

# 华东师范大学软件工程学院实验报告

实验课程:	Cloud Computing	年 级:	2022 级	实验成绩:	
实验名称:	鲲鹏云容器实验	姓 名:	李鹏达	实验日期:	2024 年 12 月 11 日
实验编号:	No. 2	学 号:	10225101460	实验时间:	第 13 - 14 周

## 1 实验内容

在鲲鹏云容器中完成 Docker 主机的安装和配置、镜像的搜索和下载、容器生命周期的基本管理、容器网络的管理，并通过 Dockerfile 来构建 nginx 镜像，了解 Dockerfile 镜像构建过程。

## 2 实验关键步骤

### 2.1 购买实验资源

#### 2.1.1 创建虚拟私有云

登录华为云，在产品中选择“虚拟私有云 VPC”，创建虚拟私有云。  
填写以下基本信息：

- 地域：华北-北京四
- 名称：vpc-docker
- 可用区：可用区 1
- 子网名称：subnet-docker
- 子网网段：默认



图 1: 创建虚拟私有云

2.1.2 创建并配置安全组

展开网络控制台左侧列表的访问控制，选择“安全组”，点击“创建安全组”。配置模板选择为“通用 Web 服务器”，名称设置为“Sys-Webserver”，然后点击“确定”。默认放通 22、3389、80、443 端口和 ICMP 协议。

区域

华北-北京四

名称

Sys-Webserver

标签 (可选)

+ 添加新标签

描述 (可选)

0/255

配置规则

安全组下规则建议不超过50条，过多将影响性能。  
安全组规则对不同规格云服务器的生效情况不同，为了避免您的安全组规则不生效，请您添加规则前，单击[此处](#)了解详情。  
当源地址选择IP地址时，您可以在一个框内同时输入或者粘贴多个IP地址，不同IP地址以“,”隔开。一个IP地址生成一条安全组规则。  
入方向规则的源地址设置为0.0.0.0/0或::/0，表示允许或拒绝所有外部IP地址访问您的实例，如果将“22、3389、8848”等[高危端口](#)暴露到公网，可能导致网络

预设规则

通用Web服务器

入方向规则出方向规则

添加规则

图 2: 创建安全组

单击安全组名称“Sys-Webserver”，进入安全组规则配置界面。点击“入方向规则”，为安全起见，将“全部”放通删除，完成安全组配置。

<input type="checkbox"/>	1	允许	IPv4	TCP : 22	0.0.0.0/0	允许外部访问安全组...	2024/12/12 13:17:48 ...	修改 复制 删除
<input type="checkbox"/>	1	允许	IPv4	全部	Sys-Webserver	允许安全组内实例通...	2024/12/12 13:17:48 ...	修改 复制 删除
<input type="checkbox"/>	1	允许	IPv4	TCP : 3389	0.0.0.0/0	允许外部访问安全组...	2024/12/12 13:17:48 ...	修改 复制 删除
<input type="checkbox"/>	1	允许	IPv4	TCP : 80	0.0.0.0/0	允许外部访问安全组...	2024/12/12 13:17:48 ...	修改 复制 删除
<input type="checkbox"/>	1	允许	IPv4	TCP : 443	0.0.0.0/0	允许外部访问安全组...	2024/12/12 13:17:48 ...	修改 复制 删除
<input type="checkbox"/>	1	允许	IPv6	全部	Sys-Webserver	允许安全组内实例通...	2024/12/12 13:17:48 ...	修改 复制 删除

图 3: 配置安全组

2.1.3 购买弹性云服务器

选择产品 → 基础服务 → 弹性云服务器 ECS，然后点击“立即购买”。  
填写如下基础配置信息：

- 计费模式：按需计费
- 地域：华北-北京四
- 可用区：随机分配
- CPU 架构：鲲鹏计算
- 镜像：公共镜像 openEuler 20.03 64bit with ARM(40GB)
- 系统盘：普通 IO/高 IO | 40G
- 规格：鲲鹏通用计算增强型 | kc1.large.2 | 2vCPUs | 4GB

基础配置

计费模式

包年/包月

按需计费

竞价计费

按需计费实例不支持备案。[了解备案限制](#)

区域

华北-北京四

推荐区域 华北-北京四 华南-广州 华东-上海一 西南-贵阳一 华北-乌兰察布一

云服务器创建后无法更改区域；不同区域之间内网互不相通，请就近选择靠近您业务的区域，减少网络时延。[如何选择区域](#)

可用区

随机分配

可用区1

可用区2

可用区3

可用区7

随机至多可用区

图 4: 填写配置信息 (1)

实例

规格类型选型

业务场景选型

CPU架构

x86计算

鲲鹏计算

实例筛选

--请选择vCPUs--

--请选择内存--

请输入规格名称模糊搜索

隐藏售罄的规格

AI加速型

鲲鹏通用计算增强型

鲲鹏内存优化型

鲲鹏超高IO型

全部

kc1

kc2

特性

kc1搭载鲲鹏920处理器及25GE智能高速网卡，提供强劲鲲鹏算力和高性能网络，更好满足政府、互联网等各类企业对云上业务高性价比、安全可靠等诉求。

场景

政企金融/大数据/HPC/建站/电商

实例类型	规格名称	vCPUs	内存	CPU	基准 / 最大带宽	内网收发包	IPv6	规格参考价
<input type="radio"/> 鲲鹏通用计算增...	kc1.small.1	1vCPUs	1GiB	Huawei Kunp...	0.5 / 2 Gbit/s	20万PPS	否	¥0.12/小时
<input checked="" type="radio"/> 鲲鹏通用计算增...	kc1.large.2	2vCPUs	4GiB	Huawei Kunp...	0.8 / 3 Gbit/s	30万PPS	否	¥0.30/小时
<input type="radio"/> 鲲鹏通用计算增...	kc1.large.4	2vCPUs	8GiB	Huawei Kunp...	0.8 / 3 Gbit/s	30万PPS	是	¥0.41/小时
<input type="radio"/> 鲲鹏通用计算增...	kc1.xlarge.2	4vCPUs	8GiB	Huawei Kunp...	1.5 / 5 Gbit/s	50万PPS	否	¥0.60/小时

图 5: 填写配置信息 (2)

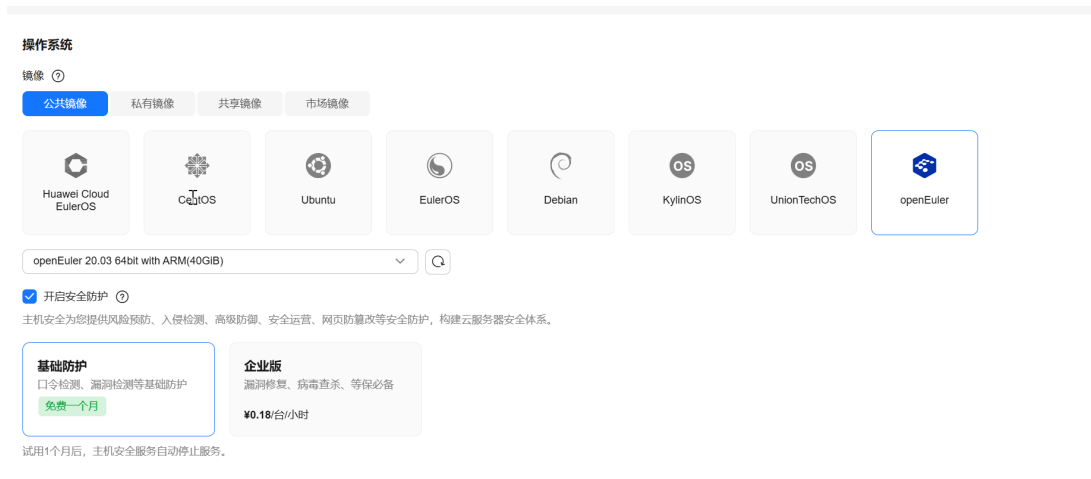


图 6: 填写配置信息 (3)

