DOCKER INTRODUCTION

JULIEN MAITREHENRY

Dev & Ops @PetalMD

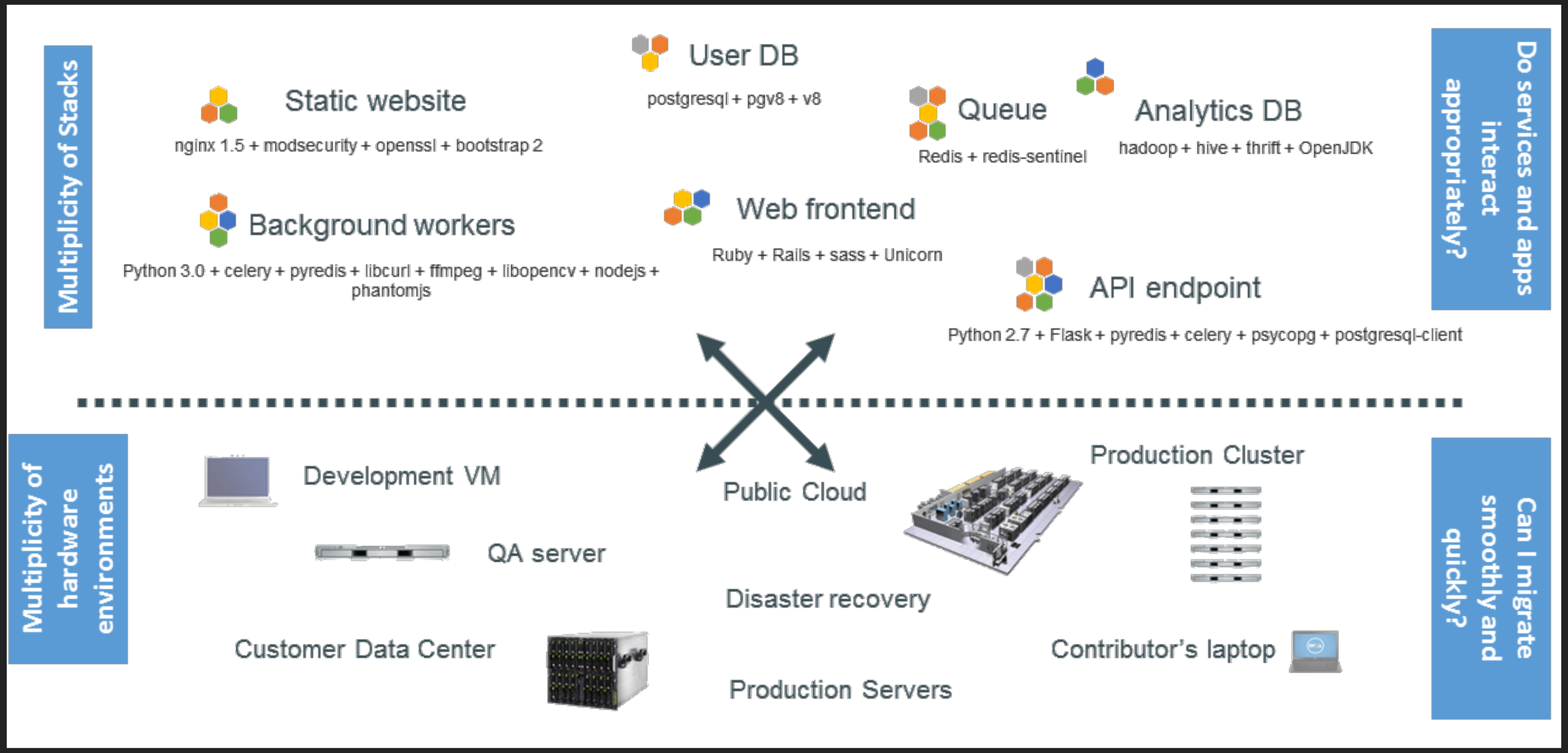
https://github.com/jmaitrehenry

@jmaitrehenry

AGENDA

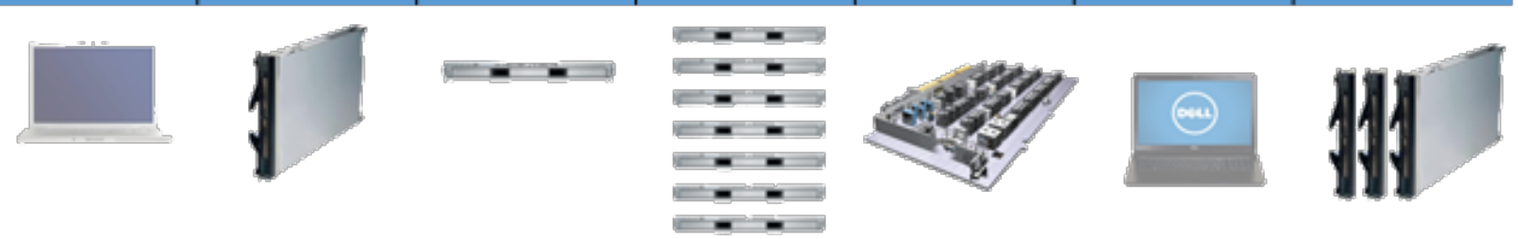
- The Challenge
- Docker as solution
- But What is Docker?
- What's an image?
- What's a container?
- Questions?

