZE
openjdk                                                   8         40a78b1544f1   28 hours ago     621MB
swr.cn-north-4.myhuaweicloud.com/devcloud-t/openjdk       8         40a78b1544f1   28 hours ago     621MB
centos                                                    7         eeb6ee3f44bd   15 months ago    204MB
swr.cn-north-4.myhuaweicloud.com/root/swr-demo-2048       latest    647c57f8354f   4 years ago      109MB
swr.cn-north-4.myhuaweicloud.com/testawa0306/swr-demo-2048   latest    647c57f8354f   4 years ago      109MB
```

步骤5 登录容器镜像服务控制台，并创建一个组织。

示例：这里我们创建一个名为testawa0306的组织。

步骤6 上传镜像到5的组织下。

1. 以root用户登录容器镜像服务控制台。
2. 为镜像打标签。

   *示例如下：**docker tag openjdk:8 swr.cn-north-4.myhuaweicloud.com/testawa0306/0penjdk:v8.8***
3. 上传镜像到步骤5的组织下。

docker push swr.cn-north-4.myhuaweicloud.com/testawa0306/0penjdk:v8.8

镜像上传成功后，我们可以在容器镜像服务控制台-“我的镜像”中找到刚刚上传成功的镜像。

步骤7 （可选）镜像上传成功后，您可以使用已上传的镜像在CCE中部署工作负载。

----结束

3 配置访问网络

3.1 概述

当我们使用云上的ECS或CCE节点作为安装容器引擎的客户端时，如何配置网络，才能保证上传下载的最佳运行速度？这里将介绍3种常见方案，用户可以根据自己的实际使用场景来选择。

3.2 内网访问

当您使用的安装容器引擎的客户端为云上的ECS或CCE节点，且机器与容器镜像仓库在同一区域时，上传下载镜像走内网链路。

您无需进行任何访问配置，直接访问SWR即可。

⚠ 注意

如果您通过配置"VPC终端节点"的方式来访问OBS，并在"VPC终端节点"设置了双端固定策略，需要将SWR存储容器镜像的OBS桶集群(可能存在一个或者多个)加入到允许访问策略范围中，否则可能导致下载镜像失败或者偶现失败，请**提交工单**获取不同局点SWR对应的OBS桶集群信息

3.3 公网访问

场景描述

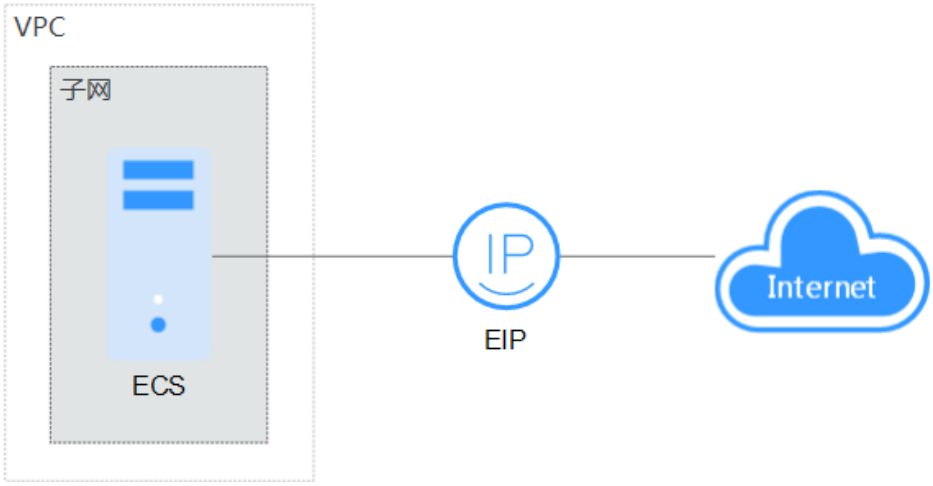
该场景下安装容器引擎的机器为云上的ECS或CCE节点，机器与容器镜像仓库**不在**同一区域，上传下载镜像走公网链路，机器需要绑定弹性公网IP。在此场景下，又分为2种情形。

- **单个ECS访问公网**
- **多个ECS访问公网**

单个 ECS 访问公网

当您的某台ECS需要主动访问公网，可以为ECS绑定EIP，即可实现公网访问。华为云提供多种计费方式（按需、按流量等）供您选择，无需使用时支持灵活解绑。

图 3-1 组网图



步骤1 登录管理控制台。

步骤2 单击管理控制台左上角的，选择区域和项目。

步骤3 单击，选择“计算 > 弹性云服务器”。

步骤4 在弹性云服务器列表中，选中要绑定弹性公网IP的机器，单击“操作”列下的“更多 > 网络/安全组 > 绑定弹性公网IP”。

步骤5 选择一个弹性公网IP，单击“确定”。

图 3-2 绑定弹性公网 IP

步骤6 完成绑定后，可以在云服务器列表页查看已绑定的弹性公网IP。

说明

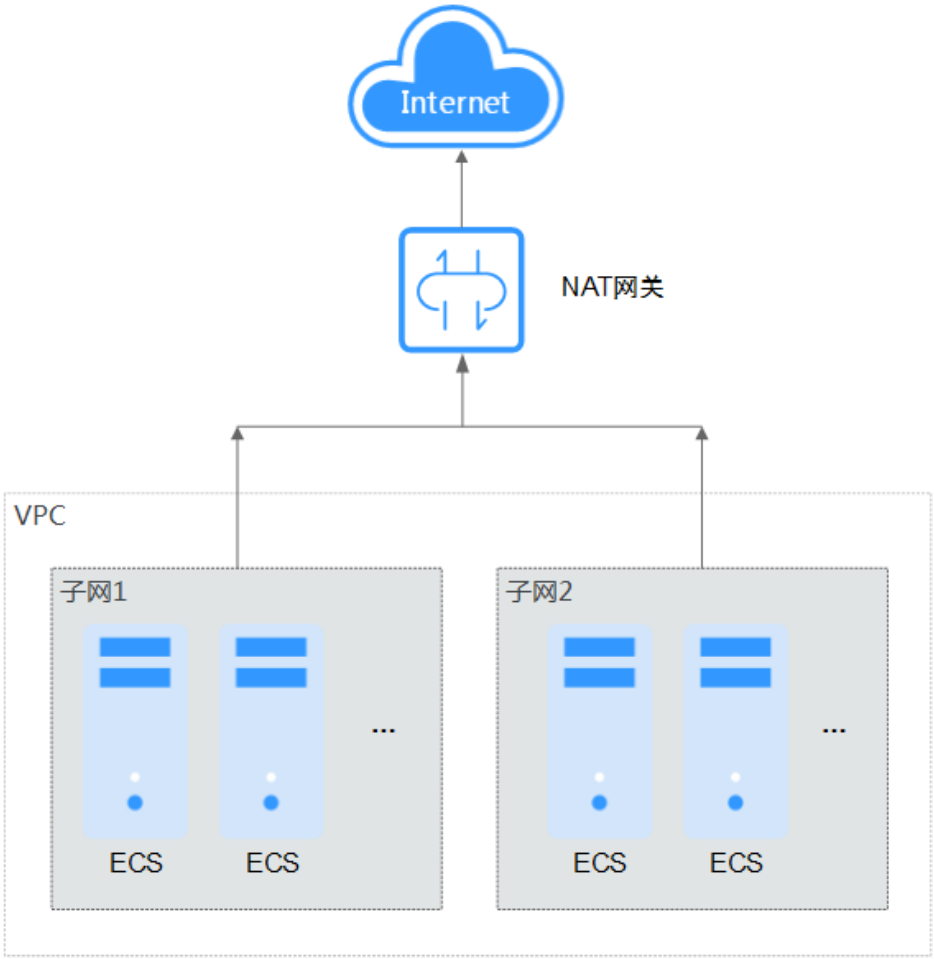
如果当前区域没有可用的弹性公网IP，则弹性公网IP列表为空，请购买弹性公网IP后重新执行绑定操作。