## 存储与备份

### 系统盘 ?

磁盘类型

系统盘大小(GiB)

高IO ▼ — 40 +

IOPS上限2,120, IOPS突发上限5,000 [高级设置](#)

⊕ 增加一块数据盘

您还可以挂载22块VBD磁盘或22块SCSI磁盘。

☐ 开启备份

备份可以帮助您在服务器故障时恢复数据，为了您的数据安全，建议您开启备份。

图 7: 填写配置信息 (4)

然后填写如下网络配置信息：

- 网络：选择已创建的网络和子网，如 vpc-docker 和 subnet-docker
- 安全组：Sys-Webserver
- 弹性公网 IP：现在购买
- 规格：全动态 BGP
- 计费方式：按带宽计费
- 带宽：5 Mbit/s

**网络**

虚拟私有云 <sup>①</sup>

vpc-docker(192.168.0.0/16)  [新建虚拟私有云](#)

主网卡

subnet-docker(192.168.0.0/24)  自动分配IP地址  可用私有IP数量250个

<sup>①</sup> 新增扩展网卡

您还可以增加 1 块网卡



源/目的检查 <sup>①</sup>

☒

图 8: 填写网络配置信息 (1)

**安全组**

选择安全组 <sup>①</sup>

Sys-Webserver(843e6257-55ad-4901-8201-50dbabcd7152)   [新建安全组](#)

请确保所选安全组已放行22端口（Linux SSH登录），3389端口（Windows远程登录）和 ICMP 协议（Ping）。[配置安全组规则](#)


展开安全组规则 

图 9: 填写网络配置信息 (2)

**公网访问**

弹性公网IP <sup>①</sup>

线路 <sup>①</sup>

☒ 不低于99.95%可用性保障

公网带宽 <sup>①</sup>

流量较大或较稳定的场景

流量小或流量波动较大场景

多业务流量错峰分布场景

指定带宽上限，按实际使用的出公网流量计费，与使用时间无关。

带宽大小

☒ 开启DDoS基础防护 <sup>①</sup>

释放行为

☐ 随实例释放

对于设置了随实例释放的弹性公网IP，将在删除云服务器同时执行删除。

图 10: 填写网络配置信息 (3)

接着，填写如下高级配置信息：

- 云服务器名称：ecs-docker-姓名全拼
- 登录凭证：密码
- 密码/确认密码：自行设置密码，要求 8 位以上且包含大小写字母、数字、特殊字符中三种以上字符

**云服务器管理**

云服务器名称  ☐ 允许重名

购买多台云服务器时，支持自动增加数字后缀命名或者自定义规则命名。[了解更多](#)

登录凭证

请牢记密码，如忘记密码可登录ECS控制台重置密码。

用户名

密码  确认密码

标签

如果您需要使用同一标签标识多种云资源，即所有服务均可在标签输入框下拉选择同一标签，建议在TMS中 [创建预定义标签](#)

您还可以添加10个标签。

图 11: 填写高级配置信息

点击创建云服务器，等待创建完成。

创建完成后，可以在控制台中查看创建的云服务器信息。复制弹性公网 IP 地址，用于后续登录。

名称/ID	监控	安全	状态	可...	规格/镜像	操作...	IP地址	计费模式	标签	操作
ecs-docker-lipengda dc8f4c98-8bc7-466f-81...			运行中	可用区2	2vCPUs   4GiB   kc1.larg... openEuler 20.03 64bit wi...	Linux	139.9.139.90... 192.168.0.20...	按需计费 2024/12/12 13:43:...	--	<a href="#">远程登录</a> <a href="#">更多</a>

图 12: 查看云服务器信息

使用 ssh 登录到创建的云服务器。

```
1 ssh root@139.9.139.90
```

可以看到出现 “welcome to Huawei Cloud Service”，表示登录成功。

```
pssh ~
> ssh root@139.9.139.90
The authenticity of host '139.9.139.90 (139.9.139.90)' can't be established.
ED25519 key fingerprint is SHA256:L5Q6Z/o+BlQCmAzdom8AedaGAZg3Yf9UbwjuDnJTOL4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '139.9.139.90' (ED25519) to the list of known hosts.

Authorized users only. All activities may be monitored and reported.
root@139.9.139.90's password:
Welcome to Huawei Cloud Service

Last failed login: Thu Dec 12 13:47:28 CST 2024 from 89.169.55.26 on ssh:notty
There was 1 failed login attempt since the last successful login.

Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time: Thu Dec 12 13:50:54 CST 2024

System load: 0.00
Processes: 144
Memory used: 10.5%
Swap used: 0.0%
Usage On: 9%
IP address: 192.168.0.206
Users online: 1
```

图 13: 登录到云服务器

检查内核版本。

```
1 uname -r
```

```
[root@ecs-docker-lipengda ~]# uname -r
4.19.90-2110.8.0.0119.oe1.aarch64
```

图 14: 检查内核版本

移除旧版本 docker。

```
1 yum remove docker docker-client docker-client-latest docker-common docker-latest
  docker-latest-logrotate docker-logrotate docker-selinux docker-engine-
  selinux docker-engine
```

```
[root@ecs-docker-lipengda ~]# yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-selinux docker-engine-selinux docker-engine
No match for argument: docker
No match for argument: docker-client
No match for argument: docker-client-latest
No match for argument: docker-common
No match for argument: docker-latest
No match for argument: docker-latest-logrotate
No match for argument: docker-logrotate
No match for argument: docker-selinux
No match for argument: docker-engine-selinux
No match for argument: docker-engine
No packages marked for removal.
Dependencies resolved.
Nothing to do.
Complete!
```

图 15: 移除旧版本 docker

安装 Docker 依赖工具。

```
1 yum install -y device-mapper-persistent-data lvm2
```

```
[root@ecs-docker-lipengda ~]# yum install -y device-mapper-persistent-data lvm2
Last metadata expiration check: 0:04:42 ago on Thu 12 Dec 2024 01:54:12 PM CST.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
<b>Installing:</b>				
lvm2	aarch64	8:2.02.187-3.oe1	update	1.5 M
thin-provisioning-tools	aarch64	0.7.6-6.oe1	update	353 k
<b>Installing dependencies:</b>				
device-mapper-event	aarch64	8:1.02.170-3.oe1	update	42 k
libaio	aarch64	0.3.112-1.oe1	update	22 k
<b>Transaction Summary</b>				
Install 4 Packages				
Total download size: 1.9 M				
Installed size: 8.0 M				
Downloading Packages:				
(1/4): libaio-0.3.112-1.oe1.aarch64.rpm			489 kB/s   22 kB	00:00
(2/4): device-mapper-event-1.02.170-3.oe1.aarch64.rpm			613 kB/s   42 kB	00:00
(3/4): lvm2-2.02.187-3.oe1.aarch64.rpm			15 MB/s   1.5 MB	00:00
(4/4): thin-provisioning-tools-0.7.6-6.oe1.aarch64.rpm			6.1 MB/s   353 kB	00:00
<b>Total</b>			<b>18 MB/s   1.9 MB</b>	<b>00:00</b>
Running transaction check				

图 16: 安装 Docker 依赖工具

## 2.2 Docker 的安装和配置

### 2.2.1 安装 Docker

安装 Docker。

