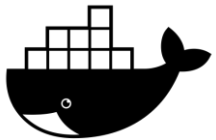
UBC
LAUNCH
PAD

A DEVELOPER'S GUIDE TO DOCKER



WORKSHOP

BRUNO BACHMANN, ROBERT LIN

Slides!

slides.ubclaunchpad.com/workshops/guide-to-docker.pdf

Install Docker!

docs.docker.com/install/#supported-platforms

Why Containers

You're building a basic web app...

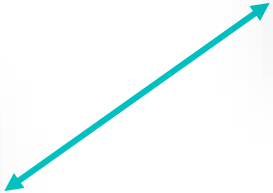
APP

DATABASE

Why Containers

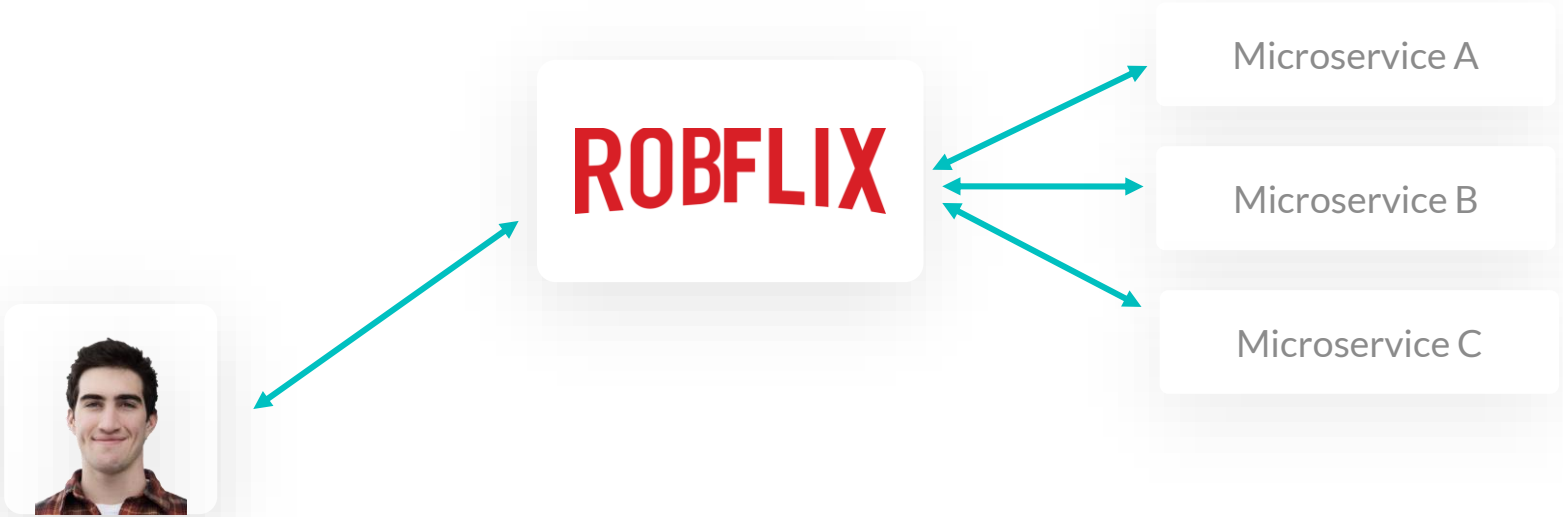
You're providing a service...