----结束

多个 ECS 访问公网

当您的VPC内ECS都有公网访问需求时，可以使用NAT网关服务，按子网配置SNAT规则，轻松构建VPC的公网出口。对比EIP访问公网，在未配置SNAT规则时，外部用户无法通过公网直接访问NAT网关的公网地址，保证了ECS的相对安全。

图 3-3 组网图



步骤1 按照**单个ECS访问公网**的操作步骤，为ECS绑定弹性公网IP。

步骤2 创建NAT网关，具体步骤详见**购买NAT网关**。

a. 登录管理控制台。

b. 在管理控制台左上角单击 ，选择区域和项目。

c. 在控制台首页，单击左上角的 ，在展开的列表中单击“网络 > NAT网关”。

d. 在NAT网关页面，单击右上角的“购买公网NAT网关”。

e. 根据界面提示配置参数。

图 3-4 购买公网 NAT 网关

步骤3 配置SNAT规则，为子网绑定弹性公网IP，具体请参见**添加SNAT规则**。

a. 登录管理控制台。

b. 在管理控制台左上角单击 ，选择区域和项目。

c. 在控制台首页，单击左上角的 ，在展开的列表中单击“网络 >NAT网关”。

d. 在NAT网关页面，单击需要添加SNAT规则的NAT网关名称。

e. 在SNAT规则页签中，单击“添加SNAT规则”。

f. 根据界面提示配置参数。

图 3-5 添加 SNAT 规则

----结束

注意

如果您通过配置本地/etc/hosts的方式访问OBS，需要将SWR存储容器镜像的OBS桶集群(可能存在一个或者多个)的域名配置到本地/etc/hosts，否则可能导致下载镜像失败或者偶现失败，请**提交工单**获取不同局点SWR对应的OBS桶集群信息

3.4 VPN/云专线访问

场景描述

该场景下用户本地数据中心或私有网络无法通过公网访问SWR。这种情况下可以通过云专线或VPN连接华为云VPC，通过专线或VPN打通用户本地数据中心到VPC间的网络，使用VPC终端节点访问SWR。

此场景仅适用于通过容器镜像服务上传镜像，当您需要下载镜像时，您还需要**配置访问OBS服务内网地址的终端节点**。

操作步骤

步骤1 **创建VPC。**详细步骤请参考**创建虚拟私有云和子网**。

步骤2 **创建云专线或VPN，使得数据中心能通过专线/VPN连接VPC。**

步骤3 **创建终端节点。**

a. 登录管理控制台。

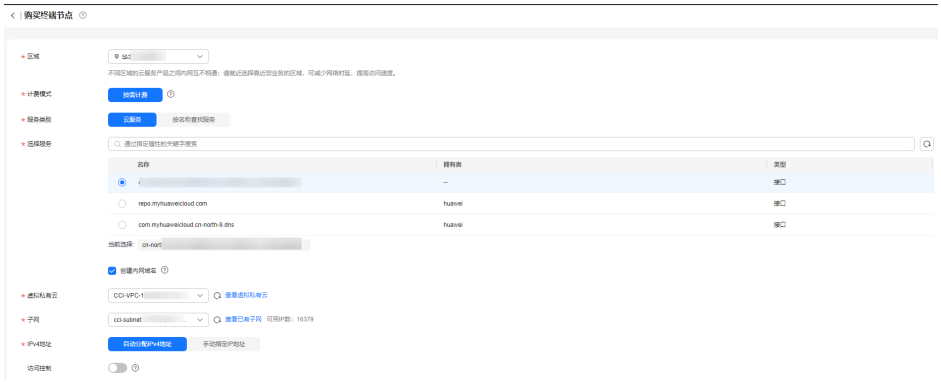
b. 在管理控制台左上角单击图标，选择区域和项目。

c. 在左侧导航栏，选择“网络”>“VPC终端节点”。

d. 在终端节点页面，单击“购买终端节点”。

e. 根据界面提示，配置相关参数。

图 3-6 购买终端节点



f. 单击“立即购买”。

g. 确认订单详情，单击“提交”。

步骤4 获取VPC内网访问IP及域名。

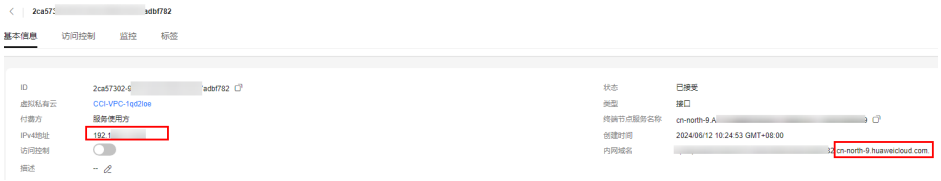
说明

华为云VPC内默认会自动加域名解析规则，不需要配置hosts；
非华为云节点需要配置hosts。

a. 进入终端节点列表页。

b. 找到购买成功的终端节点，单击该节点ID进入节点详情页。

c. 在节点详情页，我们可以查看到此终端节点的节点IP和内网域名。



步骤5 在用户本地数据中心节点上配置hosts，hosts地址由两部分组成，节点IP和内网域名。示例如下：

图 3-7 hosts 示例

⚠ 注意

本章节中的hosts示例：172.16.0.96 swr.cn-north-4.myhuaweicloud.com 仅为说明，请以实际获取到的节点IP和内网域名为准。

具体配置方式有2种：

Linux操作系统配置Hosts

自定义DNS Hosts

- **Linux操作系统配置步骤如下：**

1. 运行以下命令，打开/etc/hosts
   ```
   sudo vim /etc/hosts
   ```
2. 添加自定义域名：xx.xx.xx.xx swr.xx-xx.myhuaweicloud.com

   说明

   xx.xx.xx.xx为**步骤4**中查询到的节点IP，swr.xx-xx.myhuaweicloud.com为**步骤4**中查询到的内网域名。
3. 重启网络

