

🔗 动手实战学Docker (/courses/498)

镜像管理

1. 课程说明

课程为纯动手实验教程，为了能说清楚实验中的一些操作会加入理论内容。理论内容我们不会写太多，已经有太多好文章了，会精选最值得读的文章推荐给你，在动手实践的同时扎实理论基础。

实验环境中可以联网，不受实验楼网络限制。

2. 学习方法

实验楼的 Docker 课程包含 14 个实验，每个实验都提供详细的步骤和截图，适用于有一定 Linux 系统基础，想快速上手 Docker 的同学。

学习方法是多实践，多提问。启动实验后按照实验步骤逐步操作，同时理解每一步的详细内容。

如果实验开始部分有推荐阅读的材料，请务必先阅读后再继续实验，理论知识是实践必要的基础。

3. 本节内容简介

第一节实验中我们已经接触了一些镜像的概念，简单的说镜像就是一个容器的只读模板，用来创建容器。当运行容器时需要指定镜像，如果本地没有该镜像，则会从 Docker Registry 下载。默认查找的是 Docker Hub。Docker 的镜像是增量的修改，每次创建新的镜像都会在老的镜像上面构建一个增量的层，使用到的技术是 Another Union File System(AUFS)，感兴趣的同学可以学习文档 [InfoQ:剖析Docker文件系统：Aufs与Devicemapper](http://www.infoq.com/cn/articles/analysis-of-docker-file-system-aufs-and-devicemapper/) (http://www.infoq.com/cn/articles/analysis-of-docker-file-system-aufs-and-devicemapper/)。

本节中，我们需要依次完成下面几项任务：

1. 查看镜像列表
2. 查看镜像详细信息
3. 查看镜像信息
4. 拉取镜像
5. 构建镜像
6. 删除镜像

4 镜像

动手实战学 Docker (/courses/498)

如果能够熟练的使用上述命令，那么这里关于镜像的部分操作，类比 Management Commands 的特性，可以很容易的学习相应的命令。


镜像存储中的核心概念仓库 (Repository) 是镜像存储的位置。Docker 注册服务器 (Registry) 是仓库存储的位置。每个仓库包含不同的镜像。

比如一个镜像名称 `ubuntu:14.04`，冒号前面的 `ubuntu` 是仓库名，后面的 `14.04` 是 TAG，不同的 TAG 可以对应相同的镜像，TAG 通常设置为镜像的版本号。

Docker Hub 是 Docker 官方提供的公共仓库，提供大量的常用镜像，由于国内网络原因经常连接 Docker Hub 会比较慢。

并且 Docker 的镜像是分层存储，每一个镜像都是由很多层组成的。而一些镜像会共享一些相同的层。对于实验环境中的 docker 来说，其使用的存储驱动是 aufs，如下图所示：

```
shiyancelou:~/ $ docker system info
Containers: 1
  Running: 1
  Paused: 0
  Stopped: 0
Images: 8
Server Version: 17.05.0-ce
Storage Driver: aufs
Root Dir: /var/lib/docker/aufs
Backing Filesystem: extfs
Dirs: 16
Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
  Volume: local
  Network: bridge host macvlan null overlay
```



图片中显示的是 aufs，但是对于如果是在自己的 Linux 环境下安装的 docker，其版本是高于 17.05，显示的有可能是 overlay2，其基本原理和 aufs 类似。

aufs 是一种联合文件系统(UnionFS)，理解其原理对于我们理解 Docker 镜像有很大的帮助，有兴趣的同学可以尝试学习 Linux 文件系统之 aufs (<https://segmentfault.com/a/1190000008489207>)

4.1 查看镜像列表


我们查看镜像可以使用如下命令：

动手实战学 Docker (/courses/498)

也可以查看指定仓库的镜像，例如。查看 ubuntu 仓库的镜像：

```
$ docker image ls ubuntu
```

```
shiyanolou:~/ $ docker image ls
REPOSITORY      TAG              IMAGE ID        CREATED         SIZE
busybox         latest          807fd4df40d1    2 weeks ago    1.14MB
hello-world     latest          f2a91732366c    2 months ago   1.85kB
ubuntu          latest          c69811d4e993    5 months ago   188MB
shiyanolou:~/ $
shiyanolou:~/ $ docker image ls ubuntu
REPOSITORY      TAG              IMAGE ID        CREATED         SIZE
ubuntu          latest          c69811d4e993    5 months ago   188MB
shiyanolou:~/ $
shiyanolou:~/ $
```