```
1 yum -y install docker --nogpgcheck
```



```
[root@ecs-docker-lipengda ~]# yum -y install docker --nogpgcheck
Last metadata expiration check: 0:11:36 ago on Thu 12 Dec 2024 01:54:12 PM CST.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing: docker-engine	aarch64	18.09.0-202.0e1	update	35 M
Installing dependencies: libcgroup	aarch64	0.41-23.0e1	OS	94 k

```
Transaction Summary
Install 2 Packages

Total size: 35 M
Installed size: 173 M
Downloading Packages:
[SKIPPED] libcgroup-0.41-23.0e1.aarch64.rpm: Already downloaded
[SKIPPED] docker-engine-18.09.0-202.0e1.aarch64.rpm: Already downloaded
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                1/1
  Running scriptlet: libcgroup-0.41-23.0e1.aarch64 1/2
  Installing      : libcgroup-0.41-23.0e1.aarch64 1/2
  Running scriptlet: libcgroup-0.41-23.0e1.aarch64 1/2
  Installing      : docker-engine-18.09.0-202.0e1.aarch64 2/2
  Running scriptlet: docker-engine-18.09.0-202.0e1.aarch64 2/2
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.

  Verifying      : libcgroup-0.41-23.0e1.aarch64 1/2
  Verifying      : docker-engine-18.09.0-202.0e1.aarch64 2/2

Installed:
  docker-engine-18.09.0-202.0e1.aarch64          libcgroup-0.41-23.0e1.aarch64

Complete!
```

图 17: 安装 Docker

启动 Docker 后台服务。

```
1 systemctl start docker
```

### 2.2.2 配置镜像加速

在华为云所有服务 → 容器 → 容器镜像服务 SWR → 镜像资源 → 镜像中心 → 镜像加速器，复制加速器地址。

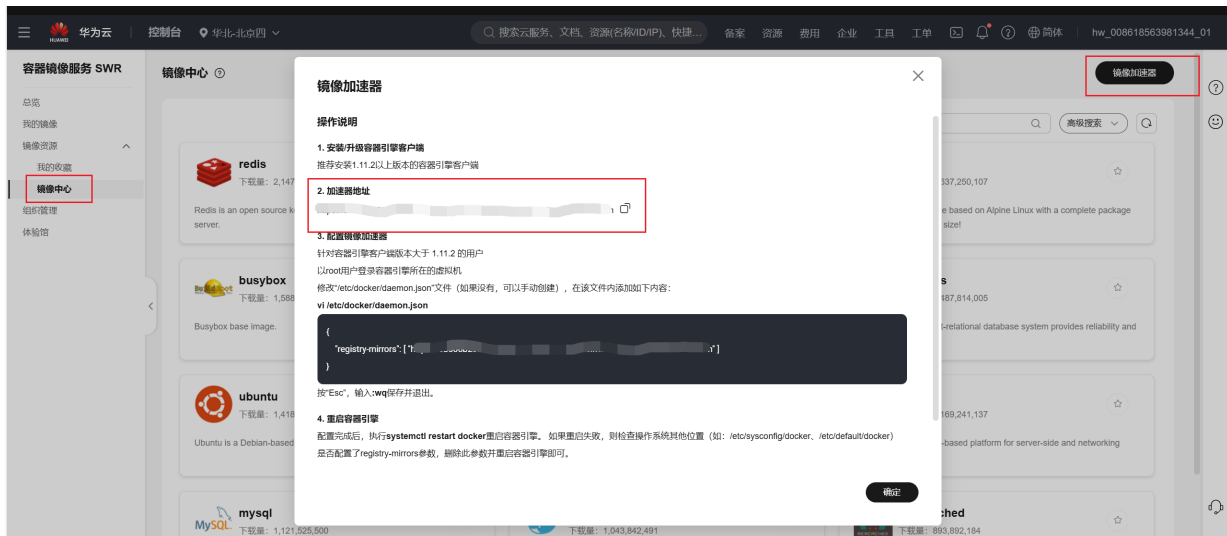


图 18: 复制镜像加速器地址

修改 Docker 配置文件。

```
1 vi /etc/docker/daemon.json
```

添加如下内容：

```
1 {
2     "registry-mirrors": ["加速器地址"]
3 }
```

保存退出后，重启 Docker 服务。

```
1 systemctl daemon-reload
2 systemctl restart docker
```

### 2.2.3 测试 Docker

测试运行 hello-world 镜像。

```
1 docker run hello-world
```

```
[root@ecs-docker-lipengda ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
478afc919002: Pull complete
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (arm64v8)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

图 19: 运行 hello-world 镜像

查看下载的 hello-world 镜像。

```
1 docker images
```

```
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest             ee301c921b8a       19 months ago      9.14kB
```

图 20: 查看下载的 hello-world 镜像

## 2.3 镜像的基本操作

### 2.3.1 获取镜像

下载 nginx 镜像

```
1 docker pull nginx
```

```
[root@ecs-docker-lipengda ~]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
bb3f2b52e6af: Pull complete
e4bc5c1a6721: Pull complete
e93f7200eab8: Pull complete
1bd52ec2c0cb: Pull complete
411a98463f95: Pull complete
ad5932596f78: Pull complete
df25b2e5edb3: Pull complete
Digest: sha256:fb197595ebe76b9c0c14ab68159fd3c08bd067ec62300583543f0ebda353b5be
Status: Downloaded newer image for nginx:latest
```

图 21: 下载 nginx 镜像

### 2.3.2 查询及删除镜像

查询已经下载的镜像

```
1 docker images
```

或

```
1 docker image ls
```

```
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest             bdf62fd3a32f       2 weeks ago        197MB
hello-world         latest             ee301c921b8a       19 months ago      9.14kB
```

图 22: 查询已下载的镜像

查询部分镜像

```
1 docker image ls nginx
```

```
[root@ecs-docker-lipengda ~]# docker image ls nginx
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest             bdf62fd3a32f       2 weeks ago        197MB
```

图 23: 查询部分镜像

查看镜像的大小

```
1 docker system df
```

```
[root@ecs-docker-lipengda ~]# docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	2	1	197.1MB	197.1MB (99%)
Containers	1	0	0B	0B
Local Volumes	0	0	0B	0B
Build Cache	0	0	0B	0B

图 24: 查看镜像的大小

通过短 ID 或完整 ID 删除镜像

```
1 docker images
2 docker rmi <ID>
```

```
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx          latest    bdf62fd3a32f   2 weeks ago    197MB
hello-world    latest    ee301c921b8a   19 months ago  9.14kB
[root@ecs-docker-lipengda ~]# docker rmi bdf
Untagged: nginx:latest
Untagged: nginx@sha256:fb197595ebe76b9c0c14ab68159fd3c08bd067ec62300583543f0ebda353b5be
Deleted: sha256:bdf62fd3a32f1209270ede068b6e08450dfe125c79b1a8ba8f5685090023bf7f
Deleted: sha256:bf74a794f54dda4a120c2341b9e3eeca19ab423649909edf66090bcab8a007
Deleted: sha256:d8ab2f8a77485ff82666c0b49a3f098e35643462313f391abe77c2ef0fdcfcfe
Deleted: sha256:c95a6c3fcab6af748cbf95a166ab17d1d00a87fe03b47181f72996b419da5693
Deleted: sha256:ad440851c8e2f92041d499b40dd0aa17fa33d9c8b2e1eec66581fe275b53bc32
Deleted: sha256:1555b493613f3dc11dac1ea991d968c2e068627ba42b33cf8b86702c15f78ff2
Deleted: sha256:bdec05ec0f48beadb288522556f5650baea79e24b9ca9df5424d06c5a328a8f
Deleted: sha256:3e620c160447d1acff162610a533282fc64863123cba28ce40eaf98c17dde780
```

