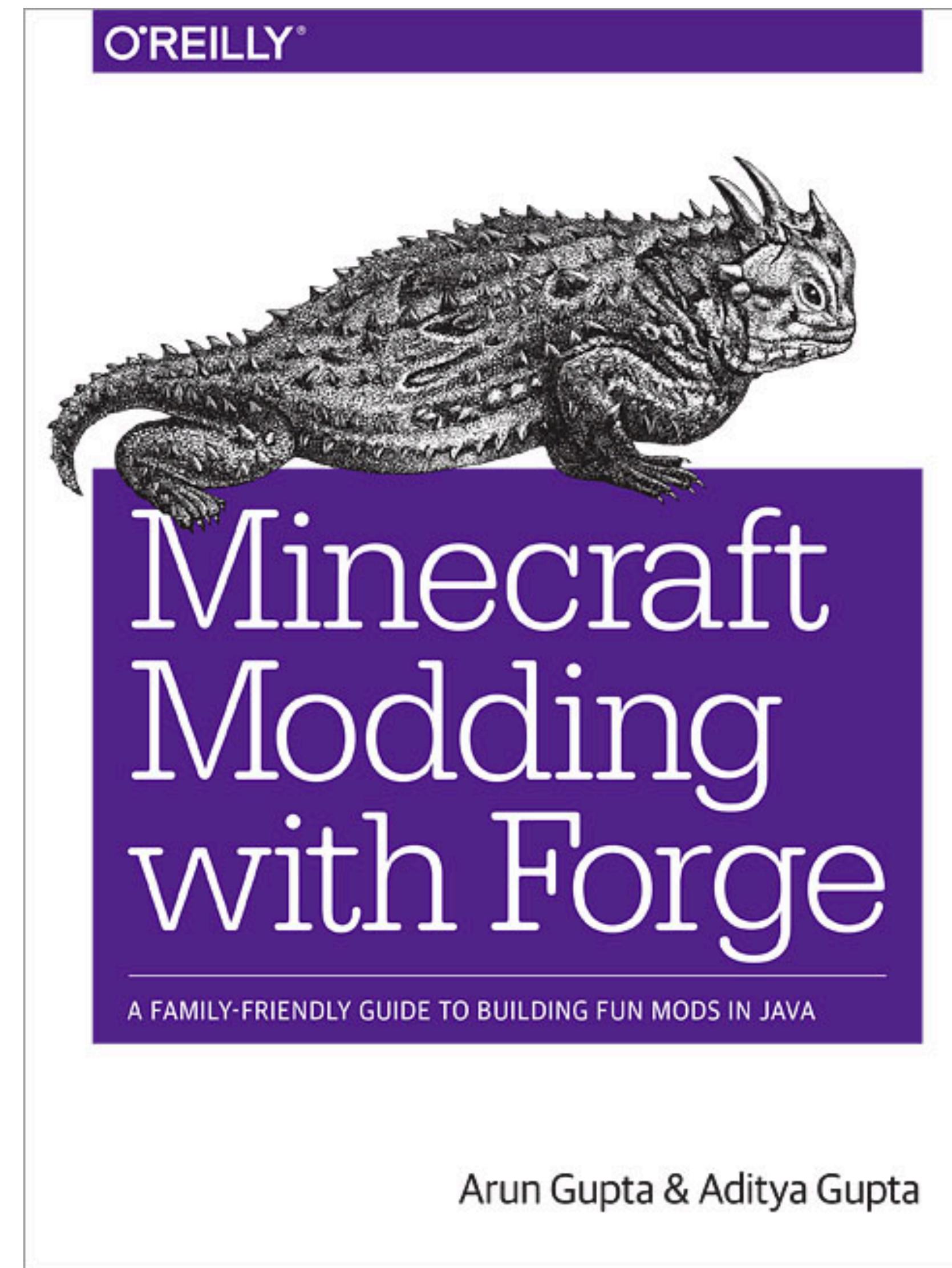


docker

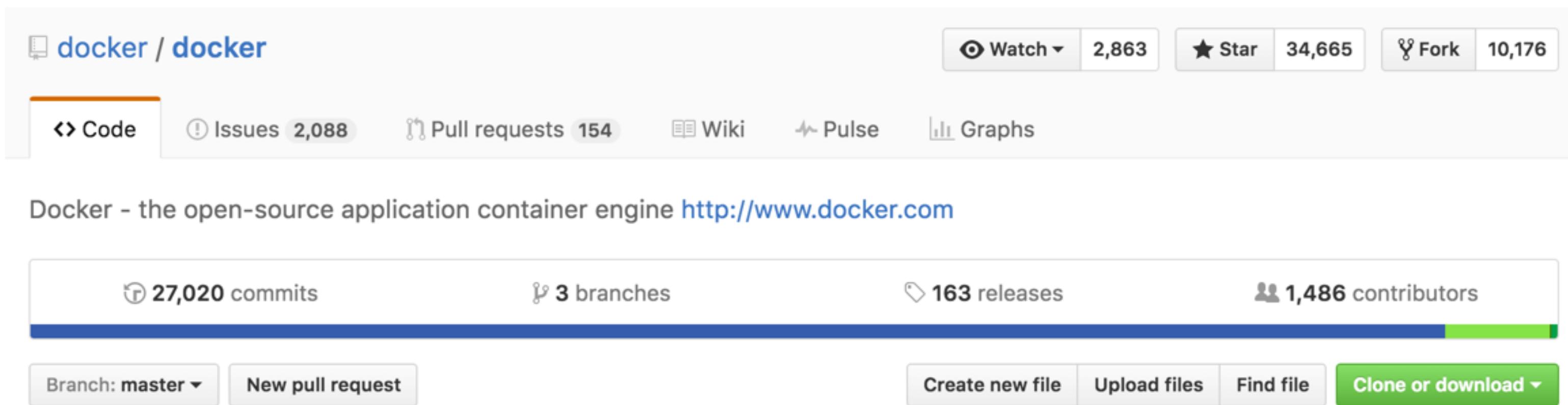
# Docker for Java Developers

Arun Gupta, @arungupta  
VP Developer Advocacy, Couchbase

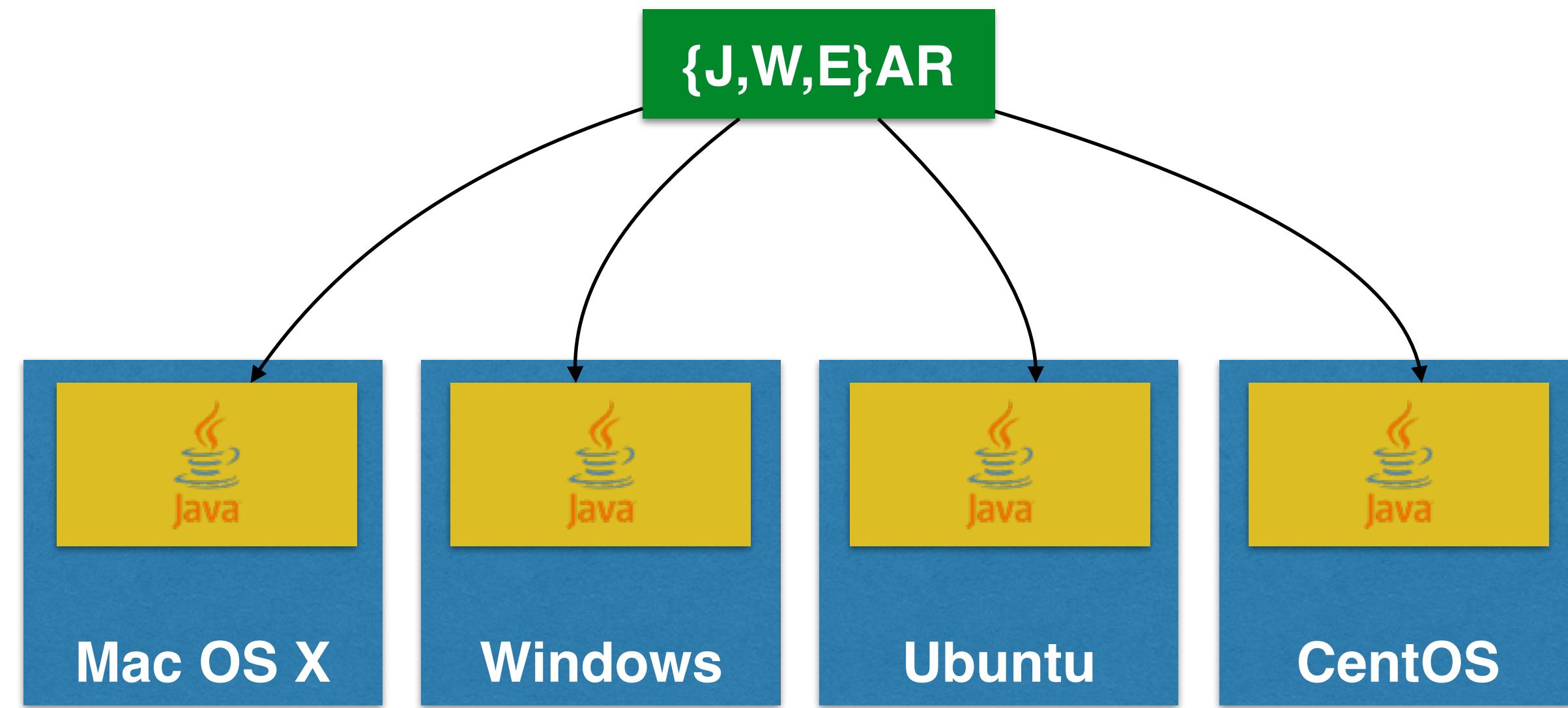


# What is Docker?

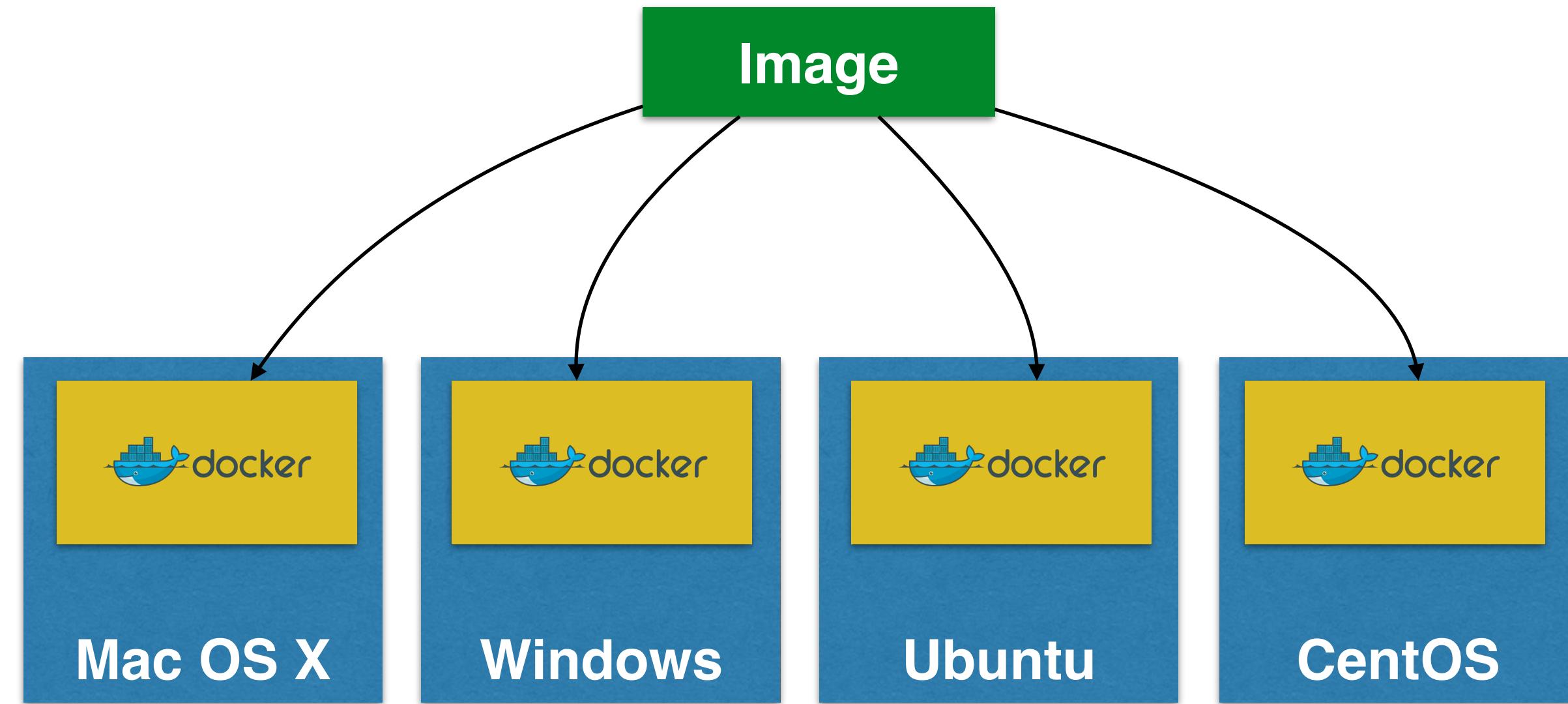
- Open source project and company



- Used to create containers for software applications

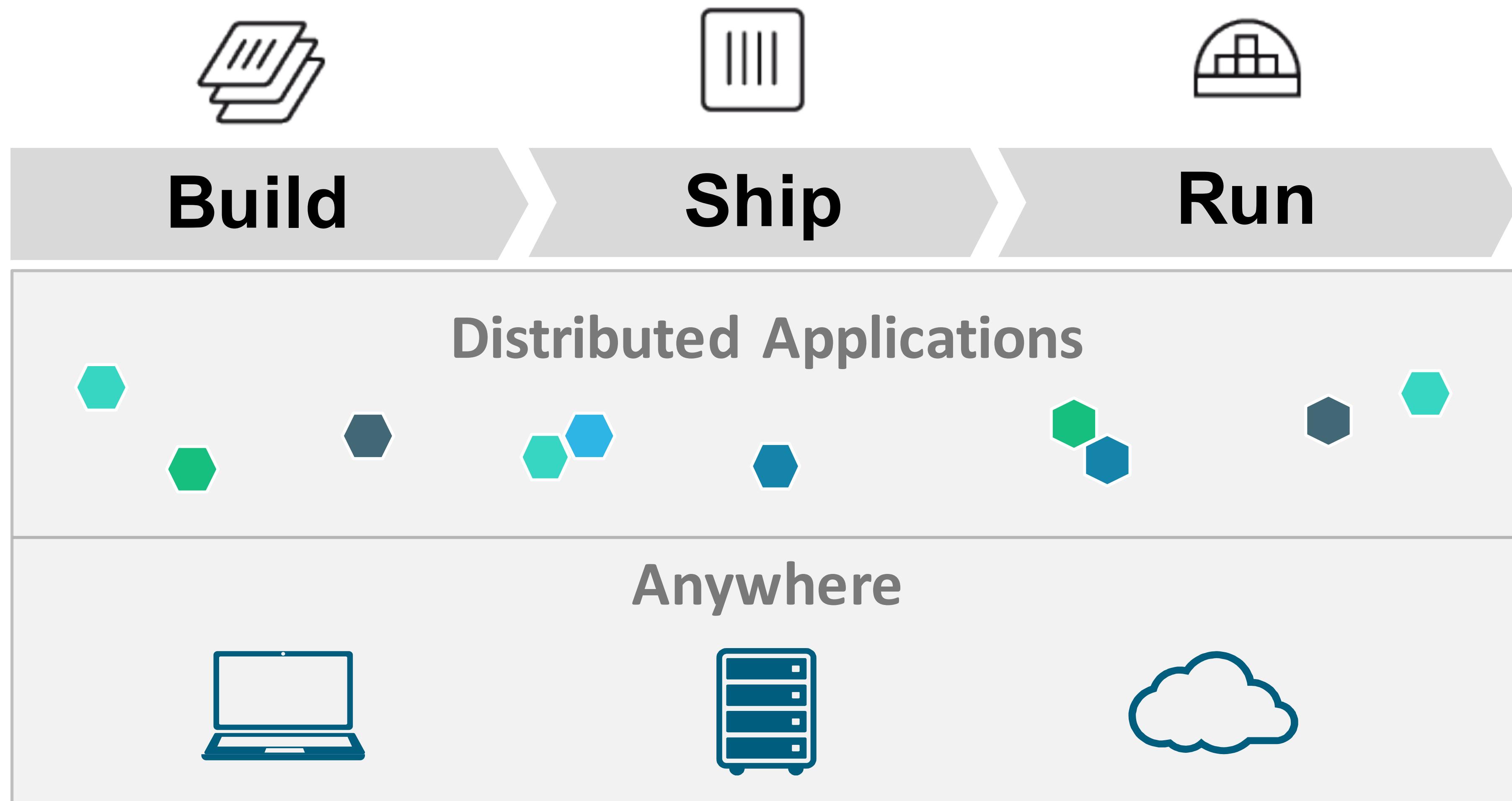


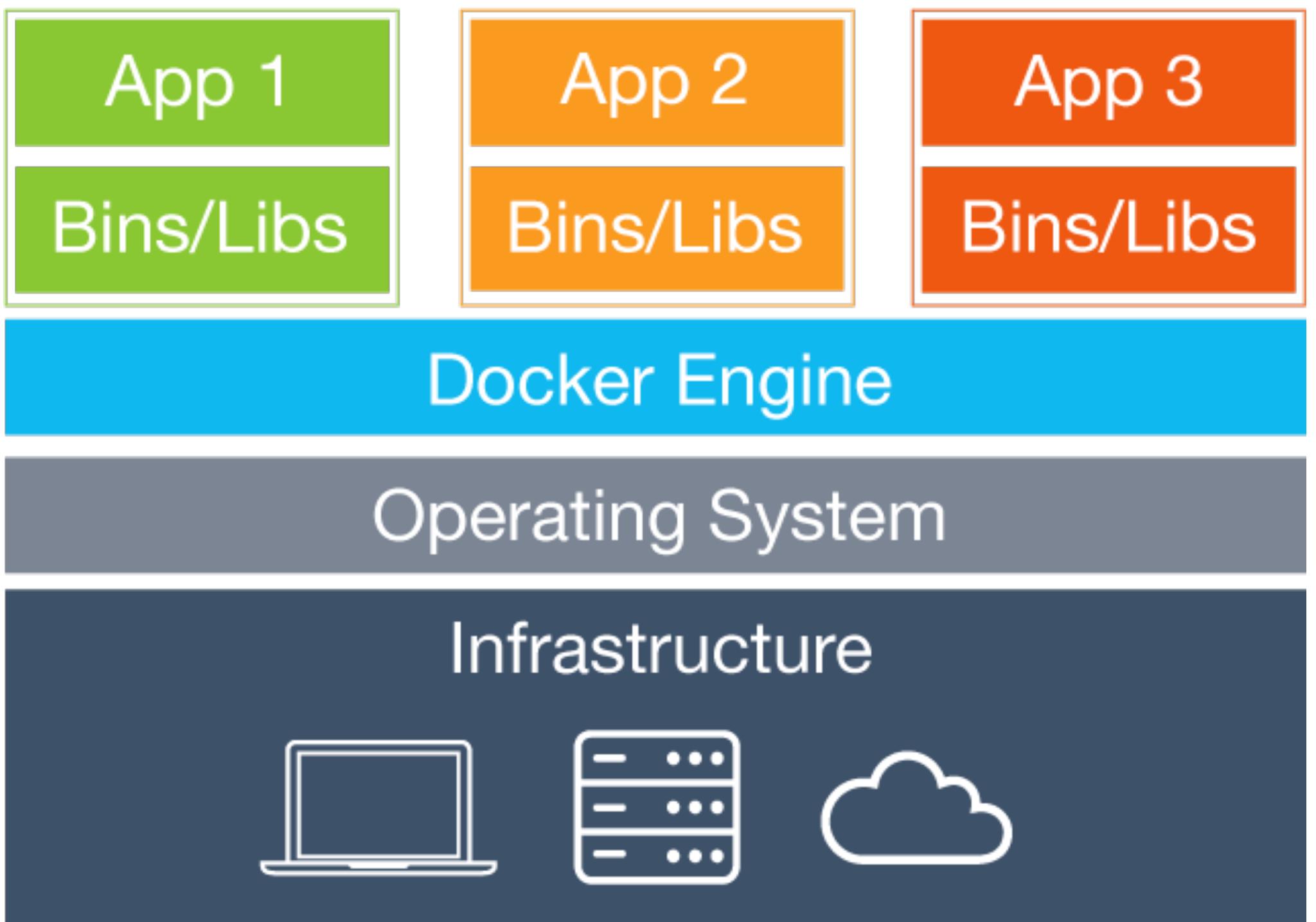
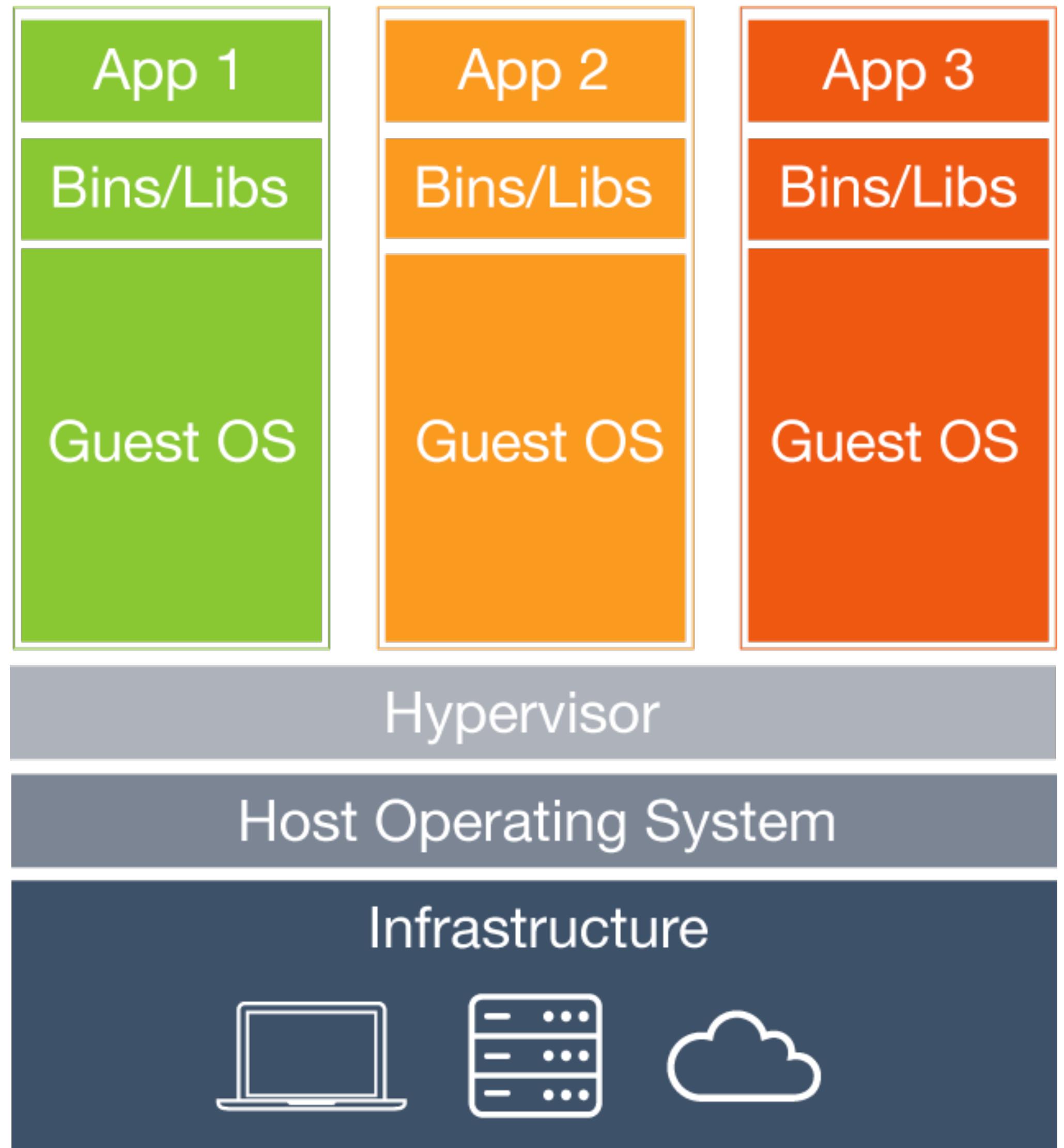
**WORA = Write Once Run Anywhere**



**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 openjdk
```

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

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

[Usage](#)

[Format](#)

[Parser directives](#)

[escape](#)

[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](#)

[Understand how CMD and ENTRYPOINT interact](#)

[VOLUME](#)

[USER](#)

[WORKDIR](#)

[ARG](#)

[Impact on build caching](#)

[ONBUILD](#)

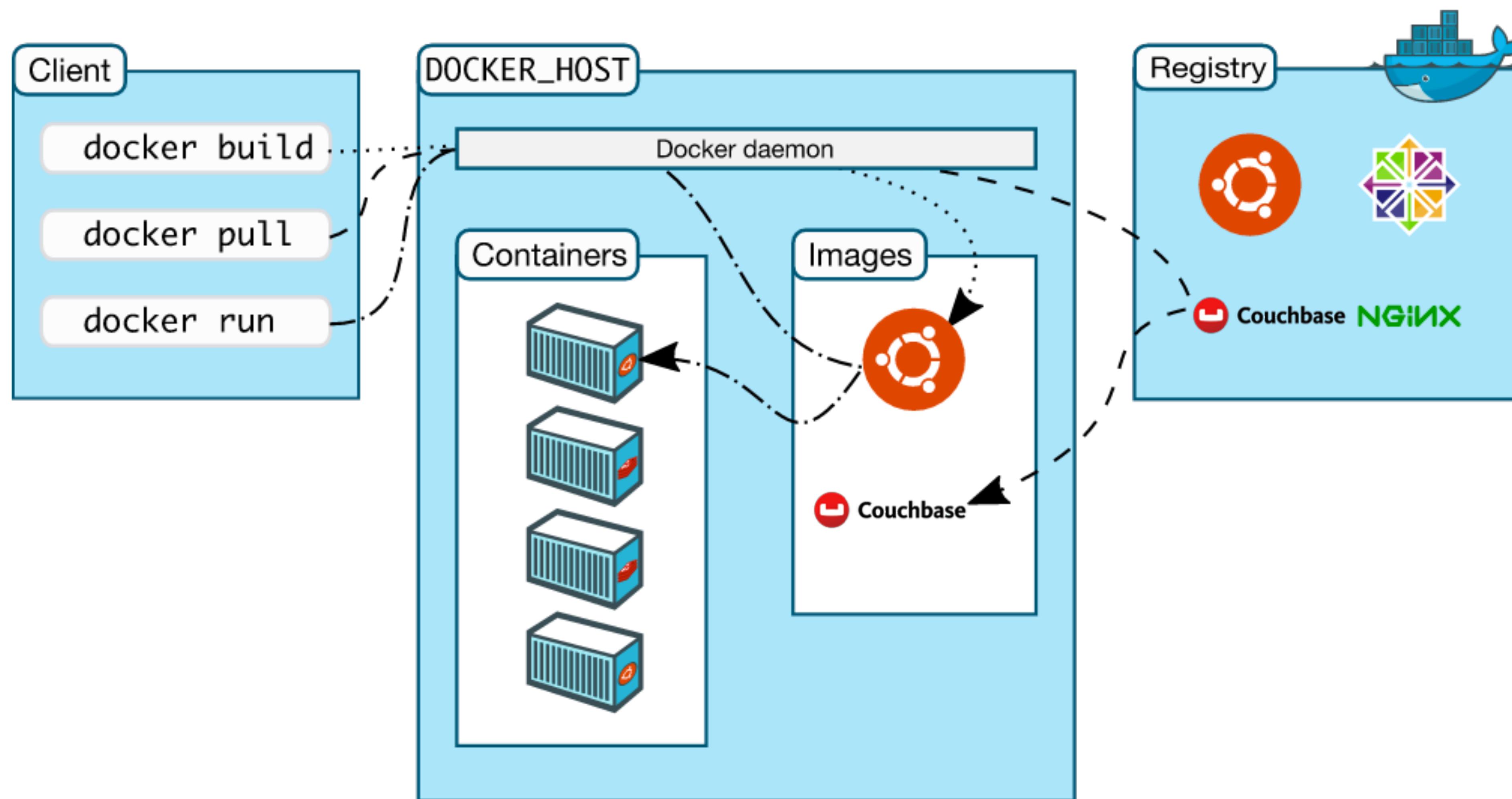
[STOPSIGNS](#)

[HEALTHCHECK](#)

[SHELL](#)

[Dockerfile examples](#)

# Docker Workflow



# Image Layers - OpenJDK

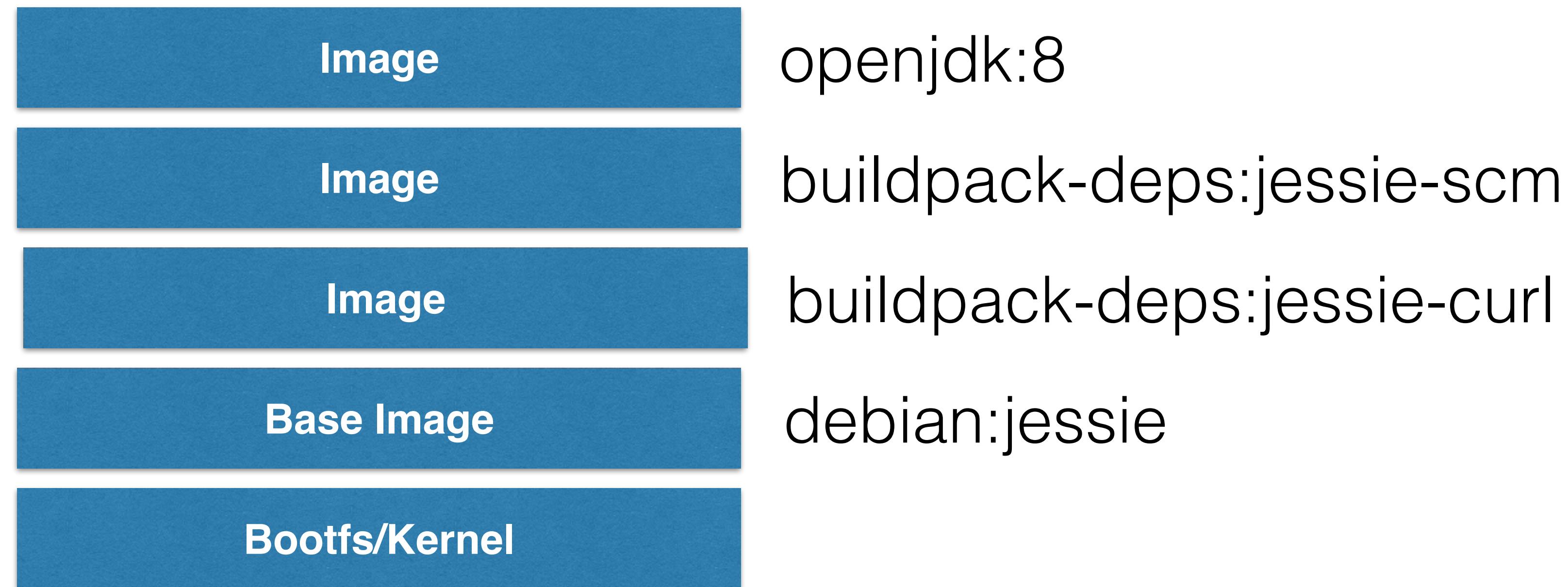
```
~ > docker images openjdk
```

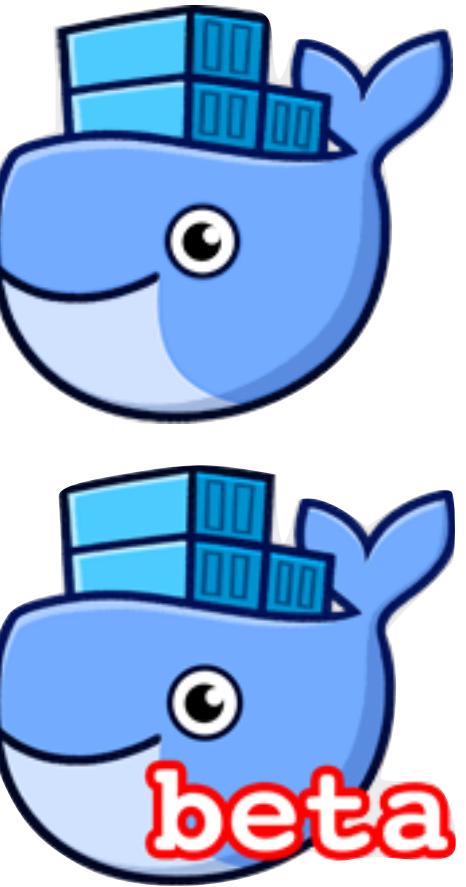
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
openjdk	latest	ea40c858f006	13 days ago	643.1 MB

```
~ > docker history openjdk
```

IMAGE	CREATED	CREATED BY	SIZE
COMMENT			
ea40c858f006	13 days ago	/bin/sh -c /var/lib/dpkg/info/ca-certificates	418.2 kB
<missing>	13 days ago	/bin/sh -c set -x && apt-get update && apt-	349.3 MB
<missing>	13 days ago	/bin/sh -c #(nop) ENV CA_CERTIFICATES_JAVA_V	0 B