4.2 查看镜像的详细信息

查看镜像的详细信息使用如下命令：

```
docker image inspect ubuntu
```

4.3 拉取镜像

上面的内容中描述了仓库和注册表的内容，这里，我们学习从注册表中获得镜像或者仓库的命令，使用如下命令：

```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
```

比较常用的配置参数为 `-a`，代表下载仓库中的所有镜像，即下载整个存储库。

如下所示，我们下载 `ubuntu:14.04` 镜像，使用如下命令：

```
$ docker image pull ubuntu:14.04
```

```
shiyancelou:~/ $ docker image pull ubuntu:14.04
14.04: Pulling from library/ubuntu
556ab54f6580: Pull complete
6ca8fd6ef32a: Pull complete
e64a7e7a2b21: Pull complete
09ed7c05bfd3: Pull complete
ec1faacd6fd9: Pull complete
Digest: sha256:f0cc0848fdadb4ae7341028a21f894df7cc2ab56e2bfe162850cbd602234d9a4
Status: Downloaded newer image for ubuntu:14.04
shiyancelou:~/ $
shiyancelou:~/ $ docker image ls ubuntu
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu               14.04              02a63d8b2bfa       7 days ago         222MB
ubuntu               latest             c69811d4e993       5 months ago       188MB
shiyancelou:~/ $
shiyancelou:~/ $
```

对于 pull 下来的镜像来说，其具体的保存路径为 `/var/lib/docker`。因为这里的存储驱动为 `aufs`，所以具体路径为 `/var/lib/docker/aufs`

4.4 构建镜像

commit

此时，对于我们 pull 的新镜像 `ubuntu:14.04` 来说，如果我们需要对其进行更新，可以创建一个容器，在容器中进行修改，然后将修改提交到一个新的镜像中。

提交修改使用如下命令：

```
docker container commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
```

该命令的解释为从一个容器的修改中创建一个新的镜像。例如，我们运行一个容器，然后在其中创建一个文件，最后使用 `commit` 命令：

```
# 使用 run 创建运行一个新命令
$ docker container run -it --name shiyancelou001 busybox /bin/sh

# 在运行的容器中创建两个文件，test1 和 test2
touch test1 test2

# 使用 ctrl + p 及 ctrl+q 键退出

# 使用提交命令，提交容器 shiyancelou001 的修改到镜像 busybox:test 中
$ docker container commit shiyancelou001 busybox:test

# 查看通过提交创建的镜像
$ docker image ls busybox
```

```
shiyancelou:~/ $ docker container run -it --name shiyancelou001 busybox /bin/sh
/ #
/ # ls
bin    dev    etc    home   proc   root   sys    tmp    usr    var
/ # touch test1 test2
/ # ls
bin    dev    etc    home   proc   root   sys    test1  test2  tmp    usr    var
/ # %
shiyancelou:~/ $ docker container commit shiyancelou001 busybox:test
sha256:03e89bfe9fa37abfc7c9d65784a461dfd75e7c8142f7151771a65f2d1f4eacad
shiyancelou:~/ $
shiyancelou:~/ $ docker image ls busybox
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
busybox             test            03e89bfe9fa3    11 seconds ago  1.14MB
busybox             latest          807fd4df40d1    2 weeks ago    1.14MB
shiyancelou:~/ $
```

通过上述操作我们创建了一个新的镜像，但是本方法不推荐用在生产系统中，未来会很难维护镜像。最好的创建镜像的方法是 Dockerfile，修改镜像的方法是修改 Dockerfile，然后重新从 Dockerfile 中构建新的镜像。

BUILD

docker 可以从一个 Dockerfile 文件中自动读取指令构建一个新的镜像。Dockerfile 是一个包含用户构建镜像命令的文本文件。在创建该文件后，我们可以使用如下命令构建镜像：

```
docker image build [OPTIONS] PATH | URL
```

构建镜像时，该过程的第一件事是将 Dockerfile 文件所在目录下的所有内容递归的发送到守护进程。所以在大多数情况下，最好是创建一个新的目录，在其中保存 Dockerfile，并在其中添加构建 Dockerfile 所需的文件。

