Containers

Docker and 12 Factor Apps

Who we are

Luk Burchard

Computer Science Student
Software Engineer @ Loodse

twitter.com/lukburchard
github.com/realfake

Matthias Loibl

Computer Science Student
Software Engineer @ Loodse

twitter.com/metalmatze
github.com/metalmatze

Who are you?

- Developer?
- SysAdmin/Ops?
- Data Science?
- ...

Who are you?

- Who knows about Docker?
- Who knows about Kubernetes?
- Who uses Docker for Development?
- Who uses Docker in Production?

Who tried to use Docker, but couldn't do it?

Deployment

How do you deploy your apps?

Deployment

How do you deploy your apps?

Do you like SSH?

Deployment

How do you deploy your apps?

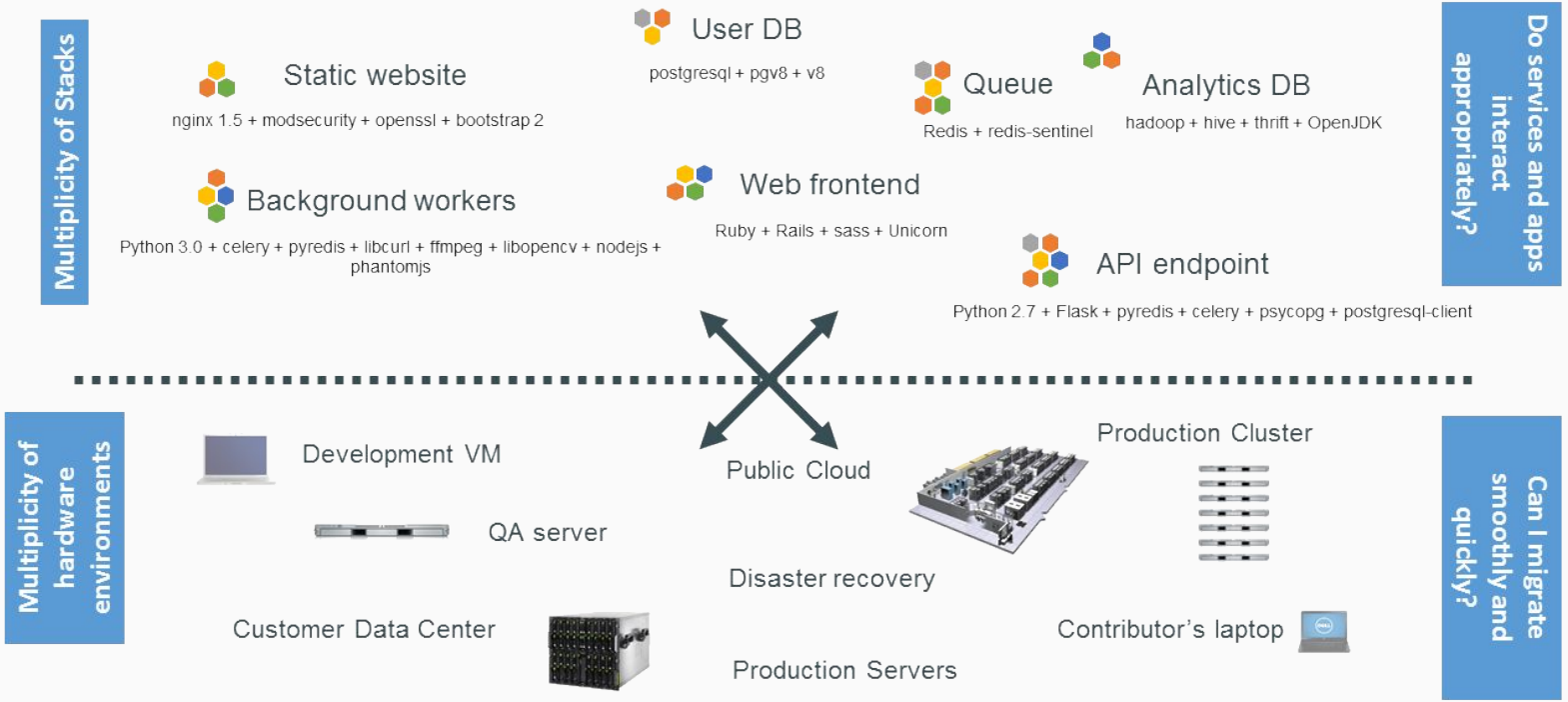
Do you like SSH?
Do you like SSH on 5 Servers?

