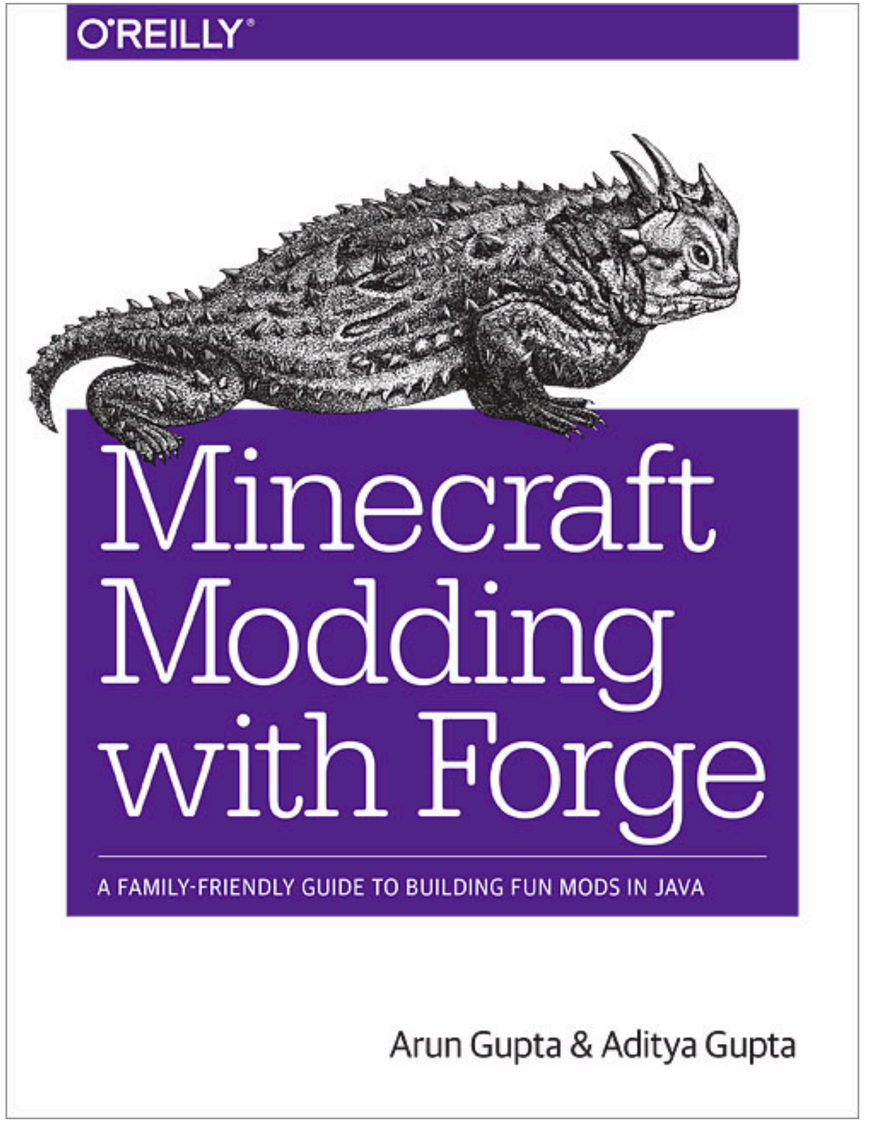
Docker for Java Developers

Arun Gupta, @arungupta
VP Developer Advocacy, Couchbase

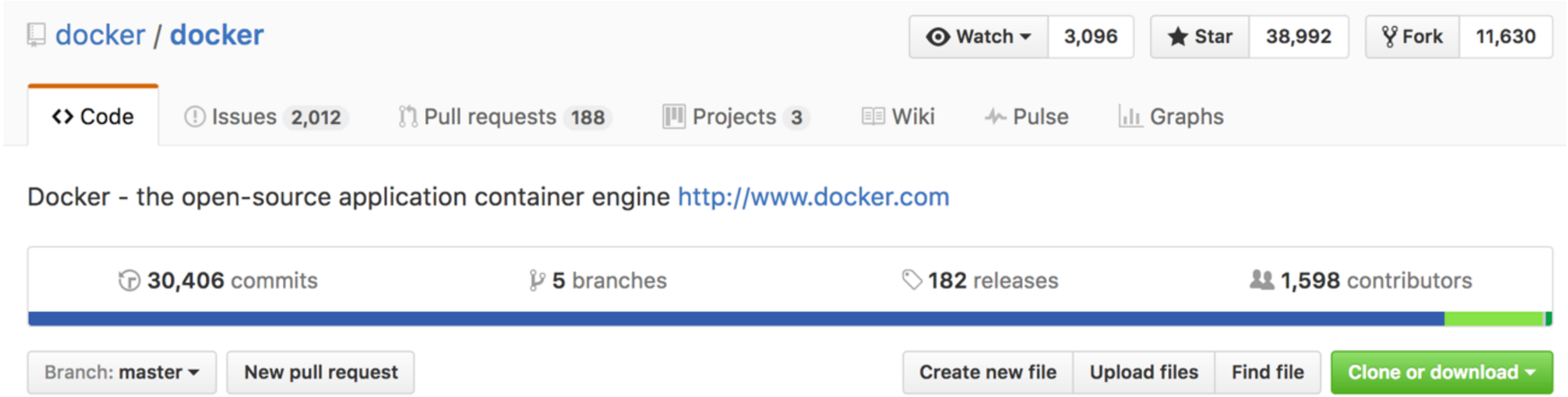
Couchbase

DEVOXX™ 4KIDS USA

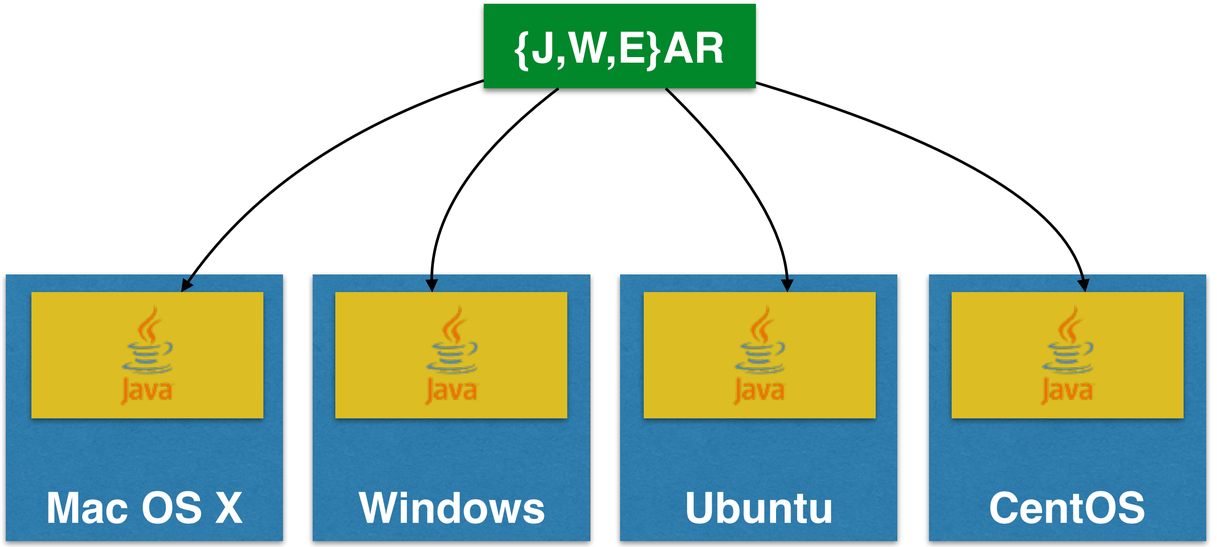


What is Docker?

