

Docker in the real world

Ed Courtenay

Agenda

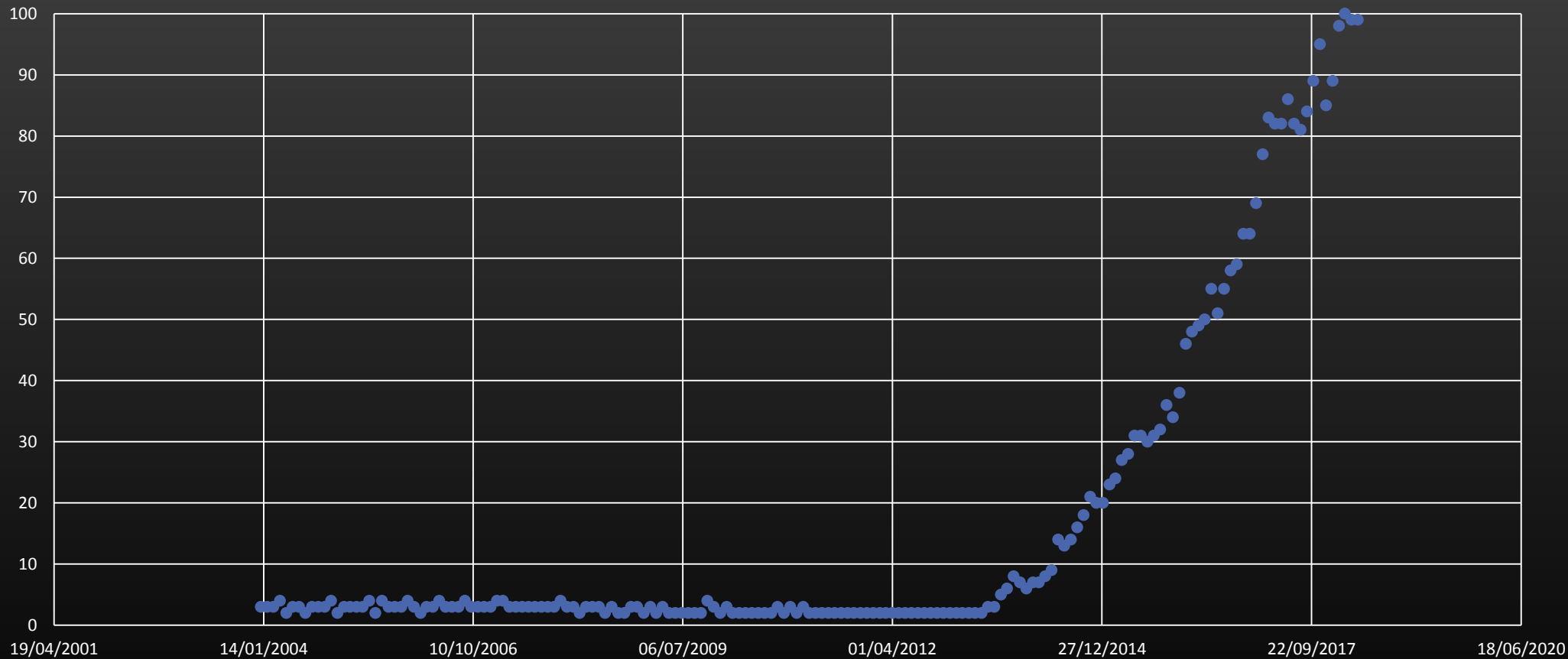
- Who am I?
- A (very) brief history of containers
- Getting started
- Writing your own Dockerfile
- Multi stage builds