<missing>	13 days ago	/bin/sh -c #(nop) ENV JAVA_DEBIAN_VERSION=8u	0 B
<missing>	13 days ago	/bin/sh -c #(nop) ENV JAVA_VERSION=8u102	0 B
<missing>	13 days ago	/bin/sh -c #(nop) ENV JAVA_HOME=/usr/lib/jvm	0 B
<missing>	13 days ago	/bin/sh -c { echo '#!/bin/sh'; echo 'set	87 B
<missing>	13 days ago	/bin/sh -c #(nop) ENV LANG=C.UTF-8	0 B
<missing>	13 days ago	/bin/sh -c echo 'deb http://httpredir.debian.	61 B
<missing>	13 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	44.31 MB
<missing>	2 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0 B
<missing>	2 weeks ago	/bin/sh -c #(nop) ADD file:f2453b914e7e026efd	125.1 MB

# Union File System





# Docker for Mac/Windows

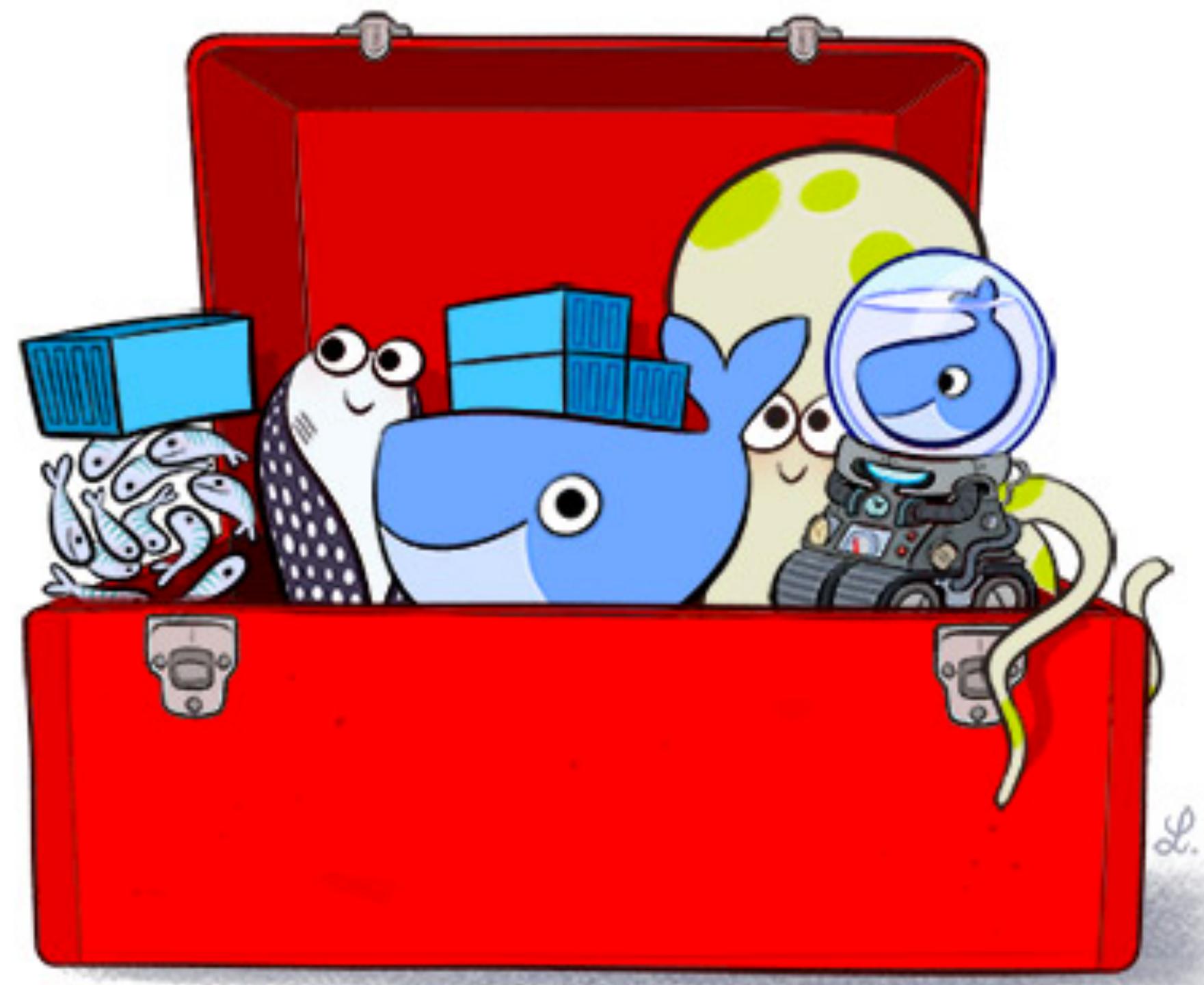
- Native application and UI
- Auto update capability
- No additional software required, e.g. VirtualBox
  - OSX: xhyve VM using `Hypervisor.framework`
  - Windows: Hyper-V VM
- Download: [docker.com/getdocker](http://docker.com/getdocker)
- Requires Yosemite 10.10+ or Windows 10 64-bit

# Docker for AWS/Azure

- Amazon Web Services
  - Amazon CloudFormation templates
  - Integrated with Autoscaling, ELB, and EBS
- Azure
  - Integrated with VM Scale Sets for autoscaling, Azure Load Balancer, Azure Storage
- [beta.docker.com](https://beta.docker.com) (restricted availability)

# Docker Toolbox

- Docker Engine
- Docker Machine
- Docker Compose
- Docker Kitematic
- Virtual box
- Quickstart Terminal





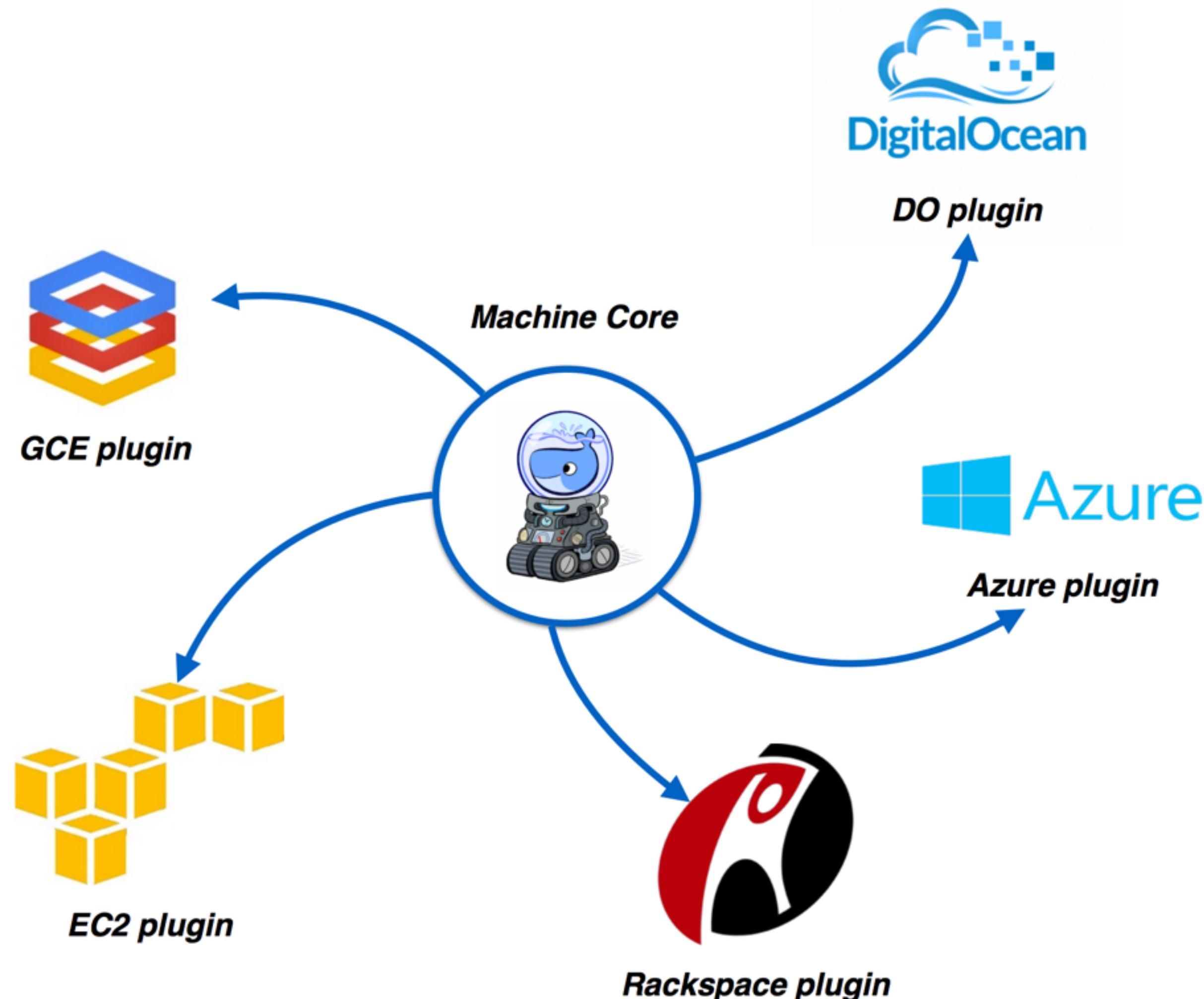
# Docker Machine

- 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 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 up -d`

# 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:  
    - 80:8080  
  
db-prod:  
  image: . . .
```

