Docker From The Ground Up Part 1

A whistle-stop tour of containerisation, Docker images, Dockerfiles, sharing images, running containers and relevant aspects of best practice.

Who's This Guy?

Matt Todd

Principal Archiect @ Resonate.tech

Cloud Native Advocate

`github.com/mattjtodd`

`hub.docker.com/u/mattjtodd`

Thanks to our Sponsor!

BlackCat /


DID SOMEONE SAY
BEER?

The dawn of containers……. (Waaaaayyyyyy Pre DevOps)

Chroot

Chroot (Jail)

- Restricts a processes (and all children's) view of a servers FS to a specific portion
- Maps a process root FS to some other directory
- Prevents access to outside of the new root
- Introduced in **1979** Unix V7 the BSD **1982**

Fast Forward to 2000.....

FreeBSD Jails

- Allows administrators to partition a FreeBSD computer system into several independent, smaller systems – called “jails”
- Ability to assign an IP address for each system and configuration.

2001 - 2005

- 2001 - Linux VServer
- 2004 - Solaris COntainers
- 2005 - Open Virtuozzo

2006 - Process Containers

- Launched by Google in 2006) was
- designed for limiting, accounting and isolating resource usage (CPU, memory, disk I/O, network) of a collection of processes.
- It was renamed “Control Groups (cgroups)” a year later
- Merged to Linux kernel 2.6.24.

2008 LXC (LinuX Containers)

- First, most complete implementation of Linux container manager
- Implemented in 2008 using cgroups and Linux namespaces
- Works on a single Linux kernel without requiring any patches
- First “Standard” container platform

Warden & LMCTFY

- CloudFoundry started Warden in 2011
- using LXC in the early stages and later replacing it with its own
- Warden can isolate environments on any operating system, running as a daemon
- providing an API for container management.
- It developed a client-server model to manage a collection of containers across multiple hosts, and Warden includes a service to manage cgroups, namespaces and the process life cycle.

- Let Me Contain That For You (LMCTFY) kicked off in 2013 as an open-source version of Google's container stack, providing Linux application containers.
- Applications can be made “container aware,” creating and managing their own subcontainers.
- Active deployment in LMCTFY stopped in 2015 after Google started contributing core LMCTFY concepts to libcontainer, which is now part of the Open Container Foundation.

Container Process Isolation

Namespaces

- Resource allocation / isolation
  - Mount (Filesystem)
  - Process (PID)
  - User ID
  - Networks (Virtual Devices)
  - ...

CGroups

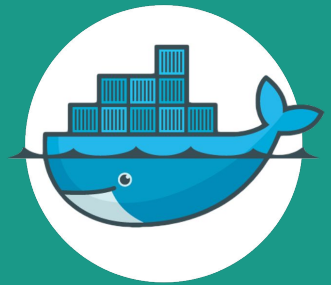
- Resource *usage* limits & Prioritisation
  - Memory
  - CPU DiskIO
  - ...

Solomon Hykes

Docker container engine

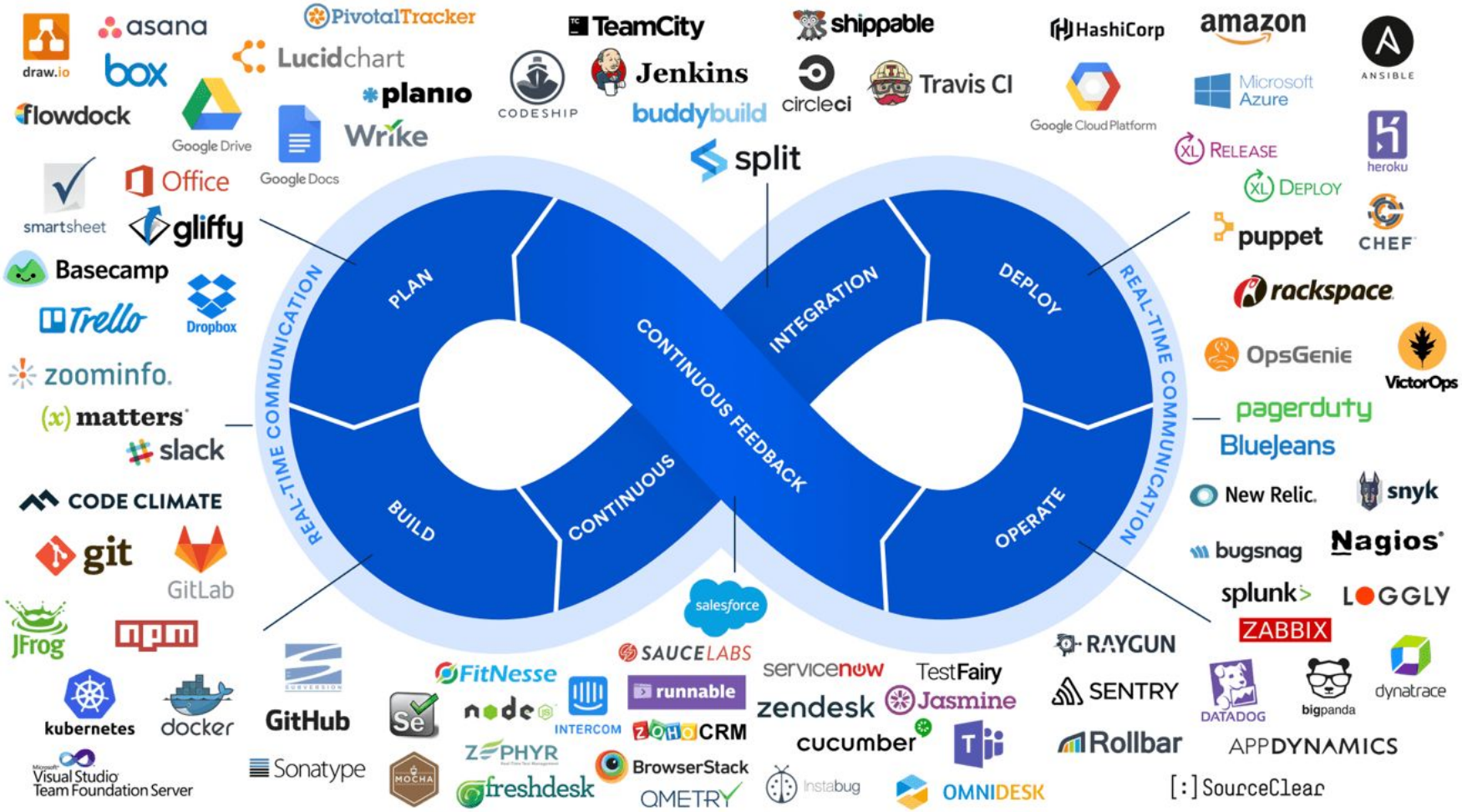
- LXC
- Libcontainer (https://github.com/docker/libcontainer)
- RunC (https://github.com/opencontainers/runc)
- Donated to Open Container Initiative 2015
    - https://www.opencontainers.org/
    - https://github.com/opencontainers/runc

