DOCKER

FRANCISCO RODRIGUEZ (AKA CURRO)

- Devops - Sysadmin
- Working at source{d}
- Almost two years with Docker.
- @curratore
- frodriguezd@gmail.com
- curro@sourced.tech
- http://sourced.tech

AGENDA OF THIS WORKSHOP

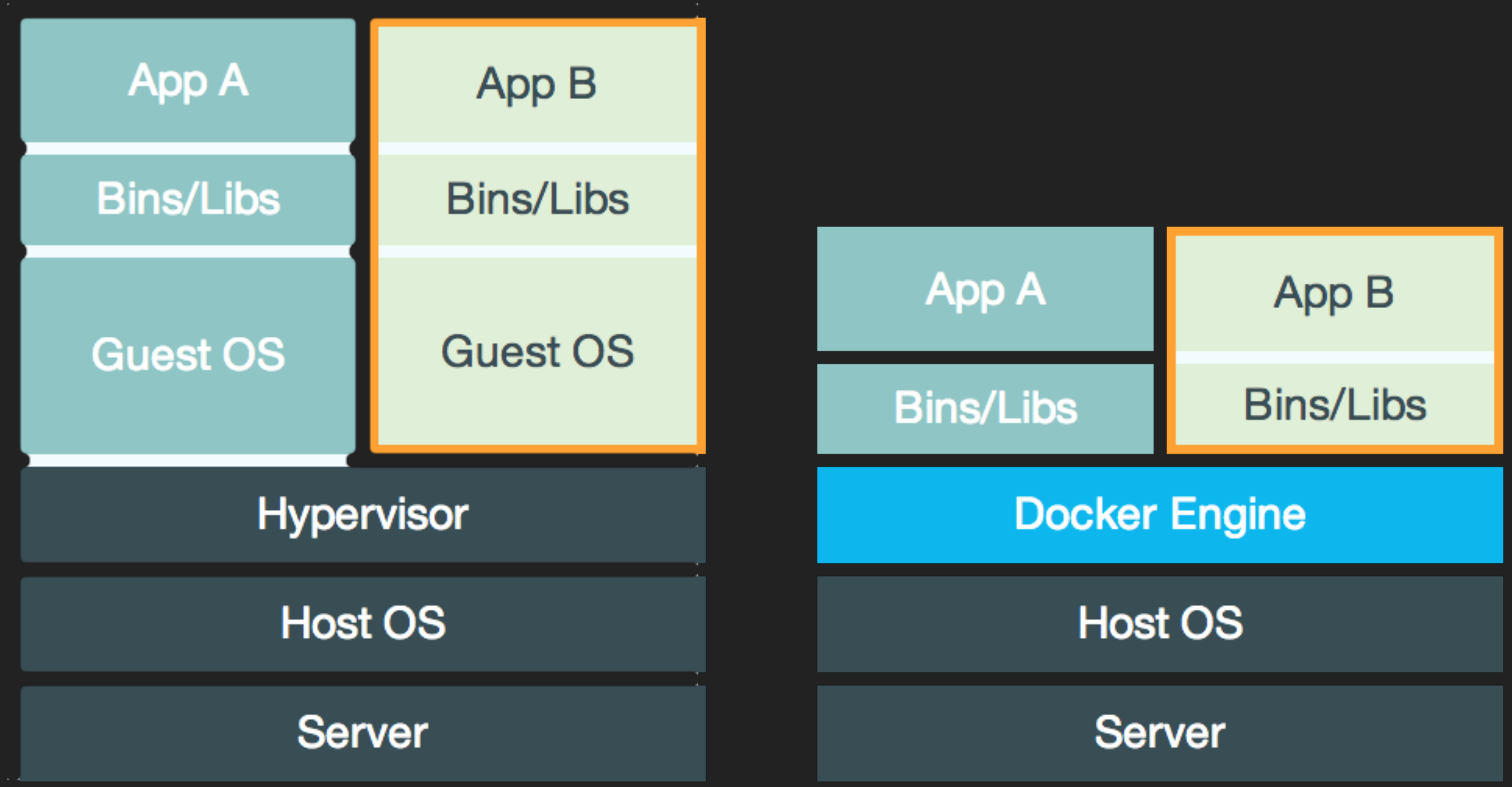
- Introduction
- Docker Platform.
- Docker basic elements.
- Docker Docker Docker.

MILESTONES OF THIS WORKSHOP

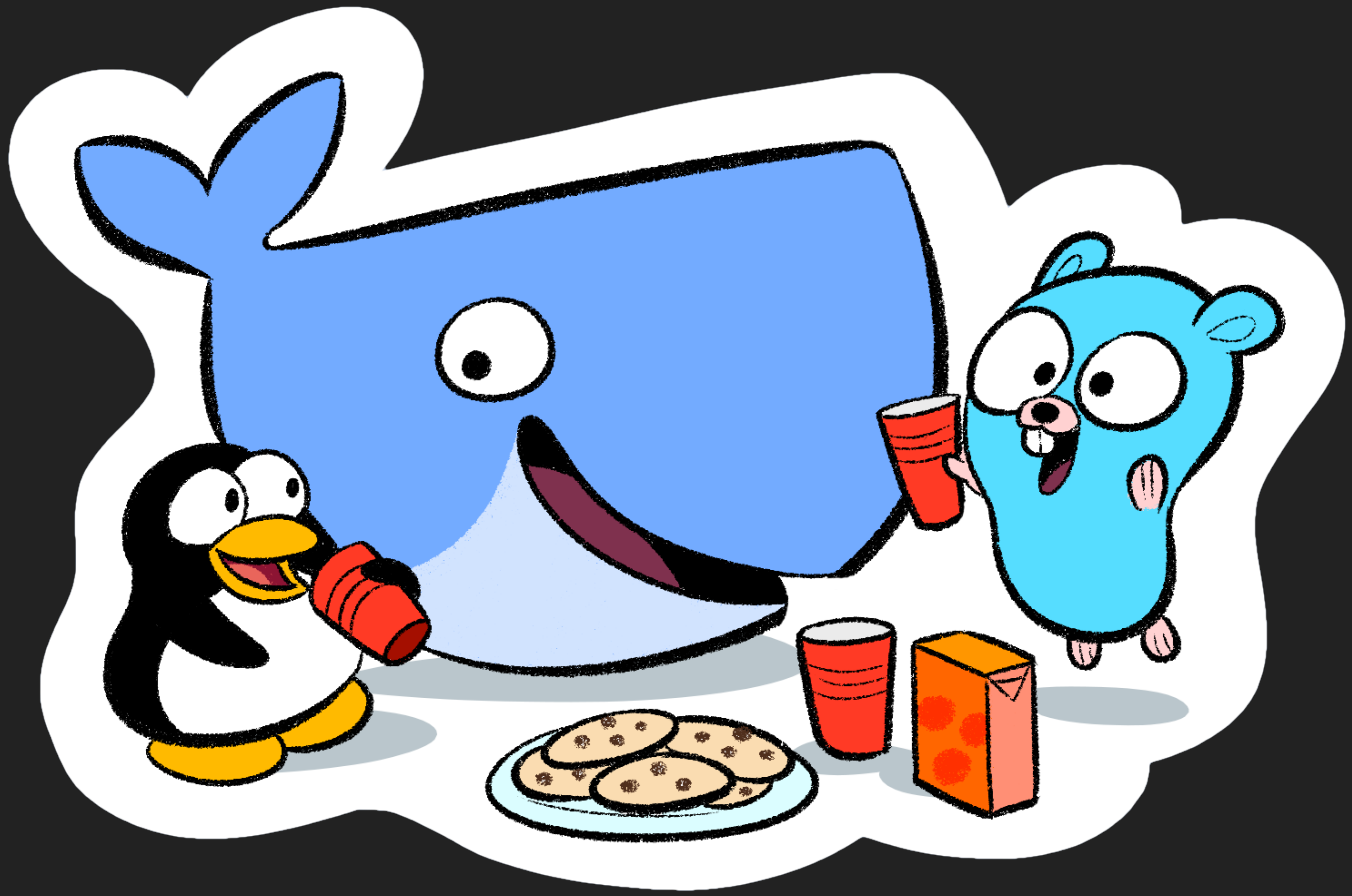
- See how Docker works at a high level.
- Understand the architecture of Docker.
- Know “Docker ecosystem”.
- Discover Docker’s features.
- Learn basic components.
- Create containers and images.
- …

INTRODUCTION – WHAT IS DOCKER?