ROBFLIX



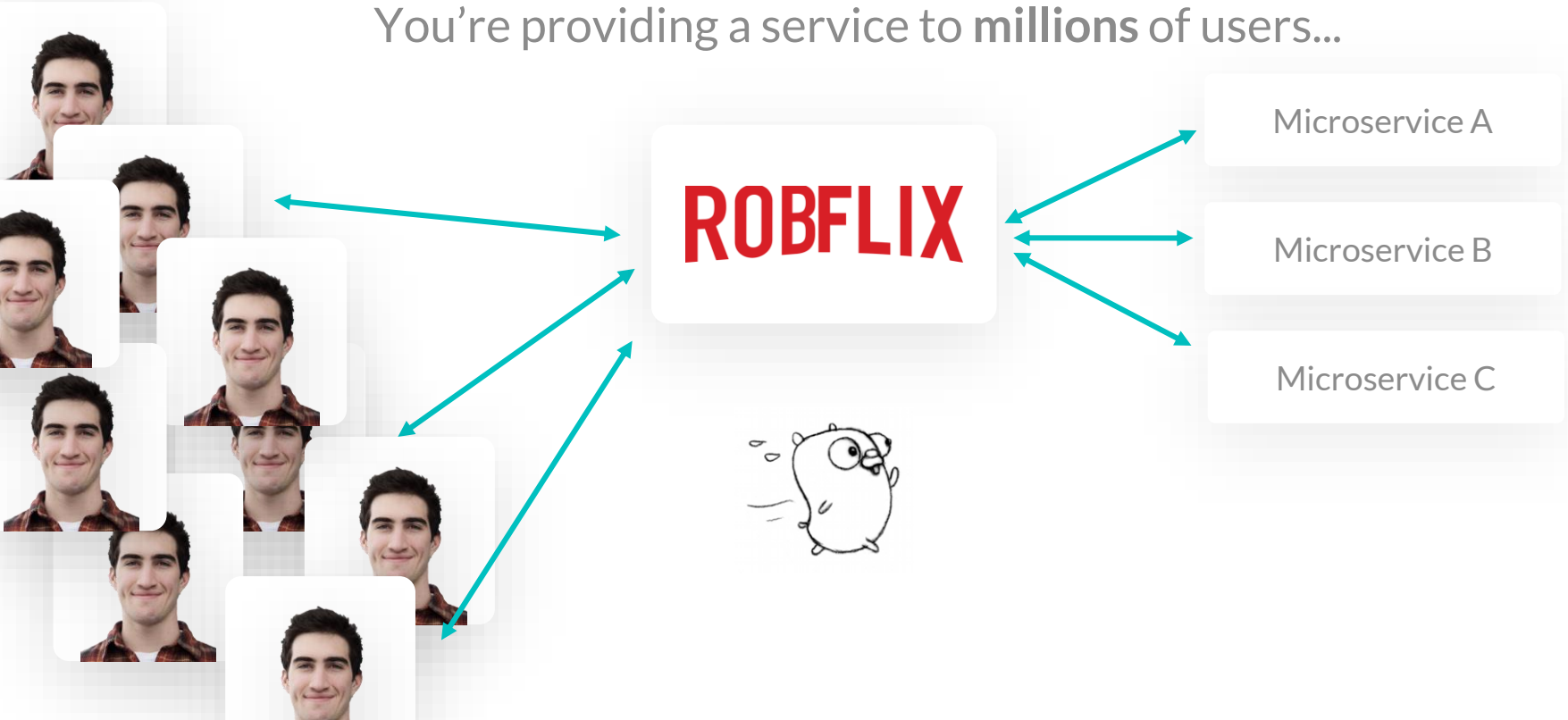
Why Containers

You're providing a service...



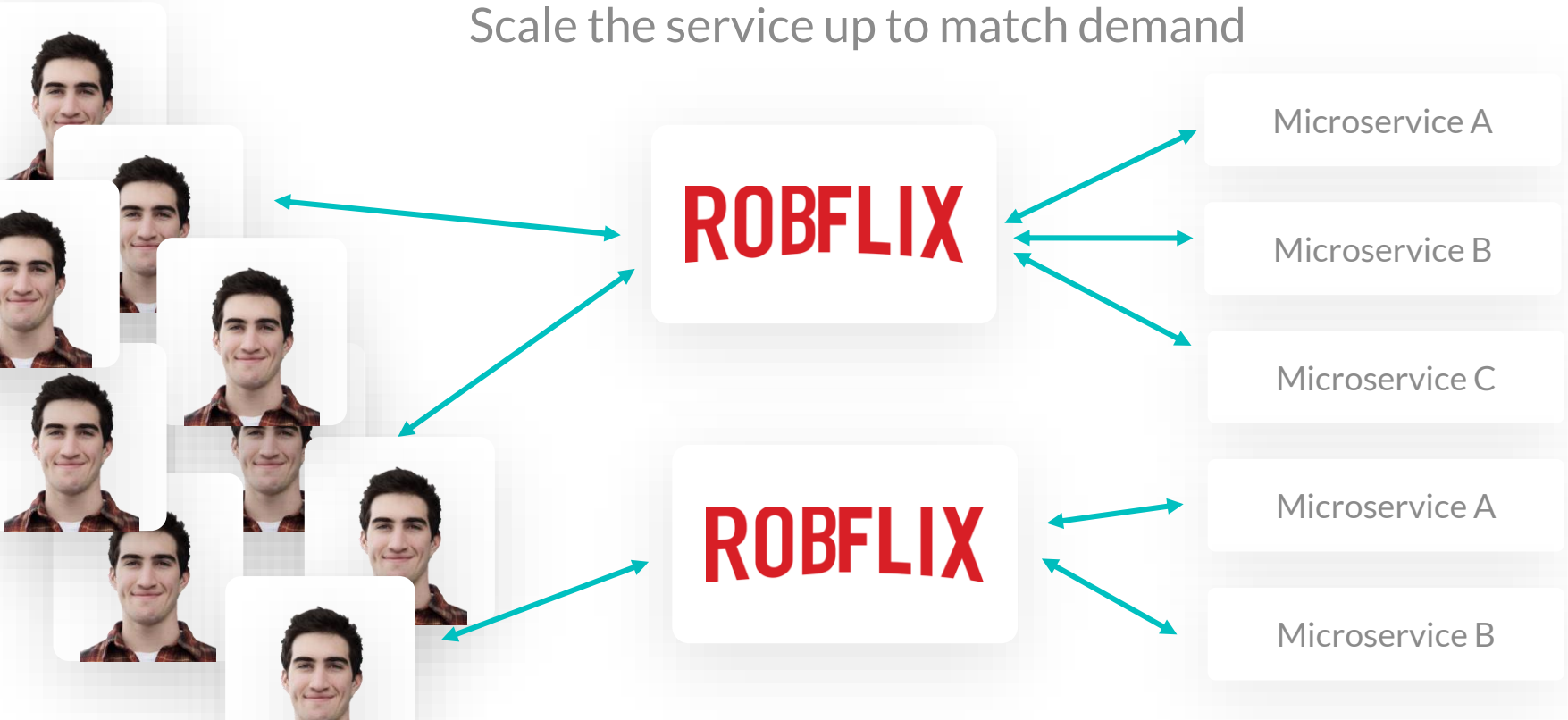
Why Containers

You're providing a service to **millions** of users...