Build, Ship, Run



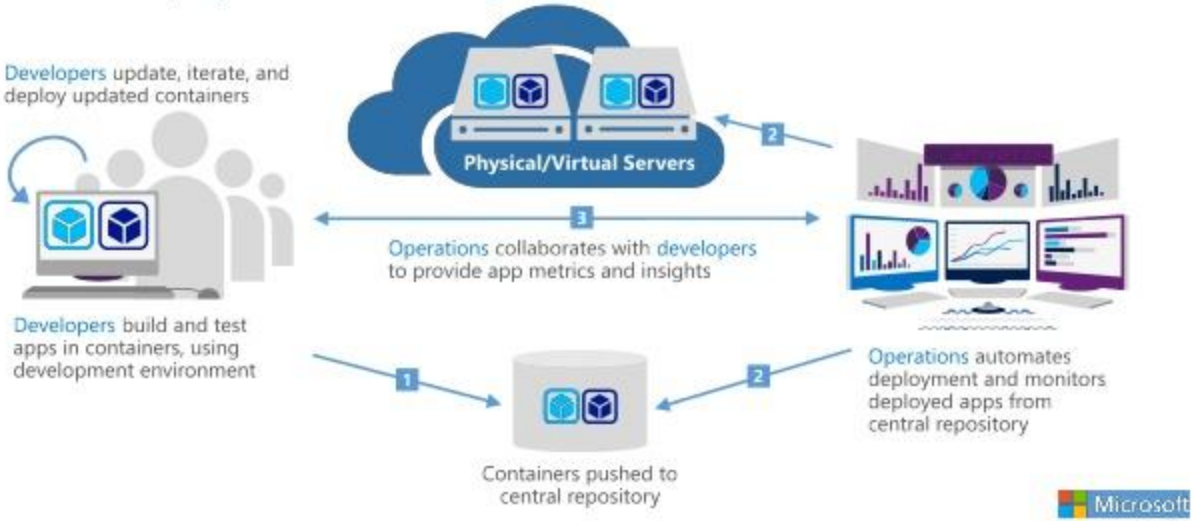
Docker / Containerisation

- Runtime Standardisation
- Scalability
- Dependency Management & Versioning
- Lightweight & Resource Isolated
- Shared FS Layering
- Repeatability of Builds
- Portability
- Cloud Native (CNCF)
- Immutability (Cattle Not Pets)
- Rich Sharing Ecosystem