- Docker is a platform for developers and sysadmins to develop, ship, and run applications. Docker lets you quickly assemble applications from components and eliminates the friction that can come when shipping code. Docker lets you get your code tested and deployed into production as fast as possible.
- Docker allows you to package an application with all of its dependencies.
- Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run.
- Docker is Open Source.



- Docker vs VM eliminates the Hypervisor layer and provides more performance and better isolation.
- Docker is lighter than VMs.
- Docker is faster than VMs.



DOCKER UNDERLYING (FRIENDS)

UNDERLYING TECHNOLOGY

- Docker is written in Go.
- Uses kernel features to achieve his functionality.
    - KERNEL NAMESPACES
    - CGROUPS
    - UNIONFS

DOCKER PLATFORM

- Docker Engine.
- Docker Hub.
- Docker Machine.
- Docker Compose.
- Docker Swarm.
- Kitematic.
- Docker Registry / Docker Trusted Registry

DOCKER ENGINE (DOCKER)

- Lightweight and open source container technology combined with flow for building and containerising your applications.
- Basic and main component of the “docker world”.
- Allows us work with Docker containers, build images, manage network, share images, etc.
- Manage data in containers.
- Network containers.

DOCKER HUB

- A hosted registry service
- Commit images
- Pull images
- Automated builds
- Link with Github and other repositories.

DOCKER REGISTRY AND DTR (DOCKER TRUSTED REGISTRY)

- Docker registries hold images. These are public or private stores from which you upload or download images. The public Docker registry is provided with the Docker Hub
- Docker registries are the distribution component of Docker.
- Docker Trusted Registry holds private images.

DOCKER COMPOSE

- Solution adopted by Docker (Fig)
- Based on orchestration different containers.
- Different configurations.
- Using YAML files.
- Simple CLI.

DOCKER MACHINE

- Allows provision Docker on computer, cloud providers and inside your own data center.
- Create hosts.
- Install Docker on them.
- Configure the docker client to talk to them.
- CLI commands.
- Beta?!?

DOCKER SWARM

- Native clustering for Docker.
- Create and manage pools of Docker hosts.
- Uses Docker API and all Docker tools.
- “Swap, plug and play”
- Swarm API compatible with Docker API.

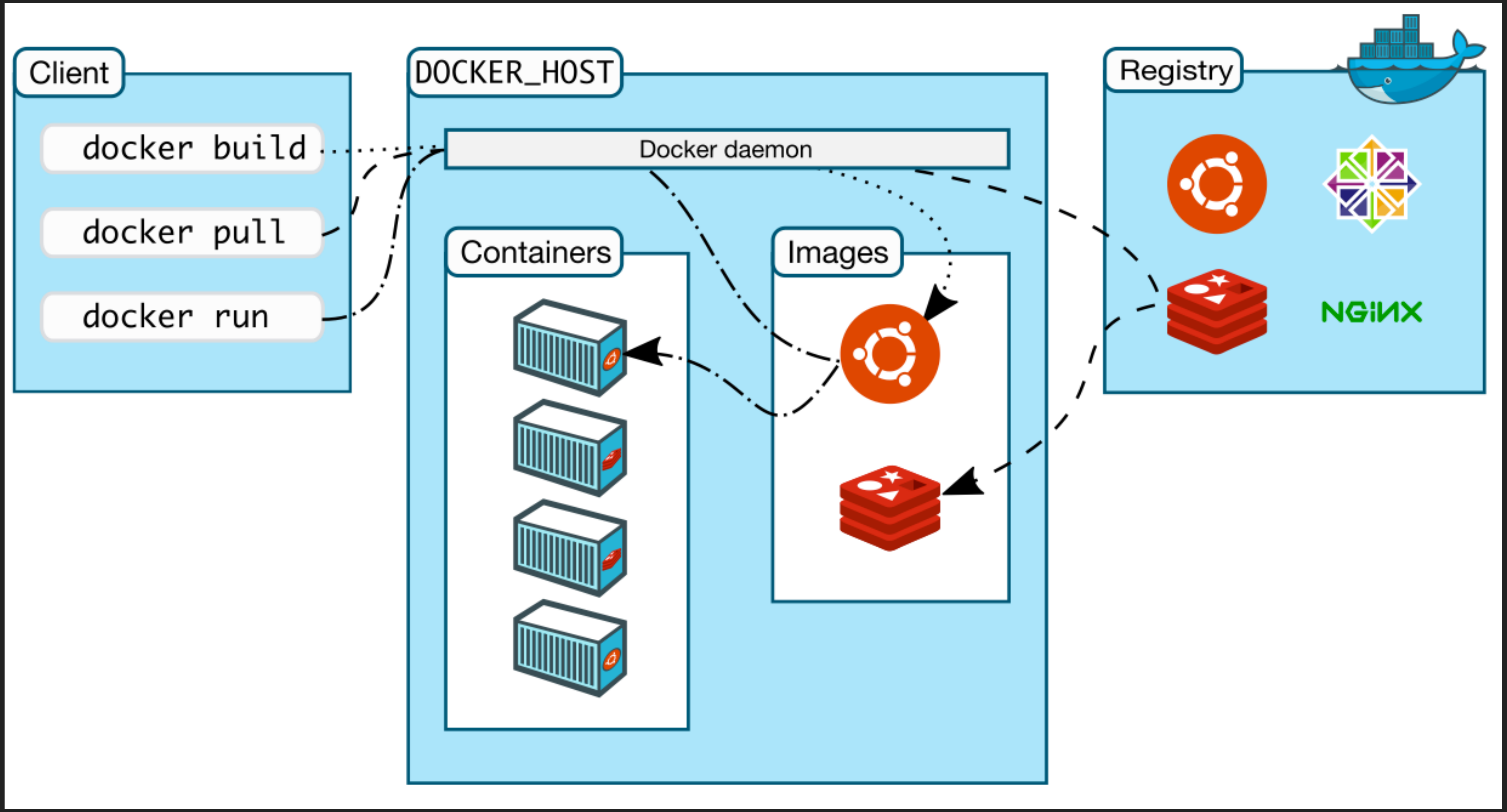
KITEMATIC

- Kitematic is the desktop GUI for Docker.
- Automates the Docker installation and setup the process.
- Integrates with Docker Machine to provision a Vbox VM installing Docker Engine locally.
- Connects with Docker Hub.
- Use buttons to deploy and build.

DOCKER BASIC ELEMENTS

- Docker Images: are the basis of containers.
- Docker Registries: hold public and private images registry maintained by Docker.
- Docker Containers: A Docker container holds everything that is needed for an application to run.

ARCHITECTURE



DOCKER ENGINE

- Docker Engine is the program that enables containers to be built, shipped and run.
- Linux Kernel namespaces and control groups.
- Namespaces give us the isolated workspace that we will call the container.
- Contains everything needed to run your apps.
- Based on one or more images.

- Installation
- Docker CLI.
- Images.
- Containers.
- Dockerfiles.

