## DOCKER IN 15 MINUTES

# LIFE ON A CONTAINER SHIP

## CONFINEMENT

- Two popular methods exist for isolating applications from each other

### VIRTUALISATION

- Used AWS or VMWare?
- Use software (hypervisor) to emulate hardware to run more software (guest OS)
- Hugely universal and lets you run any OS
- But you pay in other ways…

### CONTAINERS

- Used UQCloud?
- Created isolated slices of a shared OS for each application
- Slices can only run the host OS
- Typically negligible overhead

## WHAT IS DOCKER?

- Docker is a collection of technologies, services and software to make environments portable.

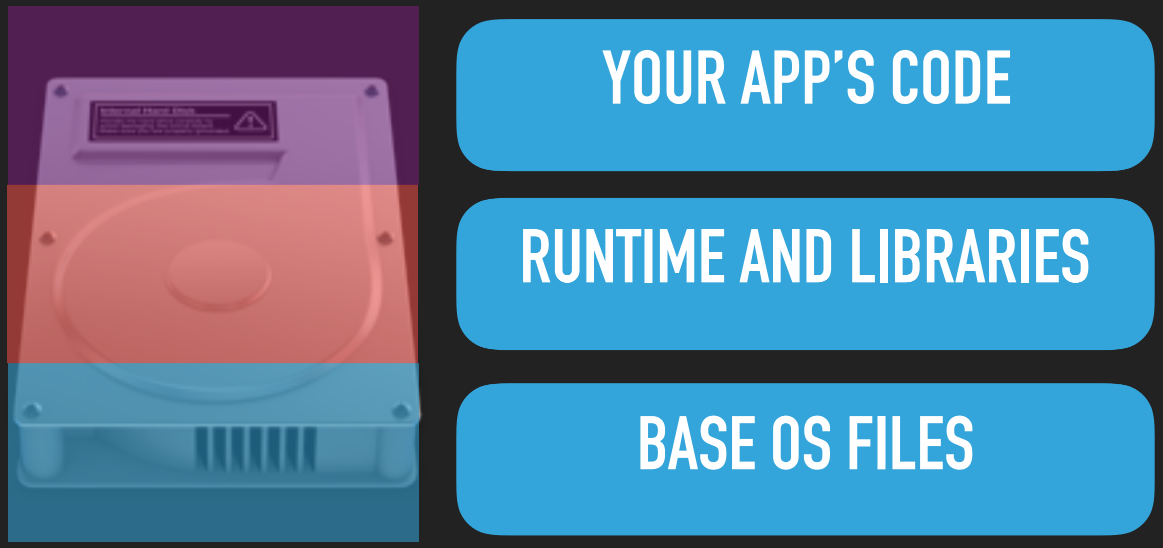
Image

Container

Host

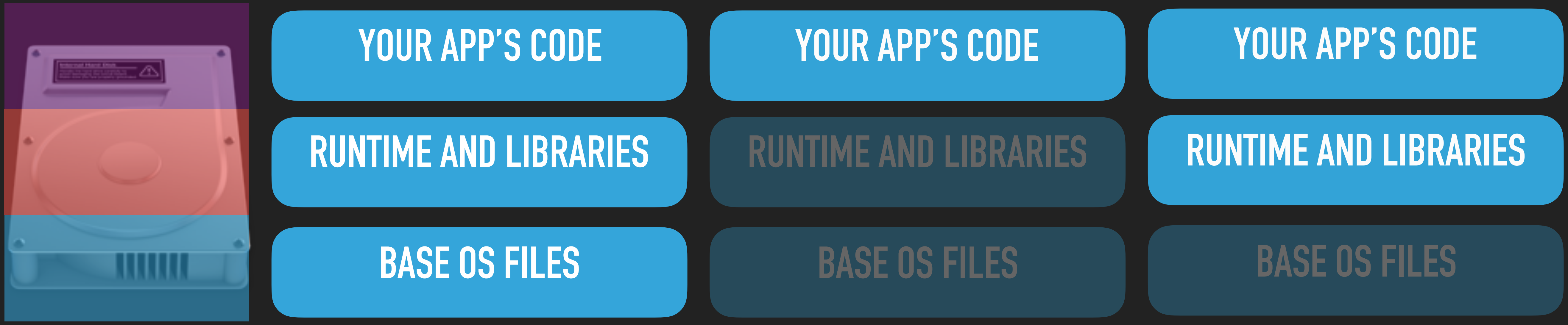
## IMAGES

- An extensible, read-only snapshot of a file system used to create multiple identical containers
- A combination of cloning, but also separation of requirements for later reuse
- Build process automated via a Dockerfile



