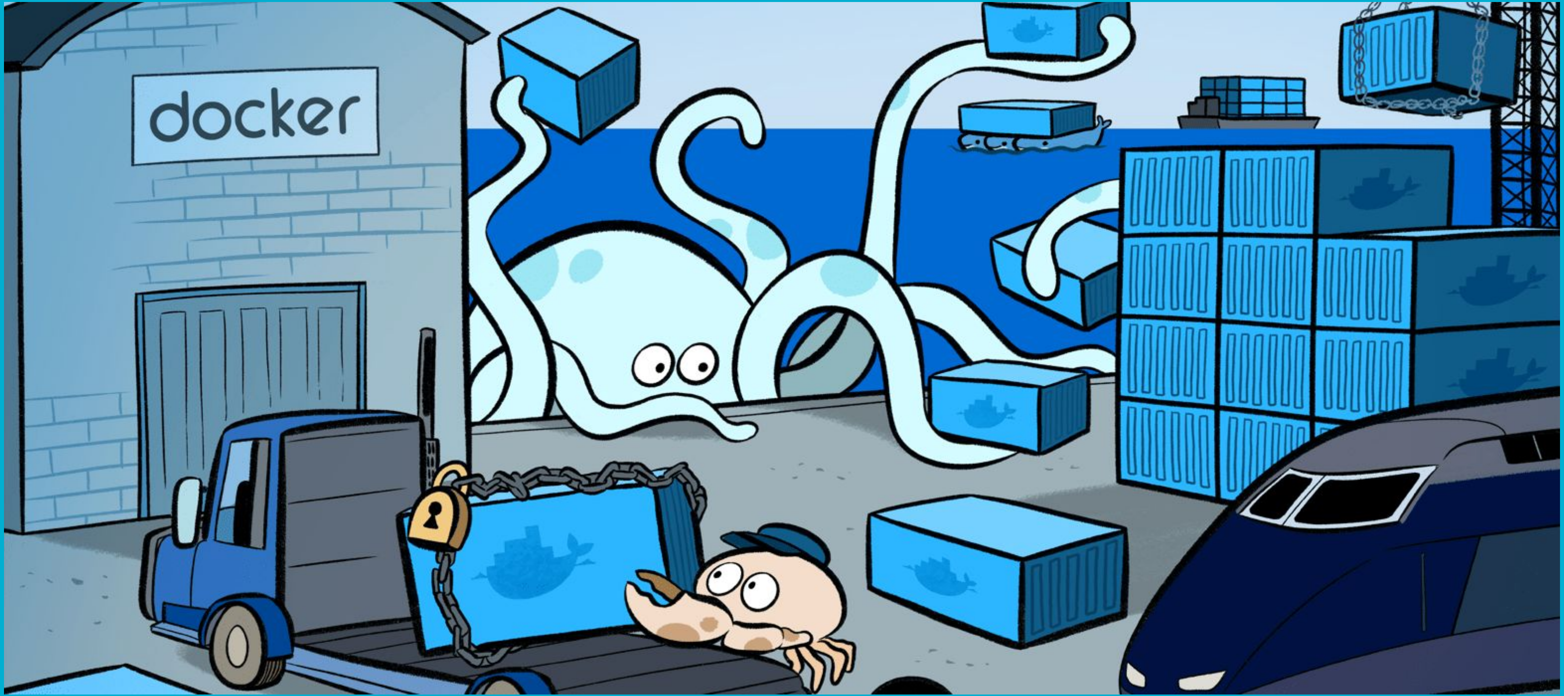