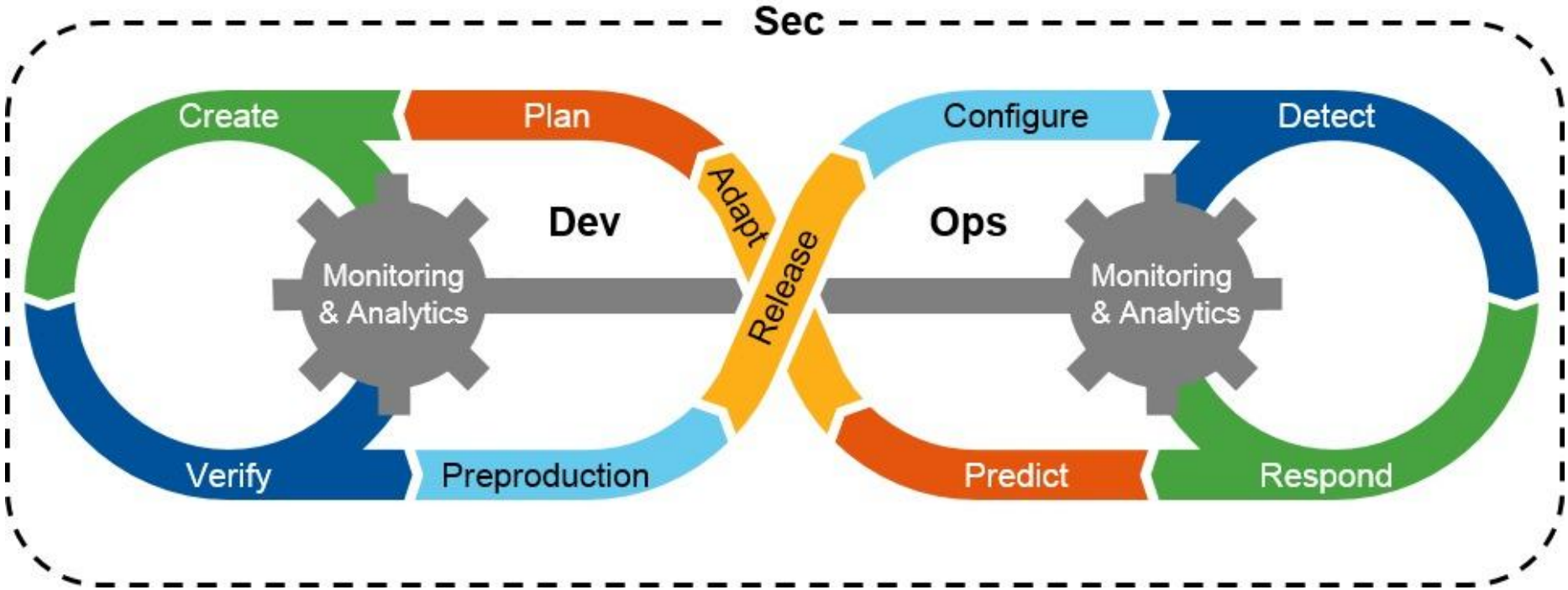

PLAN
CONTINUOUS
BUILD
CONTINUOUS FEEDBACK
INTEGRATION
DEPLOY
OPERATE
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
box
Lucidchart
planio
flowdock
Google Drive
Google Docs
Wrike
Office
smartsheet
gliffy
Basecamp
Trello
Dropbox
zoominfo.
(x) matters
slack
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
GitHub
Sonatype
CODESHIP
TeamCity
Jenkins
buddybuild
shippable
circleci
Travis CI
split
FitNesse
node
INTERCOM
runnable
ZOHO CRM
ZEPHYR
freshdesk
BrowserStack
QMETRY
SAUCELABS
salesforce
servicenow
TestFairy
zendesk
Jasmine
cucumber
Instabug
OMNIDESK
HashiCorp
amazon
Google Cloud Platform
Microsoft Azure
ANSIBLE
RELEASE
heroku
DEPLOY
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
RAYGUN
SENTRY
Rollbar
DATADOG
bigpanda
dynatrace
APPDYNAMICS
[:] SourceClear