INITIAL STEPS

- Simple install
- Multiple distributions (Windows and Mac OSX)
- Debian and RHEL
- Sources.
- Latest version 1.9.1

INSTALLATION

- DEMO

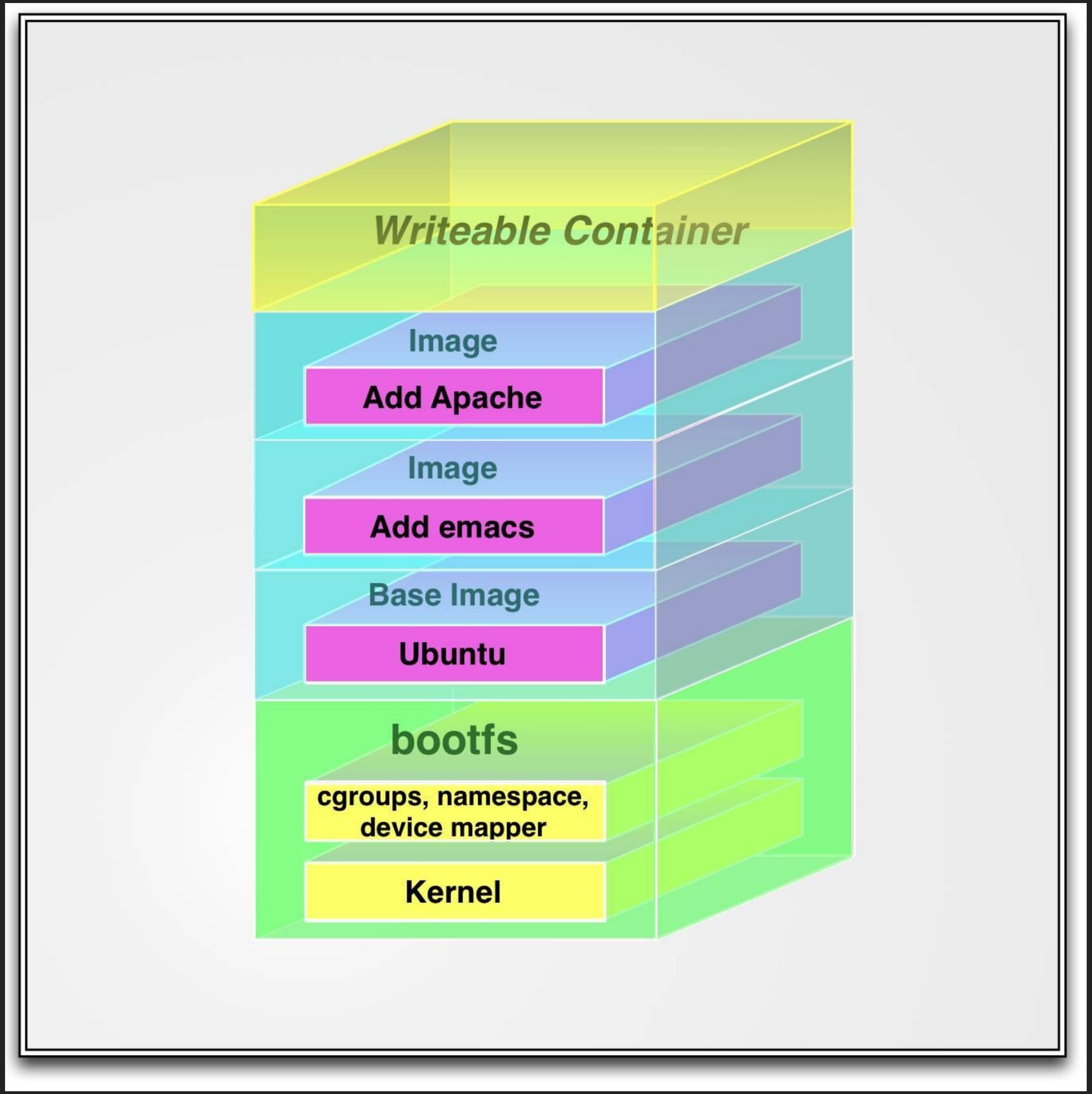
DOCKER CLI

- DEMO

DOCKER IMAGES

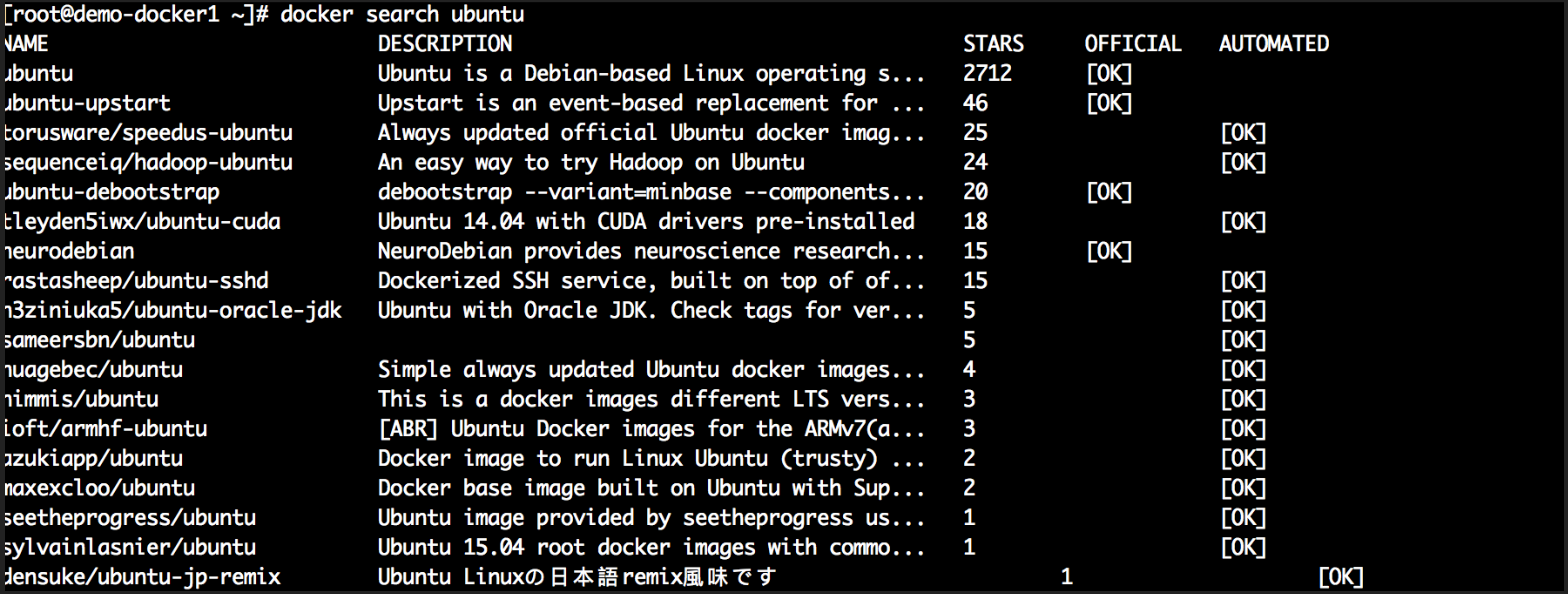
- Read only templates
- Each image consists of a series of layers.
- UnionFS
- Starts from a base layer. (CentOS, Ubuntu, Fedora, Debian, etc.)
- Base layers can be “custom”.