```
sudo/etc/init.d/networking restart
```

- **自定义DNS Hosts**。

1. 参考**4**，查看该终端节点的“节点IP”。
2. 在用户本地数据中心的DNS服务器配置相应的DNS转发规则。

   不同操作系统中配置DNS转发规则的方法不同，具体操作请参考对应DNS软件的操作指导。

   本步骤以Linux操作系统，常见的DNS软件Bind为例介绍：

   a. 配置/etc/named.conf，新增zone：
   ```
   zone " swr.xx-xx.myhuaweicloud.com " IN {
     type master;
     file " /var/named/swr.xx-xx.myhuaweicloud.com.zone";
   };
   ```

   说明

   swr.xx-xx.myhuaweicloud.com为**步骤4**中查询到的内网域名。

   b. 配置自定义的域名到IP的正向解析配置。新建一个**a**中file对应的文件：/var/named/swr.xx-xx.myhuaweicloud.com.zone
   ```
   $TTL    604800
   @ IN      SOA       swr.xx-xx.myhuaweicloud.com. root.localhost. (
                             2         ; Serial
                        604800         ; Refresh
                         86400         ; Retry
                       2419200         ; Expire
                        604800 )       ; Negative Cache TTL
   ;
   @    IN   NS   swr.xx-xx.myhuaweicloud.com.
   swr.xx-xx.myhuaweicloud.com.    IN   A     xx.xx.xx.xx
   ```

   c. 重启服务。

   /sbin/service named restart

说明

- 您可以从**地区和终端节点**中查询不同区域SWR的Endpoint信息。
- 用户本地数据中心若无DNS服务器，需要将连接DNS服务的终端节点的节点IP增加到用户本地数据中心节点的/etc/resolv.conf文件中。
- swr.xx-xx.myhuaweicloud.com为**步骤4**中的节点IP。

步骤6 配置验证。

ping swr.xx -xx.myhuaweicloud.com，看返回结果。

步骤7 后续直接按照此固定域名: **swr.xx -xx.myhuaweicloud.com**访问SWR即可。

----结束

4 容器镜像迁移

4.1 方案概述

应用现状

随着容器化技术的发展，越来越多的企业使用容器代替了虚拟机完成应用的运行部署。目前许多企业选择自建Kubernetes集群，但是自建集群往往有着沉重的运维负担，需要运维人员自己配置管理系统和监控解决方案。企业自运维大批镜像资源，意味着要付出高昂的运维、人力、管理成本，且效率不高。

容器镜像服务支持Linux、ARM等多架构容器镜像托管。企业可以将镜像仓库迁移到容器镜像服务，节省运维成本。

如何把已有的镜像仓库平滑地迁移到容器镜像服务？这里将介绍4种常见的方案，用户可以根据自己的实际使用场景来选择。

迁移方案

表 4-1 迁移方案及适用场景对比

方案类型	适用场景	注意事项
使用docker命令迁移镜像至SWR	待迁移的镜像数量较少	• 依赖磁盘存储，需要及时进行本地镜像的清理，而且落盘形成多余的时间开销，难以胜任生产场景中大量镜像的迁移。 • 依赖docker程序，docker daemon对pull/push的并发数进行了严格的限制，没法进行高并发同步。 • 一些功能只能经过HTTP api进行操作，单纯使用docker cli 没法做到，使脚本变得复杂。

方案类型	适用场景	注意事项
使用image-syncer迁移镜像至SWR	待迁移的镜像数量庞大	• 支持多对多镜像仓库同步。 • 支持基于Docker Registry V2搭建的docker镜像仓库服务 (如Docker Hub、Quay、Harbor等)。 • 同步只通过内存和网络，不依赖磁盘存储，同步速度快。 • 增量同步, 经过对同步过的镜像blob信息落盘，不重复同步已同步的镜像。 • 并发同步，能够通过配置文件调整并发数。 • 自动重试失败的同步任务，能够解决大部分镜像同步中的网络抖动问题。 • 不依赖 docker 以及其余程序。
使用image-migrator迁移镜像至SWR	待迁移的镜像数量庞大	支持将基于Docker Registry v2搭建的Docker镜像仓库中的镜像迁移到华为云SWR中。
跨云harbor同步镜像至华为云SWR	部分客户存在多云场景，并且使用某一家云上的自建Harbor作为镜像仓库	仅支持 Harbor V1.10.5 及以上版本

4.2 使用 docker 命令迁移镜像至 SWR

场景描述

容器镜像服务提供了简便、易用的镜像托管和高效分发业务。当要迁移的镜像数量较少时，企业可以通过简单的docker pull、docker push命令行，将之前维护的镜像迁移到SWR上。

操作步骤

步骤1 从源仓库下载镜像。

使用docker pull命令下载镜像。

示例：**docker pull nginx:latest**

使用**docker images**命令查看是否下载成功。