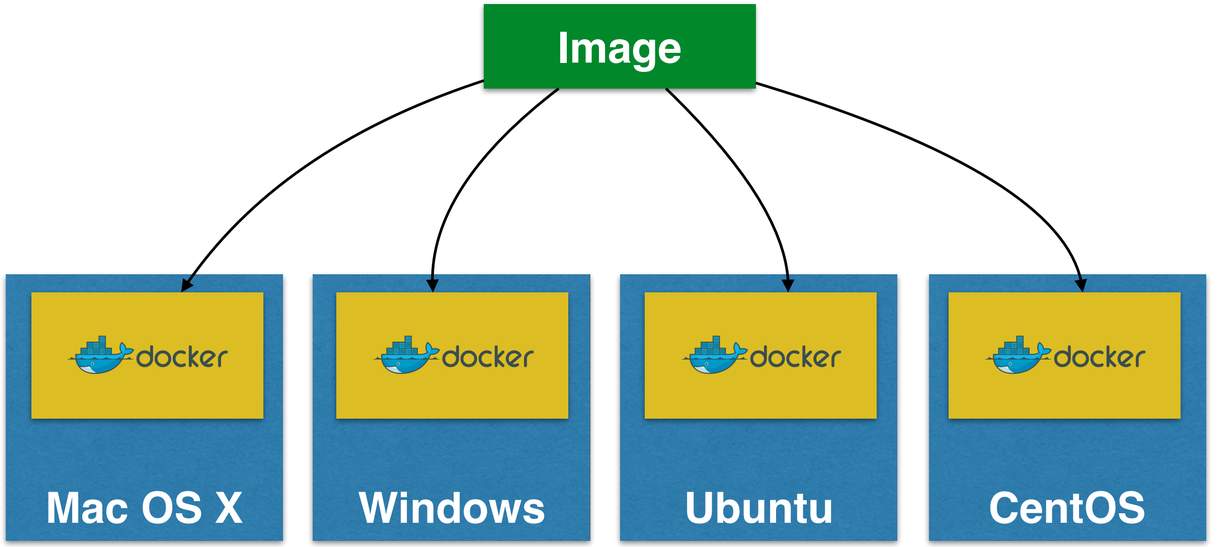
- Open source project and company



- Used to create containers for software applications

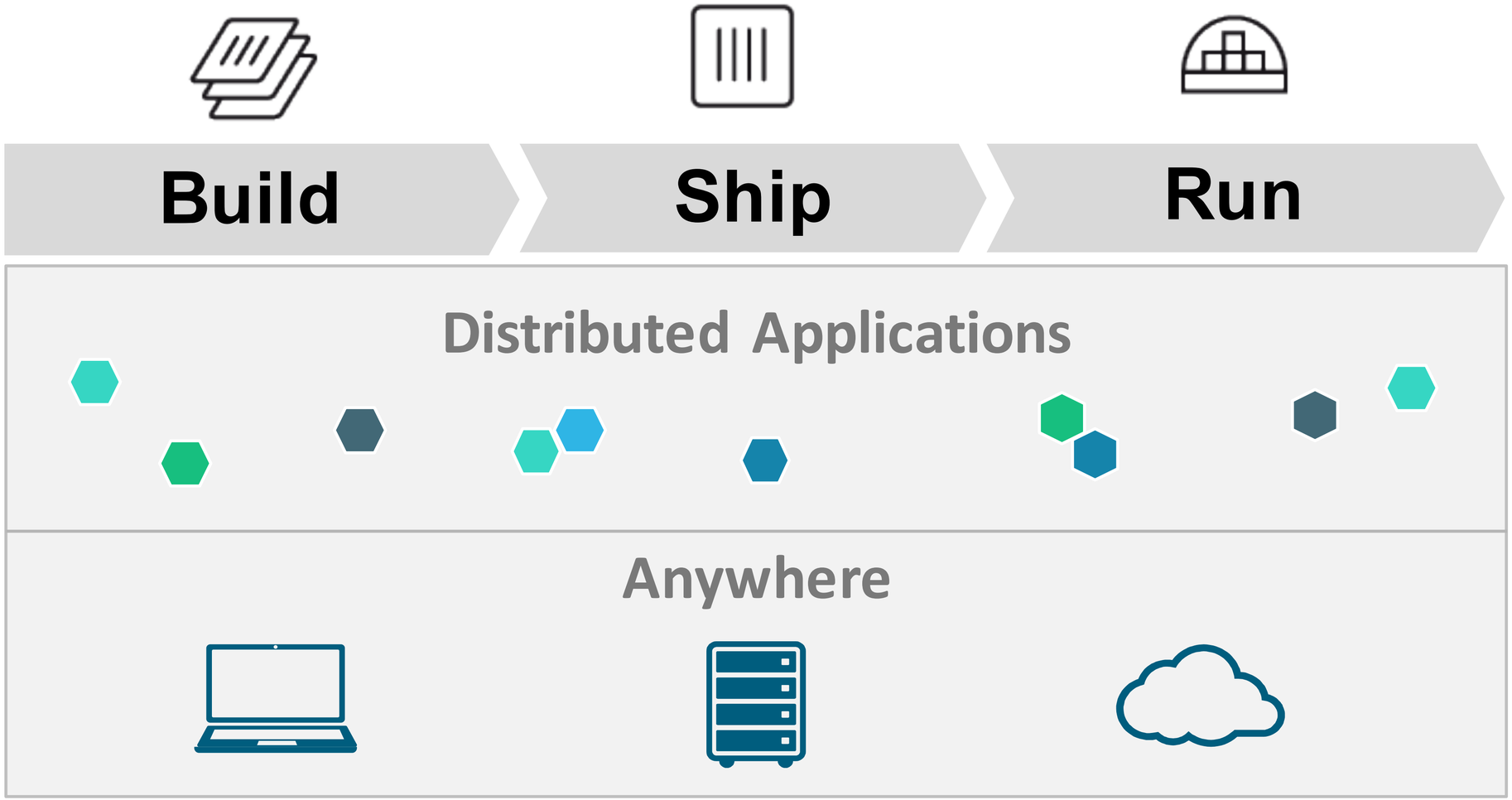


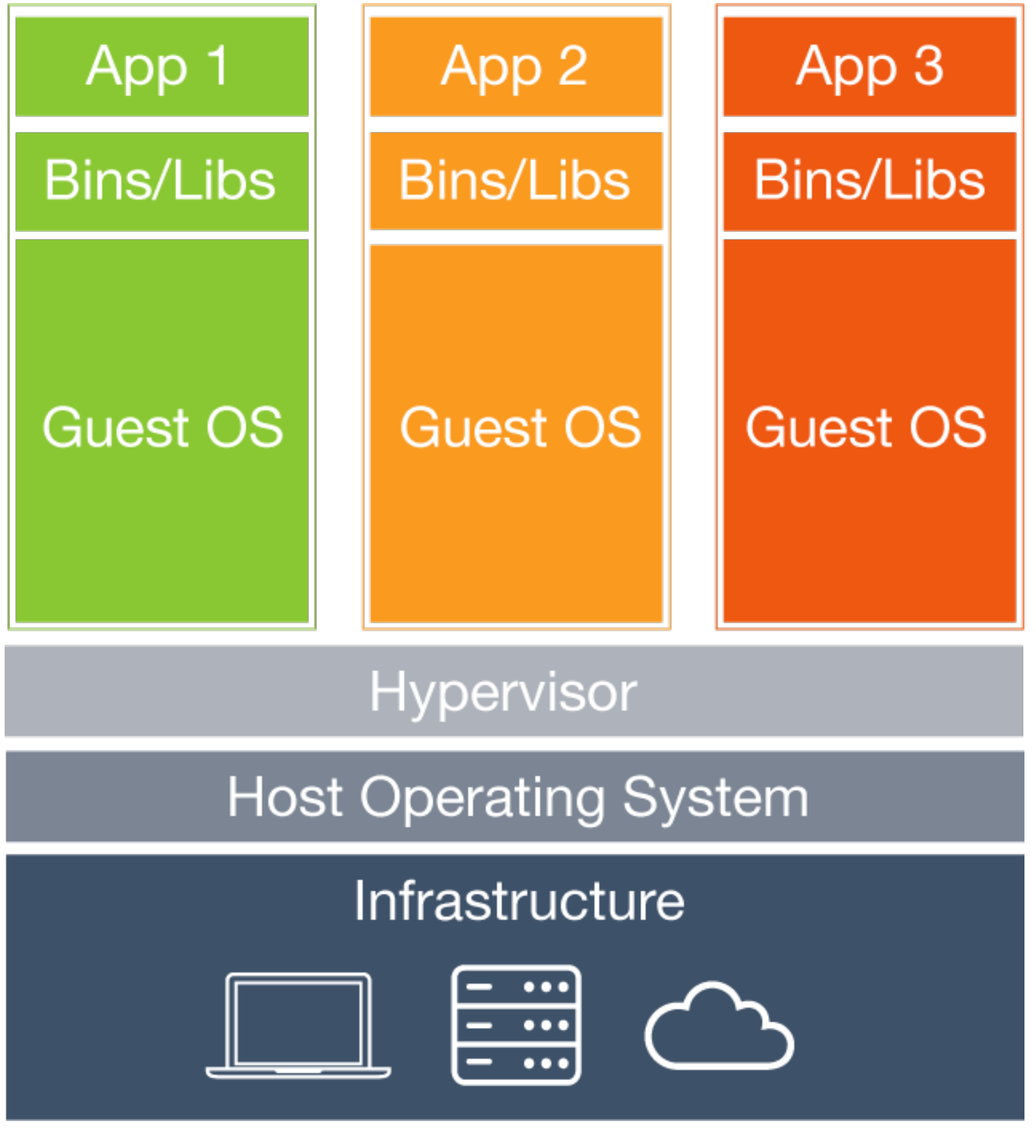
WORA = Write Once Run Anywhere

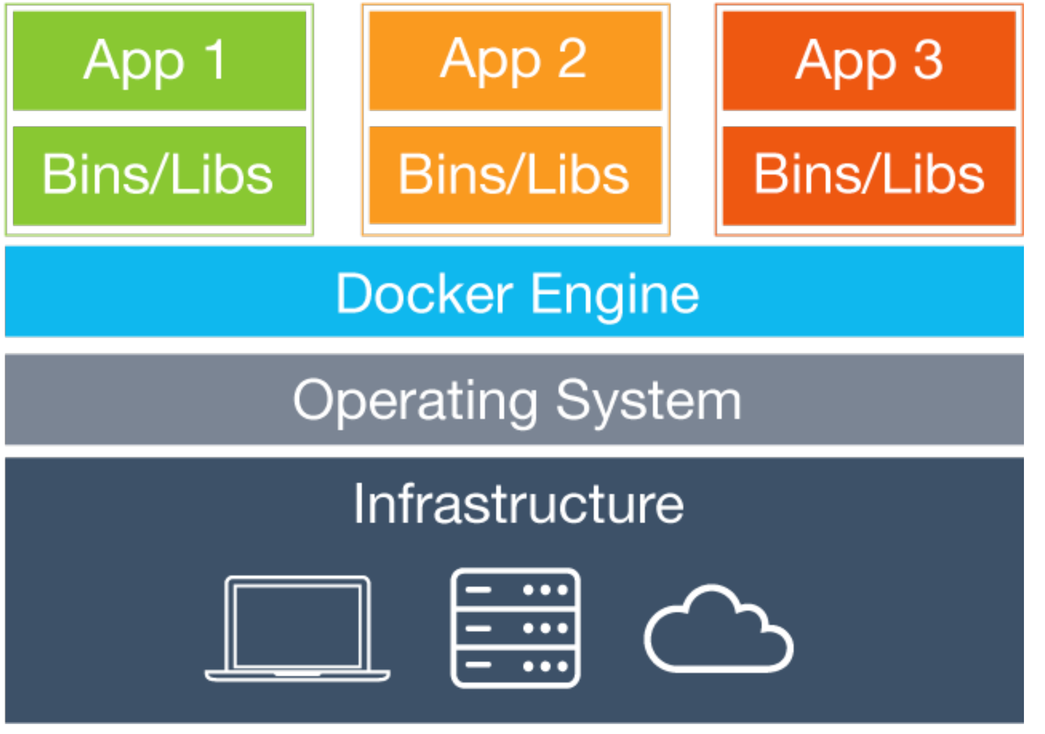


PODA = Package Once Deploy Anywhere

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
```

Dockerfile reference

Usage

Format

Parser directives

escape

Environment replacement

.dockerignore file

FROM

RUN

Known issues (RUN)

CMD

LABEL

MAINTAINER (deprecated)

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

STOPSIGNAL

HEALTHCHECK

SHELL

Dockerfile examples

https://docs.docker.com/engine/reference/builder/

Docker Workflow

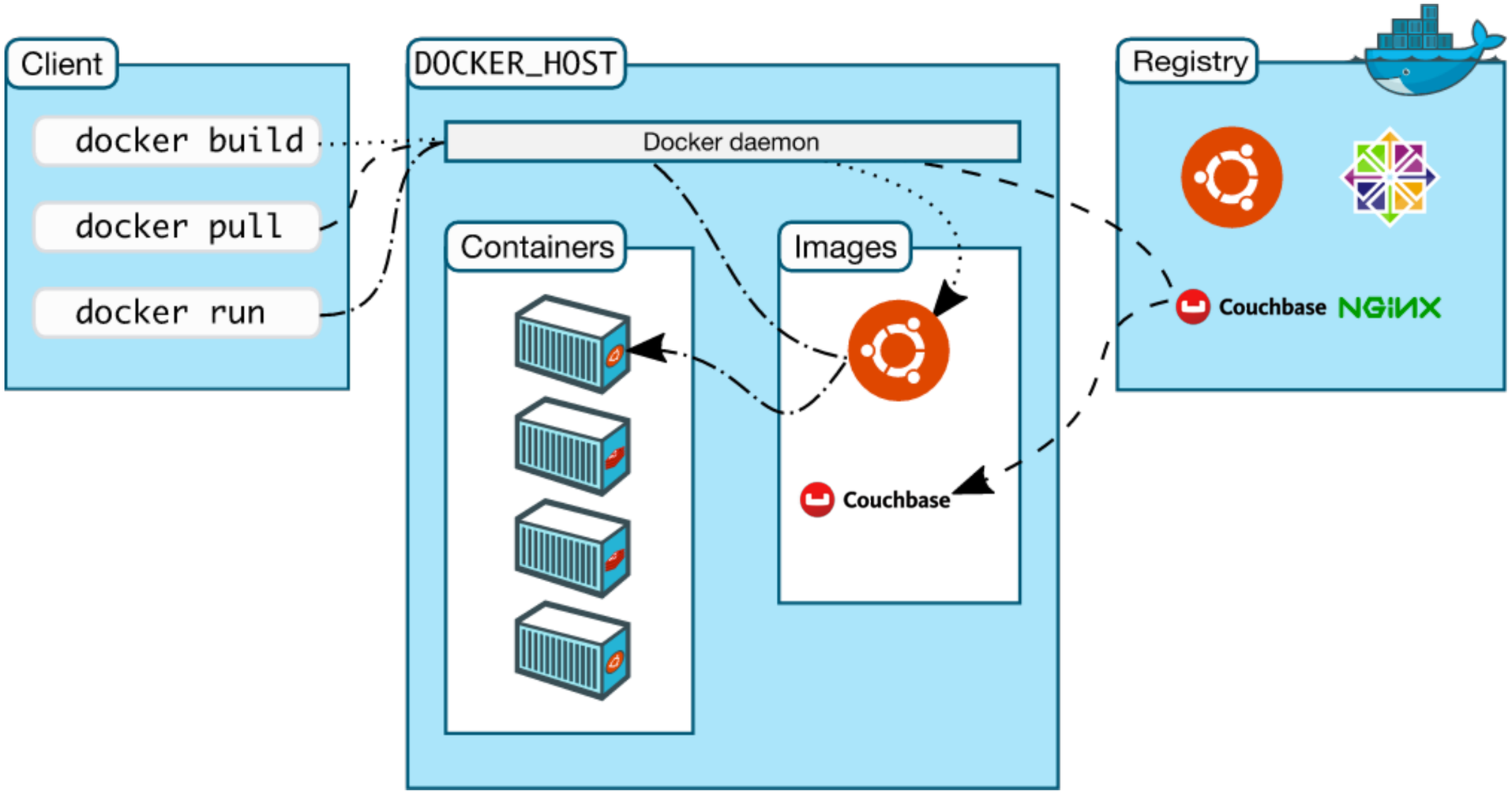


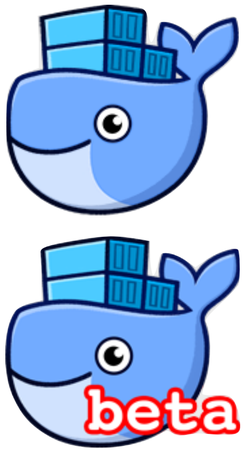
Image Layers - OpenJDK

```
~ > docker image ls openjdk
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
openjdk             latest              d23bdf5b1b1b        5 days ago          643 MB
~ > docker image history openjdk
IMAGE               CREATED             CREATED BY                                      SIZE