DOCKER IMAGES



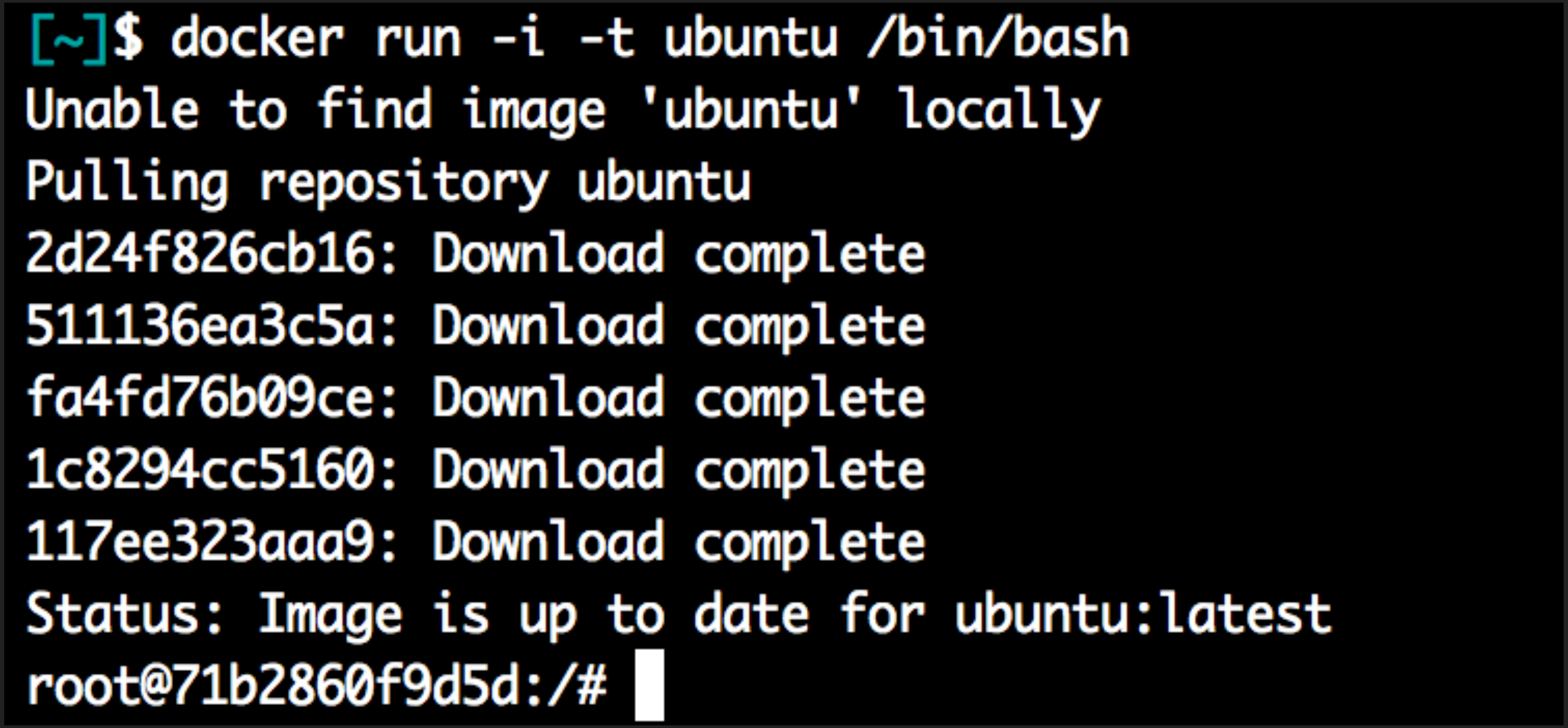
DOCKER IMAGES

- DEMO



```
[root@demo-docker1 ~]# docker search ubuntu
NAME                          DESCRIPTION                                      STARS     OFFICIAL   AUTOMATED
ubuntu                        Ubuntu is a Debian-based Linux operating s...   2712      [OK]
ubuntu-upstart                Upstart is an event-based replacement for ...   46        [OK]
torusware/speedus-ubuntu      Always updated official Ubuntu docker imag...   25                   [OK]
sequenceiq/hadoop-ubuntu      An easy way to try Hadoop on Ubuntu              24                   [OK]
ubuntu-debootstrap            debootstrap --variant=minbase --components...   20        [OK]
tleyden5iwx/ubuntu-cuda       Ubuntu 14.04 with CUDA drivers pre-installed     18                   [OK]
neurodebian                   NeuroDebian provides neuroscience research...   15        [OK]
rastasheep/ubuntu-sshd        Dockerized SSH service, built on top of of...   15                   [OK]
n3ziniuka5/ubuntu-oracle-jdk  Ubuntu with Oracle JDK. Check tags for ver...   5                    [OK]
sameersbn/ubuntu                                                               5                    [OK]
nuagebec/ubuntu               Simple always updated Ubuntu docker images...   4                    [OK]
nimmis/ubuntu                 This is a docker images different LTS vers...   3                    [OK]
ioft/armhf-ubuntu             [ABR] Ubuntu Docker images for the ARMv7(a...   3                    [OK]
azukiapp/ubuntu               Docker image to run Linux Ubuntu (trusty) ...   2                    [OK]
maxexcloo/ubuntu              Docker base image built on Ubuntu with Sup...   2                    [OK]
seetheprogress/ubuntu         Ubuntu image provided by seetheprogress us...   1                    [OK]
sylvainlasnier/ubuntu         Ubuntu 15.04 root docker images with commo...   1                    [OK]
densuke/ubuntu-jp-remix       Ubuntu Linuxの日本語remix風味です                        1                    [OK]
```

DOCKER CONTAINERS



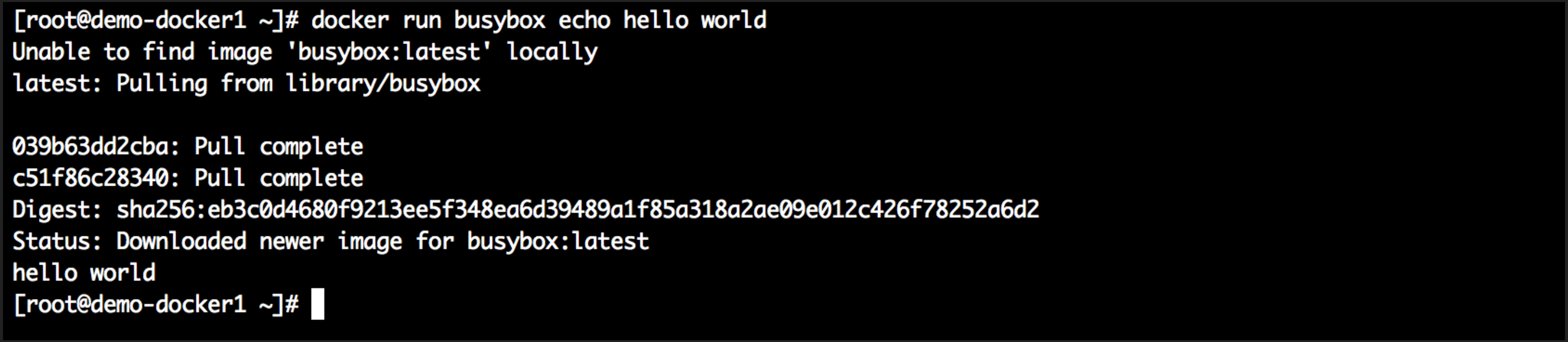
```
[~]$ docker run -i -t ubuntu /bin/bash
Unable to find image 'ubuntu' locally
Pulling repository ubuntu
2d24f826cb16: Download complete
511136ea3c5a: Download complete
fa4fd76b09ce: Download complete
1c8294cc5160: Download complete
117ee323aaa9: Download complete
Status: Image is up to date for ubuntu:latest
root@71b2860f9d5d:/#
```

DOCKER CONTAINERS

- “docker run --i --t ubuntu /bin/bash”
    - -i holds STDIN from the container
    - -t opens a pseudo-tty container
    - ubuntu image name
    - /bin/bash