```
# docker images
REPOSITORY                    TAG       IMAGE ID       CREATED         SIZE
nginx                         latest    22f2bf2e2b4f   5 hours ago     22.8MB
```

步骤2 将步骤1中下载的镜像上传到SWR。

1. 登录到目标端容器所在虚拟机，并登录SWR。详细步骤请参考客户端上传镜像。
2. 给镜像打标签。

   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]

   示例：

   docker tag nginx:v1 swr.cn-east-3.myhuaweicloud.com/cloud-develop/nginx:v1
3. 上传镜像至目标镜像仓库。

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

   示例：

   docker push swr.cn-east-3.myhuaweicloud.com/cloud-develop/nginx:v1
4. 终端显示如下信息，表明上传镜像成功。

   ```
   The push refers to repository [swr.cn-east-3.myhuaweicloud.com/cloud-develop/nginx:v1]
   fbce26647e70: Pushed
   fb04ab8effa8: Pushed
   8f736d52032f: Pushed
   009f1d338b57: Pushed
   678bbd796838: Pushed
   d1279c519351: Pushed
   f68ef921efae: Pushed
   v1: digest: sha256:0cdfc7910db531bfa7726de4c19ec556bc9190aad9bd3de93787e8bce3385f8d size: 1780
   ```

   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

4.3 使用 image-syncer 迁移镜像至 SWR

场景描述

当我们处理数量较少的镜像迁移任务时，使用命令行迁移就可以解决这个问题。但是实际生产中涉及到成千上百个镜像，几T的镜像仓库数据时，迁移过程就变得耗时很是漫长，甚至丢失数据。这时，我们可以使用开源镜像迁移工具image-syncer来处理这个任务。

操作步骤

步骤1 下载image-syncer到执行机上，解压并运行工具。

以v1.3.1版本为例，您也可以选择其他版本。

wget https://github.com/AliyunContainerService/image-syncer/releases/download/v1.3.1/image-syncer-v1.3.1-linux-amd64.tar.gz

tar -zvxf image-syncer-v1.3.1-linux-amd64.tar.gz

步骤2 创建镜像仓库的认证信息文件**auth.json**。

image-syncer支持基于Docker Registry V2搭建的docker镜像仓库，按格式填写即可。将源仓库及目标仓库认证信息写入，示例如下。