图 25: 删除镜像

通过仓库名 + 标签删除镜像，如果删除的镜像已经产生了容器实例，不管容器实例是否启动都会提示无法删除，因为镜像被占用。这时需要先删除容器实例或添加删除参数 **-f** 强制删除。

```
[root@ecs-docker-lipengda ~]# docker image rm hello-world
Error response from daemon: conflict: unable to remove repository reference "hello-world" (must force) - container 5107c
ee39eab is using its referenced image ee301c921b8a
[root@ecs-docker-lipengda ~]# docker image rm hello-world -f
Untagged: hello-world:latest
Untagged: hello-world@sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
Deleted: sha256:ee301c921b8aad002973b2e0c3da17d701dcd994b606769a7e6eaa100b81d44
```

图 26: 删除镜像

## 2.4 容器的基本操作

### 2.4.1 容器的创建与启停

创建一个基于 httpd 镜像的新容器。若主机中没有对应镜像，将会从 docker Hub 中拉取最新镜像。

```
1 docker create httpd
```

```
[root@ecs-docker-lipengda ~]# docker create httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
bb3f2b52e6af: Pull complete
000c42d1b927: Pull complete
4f4fb700ef54: Pull complete
9c4a5ab2b764: Pull complete
5ad3d4ab6f84: Pull complete
08cd2d3f3c68: Pull complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
af3ed90ccf60e11e9ae811d0b13daf772a98b77527de1a93a23a0049ac9f34cf
```

图 27: 创建容器

查看容器信息

```
1 docker ps -a
```

```
[root@ecs-docker-lipengda ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
af3ed90ccf60   httpd     "httpd-foreground"      2 minutes ago Created              thirsty_mestorf
5107cee39eab   ee301c921b8a  "/hello"                4 days ago    Exited (0) 4 days ago    awesome_lamport
```

图 28: 查看容器信息

可以看到容器 ID 为 af3ed90ccf60, 名称为 thirsty\_mestorf  
根据显示的容器 ID 或容器名称启动容器

```
1 docker start af3ed90ccf60
2 docker ps -a
```

```
[root@ecs-docker-lipengda ~]# docker start af3ed90ccf60
af3ed90ccf60
[root@ecs-docker-lipengda ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
af3ed90ccf60   httpd     "httpd-foreground"      7 minutes ago Up 4 seconds    80/tcp        thirsty_mestorf
5107cee39eab   ee301c921b8a  "/hello"                4 days ago    Exited (0) 4 days ago    awesome_lamport
```

图 29: 启动容器

停止容器运行

```
1 docker stop af3ed90ccf60
2 docker ps -a
```

```
[root@ecs-docker-lipengda ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
af3ed90ccf60   httpd     "httpd-foreground"      7 minutes ago Exited (0) 2 seconds ago              thirsty_mestorf
5107cee39eab   ee301c921b8a  "/hello"                4 days ago    Exited (0) 4 days ago    awesome_lamport
```

图 30: 停止容器

重启容器

- 1 docker restart af3ed90ccf60
- 2 docker ps -a

```
[root@ecs-docker-lipengda ~]# docker restart af3ed90ccf60
af3ed90ccf60
[root@ecs-docker-lipengda ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af3ed90ccf60	httpd	"httpd-foreground"	8 minutes ago	Up 1 second	80/tcp	thirsty_mestorf
5107cee39eab	ee301c921b8a	"/hello"	4 days ago	Exited (0) 4 days ago		awesome_lamport

图 31: 重启容器

#### 暂停容器

- 1 docker pause af3ed90ccf60
- 2 docker ps -a

```
[root@ecs-docker-lipengda ~]# docker pause af3ed90ccf60
af3ed90ccf60
[root@ecs-docker-lipengda ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af3ed90ccf60	httpd	"httpd-foreground"	9 minutes ago	Up About a minute (Paused)	80/tcp	thirsty_mestorf
5107cee39eab	ee301c921b8a	"/hello"	4 days ago	Exited (0) 4 days ago		awesome_lamport

图 32: 暂停容器

#### 恢复暂停的容器

- 1 docker unpause af3ed90ccf60
- 2 docker ps -a

```
[root@ecs-docker-lipengda ~]# docker unpause af3ed90ccf60
af3ed90ccf60
[root@ecs-docker-lipengda ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af3ed90ccf60	httpd	"httpd-foreground"	11 minutes ago	Up 2 minutes	80/tcp	thirsty_mestorf
5107cee39eab	ee301c921b8a	"/hello"	4 days ago	Exited (0) 4 days ago		awesome_lamport

图 33: 恢复暂停的容器

#### 强制停止容器

- 1 docker kill af3ed90ccf60
- 2 docker ps -a

```
[root@ecs-docker-lipengda ~]# docker kill af3ed90ccf60
af3ed90ccf60
[root@ecs-docker-lipengda ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af3ed90ccf60	httpd	"httpd-foreground"	12 minutes ago	Exited (137) 1 second ago		thirsty_mestorf
5107cee39eab	ee301c921b8a	"/hello"	4 days ago	Exited (0) 4 days ago		awesome_lamport

图 34: 强制停止容器

启动容器，给容器重新命名

```
1 docker start af3ed90ccf60
2 docker ps -a
3 docker rename af3ed90ccf60 myhttpd
4 docker ps -a
```

```
[root@ecs-docker-lipengda ~]# docker start af3ed90ccf60
af3ed90ccf60
[root@ecs-docker-lipengda ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af3ed90ccf60	httpd	"httpd-foreground"	14 minutes ago	Up 1 second	80/tcp	thirsty_mestorf
5107cee39eab	ee301c921b8a	"/hello"	4 days ago	Exited (0) 4 days ago		awesome_lamport

```
[root@ecs-docker-lipengda ~]# docker rename af3ed90ccf60 myhttpd
[root@ecs-docker-lipengda ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af3ed90ccf60	httpd	"httpd-foreground"	14 minutes ago	Up 15 seconds	80/tcp	myhttpd
5107cee39eab	ee301c921b8a	"/hello"	4 days ago	Exited (0) 4 days ago		awesome_lamport

图 35: 给容器重新命名

## 2.4.2 容器的运行

运行一个新容器，该容器基于 ubuntu:14.04。

```
1 docker run ubuntu:14.04 /bin/echo 'Hello world'
```

```
[root@ecs-docker-lipengda ~]# docker run ubuntu:14.04 /bin/echo 'Hello world'
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
d1a5a1e51f25: Pull complete
75f8eea31a63: Pull complete
a72d031efbfb: Pull complete
Digest: sha256:64483f3496c1373bfd55348e88694d1c4d0c9b660dee6bfe5e12f43b9933b30
Status: Downloaded newer image for ubuntu:14.04
Hello world
```

图 36: 运行容器

下面的命令则启动一个 bash 终端，允许用户进行交互。

```
1 docker run -it ubuntu:14.04 /bin/bash
```

执行一些命令

```
1 pwd
2 ls
```

退出容器

```
1 exit
```



```
[root@ecs-docker-lipengda ~]# docker run -it ubuntu:14.04 /bin/bash
root@8c9a4a6e6707:/# pwd
/
root@8c9a4a6e6707:/# ls
bin boot dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
root@8c9a4a6e6707:/# exit
exit
[root@ecs-docker-lipengda ~]#
```