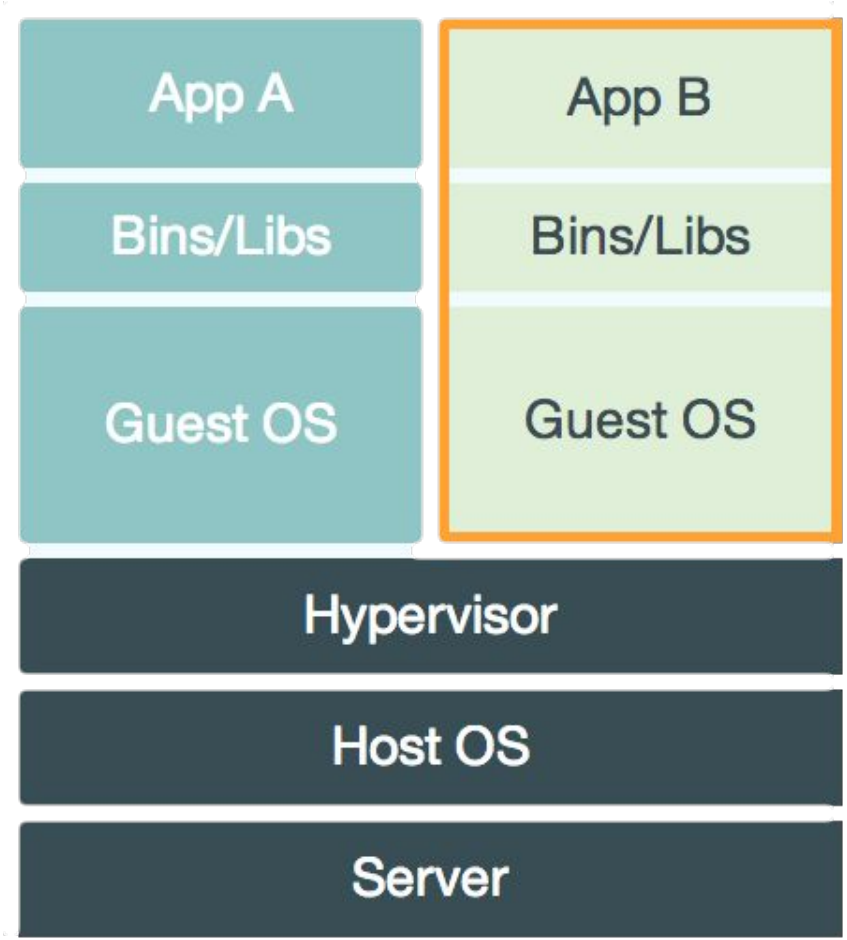
DevOps and Containers

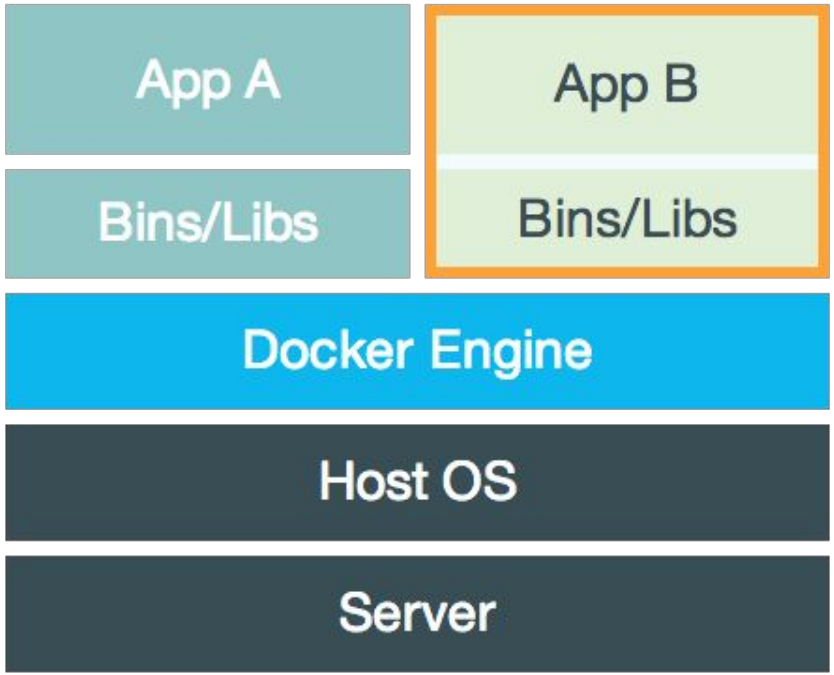
Creation, deployment, and management



Shift Left For Security


Sec
Create
Plan
Configure
Detect
Dev
Adapt
Release
Ops
Monitoring & Analytics
Monitoring & Analytics
Verify
Preproduction
Predict
Respond