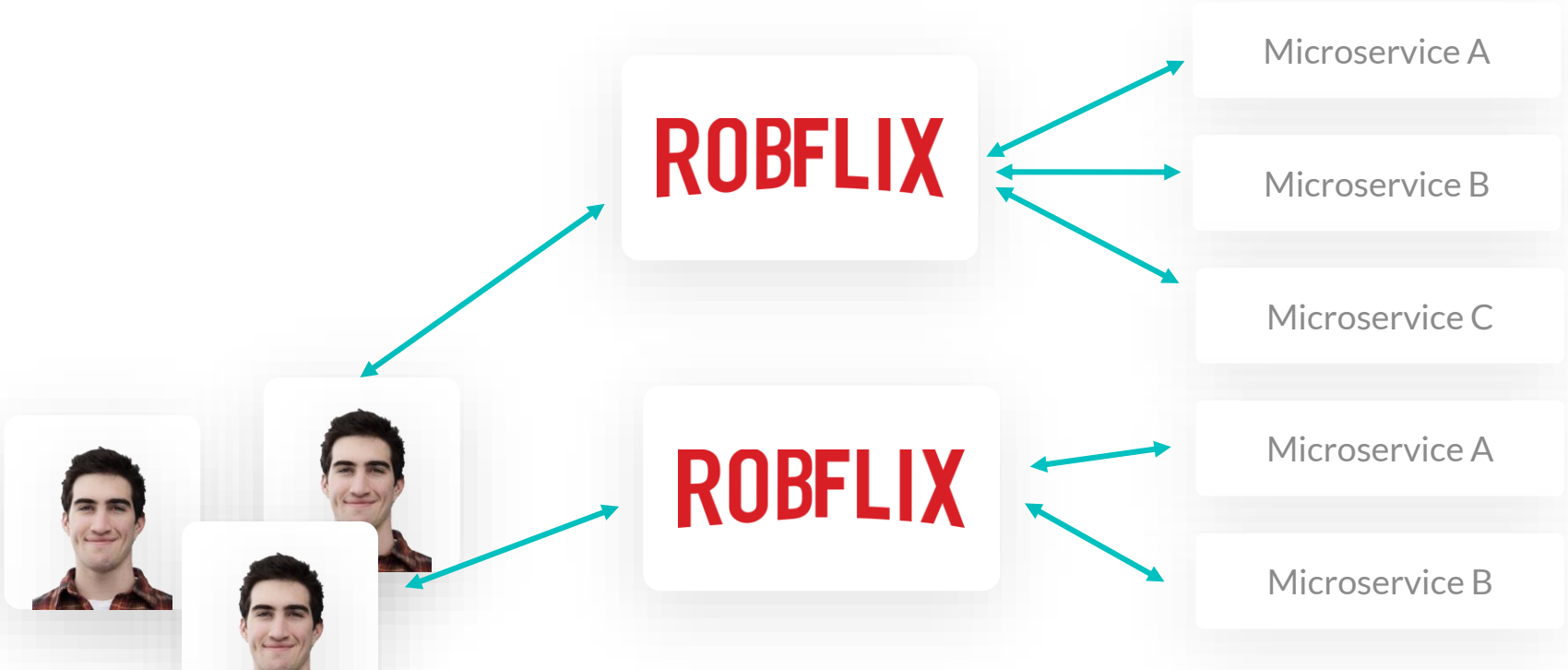
Why Containers

Scale the service up to match demand



Why Containers

Scale the service up to match demand



Why Containers

Scale the service down as needed

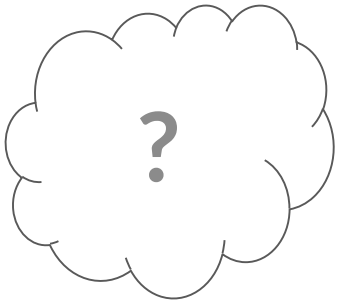
Scale back up tomorrow

Solution

What if you could encapsulate your apps within portable boxes?

What if these boxes could contain everything your program needs?

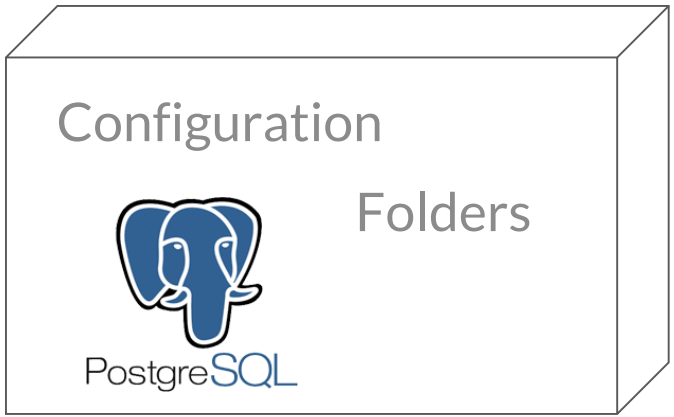
What if this box could run your program no matter where you put the box?