图 37: 运行容器启动一个 bash 终端

使用 **-d** 参数，在后台运行容器

```
1 docker run ubuntu:14.04 /bin/sh -c "while true; do echo hello world; sleep 1;
  done"
2 docker run -d ubuntu:14.04 /bin/sh -c "while true; do echo hello world; sleep 1;
  done"
```

```
[root@ecs-docker-lipengda ~]# docker run ubuntu:14.04 /bin/sh -c "while true; do echo hello world; sleep 1; done"
hello world
hello world
hello world
hello world
hello world
^C[root@ecs-docker-lipengda ~]# docker run -d ubuntu:14.04 /bin/sh -c "while true; do echo hello world; sleep 1; done"
caa86cb4a26b5e7d59cc24b7d40ed4abcf36f7950f6dc4ef9b6a6099bf33e720
[root@ecs-docker-lipengda ~]#
```

图 38: 在后台运行容器

## 获取容器的日志

```
1 docker logs caa86cb4a26b
```

[illegible]

图 39: 获取容器的日志

### 2.4.3 进入容器

某些时候需要进入容器进行操作，可以使用 `docker attach` 命令或 `docker exec` 命令。

启动一个容器

```
1 docker run -dit ubuntu:14.04
2 docker ps
```

```
[root@ecs-docker-lipengda ~]# docker run -dit ubuntu:14.04
43c7994a80316c003d68dad2cecb1962ad97d27c6d56b82cb118494e087ded8c
[root@ecs-docker-lipengda ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
43c7994a8031	ubuntu:14.04	"/bin/bash"	7 seconds ago	Up 6 seconds		peaceful_jang
af3ed90ccf60	httpd	"httpd-foreground"	39 minutes ago	Up 25 minutes	80/tcp	myhttpd

图 40: 启动一个容器

使用 attach 命令，直接进入容器启动命令的终端。

```
1 docker attach 43c7994a8031
```

执行一些命令

```
1 ps
2 exit
```

```
[root@ecs-docker-lipengda ~]# docker attach 43c7994a8031
root@43c7994a8031:/# ps
  PID TTY          TIME CMD
    1 pts/0    00:00:00 bash
   17 pts/0    00:00:00 ps
root@43c7994a8031:/# exit
exit
[root@ecs-docker-lipengda ~]#
```

图 41: 使用 attach 命令进入容器

启动一个新容器

```
1 docker run -dit ubuntu:14.04
2 docker ps
```

```
[root@ecs-docker-lipengda ~]# docker run -dit ubuntu:14.04
684f262c71081dabca7af8b59101b2f0d7e6dac32bf27803184f7a633b09b2b0
[root@ecs-docker-lipengda ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
684f262c7108	ubuntu:14.04	"/bin/bash"	5 seconds ago	Up 4 seconds		optimistic_neumann
af3ed90ccf60	httpd	"httpd-foreground"	44 minutes ago	Up 30 minutes	80/tcp	myhttpd

图 42: 启动一个新容器

通过 docker exec 进入容器

```
1 docker exec -it 684f262c7108 bash
```

执行一些命令

```
1 ps
2 exit
```

```
[root@ecs-docker-lipengda ~]# docker exec -it 684f262c7108 bash
root@684f262c7108:/# ps
  PID TTY          TIME CMD
   17 pts/1    00:00:00 bash
   32 pts/1    00:00:00 ps
root@684f262c7108:/# exit
exit
```

图 43: 通过 docker exec 进入容器

#### 2.4.4 删除容器

使用 `docker rm` 来删除一个处于终止状态的容器。若容器没有退出则无法删除，需要先停止容器。

```
1 docker ps
2 docker rm <ID>
3 docker stop <ID>
4 docker rm <ID>
5 docker ps
```

```
[root@ecs-docker-lipengda ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
00bf74e7a1c3   ubuntu:14.04   "/bin/sh -c 'while t..." 3 seconds ago  Up 2 seconds  80/tcp       adoring_ritchie
684f262c7108   ubuntu:14.04   "/bin/bash"              7 minutes ago  Up 7 minutes  80/tcp       optimistic_neumann
af3ed90ccf60   httpd          "httpd-foreground"       About an hour ago  Up 37 minutes  80/tcp       myhttpd
[root@ecs-docker-lipengda ~]# docker rm 684f262c7108
Error response from daemon: You cannot remove a running container 684f262c71081dabca7af8b59101b2f0d7e6dac32bf27803184f7a633b09b2b0. Stop the container before attempting removal or force remove
[root@ecs-docker-lipengda ~]# docker stop 684f262c7108
684f262c7108
[root@ecs-docker-lipengda ~]# docker rm 684f262c7108
684f262c7108
[root@ecs-docker-lipengda ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
00bf74e7a1c3   ubuntu:14.04   "/bin/sh -c 'while t..." 28 seconds ago  Up 27 seconds  80/tcp       adoring_ritchie
af3ed90ccf60   httpd          "httpd-foreground"       About an hour ago  Up 38 minutes  80/tcp       myhttpd
```

图 44: 删除容器

使用 `docker rm -f` 来删除一个处于运行状态的容器。

```
1 docker ps
2 docker rm -f <ID>
3 docker ps
```

```
[root@ecs-docker-lipengda ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
00bf74e7a1c3   ubuntu:14.04   "/bin/sh -c 'while t..." 28 seconds ago  Up 27 seconds  80/tcp       adoring_ritchie
af3ed90ccf60   httpd          "httpd-foreground"       About an hour ago  Up 38 minutes  80/tcp       myhttpd
[root@ecs-docker-lipengda ~]# docker rm -f 00bf74e7a1c3
00bf74e7a1c3
[root@ecs-docker-lipengda ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
af3ed90ccf60   httpd          "httpd-foreground"       About an hour ago  Up 42 minutes  80/tcp       myhttpd
```

图 45: 删除容器

删除所有已终止的容器。

```
1 docker rm -v $(docker ps -aq -f status=exited)
```

```
[root@ecs-docker-lipengda ~]# docker rm -v $(docker ps -aq -f status=exited)
43c7994a8031
8ff406ee1c45
caa86cb4a26b
8fb82cb6f9cf
af824373adab
8c9a4a6e6707
fa0e576892f6
5107cee39eab
```

图 46: 删除所有已终止的容器

## 2.5 私有镜像仓库搭建

### 2.5.1 安装运行 docker-registry

获取官方 registry 镜像并运行容器。

```
1 docker run -d -p 5000:5000 --restart=always --name registry registry
```

```
[root@ecs-docker-lipengda ~]# docker run -d -p 5000:5000 --restart=always --name registry registry
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
0dfcae9cb3f0: Pull complete
cfe29ef241d9: Pull complete
d2787542bdb4: Pull complete
4b69fee0ac89: Pull complete
bbb2de197705: Pull complete
Digest: sha256:543dade69668e02e5768d7ea2b0aa4fae6aa7384c9a5a8dbecc2be5136079ddb
Status: Downloaded newer image for registry:latest
fa502f4d2c37e9aef1689f05b416e699cfb740434aedc50efd6f5572b3fc14b0
```

图 47: 获取官方 registry 镜像并运行容器

在本机查看已有的镜像。

```
1 docker images
```

```
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd                latest             f7e624632c1c       5 months ago       177MB
registry             latest             1c6adc34955d       14 months ago      25MB
ubuntu               14.04              55b7b4f7c5d6       2 years ago        187MB
```