THE CHALLENGE

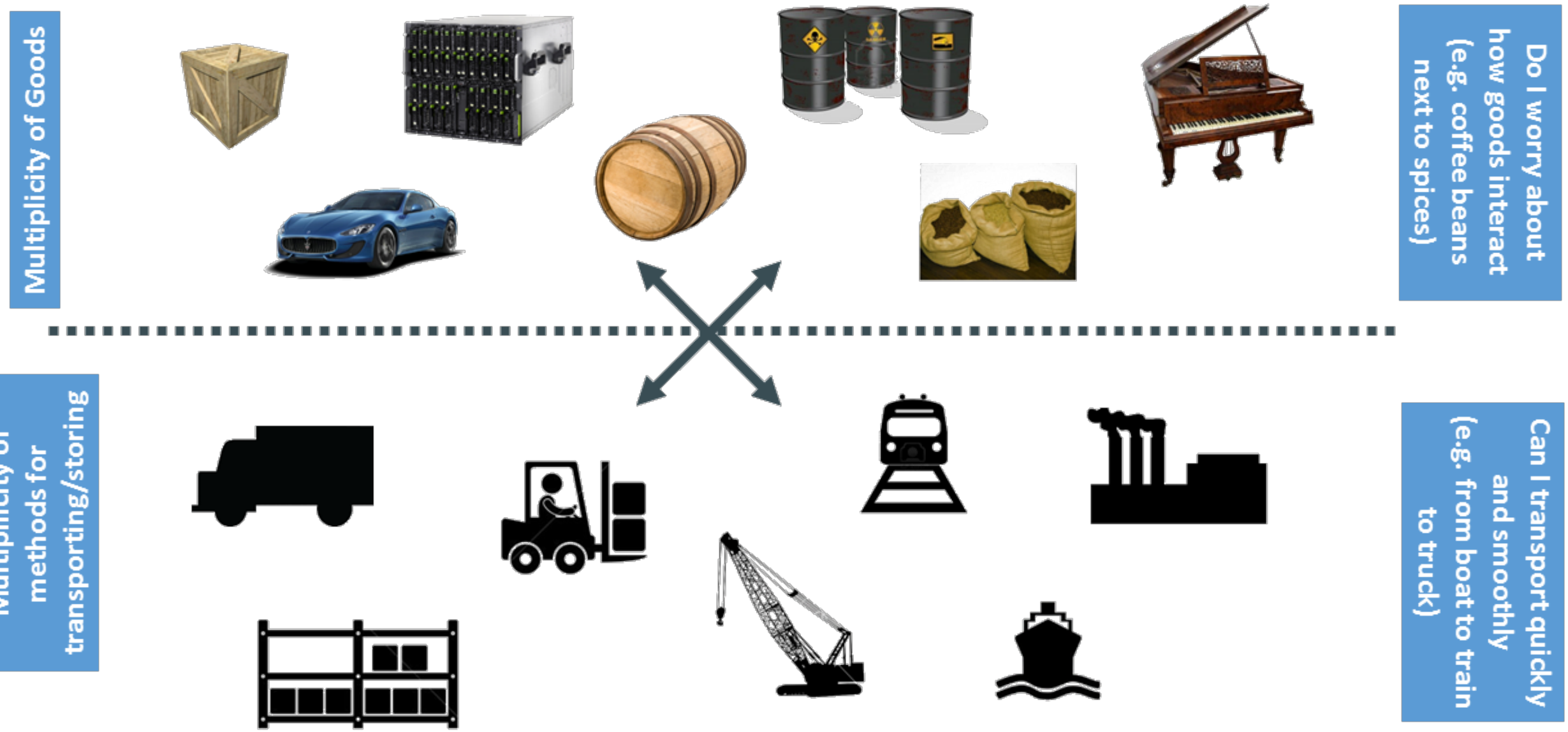


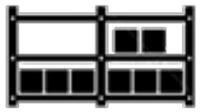
THE MATRIX FROM HELL

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?



