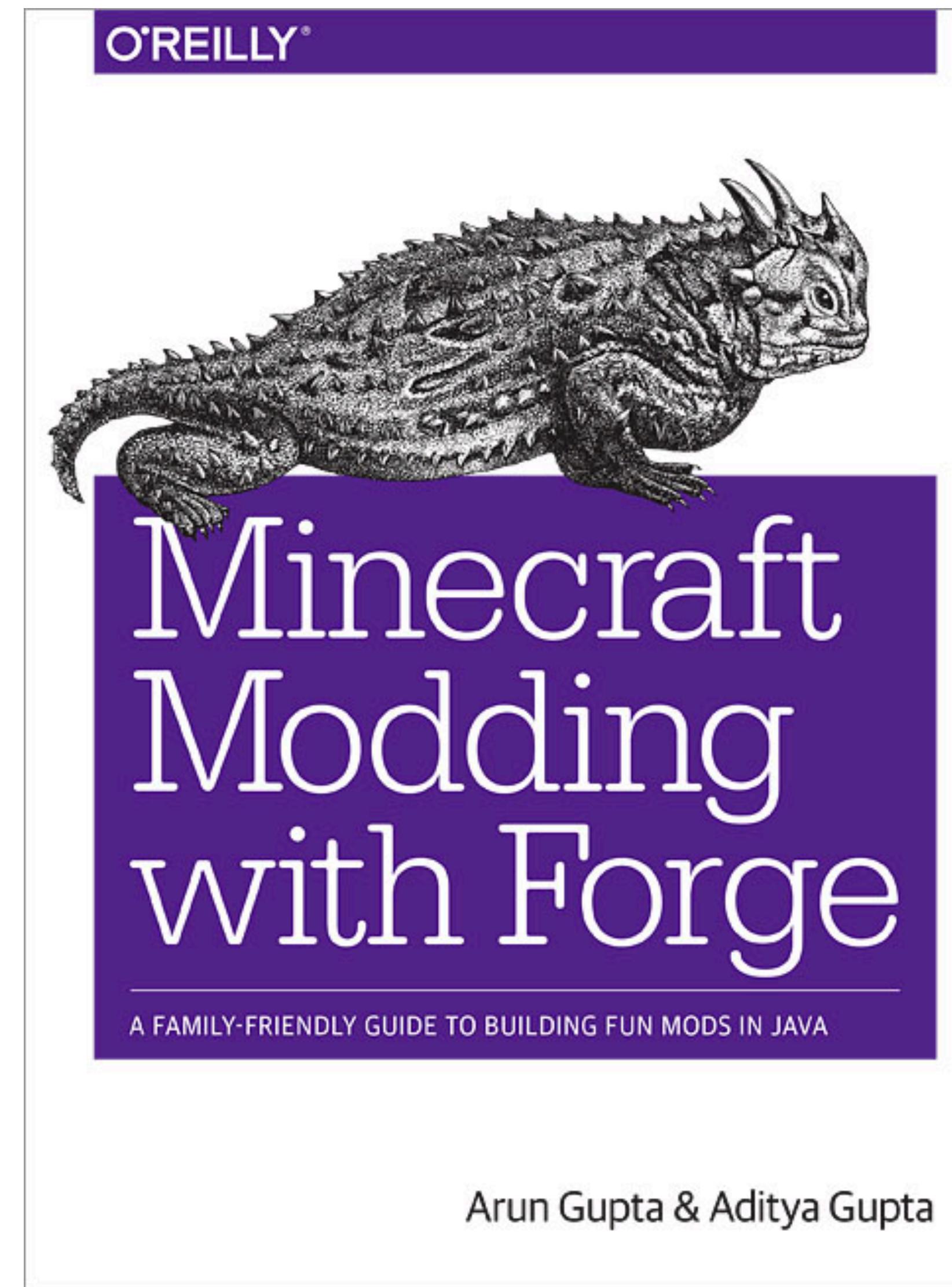


docker

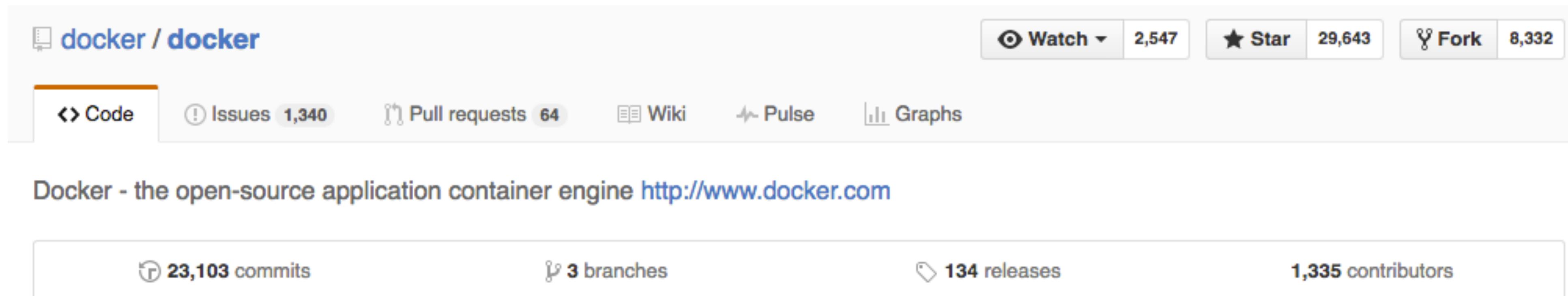
# Docker for Java Developers

Arun Gupta, @arungupta  
VP Developer Advocacy, Couchbase

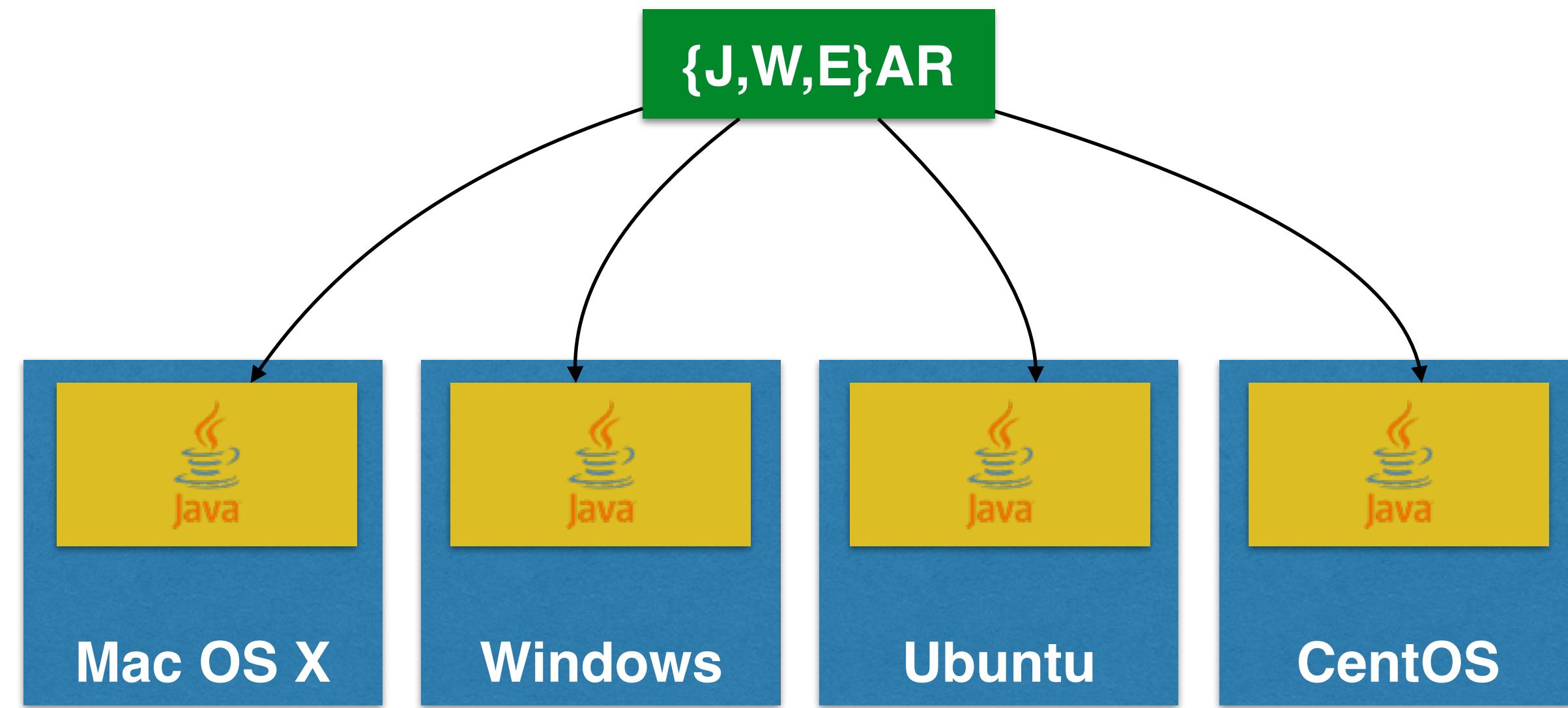


# What is Docker?

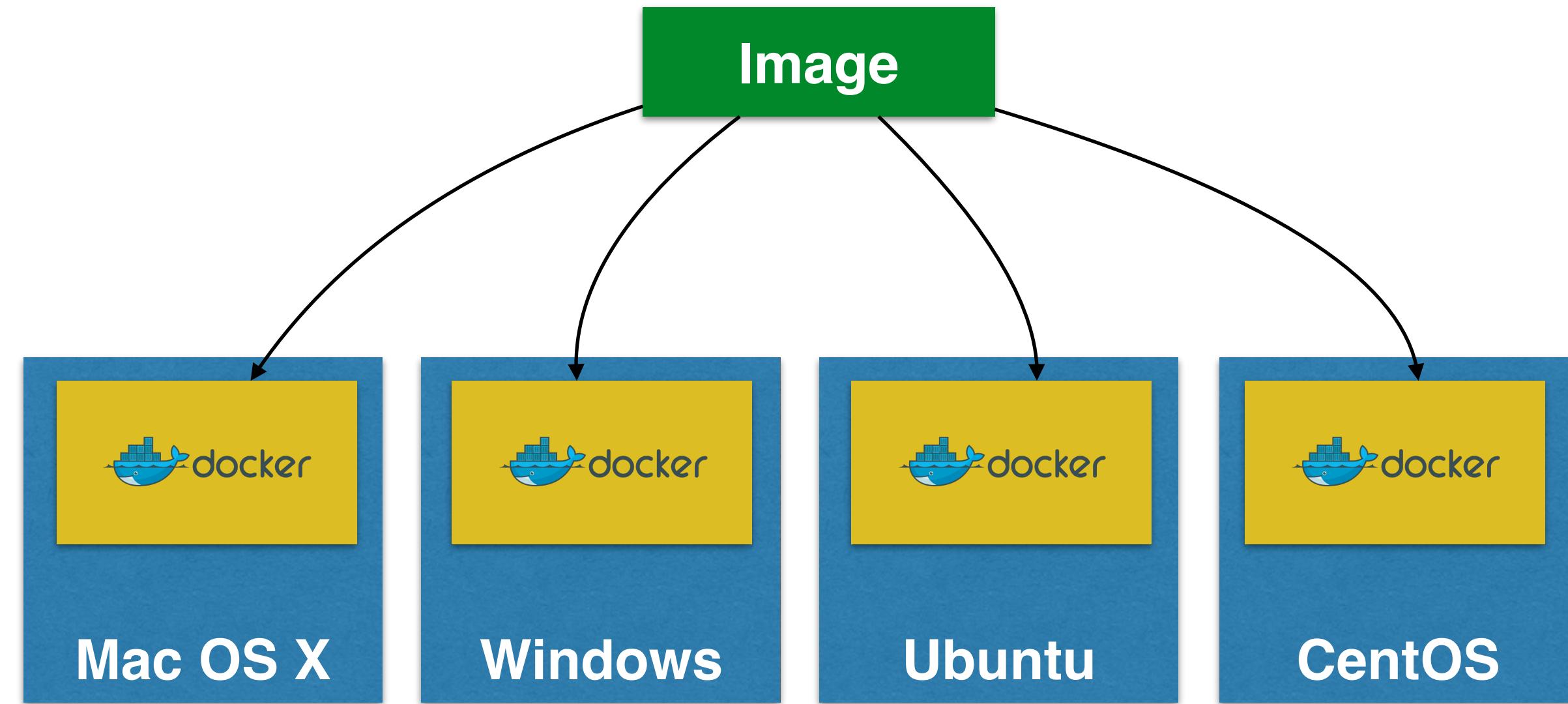
- Open source project and company



- Used to create containers for software applications
- Package Once Deploy Anywhere (PODA)