图 48: 查看已有的镜像

通过 docker tag 命令将基础镜像 ubuntu:14.04 镜像进行标记。

```
1 docker tag ubuntu:14.04 127.0.0.1:5000/myubuntu:14.04
2 docker images
```

```
[root@ecs-docker-lipengda ~]# docker tag ubuntu:14.04 127.0.0.1:5000/myubuntu:14.04
[root@ecs-docker-lipengda ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	latest	f7e624632c1c	5 months ago	177MB
registry	latest	1c6adc34955d	14 months ago	25MB
127.0.0.1:5000/myubuntu	14.04	55b7b4f7c5d6	2 years ago	187MB
ubuntu	14.04	55b7b4f7c5d6	2 years ago	187MB

图 49: 标记镜像

使用 docker push 上传标记的镜像。

```
1 docker push 127.0.0.1:5000/myubuntu:14.04
```

```
[root@ecs-docker-lipengda ~]# docker push 127.0.0.1:5000/myubuntu:14.04
The push refers to repository [127.0.0.1:5000/myubuntu]
000e628b3e71: Pushed
926ca971b512: Pushed
59199d90878e: Pushed
14.04: digest: sha256:5ed16aa332467821529d451800e6fe599d83e30471e91b096752f8696d9bf6e9 size: 945
```

图 50: 上传镜像

用 curl 查看仓库中的镜像。

```
1 curl 127.0.0.1:5000/v2/_catalog
```

```
[root@ecs-docker-lipengda ~]# curl 127.0.0.1:5000/v2/_catalog
{"repositories":["myubuntu"]}
```

图 51: 查看仓库中的镜像

删除已有镜像，再尝试从私有仓库中下载这个镜像

```
1 docker image rm 127.0.0.1:5000/myubuntu:14.04
2 docker images
3 docker pull 127.0.0.1:5000/myubuntu:14.04
4 docker images
```

```
[root@ecs-docker-lipengda ~]# docker image rm 127.0.0.1:5000/myubuntu:14.04
Untagged: 127.0.0.1:5000/myubuntu:14.04
Untagged: 127.0.0.1:5000/myubuntu@sha256:5ed16aa332467821529d451800e6fe599d83e30471e91b096752f8696d9bf6e9
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd                latest              f7e624632c1c       5 months ago       177MB
registry             latest              1c6adc34955d       14 months ago      25MB
ubuntu              14.04              55b7b4f7c5d6       2 years ago        187MB
[root@ecs-docker-lipengda ~]# docker pull 127.0.0.1:5000/myubuntu:14.04
14.04: Pulling from myubuntu
Digest: sha256:5ed16aa332467821529d451800e6fe599d83e30471e91b096752f8696d9bf6e9
Status: Downloaded newer image for 127.0.0.1:5000/myubuntu:14.04
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd                latest              f7e624632c1c       5 months ago       177MB
registry             latest              1c6adc34955d       14 months ago      25MB
127.0.0.1:5000/myubuntu 14.04              55b7b4f7c5d6       2 years ago        187MB
ubuntu              14.04              55b7b4f7c5d6       2 years ago        187MB
```

图 52: 重新下载镜像

## 2.6 Dockerfile 文件构建

### 2.6.1 构建 nginx

下载基础镜像 centos:7

```
1 docker pull centos:7
```

创建 nginx\_demo 文件夹

```
1 pwd
2 mkdir nginx_demo
3 ls
```

```
[root@ecs-docker-lipengda ~]# curl 127.0.0.1:5000/v2/_catalog
{"repositories":["myubuntu"]}
```

图 53: 创建 nginx\_demo 文件夹

进入文件夹，下载 nginx 源码压缩包。

```
1 cd nginx_demo
2 wget http://nginx.org/download/nginx-1.12.2.tar.gz
```

```
[root@ecs-docker-lipengda ~]# cd nginx_demo
[root@ecs-docker-lipengda nginx_demo]# wget http://nginx.org/download/nginx-1.12.2.tar.gz
--2024-12-17 00:06:34-- http://nginx.org/download/nginx-1.12.2.tar.gz
Resolving nginx.org (nginx.org) ... 52.58.199.22, 3.125.197.172, 2a05:d014:5e0:2601::6, ...
Connecting to nginx.org (nginx.org)|52.58.199.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 981687 (959K) [application/octet-stream]
Saving to: 'nginx-1.12.2.tar.gz'

nginx-1.12.2.tar.gz      100%[=====>] 958.68K  509KB/s  in 1.9s
2024-12-17 00:06:36 (509 KB/s) - 'nginx-1.12.2.tar.gz' saved [981687/981687]
```

图 54: 下载 nginx 源码压缩包

创建 Dockerfile 文件。

```
1 vim Dockerfile
```

输入以下内容。

由于 centos 7 已经停止维护，其中的 yum 源已经失效，需要修改 yum 源。

```
1 # base image
2 FROM centos:7
3
4 # MAINTAINER
5 MAINTAINER lipengda
6
7 # put nginx-1.12.2.tar.gz into /usr/local/src and unpack nginx
8 ADD nginx-1.12.2.tar.gz /usr/local/src
9
+ RUN sed -i s/^#.*baseurl=http/baseurl=http/g /etc/yum.repos.d/*.repo
+ RUN sed -i s/^mirrorlist=http/#mirrorlist=http/g /etc/yum.repos.d/*.repo
+ RUN sed -i s/mirror.centos.org/vault.centos.org/g /etc/yum.repos.d/*.repo
13
14 # running required command
15 RUN yum install -y gcc gcc-c++ glibc make autoconf openssl openssl-devel
16 RUN yum install -y libxslt-devel -y gd gd-devel GeoIP GeoIP-devel pcre pcre-
    devel
17 RUN useradd -M -s /sbin/nologin nginx
18
19 # change dir to /usr/local/src/nginx-1.12.2
20 WORKDIR /usr/local/src/nginx-1.12.2
21
22 # execute command to compile nginx
23 RUN ./configure \
24 --user=nginx --group=nginx \
25 --prefix=/usr/local/nginx --with-file-aio \
26 --with-http_ssl_module \
27 --with-http_realip_module \
28 --with-http_addition_module \
```

```
29 --with-http_xslt_module \
30 --with-http_image_filter_module \
31 --with-http_geoip_module \
32 --with-http_sub_module \
33 --with-http_dav_module \
34 --with-http_flv_module \
35 --with-http_mp4_module \
36 --with-http_gunzip_module \
37 --with-http_gzip_static_module \
38 --with-http_auth_request_module \
39 --with-http_random_index_module \
40 --with-http_secure_link_module \
41 --with-http_degradation_module \
42 --with-http_stub_status_module && make && make install
43
44 RUN chmod -R 755 /usr/local/nginx/
45
46 EXPOSE 80
```

通过 Dockerfile 创建 nginx 镜像。

```
1 docker build -t my_nginx:v1 .
```

```
[root@ecs-docker-lipengda nginx_demo]# vim Dockerfile
[root@ecs-docker-lipengda nginx_demo]# docker build -t my_nginx:v1 .
Sending build context to Docker daemon  986.1kB
Step 1/15 : FROM centos:7
--> c9a1fdca3387
Step 2/15 : MAINTAINER lipengda
--> Using cache
--> 9bc7f06e1685
Step 3/15 : ADD nginx-1.12.2.tar.gz /usr/local/src
--> Using cache
--> 4938f9c2ac28
Step 4/15 : RUN sed -i s/^#.*baseurl=http/baseurl=http/g /etc/yum.repos.d/*.repo
--> Using cache
--> 23704118620b
Step 5/15 : RUN sed -i s/^mirrorlist=http/#mirrorlist=http/g /etc/yum.repos.d/*.repo
--> Using cache
--> 666cba70c934
Step 6/15 : RUN sed -i s/mirror.centos.org/vault.centos.org/g /etc/yum.repos.d/*.repo
--> Using cache
--> 184c626ed5ac
Step 7/15 : RUN yum clean all
--> Running in d7e145f4a86f
Loaded plugins: fastestmirror, ovl
Cleaning repos: base extras updates
Removing intermediate container d7e145f4a86f
--> 7f038123564b
Step 8/15 : RUN yum makecache
--> Running in c6749693b77f
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
Metadata Cache Created
Removing intermediate container c6749693b77f
```

图 55: 创建 nginx 镜像 (1)