- Loads of demos

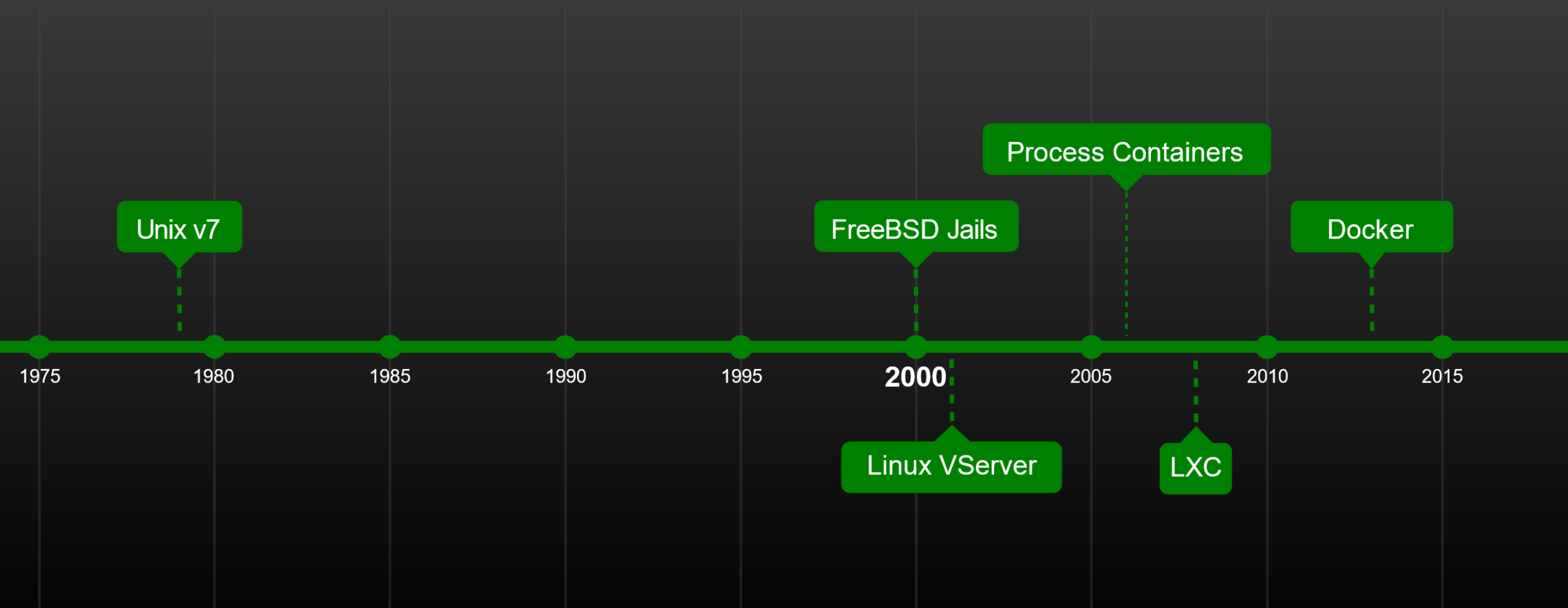
What are Docker containers?

- Containers are runtime environments. You usually run one main process in one Docker container.
- Docker containers are started by running a **Docker image**. A Docker image is a pre-built environment for a certain technology or service.

Google Trend for 'docker'

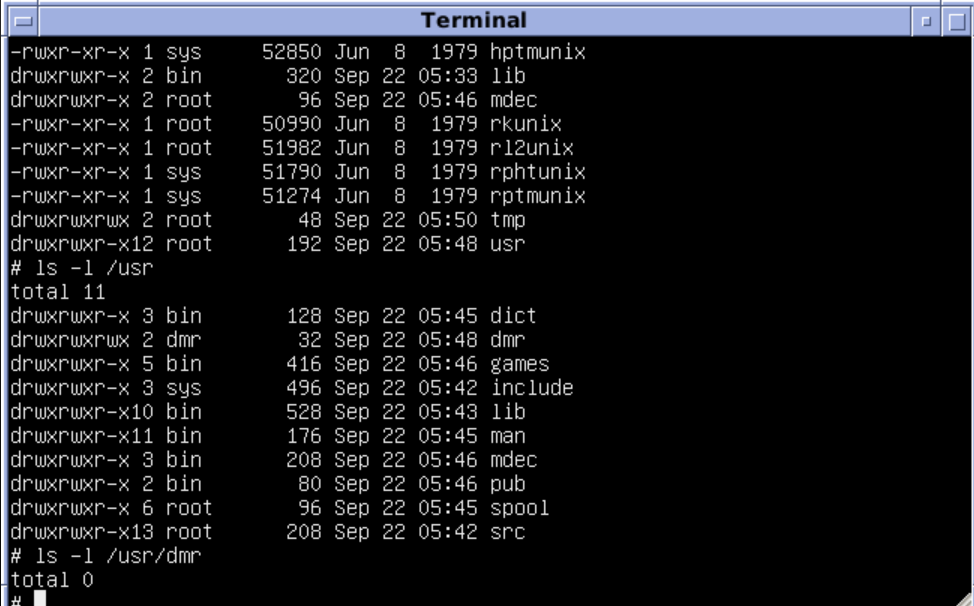


How did we get here?