对于一个 Dockerfile 文件内容来说，基本语法格式如下所示：

```
# Comment
INSTRUCTION arguments
```

使用 # 号作为注释，指令（INSTRUCTION）不区分大小写，但是为了可读性，一般将其大写。而 Dockerfile 的指令一般包含下面几个部分：

1. 基础镜像：以哪个镜像为基础进行制作，使用 FROM 指令来指定基础镜像，一个 Dockerfile 必须以 FROM 指令启动。
2. 维护者信息：可以指定该 Dockerfile 编写人的姓名及邮箱，使用 MAINTAINER 指令。
3. 镜像操作命令：对基础镜像要进行的改造命令，比如安装新的软件，进行哪些特殊配置等，常见的是 RUN 命令。

4. 容器启动命令：基于该镜像的容器启动时需要执行哪些命令，常见的是 CMD 命令或 ENTRYPOINT

动手实战学Docker (/courses/498)

例如一个最基本的 Dockerfile ：

```
# 指定基础镜像
FROM ubuntu:14.04

# 维护者信息
MAINTAINER shiyanlou/shiyanlou001@simplecloud.cn

# 镜像操作命令
RUN apt-get -yqq update && apt-get install -yqq apache2

# 容器启动命令
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

通过阅读上述内容中我们熟悉的一些 linux 指令，可以很容易的得出该命令创建了一个 apache 的镜像。包含了最基本的四项信息。

其中 FROM 指定基础镜像。RUN 命令默认使用 /bin/sh，并使用 root 权限执行。CMD 命令也是默认在 /bin/sh 中执行，但是只能有一条 CMD 指令，如果有多条则只有最后一条会被执行。

下面我们创建一个空目录，并在其中编辑 Dockerfile 文件，并基于此构建一个新的镜像，使用如下操作：

```
# 首先创建目录并切换目录
$ mkdir /home/shiyanlou/test1 && cd /home/shiyanlou/test1

# 编辑 Dockerfile 文件，默认文件名为 `Dockerfile`，也可以使用其它值，使用其它值需要在构建时通过 `-f` 参数指定，这里我们使用默认值。并在其中添加上述示例的内容
$ vim Dockerfile

# 使用 build 命令，`-t` 参数指定新的镜像
$ docker image build -t shiyanlou:1.0 .
```

```
shiyanlou:~/ $ mkdir test1 && cd test1
shiyanlou:test1/ $
shiyanlou:test1/ $ vim Dockerfile
shiyanlou:test1/ $
shiyanlou:test1/ $ docker image build -t shiyanlou:1.0 .
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM ubuntu:14.04
--> 02a63d8b2bfa
Step 2/4 : MAINTAINER shiyanlou/shiyanlou001@simplecloud.com
--> Using cache
--> 1abf71fef5a9
Step 3/4 : RUN apt-get -yqq update && apt-get install -yqq apache2
--> Running in f2cd403afe29
```



在执行构建命令后，需要花费一些时间来完成构建。在运行结束后，最后查看新创建的镜像：

👉 动手实战学Docker (/courses/498)

```
shiyanolou:test1/ $ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
shiyanolou	1.0	fcab8de84cb9	2 minutes ago	257MB
busybox	test	03e89bfe9fa3	About an hour ago	1.14MB
ubuntu	14.04	02a63d8b2bfa	7 days ago	222MB
busybox	latest	807fd4df40d1	2 weeks ago	1.14MB
hello-world	latest	f2a91732366c	2 months ago	1.85kB
ubuntu	latest	c69811d4e993	5 months ago	188MB

```
shiyanolou:test1/ $
```