Deployment

How do you deploy your apps?

Do you like SSH?
Do you like SSH on 5 Servers?
Do you like SSH on 100 Servers?

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

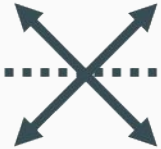
Multiplicity of Goods







Do I worry about how goods interact (e.g. coffee beans next to spices)

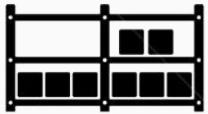


Multipilicity of methods for transporting/storing



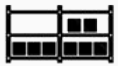






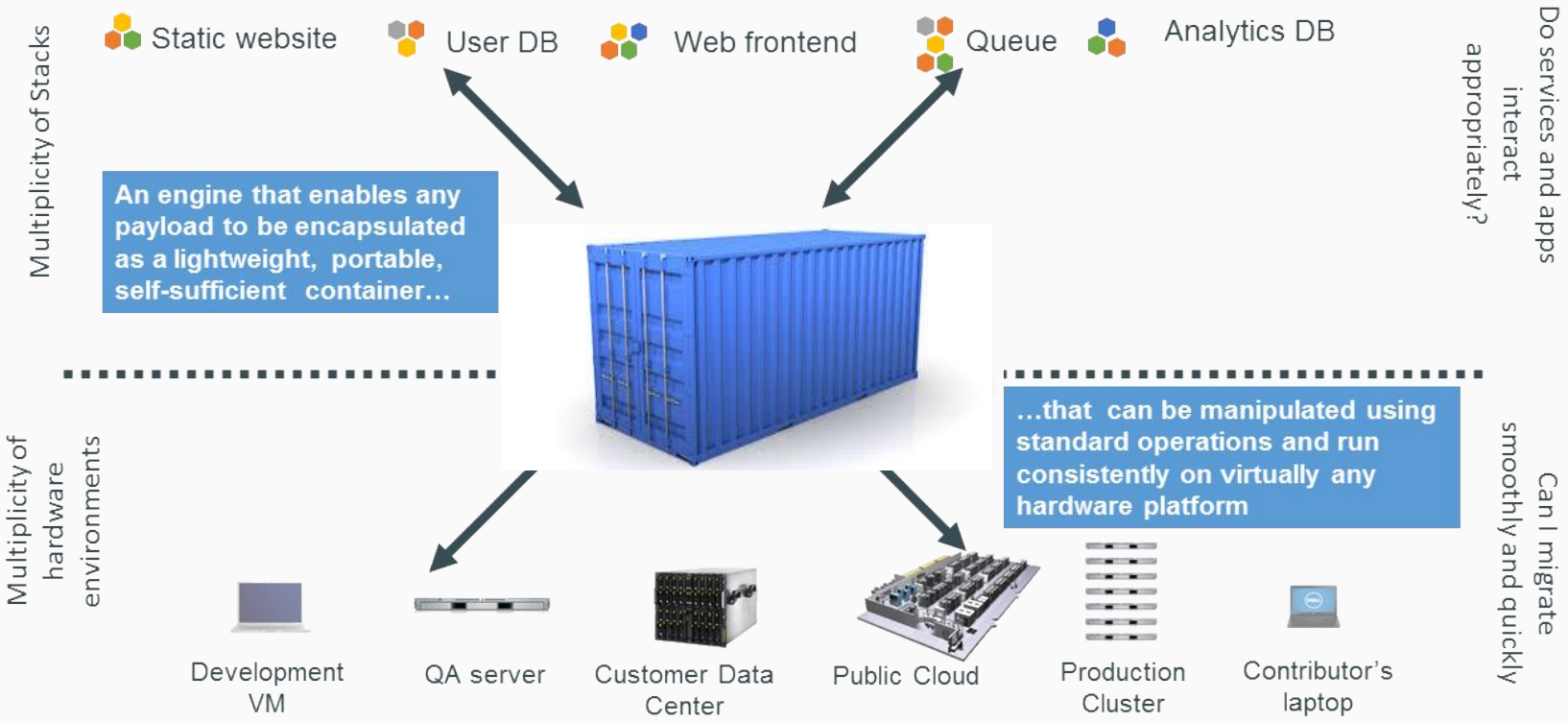
Can I transport quickly and smoothly (e.g. from boat to train to truck)