1979 – Version 7 Unix

- chroot system call was introduced
 - changes the root directory of a process and its children to a new location in the filesystem
- BSD gained chroot in 1982



```
Terminal
-rwxr-xr-x 1 sys      52850 Jun  8  1979 hptmunix
drwxrwxr-x 2 bin        320 Sep 22  05:33 lib
drwxrwxr-x 2 root      96 Sep 22  05:46 mdec
-rwxr-xr-x 1 root    50990 Jun  8  1979 rkunix
-rwxr-xr-x 1 root    51982 Jun  8  1979 r12unix
-rwxr-xr-x 1 sys     51790 Jun  8  1979 rphtunix
-rwxr-xr-x 1 sys     51274 Jun  8  1979 rptmunix
drwxrwxrwx 2 root      48 Sep 22  05:50 tmp
drwxrwxr-x12 root    192 Sep 22  05:48 usr
# ls -l /usr
total 11
drwxrwxr-x 3 bin        128 Sep 22  05:45 dict
drwxrwxrwx 2 dmr        32 Sep 22  05:48 dmr
drwxrwxr-x 5 bin       416 Sep 22  05:46 games
drwxrwxr-x 3 sys       496 Sep 22  05:42 include
drwxrwxr-x10 bin      528 Sep 22  05:43 lib
drwxrwxr-x11 bin      176 Sep 22  05:45 man
drwxrwxr-x 3 bin       208 Sep 22  05:46 mdec
drwxrwxr-x 2 bin        80 Sep 22  05:46 pub
drwxrwxr-x 6 root      96 Sep 22  05:45 spool
drwxrwxr-x13 root    208 Sep 22  05:42 src
# ls -l /usr/dmr
total 0
#
```

2000 – FreeBSD Jails

- Administrators can partition a system into several ‘jails’
 - independent, smaller systems
 - IP address for each system and configuration



2001 – Linux vServer

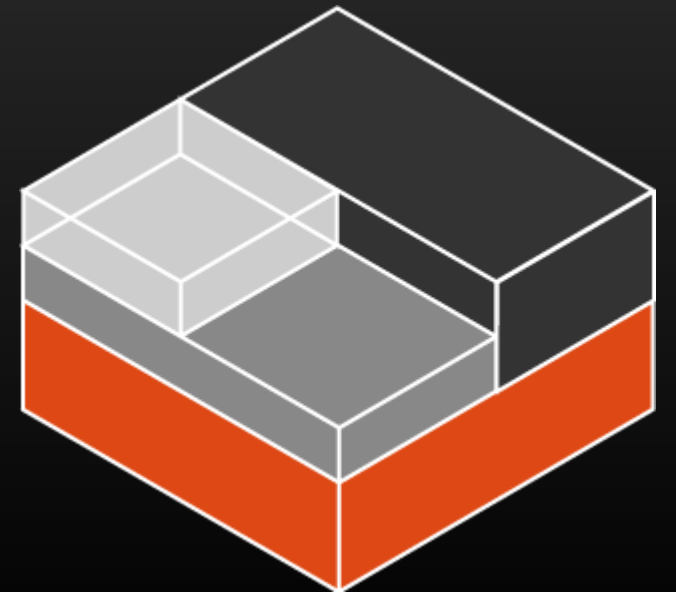
- Similar to FreeBSD jails
- Implemented by kernel patch

2006 – Process Containers

- Launched by Google in 2006
 - Renamed to 'Control Groups' in 2007
- Limits, accounting and isolation of process collections
 - CPU
 - Memory
 - Network & Disk access
- Linux kernel 2.6.24

2008 - LXC

- Linux Containers
- Built using cgroups
- Runs on single Linux kernel
 - Requires no patches



2013 - Docker

- Originally built using LXC
- Since version 0.9.0 uses it's own libcontainer library





Hyper-V Manager
JEUKBRILAP20190

Virtual Machines

Name	State	CPU Usage	Assigned Memory	Uptime	Status
MobyLinuxVM	Running	0%	2048 MB	2:01:02:27	

Checkpoints

The selected virtual machine has no checkpoints.