COMMENT
d23bdf5b1b1b        5 days ago          /bin/sh -c /var/lib/dpkg/info/ca-certifica...   419 kB
<missing>           5 days ago          /bin/sh -c set -x  && apt-get update  && a...   352 MB
<missing>           5 days ago          /bin/sh -c #(nop)  ENV CA_CERTIFICATES_JAV...   0 B
<missing>           5 days ago          /bin/sh -c #(nop)  ENV JAVA_DEBIAN_VERSION...   0 B
<missing>           5 days ago          /bin/sh -c #(nop)  ENV JAVA_VERSION=8u111       0 B
<missing>           5 days ago          /bin/sh -c #(nop)  ENV JAVA_HOME=/usr/lib/...   0 B
<missing>           5 days ago          /bin/sh -c {   echo '#!/bin/sh';   echo 's...   87 B
<missing>           5 days ago          /bin/sh -c #(nop)  ENV LANG=C.UTF-8             0 B
<missing>           5 days ago          /bin/sh -c echo 'deb http://deb.debian.org...   55 B
<missing>           5 days ago          /bin/sh -c apt-get update && apt-get insta...   1.29 MB
<missing>           5 days ago          /bin/sh -c apt-get update && apt-get insta...   123 MB
<missing>           5 days ago          /bin/sh -c apt-get update && apt-get insta...   44.3 MB
<missing>           6 days ago          /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           6 days ago          /bin/sh -c #(nop) ADD file:89ecb642d662ee7...   123 MB
```

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 (restricted availability)

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
- Requires Yosemite 10.10+ or Windows 10 64-bit

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`
- Deployed as Docker Stack
- Great for dev, staging, and CI

Docker Compose - One Service


```
version: “3”
services:
  db:
   image: couchbase
   volumes:
     - ~/couchbase:/opt/couchbase/var
   ports:
     - 8091:8091
     - 8092:8092
     - 8093:8093
     - 11210:11210
```

```
docker-compose up -d

docker stack deploy \
--compose-file=docker-compose.yml \
couchbase
```

Docker Compose - Two Services

Docker Compose - Two Services

```
version: "3"
services:
  db:
    image: arungupta/couchbase:travel
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
  web:
    image: arungupta/couchbase-wildfly-javaee:travel
    environment:
      - COUCHBASE_URI=db
    ports:
      - 8080:8080
      - 9990:9990
```

