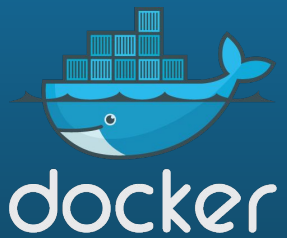
Dockerfiles Best Practices

Nicola Kabar



Agenda

- Introduction to Dockerfiles
- Image Build Process
- Base Images
- Common Dockerfile Directives
- Building Efficient Images

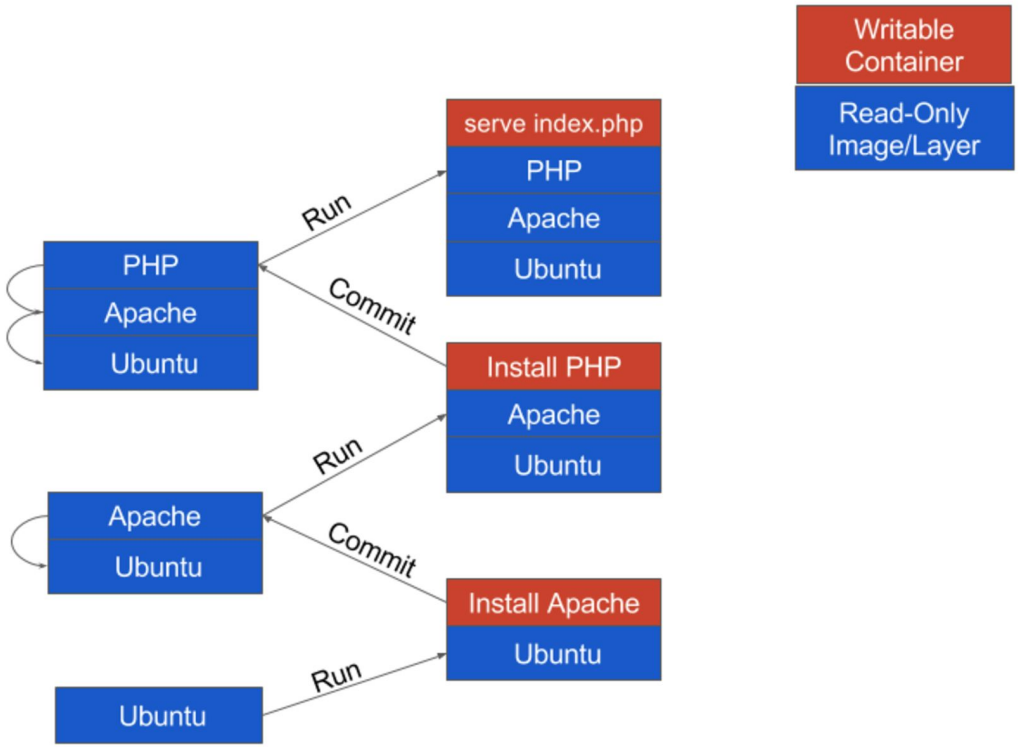
What's a Dockerfile?

- Text document containing the instructions to construct an image
- Docker client sends it along with the context to Docker engine
- Docker engine uses it and a specified *context* to build an image
- Engines goes through a process of running and committing image layers from each line in the Dockerfile
- Uses predefined directives/syntax
- Stored alongside the app's source code in VCS

