动手实战学Docker (/courses/498)

编写Dockerfile

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

在前面的实验中我们多次用到的 Dockerfile，在本实验里我们将通过完成一个实例来学习 Dockerfile 的编写。

本节中，我们需要依次完成下面几项任务：

1. Dockerfile 基本语法
2. Dockerfile 创建镜像流程

4. Dockerfile

Dockerfile 是一个文本文件，其中包含我们为了构建 Docker 镜像而手动执行的所有命令。Docker 可以从 Dockerfile 中读取指令来自动构建镜像。我们可以使用 `docker build` 命令来创建一个自动构建。

4.1 上下文

在 Docker 容器及镜像管理一节中我们有提到构建镜像的一些知识。

动手实战学Docker (/courses/498)

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。而 Dockerfile 文件所在的路径也被称为上下文（context）。

首先创建一个目录，以便开始后面的实验过程：

```
$ mkdir dir1 && cd dir1
```

下面我们简单介绍 Dockerfile 中常用的指令。

4.2 FROM

使用 FROM 指令指定一个基础镜像，后续指令将在此镜像的基础上运行：

```
FROM ubuntu:14.04
```

4.3 USER

在 Dockerfile 中可以指定一个用户，后续的 RUN ， CMD 以及 ENTRYPOINT 指令都会使用该用户去执行，但是该用户必须提前存在。

```
USER shiyanlou
```

4.4 WORKDIR

除了指定用户之外，还可以使用 WORKDIR 指定工作目录，对于 RUN ， CMD ， COPY ， ADD 指令将会在指定的工作目录中去执行。也可以理解为命令执行时的当前目录。

```
WORKDIR /
```

4.5 RUN，CMD，ENTRYPOINT

RUN 指令用于执行命令，该指令有两种形式：

- RUN <command> ，使用 shell 去执行指定的命令 command ，一般默认的 shell 为 /bin/sh -c 。
- RUN ["executable", "param1", "param2", ...] ，使用可执行的文件或程序 executable ，给予相应的参数 param 。

例如我们执行更新命令：

动手实战学Docker (/courses/498)

```
RUN apt-get update
```

CMD 的使用方式跟 RUN 类似，不过在一个 Dockerfile 文件中只能有一个 CMD 指令，如果有多个 CMD 指令，则只有最后一个会生效。该指令为我们运行容器时提供默认的命令，例如：

```
CMD echo "hello shiyanlou"
```

在构建镜像时使用了上面的 `CMD` 指令，则可以直接使用 `docker run image` ，该命令等同于 `docker run image echo "hello shiyanlou"` 。即作为默认执行容器时默认使用的命令，也可在 `docker run` 中指定需要运行的命令来覆盖默认的 `CMD` 指令。

除此之外，该指令还有一种特殊的用法，在 Dockerfile 中，如果使用了 ENTRYPOINT 指令，则 CMD 指令的值会作为 ENTRYPOINT 指令的参数：

```
CMD ["param1", "param2"]
```

ENTRYPOINT 指令会覆盖 CMD 指令作为容器运行时的默认指令，并且不会在 `docker run` 时被覆盖，如下示例：

```
FROM ubuntu:latest
ENTRYPOINT ["ls", "-a"]
CMD ["-l"]
```

上述构建的镜像，在我们使用 `docker run image` 时等同于 `docker run image ls -a -l` 命令。使用 `docker run image -i -s` 命令等同于 `docker run image ls -a -i -s` 指令。即 CMD 指令的值会被当作 ENTRYPOINT 指令的参数附加到 ENTRYPOINT 指令的后面。

4.6 COPY 和 ADD

COPY 和 ADD 都用于将文件，目录等复制到镜像中。使用方式如下：

```
ADD <src>... <dest>
ADD ["<SRC>",... "<dest>"]

COPY <src>... <dest>
COPY ["<src>",... "<dest>"]
```

`<src>` 可以指定多个，但是其路径不能超出上下文的路径，即必须在跟 Dockerfile 同级或子目录中。

动手实战学Docker (/courses/498)

`<dest>` 不需要预先存在，不存在路径时会自动创建，如果没有使用绝对路径，则 `<dest>` 为相对于工作目录的相对路径。

COPY 和 ADD 的不同之处在于，ADD 可以添加远程路径的文件，并且 `<src>` 为可识别的压缩格式，如 gzip 或 tar 归档文件等，ADD 会自动将其解压缩为目录。

4.7 ENV

ENV 指令用于设置环境变量：

```
ENV <key> <value>
ENV <key>=<value> <key>=<value>...
```

4.8 VOLUME

VOLUME 指令将会创建指定的挂载目录，在容器运行时，将创建相应的匿名卷：

```
VOLUME /data1 /data2
```

上述指令将会在容器运行时，创建两个匿名卷，并挂载到容器中的 /data1 和 /data2 目录上。

4.9 EXPOSE

EXPOSE 指定在容器运行时监听指定的网络端口，它与 `docker run` 命令的 `-p` 参数不一样，并不实际映射端口，只是将该端口暴露出来，允许外部或其它的容器进行访问。

```
EXPOSE port
```

5. 从 Dockerfile 创建镜像

了解了上面一些常用于构建 Dockerfile 的指令之后，可以通过这些指令来构建一个镜像，如下所示，搭建一个 ssh 服务:

```
# 指定基础镜像
FROM ubuntu:14.04

# 安装软件
RUN apt-get update && apt-get install -y openssh-server && mkdir /var/run/sshd

# 添加用户 shiyanlou 及设定密码
RUN useradd -g root -G sudo shiyanlou && echo "shiyanlou:123456" | chpasswd shiyanlou

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]
```

首先，我们在之前创建的一个空目录 `dir1` 中编辑 `Dockerfile` 文件，并将上面的内容复制到该文件中，相关的命令如下所示：

```
# 创建目录
$ mkdir dir1 && cd dir1

# 编辑 Dockerfile，将上面的内容写入
$ vim Dockerfile

# 最后执行构建命令
$ docker build -t sshd:test .
```

在上面的命令执行完成之后，该镜像就构建成功了，直接使用该镜像启动一个容器就可以运行一个 ssh 的服务，如下所示：

```
$ docker run -itd -p 10001:22 --rm sshd:test
```

这时就可以通过公网的 IP 地址，以及端口 10001，并且使用用户 `shiyanlou` ，密码 `123456` ，远程通过 `ssh` 连接到该容器中了。

6. 总结

本节实验中我们学习了以下内容：

1. Dockerfile 基本语法
2. Dockerfile 创建镜像流程

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。

上一节：动手实战学Docker网络管理 (/courses/498/labs/1707/document)

下一节：Docker运行MongoDB及Redis (/courses/498/labs/1709/document)