```
docker stack deploy \
--compose-file=docker-compose.yml \
couchbase
```

Overriding Services in Docker Compose

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

docker-compose.yml

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

docker-compose.override.yml

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	`docker-compose up`
Local/remote host	`DOCKER_HOST, DOCKER_TLS_VERIFY, DOCKER_CERT_PATH`
Single/multiple hosts	Integrated with Swarm
Multiple isolated environments	`docker-compose up -p <project>`

Automated test setup	`docker-compose up`
	`mvn test`
	`docker-compose down`

Dev/Prod Impedance mismatch	`docker-compose up -f docker-compose.yml -f production.yml`

Docker 1.13

- Deploy Compose services to Swarm
- CLI restructured
- Clean-up commands
- Monitoring commands
- Build improvements
- Improved CLI backwards compatibility
- Docker for AWS/Azure for Production

Docker 1.13 - Compose v3

- `docker stack deploy` now supports Compose file
  - Number of desired instances of each service
  - Rolling update
  - Server constraints

Docker 1.13 - CLI Restructured

```
Management Commands:
  checkpoint  Manage checkpoints
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  volume      Manage volumes
```

Docker 1.13 - Cleanup Commands

- `docker system df` and `docker system cleanup`

```
docker system df
TYPE                TOTAL               ACTIVE              SIZE                RECLAIMABLE
Images              15                  1                   5.081 GB            4.498 GB (88%)
Containers          1                   0                   130.1 kB            130.1 kB (100%)
Local Volumes       7                   0                   110.1 MB            110.1 MB (100%)
```

Docker 1.13 - Monitoring Commands

- `docker service logs` and Prometheus endpoint