Sample Dockerfile

```
# A basic apache server. To use either add or bind mount content under /var/www
FROM ubuntu:12.04
MAINTAINER Kimbro Staken version: 0.1

RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

Image Building Process

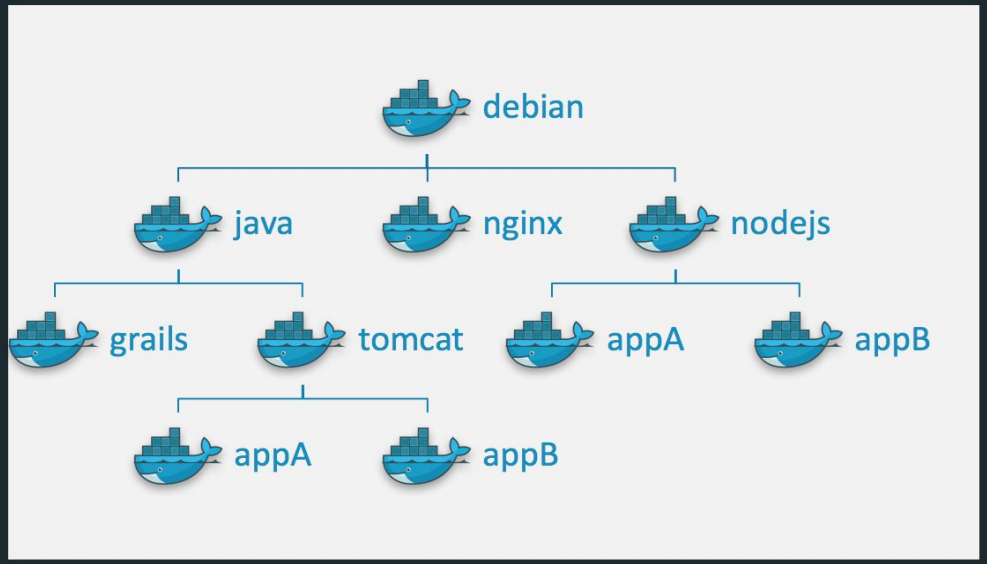


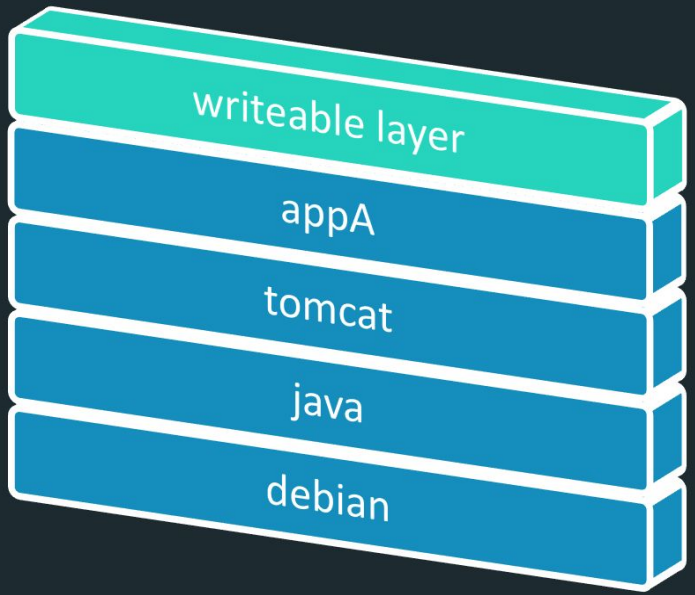
docker

Base Images

- First line in Dockerfile must always be FROM
- This is the base image in which Docker starts from
- Utilize minimal base images (e.g alpine)
- Recommend to build in-house (middleware, os, systems, ops)
- Should be Scanned and Signed
- Common base images
  - Alpine (all Docker images use alpine)
  - Ubuntu
  - Centos

Hierarchical Order





Common Dockerfile Directives

1) FROM: specifies base image
2) MAINTAINER: identifies the maintainer of the image(depreciated :( )
3) LABEL: arbitrary key-value labels for your images
4) RUN: shell or cmd to run
5) CMD: instruction to run software of the container
6) ENTRYPOINT: turns container to an executable
7) EXPOSE: specify which ports container will listen on
8) ENV: specify environment variables
9) ADD or COPY: copy files/dirs from context/URL to container
10) USER: user to use within the container
11) HEALTHCHECK: custom health-check of container
12) WORKDIR: working directory within the container
13) SHELL: specify custom shell (default /bin/sh)
14) ARG: build-time arguments

Building Efficient Images

1) Minimize Number of Layers

BAD!

```
RUN apt-get update
RUN apt-get install -y wget
RUN rm -rf
/var/lib/apt/lists/*
```

Good!

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

Building Efficient Images

2) Use cache-busting: ensuring you never use cache outdated updates

BAD!

```
RUN apt-get update
RUN apt-get install -y wget
RUN rm -rf
/var/lib/apt/lists/*
```

Good!

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

Building Efficient Images

3) Update by version

Good!

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

Better !

```
RUN apt-get update &&\
  apt-get install -y
wget=1.17.1-1ubuntu1.1 &&\
  rm -rf /var/lib/apt/lists/*
```

Building Efficient Images (cont.)

4) Sort multi-line items to avoid duplication and confusion

```
RUN apt-get update && apt-get install -y \
  bzr \
  cvs \
  git \
  mercurial \
  subversion
```

Building Efficient Images (cont.)

5) Remove caches and archives during a single RUN command so they are not included in your final image

```
RUN wget -O /tmp/tomcat7.tar.gz
http://www.us.apache.org/dist/tomcat/tomcat-7/v7.0.63/bi
n/apache-tomcat-7.0.63.tar.gz &&\
  cd /opt &&\
  tar zxf /tmp/tomcat7.tar.gz &&\
  mv /opt/apache-tomcat* /opt/tomcat && \
  rm /tmp/tomcat7.tar.gz
```

Building Efficient Images (cont)

6) Separate changes that break the cache to utilize layer caching

BAD!

```
COPY . /usr/src
RUN npm install
```

Good!

```
COPY package.json /usr/src/package.json
RUN npm install
COPY . /usr/src
```

Building Efficient Images (cont)

7) Use a .dockerignore file (supports similar patterns to .gitignore)

```
*.log
*.git
...
```

Building Efficient Images (cont)

8) Don’t boot init because containers model processes not machines.

9) Don’t upgrade in application image builds. Upgrades should be handled in upstreams base images.

10) Use specific tags (prefer git SHAs)

11) Use CMD and Entrypoint Together.

12) Use non-root user when possible

13) One container = one responsibility = one process

14) Install what you absolutely **NEED**

References

1) docs.docker.com :)
2) https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/
3) https://docs.docker.com/engine/reference/builder/#run
4) https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/