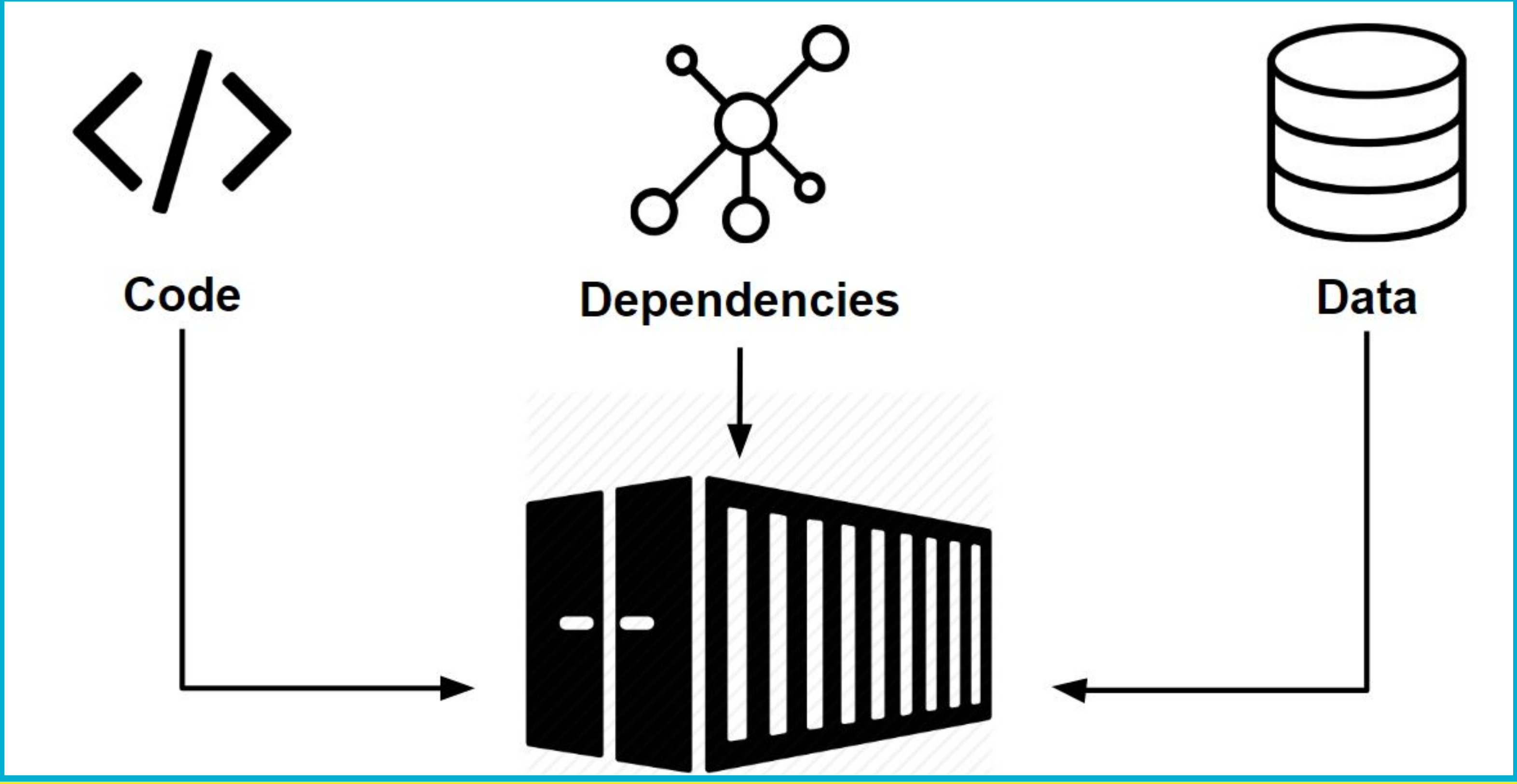
Introduction to Docker