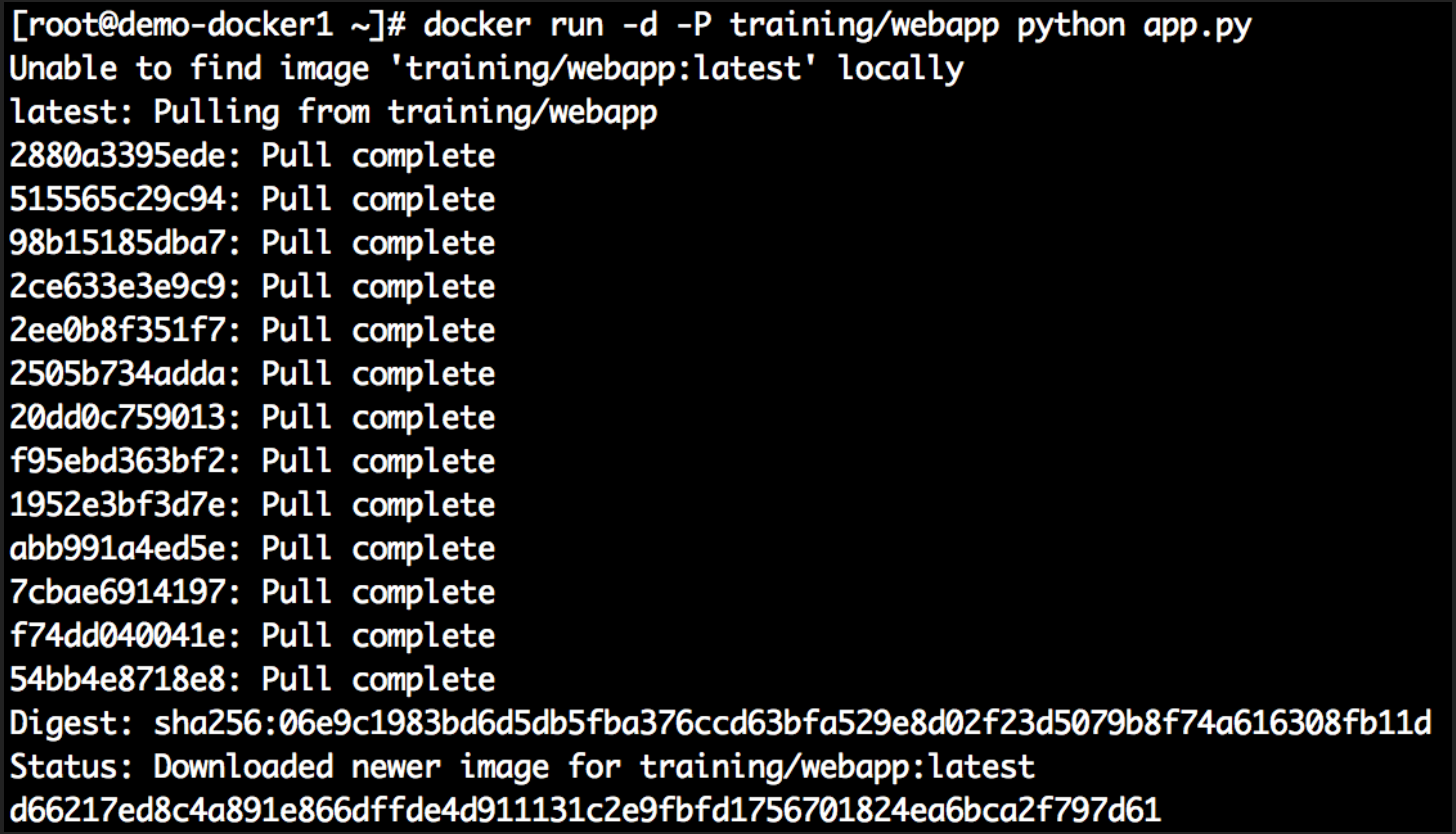
DOCKER CONTAINERS

DEMO HOLA MUNDO



```
[root@demo-docker1 ~]# docker run busybox echo hello world
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox

039b63dd2cba: Pull complete
c51f86c28340: Pull complete
Digest: sha256:eb3c0d4680f9213ee5f348ea6d39489a1f85a318a2ae09e012c426f78252a6d2
Status: Downloaded newer image for busybox:latest
hello world
[root@demo-docker1 ~]# 
```

DOCKER – RUNNING APPLICATIONS



```
[root@demo-docker1 ~]# docker run -d -P training/webapp python app.py
Unable to find image 'training/webapp:latest' locally
latest: Pulling from training/webapp
2880a3395ede: Pull complete
515565c29c94: Pull complete
98b15185dba7: Pull complete
2ce633e3e9c9: Pull complete
2ee0b8f351f7: Pull complete
2505b734adda: Pull complete
20dd0c759013: Pull complete
f95ebd363bf2: Pull complete
1952e3bf3d7e: Pull complete
abb991a4ed5e: Pull complete
7cbae6914197: Pull complete
f74dd040041e: Pull complete
54bb4e8718e8: Pull complete
Digest: sha256:06e9c1983bd6d5db5fba376ccd63bfa529e8d02f23d5079b8f74a616308fb11d
Status: Downloaded newer image for training/webapp:latest
d66217ed8c4a891e866dffde4d911131c2e9fbfd1756701824ea6bca2f797d61
```

DOCKER CONTAINERS - COMMANDS

- docker version
- docker ps
- docker logs
- docker port
- docker stop
- docker start

DOCKER CONTAINERS – MORE COMMANDS

- docker inspect
- docker attach
- docker exec
- docker kill
- docker network
- docker rm

DOCKER IMAGES

- Search.
- Listing images. (host)
- Get images
- Building images
- Destroying images

DOCKER IMAGES - COMMANDS

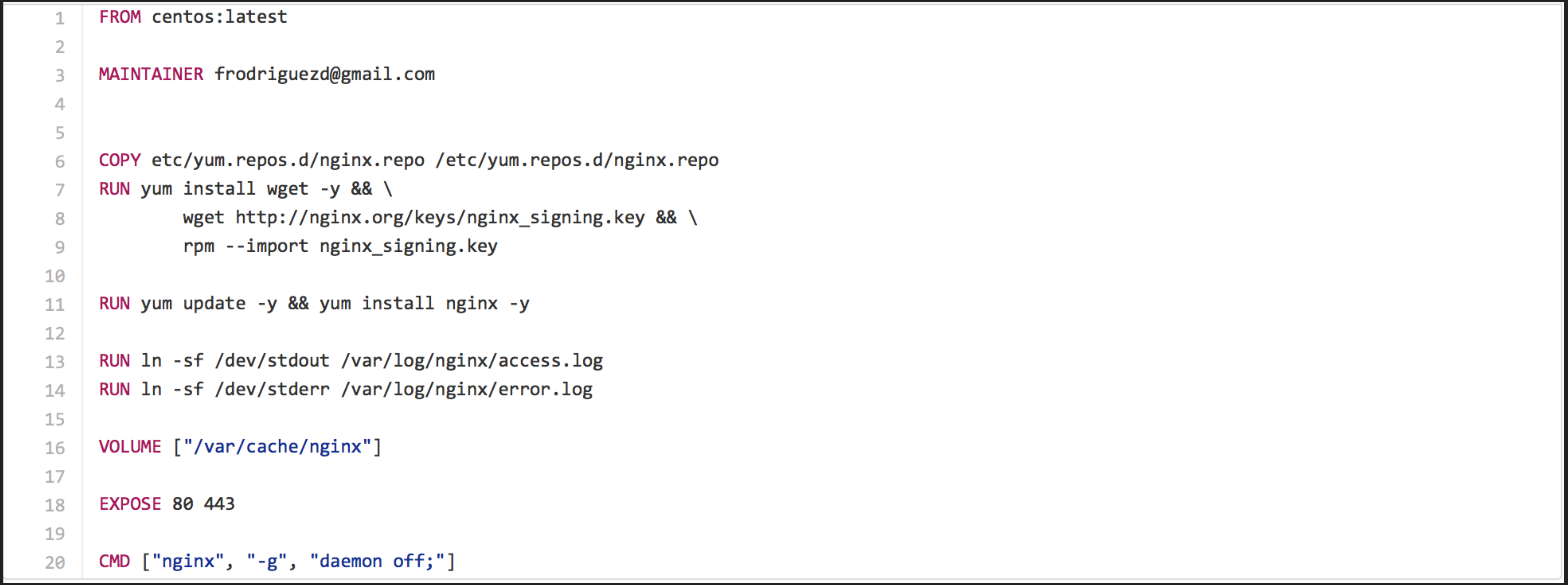
- docker pull
- docker commit
- docker push
- docker tag
- docker history

DOCKER IMAGES - DOCKERFILES

- Dockerfiles are text document that contains instructions
- Docker can build images automatically reading dockerfiles
- Docker build is done by the Daemon not by the CLI
- Dockerfiles uses a DSL (Domain Specific Language).