## IMAGES

- An extensible, read-only snapshot of a file system used to create multiple identical containers
- A combination of cloning, but also separation of requirements for later reuse
- Build process automated via a Dockerfile



## DOCKER

Alpine Linux
Base OS

```
FROM scratch
ADD rootfs.tar.gz /
```

Node.js
Runtime and
Libs

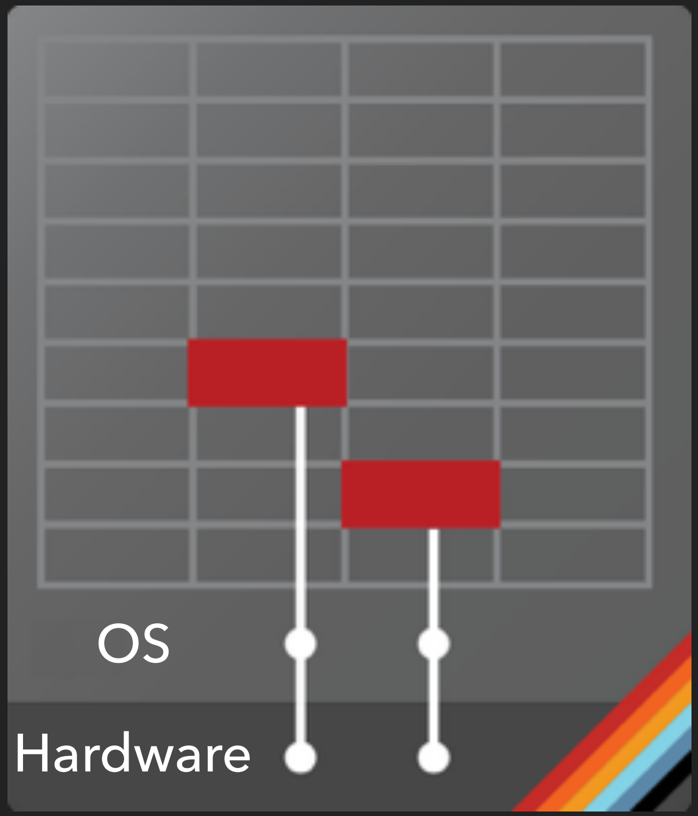
```
FROM alpine:3.3

RUN apk add --no-cache curl make gcc g++ binutils-gold python linux-headers paxctl libgcc libstdc++ gnupg && \
  gpg --verify SHASUMS256.txt.asc && \
  grep node-${VERSION}.tar.gz SHASUMS256.txt.asc | sha256sum -c - && \
  tar -zxf node-${VERSION}.tar.gz && \
  cd /node-${VERSION} && \
  ./configure --prefix=/usr ${CONFIG_FLAGS} && \
  make -j$(grep -c ^processor /proc/cpuinfo 2>/dev/null || 1) && \
  make install && \
  paxctl -cm /usr/bin/node && \
  cd / && \
  if [ -x /usr/bin/npm ]; then \
    npm install -g npm@${NPM_VERSION} && \
    find /usr/lib/node_modules/npm -name test -o -name .bin -type d | xargs rm -rf; \
  fi && \
  apk del curl make gcc g++ binutils-gold python linux-headers paxctl gnupg ${DEL_PKGS} && \
  rm -rf /etc/ssl /node-${VERSION}.tar.gz /SHASUMS256.txt.asc /node-${VERSION} ${RM_DIRS} \
    /usr/share/man /tmp/* /var/cache/apk/* /root/.npm /root/.node-gyp /root/.gnupg \
    /usr/lib/node_modules/npm/man /usr/lib/node_modules/npm/doc /usr/lib/node_modules/npm/html
```