Docker allows us to package and run applications in an isolated environment

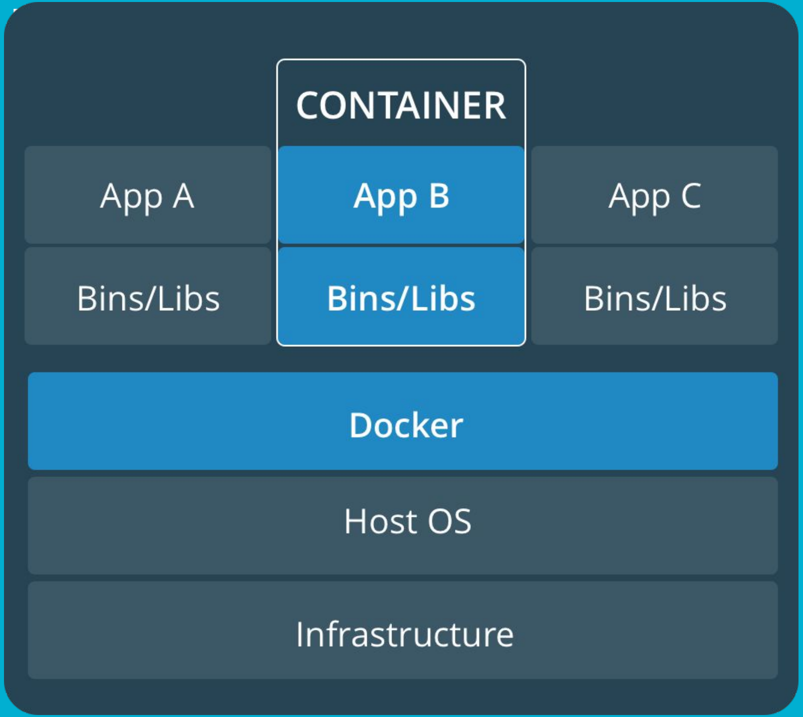
Shipping Container Analogy

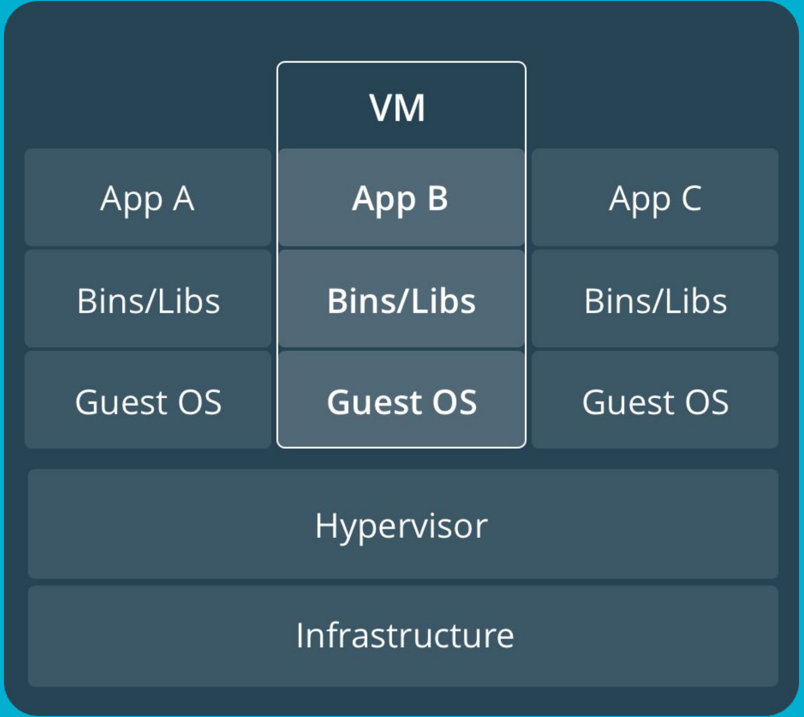


Software Containers

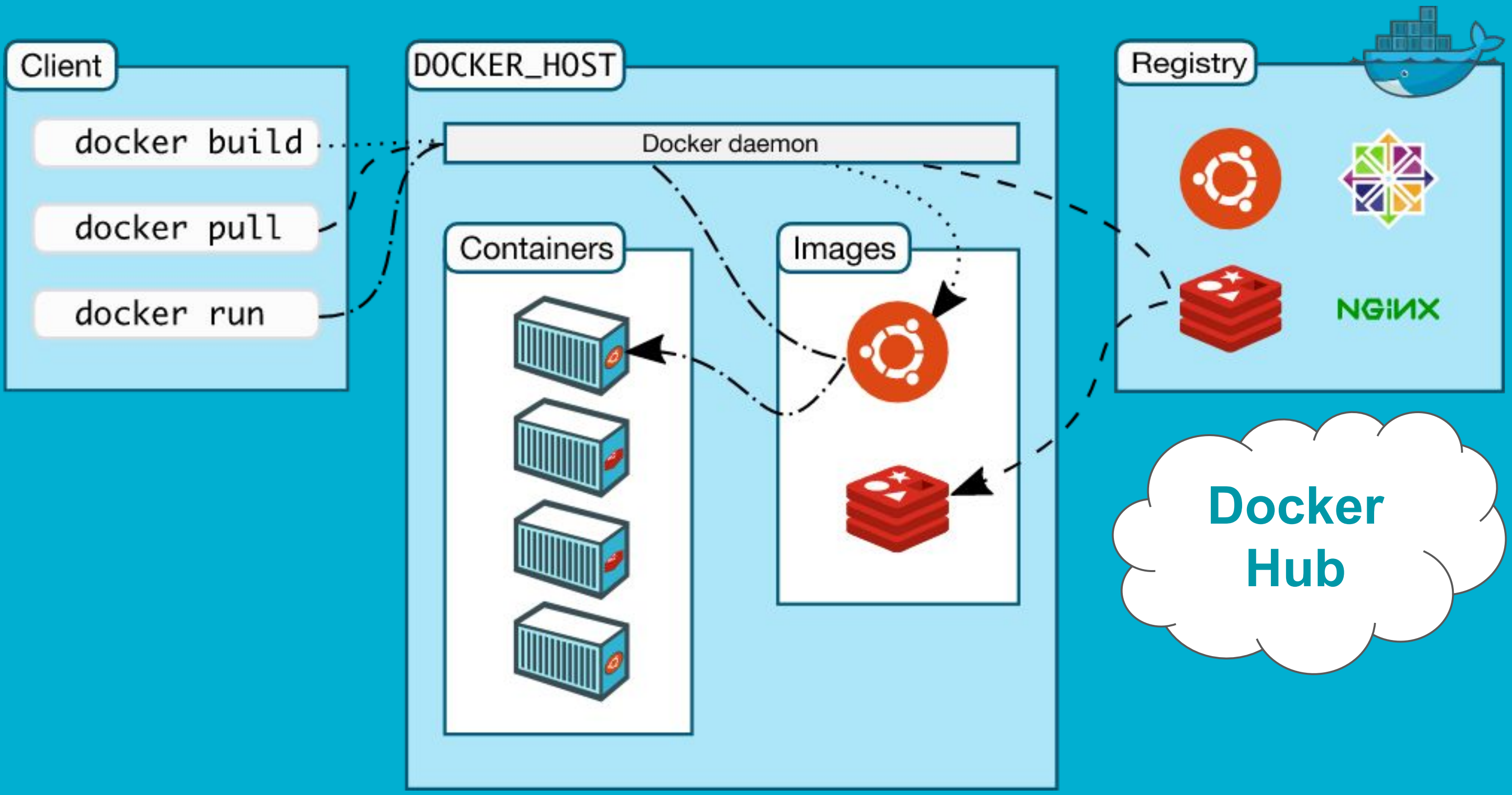


Docker Containers vs Virtual Machines