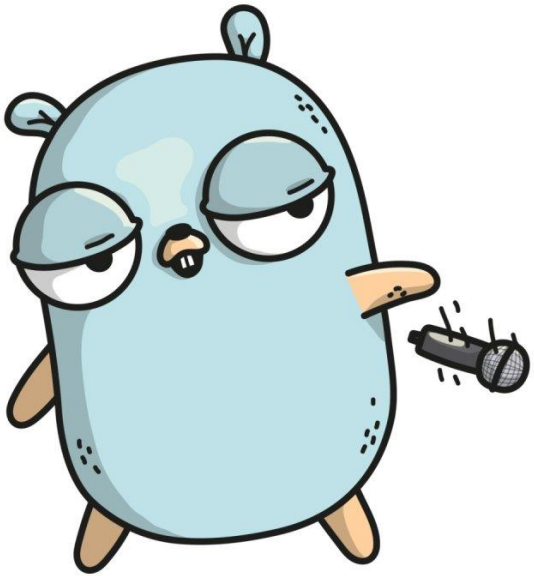
Solution

You could call these boxes…

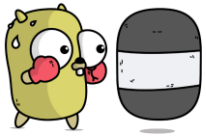
CONTAINERS



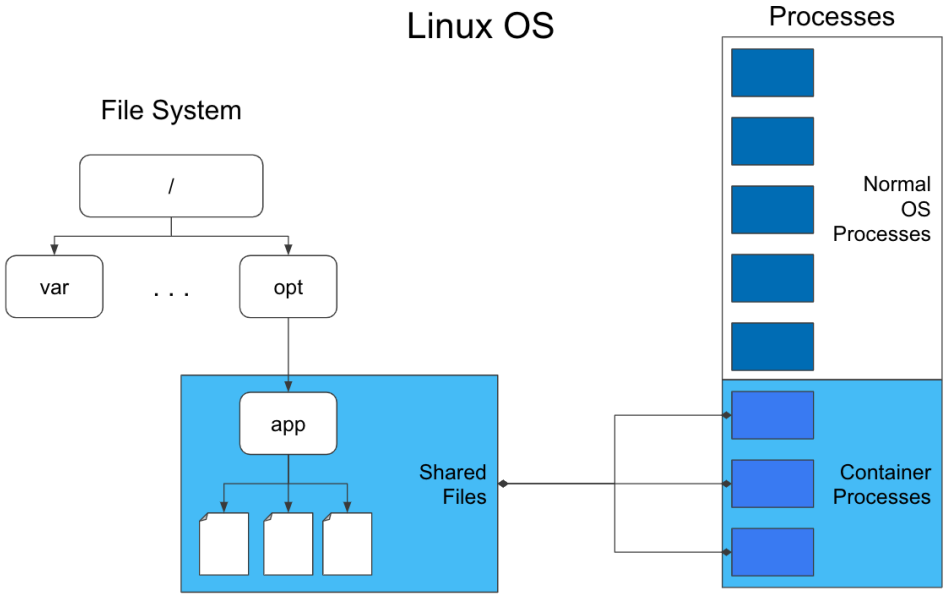
Q & A



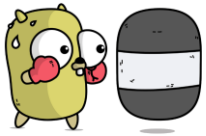
Containerization



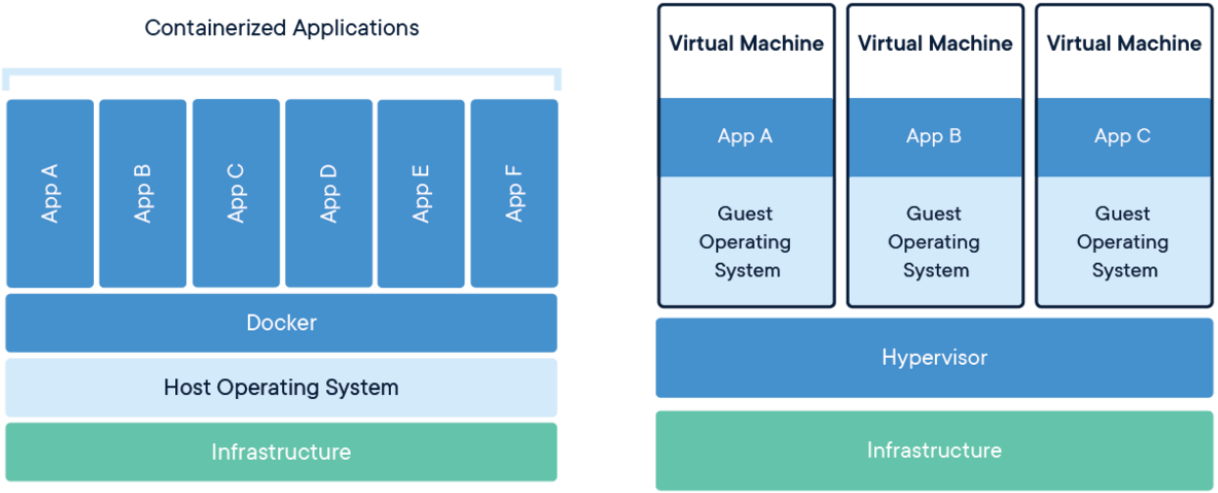
Technical Overview



Containerization



Technical Overview

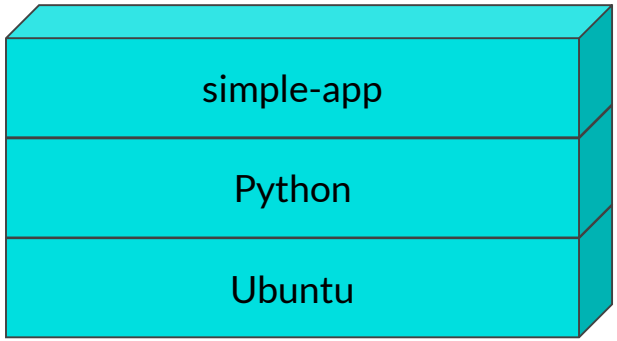


www.docker.com/resources/what-container

Building Blocks

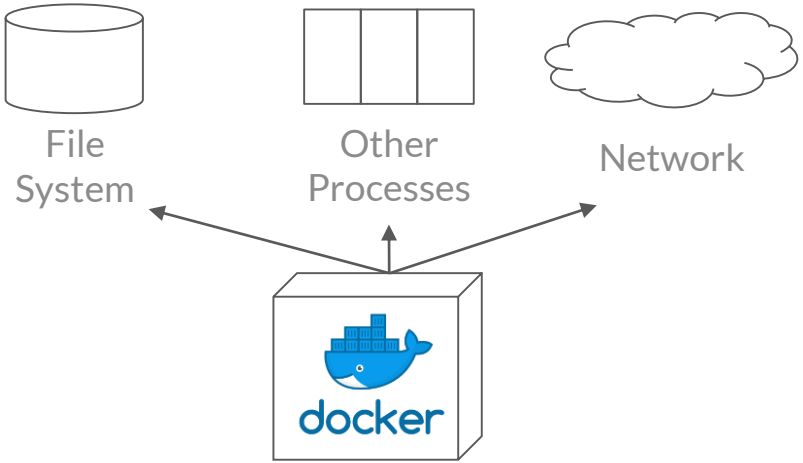
IMAGES