MobyLinuxVM

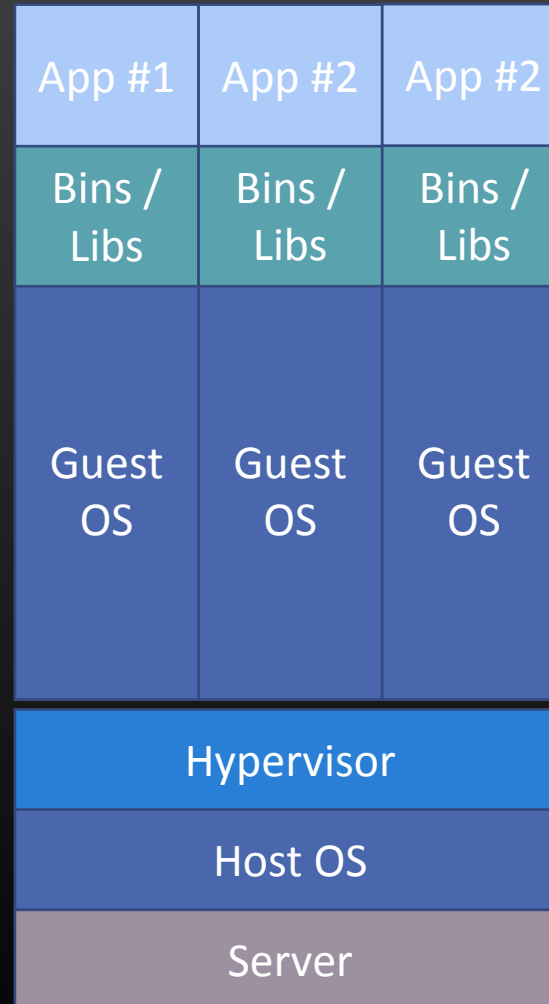
Created: 11/05/2018 08:30:21
Configuration Version: 8.2
Generation: 2
Notes: None

Clustered: No
Heartbeat: OK (No Application Data)

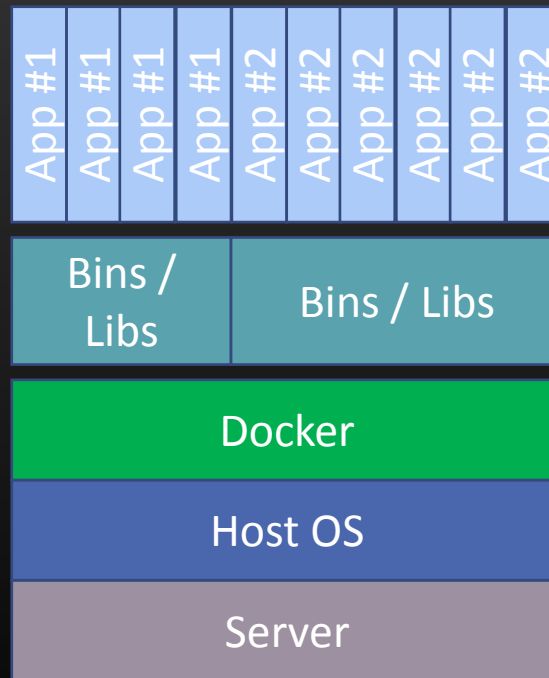
Actions

- JEUKBRILAP20190
 - Quick Create...
 - New
 - Import Virtual Machine...
 - Hyper-V Settings...
 - Virtual Switch Manager...
 - Virtual SAN Manager...
 - Edit Disk...
 - Inspect Disk...
 - Stop Service
 - Remove Server
 - Refresh
 - View
 - Help
- MobyLinuxVM
 - Connect...
 - Settings...
 - Turn Off...
 - Shut Down...
 - Save