Another Matrix from Hell

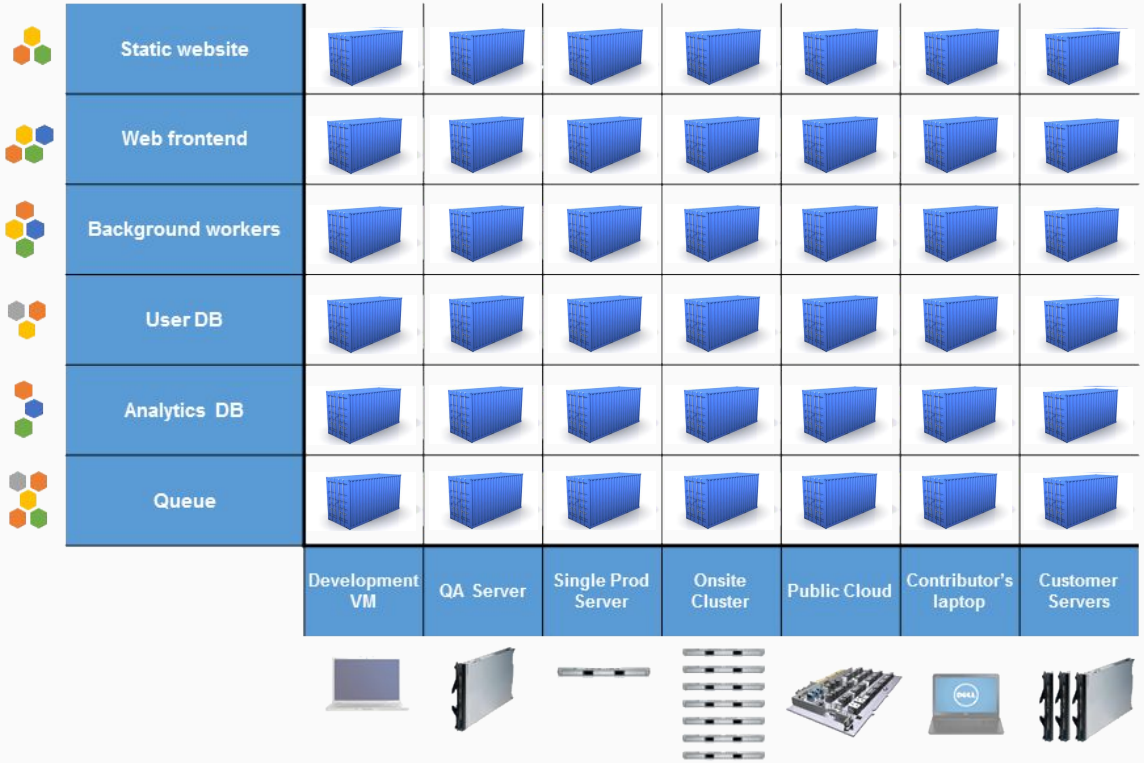
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