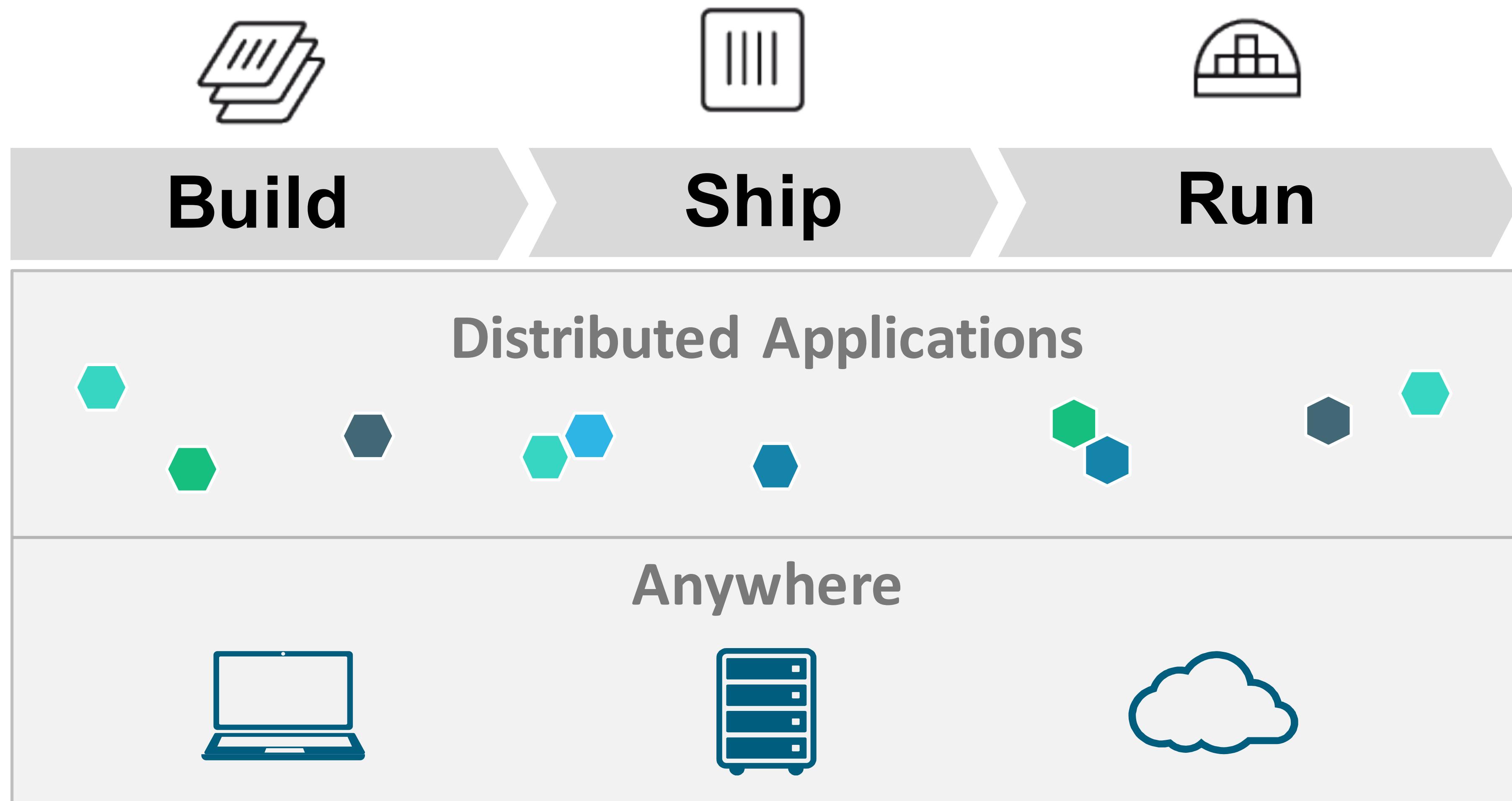


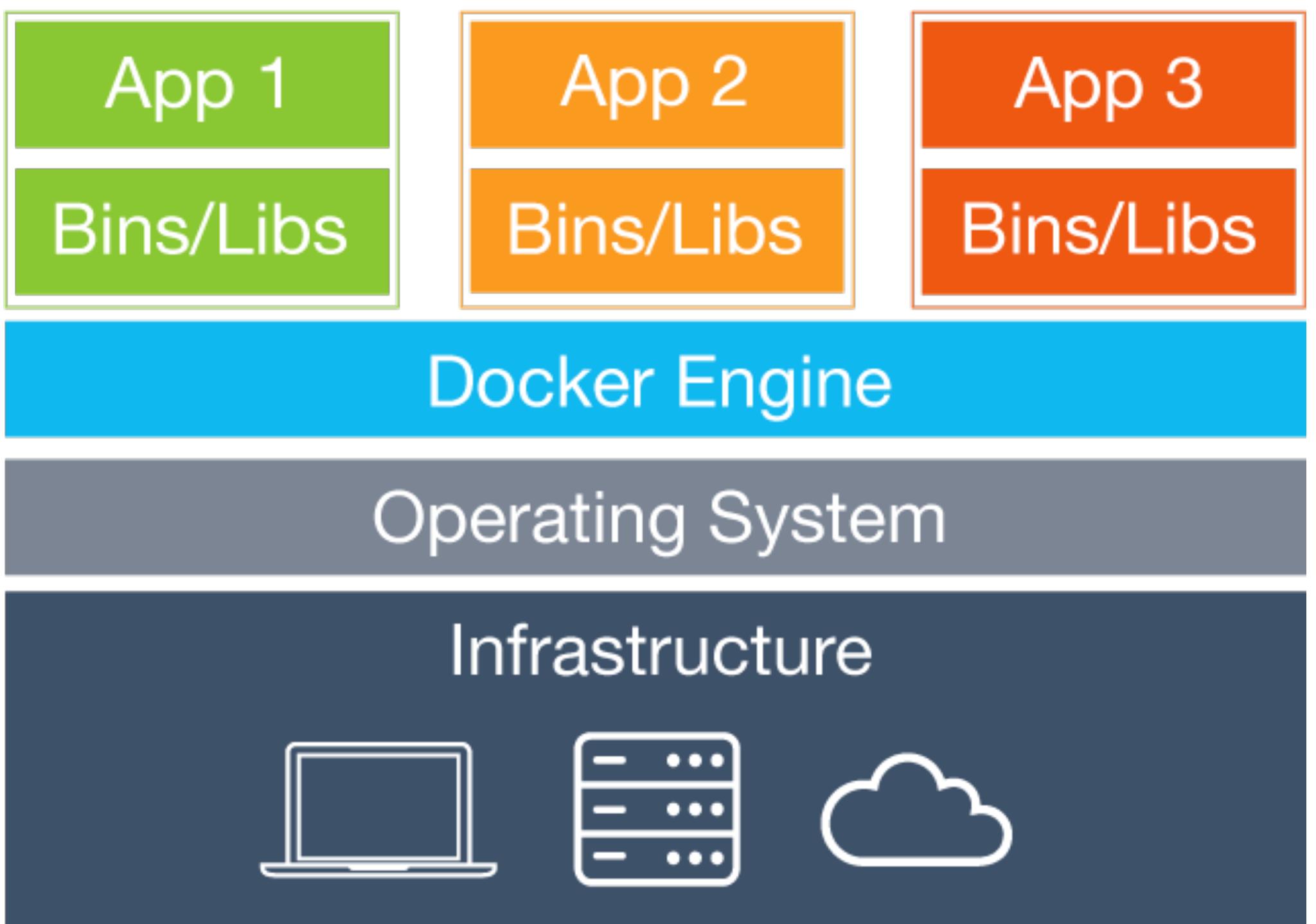
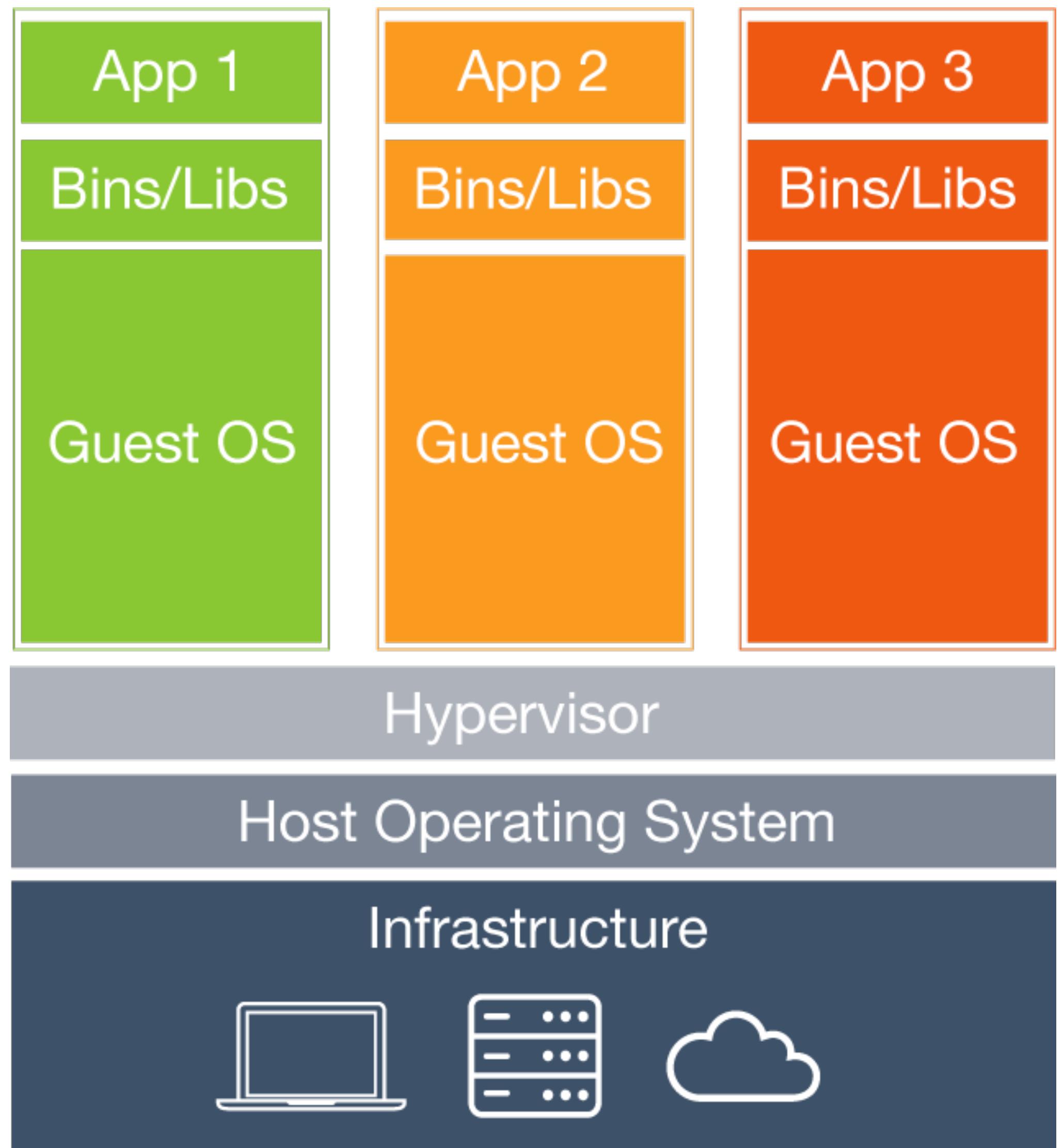
**WORA = Write Once Run Anywhere**



**PODA = Package Once Deploy Anywhere**

# Docker Mission







## Build

Develop an app using Docker containers with  
any language and any toolchain.

```
FROM ubuntu
```

```
CMD echo "Hello world"
```

```
FROM java
```

```
COPY target/hello.jar /usr/src/hello.jar
```

```
CMD java -cp /usr/src/hello.jar org.example.App
```

## Dockerfile reference

Usage

Format

    Environment replacement

    .dockerignore file

FROM

MAINTAINER

RUN

    Known issues (RUN)

CMD

LABEL

EXPOSE

ENV

ADD

COPY

ENTRYPOINT

    Exec form ENTRYPOINT example

    Shell form ENTRYPOINT example

VOLUME

USER

WORKDIR

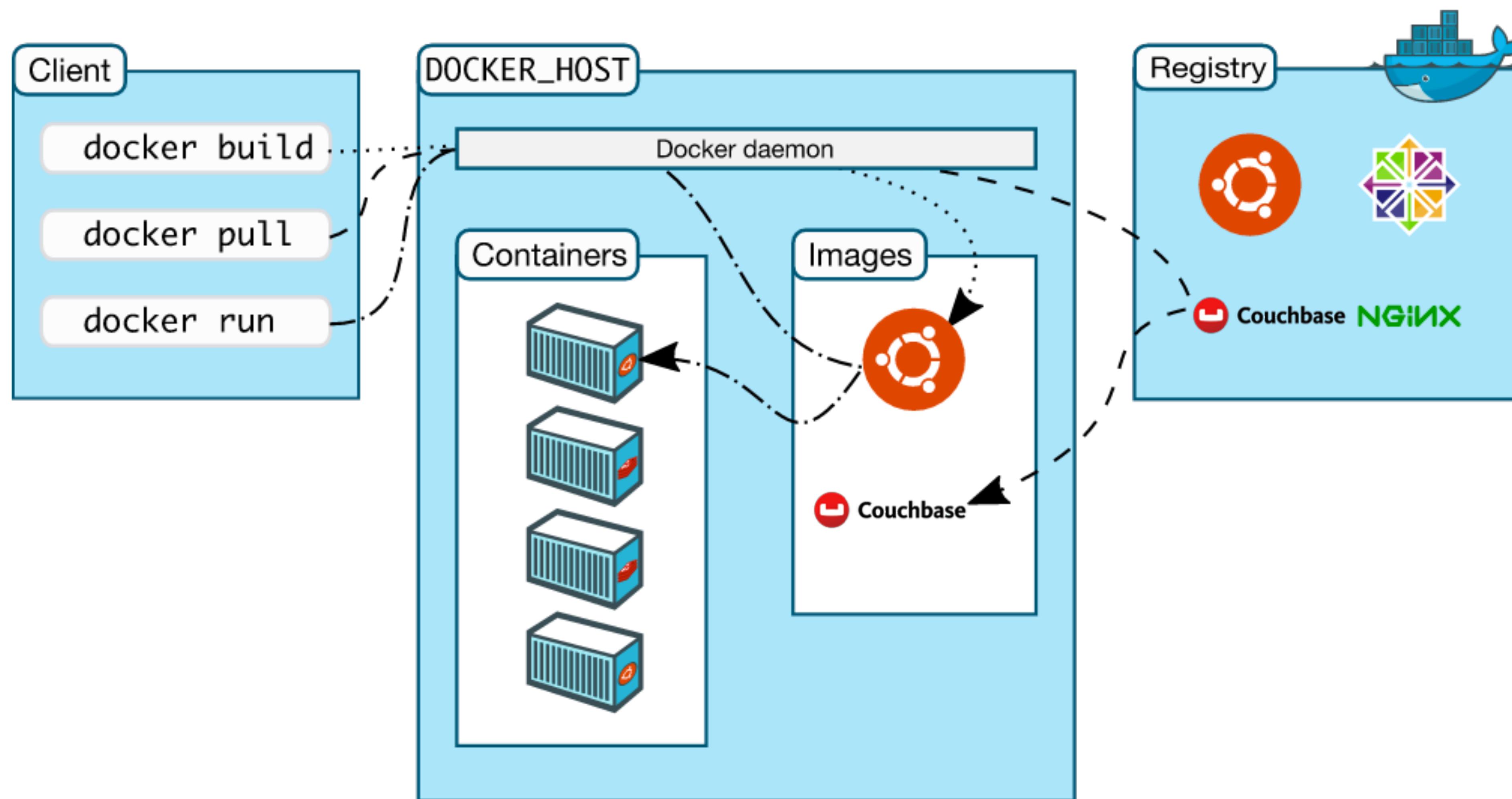
ARG

ONBUILD

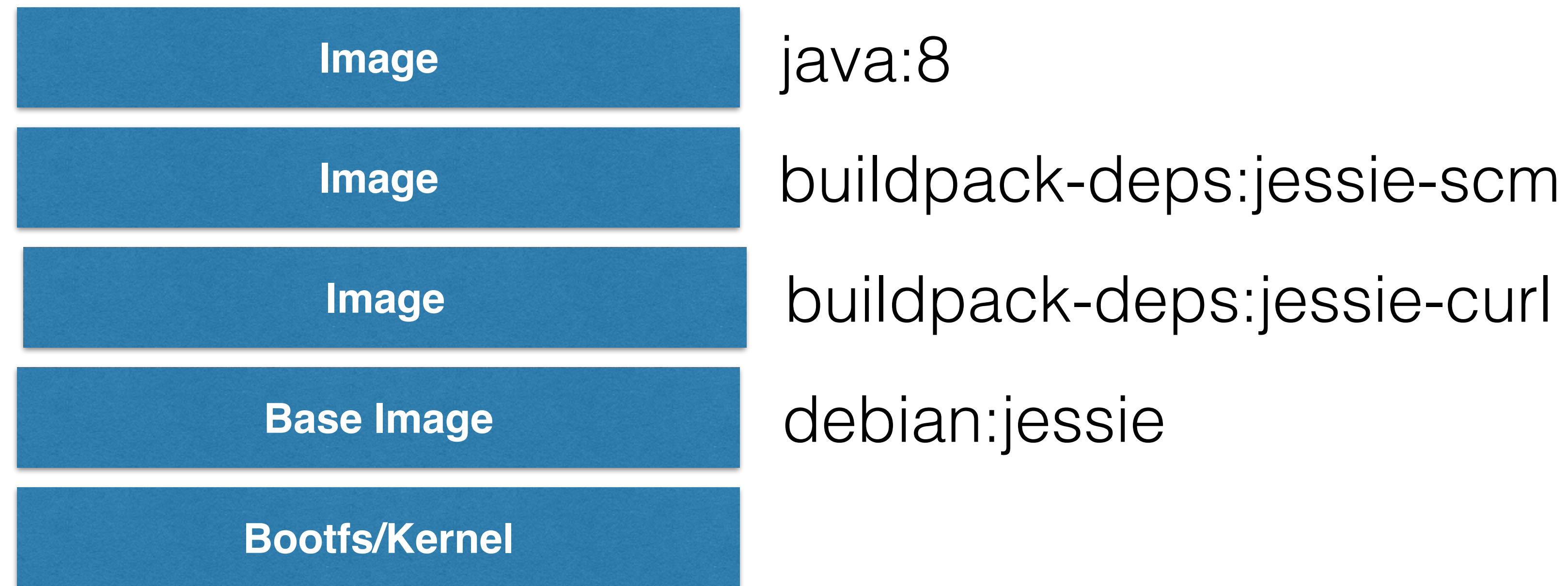
STOPSIG

Dockerfile examples

# Docker Workflow



# Union File System



# Image Layers - Couchbase

```
~ > docker images couchbase
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
couchbase	latest	45abdd57689a	3 weeks ago	372 MB