My Code

```
FROM mhart/alpine-node:latest

MAINTAINER  Dave Finster <davefinster@me.com>

WORKDIR /usr/src/app

COPY ./ ./

RUN apk add --no-cache --virtual .npm-deps git python make gcc linux-headers alpine-sdk\
      && npm install --production \
      && apk del .npm-deps

EXPOSE 8080

CMD [ "node", "app.js" ]
```

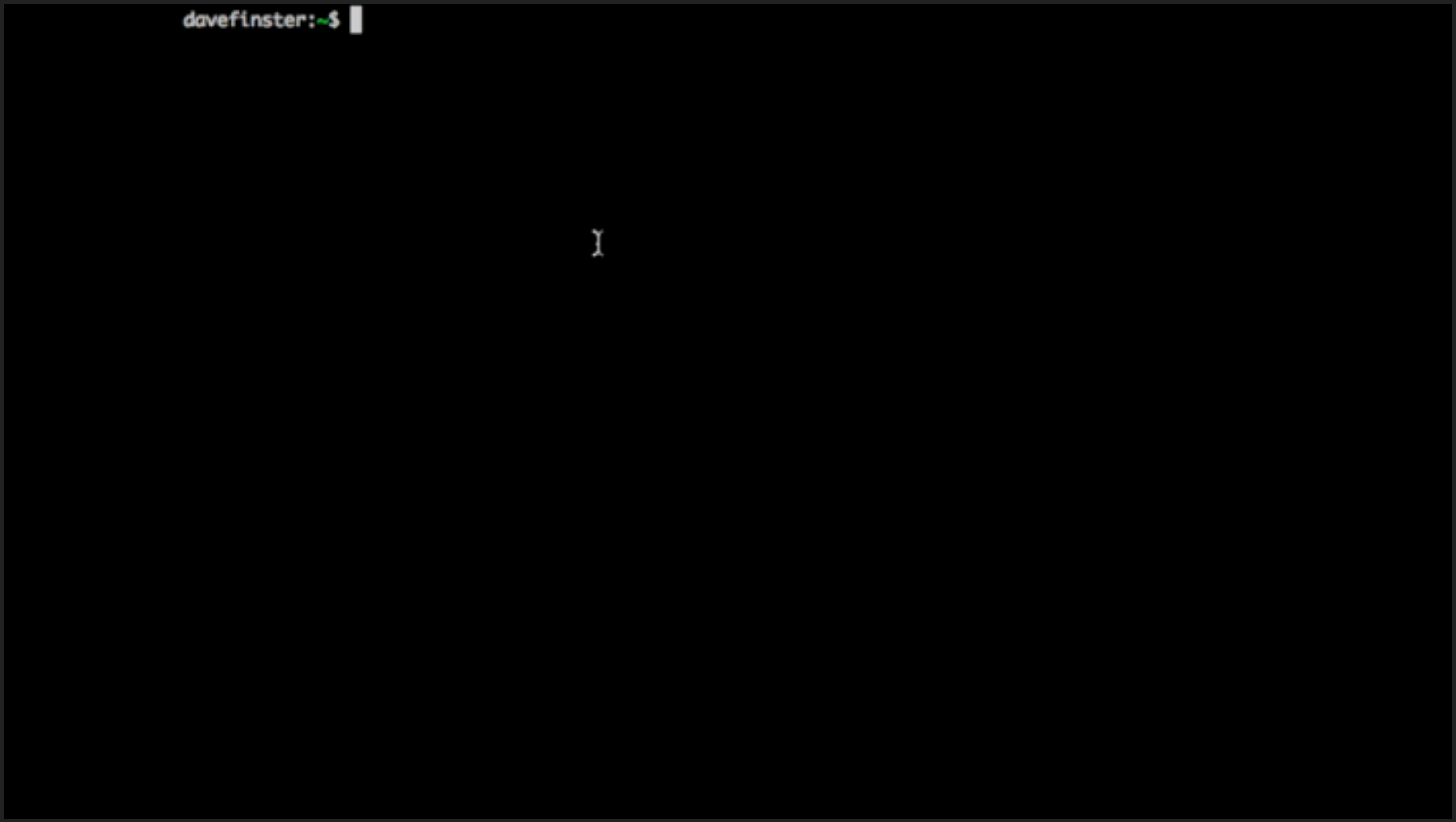
## CONTAINERS

- Lightweight isolation mechanism for applications
- Each container has its own memory and compute allocation, along with an isolated filesystem enforced by the OS
- Provisioning is wicked fast because there is very little work to do