```
{
    "swr.××××.myhuaweicloud.com": {
        "username": "××××@F1I3Q·····",
        "password": "2fd4c869ea0·····"
    },
    "swr.××××.myhuaweicloud.com": {
        "username": "××××@4N3FA·····",
        "password": "f1c82b57855f9d35·····"
    }
```

其中swr.××××.myhuaweicloud.com 为镜像仓库地址，username、password可以在登录命令中获取，获取方法如下：

登录SWR控制台，在总览页面右上角单击“登录指令”，在弹出的窗口中获取登录指令，如下图所示。

图 4-1 登录指令

步骤3 创建同步镜像描述文件**images.json**。

如下示例，左边是源仓库的地址，右边是目的仓库地址。image-syncer还支持其他描述方式，具体请参见**README-zh_CN.md**。

```
{
"swr.cn-north-4.myhuaweicloud.com/org-ss/canary-consumer": "swr.cn-east-3.myhuaweicloud.com/dev-container/canary-consumer"

}
```

步骤4 执行如下命令将镜像迁移至SWR。

./image-syncer --auth=./auth.json --images=./images.json --namespace=dev-container --registry=swr.cn-east-3.myhuaweicloud.com --retries=3 --log=./log

表 4-2 命令行参数说明

参数	说明
--config	设置用户提供的配置文件路径，使用之前需要创建此文件，默认为当前工作目录下的config.json文件。这个参数与 --auth和--images 的作用相同，分解成两个参数可以更好地区分认证信息与镜像仓库同步规则。建议使用 --auth 和 --images。
--images	设置用户提供的镜像同步规则文件所在路径，使用之前需要创建此文件，默认为当前工作目录下的images.json文件。
--auth	设置用户提供的认证文件所在路径，使用之前需要创建此认证文件，默认为当前工作目录下的auth.json文件。
--log	打印出来的log文件路径，默认打印到标准错误输出，如果将日志打印到文件将不会有命令行输出，此时需要通过cat对应的日志文件查看。
--namespace	设置默认的目标namespace，当配置文件内一条images规则的目标仓库为空，并且默认registry也不为空时有效，可以通过环境变量DEFAULT_NAMESPACE设置，同时传入命令行参数会优先使用命令行参数值。
--proc	并发数，进行镜像同步的并发goroutine数量，默认为5。不建议修改该参数。
--retries	失败同步任务的重试次数，默认为2，重试会在所有任务都被执行一遍之后开始，并且也会重新尝试对应次数生成失败任务的生成。一些偶尔出现的网络错误比如io timeout、TLS handshake timeout，都可以通过设置重试次数来减少失败的任务数量。
--registry	设置默认的目标registry，当配置文件内一条images规则的目标仓库为空，并且默认namespace也不为空时有效，可以通过环境变量DEFAULT_REGISTRY设置，同时传入命令行参数会优先使用命令行参数值。

步骤5 迁移命令执行后，可登录目标镜像仓库，查看已迁移的镜像。

----结束

4.4 使用 image-migrator 迁移镜像至 SWR

为保证集群迁移后容器镜像可正常拉取，提升容器部署效率，建议您将自建镜像仓库迁移至华为云容器镜像服务（ SWR ）。

image-migrator是一个镜像迁移工具，能够自动将基于Docker Registry v2搭建的Docker镜像仓库中的镜像迁移到SWR中。

准备工作

在开始迁移之前，请确保您已准备了一台安装了kubectl的服务器，用于连接源集群和目标集群。该服务器需要至少拥有5GB左右的本地磁盘空间和≥8G的内存，以确保迁移工具可以正常运行，并存储相关数据，如源集群的采集数据和目标集群的推荐数据等。

迁移工具支持在Linux（x86、arm）、Windows环境中运行，因此您可以在这些操作系统中任选一种作为服务器的操作系统。

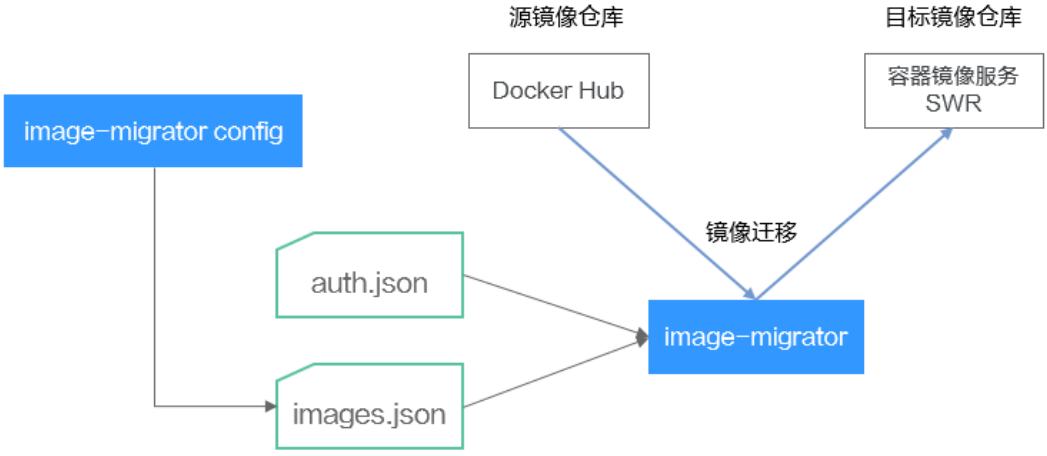
对于Linux操作系统来说，使用image-migrator前，需要运行chmod u+x 工具名 命令（例如chmod u+x image-migrator-linux-amd64），授予可执行权限。

表 4-3 image-migrator 工具包获取

image-migrator	image-migrator是一个镜像迁移工具，能够自动将基于Docker Registry v2搭建的Docker镜像仓库或第三方云镜像仓库中的镜像迁移到SWR中。	Linux x86：**https://ucs-migration.obs.cn-north-4.myhuaweicloud.com/toolkits/image-migrator-linux-amd64** Linux arm：**https://ucs-migration.obs.cn-north-4.myhuaweicloud.com/toolkits/image-migrator-linux-arm64** Windows：**https://ucs-migration.obs.cn-north-4.myhuaweicloud.com/toolkits/image-migrator-windows-amd64.exe**

image-migrator 工作原理

图 4-2 image-migrator 工作原理



使用image-migrator工具将镜像迁移到SWR时，需要准备两个文件，一个为镜像仓库访问权限文件“auth.json”，该文件中的两个对象分别为源镜像仓库和目标镜像仓库（即Registry）的账号和密码；另一个为镜像列表文件“images.json”，文件内容由多条镜像同步规则组成，每一条规则包括一个源镜像仓库（键）和一个目标镜像仓库（值）。将这两个文件准备好以后，放在image-migrator工具所在目录下，执行一条简单的命令，就可以完成镜像的迁移。关于两个文件的详细介绍如下：

- **“auth.json”文件**

  “auth.json”为镜像仓库访问权限文件，其中每个对象为一个registry的用户名和密码。通常源镜像仓库需要具有pull以及访问tags权限，目标镜像仓库需要拥有

push以及创建仓库权限。如果是匿名访问镜像仓库，则不需要填写用户名、密码等信息。“auth.json”文件的结构如下：