```
~ > docker history couchbase
```

IMAGE	CREATED	CREATED BY	SIZE
45abdd57689a	3 weeks ago	/bin/sh -c #(nop) VOLUME [/opt/couchbase/var]	0 B
dd8c5611343d	3 weeks ago	/bin/sh -c #(nop) EXPOSE 11207/tcp 11210/tcp	0 B
30852bbad62b	3 weeks ago	/bin/sh -c #(nop) CMD ["couchbase-server"]	0 B
5537747ea12f	3 weeks ago	/bin/sh -c #(nop) ENTRYPOINT &{ ["/entrypoint.	0 B
e8a83a5448df	3 weeks ago	/bin/sh -c #(nop) COPY file:cbb44c9c65b64a9dc	182 B
18165b90fef9	3 weeks ago	/bin/sh -c #(nop) COPY file:34e32c52f0895191f	389 B
5f37b8bdc5a6	3 weeks ago	/bin/sh -c wget -N \$CB_RELEASE_URL/\$CB_VERSI0	212.1 MB
1a8da511d01b	3 weeks ago	/bin/sh -c groupadd -g 1000 couchbase && user	328.7 kB
d9b2222c39b4	3 weeks ago	/bin/sh -c #(nop) ENV CB_VERSION=4.0.0 CB_REL	0 B
815f08b3c781	3 weeks ago	/bin/sh -c apt-get update && apt-get inst	23.57 MB
fc38f156c0ea	3 weeks ago	/bin/sh -c #(nop) MAINTAINER Couchbase Docker	0 B
2a7a952931ec	3 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0 B
10f1b5844a9c	3 weeks ago	/bin/sh -c sed -i 's/^#\s*/(deb.*universe\)\$/'	1.911 kB
23c388b926b6	3 weeks ago	/bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic	156.2 kB
b45376f323f5	3 weeks ago	/bin/sh -c #(nop) ADD file:4a9e089e81d6581a54	135.9 MB

# Image Layers - Java

```
~ > docker images java
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
java	openjdk-8-jdk-alpine	78afabc37d4f	6 days ago	145.5 MB
java	8	f298aed75633	6 days ago	642.9 MB
java	latest	f298aed75633	6 days ago	642.9 MB

```
~ > docker history java
```

IMAGE	CREATED	CREATED BY	SIZE
COMMENT			
f298aed75633	6 days ago	/bin/sh -c /var/lib/dpkg/info/ca-certificates	418.2 kB
<missing>	6 days ago	/bin/sh -c set -x && apt-get update && apt-	349.2 MB
<missing>	6 days ago	/bin/sh -c #(nop) ENV CA_CERTIFICATES_JAVA_VE	0 B
<missing>	6 days ago	/bin/sh -c #(nop) ENV JAVA_DEBIAN_VERSION=8u7	0 B
<missing>	6 days ago	/bin/sh -c #(nop) ENV JAVA_VERSION=8u72	0 B
<missing>	6 days ago	/bin/sh -c #(nop) ENV JAVA_HOME=/usr/lib/jvm/	0 B
<missing>	6 days ago	/bin/sh -c { echo '#!/bin/sh'; echo 'set	87 B
<missing>	6 days ago	/bin/sh -c #(nop) ENV LANG=C.UTF-8	0 B
<missing>	6 days ago	/bin/sh -c echo 'deb http://httpredir.debian.	61 B
<missing>	6 days ago	/bin/sh -c apt-get update && apt-get install	1.289 MB
<missing>	2 weeks ago	/bin/sh -c apt-get update && apt-get install	122.6 MB
<missing>	2 weeks ago	/bin/sh -c apt-get update && apt-get install	44.32 MB
<missing>	2 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0 B
<missing>	2 weeks ago	/bin/sh -c #(nop) ADD file:b5391cb13172fb513d	125.1 MB



# Docker for Mac/Window

- Native application and UI
- Auto update capability
- No additional software required, e.g. VirtualBox
  - OSX: xhyve VM using Hypervisor.framework
  - Windows: Hyper-V VM
- Better networking and filesystem mounting/notification
- Public Beta: [docker.com/getdocker](https://docker.com/getdocker)
- Requires Yosemite 10.10+ or Windows 10 64-bit

# Docker for AWS

- Amazon CloudFormation template to start a Swarm of Docker Engines
- Integrated with Autoscaling, ELB, and EBS
- Only a browser and AWS account
- Sign up at [beta.docker.com](http://beta.docker.com) (restricted availability)
- Show demo?



AWS ▾

Services ▾

Edit ▾

arun.gupta@couchbase.com @ ... ▾

N. California ▾

Support ▾

# Create stack

[Select Template](#)[Review](#)[Specify Details](#)[Options](#)[Template](#)[Review](#)**Template URL**[REDACTED]  
https://s3.amazonaws.com/d1.12.0-rc3-beta1/docker\_for\_aws.json**Description** Docker for AWS Beta 1 (docker 1.12.0-rc3)**Estimate cost**[Cost](#)[Details](#)**Stack name** Docker4AWS**Swarm Size****ManagerSize** 1**ClusterSize** 3**Swarm Properties****ManagerInstanceType** t2.micro**InstanceType** m3.medium**KeyName** arun@couchbase**Create IAM resources** Yes

# Docker for Azure

- Same as Docker for AWS, but for Azure
- Integrated with VM Scale Sets for autoscaling, Azure Load Balancer, Azure Storage



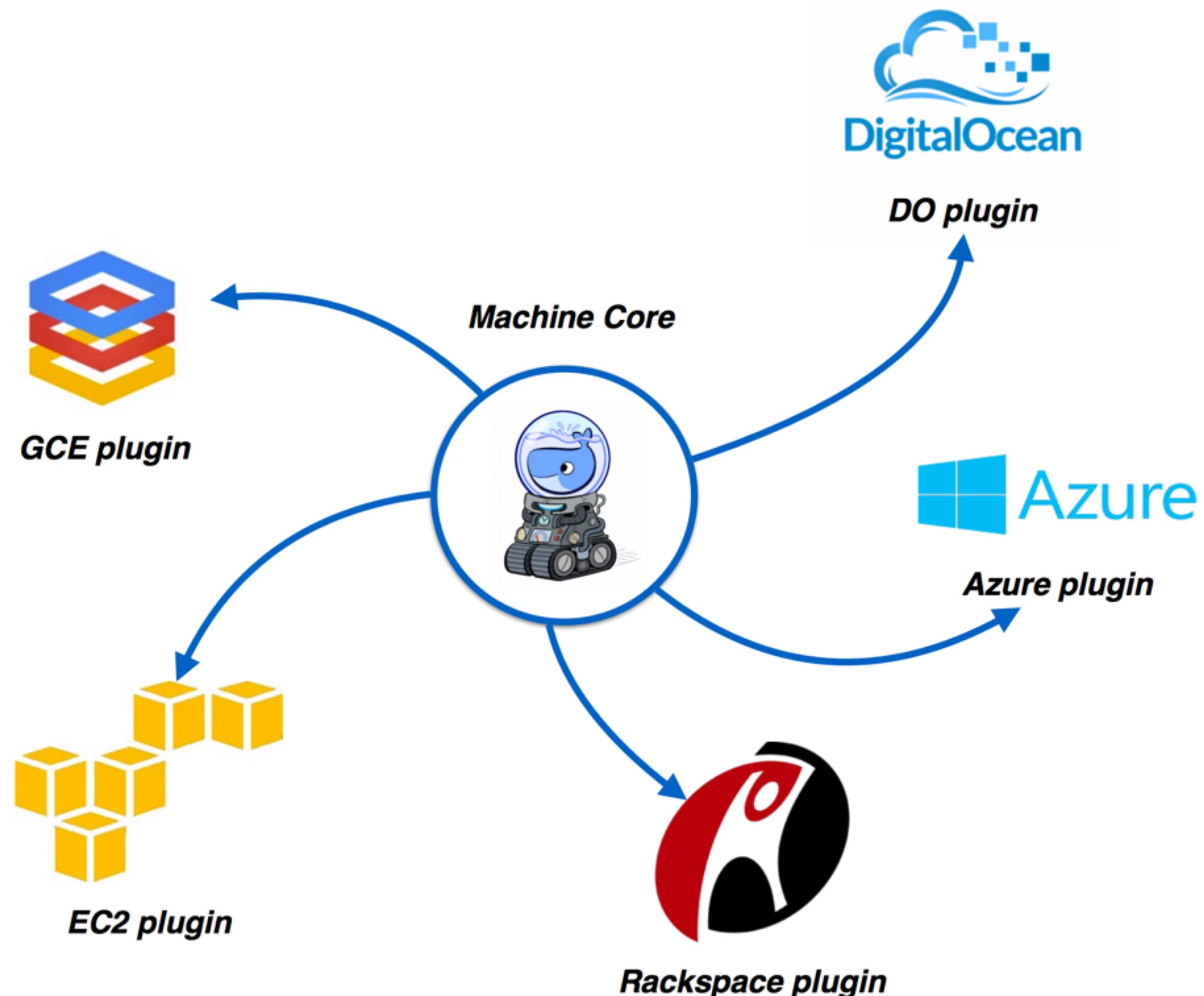
# Docker Machine

- Pre-Docker 1.12
- Create Docker Host on computer or cloud provider

```
docker-machine create --driver=virtualbox myhost
```

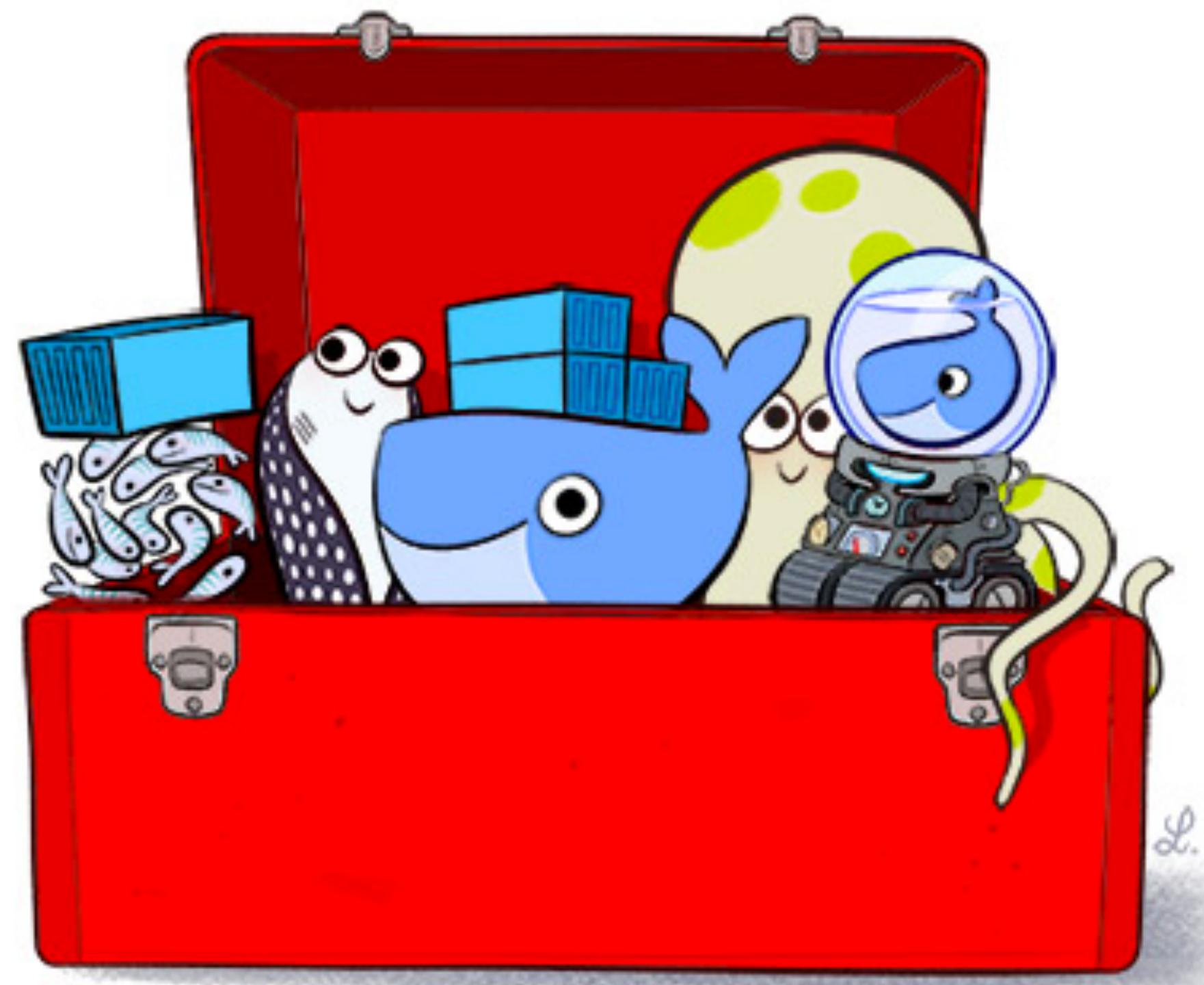
- Configure Docker client to talk to host
- Create and pull images
- Start, stop, restart containers
- Upgrade Docker