layered description of the file system

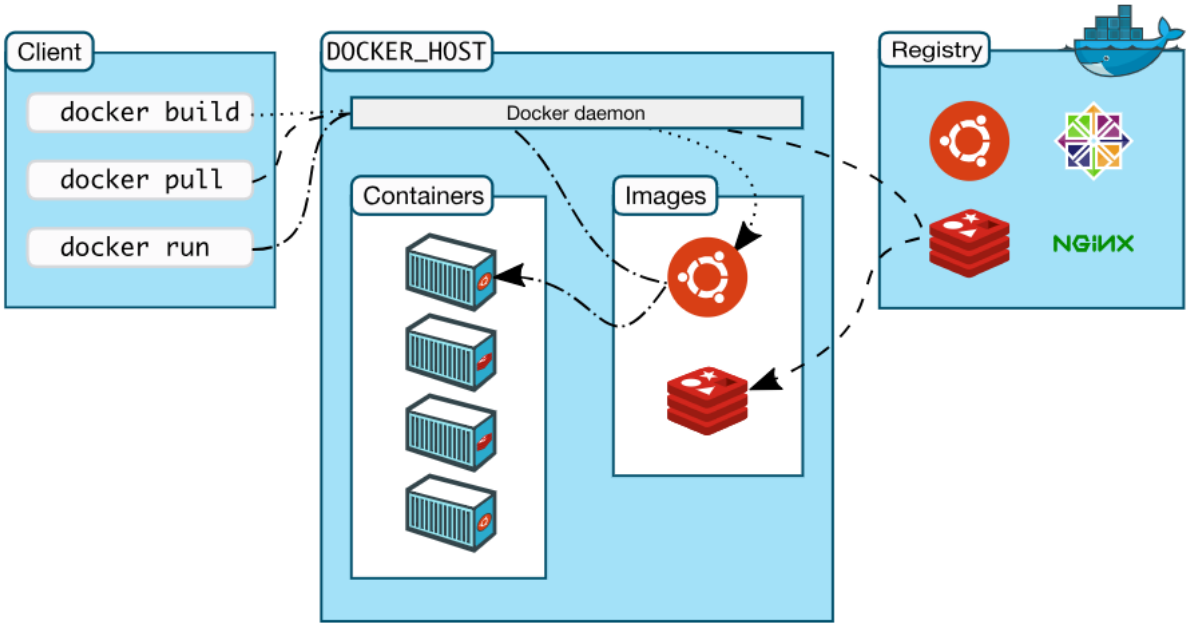


CONTAINERS

sets of processes with isolated views of the system



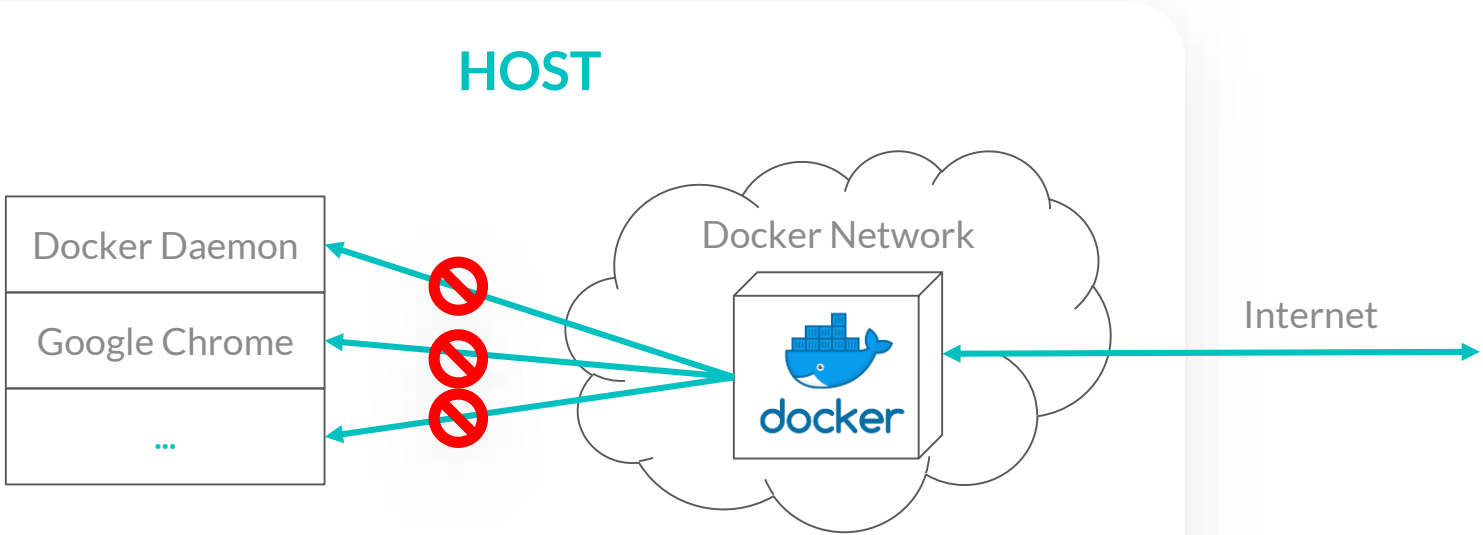
Docker Architecture



docs.docker.com/engine/docker-overview/#docker-architecture

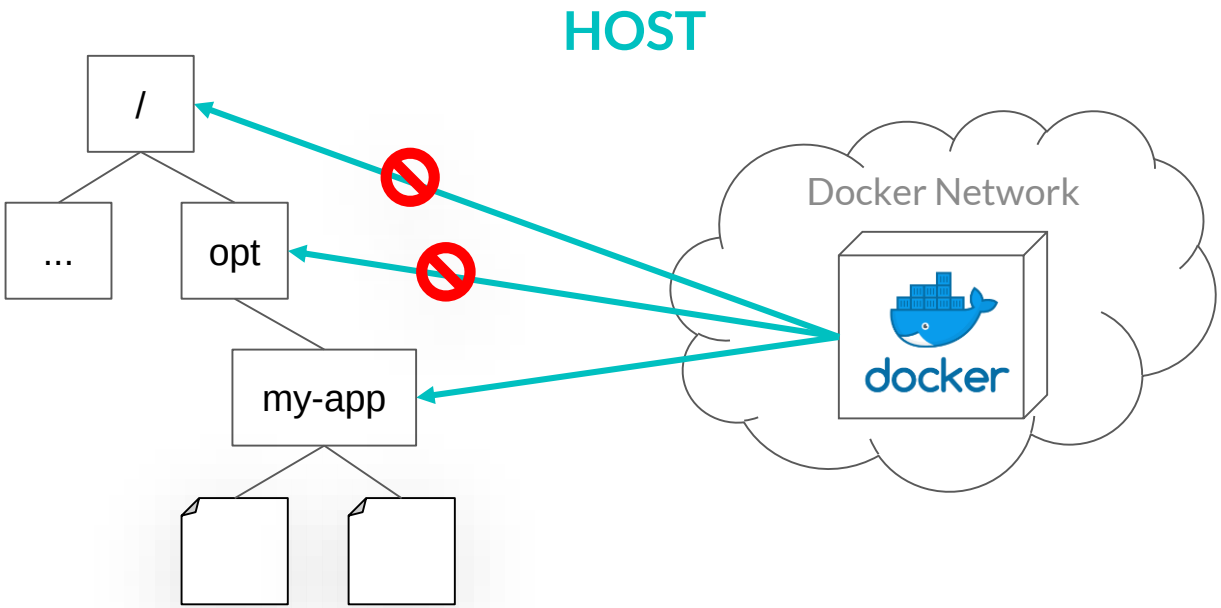
Docker Architecture

Networking



Docker Architecture

Volumes



Docker in the Wild

NETFLIX

130,000,000+
SUBSCRIBERS

140,000,000+
HOURS
WATCHED
PER DAY

www.statista.com/topics/842/netflix