```

test -f '/usr/local/nginx/conf/fastcgi.conf' \
|| cp conf/fastcgi.conf '/usr/local/nginx/conf'
cp conf/fastcgi.conf '/usr/local/nginx/conf/fastcgi.conf.default'
test -f '/usr/local/nginx/conf/uwsgi_params' \
|| cp conf/uwsgi_params '/usr/local/nginx/conf'
cp conf/uwsgi_params \
'/usr/local/nginx/conf/uwsgi_params.default'
test -f '/usr/local/nginx/conf/scgi_params' \
|| cp conf/scgi_params '/usr/local/nginx/conf'
cp conf/scgi_params \
'/usr/local/nginx/conf/scgi_params.default'
test -f '/usr/local/nginx/conf/nginx.conf' \
|| cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf'
cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf.default'
test -d '/usr/local/nginx/logs' \
|| mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/logs' \
|| mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/html' \
|| cp -R html '/usr/local/nginx'
test -d '/usr/local/nginx/logs' \
|| mkdir -p '/usr/local/nginx/logs'
make[1]: Leaving directory '/usr/local/src/nginx-1.12.2'
Removing intermediate container 420722834c8e
--> 43c7d4bd61a3
Step 14/15 : RUN chmod -R 755 /usr/local/nginx/
--> Running in d7cb40256582
Removing intermediate container d7cb40256582
--> 6aef46e59e1c
Step 15/15 : EXPOSE 80
--> Running in 4e30f0792e9a
Removing intermediate container 4e30f0792e9a
--> 39de736ad4f7
Successfully built 39de736ad4f7
Successfully tagged my_nginx:v1
[root@ecs-docker-lipengda nginx_demo]#

```

图 56: 创建 nginx 镜像 (2)

查看构建的镜像。

```
1 docker images
```

```

[root@ecs-docker-lipengda nginx_demo]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
my_nginx             v1                  39de736ad4f7       4 minutes ago      965MB
httpd                latest              f7e624632c1c       5 months ago       177MB
registry             latest              1c6adc34955d       14 months ago      25MB
ubuntu               14.04              55b7b4f7c5d6       2 years ago        187MB
127.0.0.1:5000/myubuntu 14.04              55b7b4f7c5d6       2 years ago        187MB
centos                7                   c9a1fdca3387       2 years ago        301MB

```

图 57: 查看构建的镜像

## 2.6.2 nginx 镜像验证

通过构建的镜像，运行一个容器，将端口进行映射。

```
1 docker run -d -p 80:80 my_nginx:v1 /usr/local/nginx/sbin/nginx -g "daemon off;"
```

查看容器状态

```
1 docker ps
```

```
[root@ecs-docker-lipengda nginx_demo]# docker run -d -p 80:80 my_nginx:v1 /usr/local/nginx/sbin/nginx -g "daemon off;"
1367aa6037079e4d3c71b8c3a2d5eaadd9a2f1273362903d05656f53bbb48860
[root@ecs-docker-lipengda nginx_demo]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1367aa603707	my_nginx:v1	"/usr/local/nginx/sb..."	33 seconds ago	Up 32 seconds	0.0.0.0:80→80/tcp	unruffled_booth
fa502f4d2c37	registry	"/entrypoint.sh /etc..."	About an hour ago	Up About an hour	0.0.0.0:5000→5000/tcp	registry
af3ed90ccf60	httpd	"httpd-foreground"	2 hours ago	Up 2 hours	80/tcp	myhttpd

图 58: 运行容器并查看容器状态

打开浏览器，输入 ecs-docker 弹性 IP 地址，默认端口为 80，进行验证，显示 “Welcome to nginx!”，说明容器运行正常。



图 59: 验证 nginx 镜像

### 2.6.3 Dockerfile 指令的添加

我们也可以基于以上 Dockerfile 文件依次添加其他的指令进行构建，比如我们可以添加 CMD 命令，设置 nginx 非 daemon 守护进程，这样容器启动时不会自动退出。

```
1 vim Dockerfile
```

在原有 Dockerfile 基础上，增加如下内容到 Dockerfile 最后一行。

```
1 CMD /usr/local/nginx/sbin/nginx -g "daemon off;"
```

重新构建镜像。

```
1 docker build -t my_nginx:v2 .
```

```

--> 7f038123564b
Step 8/16 : RUN yum makecache
--> Using cache
--> 5621b8f67ce0
Step 9/16 : RUN yum install -y gcc gcc-c++ glibc make autoconf openssl openssl-devel
--> Using cache
--> b419e768e149
Step 10/16 : RUN yum install -y libxslt-devel -y gd gd-devel GeoIP GeoIP-devel pcre pcre-devel
--> Using cache
--> 1a63f68f14d9
Step 11/16 : RUN useradd -M -s /sbin/nologin nginx
--> Using cache
--> b2fe527afc55
Step 12/16 : WORKDIR /usr/local/src/nginx-1.12.2
--> Using cache
--> 1e3a1252bc30
Step 13/16 : RUN ./configure --user=nginx --group=nginx --prefix=/usr/local/nginx --with-file-aio --with-http_ssl_module --with-ht
p_realip_module --with-http_addition_module --with-http_xslt_module --with-http_image_filter_module --with-http_geoip_module --
with-http_sub_module --with-http_dav_module --with-http_flv_module --with-http_mp4_module --with-http_gunzip_module --with-ht
p_gzip_static_module --with-http_auth_request_module --with-http_random_index_module --with-http_secure_link_module --with-ht
t_gradation_module --with-http_stub_status_module && make && make install
--> Using cache
--> 43c7d4bd61a3
Step 14/16 : RUN chmod -R 755 /usr/local/nginx/
--> Using cache
--> 6aef46e59e1c
Step 15/16 : EXPOSE 80
--> Using cache
--> 39de736ad4f7
Step 16/16 : CMD /usr/local/nginx/sbin/nginx -g "daemon off;"
--> Running in 6765b22e10fb
Removing intermediate container 6765b22e10fb
--> 060b83ba7a3d
Successfully built 060b83ba7a3d
Successfully tagged my_nginx:v2
[root@ecs-docker-lipengda nginx_demo]#

```

图 60: 重新构建镜像

查看构建的镜像。

```
1 docker images
```

```

[root@ecs-docker-lipengda nginx_demo]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
my_nginx             v2                 060b83ba7a3d       About a minute ago  965MB
my_nginx             v1                 39de736ad4f7       15 minutes ago     965MB
httpd                latest             f7e624632c1c       5 months ago       177MB
registry             latest             1c6adc34955d       14 months ago      25MB
127.0.0.1:5000/myubuntu 14.04             55b7b4f7c5d6       2 years ago        187MB
ubuntu               14.04             55b7b4f7c5d6       2 years ago        187MB
centos                7                 c9alfdc3387        2 years ago        301MB