```
{
   "源镜像仓库地址": { },
   "目标镜像仓库地址": {
      "username": "xxxxxx",
      "password": "xxxxxx",
      "insecure": true
   }
}
```

其中，

- “源镜像仓库地址”和“目标镜像仓库地址”支持“registry”和“registry/namespace”两种形式，需要跟下述“images.json”中的registry或registry/namespace对应。images中被匹配到的url会使用对应用户名密码进行镜像同步，优先匹配“registry/namespace”的形式。

  目标镜像仓库地址如果为“registry”形式，可以从SWR控制台页面获取，具体方法如下：在“总览”页面单击右上角“登录指令”，登录指令末尾的域名即为SWR镜像仓库地址，例如swr.cn-north-4.myhuaweicloud.com。注意每个Region的地址不同，请切换到对应Region获取。如果为“registry/namespace”形式，还要将namespace替换为SWR的组织名称。
- username：（可选）用户名，values可以填写具体取值，也可以使用“${env}”或者“$env”类型的字符串引用环境变量。
- password：（可选）密码，values可以填写具体取值，也可以使用“${env}”或者“$env”类型的字符串引用环境变量。
- insecure：（可选）registry是否为http服务，如果是，insecure为true；默认是false。

说明

目标镜像仓库SWR的用户名形式为：区域项目名称@AK；密码为AK和SK经过加密处理后的登录密钥，详细指导请参考**获取长期有效登录指令**。

示例：

```
{
   "quay.io/coreos": { },
   "swr.cn-north-4.myhuaweicloud.com": {
      "username": "cn-north-4@RVHVMX******",
      "password": "cab4ceab4a1545***************",
      "insecure": true
   }
}
```

- **“images.json”文件**

该文件本质上是一个待迁移的镜像清单，由多条镜像同步规则组成，每一条规则包括一个源镜像仓库（键）和一个目标镜像仓库（值）。具体的要求如下：

a. 同步的最大单位是仓库（repository），不支持通过一条规则同步整个namespace以及registry。

b. 源仓库和目标仓库的格式与docker pull/push命令使用的镜像url类似（registry/namespace/repository:tag）。

c. 源仓库和目标仓库（如果目标仓库不为空字符串）都至少包含registry/namespace/repository。

d. 源仓库字段不能为空，如果需要将一个源仓库同步到多个目标仓库需要配置多条规则。

e. 目标仓库名可以和源仓库名不同，此时同步功能类似于：docker pull + docker tag + docker push。

f. 当源仓库字段中不包含tag时，表示将该仓库所有tag同步到目标仓库，此时目标仓库不能包含tag。

g. 当源仓库字段中包含tag时，表示只同步源仓库中的一个tag到目标仓库，如果目标仓库中不包含tag，则默认使用源tag。

h. 当目标仓库为空字符串时，会将源镜像同步到默认registry的默认namespace下，并且repository以及tag与源仓库相同，默认registry和默认namespace可以通过命令行参数以及环境变量配置。

示例如下：

```
{
    "quay.io/coreos/etcd:1.0.0": "swr.cn-north-4.myhuaweicloud.com/test/etcd:1.0.0",
    "quay.io/coreos/etcd": "swr.cn-north-4.myhuaweicloud.com/test/etcd",
    "quay.io/coreos/etcd:2.7.3": "swr.cn-north-4.myhuaweicloud.com/test/etcd"
}
```

我们提供了一个自动获取集群中工作负载正在使用的镜像的方法，即image-migrator工具的config子命令，具体用法请参见**image-migrator config使用方法**。得到images.json文件后，您还可以根据需要进行修改、添加或删除。

image-migrator 使用方法

说明

image-migrator工具支持在Linux（x86、arm）和Windows环境中运行，使用方法相似。本文将以Linux（x86）环境为例进行介绍。

若使用Linux（arm）或Windows环境，请将下述命令中的**image-migrator-linux-amd64**分别替换为**image-migrator-linux-arm64**或**image-migrator-windows-amd64.exe**。

在image-migrator工具所在目录下执行**./image-migrator-linux-amd64 -h**，可以查看image-migrator工具的使用方法。

- --auth：指定auth.json的路径，默认在image-migrator所在目录下。
- --images：指定images.json的路径，默认在image-migrator所在目录下。
- --log：指定image-migrator生成日志的路径，默认是image-migrator当前目录下的image-migrator.log。
- --namespace：默认的目标仓库的namespace，也就是说，如果images.json中没有指定目标仓库中的namespace，可以在执行迁移命令时指定。
- --registry：默认的目标仓库的registry，也就是说，如果images.json中没有指定目标仓库中的registry，可以在执行迁移命令时指定。
- --retries：迁移失败时的重试次数，默认为3。
- --workers：镜像搬迁的worker数量（并发数），默认是7。

```
$ ./image-migrator-linux-amd64 -h
A Fast and Flexible docker registry image images tool implement by Go.