media.netflix.com/en/press-releases/2017-on-netflix-a-year-in-bingeing

Docker in the Wild

NETFLIX

7
REGIONAL DEPLOYMENTS

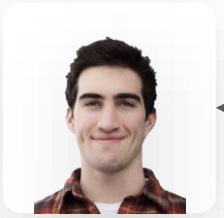
3,000,000+
CONTAINERS LAUNCHED
PER WEEK

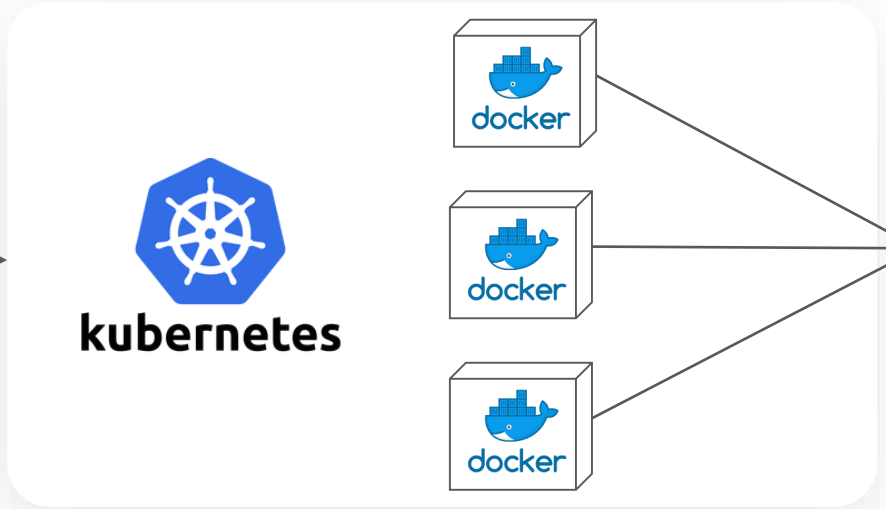
10,000+
EC2 VIRTUAL MACHINES

medium.com/netflix-techblog/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436

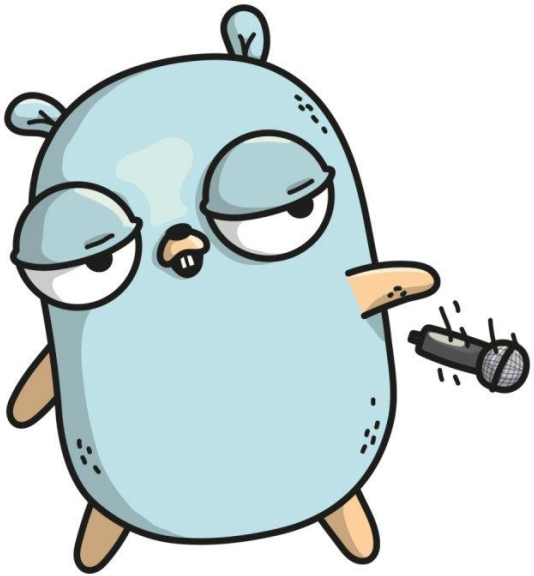
Docker in the Wild

demonware





Q & A

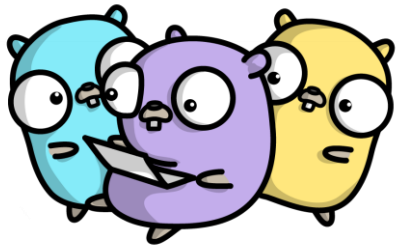


DIY!