App A
App B
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host OS
Server

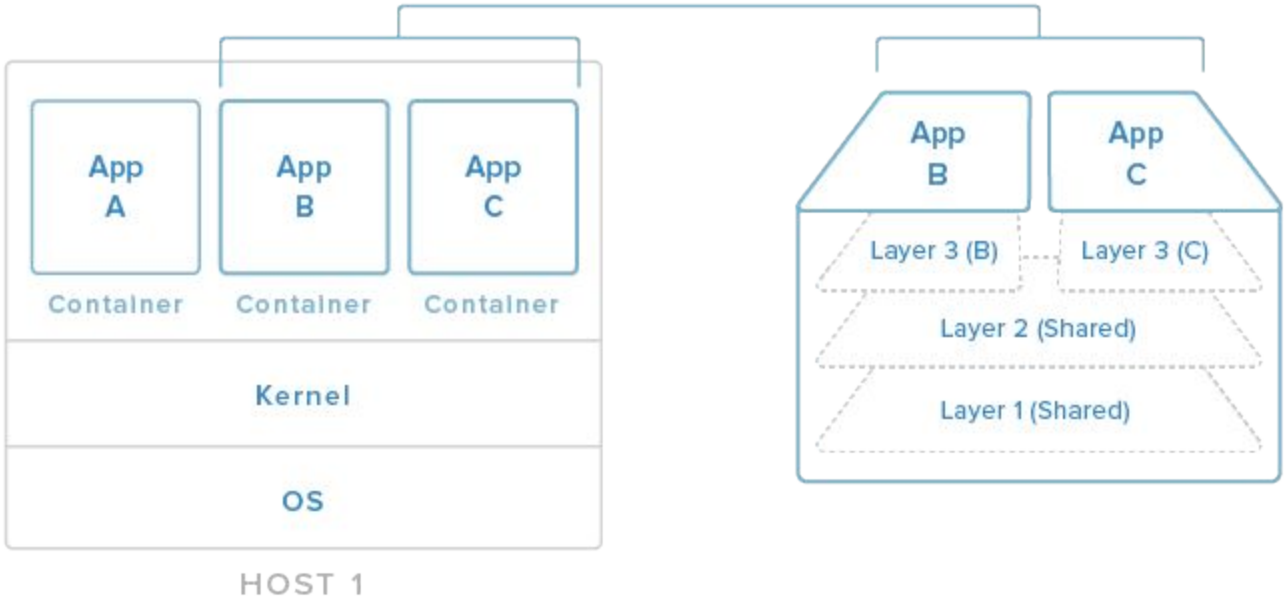

App A
App B
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server


WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

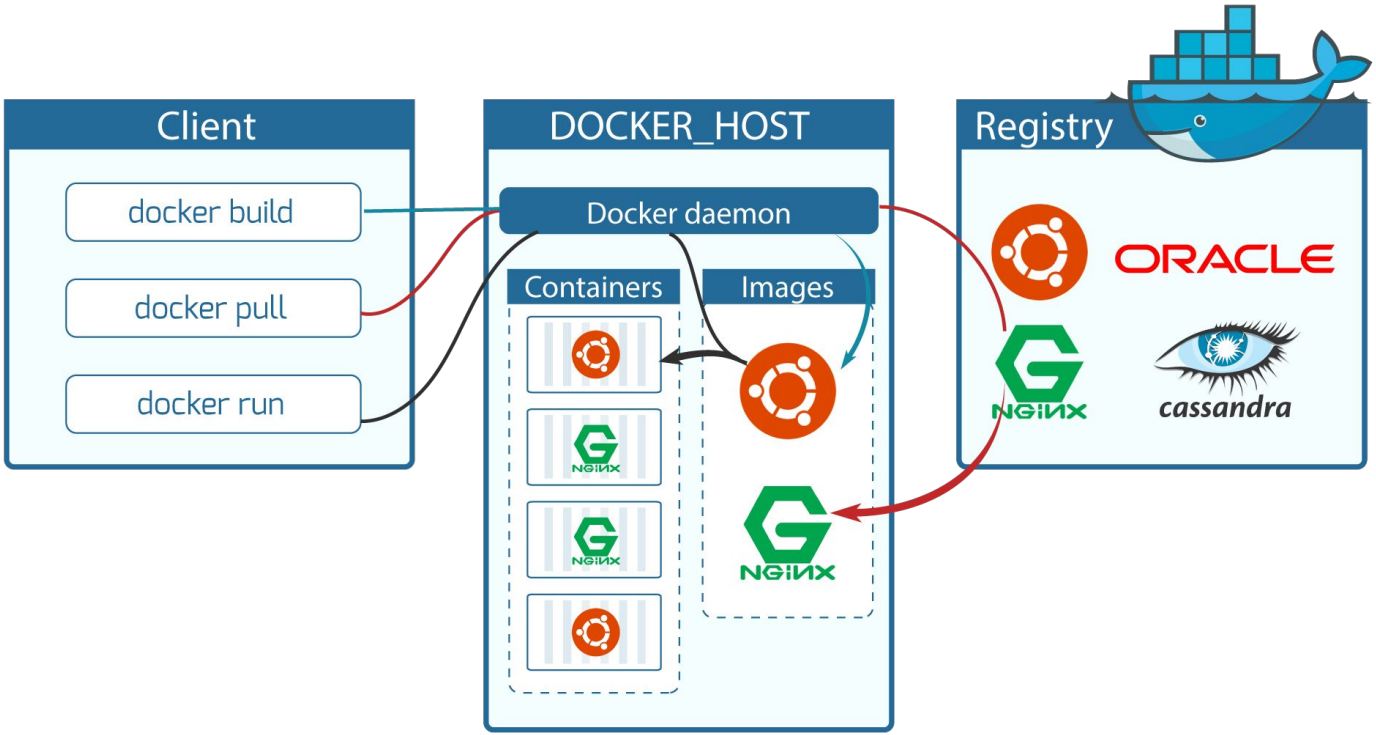
Container Use Cases

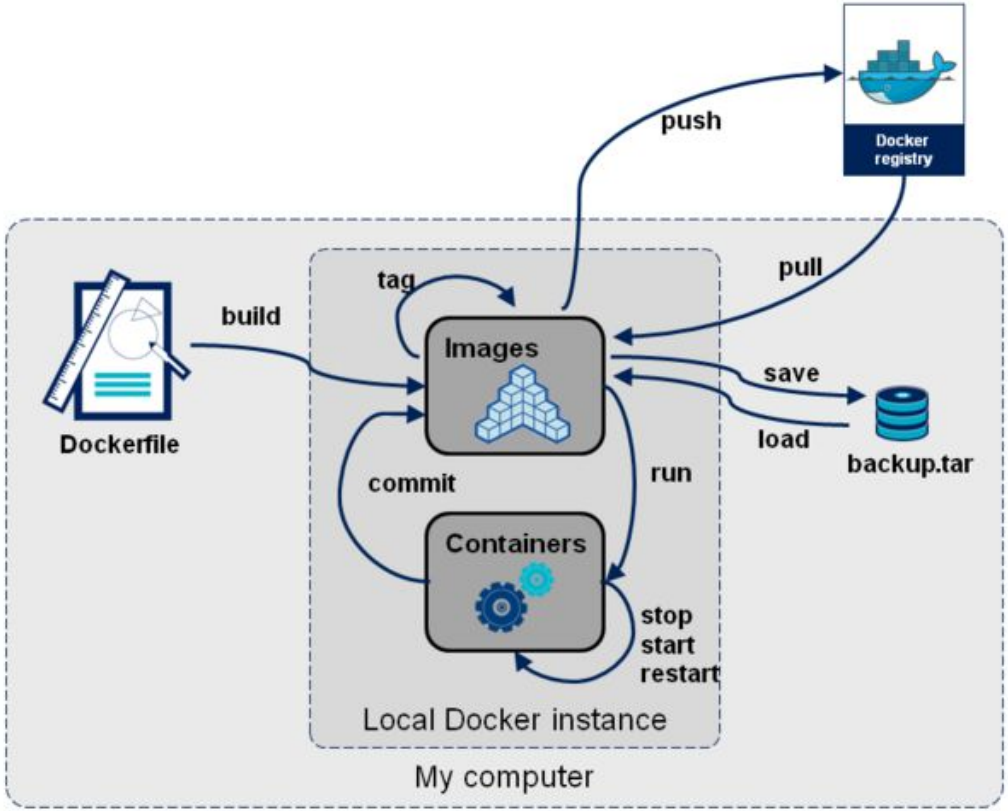
- Applications
- Databases
- Services
- Build Agents
- Test harnesses
- POCs