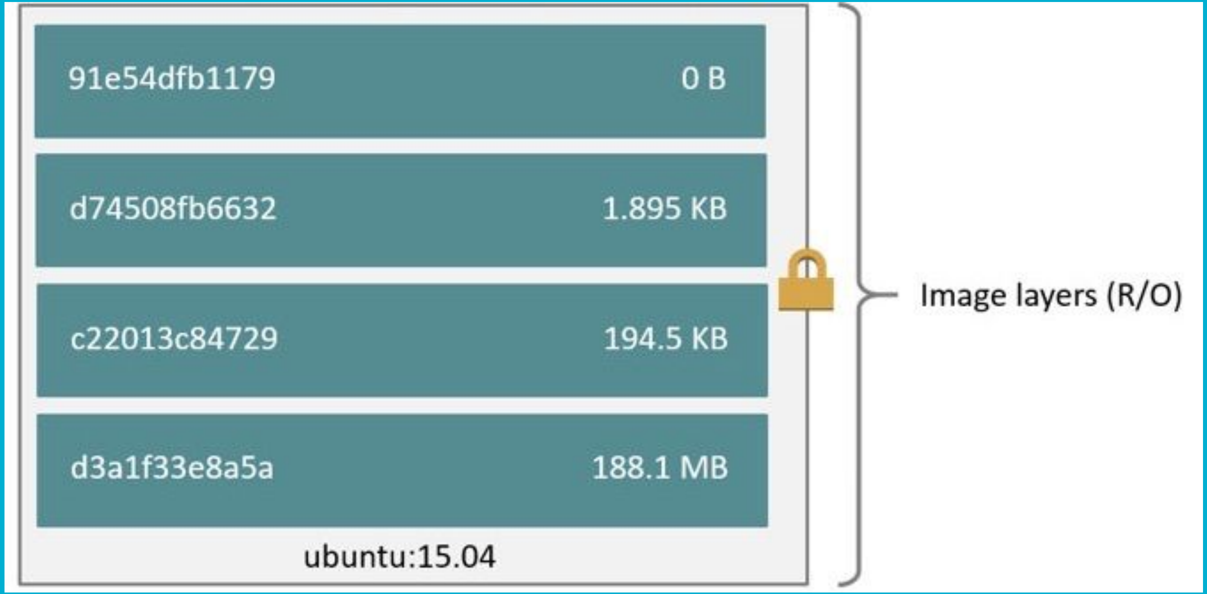


Docker Architecture: Overview



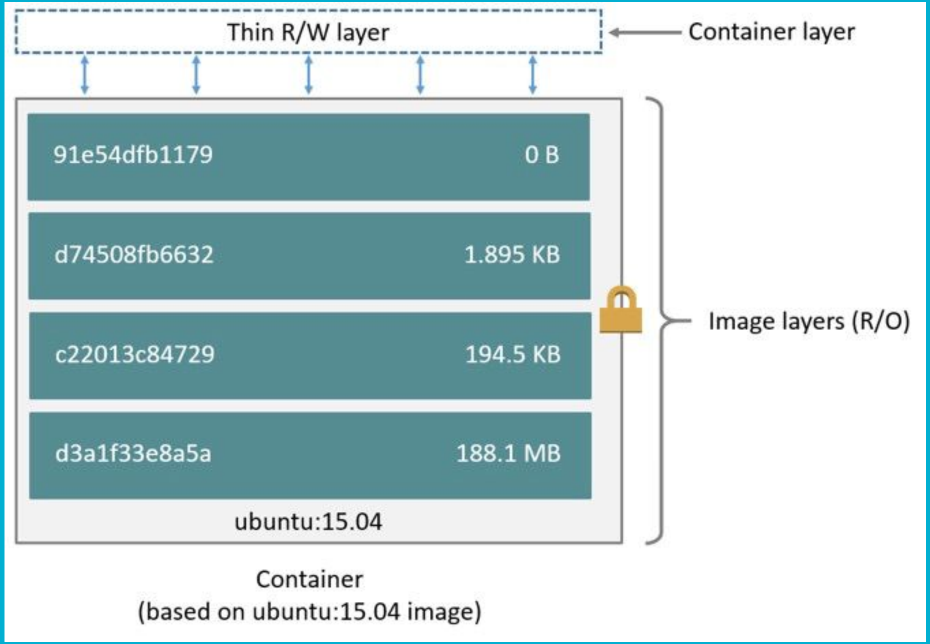
Docker Image

- A frozen snapshot of a container



Docker Containers

Runtime instance: docker run [image]



Object-Oriented Programming Analogy

- Images : Classes