Repository

github.com/ubclaunchpad/workshop-guide-to-docker

```
git clone https://github.com/ubclaunchpad/workshop-guide-to-docker.git
cd workshop-guide-to-docker
```



demo.Dockerfile

Dockerfiles are used to define and build Docker images

Each step is a command that makes up a “layer”

```
# Start with a basic image with Node.js
FROM node:8.12-alpine

# Install python 3
RUN apk add --update --no-cache python3

# Start up a simple Python HTTP server
CMD python3 -m http.server 8000
```

Getting Started

1. Make sure Docker is running
2. Build the example image:

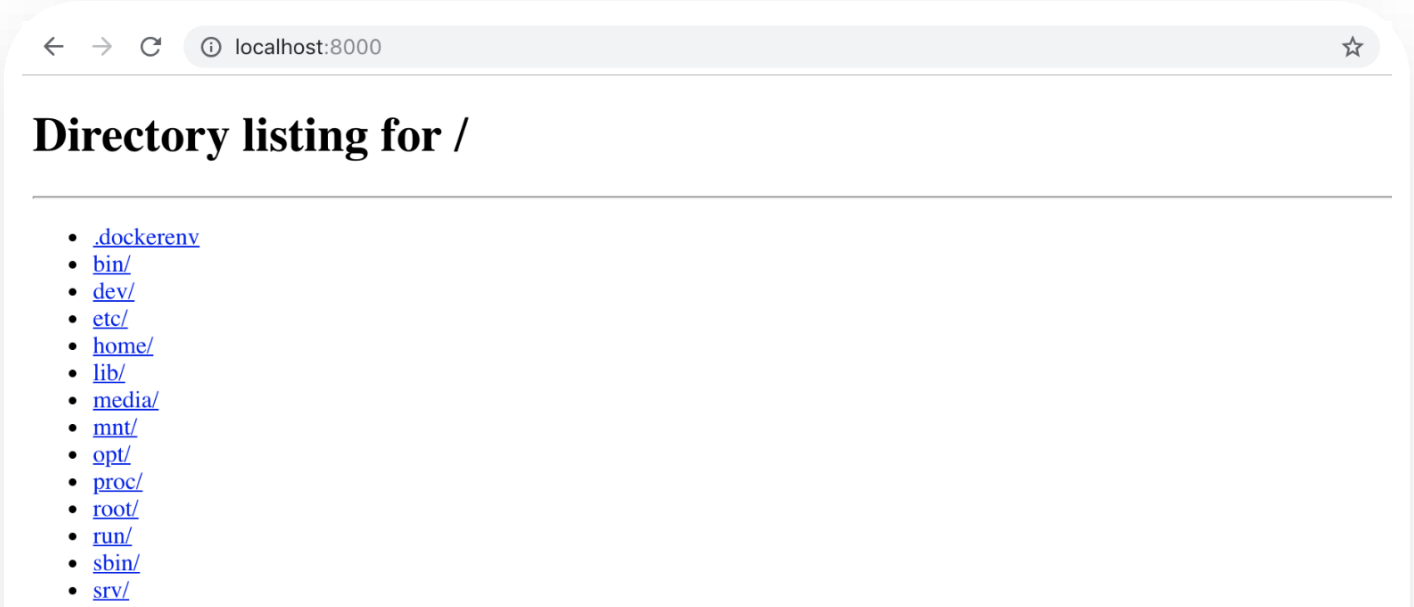
```
docker build -f demo.Dockerfile -t my-app .
```

3. Start a container from your ima

```
docker run -it -p 8000:8000 my-app
```

Getting Started

Visit the server that the container starts: http://localhost:8000/



What did we do?

```
docker build -f demo.Dockerfile -t my-app .
```

“Hey Docker!”

“build an image using demo.Dockerfile”

“call this image ‘my-app’”

What did we do?

```
docker run --rm -it -p 8000:8000 my-app
```

“Hey Docker!”

“create and run a container in interactive mode and remove it afterwards”

“I want to access this container’s port 8000 on my computer’s port 8000”

“create this container using the ‘my-app’ image”

What did we do?

```
# Start with a basic image with Node.js
FROM node:8.12-alpine

# Install python 3
RUN apk add --update --no-cache python3

# Start up a simple Python HTTP server
CMD python3 -m http.server 8000
```

this command is run when the container starts

DIY!

Activity: Docker for Development

start with an image that has Node.js

add an “entrypoint” command to install dependencies and run the website inside a Docker container

leverage volume mounting to allow code changes to be reflected instantly in a container

Useful Tidbits

creating a directory and setting path in image:

```
RUN mkdir -p /src/app
WORKDIR /src/app
```

building an image:

```
docker build -f dev.Dockerfile -t dev-image .
```

exposing ports and mounting directories:

```
docker run --rm -p $HOST_PORT:$CONTAINER_PORT -v ${PWD}:$CONTAINER_DIR dev-image
```

to install dependencies and run the web app on port 3000:

```
yarn install
yarn start
```

DIY!

Activity: Docker for Development

build your new image and start a new container!

visit localhost:3000

try editing a file like `src/app.js`

your changes should reflect after a moment on the website!

dev.Dockerfile

1 Start with Node.js image

2 Make a directory to put your code in, and set that as your “working directory”

3 Install and start web app when running the container

```
# Start with a basic image with Node.js
FROM node:8.12-alpine

# Set where to run commands from (mount code here!)
RUN mkdir -p /src/app
WORKDIR /src/app

# install dependencies and run app when container starts
CMD yarn install && yarn start
```

DIY!

We’ve just set up a completely containerized development environment

No need to install Node.js, just build the image and container and GO

Easy for every developer to reproduce

DIY!

Activity: Docker for Distribution

However, dev.Dockerfile is **not good** for distribution

- Installs dependencies every time
- Webpack-Dev-Server has a lot of overhead
- No need for live-updating code
- No need for development dependencies

DIY!

Activity: Docker for Distribution

build a lightweight image optimized for distribution

run the container to serve the website

upload to Docker Hub so anyone can download and deploy your image (optional!)

hint: remember the Python server command in demo.Dockerfile!