- X11 services (Jessie Frazelle) https://blog.jessfraz.com/)
- …. It's just a process after all….

CONTAINER OVERVIEW



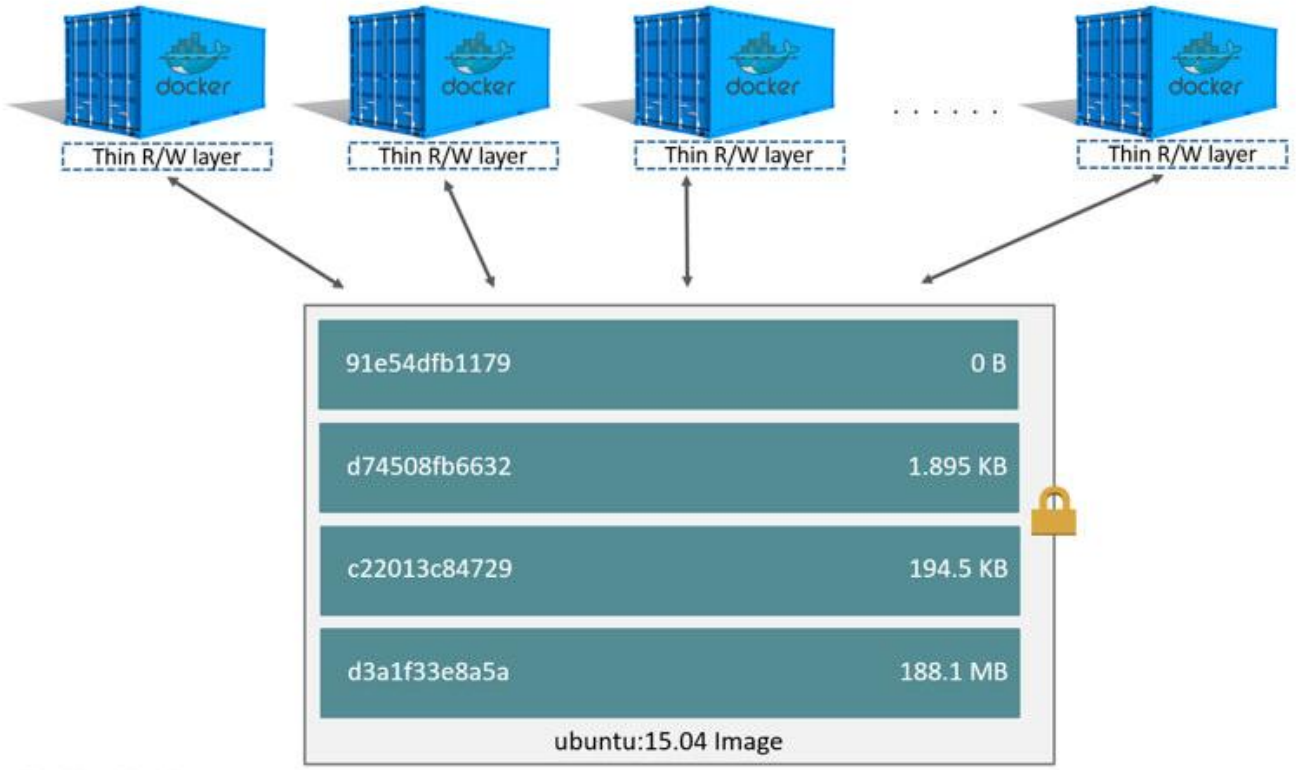
DOCKER COMPONENTS




Docker registry
push
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer

Docker Daemon CLI Communication

- Non networked unix file socket (*/var/run/docker.sock*)
- HTTP / HTTPS (Please configure TLS!)
- SSH tunnel port forward socket.
- *-H* param can be used to configure this