Solution: Shipping Container

Docker: Container for shipping Software



Eliminate the Matrix from Hell



What is a Container?

chroot

- chroot = change root
- Extract a filesystem to `/mnt`
- Change the root to `/mnt`
  - Uses the same (Linux) Kernel as before

Installing or repairing a Linux System with chroot

cgroups & namespaces

cgroups

limit & isolate the resource usage

Example:
Kill process using more than 256MB memory

namespaces

isolate and virtualize system resources of a collection of processes

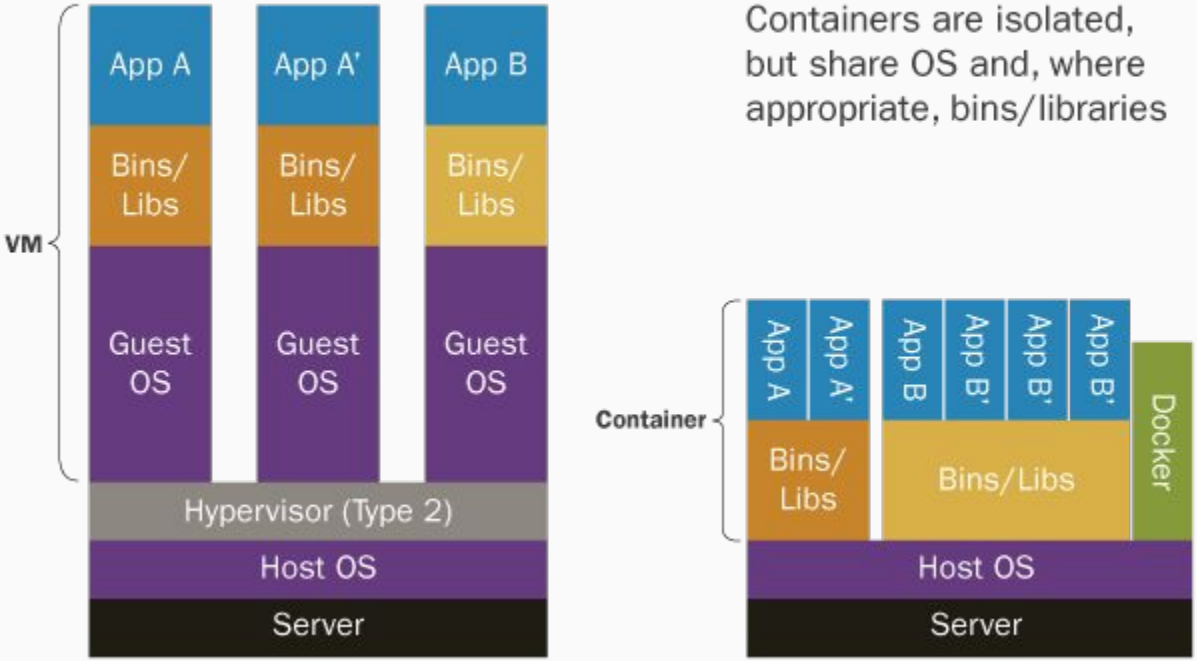
- Mount
- Process ID
- Network
- User ID
- cgroups

LXC (Linux Containers)

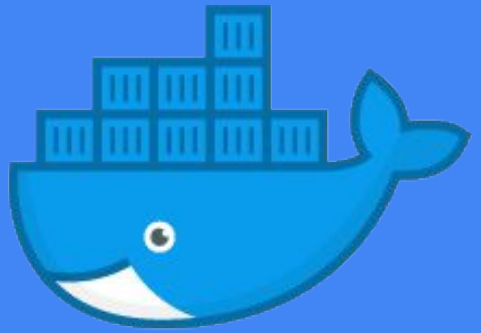
- Operating-system-level virtualization
- Run multiple isolated Linux systems on a single Linux kernel

- Combines **cgroups** and **namespaces** to run Linux Containers

Containers vs. VMs

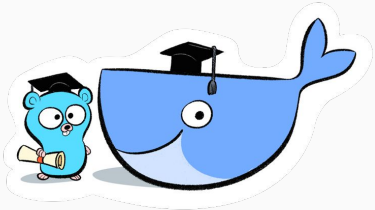


Docker



What is this Docker?