## HOW IS THIS USEFUL - DEVELOPMENT

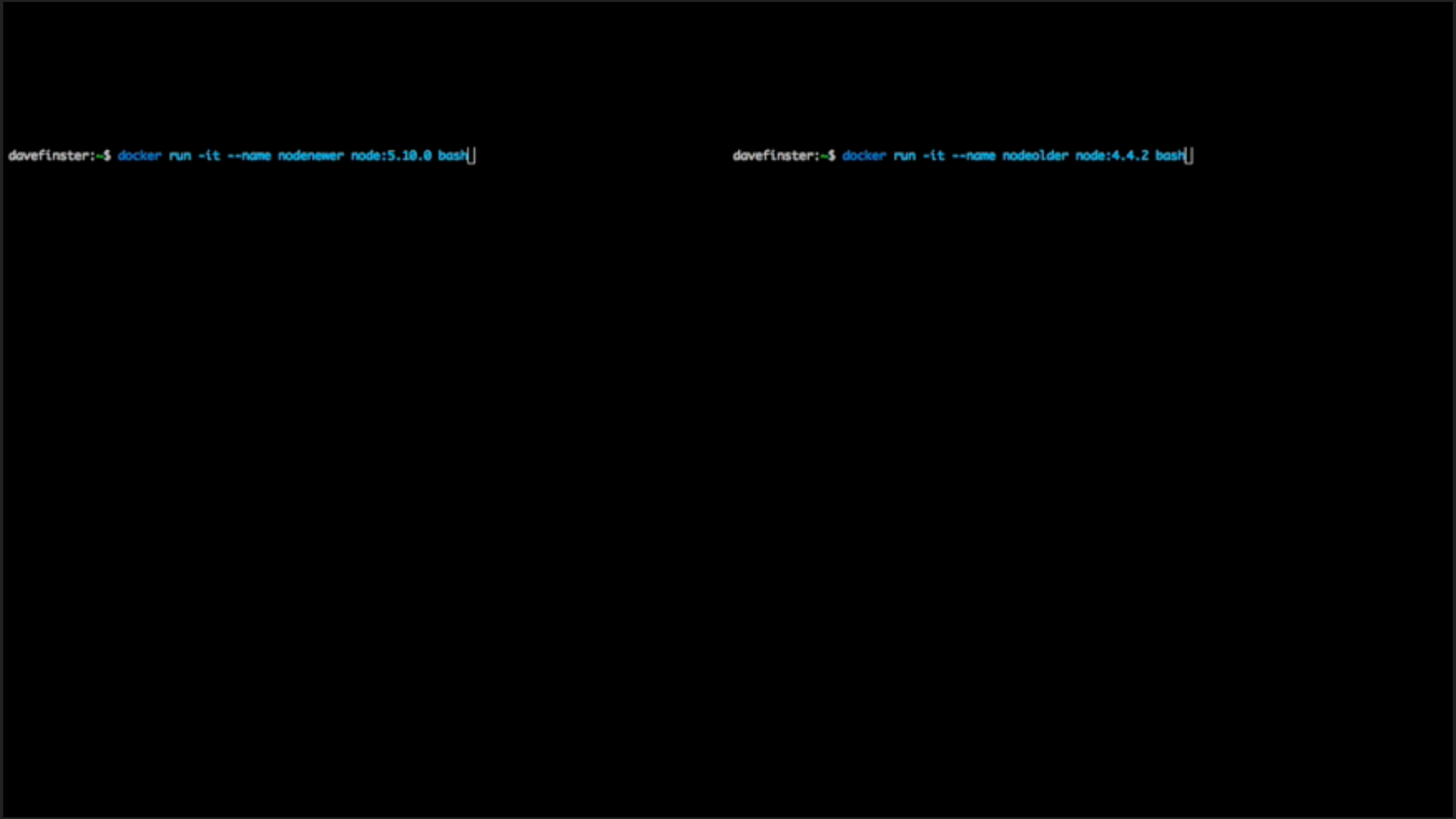
- Ever wanted to try a new language but not wreck your system?



```
davefinster:~$ 
```