production.yml

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

# Docker Compose Common Use Cases

Use Case	Command
Dev Setup	<code>docker-compose up</code>
Local/remote host	<code>DOCKER_HOST</code> , <code>DOCKER_TLS_VERIFY</code> , <code>DOCKER_CERT_PATH</code>
Single/multiple hosts	Integrated with Swarm
Multiple isolated environments	<code>docker-compose up -p &lt;project&gt;</code>
Automated test setup	<code>docker-compose up</code> <code>mvn test</code>
Dev/Prod Impedance mismatch	<code>docker-compose down</code> <code>docker-compose up -f docker-compose.yml -f production.yml</code>



# Swarm Mode

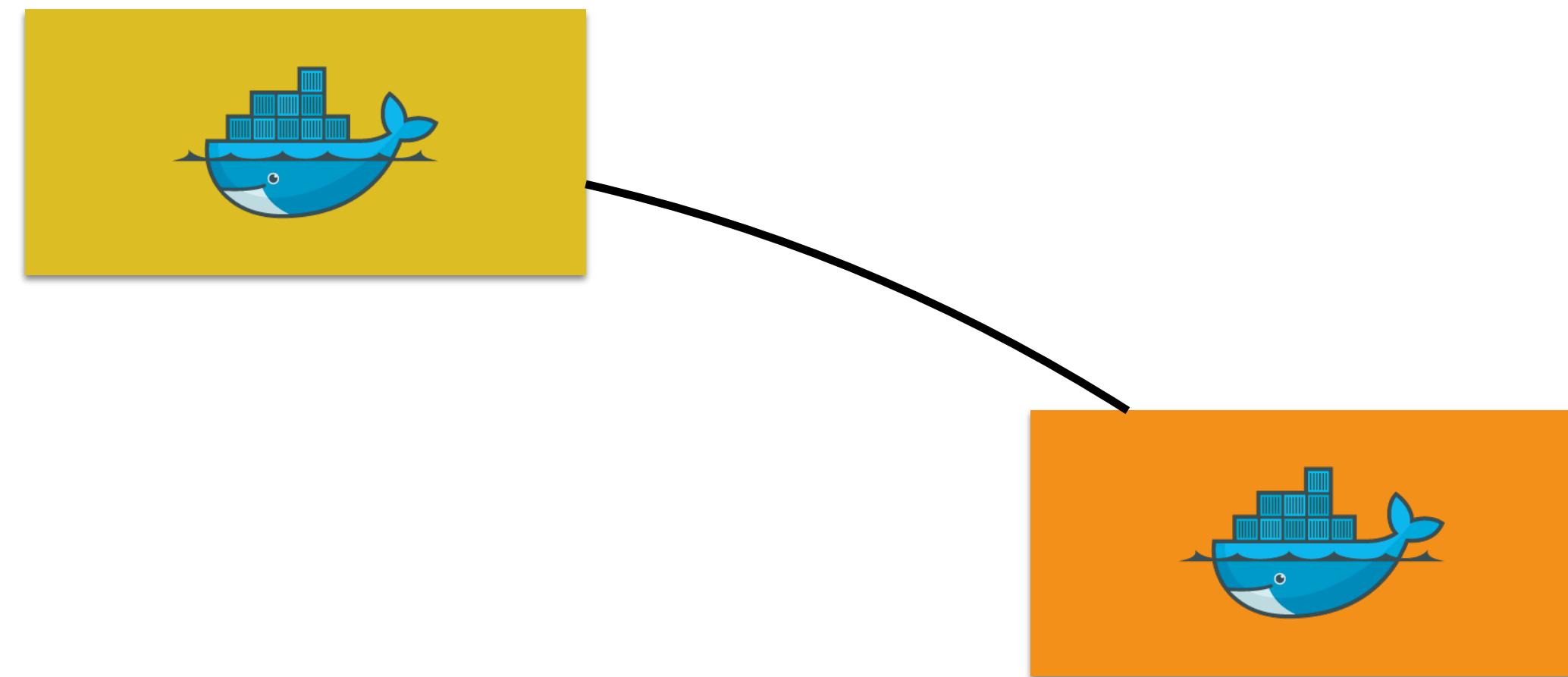
- New in 1.12
- Natively managing a cluster of Docker Engines called a Swarm
- Docker CLI to create a swarm, deploy apps, and manage swarm
- No Single Point of Failure (SPOF)
- Declarative state model
- Self-organizing, self-healing
- Service discovery, load balancing and scaling
- Rolling updates
- Optional feature, need to be explicitly enabled

# Swarm Mode: Initialize



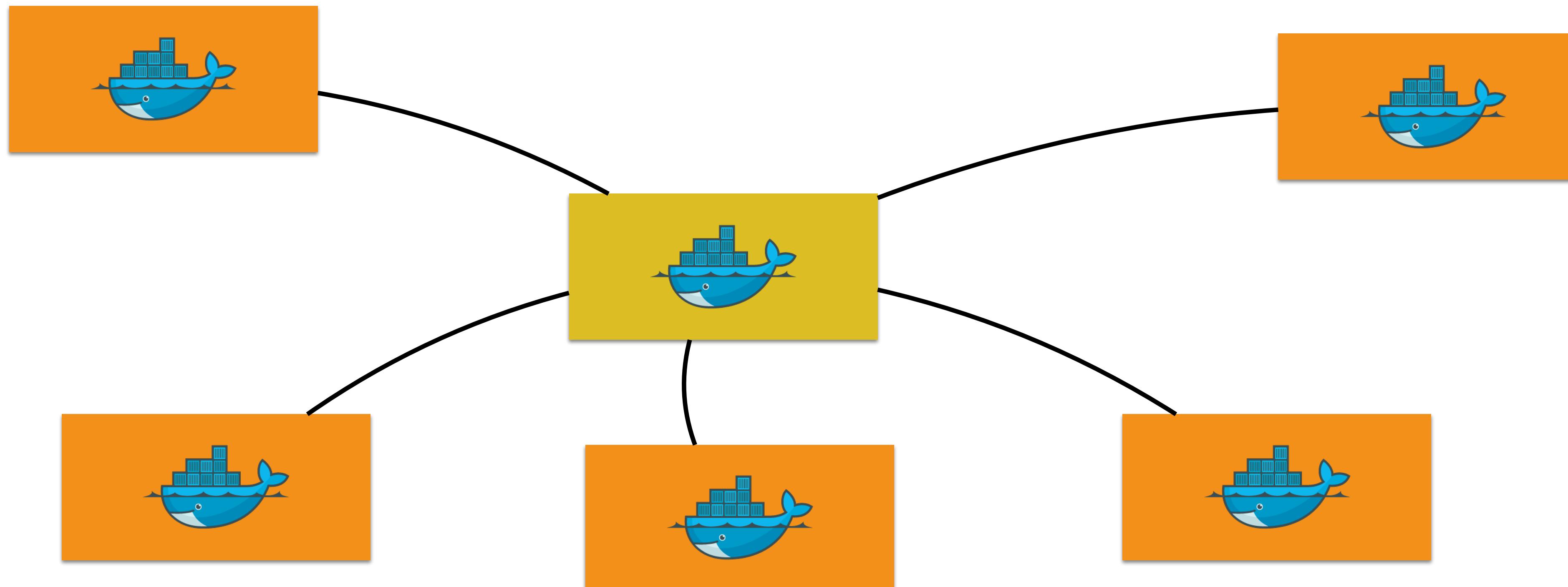
```
docker swarm init --listen-addr <ip>:2377
```

# Swarm Mode: Add Worker



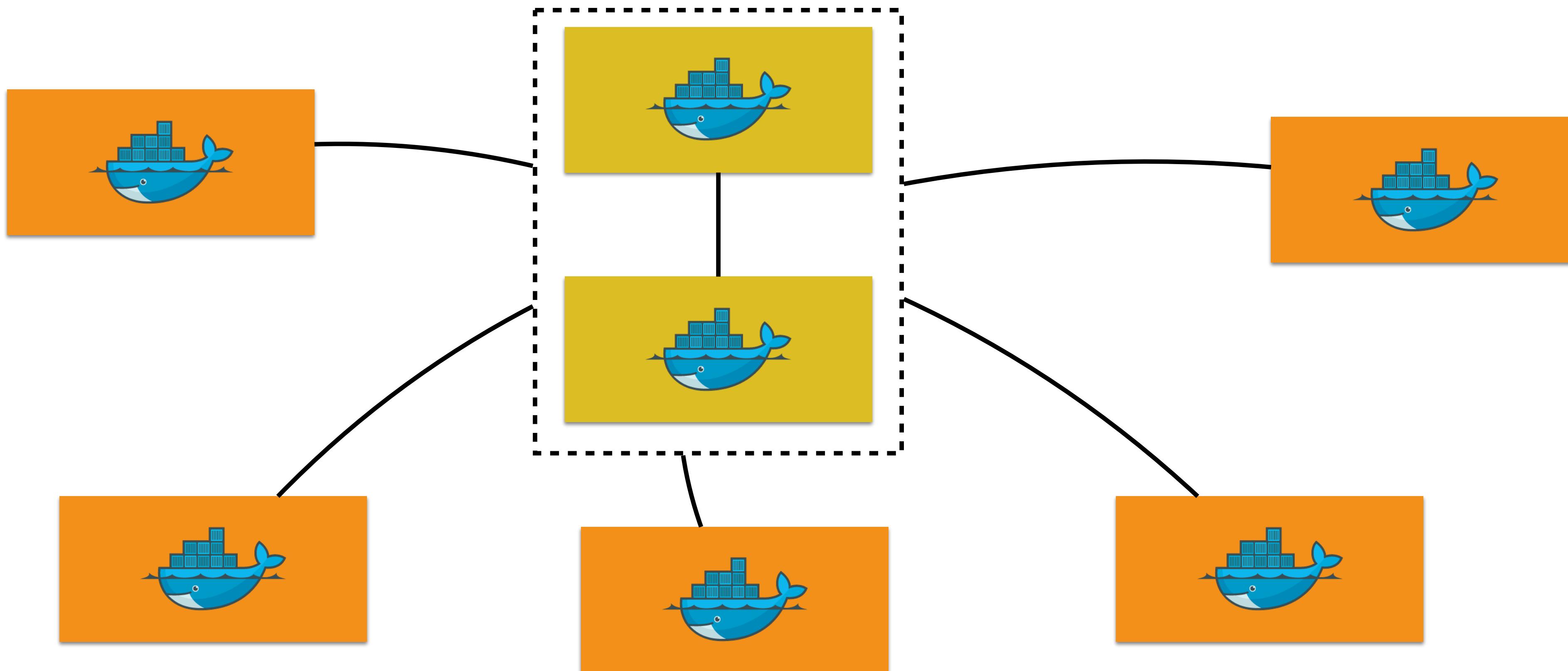
```
docker swarm join --token <worker_token> <manager>:2377
```

# Swarm Mode: Add More Workers



```
docker swarm join --token <worker_token> <manager>:2377
```

# Swarm Mode: Primary/Secondary Master

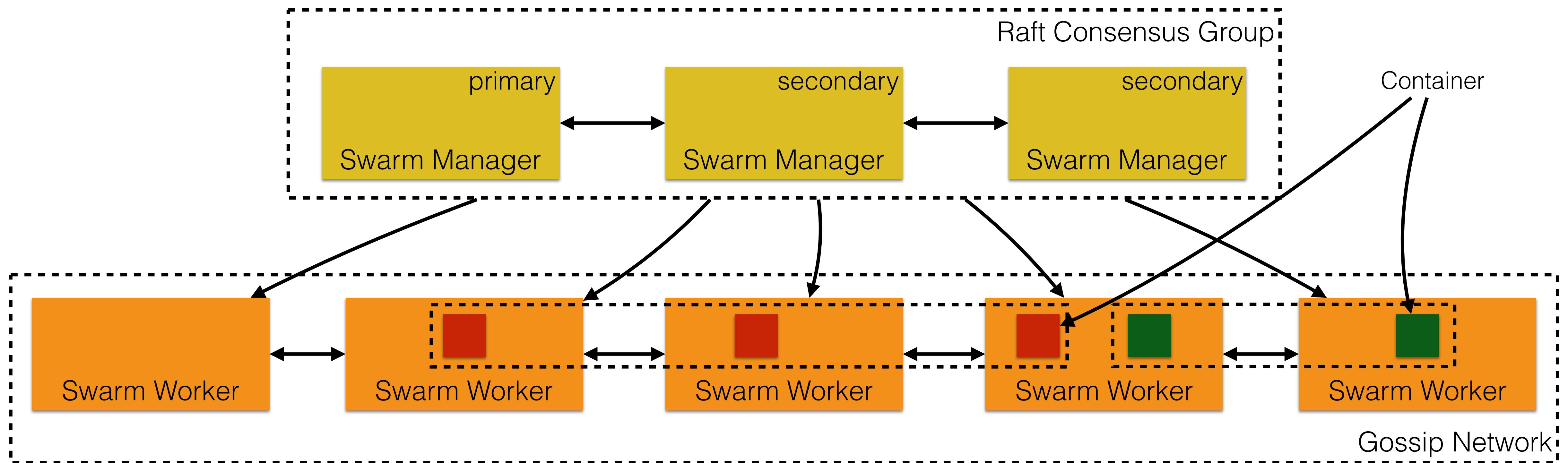


```
docker swarm join --manager --token <manager_token> --listen-  
addr <master2>:2377 <master1>:2377
```