# Docker Machine Providers



# Docker Toolbox

- Docker Client
- Docker Machine
- Docker Compose
- Docker Kitematic
- Boot2Docker ISO
- Virtualbox





# Docker Compose

- Defining and running multi-container applications
- Configuration defined in one or more files
  - `docker-compose.yml` (default)
  - `docker-compose.override.yml` (default)
  - Multiple files specified using `-f`
  - All paths relative to base configuration file
- Great for dev, staging, and CI

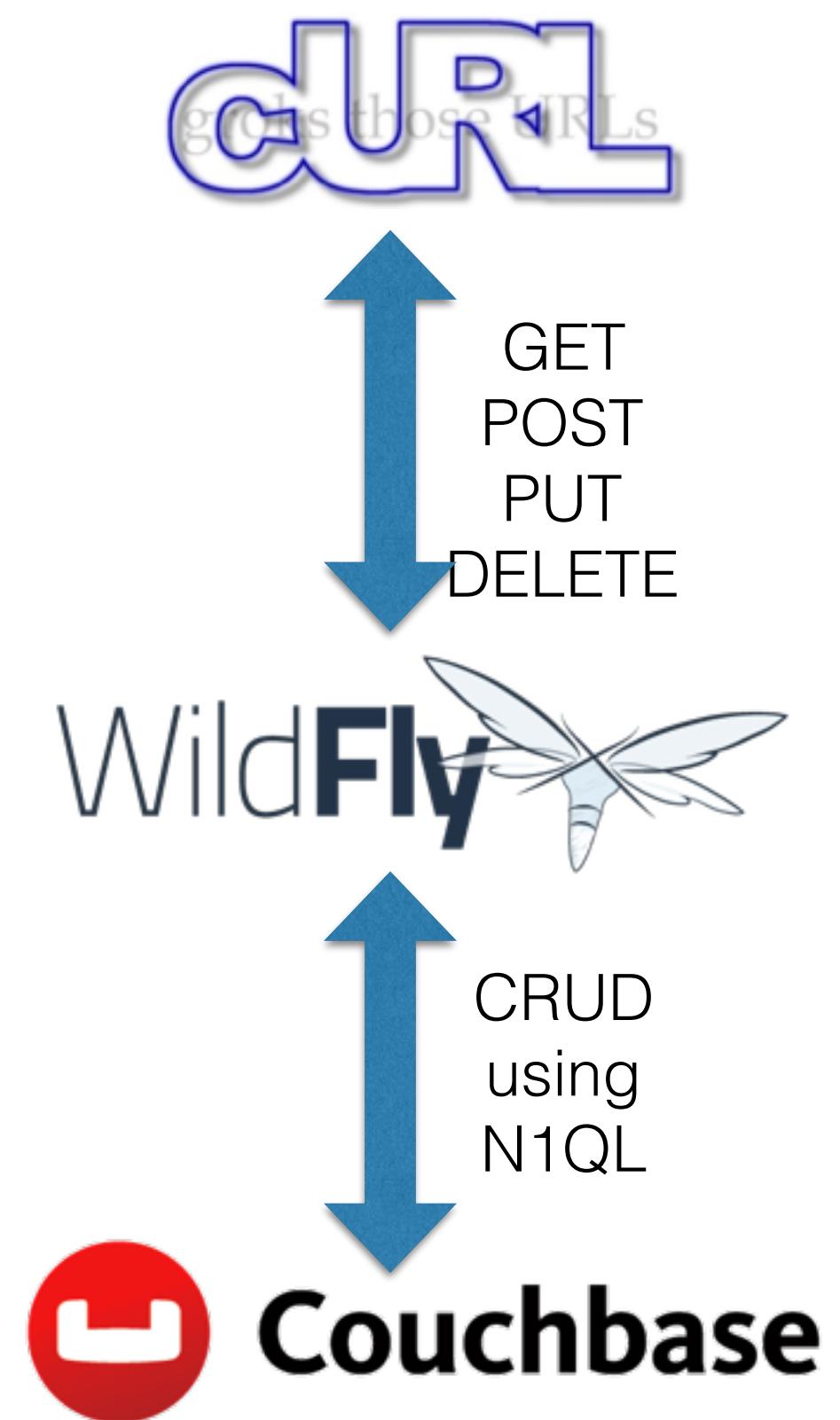


# Docker Compose - One Service

```
version: "2"
services:
  db:
    image: couchbase
    volumes:
      - ~/couchbase:/opt/couchbase/var
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
```



# Docker Compose - Two Services



# Docker Compose - Two Services

```
version: "2"
services:
  db:
    image: couchbase
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
  web:
    image: arungupta/wildfly
    environment:
      - COUCHBASE_URI=db
    ports:
      - 8080:8080
      - 9990:9990
```



# Overriding Services in Docker Compose

```
web:  
  image: jboss/wildfly  
  ports:  
    - 8080:8080
```

docker-compose.yml

```
web:  
  ports:  
    - 9080:8080
```

docker-compose.override.yml

docker-compose up -d

# Dev/Prod with Compose

```
db-dev:  
  image: arungupta/couchbase  
  ports:  
    - . . .  
  
web:  
  image: arungupta/wildfly  
  environment:  
    - COUCHBASE_URI=db-dev:8093  
  ports:  
    - 8080:8080
```

docker-compose.yml

docker-compose up -d

```
web:  
  environment:  
    - COUCHBASE_URI=db-prod:8093  
  ports:  
    - 8080:80  
  
db-prod:  
  image: . . .
```

production.yml

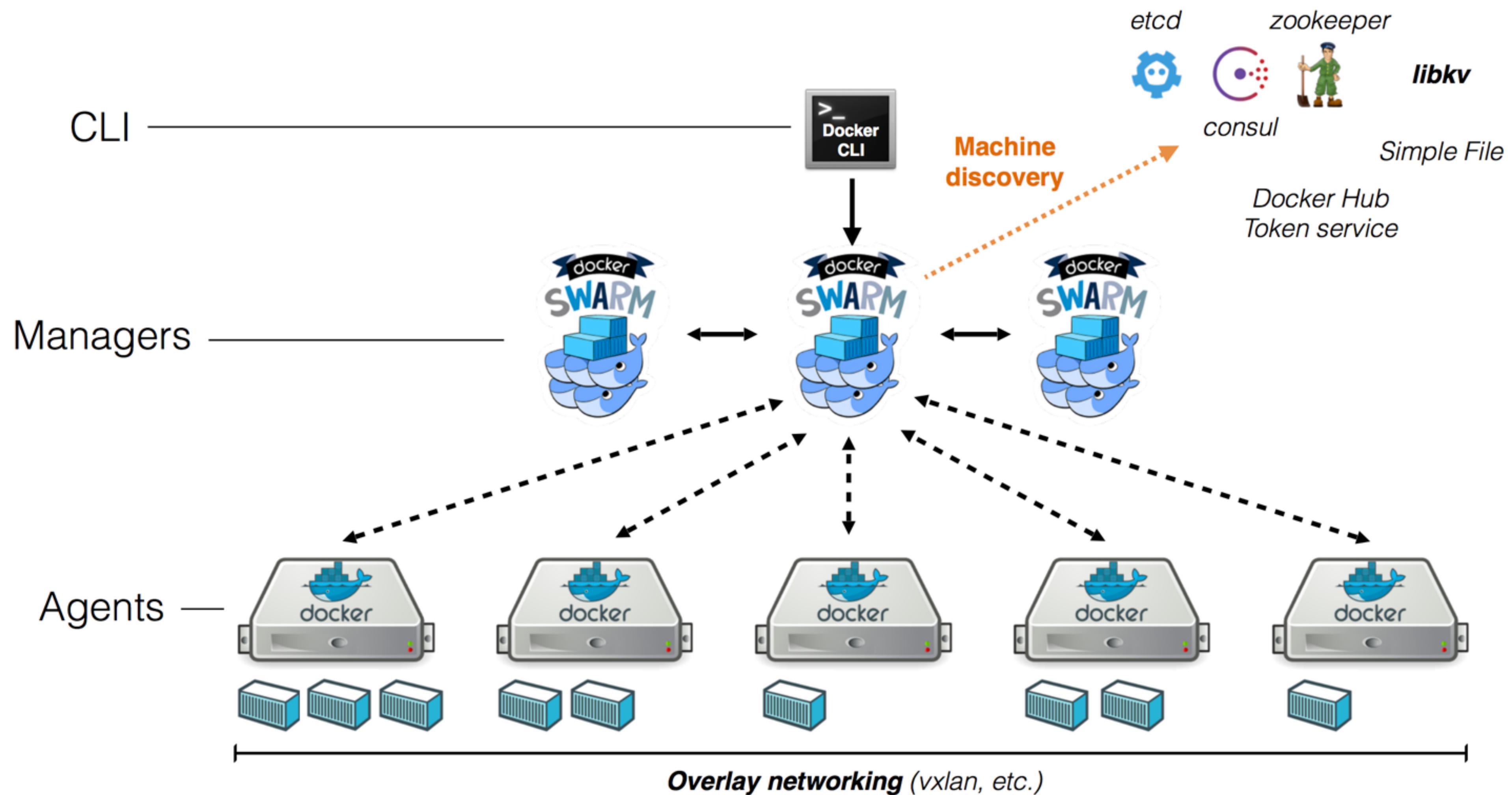
docker-compose up  
-f docker-compose.yml  
-f production.yml  
-d



# Docker Swarm

- Native clustering for Docker
- Provides a unified interface to a pool of Docker hosts
- Fully integrated with Machine and Compose
- Serves the standard Docker API
- 1.2.4 - Ready for production
  - Reschedule containers when a node fails
  - Better node management

Stress tested on 1000 EC2 nodes, ~30k containers





# Master High Availability

- Handle the failover of a manager instance
- **Primary** manager, multiple **replica** instances
- Requests to replica are proxied to primary
- `docker run swarm manage`
  - `--replication`: Enable Swarm manager replication
  - `--replication-ttl "30s"`: Leader lock release time on failure
  - `--advertise`, `--addr`: Address of the swarm manager joining the cluster



# Scheduling Backends

- Based on CPU (**-c**), RAM (**-m**), number of containers
- **docker machine create --strategy <value>**
  - **spread** (default): node with least number of running containers
  - **binpack**: node with most number of running containers
  - **random**: mostly for debugging
- API for pluggable backends (e.g. Mesos) coming

# Limiting CPU resources

```
docker run --help | grep cpu  
--cpu-shares=0  
--cpu-period=0  
--cpu-quota=0  
--cpuset-cpus=  
--cpuset-mems=
```

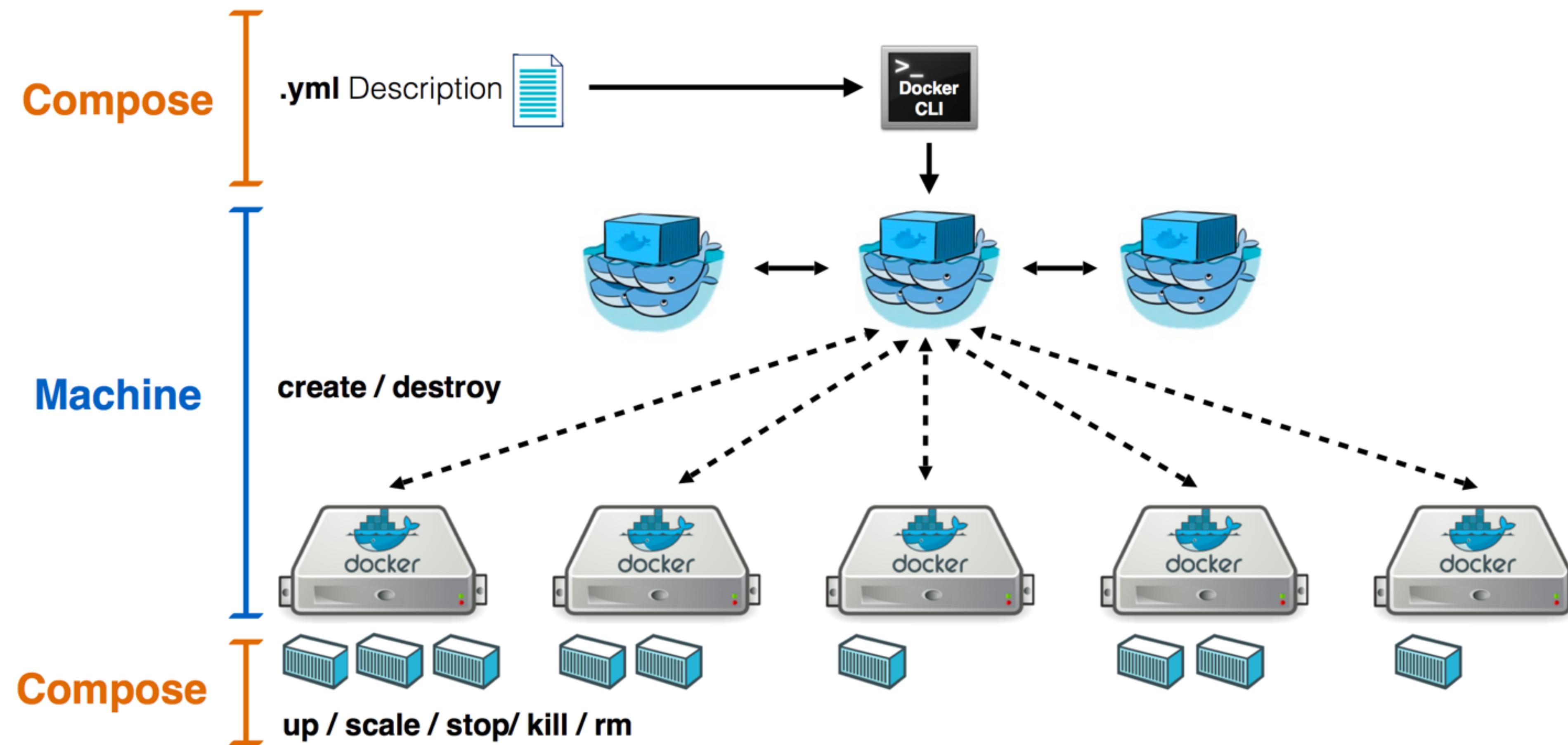
CPU shares (relative weight)  
Limit CPU CFS (Completely Fair Scheduler) period  
Limit CPU CFS (Completely Fair Scheduler) quota  
CPUs in which to allow execution (0-3, 0,1)  
MEMs in which to allow execution (0-3, 0,1)