## HOW IS THIS USEFUL – DEVELOPMENT

- Ever had to juggle runtime versions?



```
davefinster:~$ docker run -it --name nodenewer node:5.10.0 bash
```

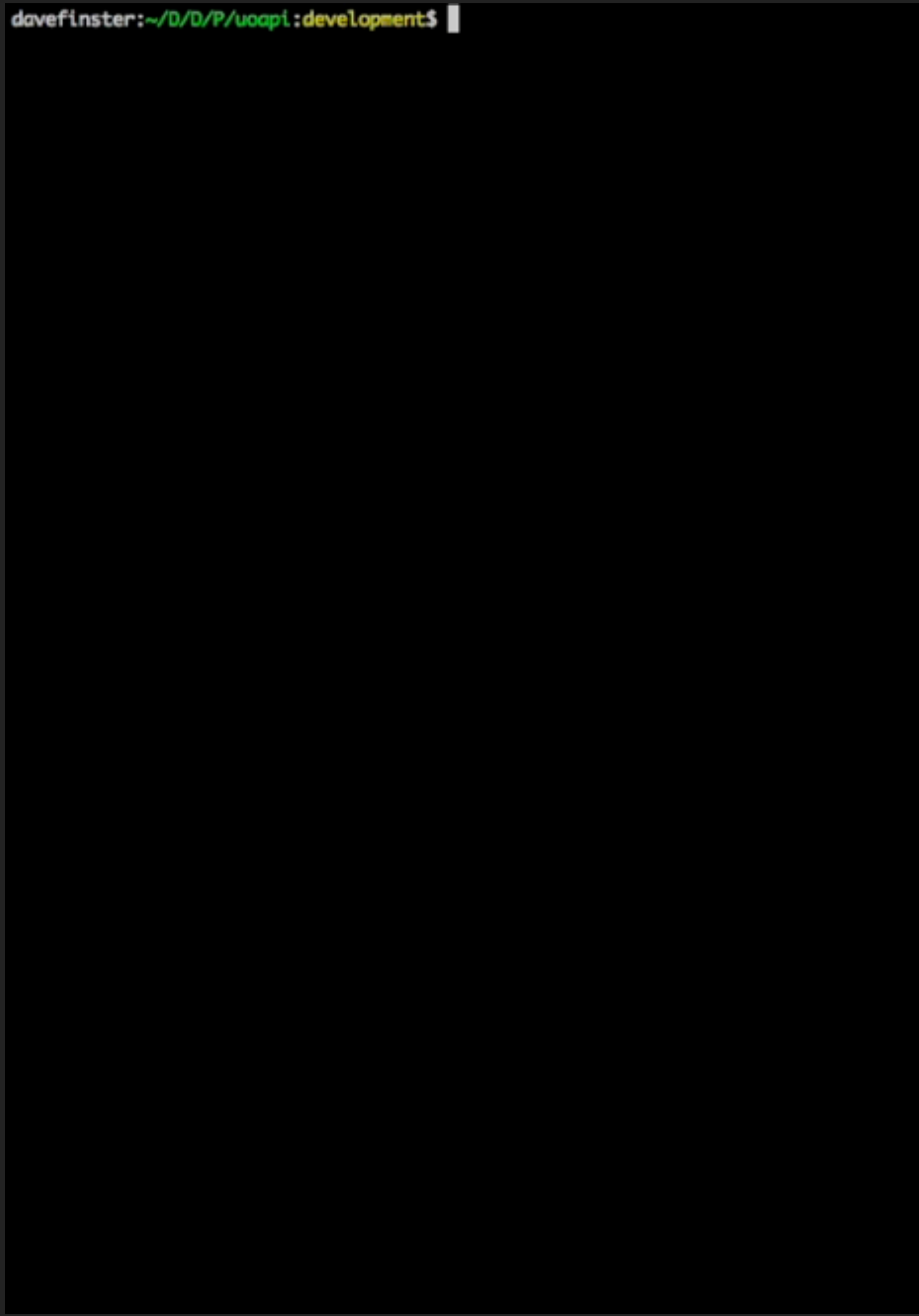
```
davefinster:~$ docker run -it --name nodeolder node:4.4.2 bash
```