CARGO TRANSPORT PRE-1960





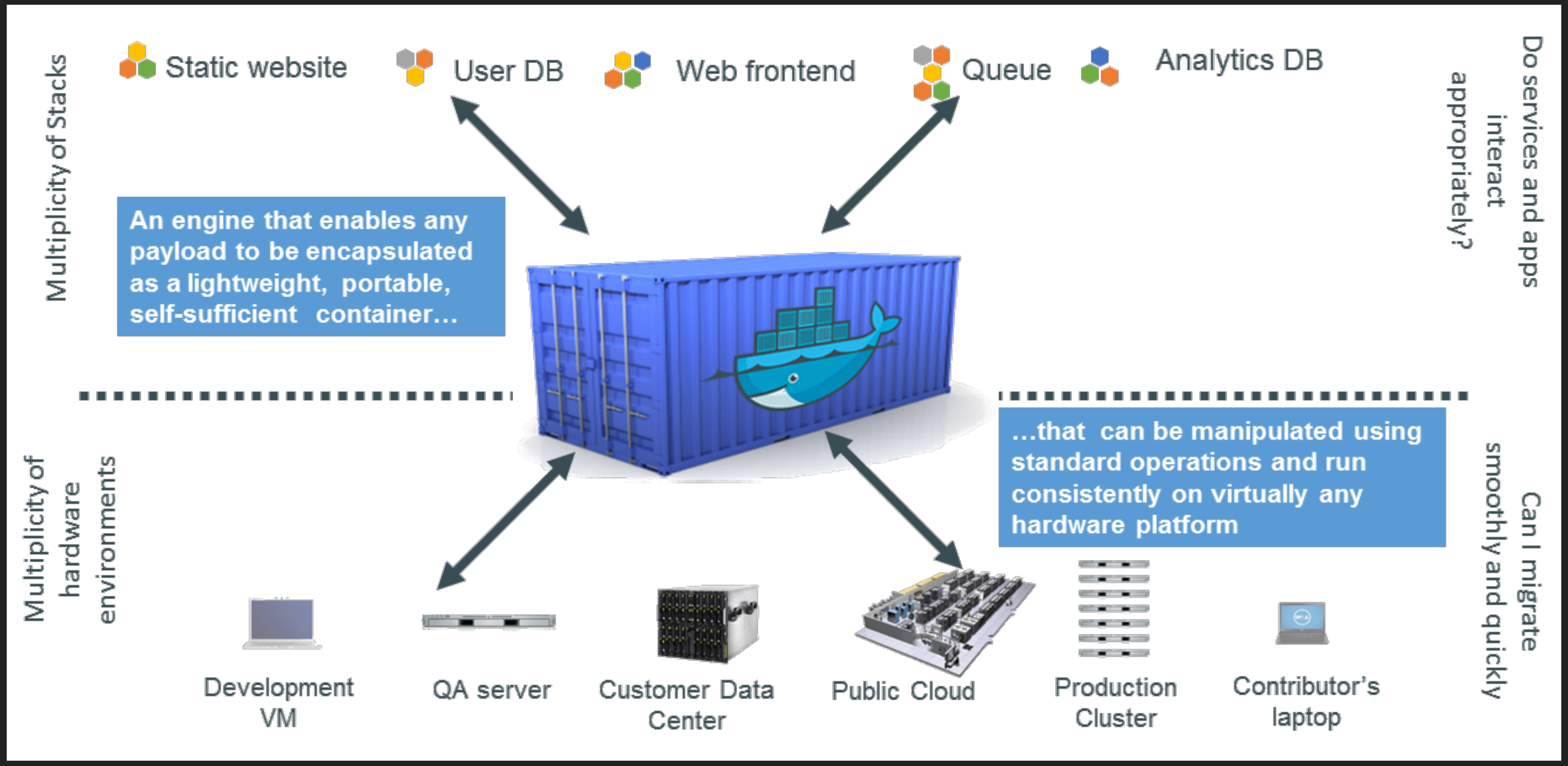
THE CHALLENGE

ALSO A MATRIX FROM HELL

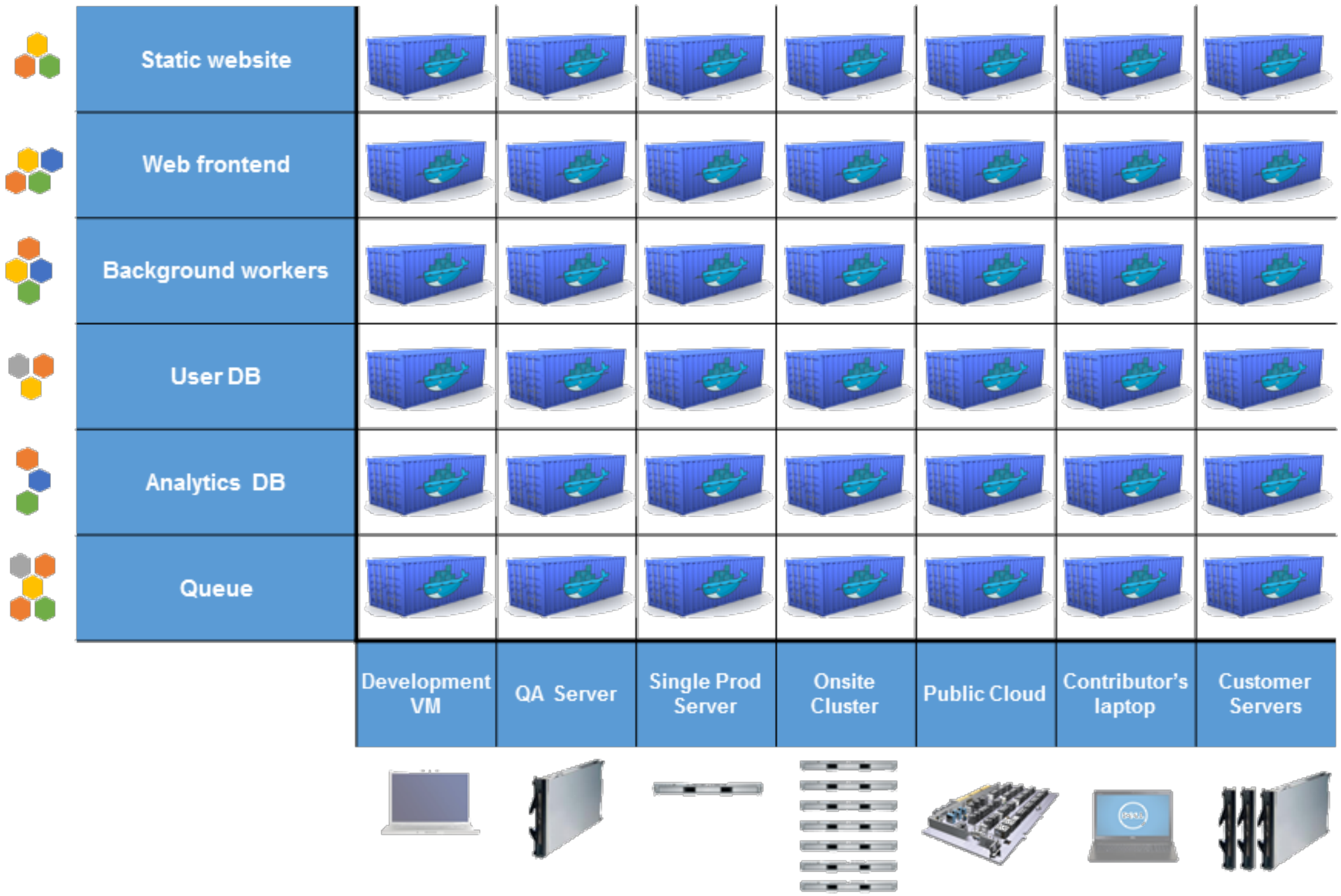
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

SOLUTION: INTERMODAL SHIPPING CONTAINER

DOCKER IS A CONTAINER SYSTEM FOR CODE



DOCKER ELIMINATES THE MATRIX FROM HELL



BUT WHAT IS DOCKER?

BUT WHAT IT'S DOCKER?

Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries — anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment

