Dockerfile

作者：civen
原文链接：https://ld246.com/article/1509679205977
来源网站：链滴

DOCKERFILE注意事项

准则

1. 尽量将Dockerfile放在空目录中，如果目录中必须有其他文件，则使用.dockerignore文件。
2. 避免安装不必须的包。
3. 每个容器应该只关注一个功能点。
4. 最小化镜像的层数。
5. 多行参数时应该分类。这样更清晰直白，便于阅读和review，另外，在每个换行符\前都增加一个格。
6. 对构建缓存要有清楚的认识。

指令注意事项

FROM

Dockerfile reference for the FROM instruction

任何时候，尽量使用官方镜像源作为你镜像的基础镜像。我们建议使用Debian Image，因为其很好地管理着，并且作为一个完整的发布包，但体积却保持着最小化（当前不足150MB）。

1. FROM必须是除了注释以外的第一行；
2. 可以有多个FROM语句，来创建多个image；
3.

LABEL

Dockerfile reference for the LABEL instruction

RUN

Dockerfile reference for the RUN instruction

RUN语句有两种格式：

1. RUN

apt-get

尽量避免使用RUN apt-get upgrade或者dist-upgrade，因为基础镜像的很多核心包不会再未权的容器中升级。

要结合RUN apt-get update和apt-get install在同一个RUN语句下一起使用。如：

```
RUN apt-get update && apt-get install -y \
  package-bar \
  package-baz \
  package-foo
```

原文链接：Dockerfile

如果将update和install分开使用，执行多个Dockerfile时，会引起缓存问题，导致后面执行的install语句会失败。

另外，执行完apt-get语句后，最后最好加上删除安装包的语句，以减小镜像的体积。如：

```
RUN apt-get update && apt-get install -y \
    aufs-tools \
    automake \
    build-essential \
 && rm -rf /var/lib/apt/lists/*
```

注意：官方的Debian和Ubuntu镜像会自动执行“RUN apt-get clean”，所以不需要明确地删指令。

管道使用

很多RUN命令都需要使用到管道，如：

```
RUN wget -O - https://some.site | wc -l > /number
```

Docker使用/bin/sh -c解释器来执行这些命令，该解释器只评估管道最后一个操作的返回值来判整个命令是否成功。在上面的例子中，只要wc -l命令成功了，即使wget命令失败了，也会创建一个镜像。为了避免上述情况，可以在语句首部加上set -o pipefail &&。比如：

```
RUN set -o pipefail && wget -O - https://some.site | wc -l > /number
```

注意：并非所有的shell都支持-o pipefail选项，比如说基于Debian的镜像下的模式shell：dash shell。这种情况下，我们可以使用exec格式的RUN命令来显示地选择shell来支持pipefail选项。如：

```
RUN ["/bin/bash", "-c", "set -o pipefail && wget -O - https://some.site | wc -l > /number"]
```

CMD

Dockerfile reference for the CMD instruction

CMD语句与RUN不同，RUN是在build镜像的时候运行，而CMD语句是在build结束后运行。一Dockerfile钟可以有多个RUN语句，虽然也可以有多个CMD语句，但是却只有最后一条CMD语句会行。CMD语句格式为：

```
CMD [“executable”, “param1”, “param2”...]
```

EXPOSE

Dockerfile reference for the EXPOSE instruction

EXPOSE指令指明容器会监听链接的端口。因此，最好使用常用的、传统的应用端口。比如，Apache web服务器使用EXPOSE 80等。

为了给外部链接使用，你需要使用docker run命令来制定容器端口和host端口的映射。

ENV

Dockerfile reference for the ENV instruction

用于设置环境变量，设置后，后面的RUM指令就可以使用之前的环境变量了。同时，还可以通过ocker run –env key=value，在容器启动时设置环境变量。如：

```
ENV PG_MAJOR 9.3
ENV PG_VERSION 9.3.4
RUN curl -SL http://example.com/postgres-$PG_VERSION.tar.xz | tar -xJC /usr/src/postgress
& …
ENV PATH /usr/local/postgres-$PG_MAJOR/bin:$PATH
```

ADD和COPY

Dockerfile reference for the ADD instruction

Dockerfile reference for the COPY instruction

虽然ADD和COPY功能相似，但一般来讲，更建议使用COPY。因为COPY比ADD更透明，COPY支持从本地文件到容器的拷贝，但是ADD还有一些其他不明显的特性（比如本地tar包解压缩和远程URL支持）。因此，ADD的最优用处是本地tar包自动解压缩到镜像中。如：ADD rootfs.tar.xz /。

如果有多个Dockerfile步骤用于处理不同的文件，建议分开COPY它们，而不是一次性拷贝。这以保证每个步骤的build缓存只在对应的文件改变时才无效。比如：

```
COPY requirements.txt /tmp/
RUN pip install --requirement /tmp/requirements.txt
COPY . /tmp/
```

镜像的大小很重要，因此不鼓励使用ADD从远端URL获取包；可以使用curl或者wget来代替。这方式你可以删除不再需要的文件，如解压缩后的tar包，从而不需要再添加额外的layer到镜像中。比，你应该避免这样使用：

```
ADD http://example.com/big.tar.xz /usr/src/things/
RUN tar -xJf /usr/src/things/big.tar.xz -C /usr/src/things
RUN make -C /usr/src/things all
```

而应该如此：

```
RUN mkdir -p /usr/src/things \
    && curl -SL http://example.com/big.tar.xz \
    | tar -xJC /usr/src/things \
    && make -C /usr/src/things all
```

对于不需要使用ADD命令tar包自动解压缩功能的文件和目录，你应该总是使用COPY。

ENTRYPOINT

Dockerfile reference for the ENTRYPOINT instruction

使用ENTRYPOINT来设置镜像的主命令，就像这个镜像运行时就是这条命令一样（然后再使用CD作为默认的flag）。

我们使用s3cmd命令作为镜像的主命令。

```
ENTRYPOINT ["s3cmd"]
CMD ["--help"]
```

VOLUME

Dockerfile reference for the VOLUME instruction

VOLUME指令一般用于数据库的存储区域，配置存储，或者docker容器创建的文件和目录。

USER

Dockerfile reference for the USER instruction

如果服务可以在不需要特权的情况下运行，那么就应该使用USER来切换用户至非root用户。可用RUN命令创建用户组和用户如：

```
RUN groupadd -r postgres && useradd -r -g postgres postgres
```

应该避免安装和使用sudo，因为它有不可预知的TTY和信号转移特性，会产生很多问题。如果的一定要使用类似sudo的功能（如root下初始化daemon，非root下运行），可以使用“gosu”。

WORKDIR

Dockerfile reference for the WORKDIR instruction

为了Dockerfile内容更加清晰和可靠，最好总是使用绝对路径。同样地，应该使用WORKDIR，不是使用类似“cd ... && do-something”这样的指令，因为那样会导致难以阅读、查找错误和维
。

ONBUILD

Dockerfile reference for the ONBUILD instruction

其他资源

Dockerfile Best Practices

Dockerfile Reference

https://github.com/docker-library/buildpack-deps/blob/master/jessie/Dockerfile

.dockerignore file

http://dockone.io/article/2034

https://docs.resin.io/deployment/build-optimisation/