Useful Tidbits

copy local directory into a directory in the container:

```
ADD . $CONTAINER_DIRECTORY
```

to install dependencies and build the web app into `./build`:

```
yarn install --production
yarn build
```

Docker Hub (optional): hub.docker.com

build and (optional) publish your image:

```
docker build -t $MY_USERNAME/docker-workshop:latest
docker push $MY_USERNAME/docker-workshop:latest
```

prod.Dockerfile

1. Start with Node.js image
2. Install Python
3. Copy your code into the container
4. Install deps and build the web app
5. Serve app from build directory

```
# Start with a basic image with Node.js
FROM node:8.12-alpine

# Install python 3
RUN apk add --update --no-cache python3

# Set where to run commands from
RUN mkdir -p /src/app
WORKDIR /src/app

# Copy code into directory
ADD . .

# Build web app into ./build
RUN yarn install --production
RUN yarn build

# Start up a simple Python HTTP server
WORKDIR /src/app/build
CMD python3 -m http.server 8000
```

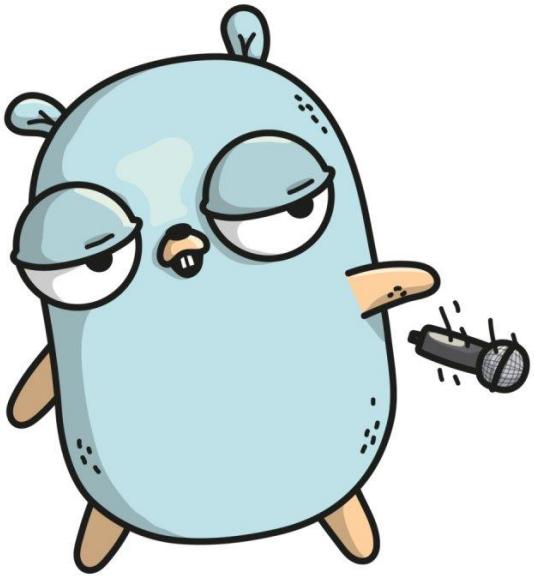
Solutions

Solutions are available in the solutions branch:

`github.com/ubclaunchpad/workshop-guide-to-docker/tree/solutions`

```
git checkout solutions
```

Q & A



Advanced Usage

Multi-Stage Builds

github.com/ubclaunchpad/inertia

```
### Part 1 - Building the Web Client
FROM node:carbon AS web-build-env
ENV BUILD_HOME=/go/src/github.com/ubclaunchpad/inertia/daemon/web
# Mount source code.
ADD ./daemon/web ${BUILD_HOME}
WORKDIR ${BUILD_HOME}
# Build and minify client.
RUN npm install --production
RUN npm run build

### Part 2 - Building the Inertia daemon
FROM golang:alpine AS daemon-build-env
ARG INERTIA_VERSION
ENV BUILD_HOME=/go/src/github.com/ubclaunchpad/inertia \
    INERTIA_VERSION=${INERTIA_VERSION}
# Mount source code.
ADD . ${BUILD_HOME}
WORKDIR ${BUILD_HOME}
# Install dependencies if not already available.
RUN if [ ! -d "vendor" ]; then \
    apk add --update --no-cache git; \
    go get -u github.com/golang/dep/cmd/dep; \
    dep ensure -v; \
    fi
```

Advanced Usage

Multi-Stage Builds

github.com/ubclaunchpad/inertia

```
# Build daemon binary.
RUN go build -o /bin/inertiad \
    -ldflags "-w -s -X main.Version=$INERTIA_VERSION" \
    ./daemon/inertiad

### Part 3 - Copy builds into combined image
FROM alpine
LABEL maintainer "UBC Launch Pad team@ubclaunchpad.com"
RUN mkdir -p /daemon
WORKDIR /daemon
COPY --from=daemon-build-env /bin/inertiad /usr/local/bin
COPY --from=web-build-env \
    /go/src/github.com/ubclaunchpad/inertia/daemon/web/public/ \
    /daemon/inertia-web

# Directories
ENV INERTIA_PROJECT_DIR=/app/host/inertia/project/ \
    INERTIA_SSL_DIR=/app/host/inertia/config/ssl/ \
    INERTIA_DATA_DIR=/app/host/inertia/data/ \
    INERTIA_GH_KEY_PATH=/app/host/.ssh/id_rsa_inertia_deploy

# Build tool versions
ENV INERTIA_DOCKERCOMPOSE=docker/compose:1.22.0

# Serve the daemon by default.
ENTRYPOINT ["inertiad", "run"]
```

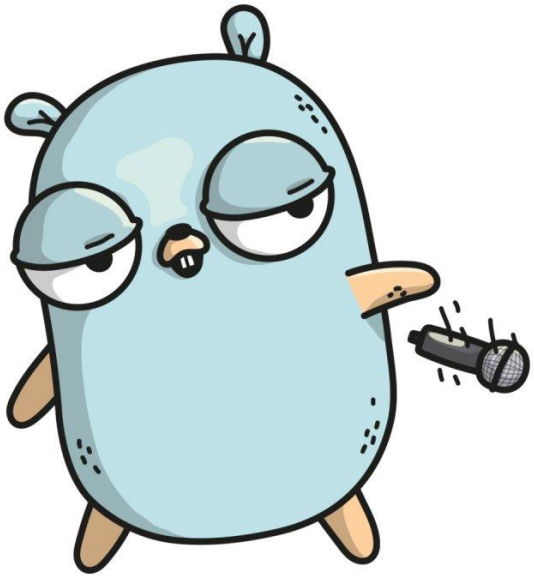
Advanced Usage

docker-compose

github.com/ubclaunchpad/rocket

```yaml
services:
  postgres:
    build:
      context: .
      dockerfile: Dockerfile.db
    ports:
      - "5432:5432"
    volumes:
      - ./pgdata:/var/lib/postgresql/data
    env_file:
      - .db.env
  rocket:
    build:
      context: .
      dockerfile: Dockerfile.app
    command: rocket
    volumes:
      - "/etc/ssl/certs:/etc/ssl/certs"
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - postgres
    env_file:
      - .app.env
    restart: on-failure
```

Q & A



Thank you!

UBCLAUNCHPAD.COM

@UBCLAUNCHPAD