- Layers : Inheritance
- Containers : Objects

Creating Docker Images

1. Freeze container using `docker commit`

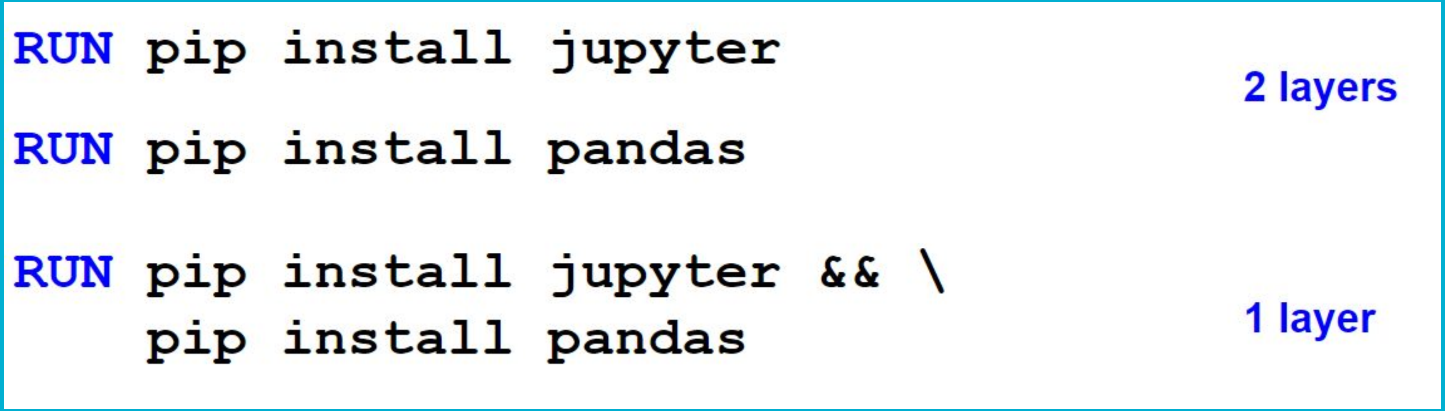
2. Dockerfile and `docker build` *** Preferred ***
    - File containing all commands used to assemble image
    - Automated build