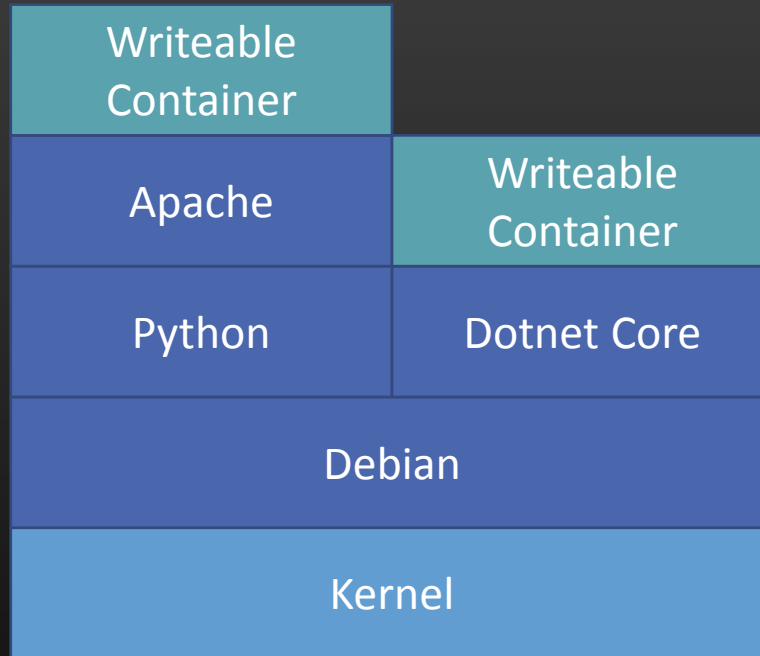
Traditional VMs



Containerised Application Model



Union Filesystem



First Steps



What is Docker?

Product

Community

Support ▾



Create Docker ID

Sign In

PRODUCT

Explore the Docker product that is right for you

GET DOCKER COMMUNITY EDITION

GET DOCKER ENTERPRISE EDITION



Overview

Subscriptions

Resources

DOCKER ENTERPRISE EDITION (EE)

Docker Enterprise Edition is the only enterprise-ready container platform that enables organizations to accelerate digital and multi-cloud initiatives by automating the delivery of legacy and modern applications using an agile operating model with integrated security.



FREEDOM OF CHOICE



AGILE OPERATIONS



INTEGRATED SECURITY

<https://www.docker.com/get-docker>

```
$ docker run hello-world
```

Basic docker cheatsheet

- `docker run`
 - Runs a command in a new container.
- `docker start`
 - Starts one or more stopped containers
- `docker stop`
 - Stops one or more running containers
- `docker build`
 - Builds an image from a Docker file
- `docker exec`
 - Runs a command in a run-time container
- `docker search`
 - Searches the Docker Hub for images

Official repositories

- Provide essential base OS repositories (for example, ubuntu, centos)
- Provide drop-in solutions for popular programming language runtimes, data stores, and other services
- Exemplify Dockerfile best practices and provide clear documentation to serve as a reference for other Dockerfile authors.
- Ensure that security updates are applied in a timely manner.

Writing your first Dockerfile

Basic Dockerfile commands

- FROM
 - The base image to use in the build. This is mandatory and must be the first command in the file.
- COPY / ADD
 - Copies a file from the host system onto the container
- CMD / ENTRYPOINT
 - The command that runs when the container starts
- ENV
 - Sets an environment variable in the new container
- EXPOSE
 - Opens a port for linked containers
- RUN
 - Executes a command and save the result as a new layer
- USER
 - Sets the default user within the container
- VOLUME
 - Creates a shared volume that can be shared among containers or by the host machine
- WORKDIR
 - Set the default working directory for the container

```
FROM node:8-slim
```

```
WORKDIR /server
```

```
COPY . /server
```

```
RUN npm install
```

```
EXPOSE 3000
```