Union Filesystem

Union File System

All layers except the topmost are readonly

Dockerfiles

- Builds layers in the union FS
- SHA-256 digest of the layer result of the directive
- Build cache for repeat build efficiency

Simple Example

```
FROM alpine:3.8

COPY greeting.txt ./

USER nobody

CMD ["cat", "greeting.txt"]
```

Simple Example

```
docker build -t hello-docker-birmingham .
```

```
Sending build context to Docker daemon  3.072kB
Step 1/4 : FROM alpine:3.8
 ---> 11cd0b38bc3c
Step 2/4 : COPY greeting.txt ./
 ---> 6e621934431d
Step 3/4 : USER nobody
 ---> Running in 36343a423445
Removing intermediate container 36343a423445
 ---> 0d89fbf00a6c
Step 4/4 : CMD ["cat", "greeting.txt"]
 ---> Running in 97c4c7d01da4
Removing intermediate container 97c4c7d01da4
 ---> 6913f6e1fd4b
Successfully built 6913f6e1fd4b
```

What's going on?

- The base image is Alpine (a tiny linux distro)
- Copy a file from the build context dir
- As the user nobody (step-down from root)
- Run the cat command with the file

Docker Metadata

- Most constructs have the ability to be ***inspected***.
- For example:
  - `docker image inspect hello-docker-birmingham`
- Use ***jq*** for piping transforms
- Alternatively go formatters which can be CLI provided or in
  - `~/.docker/config.json`

Fire it Up!

```
$ docker run hello-docker-birmingham

Hello Docker Birmingham!%
```

How about a webserver?

```
FROM alpine:3.8

RUN apk add --no-cache npm

RUN npm install http-server -g

WORKDIR /etc/files

COPY greeting.txt ./

USER nobody

CMD ["http-server", "./"]
```

How about a webserver?

```
$ docker build -t http-server .

$ docker history http-server

IMAGE               CREATED             CREATED BY
SIZE                COMMENT
a2ad731ff8ba        4 hours ago         /bin/sh -c #(nop)  CMD ["http-server" "./"]
0B
ceacd9e50527        4 hours ago         /bin/sh -c #(nop)  USER nobody
0B
4f707fe61903        4 hours ago         /bin/sh -c chown nobody:nobody *
12B
af8d819dae00        4 hours ago         /bin/sh -c #(nop) COPY file:2c0ebb879350f3ff…
12B
797657cae401        4 hours ago         /bin/sh -c #(nop) WORKDIR /etc/files
```

How about a webserver?

```
$ docker run -name server -p 127.0.0.1:9090:8080 http-server
Starting up http-server, serving ./
Available on:
  http://127.0.0.1:8080
  http://172.17.0.2:8080
Hit CTRL-C to stop the server
$ curl localhost:9090/greeting.txt
Hello World!%
```

What's going on?

- Webserver process is being started
- The webserver's port 8080 is being made available to localhost ***only***
- When the service terminates the container definition is deleted
- Let's leave it running for now……..

Docker Containers

- Debugging
- Instrumentation
- Resource Constraints
- Labels
- Volumes
- Process Managers
- Logs

Debugging

- `docker exec -it server top`
- You can invoke any app installed in the container FS and you have permission to do so:
  - Install Packages using your ***favourite*** package manager
  - List process `ps -ax`
  - ....
- Will work on the ***writeable*** layer of the container*
- No need for ***sshd!***

Container Stats

- CLI Stats / Endpoint
- Prometheus Endpoint (https://docs.docker.com/config/thirdparty/prometheus/)
- Health

```
$ docker stats
```

Labels

- Can be added to both images and containers
- Consider using the Label Schema (https://label-schema.org/rc1/)
- Immensely useful when running in production!

Resource Constraints

- CPU
- Memory
- DiskIO
- BlkIO

```
docker run --rm -it --cpus=0.1 -m 80m tomcat:9
```