- Written in Go
- Released on March 13th, 2013
- Client-Server:
  - Docker Engine (daemon)
  - Docker Client, CLI
- Ready for production use
- Used LCX to run Containers
  - Uses cgroup, namespaces and OverlayFS
- Use their own libcontainer implementation



What does Docker provide?

- Run in the same environment
  Run in a lightweight environment
- Run in a sandboxed environment
- Pull images with all its dependencies

OCI (Open Container Initiative)

- Standard for container formats and runtimes
  - Standardizes how images are unpacked on the filesystem
  - Standardizes how containers are run from images
- Under auspices of the Linux Foundation
- **docker**, **rkt** and others now run the same specification
- **runc** is an OCI implementation

Install Docker

- Docker on Linux, ask your package manager
- Docker for Mac
- Docker for Windows

Run $ docker version

Use our GCP Codes

Docker Group on Linux

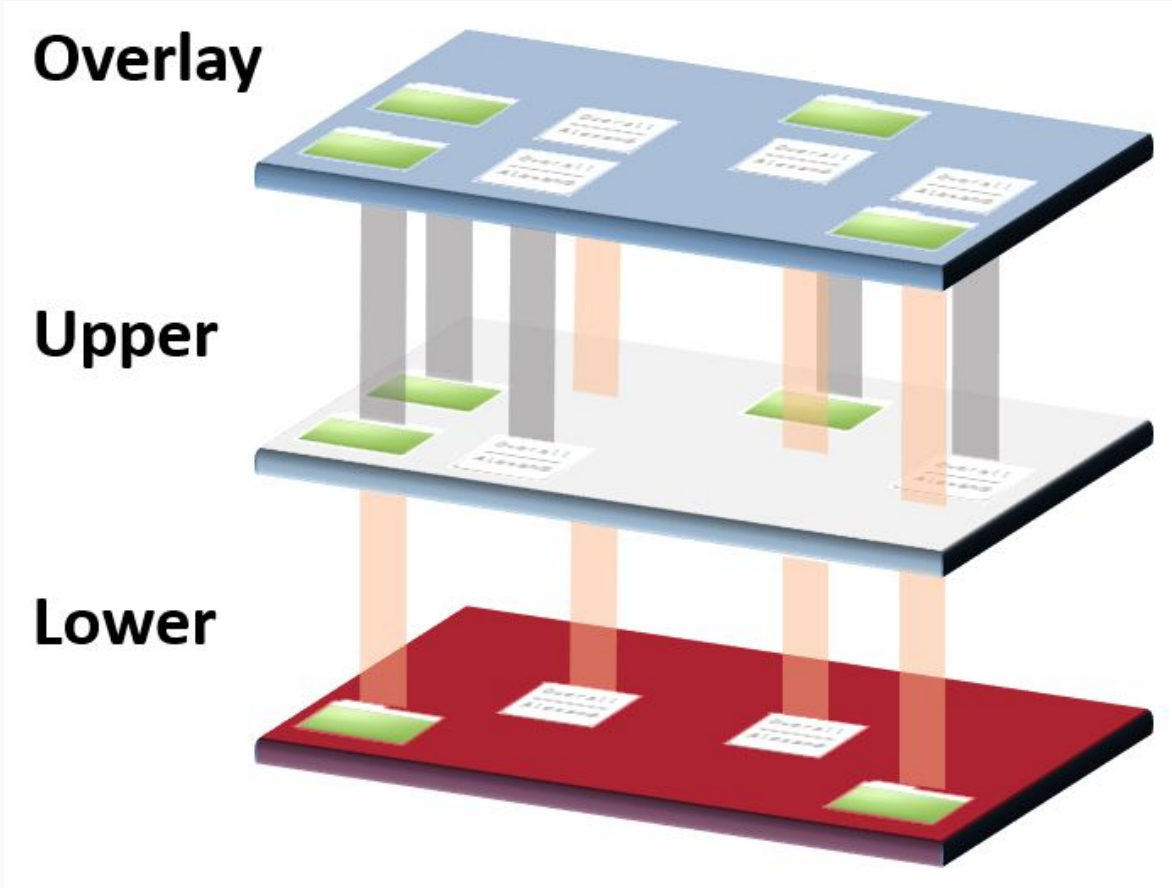
```
# Add the Docker group
$ sudo groupadd docker

# Add yourself to the group
$ sudo gpasswd -a $USER docker

# Restart the Docker daemon
$ sudo systemctl restart docker

$ docker ps # run docker without sudo
```