# Swarm Mode using Docker Machine

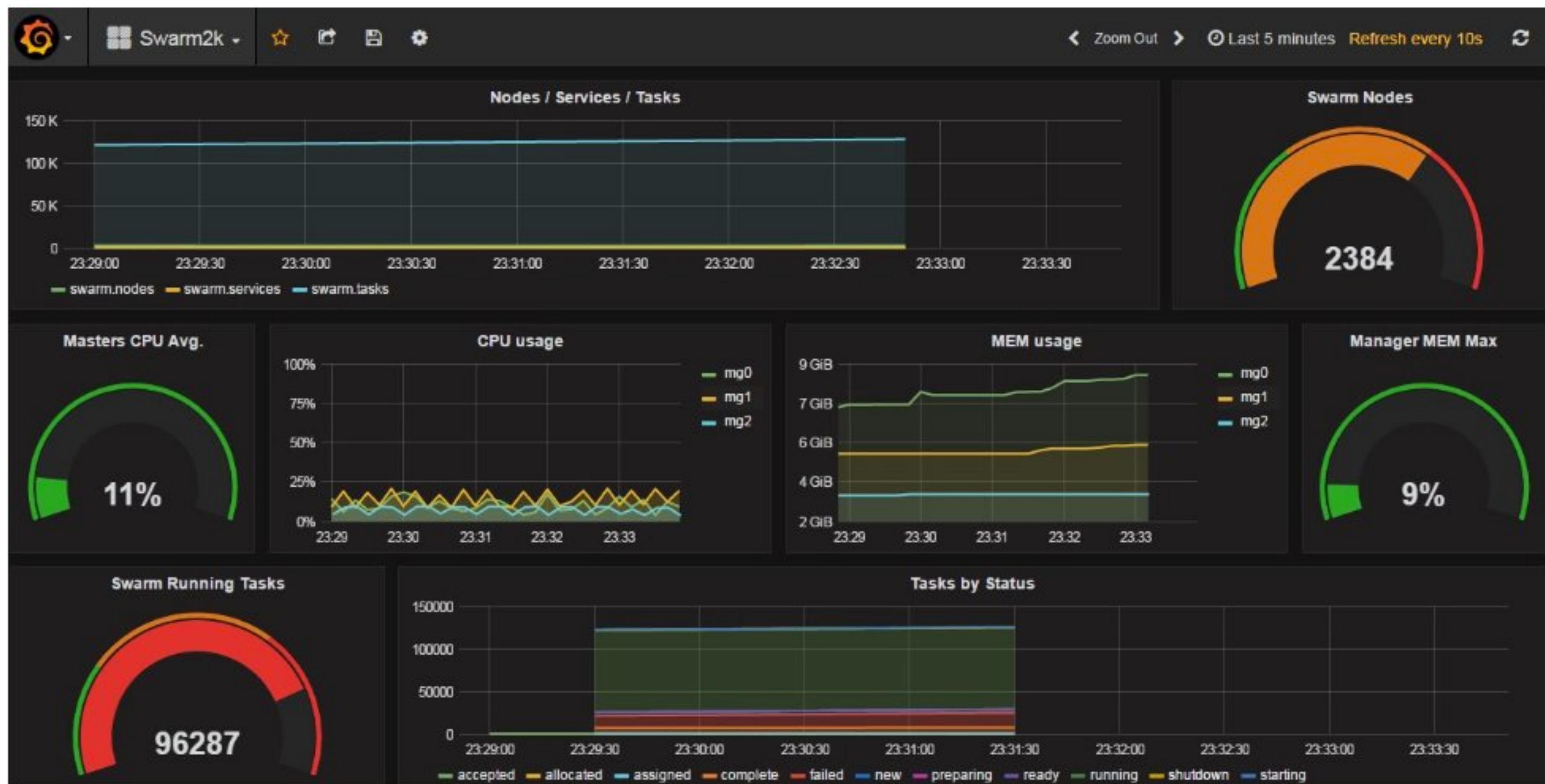
Task	Command
Create manager	<code>docker-machine create -d virtualbox managerX</code>
Create worker	<code>docker-machine create -d virtualbox workerX</code>
Initialize Swarm mode	<code>docker swarm init --listen-addr &lt;ip1&gt; --advertise-addr &lt;ip1&gt;</code>
Manager token	<code>docker swarm join-token manager -q</code>
Worker token	<code>docker swarm join-token worker -q</code>
Manager X join	<code>docker swarm join --token manager_token --listen-addr &lt;ipX&gt; --advertise-addr &lt;ipX&gt; &lt;ip1&gt;</code>
Worker X join	<code>docker swarm join --token worker_token --listen-addr &lt;ipX&gt; --advertise-add &lt;ipX&gt; &lt;ip1&gt;</code>

# Swarm Mode: Protocols

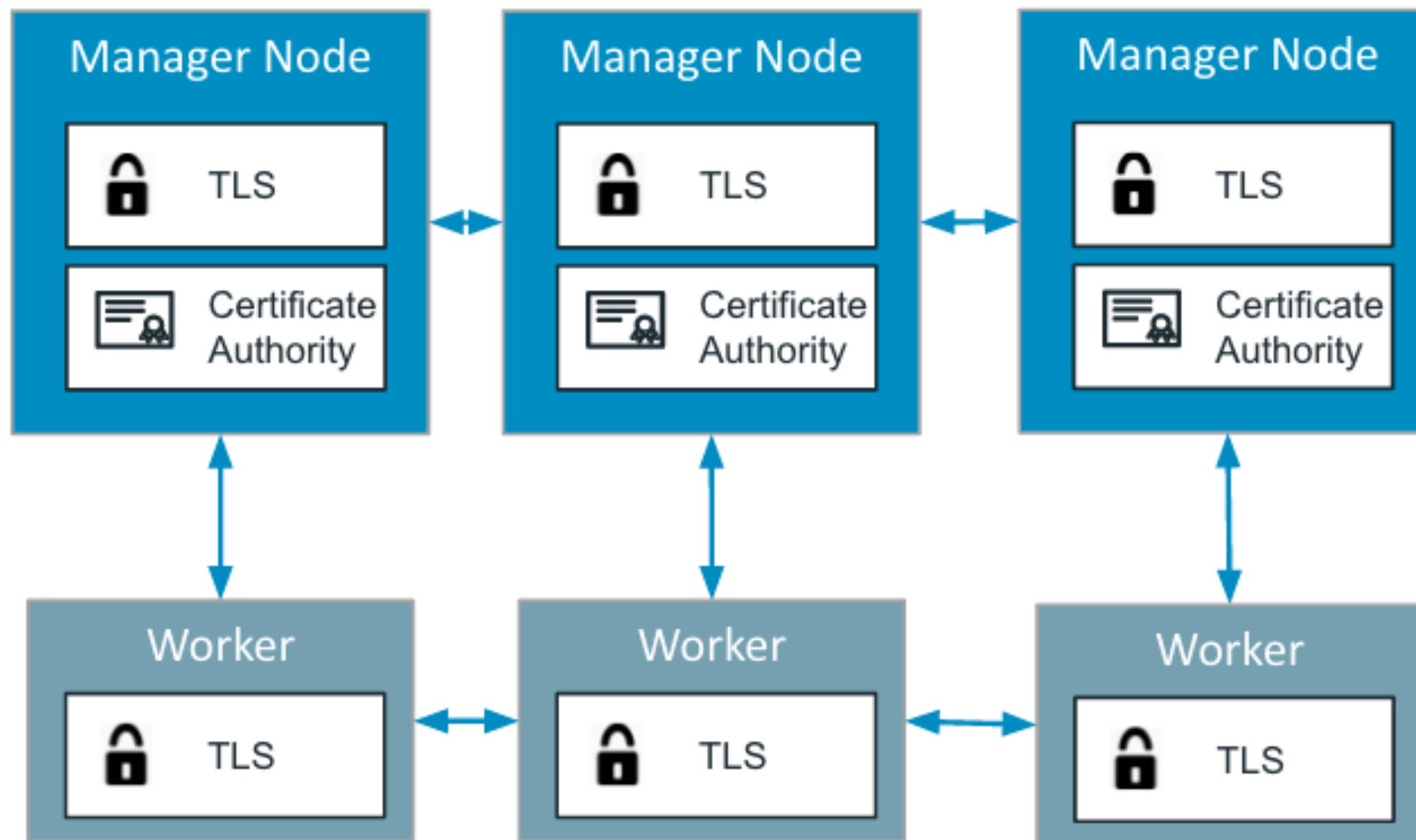


Strongly consistent  
Replicated (Raft based)  
Extremely fast (in-memory reads)

# Swarm Mode in Production

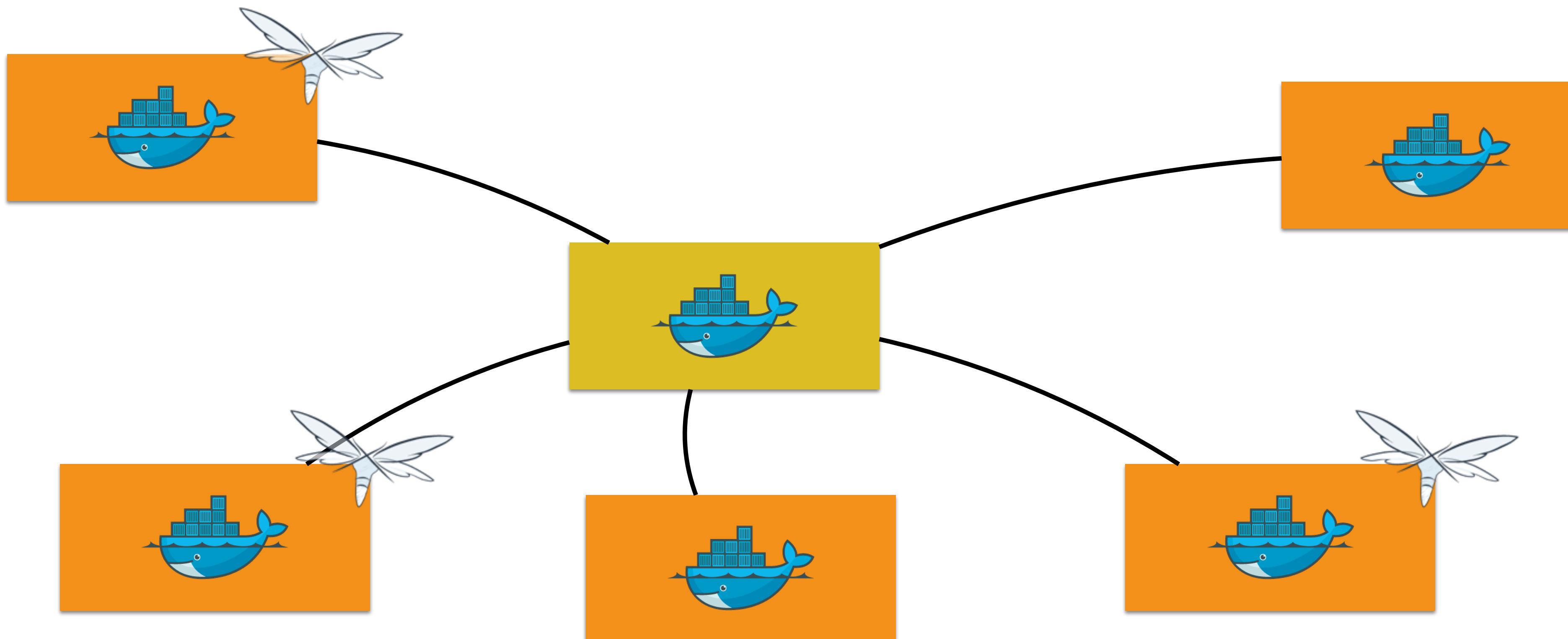


# Secure by Default



- Cryptographic node identity
- Automatic encryption and mutual authentication (TLS)
- Automatic cert rotation (90 days, can be up to 30 mins)
- External CA integration

# Swarm Mode: Replicated Service

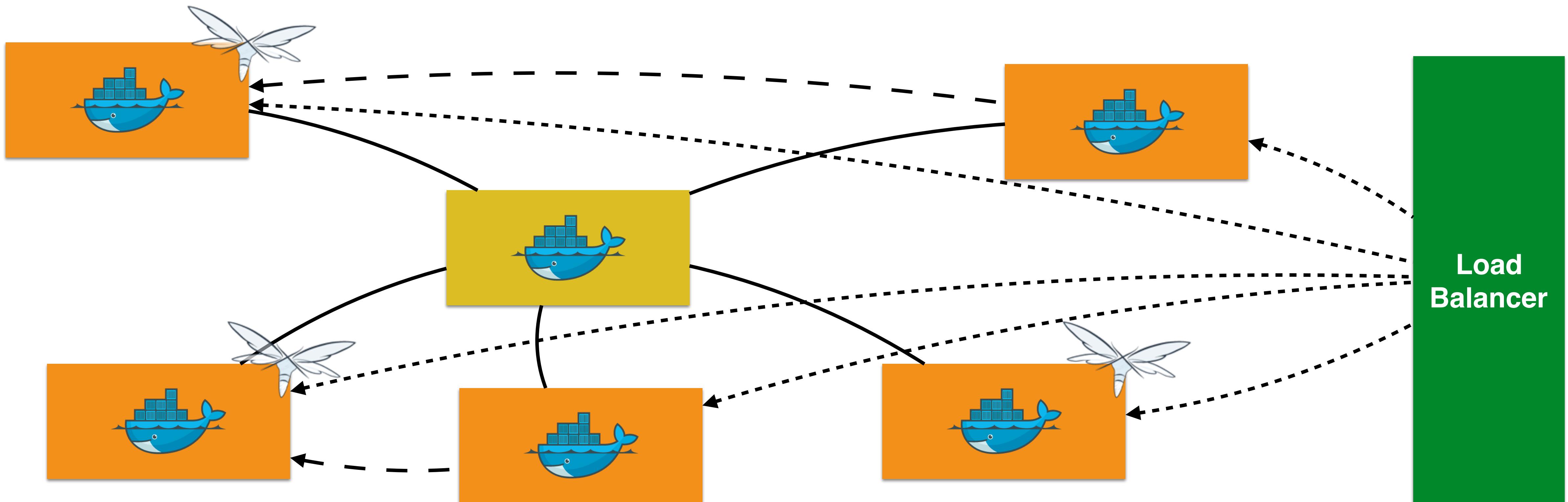


```
docker service create --replicas 3 --name web jboss/wildfly
```

# Swarm Mode - Routing Mesh

- Load balancers are host-aware, not container-aware
- Swarm mode introduces container-aware routing mesh
- Reroutes traffic from any host to a container
  - Reserves a Swarm-wide ingress port
  - Uses DNS-based service discovery

# Swarm Mode: Routing Mesh



```
docker service create --replicas 3 --name web -p 8080:8080 jboss/wildfly
```

# Why NGINX Load Balancer?

- SSL Termination
- Content-based routing
- Access control and authorization
- Rewrites and redirects
- Load Balancing Algorithms
- Multiprotocol support - HTTP/2, WebSockets
- Advanced logging

# Load Balancing using NGINX

```
version: "2"
services:
  web:
    image: arungupta/wildfly-app
  lb:
    image: nginx
  ports:
    - 80:80
    - 8080:8080
```

```
upstream wildfly {
  server IP1:8080;
  server IP2:8080;
  server IP3:8080;
}

server {
  listen 8080;
  location / {
    proxy_pass http://wildfly;
  }
}
```

# Load Balancing using NGINX

```
upstream wildfly {  
    least_conn;  
  
    server IP1:8080;  
    server IP2:8080;  
    server IP3:8080;  
}
```

Least  
number of  
active  
connections

```
upstream wildfly {  
    ip_hash;  
  
    server IP1:8080;  
    server IP2:8080;  
    server IP3:8080;  
}
```

Sticky  
Sessions

```
upstream wildfly {  
    server IP1:8080 weight=2;  
    server IP2:8080;  
    server IP3:8080;  
}
```

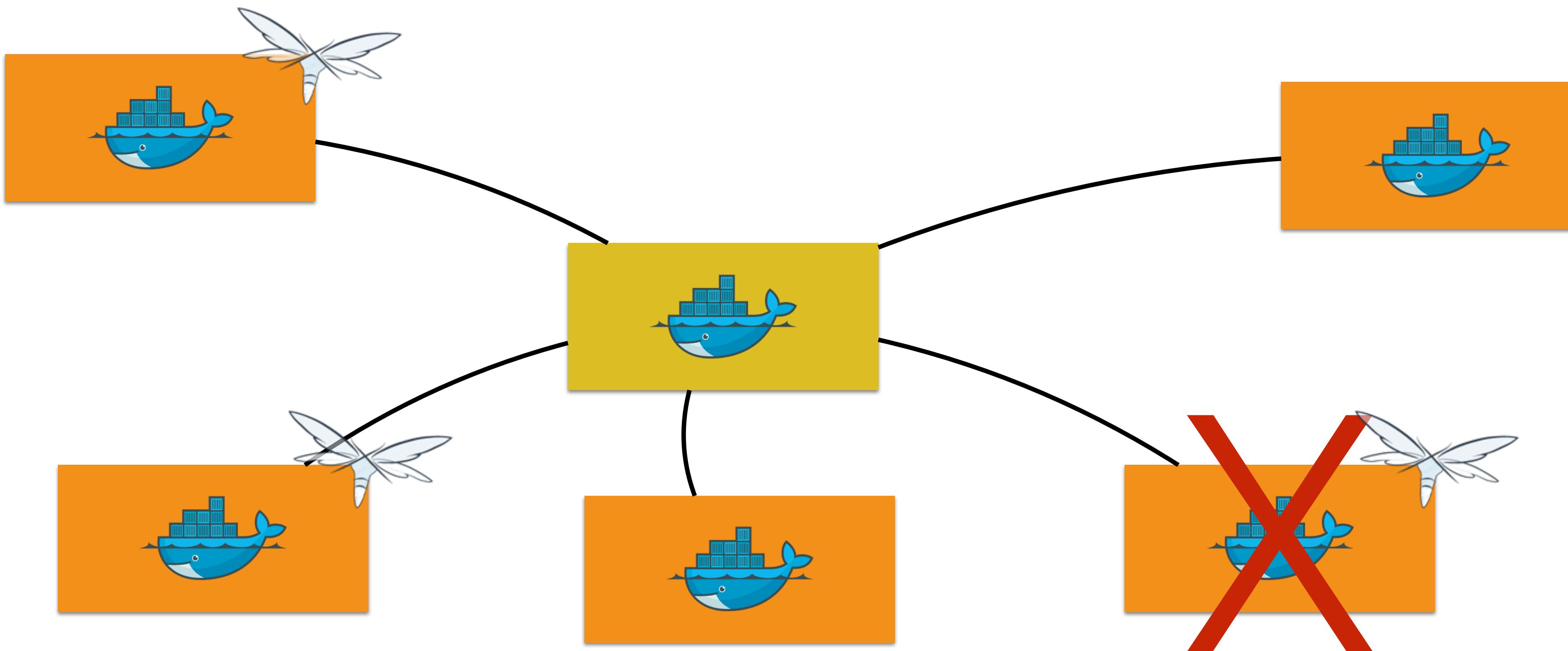
Weighted  
Load  
Balancing

# Load Balancing a Swarm Service

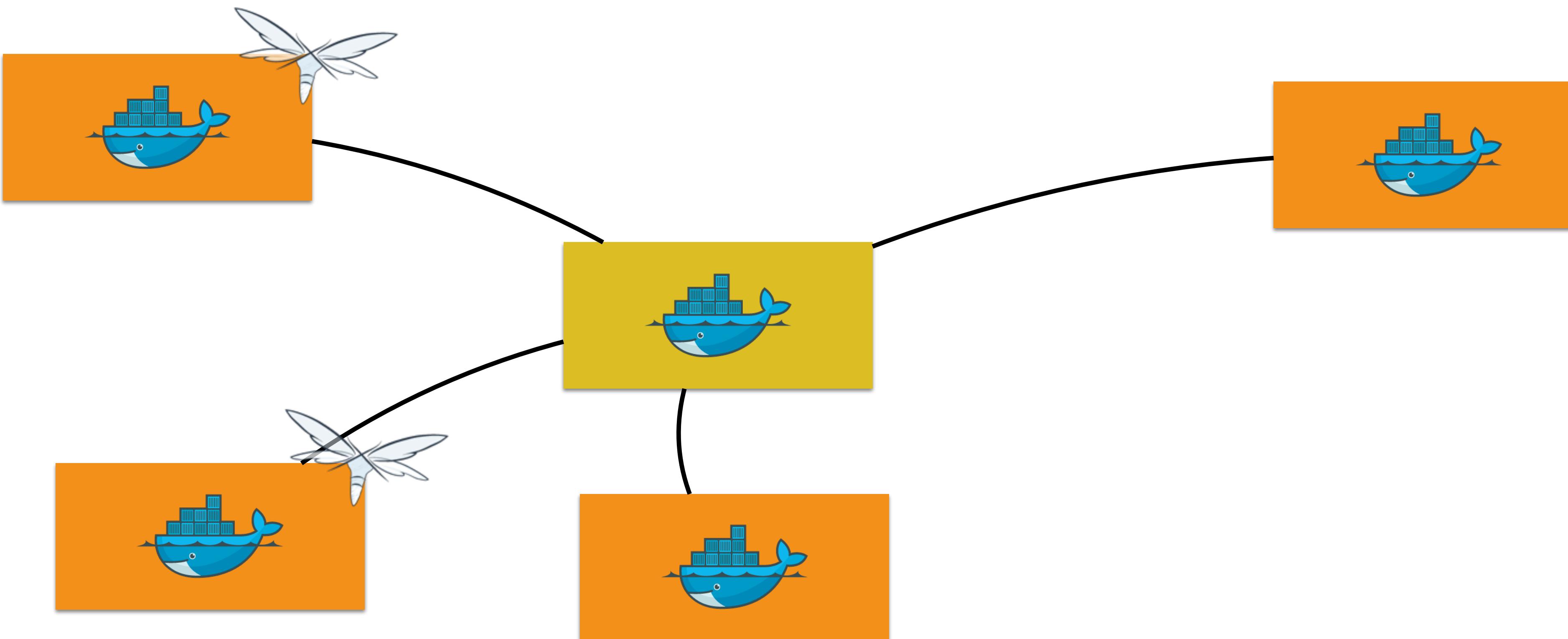
```
upstream wildfly {
    server backend-service;
}

server {
    listen 8080;
    location / {
        proxy_pass http://wildfly;
    }
}
```