```

图 61: 查看构建的镜像

通过构建的镜像，运行一个容器，将端口进行映射，将容器的 80 端口映射到主机的 81 端口。

```
1 docker run -d -p 81:80 my_nginx:v2
```

查看容器状态

```
1 docker ps
```

```
[root@ecs-docker-lipengda nginx_demo]# docker run -d -p 81:80 my_nginx:v2
f721ace68667c37f6a2636c0aa3d99d07f8785424a19148b7561476981e2cffe
[root@ecs-docker-lipengda nginx_demo]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f721ace68667	my_nginx:v2	"/bin/sh -c '/usr/lo..."	6 seconds ago	Up 6 seconds	0.0.0.0:81→80/tcp	loving_payne
1367aa603707	my_nginx:v1	"/usr/local/nginx/sb..."	9 minutes ago	Up 9 minutes	0.0.0.0:80→80/tcp	unruffled_booth
fa502f4d2c37	registry	"/entrypoint.sh /etc..."	About an hour ago	Up About an hour	0.0.0.0:5000→5000/tcp	registry
af3ed90ccf60	httpd	"httpd-foreground"	2 hours ago	Up 2 hours	80/tcp	myhttpd

图 62: 运行容器并查看容器状态

打开浏览器进行验证，打开浏览器，输入弹性 IP 地址，端口为 81，进行验证，显示 “Welcome to nginx!”，说明容器运行正常。



图 63: 验证 nginx 镜像

## 2.7 删除弹性云服务器及相关资源

打开云服务器控制台，在需要删除的云服务器后面选择“更多 > 删除”。步骤 2 在弹出对话框中勾选“释放云服务器绑定的弹性公网 IP 地址”和“删除云服务器挂载的数据盘”，然后点击“是”。

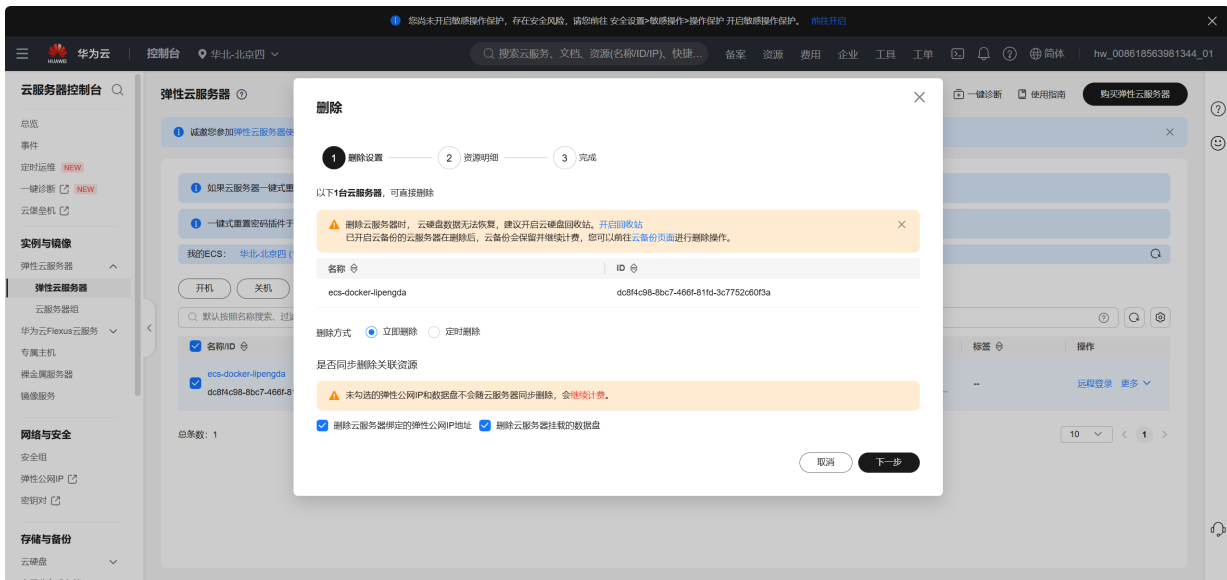


图 64: 删除云服务器

查看到列表中已没有资源时，表示弹性云服务器已删除。



图 65: 删除完成

### 3 实验结果

本次实验中，我们成功完成了鲲鹏云容器的安装和配置，并掌握了 Docker 的基本操作。以下是实验过程中取得的主要结果。

#### 3.1 创建云服务器

在鲲鹏云容器控制台中，成功创建了一个云服务器，如图 66 所示。



图 66: 创建云服务器

#### 3.2 登录云服务器

成功登录到云服务器，验证了服务器的正常运行情况，如图 67 所示。

```
pssh ~
> ssh root@139.9.139.90
The authenticity of host '139.9.139.90 (139.9.139.90)' can't be established.
ED25519 key fingerprint is SHA256:L5Q6Z/o+BlQCmAzdom8AedaGAZg3Yf9UbwjuDnJTOL4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '139.9.139.90' (ED25519) to the list of known hosts.

Authorized users only. All activities may be monitored and reported.
root@139.9.139.90's password:
Welcome to Huawei Cloud Service

Last failed login: Thu Dec 12 13:47:28 CST 2024 from 89.169.55.26 on ssh:notty
There was 1 failed login attempt since the last successful login.

Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time: Thu Dec 12 13:50:54 CST 2024

System load: 0.00
Processes: 144
Memory used: 10.5%
Swap used: 0.0%
Usage On: 9%
IP address: 192.168.0.206
Users online: 1
```

图 67: 登录云服务器

### 3.3 查看已下载的镜像

使用命令 `docker images` 查看已经下载的镜像，可以看到 `hello-world` 镜像已经成功下载，如图 68 所示。

```
[root@ecs-docker-lipengda ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest             ee301c921b8a       19 months ago      9.14kB
```

图 68: 查看已下载的镜像

### 3.4 镜像的基本操作

通过执行镜像的查询、下载和删除等操作，加深了对镜像管理的理解。详细的操作过程和结果可参考第 2.3(点击以跳转) 节。

### 3.5 容器的基本操作

掌握了容器的创建、启动、停止、删除等基本操作。具体的步骤和结果请参见第 2.4 节。

### 3.6 查看私有仓库中的镜像

在搭建私有镜像仓库后，使用 `curl` 命令查看仓库中的镜像，成功获取到了仓库中包含的镜像列表，如图 69 所示。

```
[root@ecs-docker-lipengda ~]# curl 127.0.0.1:5000/v2/_catalog
{"repositories":["myubuntu"]}
```

图 69: 查看私有仓库中的镜像

### 3.7 验证 Nginx 镜像

通过浏览器访问部署在容器中的 Nginx 服务，成功显示了欢迎页面，说明容器运行正常，如图 70 和图 71 所示。



图 70: 验证 Nginx 镜像（端口 80）



图 71: 验证 Nginx 镜像（端口 81）

## 4 实验总结

在本次实验中，我成功完成了云服务器的创建、Docker 的安装和配置，熟悉了 Docker 镜像和容器的基本操作，掌握了使用 Dockerfile 构建自定义镜像的方法。同时，学习了如何搭建私有镜像仓库。通过动手实践，对容器技术有了更深入的理解，为后续的学习和应用奠定了坚实的基础。

在实验过程中，遇到了一些问题。例如：1. CentOS 7 已停止维护，需要手动修改 yum 源配置；2. 需要为 Docker 配置国内镜像源，否则无法下载镜像。这提高了解决实际问题的能力。

总体来说，实验达到了预期效果，加深了对云计算和容器技术的认识。