## HOW IS THIS USEFUL - DEVELOPMENT

- Ever developed alongside that one Windows user?
    - docker run -it <your image>
    - Or www.apple.com
    - Or http://releases.ubuntu.com/15.10/ubuntu-15.10-desktop-amd64.iso

## HOW IS THIS USEFUL - PRODUCTION

- Ever get sick of provisioning virtual machines by hand



```
davefinster:~/D/D/P/uoapi:development$
```

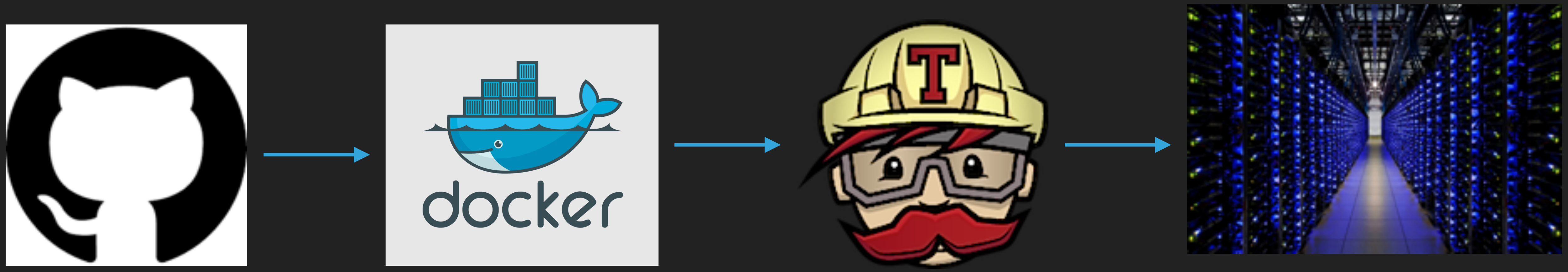
## HOW IS THIS USEFUL - PRODUCTION

- Sure your app's dependencies got updated in production?

```
davefinster:~/D/D/P/uoapi:development$ 
```

## HOW IS THIS USEFUL – AUTOMATION

- Docker encourages the complete automation of image construction and container deployment
- Source code repositories can be connected to Docker Hub such that pushes result in automated image builds
- Humans are terrible at remembering boring processes. Automated is better
- Minimal effort from developer’s desk to production (with appropriate checks) can only be a good thing



## GETTING STARTED

## WHAT YOU'LL NEED

- Just head to docker.com and download the Toolbox. Install as per their instructions (platform specific). This will allow you to download and run images/containers.
- You'll need a Docker Hub account from hub.docker.com to push images to the registry. Not needed to pull/browse
- Thats it!

## PRODUCTION IN REALITY

- Linux container technologies are relatively new and were not built with security in mind
- As such,  it isn't sufficiently trusted (yet) to allow containers from multiple tenants to co-habit the same Docker Engine
- To solve this, everyone running a multi-tenant cloud is deploying Docker Engine inside hardware virtualisation (except Joyent - that I know of). There are cost implications here as well
- There are also some communication backchannels between the container and Docker Engine that need locking down

## ALSO NOTE

- Not a magic fix to scalability issues. All Docker fundamentally does is help make environments portable and deployments more consistent, reliable and easier.
- If your code is terrible, it'll just ship and start sucking CPU cycles faster
- Suddenly working with a dynamic infrastructure can be overwhelming with nasty implications if done badly
    - See "distributed systems"

## THANKS

- Questions?
- I’m @df on UQCS Slack