Usage:
  image-migrator [flags]

Aliases:
  image-migrator, image-migrator

Flags:
      --auth string        auth file path. This flag need to be pair used with --images. (default "./auth.json")
  -h, --help               help for image-migrator
      --images string      images file path. This flag need to be pair used with --auth (default "./images.json")
      --log string         log file path (default "./image-migrator.log")
      --namespace string   default target namespace when target namespace is not given in the images
config file, can also be set with DEFAULT_NAMESPACE environment value
```



```
    --registry string    default target registry url when target registry is not given in the images config file, can also be set with DEFAULT_REGISTRY environment value
  -r, --retries int        times to retry failed tasks (default 3)
  -w, --workers int        numbers of working goroutines (default 7)

$./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test \
--registry=swr.cn-north-4.myhuaweicloud.com --retries=2
$ ./image-migrator
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

示例如下：

./image-migrator --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

该命令表示将“images.json”文件中的镜像迁移至“swr.cn-north-4.myhuaweicloud.com/test”镜像仓库下，迁移失败时可以重试2次，一次可以同时搬迁5个镜像。

image-migrator config 使用方法

image-migrator工具的config子命令可用于获取集群应用中使用的镜像，在工具所在目录下生成images.json。执行**./image-migrator-linux-amd64 config -h**命令可以查看config子命令的使用方法。

- -k, --kubeconfig：指定kubectl的kubeConfig位置，默认是$HOME/.kube/config。kubeConfig文件：用于配置对Kubernetes集群的访问，KubeConfig文件中包含访问注册kubernetes集群所需要的认证凭据以及Endpoint（访问地址），详细介绍可参见**Kubernetes文档**。
- -n, --namespaces：指定获取镜像的命名空间，多个命名空间用逗号分隔（如：ns1,ns2,ns3），默认是""，表示获取所有命名空间的镜像。
- -t, --repo：指定目标仓库的地址（registry/namespace）。

```
$ ./image-migrator-linux-amd64 config -h
generate images.json

Usage:
   image-migrator config [flags]

Flags:
   -h, --help               help for config
   -k, --kubeconfig string   The kubeconfig of k8s cluster's. Default is the $HOME/.kube/config. (default "/root/.kube/config")
   -n, --namespaces string   Specify a namespace for information collection. If multiple namespaces are specified, separate them with commas (,), such as ns1,ns2. default("") is all namespaces
   -t, --repo string         target repo,such as swr.cn-north-4.myhuaweicloud.com/test
```

示例如下：

- 指定一个命名空间

  ./image-migrator-linux-amd64 config -n default -t swr.cn-north-4.myhuaweicloud.com/test
- 指定多个命名空间

  ./image-migrator-linux-amd64 config -n default,kube-system -t swr.cn-north-4.myhuaweicloud.com/test
- 不指定命名空间（表示获取所有命名空间的镜像）

  ./image-migrator-linux-amd64 config -t swr.cn-north-4.myhuaweicloud.com/test

镜像迁移操作步骤

步骤1 准备镜像仓库访问权限文件：auth.json。

新建一个auth.json文件，并按照格式修改，如果是匿名访问仓库，则不需要填写用户名、密码等信息。将文件放置在image-migrator所在目录下。

示例：

```
{
    "quay.io/coreos": { },
    "swr.cn-north-4.myhuaweicloud.com": {
        "username": "cn-north-4@RVHVMX******",
        "password": "cab4ceab4a1545***************",
        "insecure": true
    }
}
```

详细的参数说明请参见“**auth.json**”**文件**。

步骤2 准备镜像列表文件：images.json。

1. 通过kubectl连接源集群。具体方法可参考**使用kubectl连接集群**。
2. 执行镜像迁移config子命令，生成images.json文件。

   您可以参考**image-migrator config使用方法**中的方法和示例，不指定命名空间，或者指定一个、多个命名空间来获取源集群应用中使用的镜像。
3. 根据需求调整images.json文件内容，但要遵循“**images.json**”**文件**中所讲的八项要求。

步骤3 镜像迁移。

您可以执行默认的**./image-migrator-linux-amd64**命令进行镜像迁移，也可以根据需要设置image-migrator的参数。

例如以下命令：

./image-migrator-linux-amd64 --workers=5 --auth=./auth.json --images=./images.json --namespace=test --registry=swr.cn-north-4.myhuaweicloud.com --retries=2

示例：

```
$ ./image-migrator-linux-amd64
Start to generate images tasks, please wait ...
Start to handle images tasks, please wait ...
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

步骤4 结果查看。

上述命令执行完毕后，回显如下类似信息：