DOCKER STATS

32,000+
GitHub Stars

4B+
Docker Container Downloads

450,000+
Dockerized Apps In Docker Hub

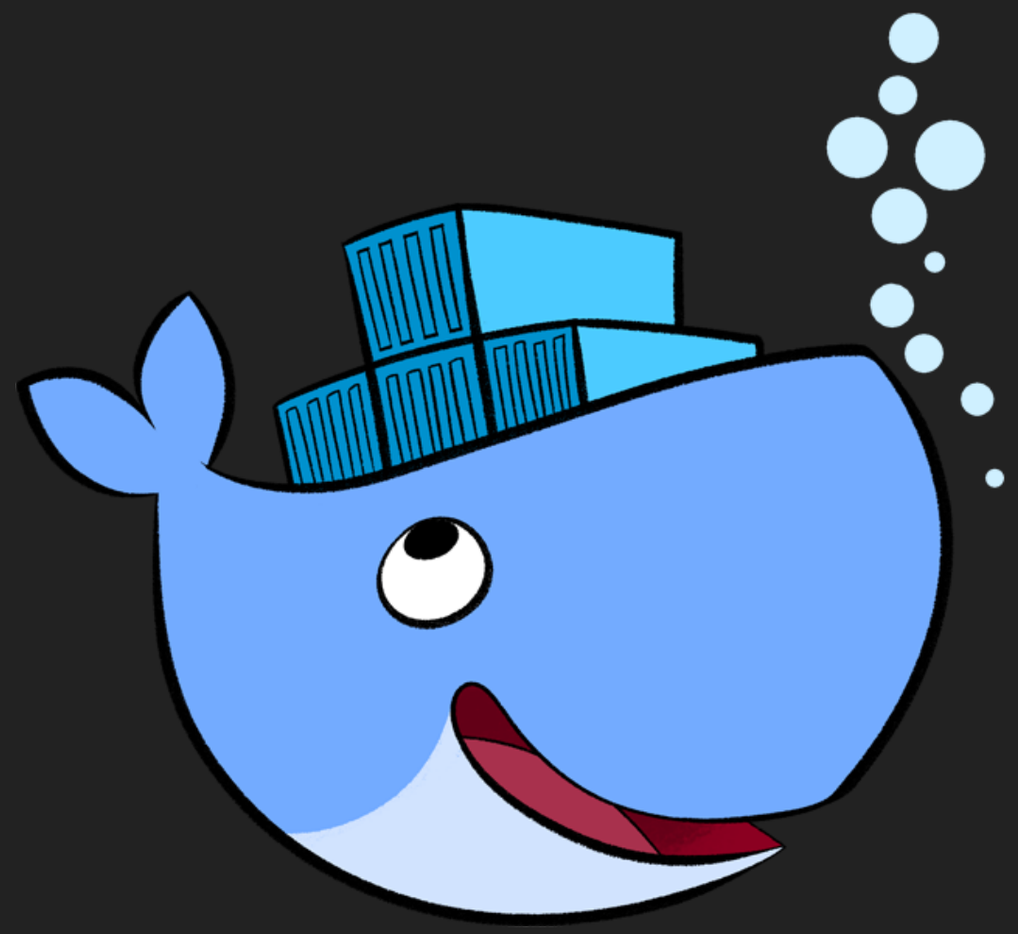
250+
Meetup Groups In 70+ Countries

2900+
Community Contributors

95,000+
Third Party Projects Using Docker

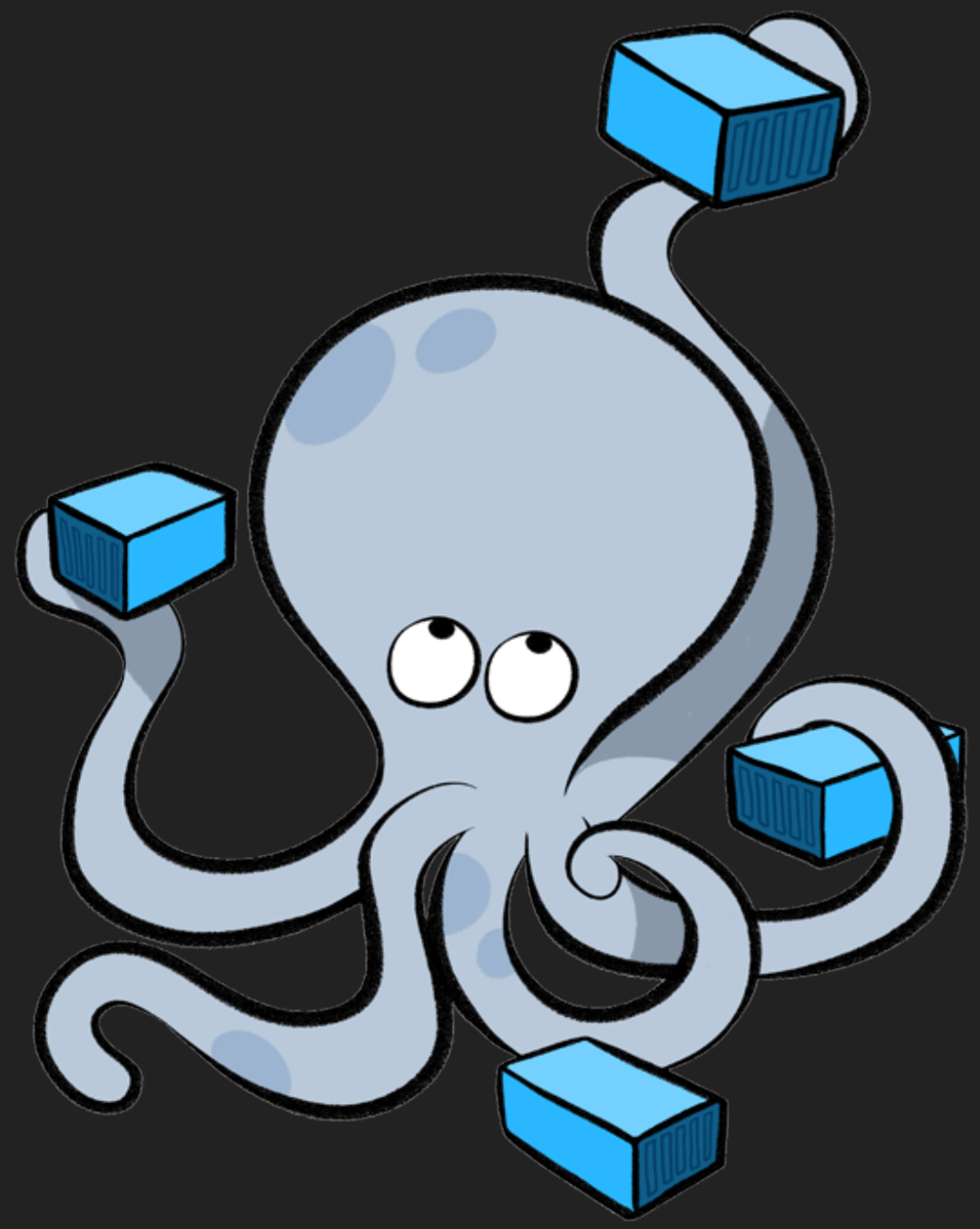
DOCKER ENGINE

- Lightweight runtime and robust tooling that:
  - Build image
  - Ship image
  - Run containers



DOCKER COMPOSE

- Define multi-container app with multiple dependencies in one file
- Hold structure and configuration in a single place
- Spin application up (or down) with a single command