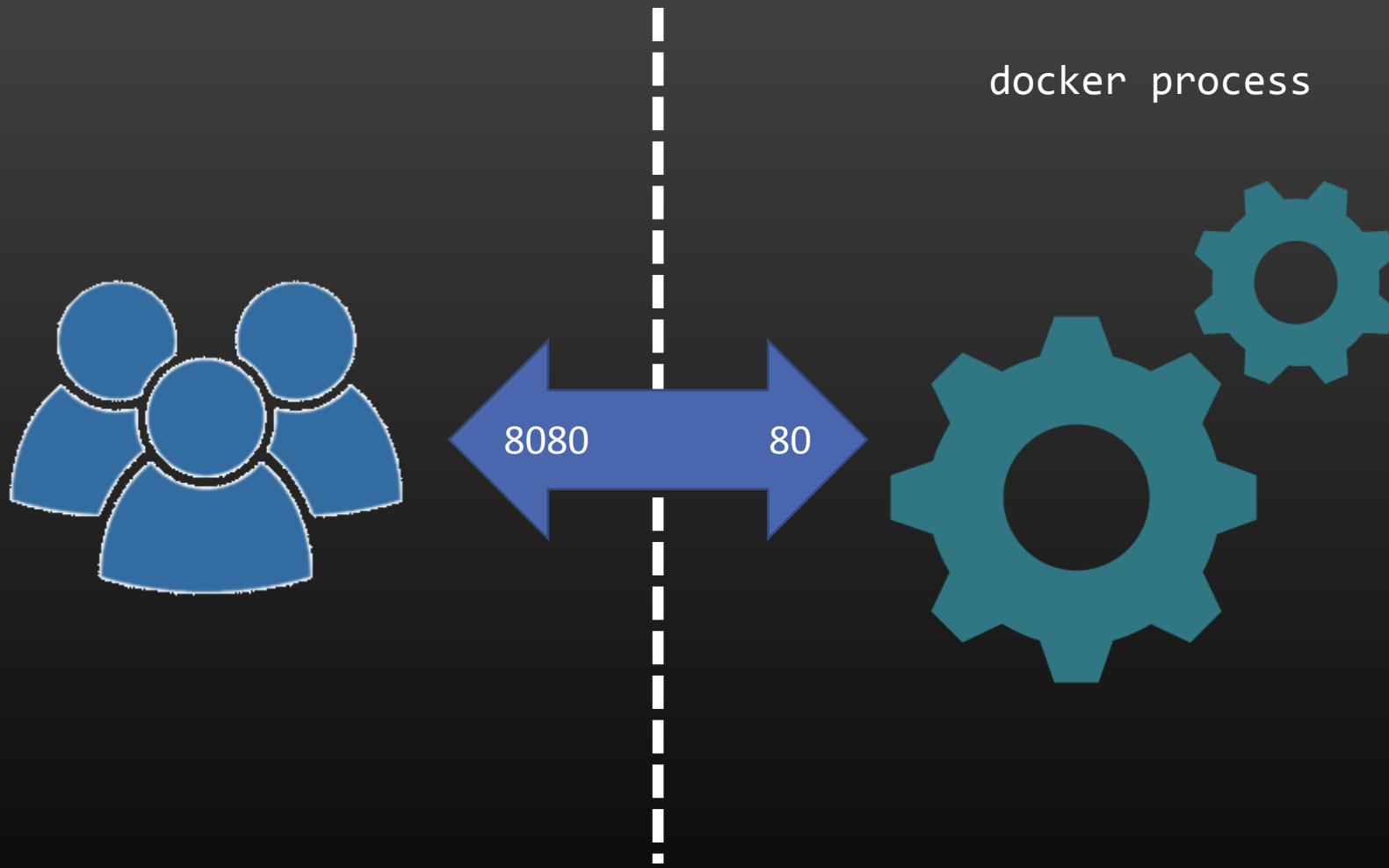
```
CMD [ "npm", "start" ]
```

Shell Form vs. Exec Form

- CMD executable param1 param2
 - Executable is launched using `/bin/sh -c`
- CMD ["executable", "param1", "param2"]
 - Executable is launched without the use of shell

Exec form is preferred in most cases

Exposing yourself to the
world

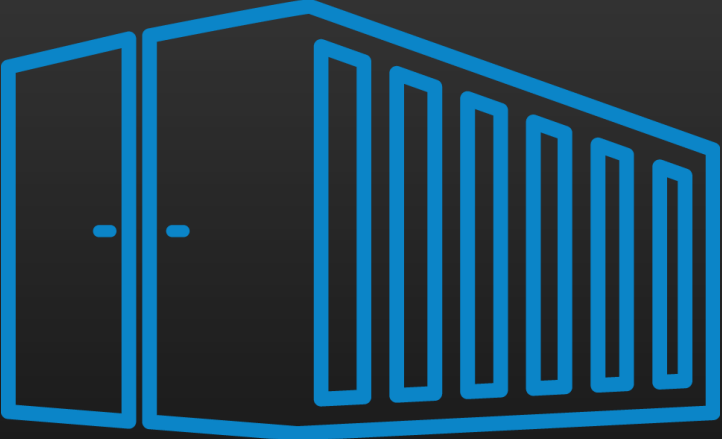


```
docker run --publish 8080:80 ...
```

Debugging a NodeJS application in VSCode



Attach to process



docker-compose

- Tool for defining and running multi-container docker applications
- YAML file to configure your application's services
- Create and start all the services from your configuration with a single command

Multi-Stage Build Files

```
FROM golang:1.7.3
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .
```

```
FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=0 /go/src/github.com/alexellis/href-counter/app .
CMD ["/app"]
```

Multiple containers

reverseproxy

nginx

proget

postgres

volume mounts

volume mounts

Going the whole hog...

- Create an ASP.NET Core MVC app
- Create a multi-stage Docker build
 - microsoft/aspnetcore-build for build process
 - microsoft/aspnetcore for the executable image
- Push image to Amazon AWS repository
- Run the container in the cloud

Questions?