DOCKER IMAGES- DOCKERFILES

- DEMO



```
FROM centos:latest

MAINTAINER frodriguezd@gmail.com


COPY etc/yum.repos.d/nginx.repo /etc/yum.repos.d/nginx.repo
RUN yum install wget -y && \
        wget http://nginx.org/keys/nginx_signing.key && \
        rpm --import nginx_signing.key

RUN yum update -y && yum install nginx -y

RUN ln -sf /dev/stdout /var/log/nginx/access.log
RUN ln -sf /dev/stderr /var/log/nginx/error.log

VOLUME ["/var/cache/nginx"]

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

DOCKER IMAGES - DOCKERFILES

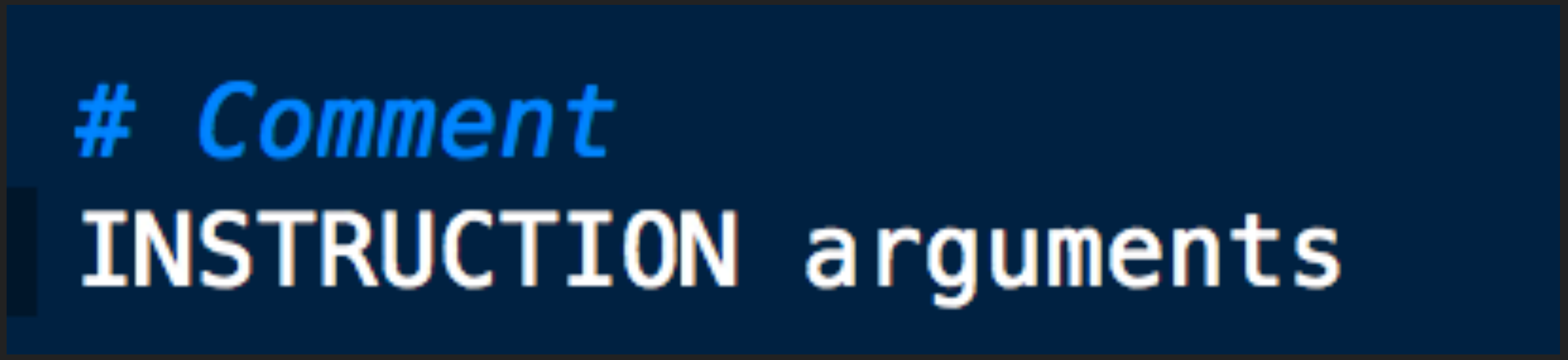
- Easy to repeat.
- Control.
- Transparency
- Easy to maintain
- Clean
- Git.

DOCKER IMAGES - BEST PRACTICES

- Must be ephemeral.
- Use .dockerignore file (similar to .gitignore)
- Avoid install unnecessary packages.
- One process by container.
- Minimize number of layers
- Sort multi-line arguments

DOCKER IMAGES - DOCKERFILES

- No capital letter sensitive.
- Using caps by convention.
- Comments started by #



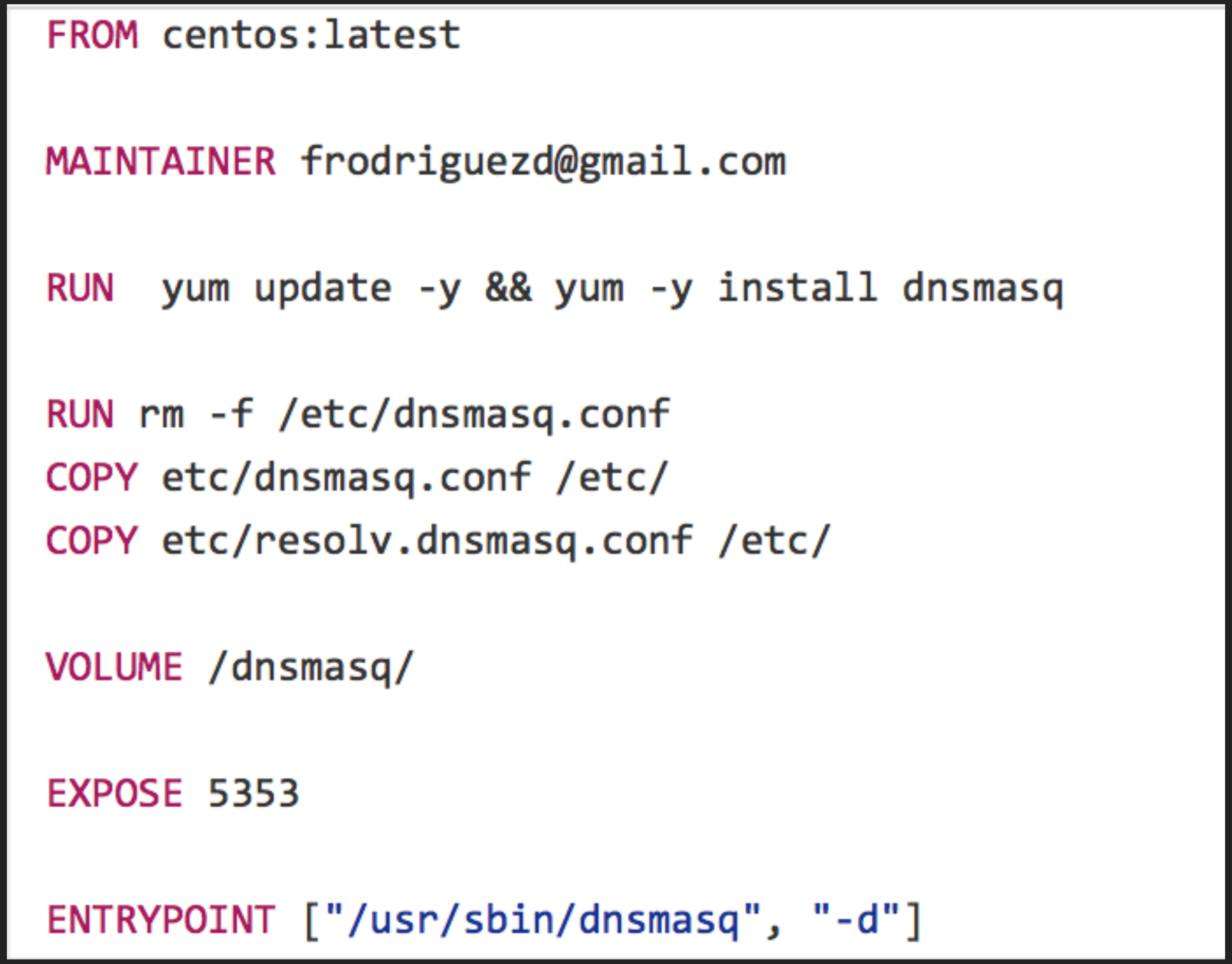
```
# Comment
INSTRUCTION arguments
```

DOCKER IMAGES – DOCKERFILES

- FROM
- MAINTAINER
- RUN
- CMD
- LABEL
- EXPOSE
- ENV
- ADD
- COPY
- ENTRYPOINT
- VOLUME
- USER
- WORKDIR

DOCKER CONTAINERS - MORE COMMANDS

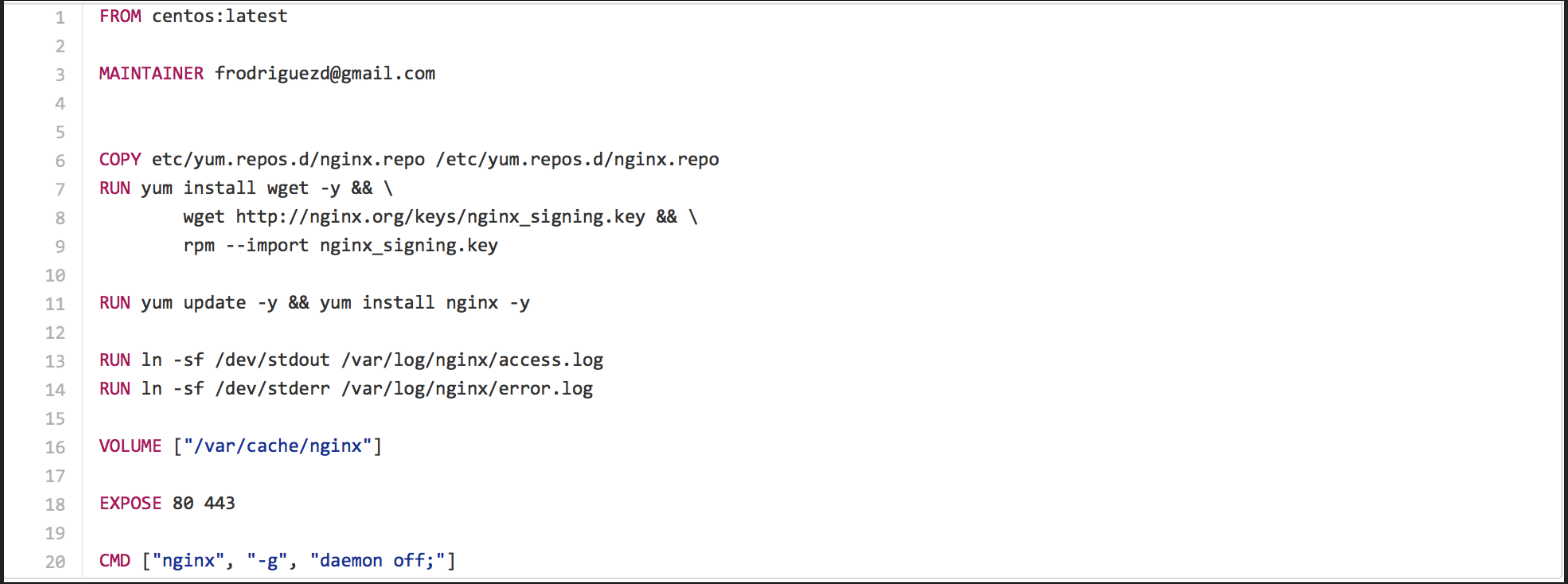
- The first instruction must be `FROM` in order to specify the Base Image from which you are building.