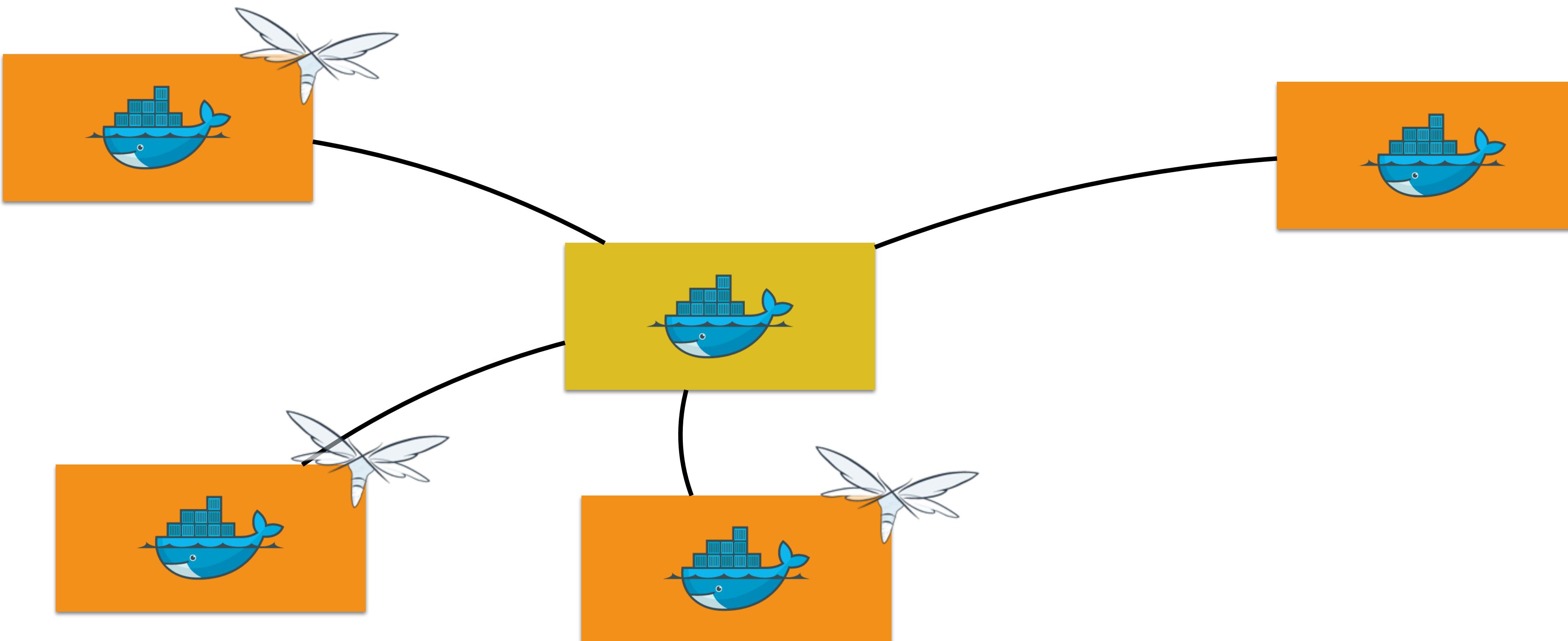
# Swarm Mode: Node Failure



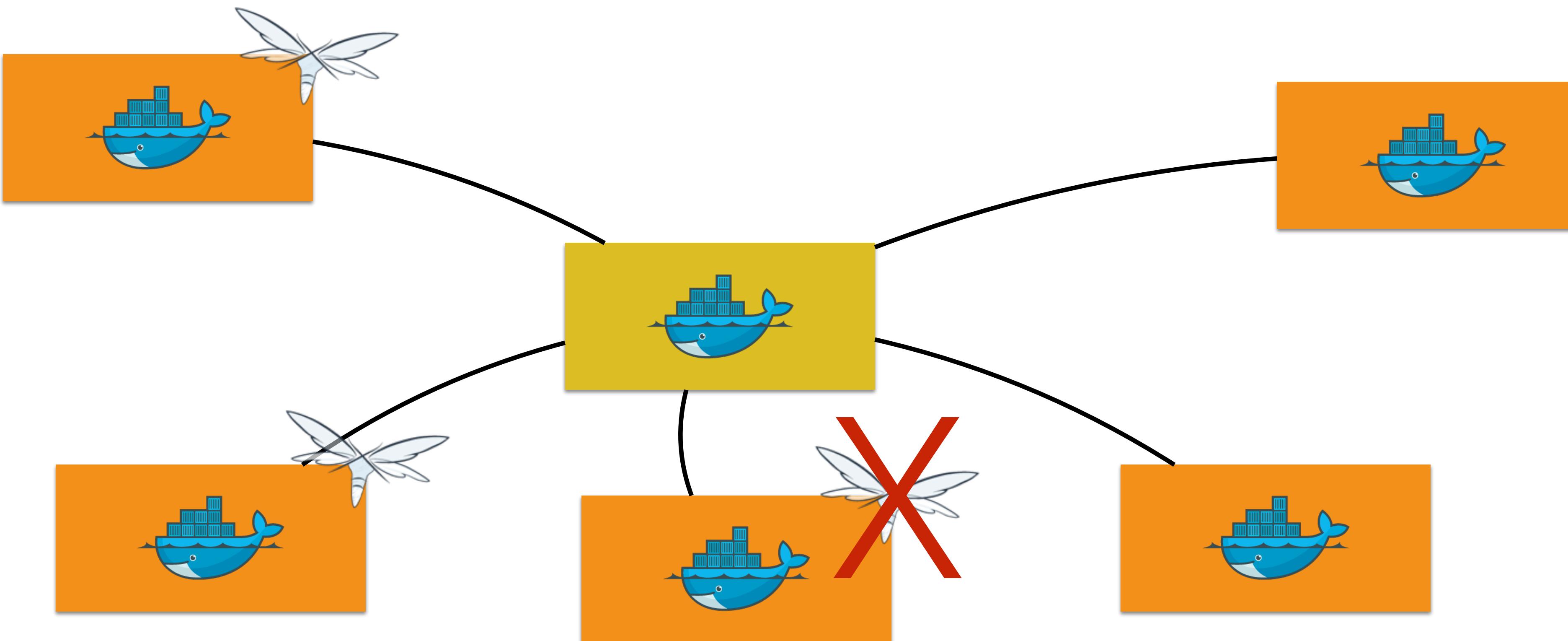
# Swarm Mode: Desired != Actual



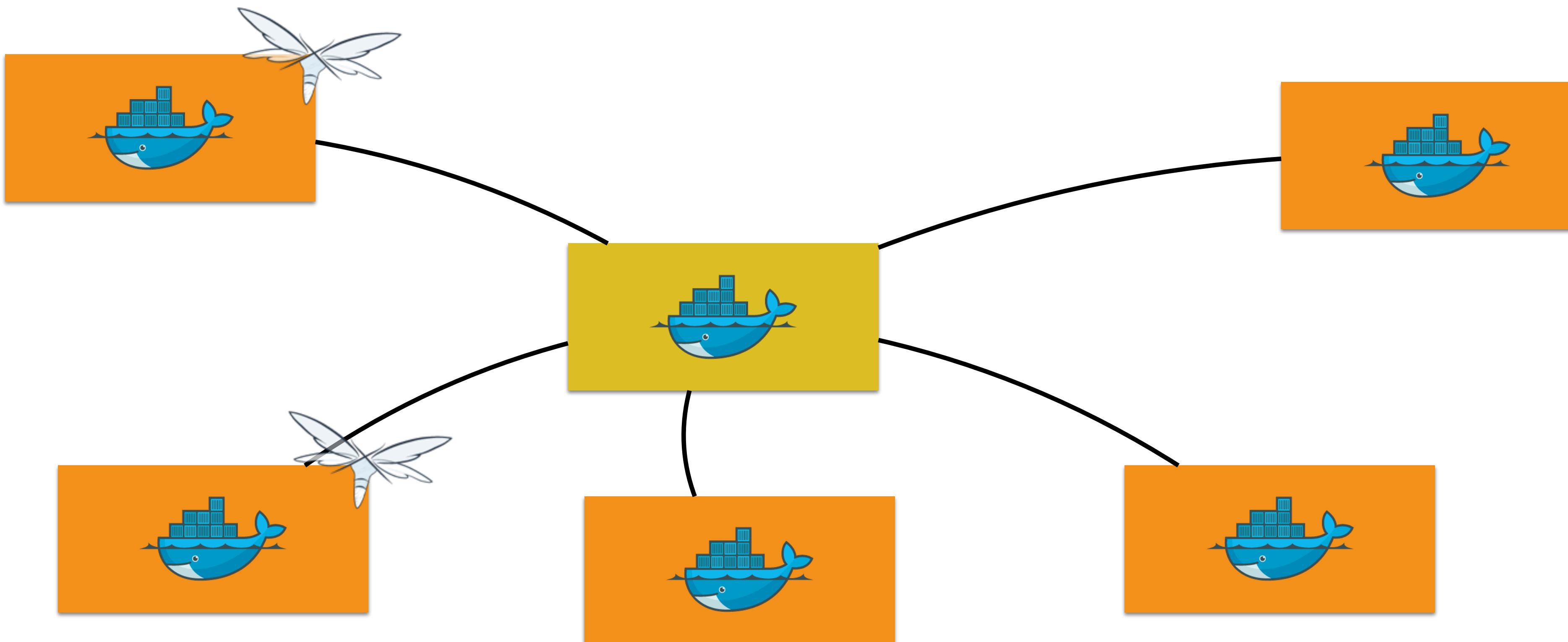
# Swarm Mode: Reconcile



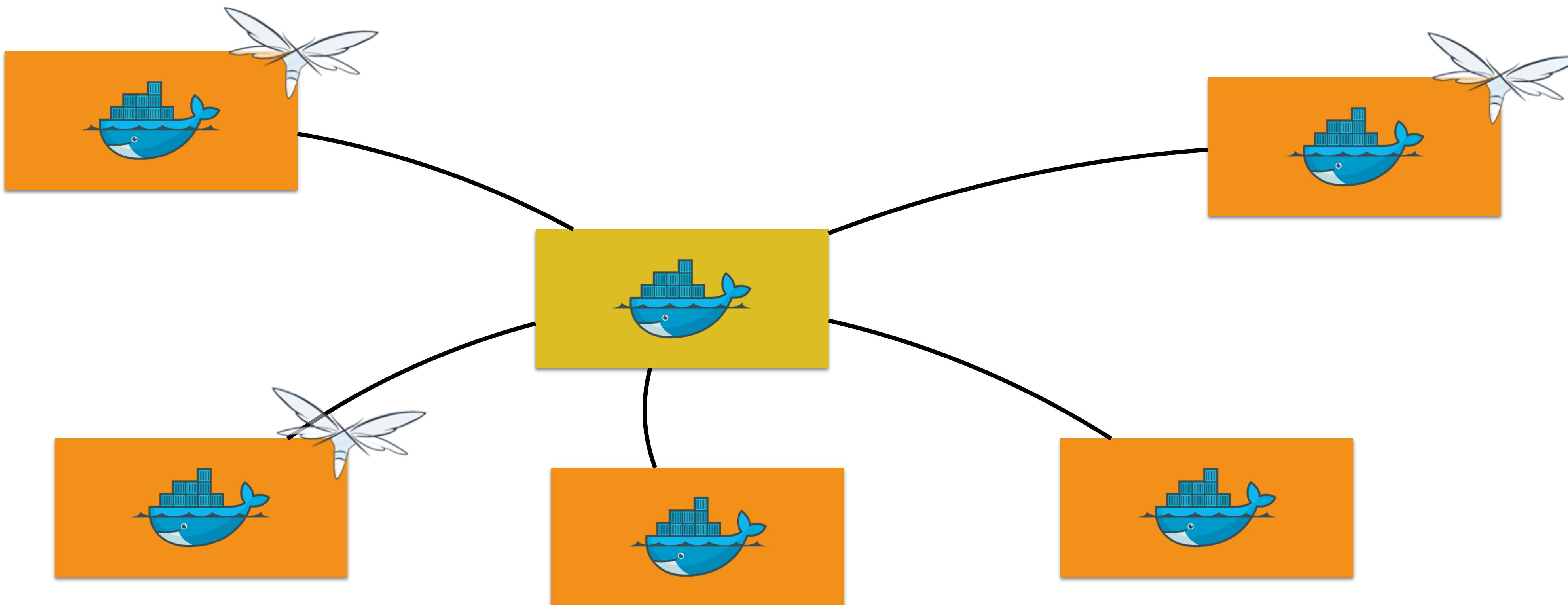
# Swarm Mode: Container Failure



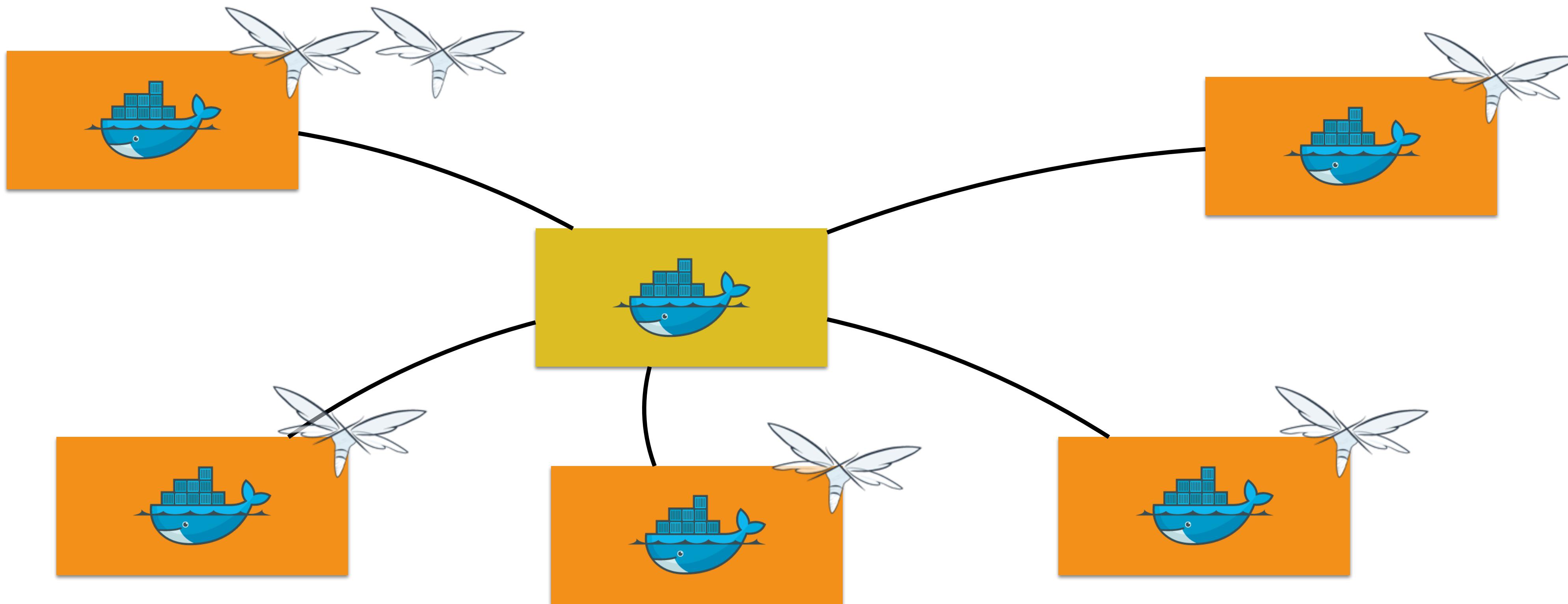
# Swarm Mode: Desired != Actual



# Swarm Mode: Reconcile

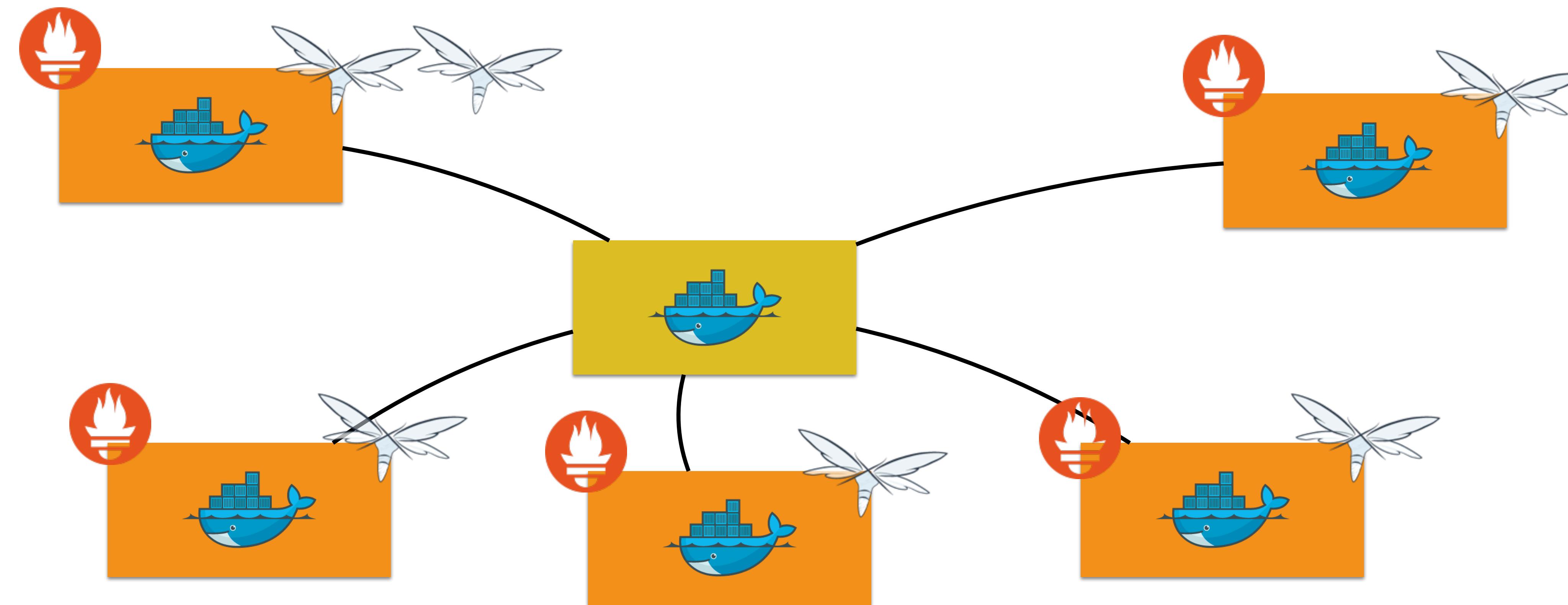


# Swarm Mode: Scale



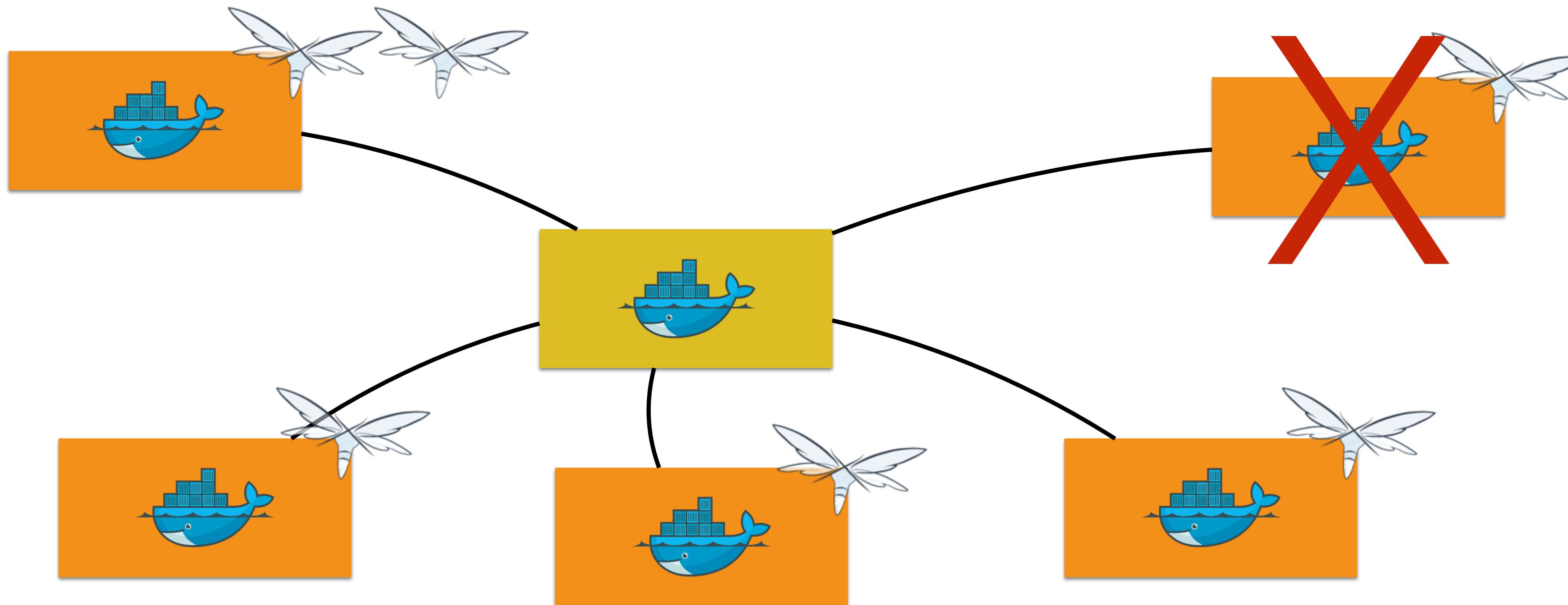
`docker service scale web=6`

# Swarm Mode: Global Service



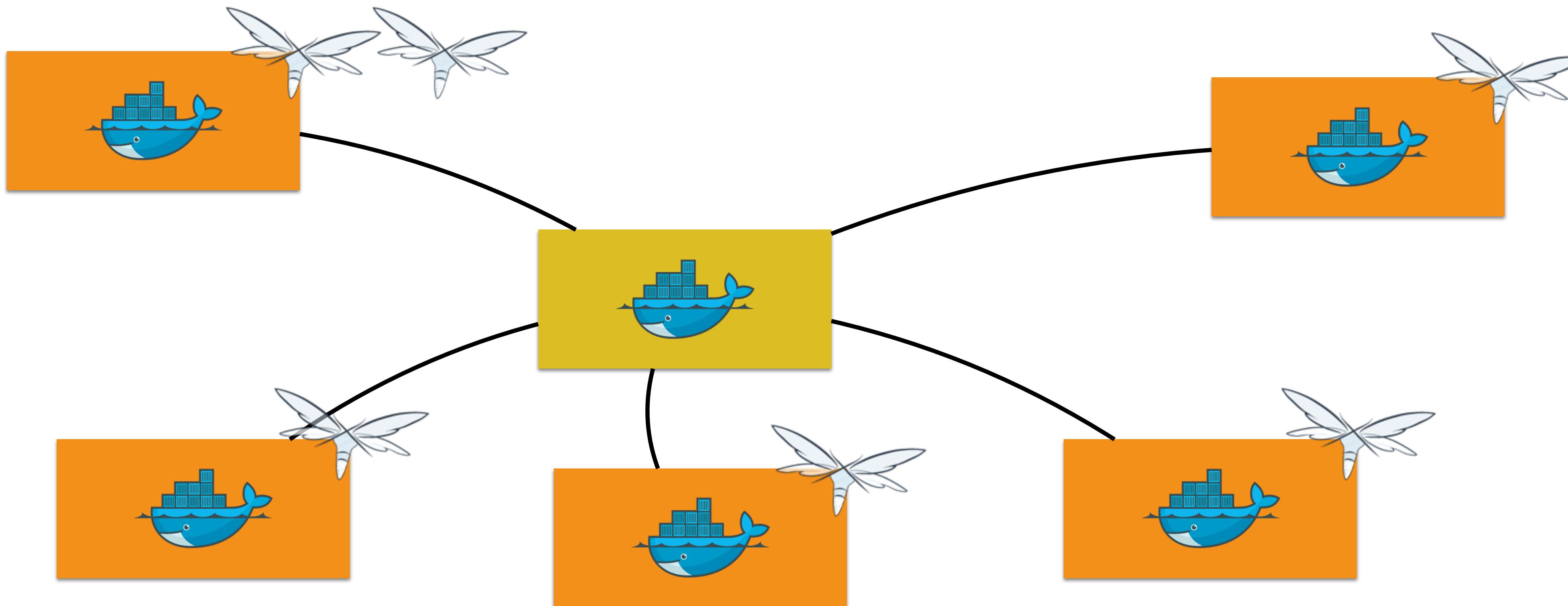
```