```
couchbase_db.1.rchu2uykeuuj@moby    | ++ set -m
couchbase_db.1.rchu2uykeuuj@moby    | ++ sleep 15
couchbase_db.1.rchu2uykeuuj@moby    | ++ /entrypoint.sh couchbase-server
couchbase_db.2.kjy7l14weao8@moby    | ++ set -m
couchbase_db.2.kjy7l14weao8@moby    | ++ sleep 15
couchbase_db.1.rchu2uykeuuj@moby    | Starting Couchbase Server -- Web UI av
couchbase_db.1.rchu2uykeuuj@moby    | ++ curl -v -X POST http://127.0.0.1:80
couchbase_db.2.kjy7l14weao8@moby    | ++ /entrypoint.sh couchbase-server
couchbase_db.2.kjy7l14weao8@moby    | Starting Couchbase Server -- Web UI av
```

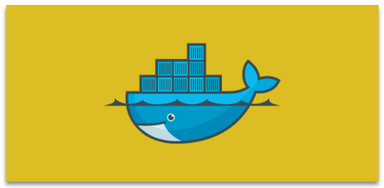
Docker 1.13 - Build Improvements

- **`docker build --squash`**: Squash newly built layers into a single layer
- **`docker build --compress`**: Compress the build context using gzip

Swarm Mode

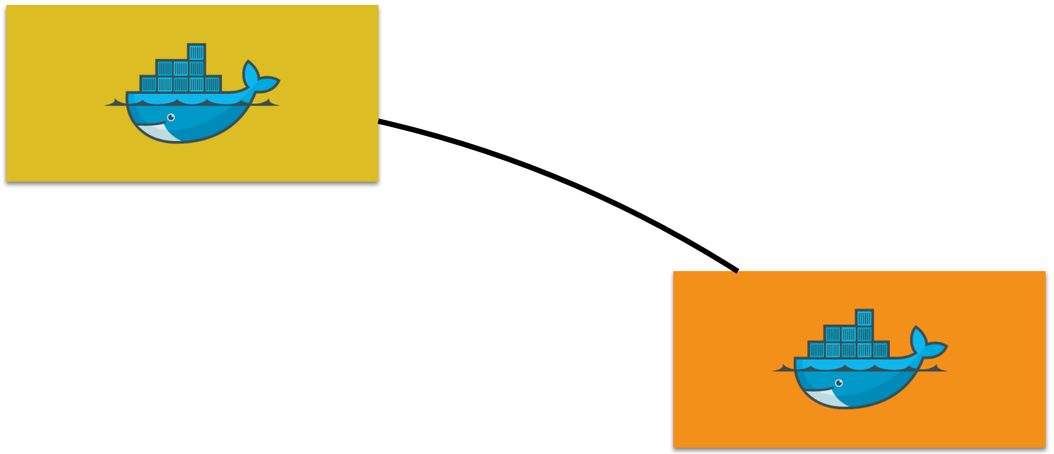
- New in 1.12
- Natively managing a cluster of Docker Engines called a Swarm
- Docker CLI to create a swarm, deploy apps, and manage swarm
  - Optional feature, need to be explicitly enabled
- No Single Point of Failure (SPOF)
- Declarative state model
- Self-organizing, self-healing
- Service discovery,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

Swarm Mode: Initialize



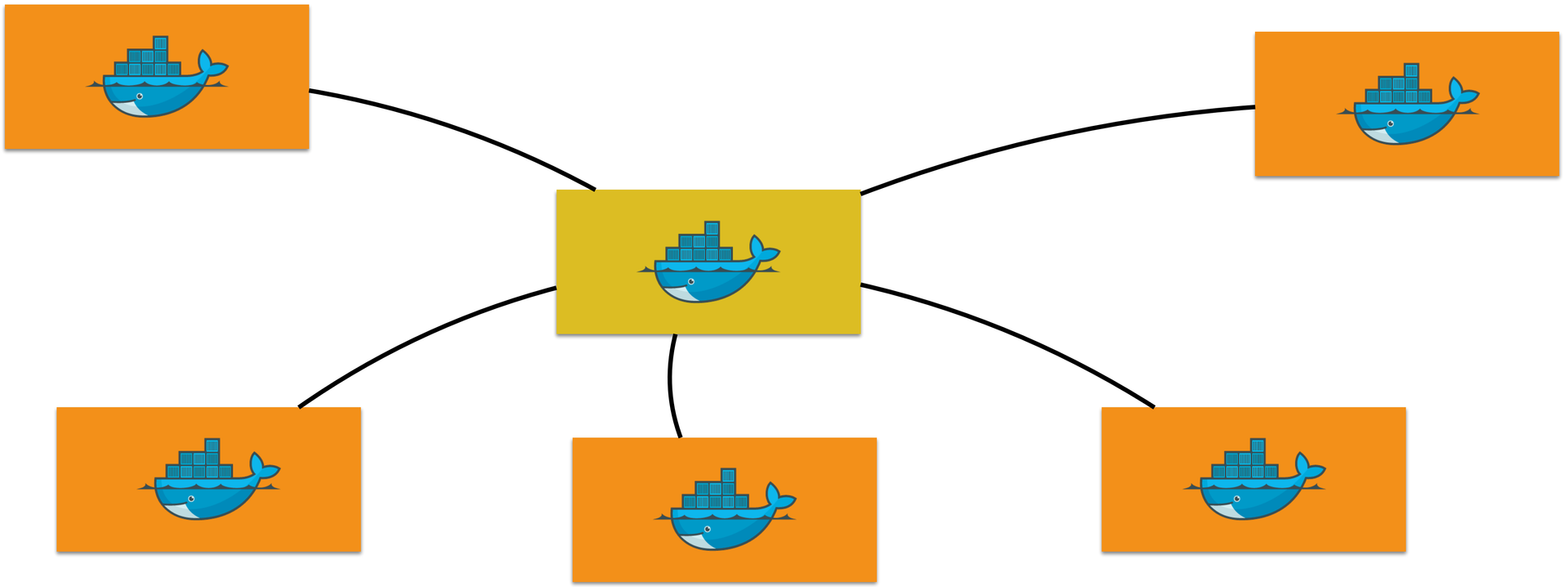
```
docker swarm init --listen-addr <ip>:2377
```

Swarm Mode: Add Worker



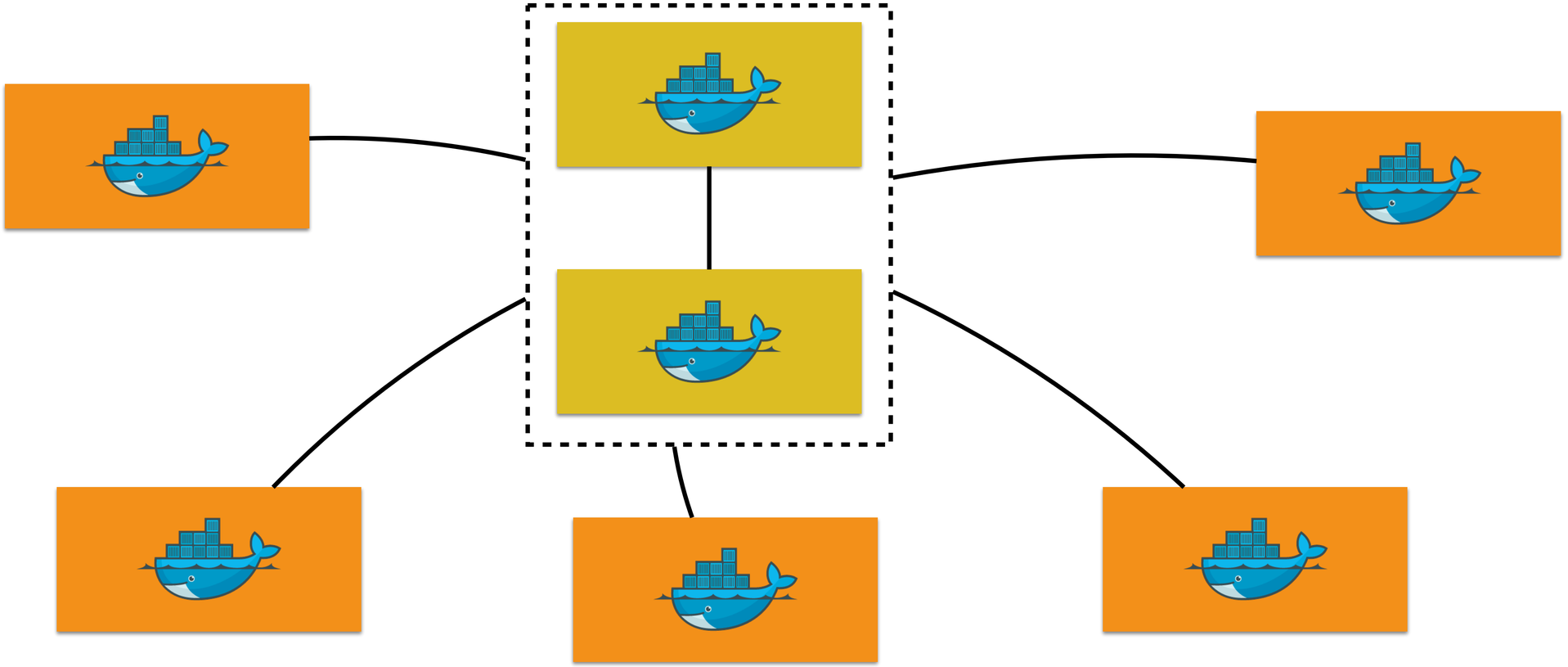