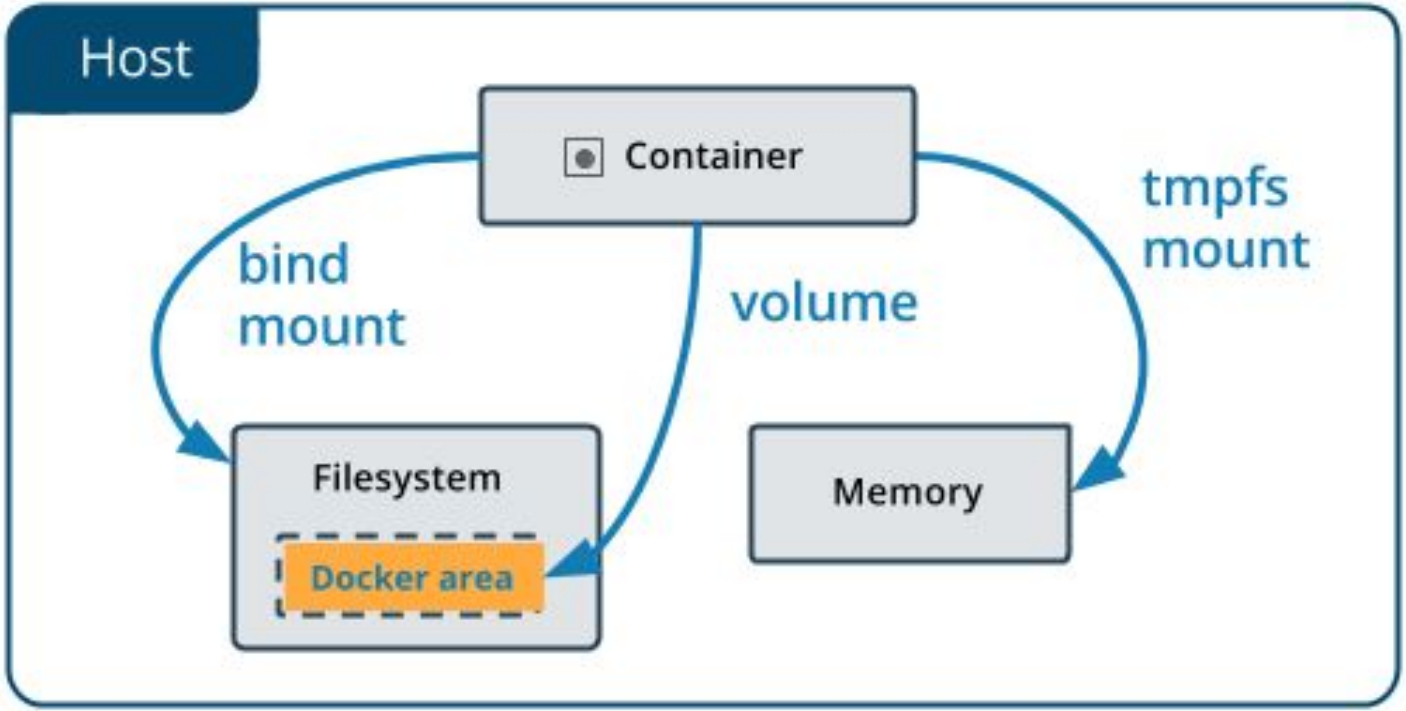
```
docker run --rm -it --cpus=1 -m 10m tomcat:9
```


I do not introduce the log.

Logs

- Stdout / Stderr are logged to the configured driver
- Configurable logging drivers
  - Json-file (default)
  - FluentD
  - Gelf
  - Splunk
  - Awslogs (Cloud Watch)
  - ...
- For json-file you can use
  - `docker logs -f server`
- For other drivers, check the agg. service!

Vo


Host
Container
bind
mount
volume
tmpfs
mount
Filesystem
Docker area
Memory

Volumes

- Physical (Bind)
- Logical (Named)
- Drivers available
  - Local (only visible on host)
  - NFS
  - Azure
  - Portworx
  - REX-Ray
  - … Lots more open source!

Bind Volumes

- Mounts are visible to the writable container layer to the filesystem

```
$ docker run --rm -v $PWD:/etc/files:ro -p 9091:8080 http-server
```

- Watch your permissions when using **:rw**!
- Portability issues between hosts
- Great for build agents!

Named Volumes

- Are managed and created and linked to zero or more containers
- Like virtual USB sticks!
- Persist beyond container restarts
- Managed via pruning

```
$ docker run --rm -v http-server:/etc/files:ro -p 9091:8080
http-server
```

Docker Hub & Publishing

Publishing

- Docker Hub!
- “Official” Repos
- Private Repos, e.g. Nexus, Artifactory
- Image Signing
- Image scanning
- Images must be *tagged* with the repo coordinates

```
$ docker build -t http-server -t mattjtodd/http-server .

$ docker tag http-server mattjtodd/http-server

$ docker login && docker push mattjtodd/http-server
```

Whoaaaa!

Some Best Practices

- Label all the things! (https://label-schema.org/rc1/)
- Take care to understand your process owner
- User a process manager (tini, gosu and --init)
- Don’t bloat; alpine where possible avoid unnecessary layers esp. Subsequent deletes
- Use the image directive cache ***wisely….***
- Be careful when mapping to
  - Bind volume mounts
  - Ports to `0.0.0.0:<port>`
- Whitelist the build context with `.dockerignore`

https://www.katacoda.com/courses/docker/

Get ready for next time……..

Homework!

- Install Docker
- Pull images from the repo for next time
- I'll post which are needed before the next session
- Ask on the Slack channel is you have any issues / questions