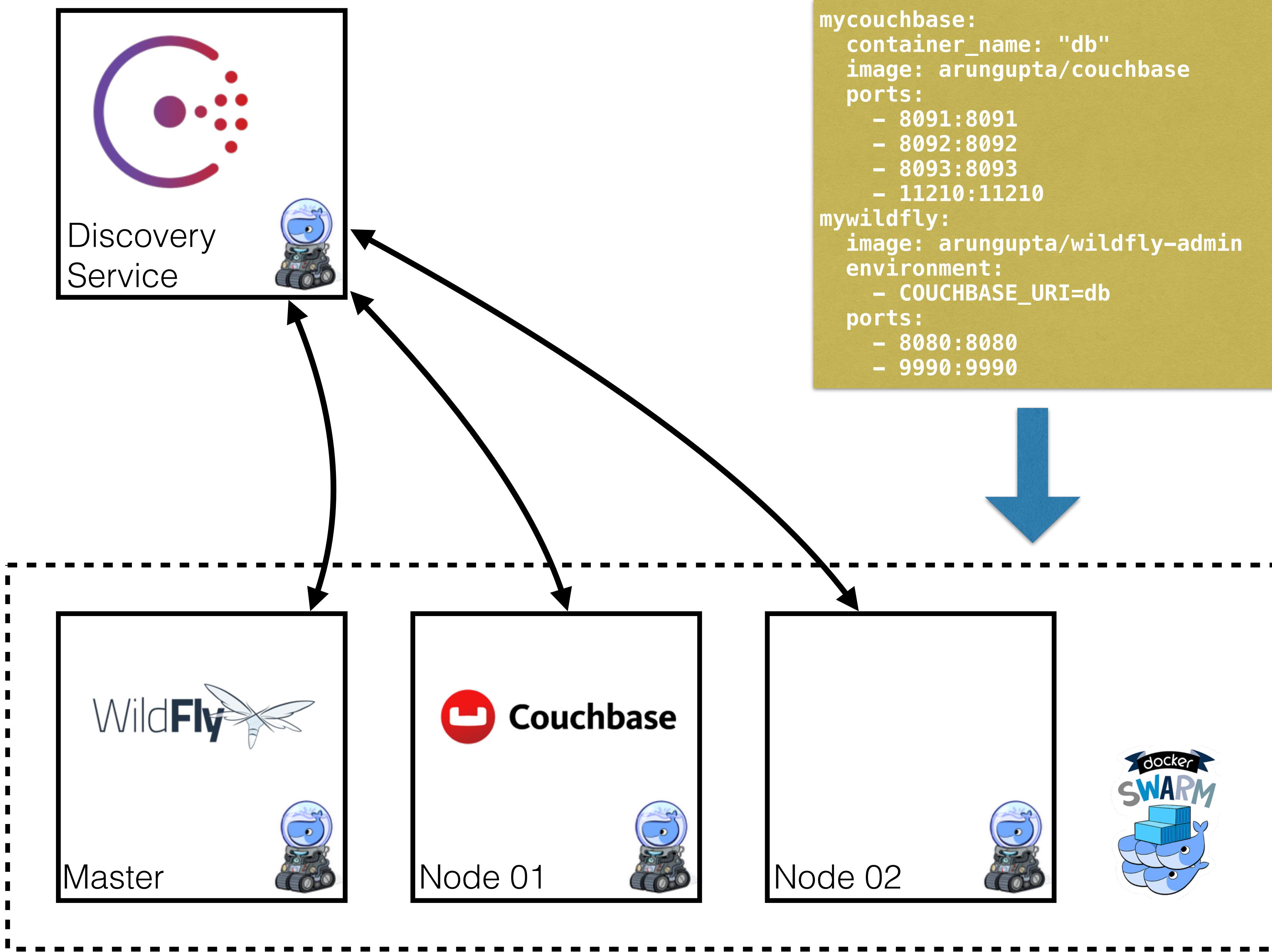
-c=1024

-c=512

-c=512

# Machine + Swarm + Compose





# Swarm Mode Overview

- New in 1.12, may be backwards incompatible
- Natively managing a cluster of Docker Engines called a Swarm
- Use Docker CLI to create a swarm, deploy apps, and manage swarm
- Uses SwarmKit

# Swarm Kit

- Toolkit for orchestrating distributed systems
- Raft-based consensus
- Swarm node - Worker and/or Manager
  - Worker comes with Docker Container Executor
  - Tasks run on Worker
  - Tasks are organized in Services
  - Service has desired state (e.g. how many replicas)
- Resource/strategy aware - *spread* is default
- Reconcile the desired and actual state
- Secure OOTB using mutual TLS

# Swarm Mode Features

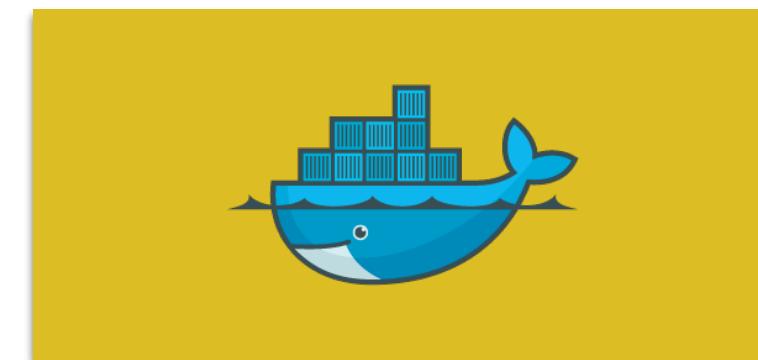
- Cluster management integrated with Docker Engine
- Decentralized design
- Declarative service model
- Scaling
- Desired state reconciliation
- Multi-host networking
- Service discovery
- Load balancing
- Secure by default
- Rolling updates

# Swarm Mode cluster on Amazon

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>
SSH	TCP	22	Anywhere <small>0.0.0.0/0</small>
Custom TCP Rule	TCP	2377	Anywhere <small>0.0.0.0/0</small>
Custom TCP Rule	TCP	7946	Anywhere <small>0.0.0.0/0</small>
Custom UDP Rule	UDP	7946	Anywhere <small>0.0.0.0/0</small>
Custom TCP Rule	TCP	4789	Anywhere <small>0.0.0.0/0</small>
Custom UDP Rule	UDP	4789	Anywhere <small>0.0.0.0/0</small>

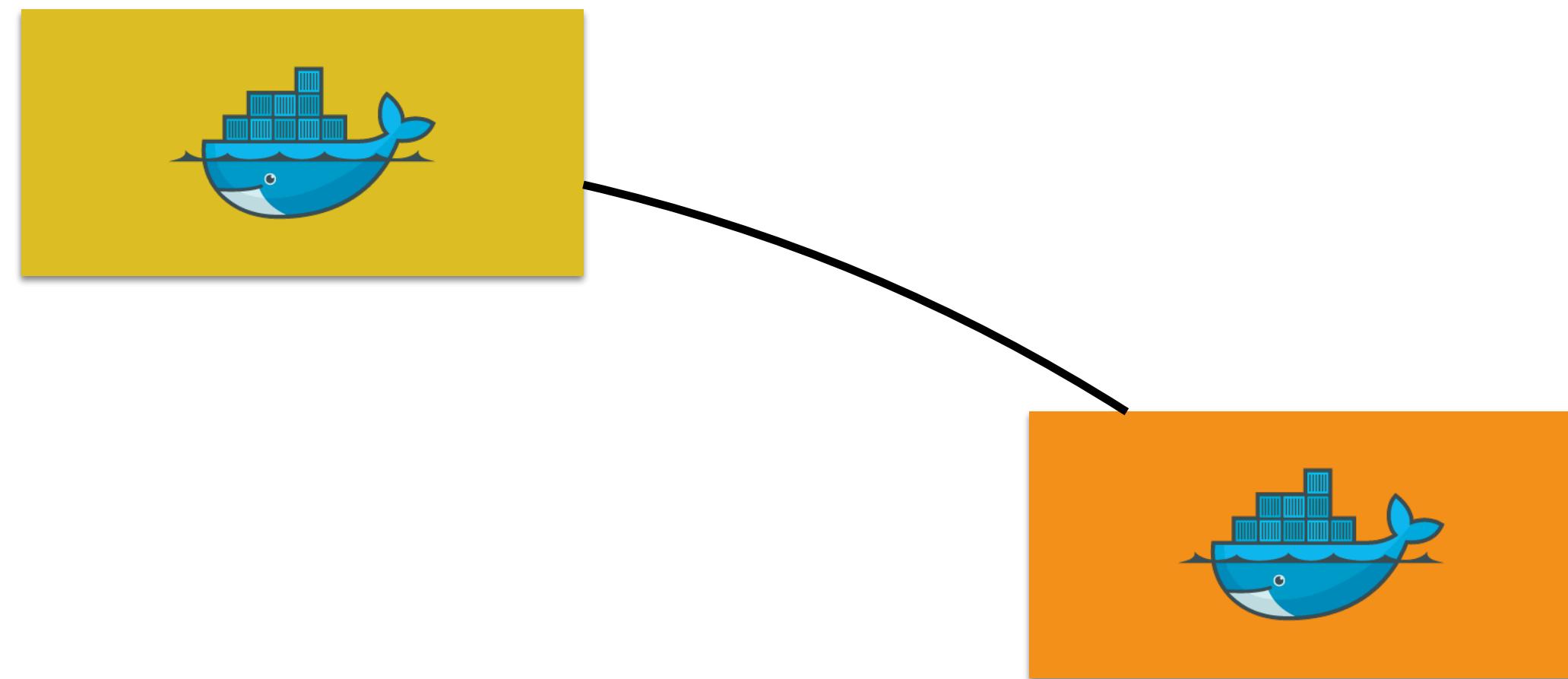
- 2377: Cluster management communications
- 7946: Communication among nodes
- 4789: Overlay network traffic

# Swarm Mode: Initialize



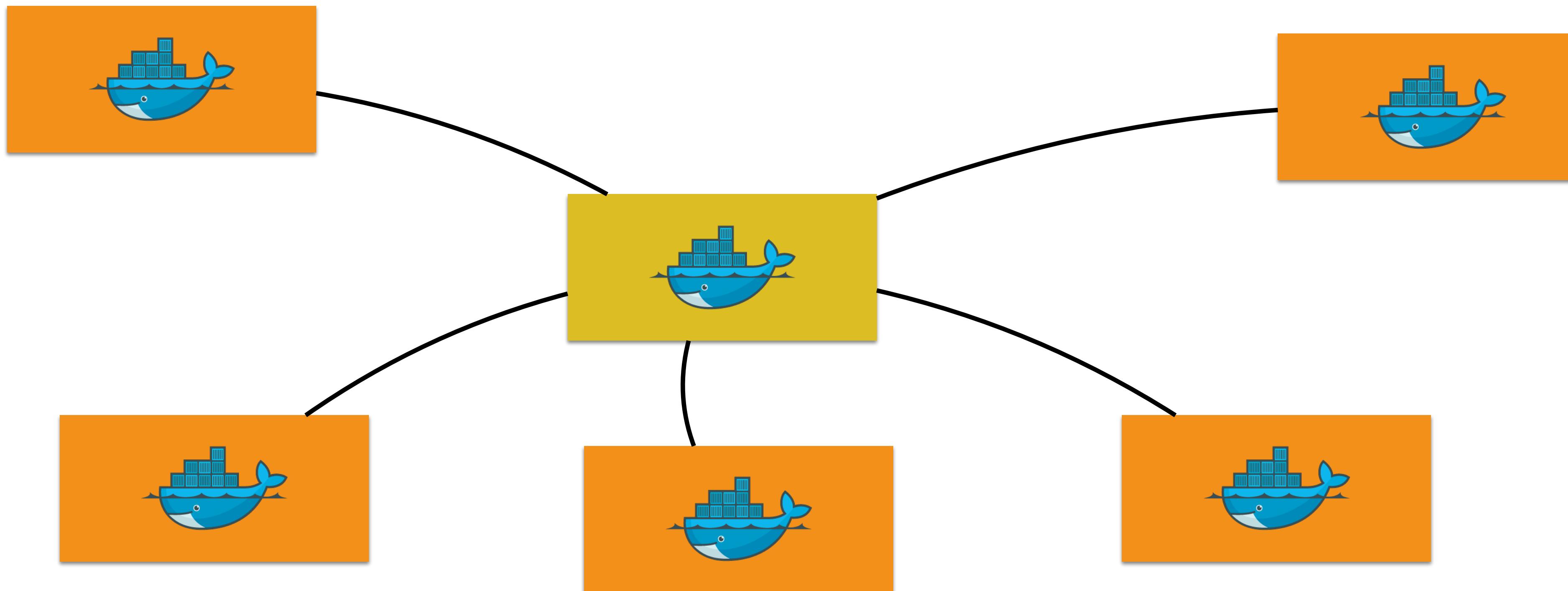
```
docker swarm init --listen-addr <private-ip>:2377
```

# Swarm Mode: Add Worker



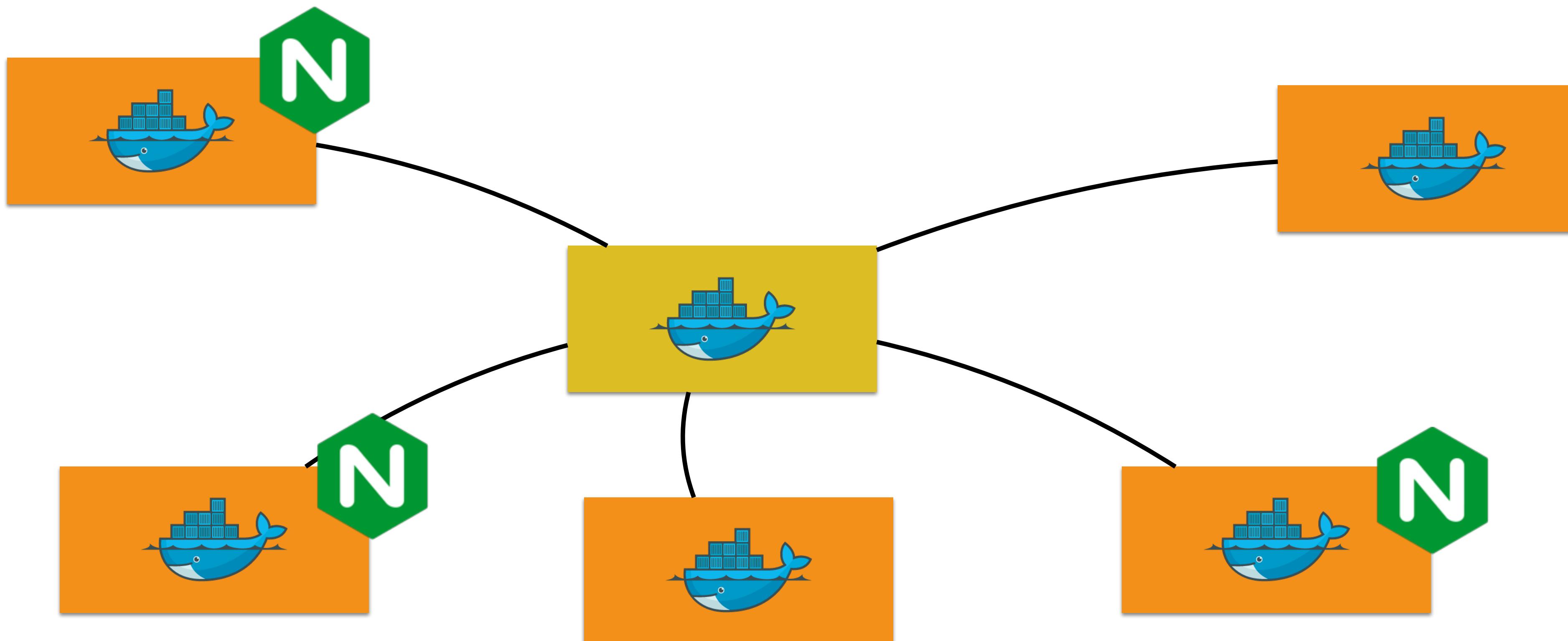
```
docker swarm join --secret <SECRET> <IP of manager>:2377
```