docker service create --mode=global --name=prom prom/prometheus
```

# Swarm Mode: Pause Node



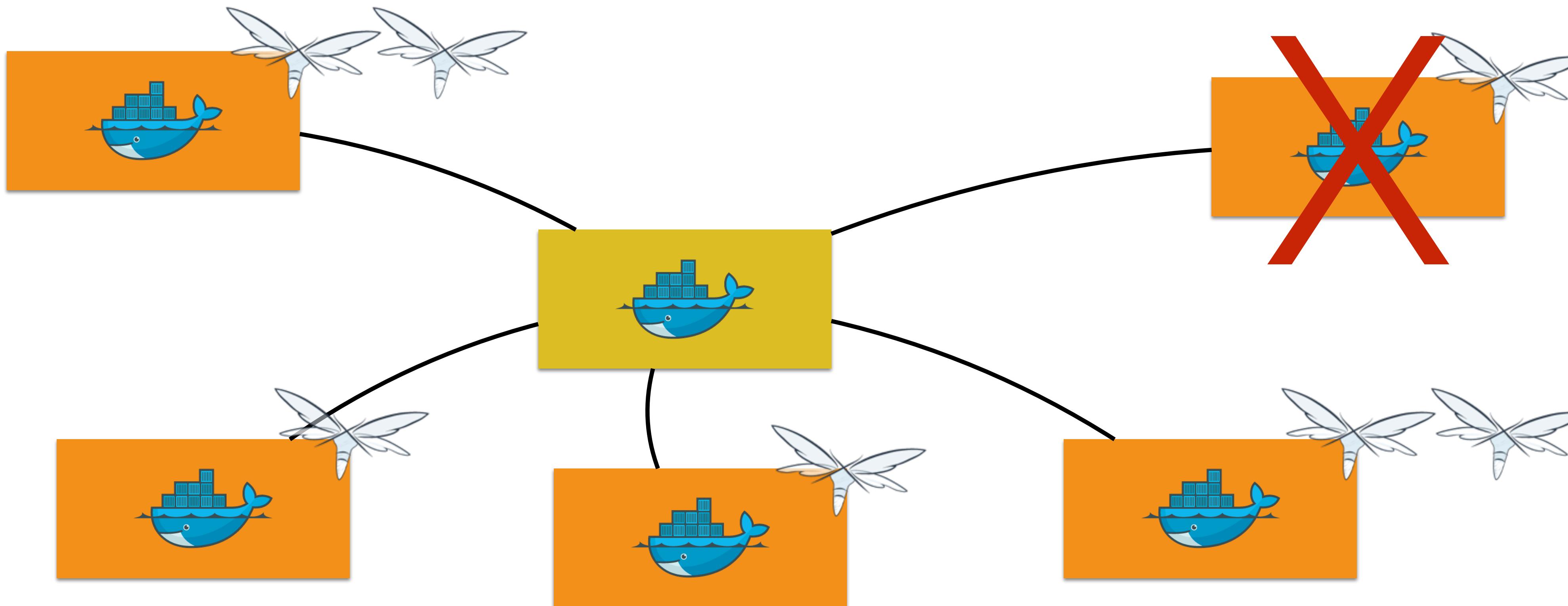
```
docker node update --availability pause <nodename>
```

# Swarm Mode: Active Node



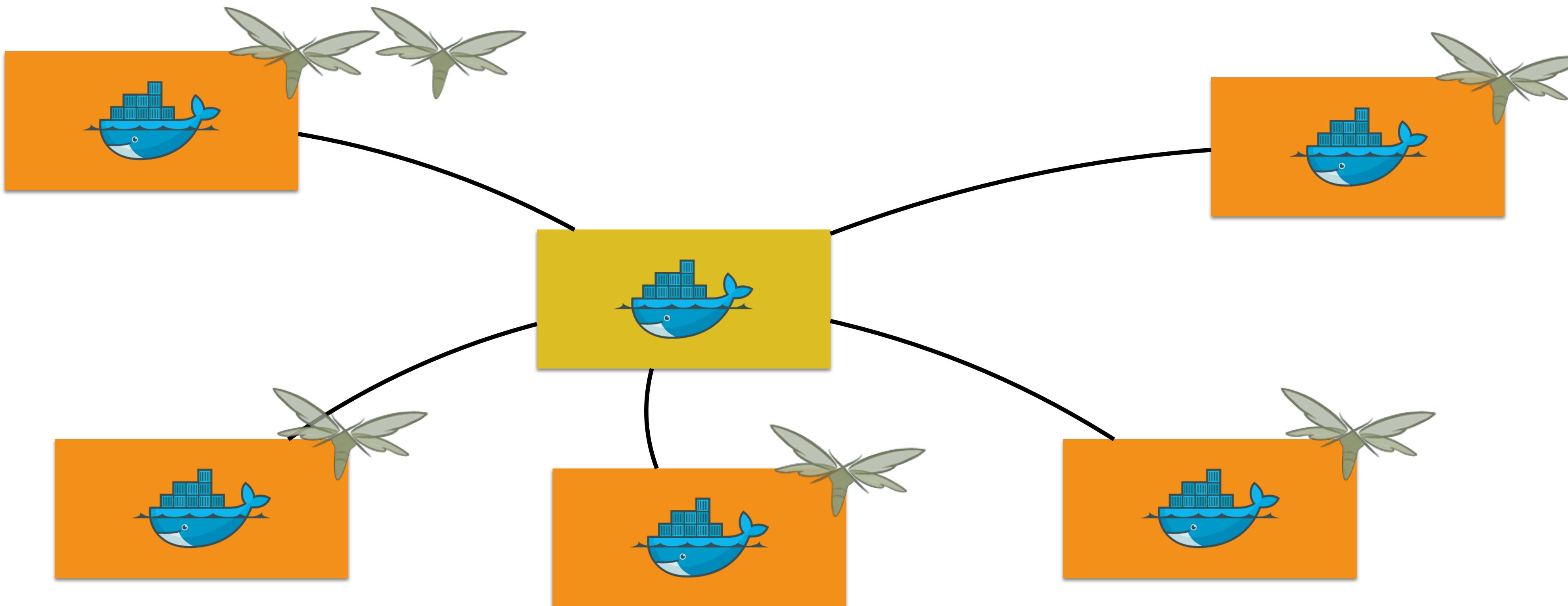
```
docker node update --availability active <nodename>
```

# Swarm Mode: Drain Node



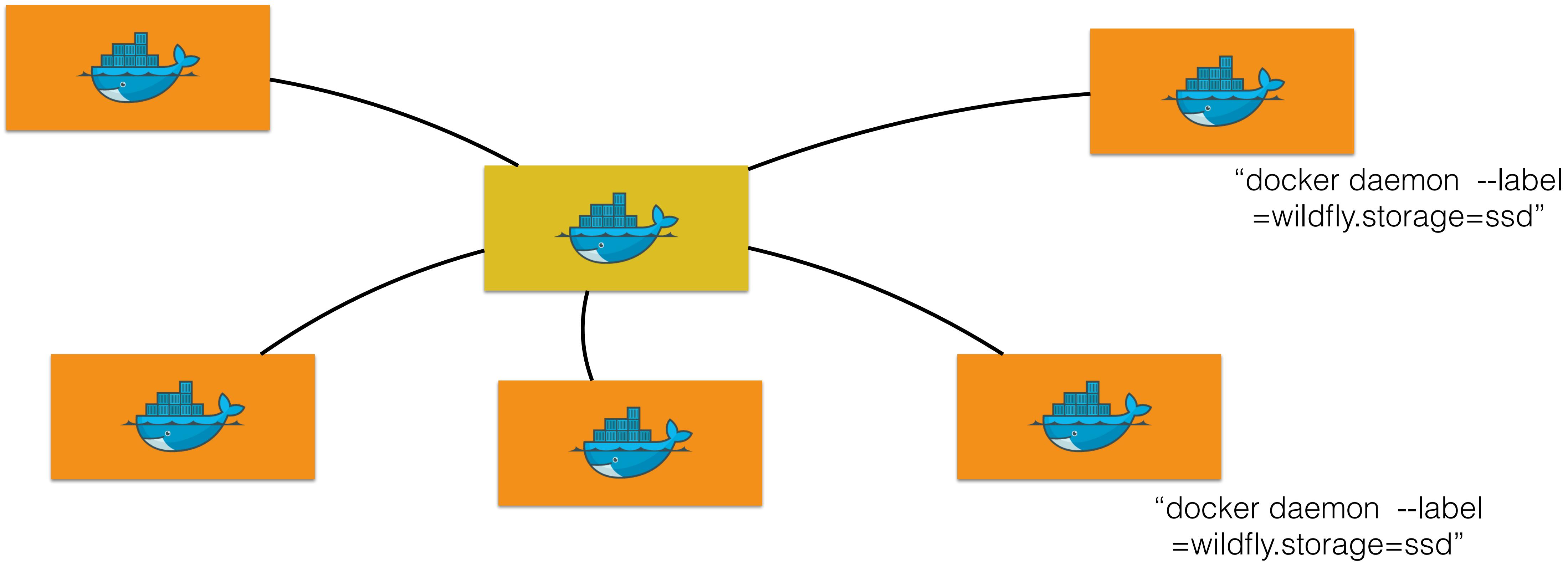
```
docker node update --availability drain <nodename>
```

# Swarm Mode: Rolling Updates



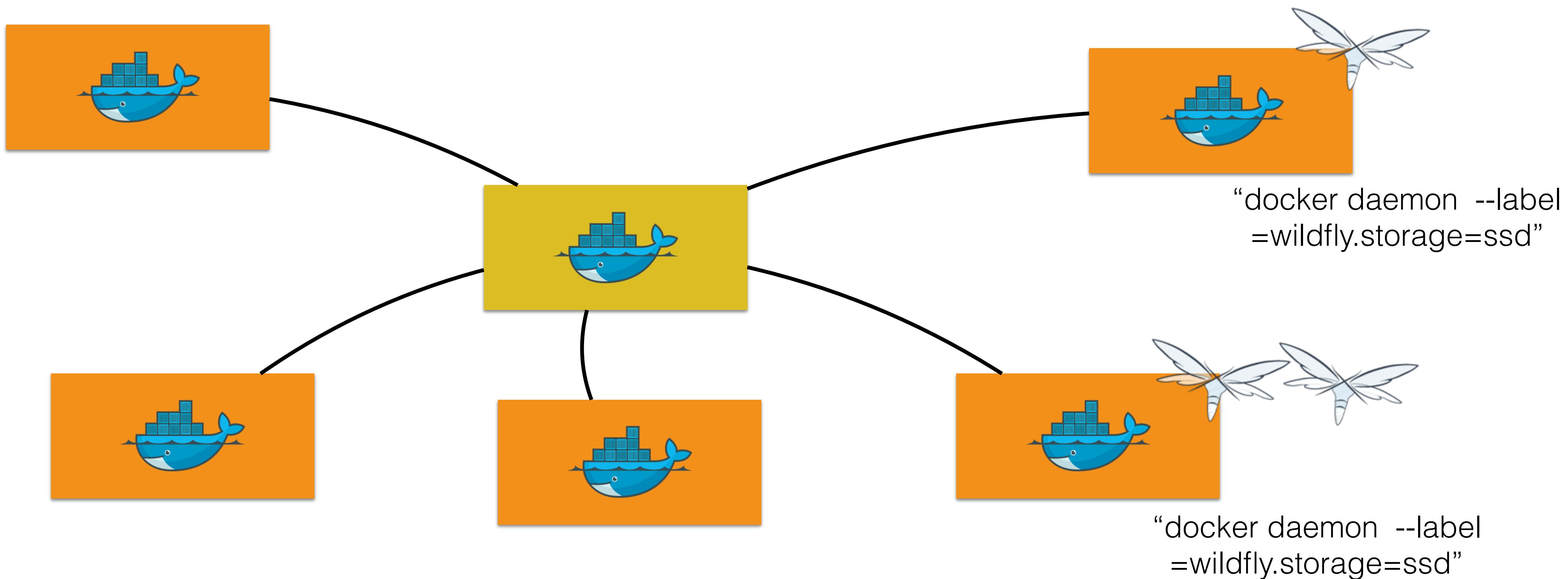
```
docker service update web --image wildfly:2 --update-parallelism  
2 --update-delay 10s
```