```
Images(38) migration finished, 0 images tasks failed, 0 tasks generate failed
```

表示按照配置，成功将38个镜像迁移到SWR仓库中。

----结束

4.5 跨云 Harbor 同步镜像至华为云 SWR

场景描述

部分客户存在多云场景，并且使用某一家云上的自建Harbor作为镜像仓库。跨云Harbor同步镜像至SWR存在两种场景：

1. Harbor可以通过公网访问SWR，配置方法参见**公网访问场景**。
2. 通过专线打通Harbor到VPC间的网络，使用VPC终端节点访问SWR，配置方法参见**专线打通场景**。

背景知识

Harbor是VMware公司开源的企业级Docker Registry管理项目，在开源Docker Distribution能力基础上扩展了例如权限管理（RBAC）、镜像安全扫描、镜像复制等功能。Harbor目前已成为自建容器镜像托管及分发服务的首选。

公网访问场景

步骤1 Harbor上配置镜像仓库。

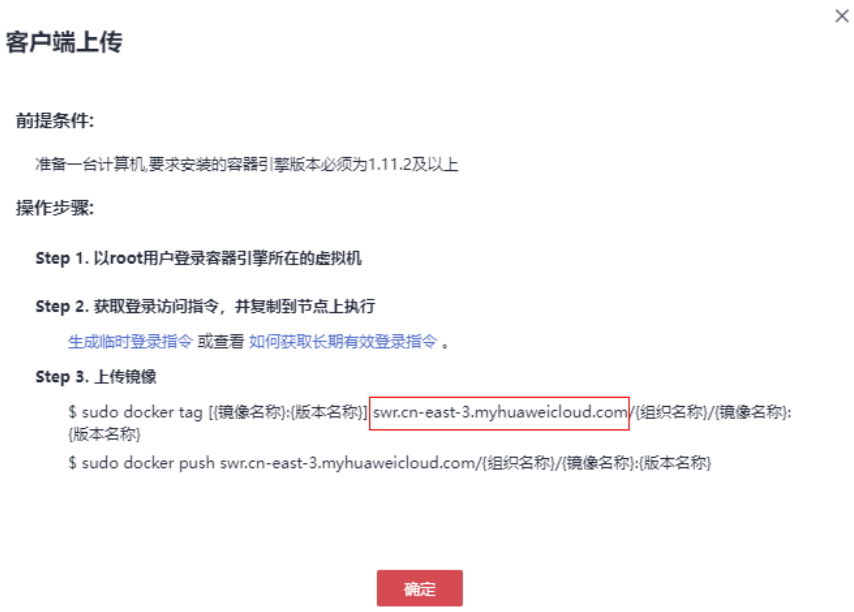
说明

Harbor在1.10.5以上版本，集成了华为云的SWR对接，只需要在目标（ENDPOINT）上选择就可以。本文以Harbor 2.4.1为例。

1. 新建目标。

2. 填写如下参数。

- 提供者：必须选“Huawei SWR”。
- 目标名：自定义。
- 目标URL：使用SWR的公网域名，格式为https://{*SWR镜像仓库地址*}，例如：https://swr.cn-east-3.myhuaweicloud.com。镜像仓库地址获取方法：登录容器镜像服务控制台，进入“我的镜像”，单击“客户端上传”，在弹出的页面即可查看SWR当前Region的镜像仓库地址。



- 访问ID：遵循SWR的长期有效的认证凭证规则，以“区域项目名称@[AK]”形式填写。
- 访问密码：遵循SWR的长期有效的认证凭证规则，需要用AK和SK来生成，详细说明请参考**获取长期有效登录指令**。
- 验证远程证书：建议**取消勾选**。

步骤2 配置同步规则。

1. 新建规则

2. 填写如下参数。

- 名称：自定义。
- 复制模式：选择“Push-based”，表示把镜像由本地Harbor推送到远端仓库。
- 源资源过滤器：根据填写的规则过滤Harbor上的镜像。
- 目标仓库：选择**步骤1**中创建的目标。
- 目标

  名称空间：填写SWR上的组织名称。

  仓库扁平化：用以在复制镜像时减少仓库的层级结构，建议选择“替换所有级”。假设Harbor仓库的层级结构为“library/nginx”，目标名称空间为dev-container，“替换所有级”对应的结果为：library/nginx -> dev-container/nginx。

- 触发模式：选择“手动”。
- 带宽：设置执行该条同步规则时的最大网络带宽，“-1”表示无限制。

步骤3 创建完成后，选中后单击“复制”即可完成同步。

----结束

专线打通场景

步骤1 配置VPC终端节点。

步骤2 获取VPC内网访问IP及域名（华为云VPC内默认会自动加域名解析规则，不需要配置hosts；非华为云节点需要配置hosts），可以在终端节点详情页的“内网域名”中查询。

步骤3 Harbor上配置镜像仓库。

说明

Harbor在1.10.5以上版本，集成了华为云的SWR对接，只需要在目标（ENDPOINT）上选择就可以。本文以Harbor 2.4.1为例。

1. 新建目标。

2. 填写如下参数。

- 提供者：必须选“Huawei SWR”。
- 目标名：自定义。
- 目标URL：**使用VPC终端节点中的内网域名**，必须以https开头，同时Harbor所在的容器内要配置域名映射。
- 访问ID：遵循SWR的长期有效的认证凭证规则，以“区域项目名称@[AK]”形式填写。
- 访问密码：遵循SWR的长期有效的认证凭证规则，需要用AK和SK来生成，详细说明请参考**获取长期有效登录指令**。
- 验证远程证书：必须**取消勾选**。

步骤4 配置同步规则。

1. 新建规则

2. 填写如下参数。

- 名称：自定义。
- 复制模式：选择“Push-based”，表示把镜像由本地Harbor推送到远端仓库。
- 源资源过滤器：根据填写的规则过滤Harbor上的镜像。
- 目标仓库：选择**步骤3**中创建的目标。
- 目标

  名称空间：填写SWR上的组织名称。

  仓库扁平化：用以在复制镜像时减少仓库的层级结构，建议选择“替换所有级”。假设Harbor仓库的层级结构为“library/nginx”，目标名称空间为dev-container，“替换所有级”对应的结果为：library/nginx -> dev-container/nginx。

- 触发模式：选择“手动”。
- 带宽：设置执行该条同步规则时的最大网络带宽，“-1”表示无限制。

步骤5 创建完成后，选中后单击“复制”即可完成同步。

----结束

