Docker Clustering

batteries included, but removable

Jessie Frazelle @frazelledazzell

Qcon London: March 5th, 2015

Outline

- Shortest ever intro to Docker
- Intro to Swarm (Docker clustering)
- Demo of Swarm
- Future of Swarm

What is Docker?

Docker is a runtime for containers.

Whoa what’s a container?

A container is a concept made from linux namespaces, cgroups, & pivot roots.

Outline

- Shortest ever intro to Docker
- Intro to Swarm (Docker clustering)
- Demo of Swarm
- Future of Swarm

Intro to Swarm

Native Clustering for Docker

Serves the standard Docker API

Transparently scale Docker API consumers to multiple hosts batteries included

but removable

Discovery

out-of-the-box

- native discovery

options

- etcd
- consul
- zookeeper

Schedulers

out-of-the-box

- bin-packing (native)

options

- random (native)
- mesos (coming soon, in the works)

How to use Swarm

```
# installing swarm
$ docker pull swarm

# create a cluster
$ docker run --rm swarm create
6856663cdefdec325839a4b7e1de38e8 # <- unique <cluster_id>

# on each of your nodes, start the swarm agent
$ docker run -d swarm join \
    --addr=<node_ip:2375> token://<cluster_id>
```

Minimal Image (small tangent)

A Whopping 7.19 MB

The **Dockerfile** -->

```
FROM scratch

COPY ./swarm /swarm
COPY ./certs/ca-certificates.crt
       /etc/ssl/certs/ca-certificates.crt

ENV SWARM_HOST :2375
EXPOSE 2375

VOLUME /.swarm

ENTRYPOINT ["/swarm"]
CMD ["--help"]
```

How to use Swarm

```
# start the manager on any machine or your laptop
$ docker run -d -p <swarm_port>:2375 \
    swarm manage token://<cluster_id>

# list nodes in your cluster
$ docker run --rm \
    swarm list token://<cluster_id> <node_ip:2375>
```

Using the Docker CLI + Swarm

```
# use the regular docker cli
$ export DOCKER_HOST=tcp://<swarm_ip:swarm_port>
$ docker info
$ docker ps
$ docker logs ...

# manage resources
$ docker run -m 1g
$ docker run -c 1
$ docker run -p 80:80
```

Constraints

```
# standard from docker info
# (storagedriver, executiondriver, kernelversion, operatingsystem)
$ docker run -e constraint:operatingsystem=debian ...
$ docker run -e constraint:storagedriver=btrfs ...

# custom with host labels
$ docker -d --label init=systemd ...
$ docker -d --label init=sysvinit ...
$ docker run -e constraint:init!=systemd ...

$ docker -d --label environment=production ...
$ docker run -e constraint:environment=production ...
```

Affinity

```
# containers
$ docker run -d --name web1 -p 80:80 nginx
$ docker run -d --name stats -e affinity:container==web1 stats

# images
$ docker run -d -e affinity:image==redis redis
$ docker run -d -e affinity:image==nginx nginx
```

Other Filters

```
# ports
$ docker run -d --name web1 -p 80:80 nginx
$ docker run -d --name web2 -p 80:80 nginx
^ defaults on different host

# dependency
$ docker run --volumes-from some-container ...
$ docker run --link some-container:alias ...
$ docker run --net container:some-container ...
```

Outline

- Shortest ever intro to Docker
- Intro to Swarm (Docker clustering)
- Demo of Swarm
- Future of Swarm