# Swarm Mode: Label



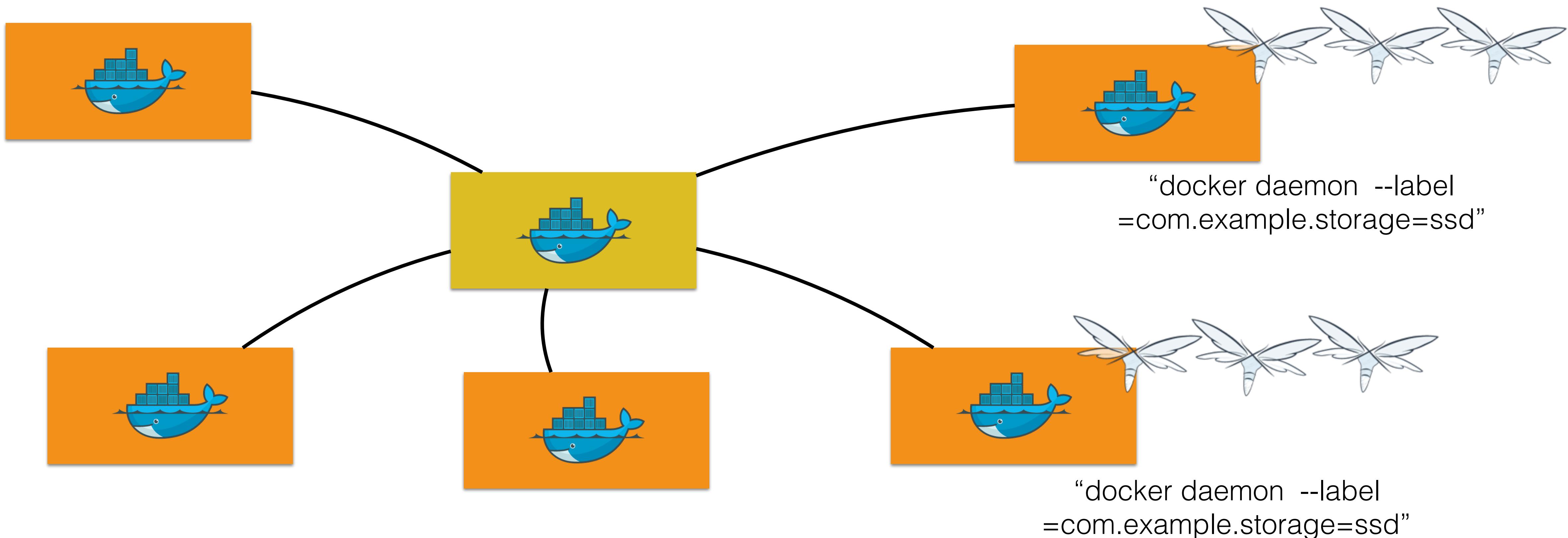
```
DOCKER_OPTS="--label=wildfly.storage=ssd"
```

# Swarm Mode: Constraints



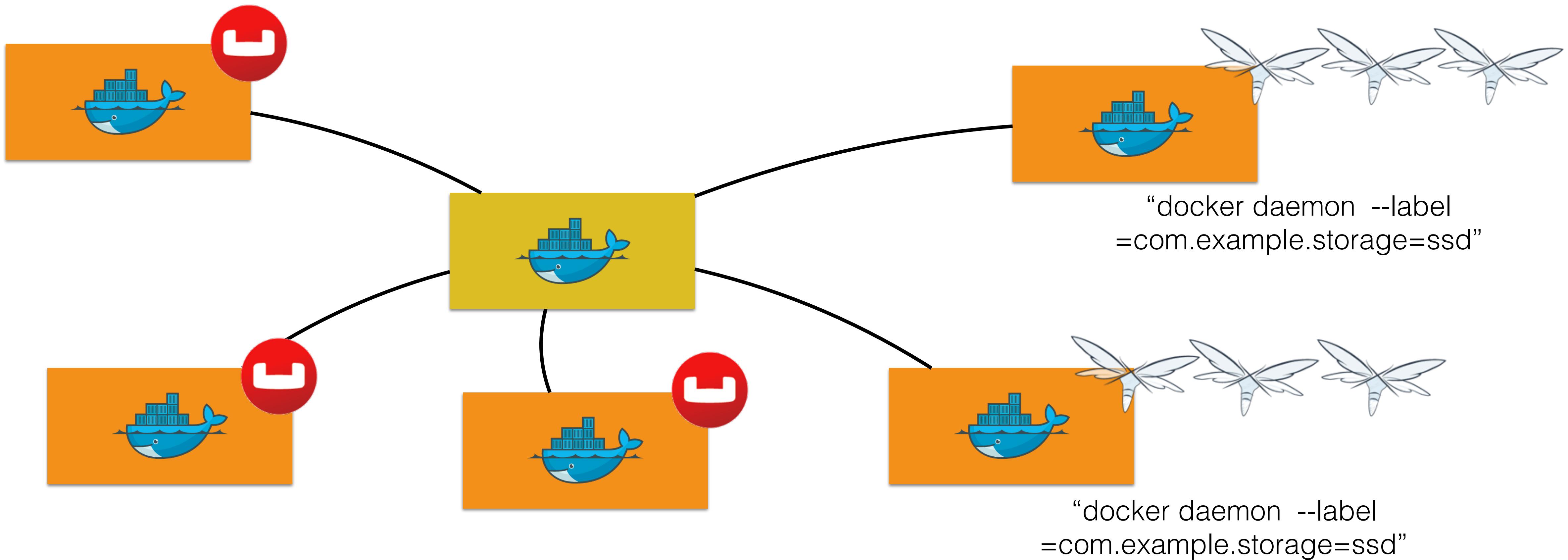
```
docker service create --replicas=3 --name=web --constraint  
engine.labels.wildfly.storage==ssd jboss/wildfly
```

# Swarm Mode: Constraints

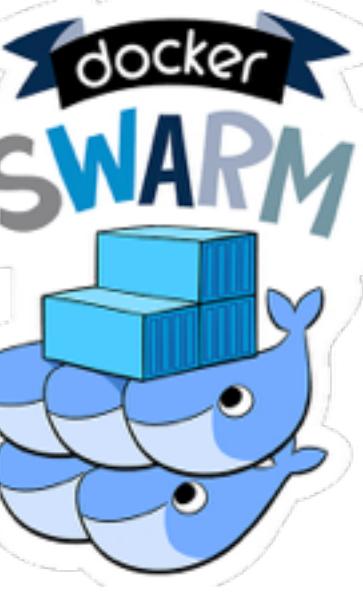


```
docker service scale web=6
```

# Swarm Mode: Constraints



```
docker service create --replicas=3 --name=db couchbase
```



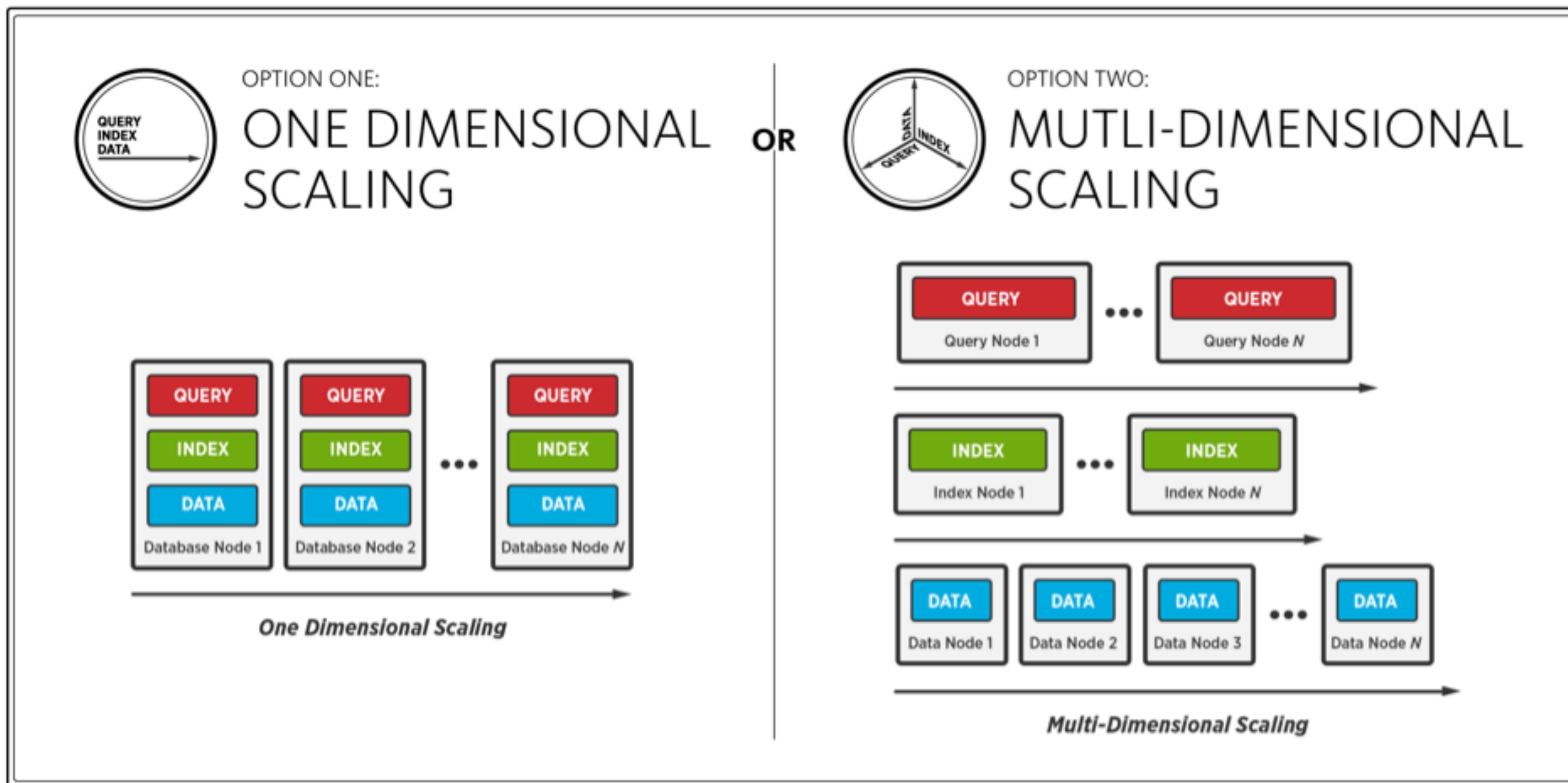
# Scheduling Backends using Filters

- **Label:** Metadata attached to Docker Daemon
- **Filters:** Used by Docker Swarm scheduler to create and run container

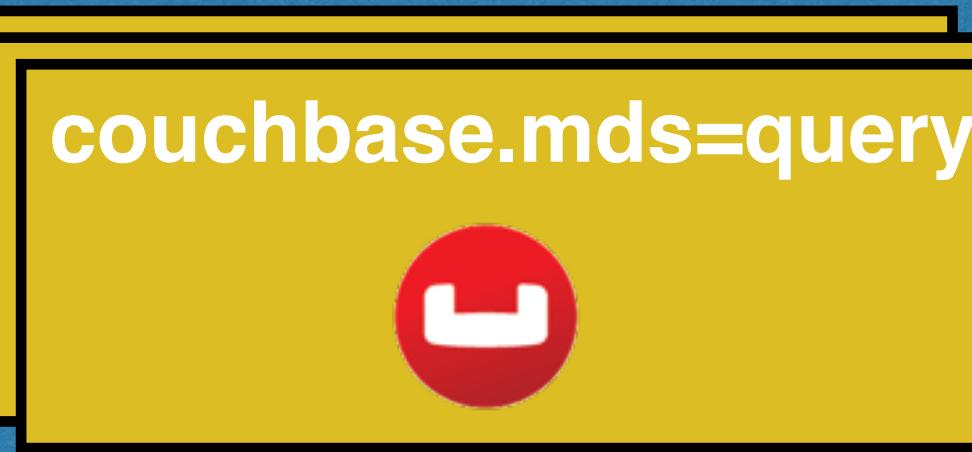
Node		
Constraint	Default or custom tags	node, operatingsystem, kernelversion, ...
Health	Schedule containers on healthy nodes only	
Container Slots	Maximum number of containers on a node	--labels containerslots=3
Container		
Affinity	“Attraction” between containers	-e affinity:container=<name>/<id>, image, ...
Dependency	Dependent containers on same node	--volumes-from=<id>, --net=container:<id>, ...
Port	Port availability on the host	-p <host>:<container>

# Couchbase Multi Dimensional Scaling

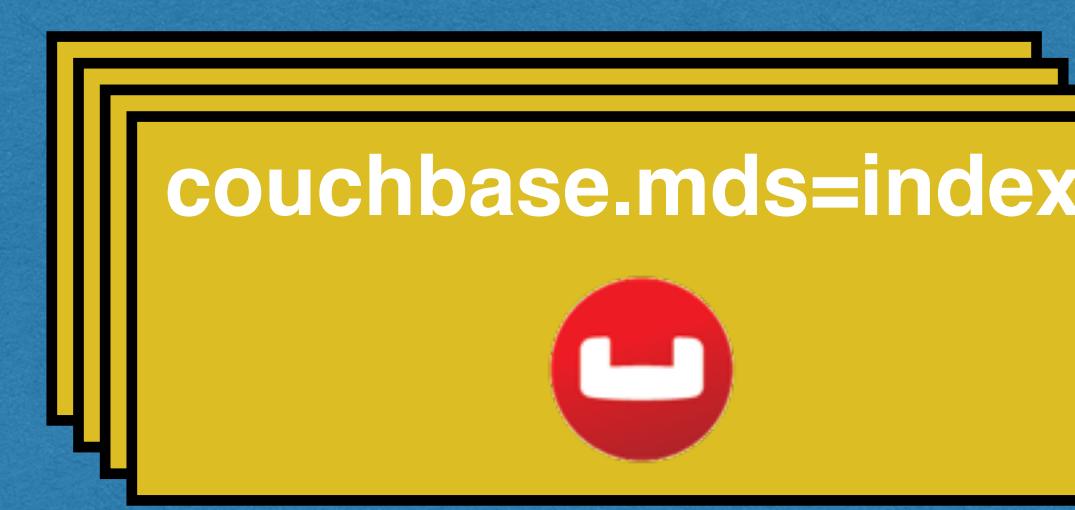
Only Couchbase gives customers two options for scaling: Standard One-Dimensional Scaling and New Multi-Dimensional Scaling.