Dockerfile Commands

- FROM - sets base image
- LABEL - adds metadata to image
  - MAINTAINER is deprecated
  - LABEL maintainer=”Aly Sivji <alysivji@gmail.com>”
- COPY - copies files / directories into image
  - .dockerignore
- ENV - sets environment variable
- WORKDIR - sets working directory

Dockerfile Commands

RUN - executes shell commands in a new layer



```
RUN pip install jupyter
RUN pip install pandas

RUN pip install jupyter && \
    pip install pandas
```

Dockerfile - Configuring Runtime

- ENTRYPOINT - configures container to run as executable
- CMD - provides default for executing container
  - CMD and ENTRYPOINT interaction
  -
- Two forms:
  - Shell `CMD python hello-world.py`
  - Exec (preferred) `CMD [“python”, “hello-world.py”]`
  -
- Additional Information

Hello World Dockerfile

```
# Use latest Python runtime as base image
FROM python:3.6.5-alpine3.7

# Set the working directory to /app and copy current dir
WORKDIR /app
COPY . /app

# Run hello_world.py when the container launches
CMD ["python", "hello_world.py"]
```

Building Image

```
$ docker build -t hello-world .

Sending build context to Docker daemon  3.072kB

Step 1/4 : FROM python:3.6.5-alpine3.7

…

Successfully built 1048f0a224fc

Successfully tagged hello-world:latest
```

Container Commands

- Create Container

```
$ docker run hello-world

Hello World!
```

- Restart Container
  - `$ docker start -ia [CONTAINER]`

`$ docker run [OPTIONS] IMAGE [COMMAND]`

- `[Options]`

`-d`	Detached (runs in background)
`-a`	Attach to STDOUT/STDERR
`-i`	Attach STDIN
`-t`	Allocates pseudo-TTY
`--name [NAME]`	Set the container name

- `[Command]`
    - Can pass in parameters or **`/bin/sh`** to get into container’s shell

Managing Data Inside Containers

- Data disappears when we delete a container
- `docker cp` to copy files in/out of containers
- Mount data volume inside container

Adding Data Volume to Container

```
$ docker run -v /full/local/path:/mounted_dir
```

Host Path

Container Path

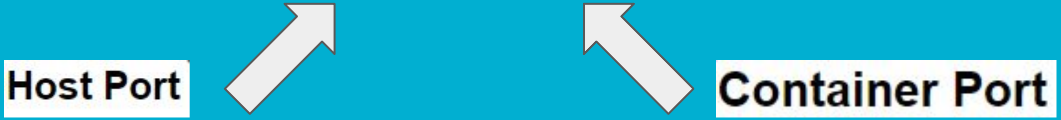
- Best Practice: Add VOLUME command to Dockerfile

```
# Create mount point for external volumes
VOLUME /mounted_dir
```

Binding Ports

- Setup port forwarding to connect to containers

```
$ docker run -p 9999:8888
```