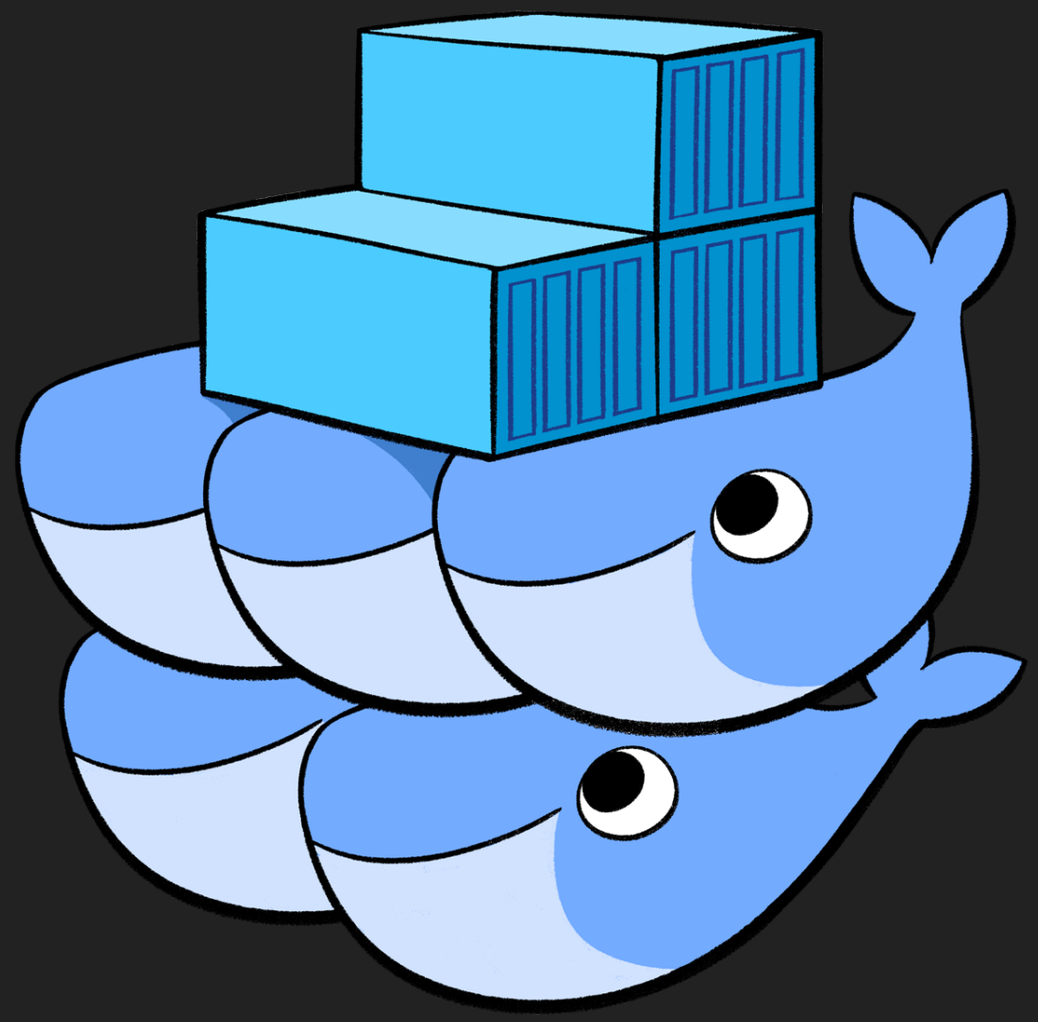


DOCKER REGISTRY

- Store and distribute Docker Image
- Many hosted registries available:
  - Docker Hub, AWS ECR, …
- Could be self hosted with many storage backends:
  - Local, AWS S3, Ceph, OpenStack Swift, …

DOCKER SWARM

- Turn group of Docker Engine into a single, virtual Docker Engine
- Native clustering capabilities
- High Scalability
- Failover & High Availability
- Built-in scheduler
- Node discovery
- ...



SEPARATION OF CONCERNS: FRANCIS THE DEVELOPER

- Inside my container:
  - My code
  - My libraries
  - My package manager
  - My app
  - My data

SEPARATION OF CONCERNS: JULIEN THE OPS GUY

- Outside the container:
  - Logging
  - Remote access
  - Network configuration
  - Monitoring
  - High Availability

WHAT'S AN IMAGE?

DEFINITION

- Each Docker image references a list of read-only layers that represent filesystem differences
- Layers are stacked on top of each other to form a base for a container's root filesystem
- Each layer are Read Only
- The Docker storage driver is responsible for stacking these layers and providing a single unified view
- Each layer can be shared with other image

IT'S A SIMPLE FILE – A DOCKERFILE

```
FROM mhart/alpine-node:6

ENV SERVICE_3000_NAME selfpro
WORKDIR /src

ADD . .
RUN apk add --update --no-cache git && \
    npm install

EXPOSE 3000

CMD npm start
```

BUILDING...