The Servers

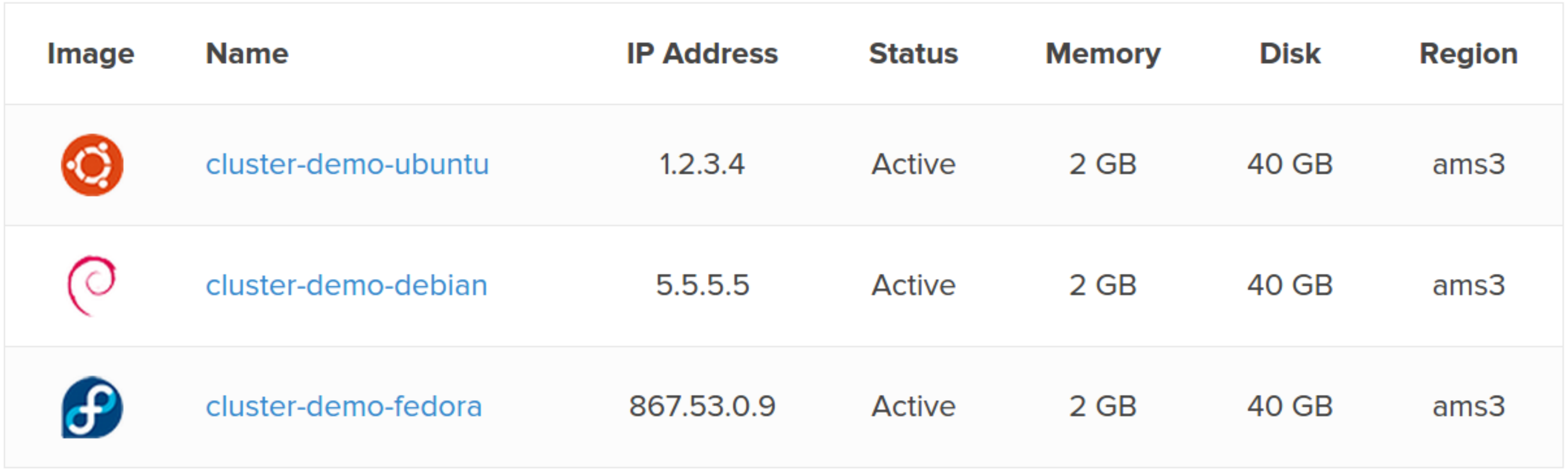

Image	Name	IP Address	Status	Memory	Disk	Region
	cluster-demo-ubuntu	1.2.3.4	Active	2 GB	40 GB	ams3
	cluster-demo-debian	5.5.5.5	Active	2 GB	40 GB	ams3
	cluster-demo-fedora	867.53.0.9	Active	2 GB	40 GB	ams3

Storage Drivers (medium tangent)

Ubuntu Host --> **AUFS**

Fedora Host --> **Device Mapper**

Debian Host --> **Overlay**

AUFS

- First storage driver implemented
- Ubuntu uses it in their default kernel for Live CD

  where root filesystem is COW (copy-on-write) between CD/DVD/USB

Pitfalls: not in mainline kernel

Device Mapper

- Used by RedHat, default to Fedora
- In mainline kernel
- Creates “pools” of blocks

  Each container & each image gets its own block device

- Each time a new block (or a copy-on-write block) is written, a block is allocated from the pool

Device Mapper

Pitfalls: By default, Docker puts data and metadata on a loop device backed by a sparse file

- Which is cool *but* has terrible performance.
- Each time a container writes to a new block...
  - a block has to be allocated from the pool...
  - and when it's written to...
  - a block has to be allocated from the sparse file...
  - and sparse file performance is not the greatest

Overlay

- The hero we all deserve
- In mainline kernel (>=3.18)
- works a lot like AUFS in that it does **not** need its own partition and **works out-of-the-box**

Pitfalls: requires kernel >= 3.18

BTRFS (not used for demo, but important)

- In mainline kernel
- Does copy-on-write at filesystem level

  integrates the snapshot and block pool management features at the filesystem level, instead of the block device level

Pitfalls: have to setup partition

Back to the demo...

Outline

- Shortest ever intro to Docker
- Intro to Swarm (Docker clustering)
- Demo of Swarm
- Future of Swarm

Future of Swarm

- Rescheduling Policies
- More backend drivers, Mesos, etc
- Leader Election (Distributed State)
- Keeping up to date feature-wise with things added to the engine

Fin.

Jessie Frazelle @frazelledazzell

jess@docker.com