- Best Practice: Add EXPOSE command to Dockerfile

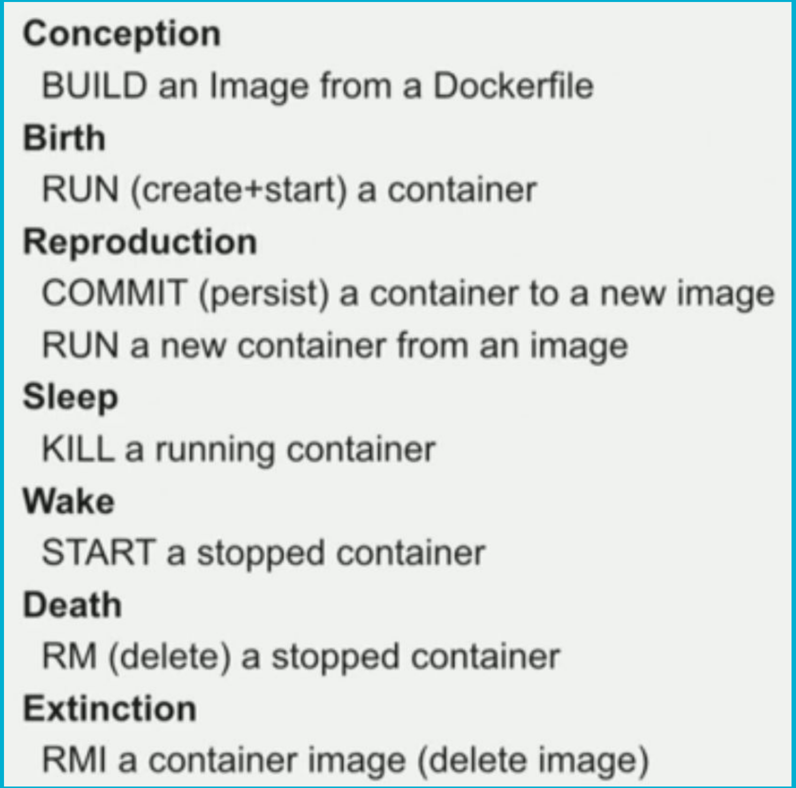
```
# Make port 8888 available to outside world
```

- `EXPOSE 8888`

Dockerfile - Best Practices

- Be explicit about build process
- Containers should be stateless
- Use `.dockerignore` file
- Avoid installing unnecessary packages
  - Clean cache after installation
- Each container should have only one concern / purpose
- Minimize the number of layers
  - Multi-line arguments, sort alphabetically
- CMD should be used to run processes inside container
  - Advanced users should use it in conjunction with ENTRYPOINT
- MAINTAINER is deprecated; use LABEL

Docker Container Lifecycle



Conception

BUILD an Image from a Dockerfile

Birth

RUN (create+start) a container

Reproduction

COMMIT (persist) a container to a new image

RUN a new container from an image

Sleep

KILL a running container

Wake

START a stopped container

Death

RM (delete) a stopped container

Extinction

RMI a container image (delete image)

Docker Commands: Images

Lifecycle

- ☆ docker images
- docker import
- ☆ docker build
- docker commit
- ☆ docker rmi
- docker load
- docker save

Info

- docker history
- docker tag

Registry

- docker login
- docker logout
- docker search
- ☆ docker pull
- ☆ docker push

Docker Commands: Containers

Lifecycle

```
  docker create
  docker rename
☆ docker run
☆ docker rm
  docker update
```

Misc

```
  docker cp
  docker export
☆ docker exec
```

Start/Stop

```
☆ docker start
  docker stop
  docker restart
  docker pause
  docker unpause
  docker wait
☆ docker kill
☆ docker attach
```

Info

```
☆ docker ps
  docker logs
☆ docker inspect
  docker events
  docker port
☆ docker top
  docker stats
  docker diff
```

Source: Docker Cheat Sheet

Tips and Tricks

- Smaller images are better. Install only the packages you need.
  - Look into different Linux distributions (Alpine Linux... only 5MB!)
  - Clear cache after installing or use no-cache flags!
- Link bash_history and keep track of commands typed inside container
- dockviz command line app to visualize docker data
- Ctrl + P + Q to detach from container while inside shell
- Instructions on mounting symbolic links
- Always set IP address for apps running inside container to `0.0.0.0`

Next Steps & Additional Resources

- How to Install Docker
- Docker Documentation: Getting Started Guide
- Nigel Poulton's Docker Deep Dive Course
- CenturyLink Developer Center
- Docker for Data Science PyCon Tutorial