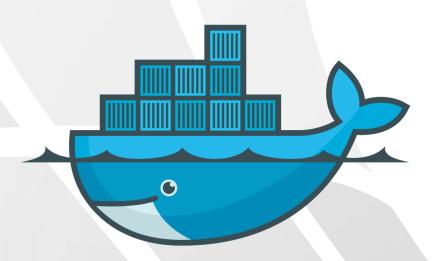


Module 06: Building Containers

Docker Workshop



## Agenda

- ★ The Dockerfile
- \$ docker build
- Dockerfile Instructions
- ★ Lab 04: Build our first container
- ★ Lab 05: Build more complex images
- Best Practices & Common Mistakes

## The Dockerfile

- ♠ Dockerfile (with capital D)
- ★ It's comprised of instructions that define how to build an image
- ★ These instructions get read one at time, from top to bottom

```
FROM ubuntu:14.04
RUN \
apt-get update && \
apt-get -y install apache2

VOLUME /myvol
ADD index.html /var/www/html/index.html

EXPOSE 80

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

♦ When we build images from Dockerfiles, any other files and directories in the same directory as the Dockerfile were going to get included in the build (build context).

## \$ docker build

```
$ docker build -t name:tag .
$ docker build github.com/creack/docker-firefox
```

- Create a new Docker image following the Dockerfile instructions
- You can specify a Dockerfile in the filesystem or build an image from a Dockerfile stored in GitHub
- ★ Tags are used to manage image versions

#### **FROM**

- ★ Sets the Base Image for subsequent instructions
- The image can be any valid image (usually a container start by pulling an image from the Docker Hub)

FROM ubuntu:15.04

#### **MAINTAINER**

- ★ Add metadata to the docker image
- Used to indicate the image maintainer

FROM ubuntu:15.04
MAINTAINER leonj@sela.co.il

#### **RUN**

- ★ Used to run commands against our images that we're building
- Every run instruction adds a layer to our image
- Run commands are the image "build steps"

FROM ubuntu:15.04
MAINTAINER leonj@sela.co.il
RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get install -y vim
RUN apt-get install -y apache2-utils

#### **CMD**

- ★ Is the command executed anytime we launch a container from this image
- ↑ This command can be overridden in the run command
- ★ Two types of sintax:

Shell: [echo "Hello World"]

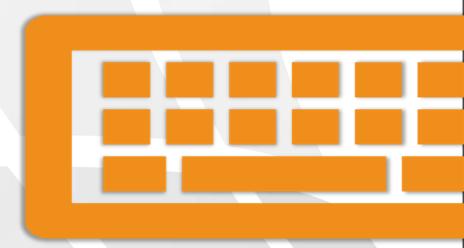
Exec: ["echo", "Hello World"]

FROM ubuntu:15.04
MAINTAINER leonj@sela.co.il
RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get install -y vim
RUN apt-get install -y apache2-utils
CMD ["echo","Hello World!"]

# Questions

## Lab 04: Building our first image

Lab



https://gitlab.com/sela-docker-workshop/lab-04

## **Build Cache**

- Nhen we build a new image the docker deamon iterates through our Dockerfile executing each instruction.
- As each instruction gets executed, the deamon checks to see whether or not it's got an image for that instruction already in its build cache
- ★ The build cache store each instruction + linked image
- (Change the docker file invalidates the build cache)

#### **EXPOSE**

- ★ The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.
- You can expose one port number and publish it externally under another number.

#### **ENTRYPOINT**

- Is the better method of specifying the default app to run inside of a container
- ★ Anything we do specify at the end of the docker run command at runtime (or CMD instruction) get interpreted as arguments to the entrypoint instruction

```
FROM ubuntu:15.04
MAINTAINER leonj@sela.co.il
RUN apt-get update && apt-get install -y \
        apache2 \
        vim \
        apache2-utils
EXPOSE 80
ENTRYPOINT ["echo"]
```

#### **ENV**

Used to assign environment variables

#### **COPY**

- ★ Copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>
- ★ Each <src> may contain wildcards

#### ADD

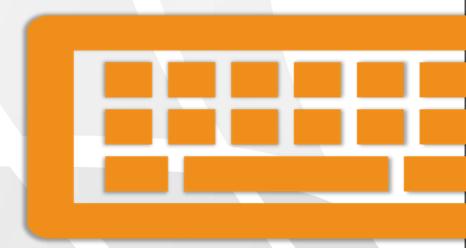
- ★ Similar than Copy but:
  - ADD allows <src> to be an URL
  - If the <src> parameter of ADD is an archive in a recognized compression format or a URL, it will be unpacked

```
FROM ubuntu:15.04
MAINTAINER leonj@sela.co.il
RUN apt-get update && apt-get install -y \
        apache2 \
        vim \
        apache2-utils
ADD test.tar.gz /mydir/
ENV var1=val1 var2=val2
EXPOSE 80
CMD $var1 $var2
ENTRYPOINT ["echo"]
```

# Questions

# Lab 05: Building more complex images

Lab



https://gitlab.com/sela-docker-workshop/lab-05

## Dockerfile Best Practices

- ★ Use a .dockerignore file
- ★ Containers should be immutable & ephemeral
- Minimize the number of layers / Consolidate instructions
- Avoid installing unnecessary packages
- ★ Sort multi-line arguments
- ★ Use Build cache
- ★ Understand CMD and ENTRYPOINT

## Dockerfile Common Mistakes

- Using "latest" tag
- Using external services during the build
- ★ Adding EXPOSE and ENV at the top of your Dockerfile
- ★ Add the app directory at the beginning of the Dockerfile
- ★ Multiple FROM statements
- Multiple services running in the same container