# Swarm Mode: Add More Workers



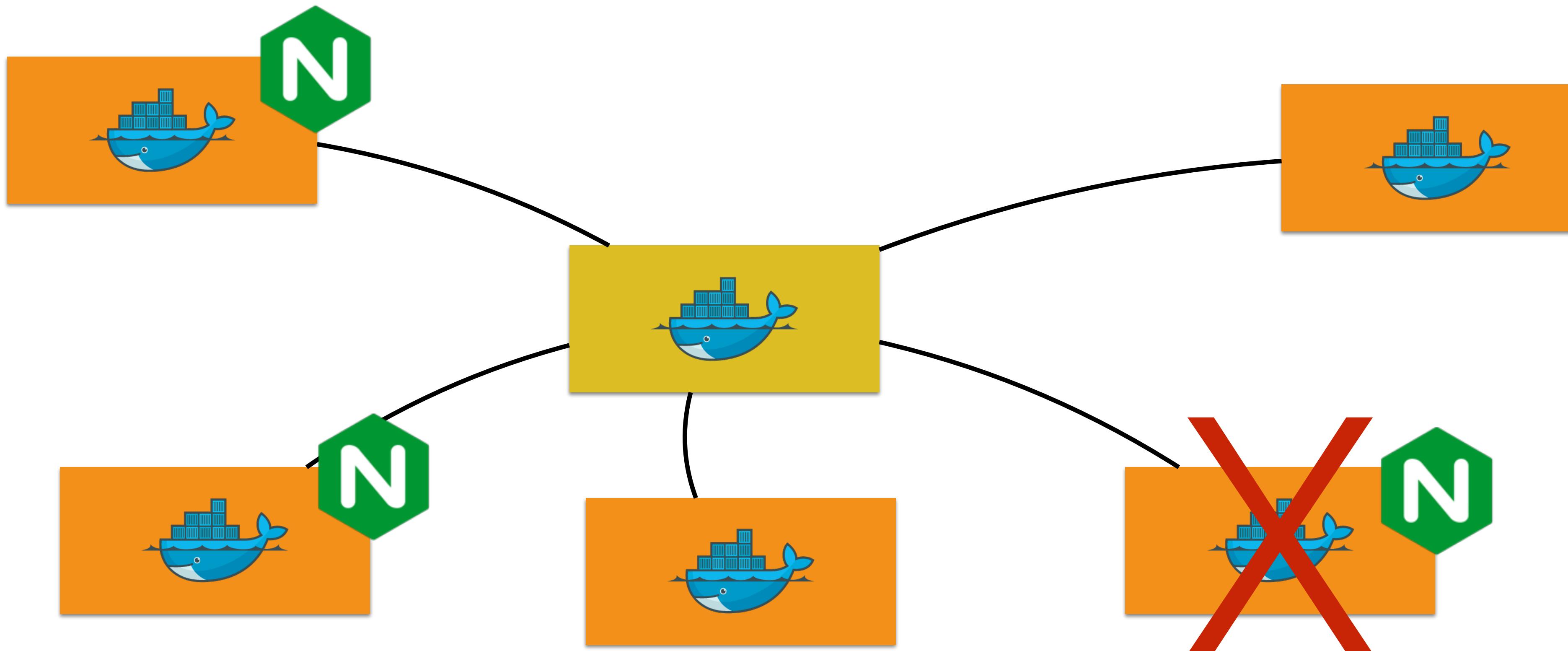
```
docker swarm join --secret <SECRET> <IP of manager>:2377
```

# Swarm Mode: Replicated Service

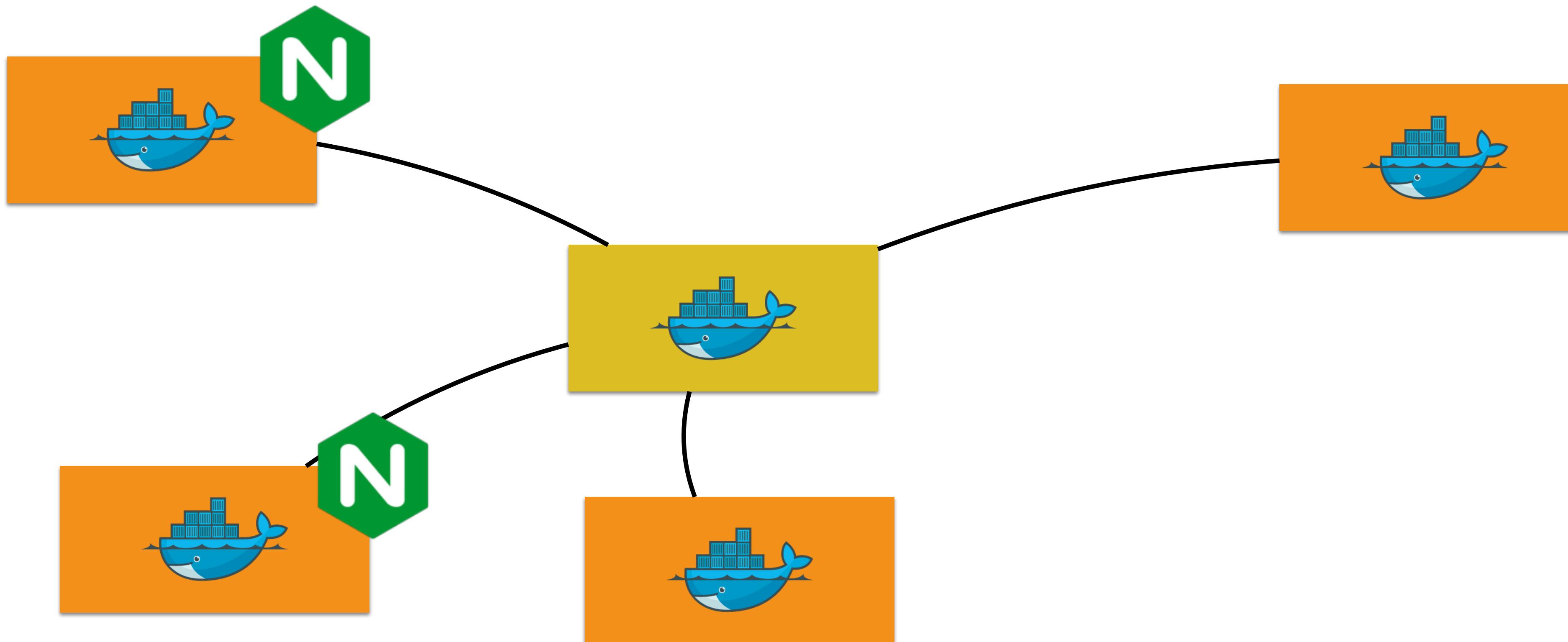


```
docker service create --replicas 3 --name frontend nginx:latest
```