```
FROM centos:latest

MAINTAINER frodriguezd@gmail.com

RUN  yum update -y && yum -y install dnsmasq

RUN rm -f /etc/dnsmasq.conf
COPY etc/dnsmasq.conf /etc/
COPY etc/resolv.dnsmasq.conf /etc/

VOLUME /dnsmasq/

EXPOSE 5353

ENTRYPOINT ["/usr/sbin/dnsmasq", "-d"]
```

DOCKER IMAGES – DOCKERFILES

- MAINTAINER



```
FROM centos:latest

MAINTAINER frodriguezd@gmail.com


COPY etc/yum.repos.d/nginx.repo /etc/yum.repos.d/nginx.repo
RUN yum install wget -y && \
        wget http://nginx.org/keys/nginx_signing.key && \
        rpm --import nginx_signing.key

RUN yum update -y && yum install nginx -y

RUN ln -sf /dev/stdout /var/log/nginx/access.log
RUN ln -sf /dev/stderr /var/log/nginx/error.log

VOLUME ["/var/cache/nginx"]

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

DOCKER IMAGES – DOCKERFILES

- RUN instruction will execute any commands in a new layer on top of the current image and commit the results.
- RUN <command> (the command is run in a shell - /bin/sh -c - shell form)
- RUN ["executable", "param1", "param2"] (exec form)
- USE A DIFFERENT SHELL   RUN ["/BIN/BASH", "-C", "ECHO HELLO"]
- THE EXEC FORM IS PARSED AS A JSON ARRAY
- UNLIKE THE SHELL FORM, THE EXEC FORM DOES NOT INVOKE A COMMAND SHELL.

DOCKER IMAGES - DOCKERFILES

- The main purpose of a CMD is to provide defaults for an executing container.
- Can include an executable or can omit it. (ENTRYPOINT)
- Provides default arguments for ENTRYPOINT
- CMD and ENTRYPOINT instructions should be specified with the JSON array format.
- JSON array format uses double-quotes.

DOCKER IMAGES - DOCKERFILES

- CMD ["executable","param1","param2"] (exec form, this is the preferred form)
- CMD ["param1","param2"] (as default parameters to ENTRYPOINT)
- CMD command param1 param2 (shell form)

DOCKER IMAGES - DOCKERFILES

- Shell form

```
FROM ubuntu
CMD echo "This is a test." | wc -
```

- Exec form

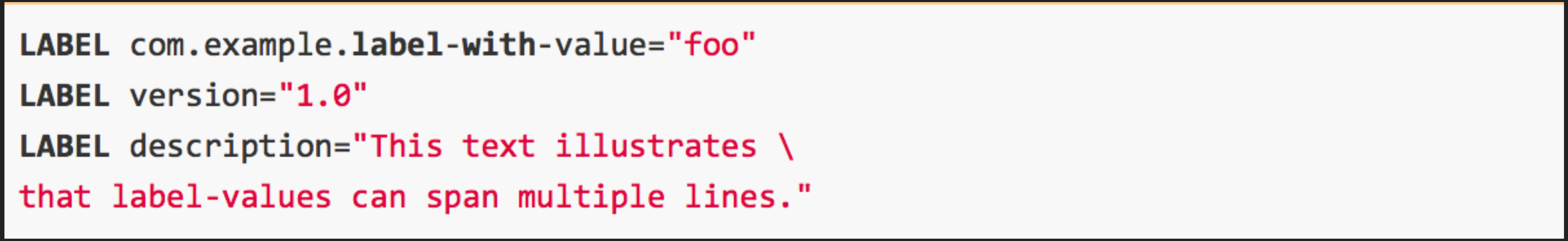
```
FROM ubuntu
CMD ["/usr/bin/wc","--help"]
```

DOCKER IMAGES - DOCKERFILES

- LABEL instruction adds metadata to an image.
- LABEL is a key value pair.
- Each LABEL instruction produces a new layer.
- LABEL <KEY>=<VALUE> <KEY>=<VALUE> <KEY>=<VALUE> ...

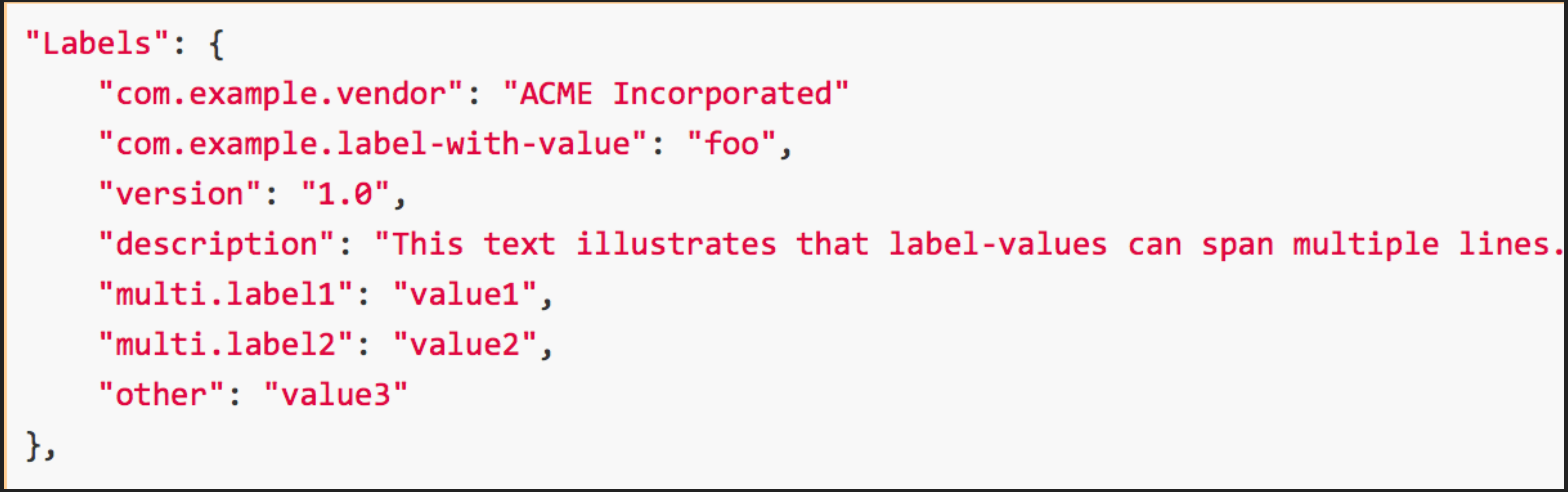
DOCKER IMAGES - DOCKERFILES

- Multiple Labels



```
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

- docker inspect