Docker Client

Excerpt of most important docker commands

docker build	Build an image from a Dockerfile
docker exec	Run a command in a running container
docker inspect	Return low-level information on Docker objects
docker kill	Kill one or more running containers
docker logs	Fetch the logs of a container
docker pull	Pull an image or a repository from a registry
docker push	Push an image or a repository to a registry
docker rm	Remove one or more images
docker run	Run a command in a new container
docker stop	Stop one or more running containers
docker tag	Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

OverlayFS

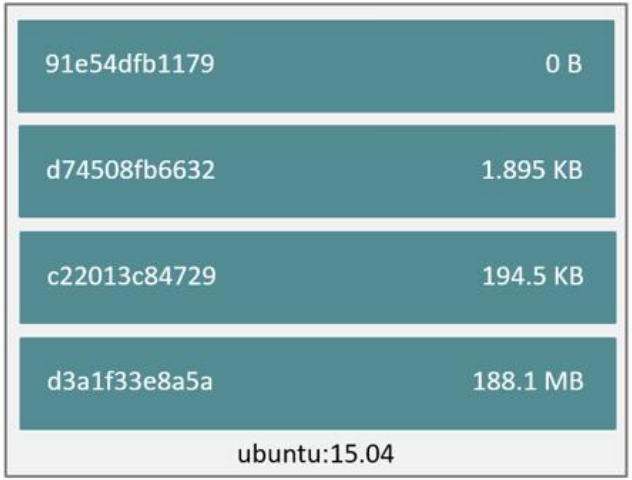


Image

OverlayFS:
each layer ‘overlays’
the lower layer

Image

- CMD ["/bin/bash"]
- mkdir -p /run/systemd && echo '...
- sed -i 's/^#\s*\(deb.*universe\...
- ADD file:280a445783f309c..