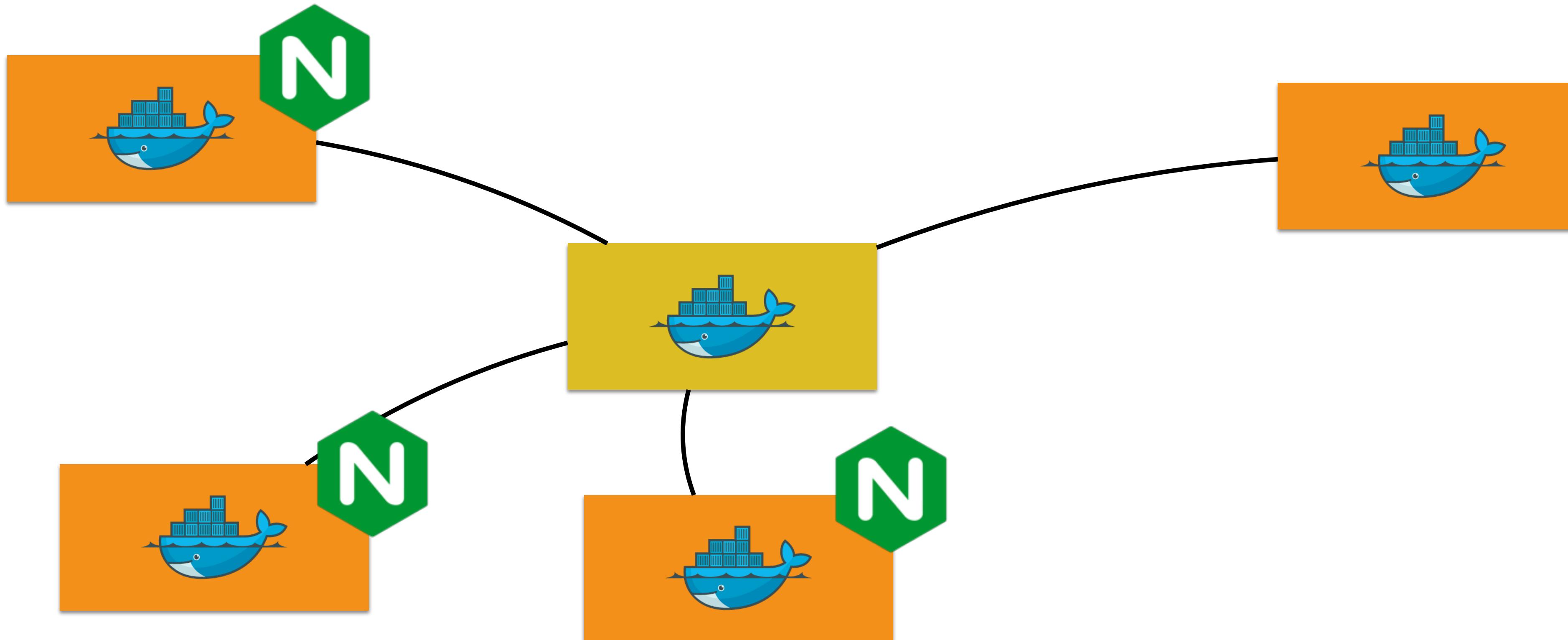
# Swarm Mode: Node Failure



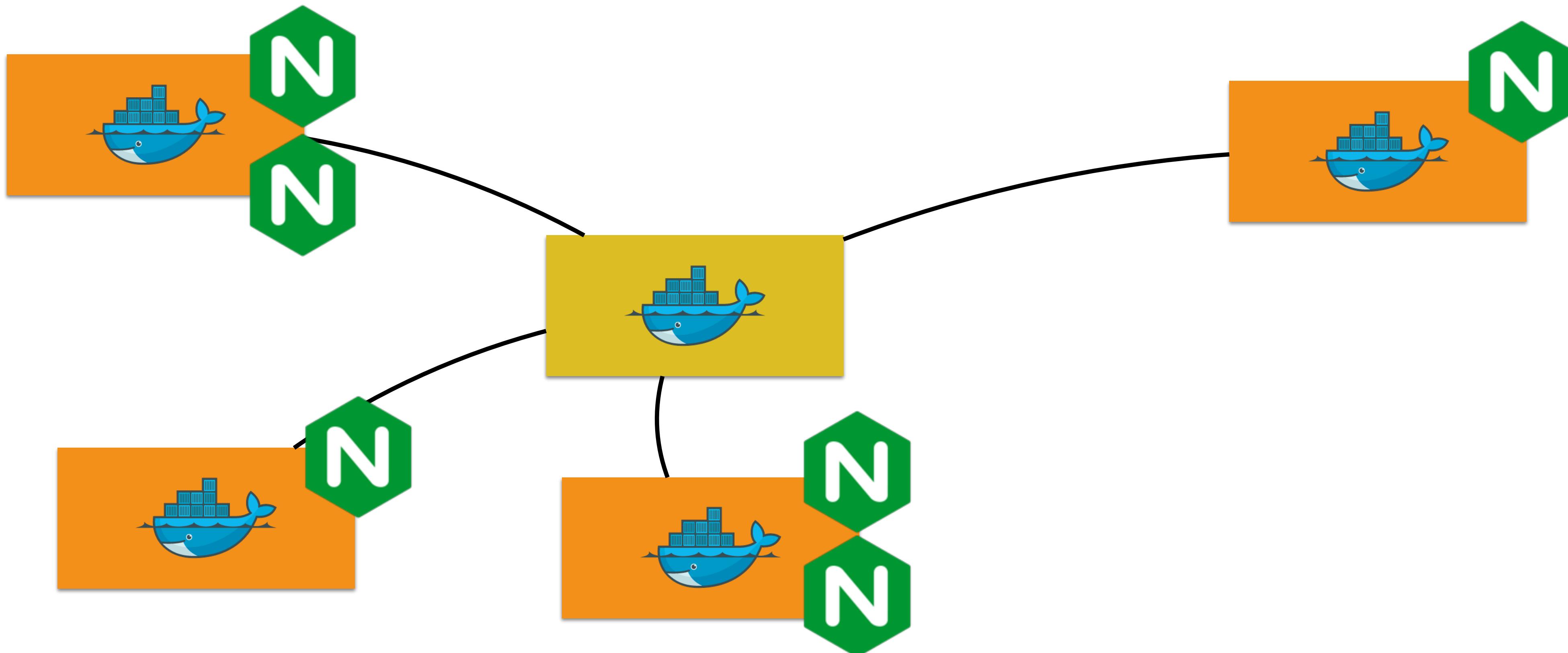
# Swarm Mode: Desired != Actual



# Swarm Mode: Reconcile

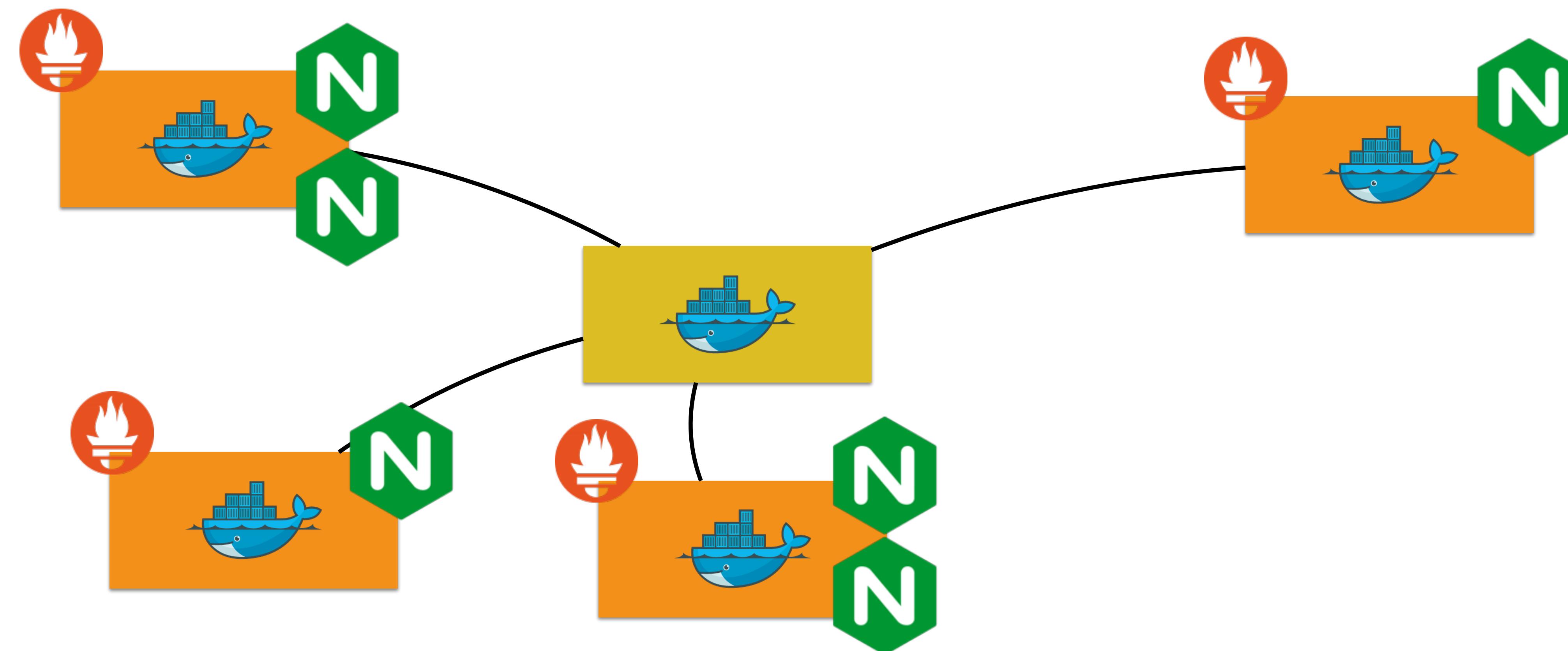


# Swarm Mode: Scale



```
docker service scale frontend=6
```

# Swarm Mode: Global Service



```
docker service create --mode=global --name=prom prom/prometheus
```

# Persistent Storage

- Data volumes - used to persist data independent of container's lifecycle
- Multiple plugins: Flocker, Ceph, . . .

```
docker volume --help

Usage: docker volume [OPTIONS] [COMMAND]

Manage Docker volumes

Commands:
  create           Create a volume
  inspect          Return low-level information on a volume
  ls               List volumes
  rm              Remove a volume
```

# Persistent Storage

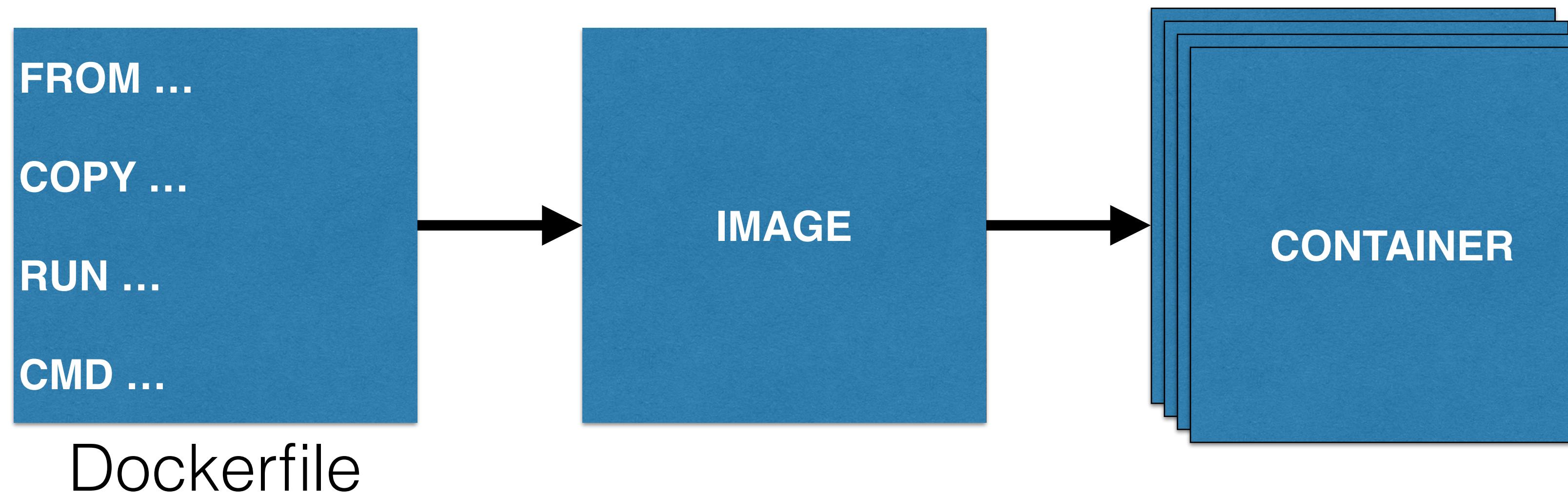
Create a volume

```
docker volume create --name=data data
```

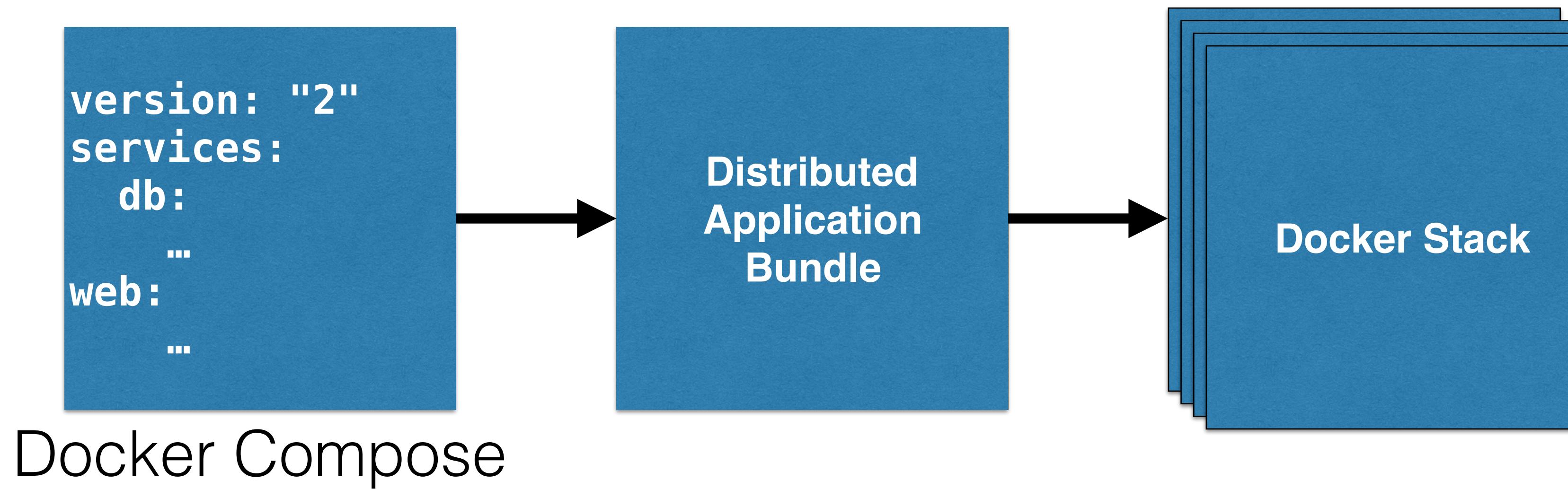
Run a container with the volume

```
docker run -it -v data:/opt/couchbase/var couchbase
```

# Docker Lifecycle



# Distributed Application Bundle





# Docker Cloud

- SaaS
- Build your images
- Deploy and manage across different clouds
  - Amazon, Digital Ocean, Microsoft, Azure, IBM SoftLayer
  - BYON





# Docker Cloud



## Build

Containerize your applications and accelerate development.



## Deploy

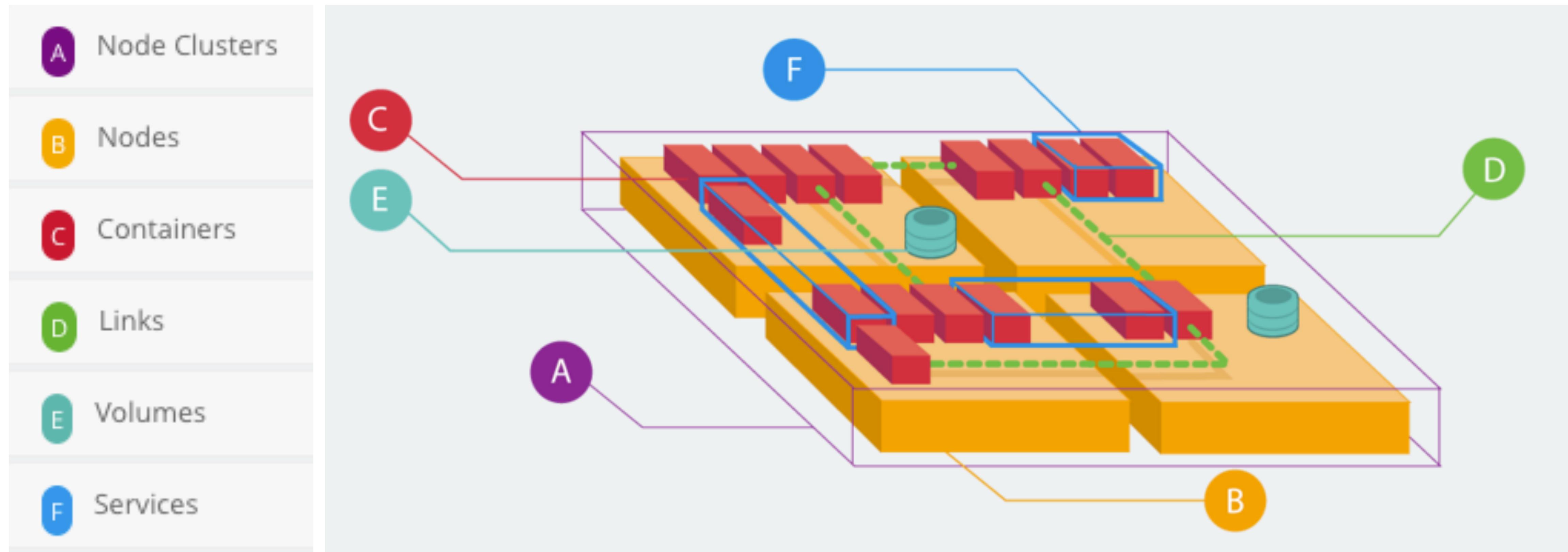
All frameworks and technologies welcomed.  
Scale with ease on any Cloud.



## Manage

Simplify operations, focus on your code, and forget about managing servers.

# Docker Cloud Architecture



# Create a node cluster



Node cluster name

Deploy tags

Provider

Region

VPC

Subnet

Image

Service name

Image tag

Stack

Deployment strategy

Number of containers

Deploy tags

Ports

Container port	Protocol	Published	Node port
11207	tcp	<input type="checkbox"/>	--
11210	tcp	<input checked="" type="checkbox"/>	dynamic
11211	tcp	<input type="checkbox"/>	--
tcp		<input type="checkbox"/>	--
tcp		<input type="checkbox"/>	--
tcp		<input type="checkbox"/>	--
tcp		<input checked="" type="checkbox"/>	dynamic
tcp		<input checked="" type="checkbox"/>	dynamic
tcp		<input checked="" type="checkbox"/>	dynamic

+ Add Port

Autodestroy

Autoredeploy  OFF

## Advanced options

couchbase-1924aaf6 / COUC...

Stop Terminate Redeploy

### ► Running

34 minutes ago

Endpoints

Logs

Environment variables

Volumes

Terminal

Timeline

Automatically refreshing

arungupta/couchbase:latest

>\_ ./entrypoint.sh /opt/couchbase/...