```
"Labels": {
    "com.example.vendor": "ACME Incorporated"
    "com.example.label-with-value": "foo",
    "version": "1.0",
    "description": "This text illustrates that label-values can span multiple lines.
    "multi.label1": "value1",
    "multi.label2": "value2",
    "other": "value3"
},
```

DOCKER IMAGES – DOCKERFILES

- EXPOSE <port> [<port>...]
- Informs to Docker that the container will listen on the specified network ports at runtime.
- EXPOSE doesn’t define which ports can be exposed to the host or make ports accessible from the host by default.

DOCKER IMAGES - DOCKERFILES

- ENV <key> <value>
- ENV <key>=<value> ...
- The ENV instruction sets the environment variable <key> to the value <value>.
- The environment variables set using ENV will persist when a container is run from the resulting image.

DOCKER IMAGES - DOCKERFILES

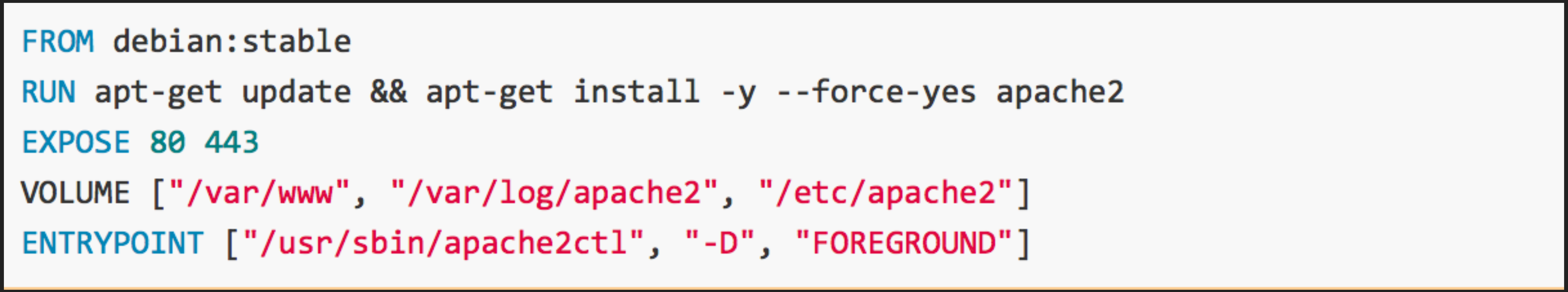
- ADD <src>... <dest>
- ADD ["<src>",... "<dest>"] (this form is required for paths containing whitespace)
- Copies directories, files and/or remote URLs
- The first encountered ADD instruction will invalidate the cache for all following instructions from the Dockerfile if the contents of <src> have changed.
- Obeys different rules

DOCKER IMAGES - DOCKERFILES

- COPY <src>... <dest>
- COPY ["<src>",... "<dest>"] (this form is required for paths containing whitespace)
- The COPY instruction copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.

DOCKER IMAGES - DOCKERFILES

- ENTRYPOINT ["executable", "param1", "param2"] (the preferred exec form)
- ENTRYPOINT command param1 param2 (shell form)
- Allows you to configure a container that will run as an executable.



```
FROM debian:stable
RUN apt-get update && apt-get install -y --force-yes apache2
EXPOSE 80 443
VOLUME ["/var/www", "/var/log/apache2", "/etc/apache2"]
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

DOCKER IMAGES - DOCKERFILES

- VOLUME creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.
- Json array format.
- Plain string.

```
VOLUME ["/var/log/"]
VOLUME /var/log /var/db
```

```
FROM ubuntu
RUN mkdir /myvol
RUN echo "hello world" > /myvol/greating
VOLUME /myvol
```

DOCKER IMAGES - DOCKERFILES

- USER foo
- USER instruction sets the user name or UID to use when running the image and for any RUN, CMD and ENTRYPOINT instructions that follow it in the Dockerfile.
- WORKDIR /path/to/workdir
- The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile.

DOCKER COMPOSE

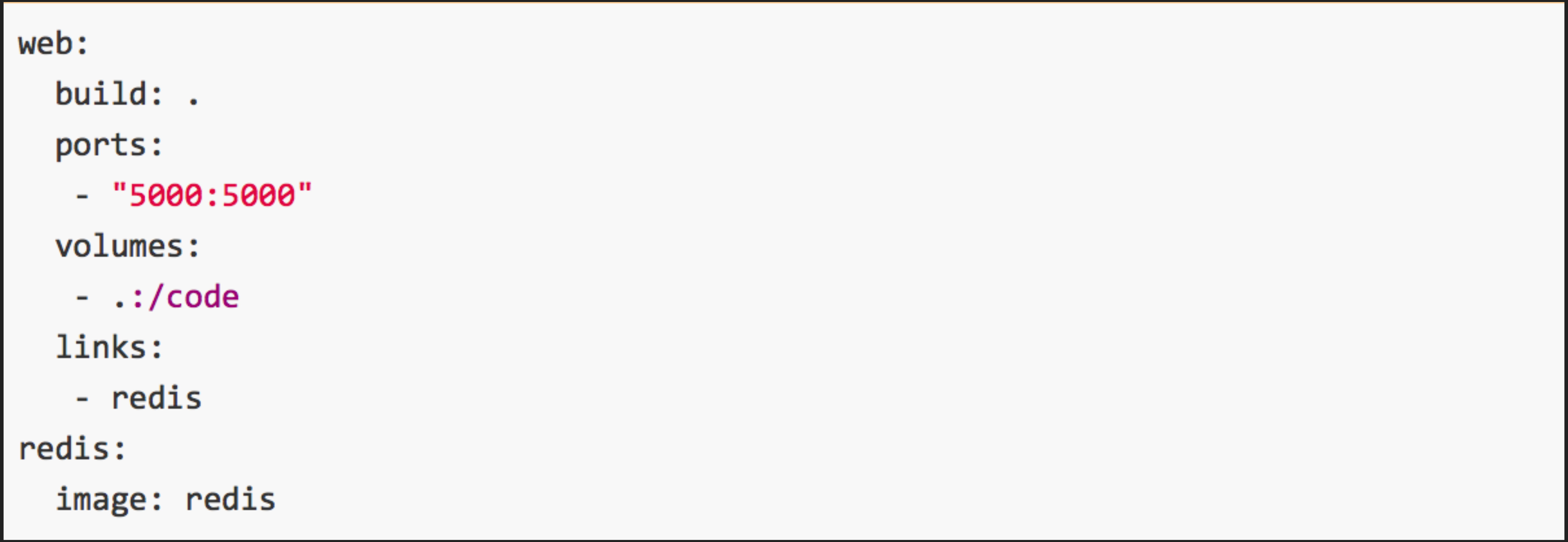
- Compose is a tool for defining and running multi-container applications with Docker. With Compose, you define a multi-container application in a single file, then spin your application up in a single command which does everything that needs to be done to get it running.
- **COMPOSE IS GREAT FOR DEVELOPMENT ENVIRONMENTS, STAGING SERVERS, AND CI. WE DON'T RECOMMEND THAT YOU USE IT IN PRODUCTION YET.**

DOCKER COMPOSE

- Define your app’s environment with a Dockerfile so it can be reproduced anywhere.
- Define the services that make up your app in “docker-compose.yml” so they can be run together in an isolated environment:
- Lastly, run “docker-compose” up and Compose will start and run your entire app.

DOCKER COMPOSE

- docker-compose.yml



```
web:
  build: .
  ports:
   - "5000:5000"
  volumes:
   - .:/code
  links:
   - redis
redis:
  image: redis
```

QUESTIONS & THANKS

- Questions.
- Thanks you.