在构建完成后，我们可以使用该镜像启动一个容器来运行 apache 服务，运行如下命令：

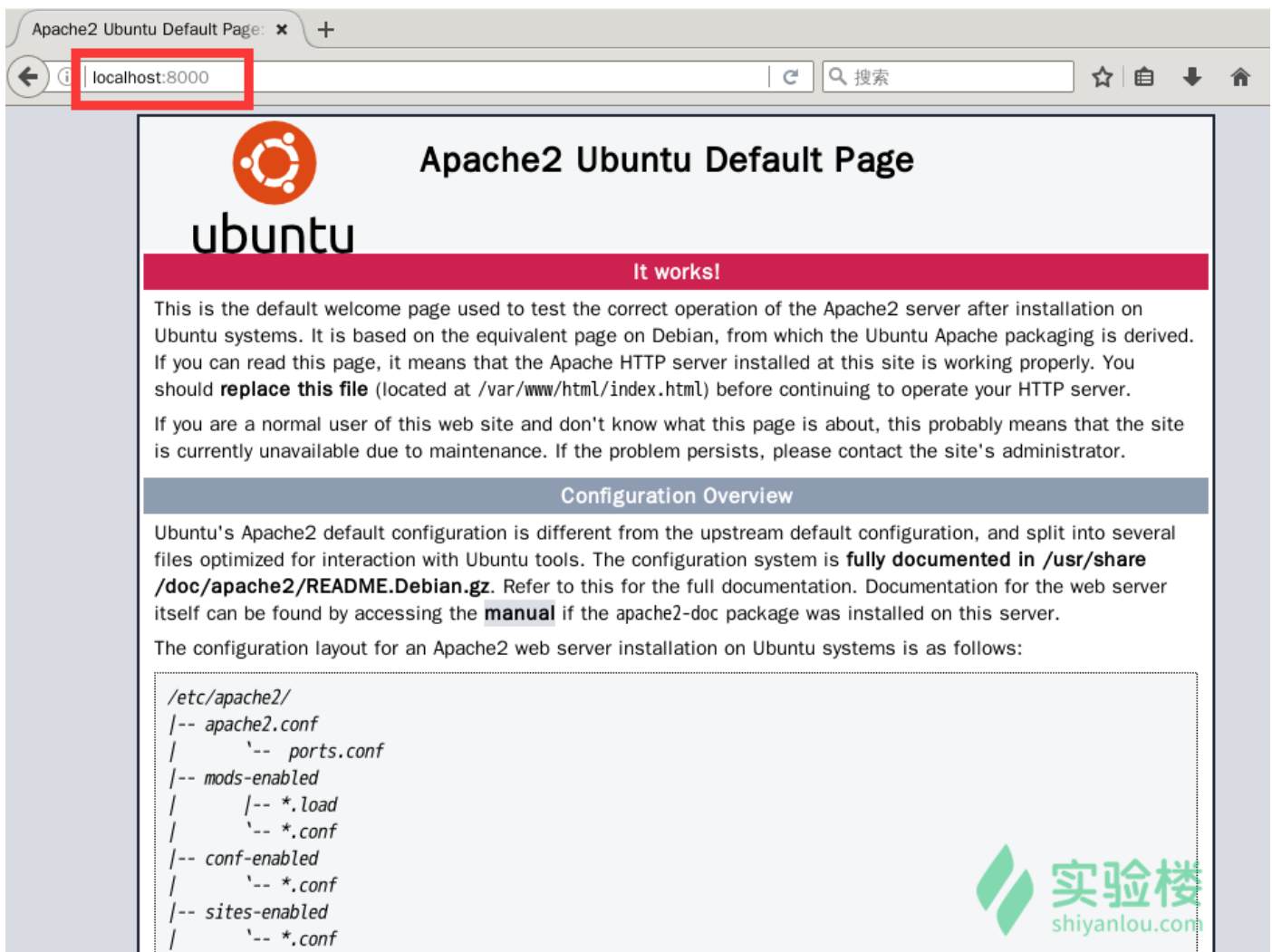
使用 -p 参数将本机的 8000 端口映射到容器中的 80 端口上。

```
$ docker container run -d -p 8000:80 --name shiyanolou002 shiyanolou:1.0
```

```
shiyanolou:test1/ $ docker container run -d -p 8000:80 --name shiyanolou002 shiyanolou:1.0
da2280d4b86a4a6a3576fc5395f09d110837575e6a1e957a96ee1c05b89e3f43
```

```
shiyanolou:test1/ $
```

此时，容器启动成功后，并且配置了端口映射，我们就可以通过本机的 8000 端口访问容器 shiyanolou002 中的 apache 服务了。我们打开浏览器，输入 localhost:8000，显示结果如下图：



更多有关于 Dockerfile 文件格式的信息可以参考官方文档
动手实战学 Docker (/courses/498)
<https://docs.docker.com/engine/reference/builder/>
(<https://docs.docker.com/engine/reference/builder/>)

4.5 删除

我们删除 `ubuntu:latest` 镜像就可以使用如下命令：

```
# 删除镜像
$ docker image rm ubuntu
```

需要注意的是，如果该镜像正在被一个容器所使用，需要将容器删除才能成功的删除镜像。

5. 总结

本节实验中我们学习了以下内容：

1. 查看镜像列表
2. 查看镜像详细信息
3. 查看镜像信息
4. 拉取镜像
5. 构建镜像
6. 删除镜像

请务必保证自己能够动手完成整个实验，只看文字很简单，真正操作的时候会遇到各种各样的问题，解决问题的过程才是收获的过程。

**本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。*

上一节：Docker 容器管理 (/courses/498/labs/1704/document)

下一节：Docker 存储管理 (/courses/498/labs/1706/document)