32775->8091/tcp 32774->8092/tcp  
32773->8093/tcp 32772->11210/tcp

11207/tcp 11211/tcp 18091/tcp  
18092/tcp

10.7.0.2

off

\* None

Bridge

```
2016-03-21T22:07:13.351371182Z * upload completely sent off: 26 out of 26 bytes
2016-03-21T22:07:13.353756594Z < HTTP/1.1 200 OK
2016-03-21T22:07:13.353780275Z < Server: Couchbase Server
2016-03-21T22:07:13.353789344Z < Pragma: no-cache
2016-03-21T22:07:13.353797575Z < Date: Mon, 21 Mar 2016 22:07:13 GMT
2016-03-21T22:07:13.353805473Z < Content-Length: 0
2016-03-21T22:07:13.353816319Z < Cache-Control: no-cache
2016-03-21T22:07:13.353824677Z <
2016-03-21T22:07:13.354209316Z 100 26 0 0 100 26 0 6772 --:--:-- --:--:-- 8666
2016-03-21T22:07:13.354226834Z * Connection #0 to host 127.0.0.1 left intact
2016-03-21T22:07:13.357390925Z * Trying 127.0.0.1...
2016-03-21T22:07:13.357713200Z * Total % Received % Xferd Average Speed Time Time Current
2016-03-21T22:07:13.357765231Z Dload Upload Total Spent Left Speed
2016-03-21T22:07:13.357940902Z 0 0 0 0 0 0 0 0 --:--:-- --:--:-- 0* Conn
2016-03-21T22:07:13.358003091Z > POST /settings/web HTTP/1.1
2016-03-21T22:07:13.358057746Z > User-Agent: curl/7.40.0-DEV
2016-03-21T22:07:13.358110137Z > Host: 127.0.0.1:8091
2016-03-21T22:07:13.358162421Z > Accept: */*
2016-03-21T22:07:13.358217086Z > Content-Length: 50
2016-03-21T22:07:13.358268561Z > Content-Type: application/x-www-form-urlencoded
2016-03-21T22:07:13.358317827Z >
2016-03-21T22:07:13.358677410Z } [50 bytes data]
2016-03-21T22:07:13.359053352Z * upload completely sent off: 50 out of 50 bytes
2016-03-21T22:07:13.939725813Z < HTTP/1.1 200 OK
2016-03-21T22:07:13.939801035Z < Server: Couchbase Server
2016-03-21T22:07:13.939854726Z < Pragma: no-cache
2016-03-21T22:07:13.939901212Z < Date: Mon, 21 Mar 2016 22:07:13 GMT
2016-03-21T22:07:13.939945663Z < Content-Type: application/json
2016-03-21T22:07:13.939991254Z < Content-Length: 39
2016-03-21T22:07:13.940043843Z < Cache-Control: no-cache
2016-03-21T22:07:13.940088813Z <
2016-03-21T22:07:13.940306545Z { [39 bytes data]
2016-03-21T22:07:13.941109018Z 100 89 100 39 100 50 66 85 --:--:-- --:--:-- 85
2016-03-21T22:07:13.941177878Z * Connection #0 to host 127.0.0.1 left intact
2016-03-21T22:07:13.942283607Z {"newBaseUri":"http://127.0.0.1:8091/"}/entrypoint.sh couchbase-server
```

# Docker Cloud CLI

- brew install docker-cloud
- docker-cloud nodecluster create -t 1 --tag couchbase couchbase-node aws us-west-1 m3.large
- docker-cloud service create --tag couchbase -p 8091:8091 -p 8092:8092 -p 8093:8093 -p 11210:11210 arungupta/couchbase
- docker-cloud service start {SERVICE\_ID}
- docker-cloud service inspect {SERVICE\_ID} | jq ".container\_ports[0].endpoint\_uri" | sed 's/tcp/http/g'



# Docker Registry

- Store and distribute Docker images
  - Control where images are stored
  - Own image distribution pipeline
  - Integrate image storage/distribution in dev workflow
- Docker Hub
  - Free-to-use and hosted
- Docker Trusted Registry
  - Commercially supported
  - RBAC, LDAP/AD integration, updates, etc

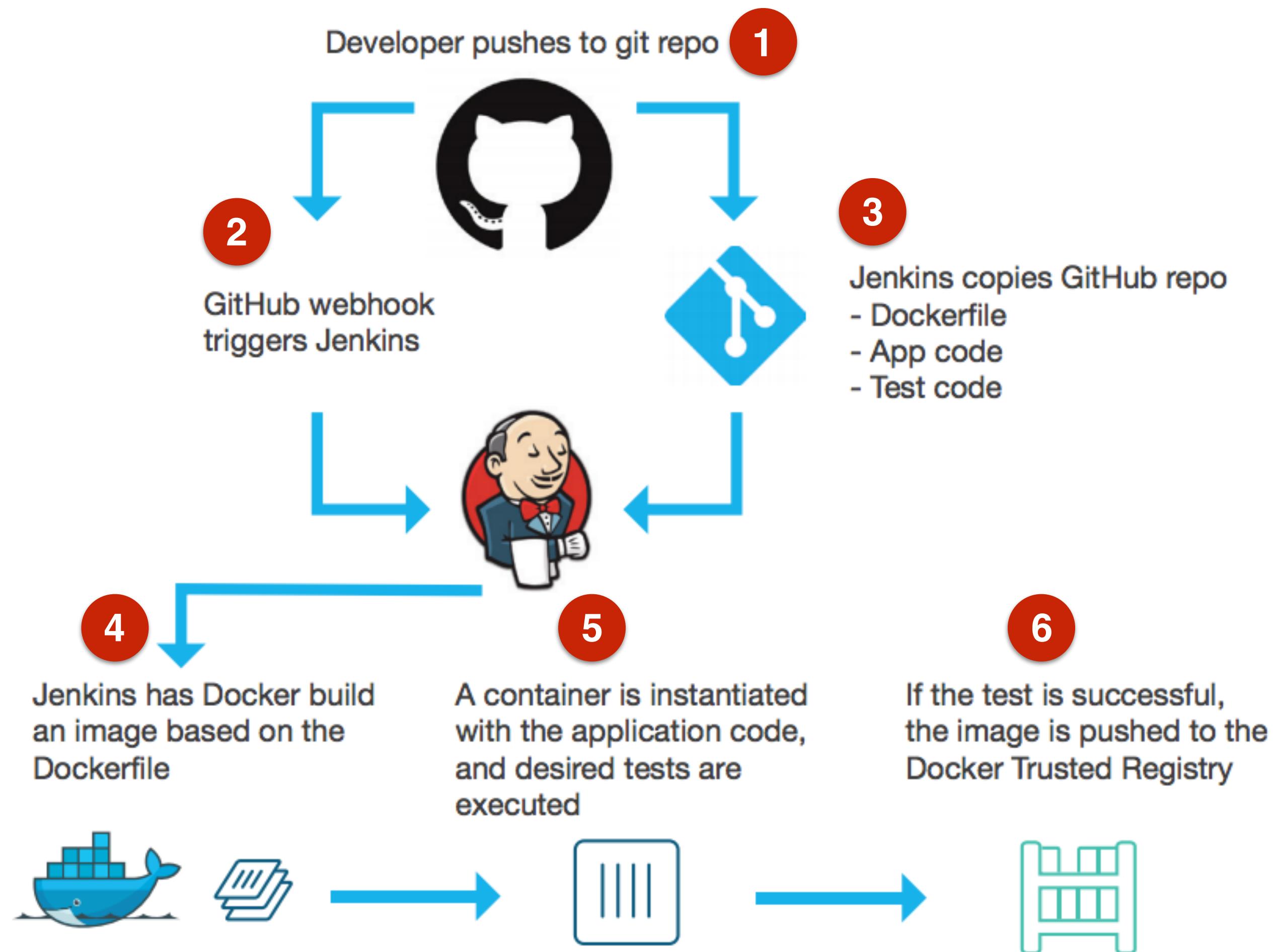


# Registry Usage

- CI/CD with Docker
  - Centrally located base images
  - Store individual build images
  - Pull tested images to production

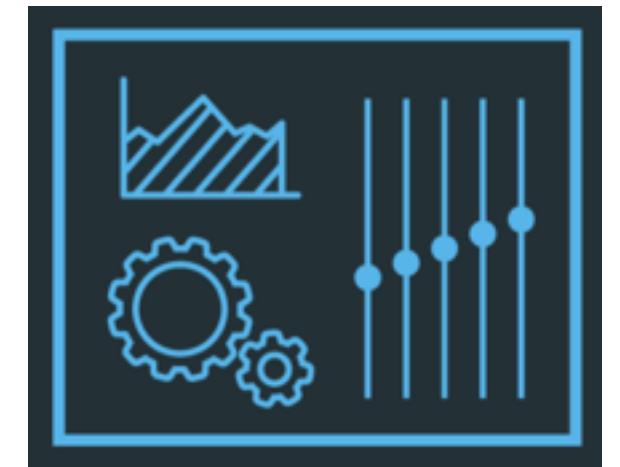
# CI/CD with Docker + Jenkins

1. Push a commit to GitHub
2. GitHub webhook
3. Jenkins copies artifacts
4. Jenkins builds Docker image
5. Runs test on Docker container
6. Pushes to DTR



# Monitoring Docker Containers

- `docker stats` command
  - LogEntries
- Docker Remote API: `/container/{container-name|cid}/stats`
- Docker Universal Control Plane
- cAdvisor
  - Prometheus
  - InfluxDB

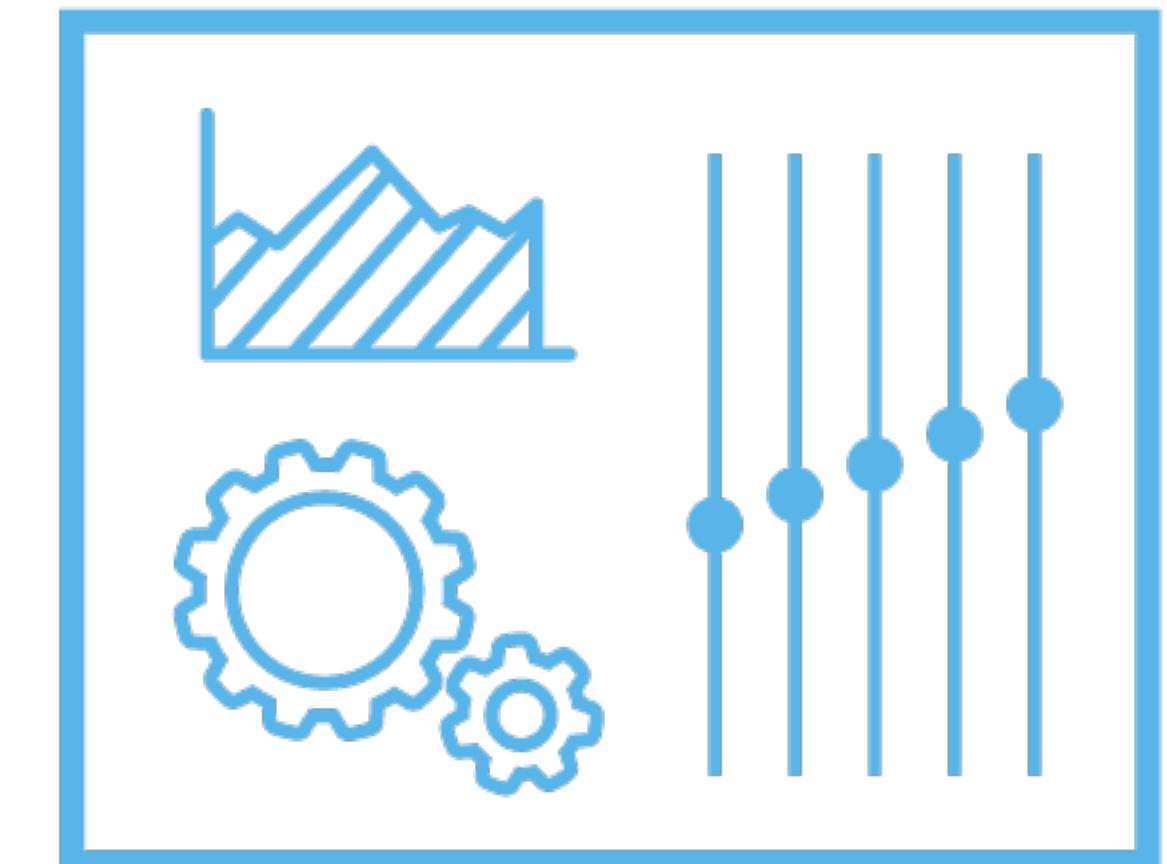


cAdvisor

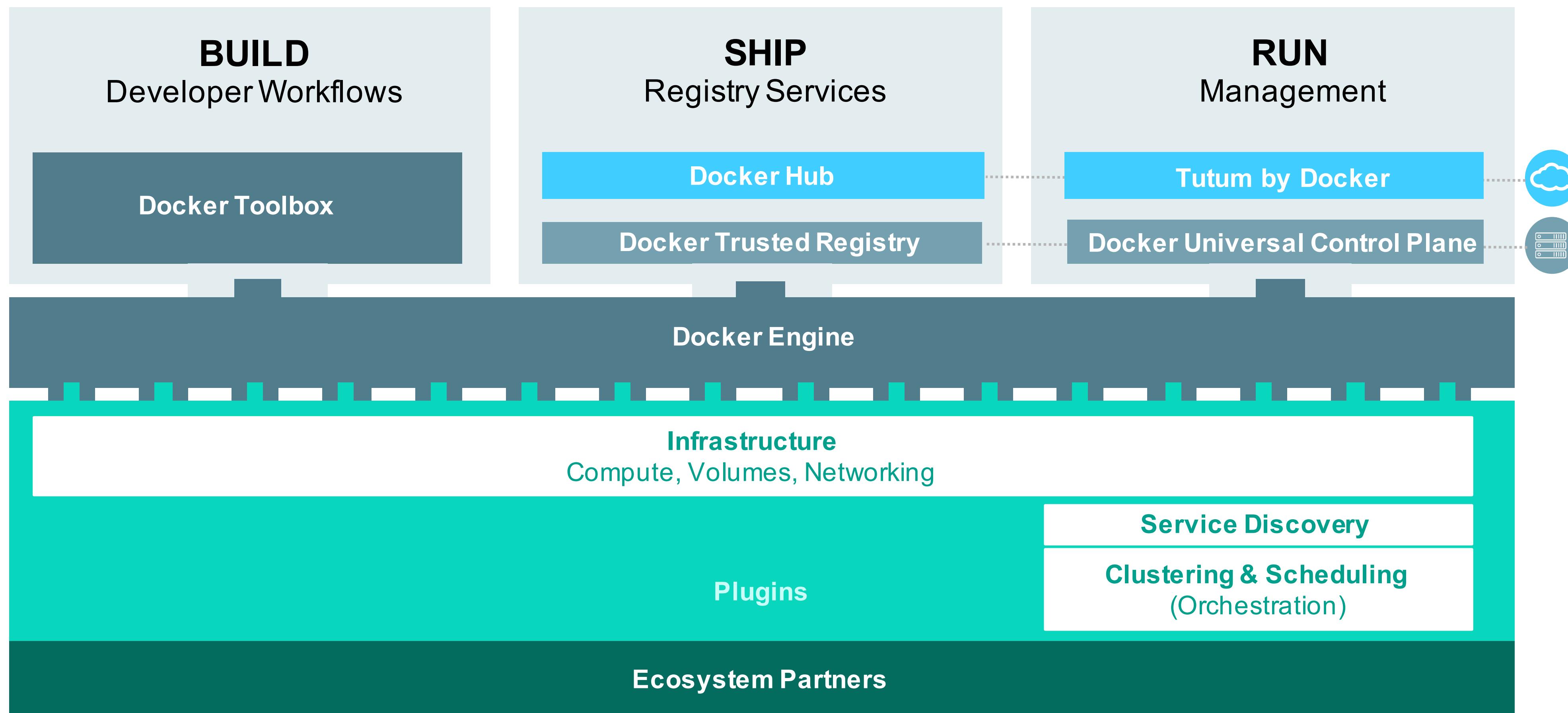


# Docker Universal Control Plane

- On-premise management for Docker containers
- Build on Docker Remote API
  - Integrated support for CLI, Swarm, Compose and Trusted Registry
- Integration with LDAP/AD
- Audit Logs



# Docker Mission



# References

- Docker for Java Tutorial: [github.com/docker/labs/tree/master/java](https://github.com/docker/labs/tree/master/java)
- Docker Documentation: [docs.docker.com](https://docs.docker.com)