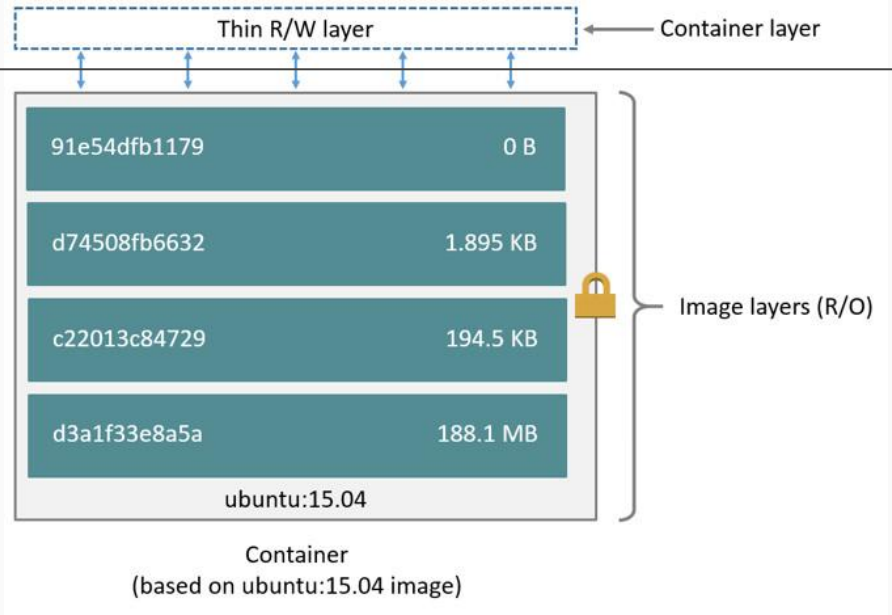


Image

Container

The Container
(a running program)

The Image
(a blueprint for a container)



Container Registries

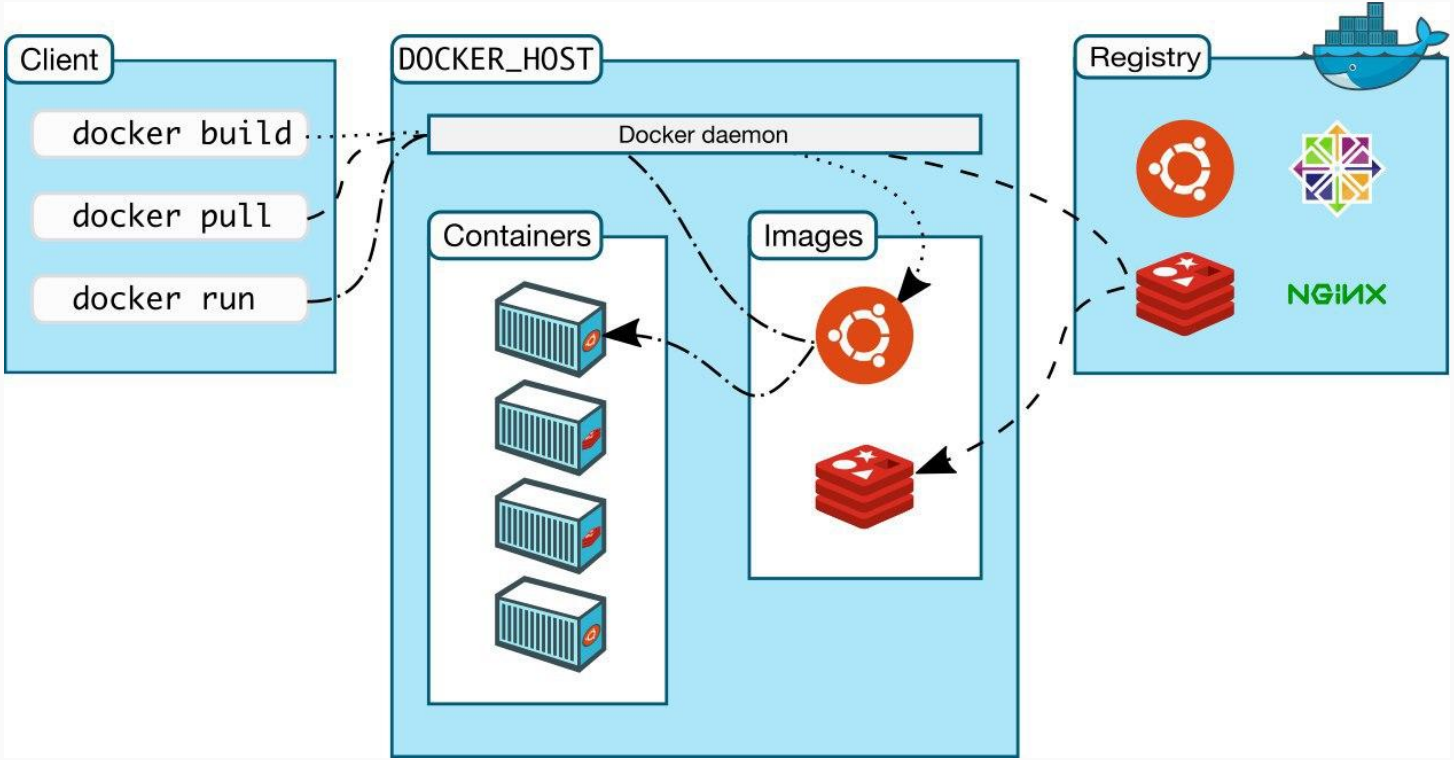
- hub.docker.com
  - Docker's official Registry
- quay.io
  - Public Registry by CoreOS
- cloud.google.com/container-registry
  - Shorter: *gcr.io/google_containers/pause-amd64*
  - Often used in combination with Kubernetes
- Host your own private Registry