docker build -t petalmd/selfpro -f docker/Dockerfile.app .

```
Sending build context to Docker daemon 1.398 MB
Step 1 : FROM mhart/alpine-node:6
 ---> 2e8721f40082
Step 2 : ENV SERVICE_3000_NAME selfpro
 ---> Running in cd8e6156fbec
 ---> 9d9b1f86a696
Removing intermediate container cd8e6156fbec

[...]

Step 7 : CMD npm start
 ---> Running in 71bab8276b39
 ---> cce4c583c3a0
Removing intermediate container 71bab8276b39
Successfully built cce4c583c3a0
```

RESULT ON A STACK OF LAYERS

docker history cce4c583c3a0

docker history petalmd/selfpro

```
IMAGE          CREATED        CREATED BY                                      SIZE
cce4c583c3a0   2 minutes ago  /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "npm s   0 B
110c49285fe1   2 minutes ago  /bin/sh -c #(nop)  EXPOSE 3000/tcp              0 B
579469b24848   2 minutes ago  /bin/sh -c apk add --update --no-cache git &&   54.16 MB
5b34c5ef9a7f   3 minutes ago  /bin/sh -c #(nop) %s %s in %s  ADD dir:b07dd1   1.044 MB
6ba4a61f509a   3 minutes ago  /bin/sh -c #(nop)  WORKDIR /src                 0 B
9d9b1f86a696   3 minutes ago  /bin/sh -c #(nop)  ENV SERVICE_3000_NAME=self   0 B
2e8721f40082   3 weeks ago    /bin/sh -c apk add --no-cache curl make gcc g   41.44 MB
<missing>      3 weeks ago    /bin/sh -c #(nop) ENV VERSION=v6.2.1 NPM_VERS   0 B
<missing>      3 weeks ago    /bin/sh -c #(nop) ADD file:701fd33a2f463fd4bd   4.799 MB
```

WHY IMAGE ARE SO LIGHTWEIGHT?

- Each image can share layers
- An image can be build on top of other images
  - A static website can be build on top of nginx image which is build on top of an alpine image
- Building an image could result on rebuilding only last top layers

EXAMPLE

redis:alpine

CMD ["redis-server"]

[9 other layers]

ENV REDIS_VERSION=3.2.1

apk add --no-cache 'su-exec>=0.2' [...]

addgroup -S redis && adduser -S -G [...]

nginx:alpine

CMD ["nginx" "-g" "daemon off;"] 0 B

EXPOSE 443/tcp 80/tcp

[5 other layers]

ENV NGINX_VERSION=1.11.1

MAINTAINER NGINX Docker Maintainers

/bin/sh -c #(nop) ADD file:614a9122187935fccf 4.797 MB

alpine:3.3

WHAT'S A CONTAINER

IT IS A RUNNING IMAGE
WITH A R/W TOP LAYER

RUN EVERYWHERE

- Regardless of kernel version
- Regardless of host distribution
- Physical or virtual, cloud or not
- Container and host architecture should match