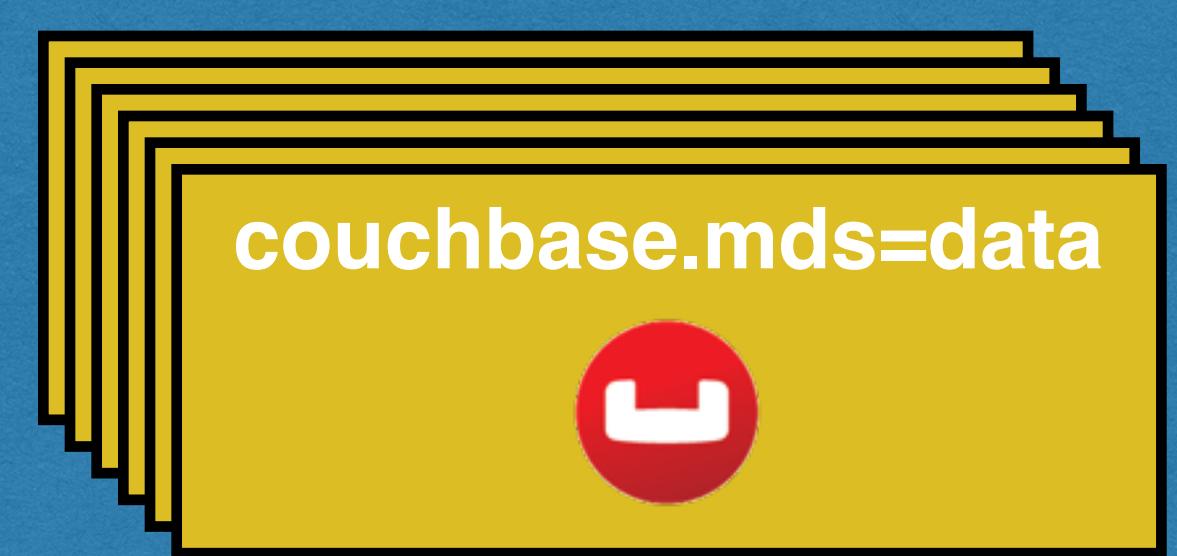
# Optimal Utilization of Resources



CPU intensive



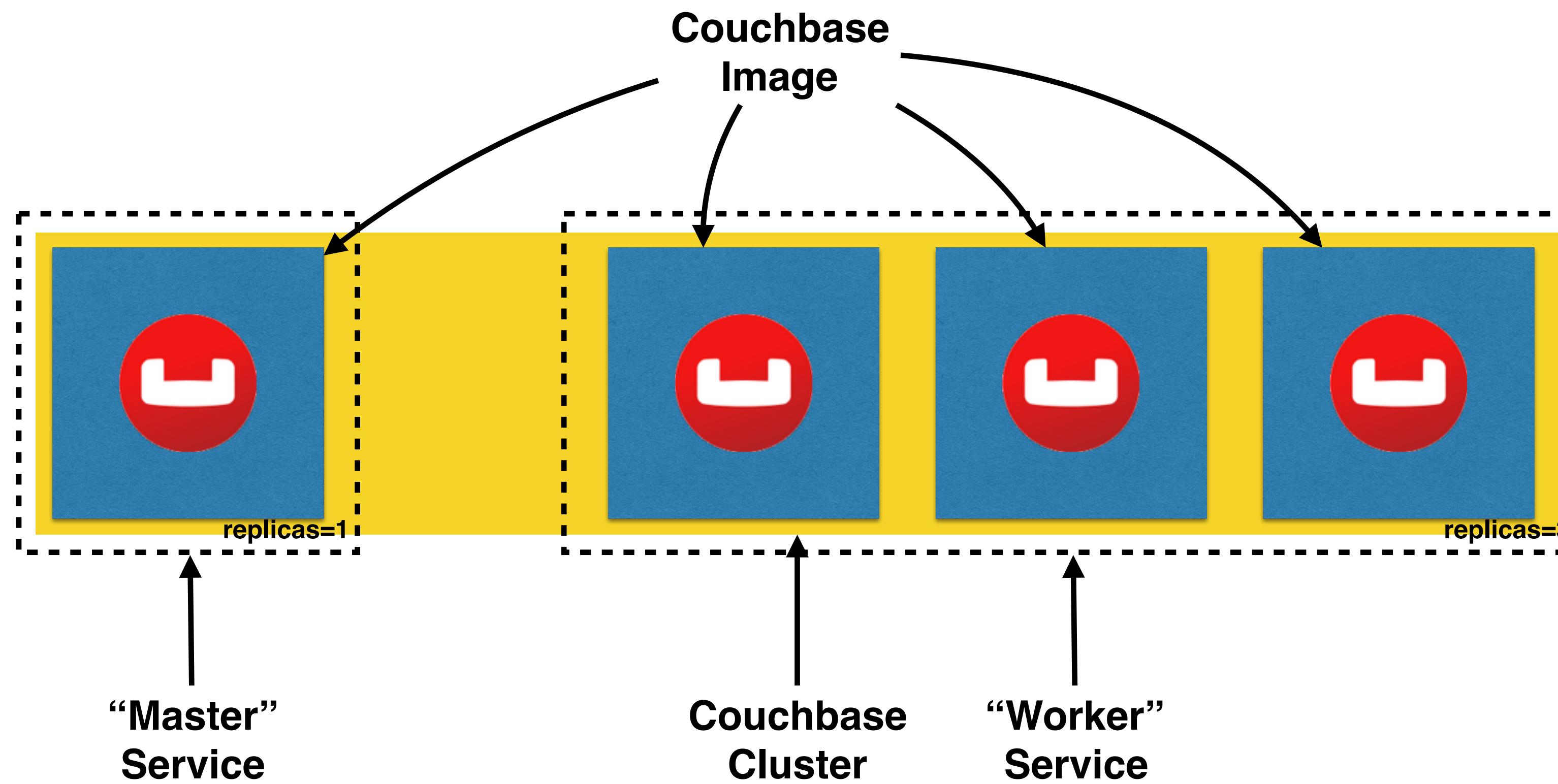
Disk intensive



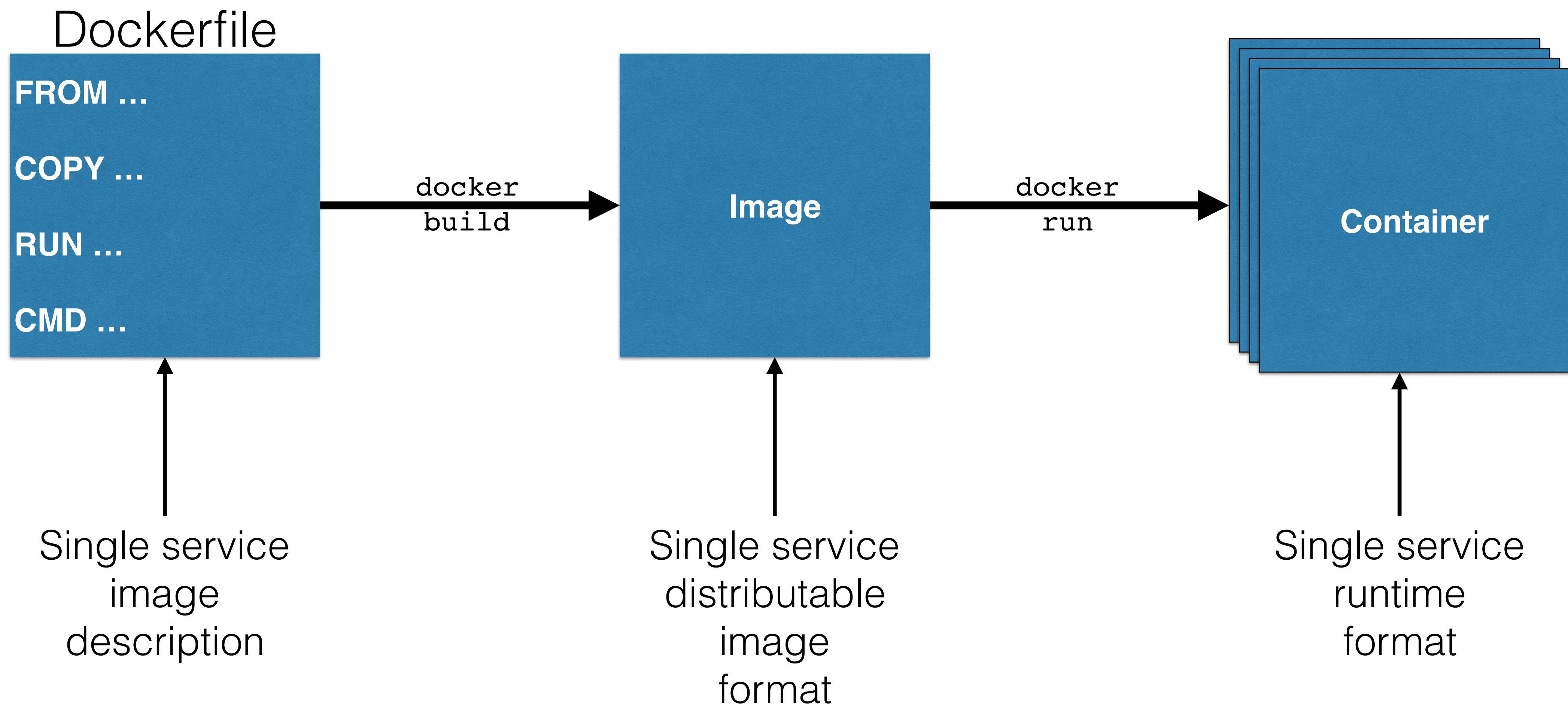
Memory +  
Fast read/write

- **Attach labels:** DOCKER\_OPTS="--label.couchbase.mds=data"
- **Run Containers:** docker service create --constraint engine.labels.couchbase.mds==index couchbase

# Couchbase Cluster using Docker Services



# Docker Lifecycle



Experimental  
feature

# Distributed Application Bundle

`docker-compose.yml`

```
version: "2"
services:
  db:
  ...
  web:
  ...
```

`docker-compose`  
build

Distributed  
Application  
Bundle

`docker`  
deploy

Stack

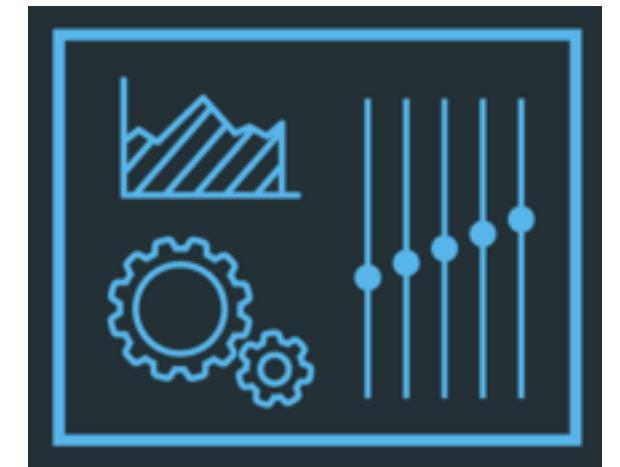
Multiservice  
image  
description

Multiservice  
distributable  
image  
format

Multiservice  
runtime  
format

# Monitoring Docker Containers

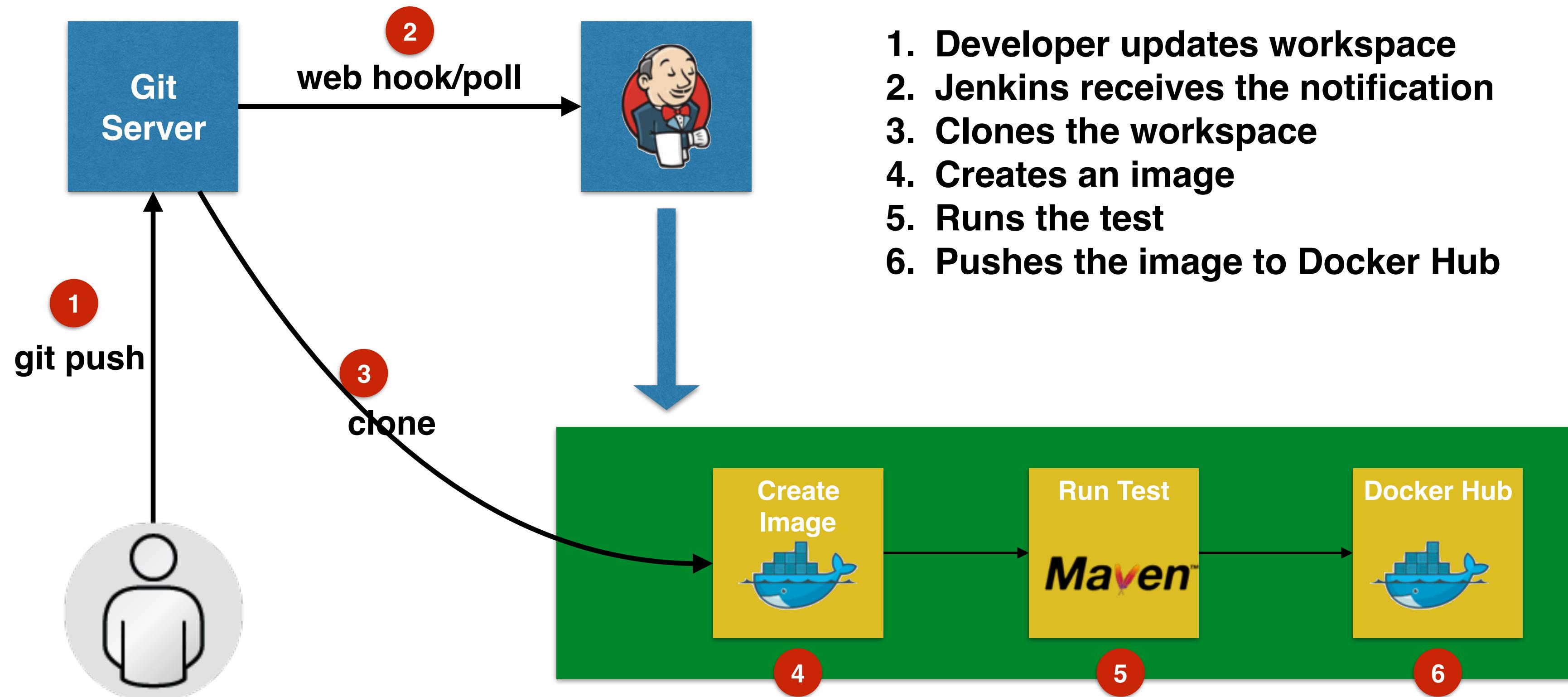
- `docker stats` command
  - LogEntries
- Docker Remote API: `/container/{container-name|cid}/stats`
- Docker Universal Control Plane
- cAdvisor
  - Prometheus
  - InfluxDB



cAdvisor



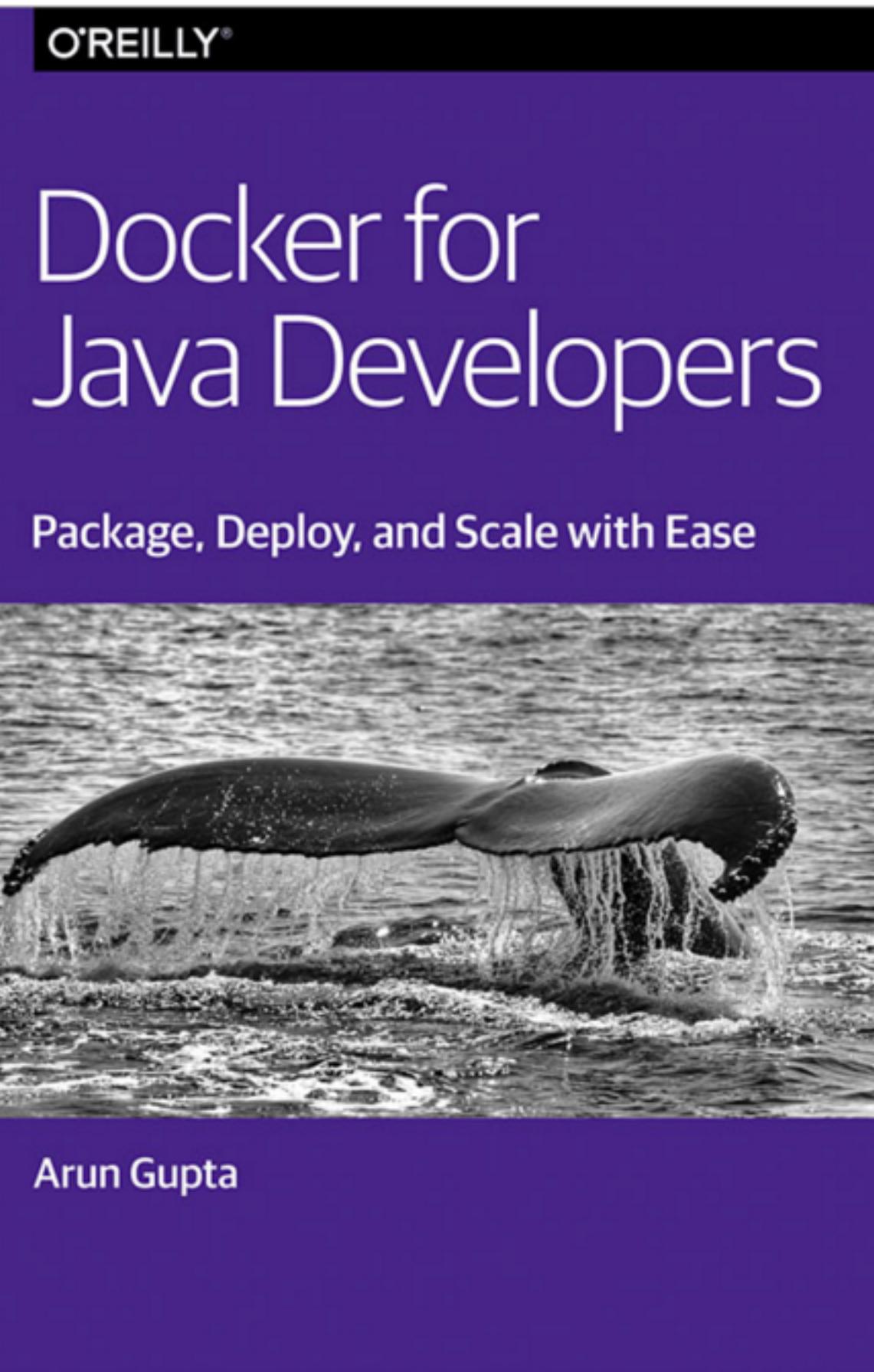
# CI/CD with Docker + Jenkins



# Docker Support in Java IDEs



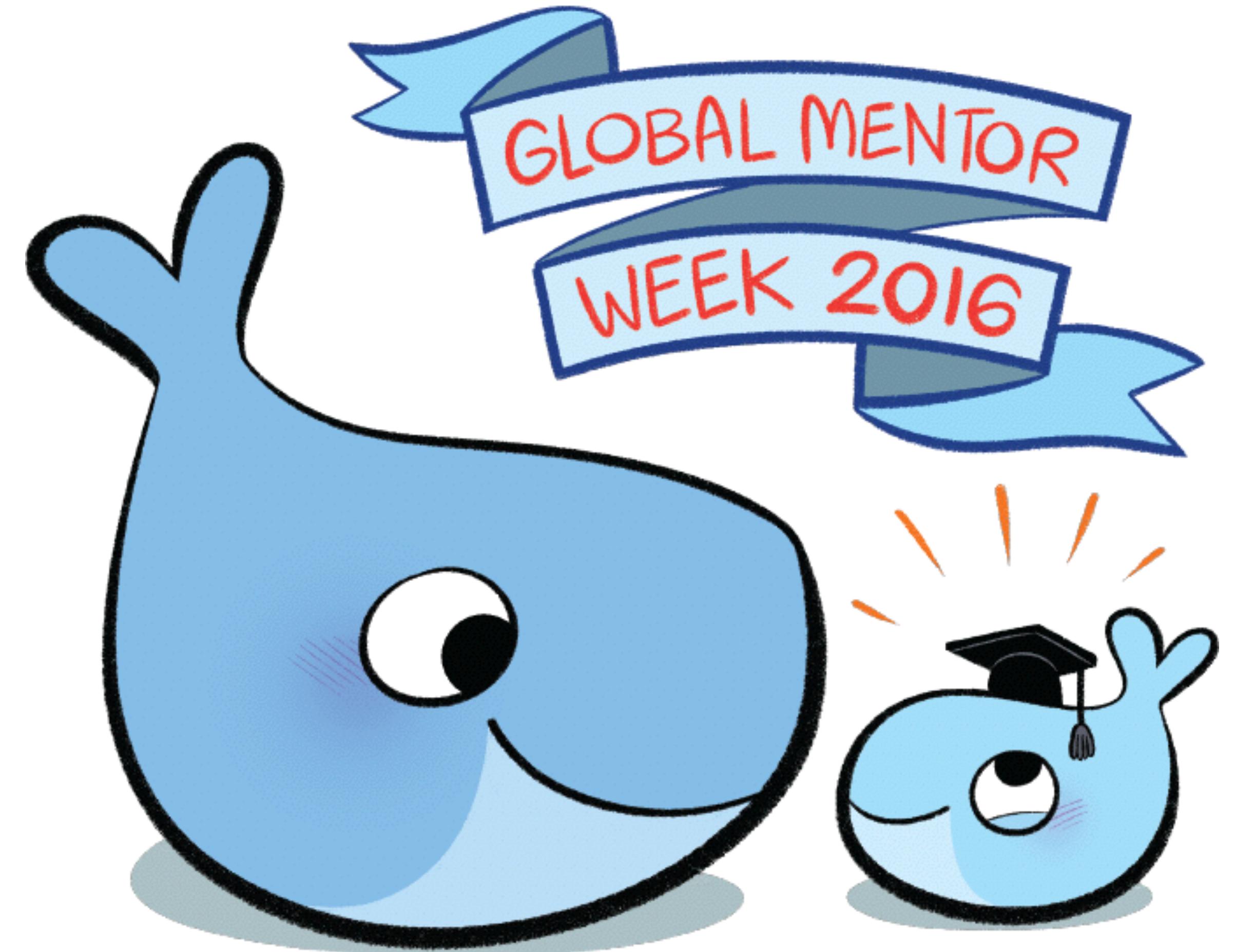
[bit.ly/dockerjava](http://bit.ly/dockerjava)



[bit.ly/kubejava](http://bit.ly/kubejava)



- Training to both newcomers and intermediate Docker users
- Advanced users will act as mentors
- Encourage connection and collaboration



 #learndocker  
November 14-20, 2016

# References

- Slides: [github.com/docker/labs/tree/master/slides](https://github.com/docker/labs/tree/master/slides)
- Workshop: [github.com/docker/labs/tree/master/java](https://github.com/docker/labs/tree/master/java)
- Docs: [docs.docker.com](https://docs.docker.com)