Container Registry Commands

Use docker CLI to authenticate

```
$ docker login
$ docker logout

# Login to a private registry
$ docker login registry.example.com
```

Container Architecture



Run a Container

$ docker run alpine echo 'hello world'
$ docker ps

What did just happen?

- Pulled alpine image from the registry
- Created a new container
- Allocated a filesystem and mounts a read-write layer
- Allocated a network/bridge interface
- Sets up an IP address
- Executes a process that you specify (/bin/bash)
- Captures and provides application output

Run a long-lived Container

$ docker run --name hw alpine /bin/sh -c "while true; do echo hello world; sleep 1; done"
$ docker ps
$ docker logs (-f) hw

Ctrl+C the container
$ docker ps
$ docker ps -a

Run nginx in a Container

```
# Ports
$ docker run --rm -p 8080:80 nginx
$ docker run --rm -p 8080:80 nginx:1.13
$ docker run -d --name nginx -p 8080:80 nginx

# Volumes
$ docker run --rm -p 8080:80 -v /tmp/nginx:/usr/share/nginx/html:ro nginx
```

Dockerfile

- Build steps to create an image
- Invoke with “$docker build .”
- Output is and image
- Cache image layers

```
FROM alpine:latest

ADD hostsrc /containerdest
WORKDIR /pwdofcontainerstart

CMD ./main
```

Docker: "don't"s

- Don't store data in containers
  - All data will be lost
- Don't create large images
  - Use alpine
- Don't use only the latest tag
  - How would you rollback?
- Don't run more than one process in a single container

12-Factor Apps

https://12factor.net

12 Factors

1. Codebase
   Use something like **git**
2. Dependencies
   Use **dep, pip, gem, npm** etc…
3. Configuration
   Use **EnvVars**, not config files
4. Backing services
   Independent of depended services
   Example: DB, MySQL or RDS
5. Build, release, run
   Build a immutable release, use CI/CD
6. Processes
   Apps are just a stateless process
   Containers ;-)

12 Factors

7. Port binding
   Expose Apps via Ports
   Example: **HTTP:80**, Postgres:5432
8. Concurrency
   Keep horizontal scaling in mind
9. Disposability
   Fast start time, terminate on **SIGTERM**
   Container send SIGTERM ;-)
10. Dev/prod parity
    Deploy often, DevOps, run same containers in dev
11. Logs
    Streams, not files. Write to **stdout**
12. Admin processes
    Run admin tasks as one-off processes
    Example: Run script to migrate DB

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

12 Factor - Implications

- Portability
- Deployability
- Scalability
- Immutability

Let’s begin!

Shall we?

```
$ curl -L
https://storage.googleapis.com/cotbat/cotbat.zip >
cotbat.zip

unzip cotbat.zip
```

github.com/realfake/cotlaader
github.com/realfake/cotbat

Container Registry

You can host your own!
It's just a docker container with BasicAuth

```
# Run a registry locally
$ docker run -d -p 5000:5000 --name registry registry:2

# Use your images
$ docker tag project:1.2.3 registry.example.com/project:1.2.3
$ docker push registry.example.com/project:1.2.3
```