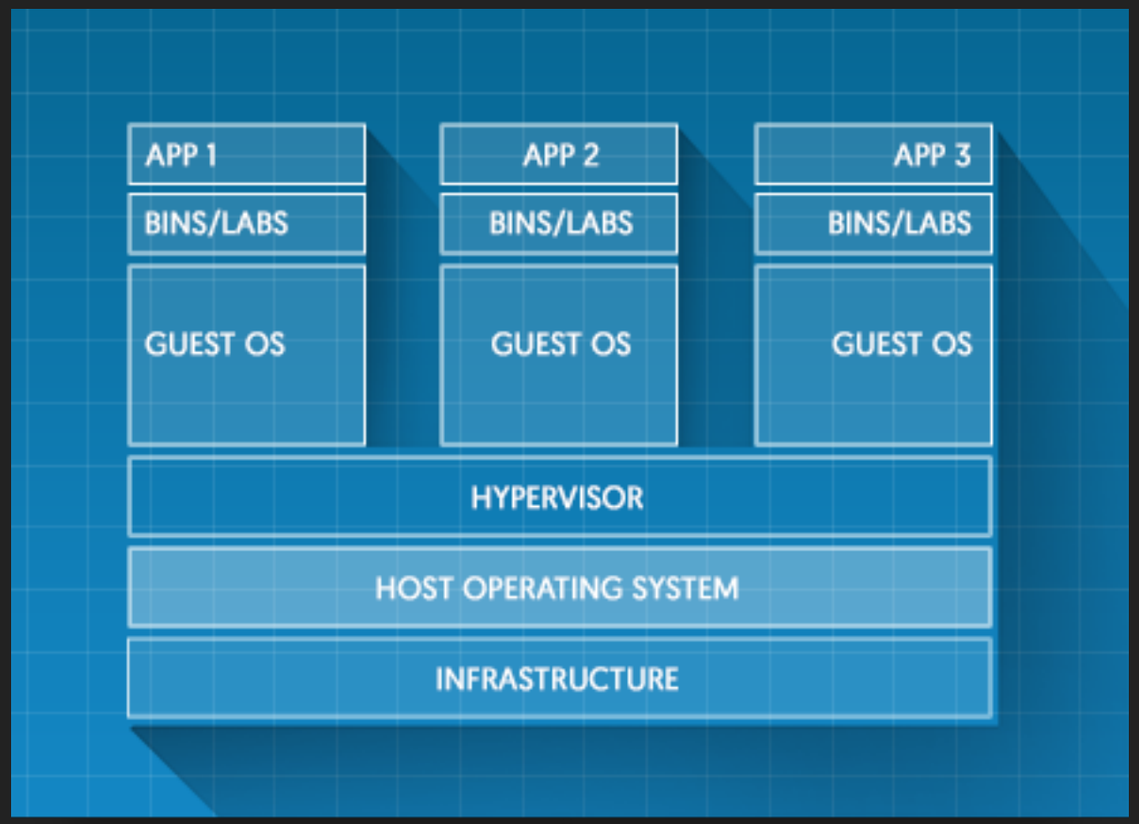
IT'S A LIGHTWEIGHT VM

- Own process space
- Own network interface
- Can run stuff as root or as any custom user

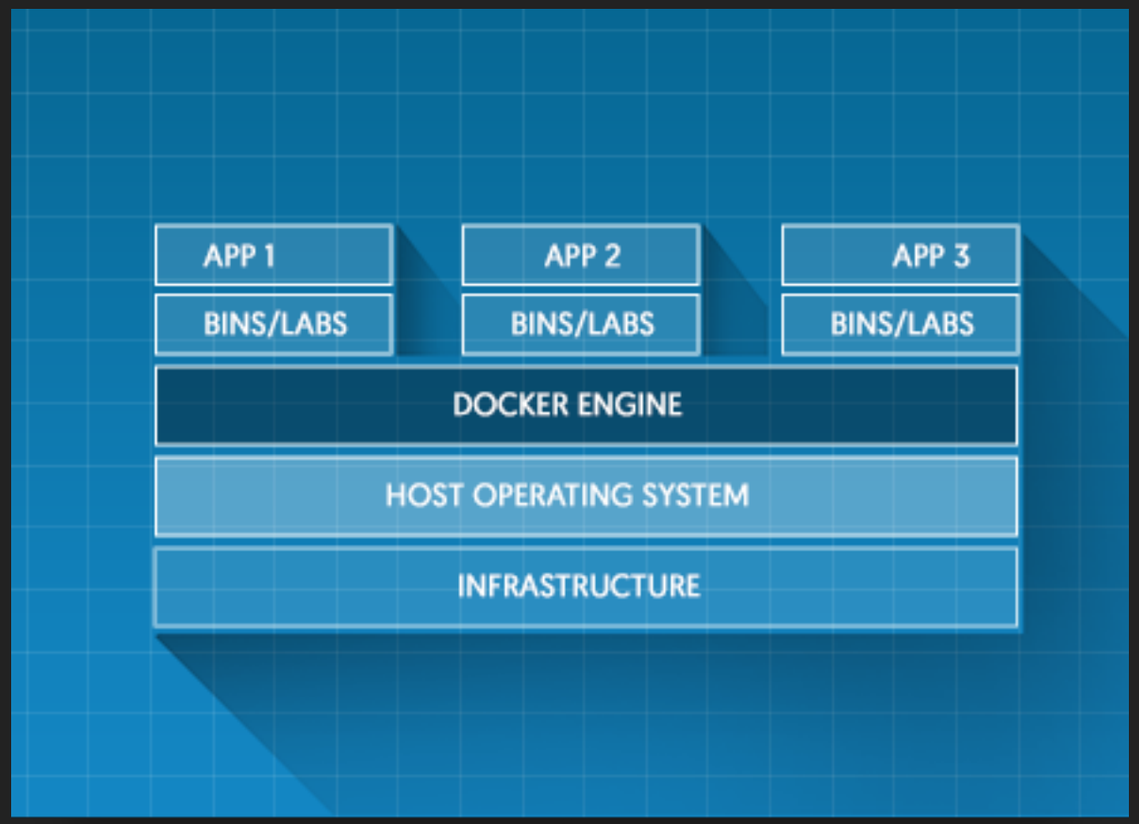
IT'S A CHROOT ON STEROIDS

- Container = isolated process
- Share kernel with host (linux and windows containers)
- No device emulation (neither HVM nor PV)

WHAT'S A CONTAINER



VIRTUAL MACHINES



CONTAINERS

COMPUTE EFFICIENCY

- CPU performance = native performance
- Memory performance = ~native performance
- Network performance = small overhead; can be reduced to zero

STORAGE EFFICIENCY

- Depend on storage driver

AUFS	stable	production-ready	good memory use	smooth Docker experience	high write activity	PaaS-type work
Devicemapper (loop)	stable	in mainline kernel	smooth Docker experience	production	performance	lab testing
Devicemapper (direct-lvm)	stable	production-ready	in mainline kernel	smooth Docker experience	PaaS-type work	
Btrfs	in mainline kernel	high write activity	container churn	build pools		
Overlay	stable	good memory use	in mainline kernel	container churn	lab testing	
ZFS native (ZoL)	PaaS-type work					
ZFS FUSE	stable	lab testing	production			

Key

Has attribute	attribute
If good for use case	use case
If bad for use case	use case

QUESTIONS?