```
docker swarm join --token <worker_token> <manager>:2377
```

Swarm Mode: Add More Workers



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

Swarm Mode: Primary/Secondary Master



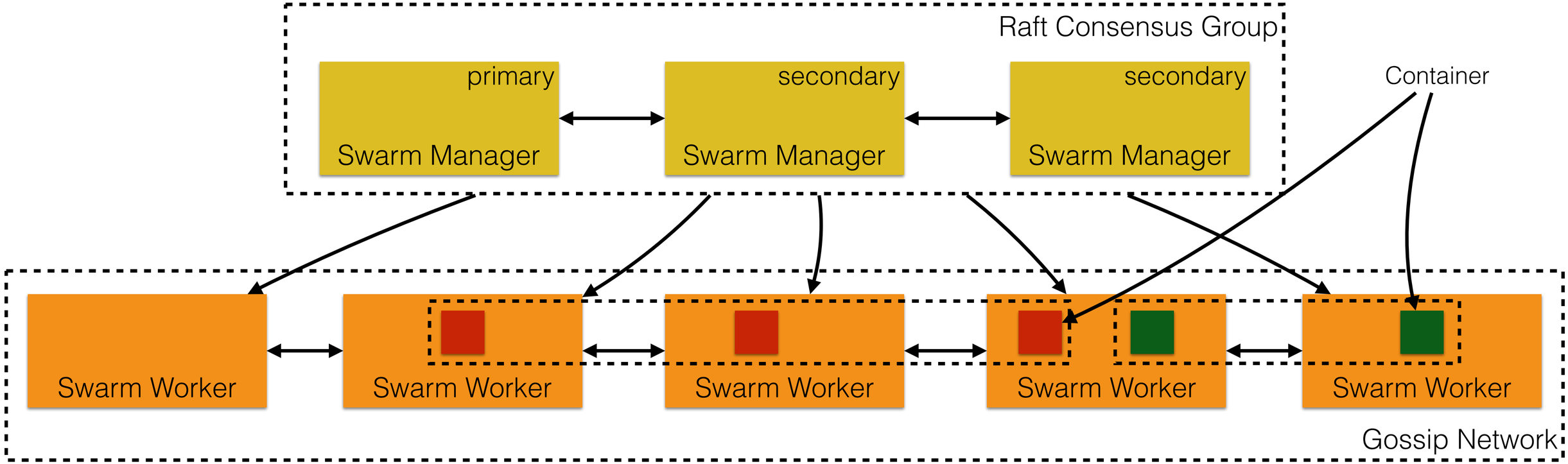
```
docker swarm join --manager --token <manager_token> --listen-
addr <master2>:2377 <master1>:2377
```

Swarm Mode using Docker Machine

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

https://github.com/docker/labs/blob/master/swarm-mode/quickstart/buildswarm-node-vbox-setup.sh

Swarm Mode: Protocols

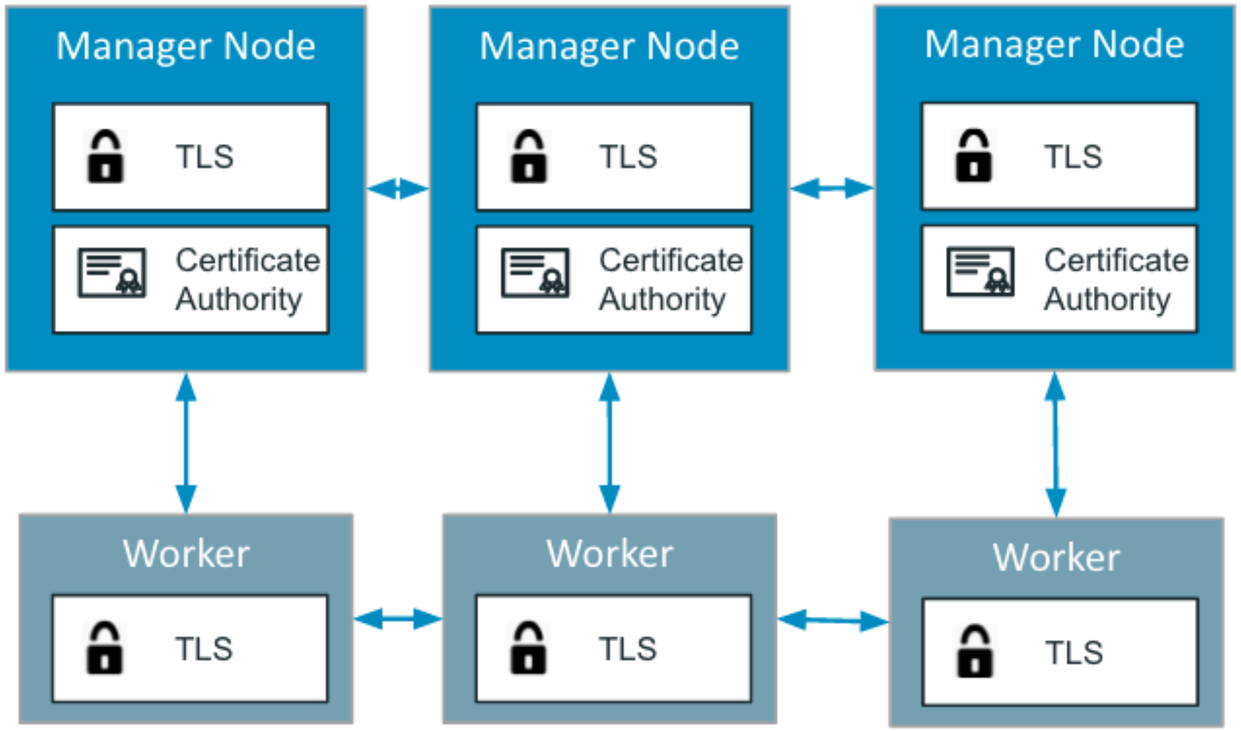


Strongly consistent
Replicated (Raft based)
Extremely fast (in-memory reads)

Swarm Mode in Production

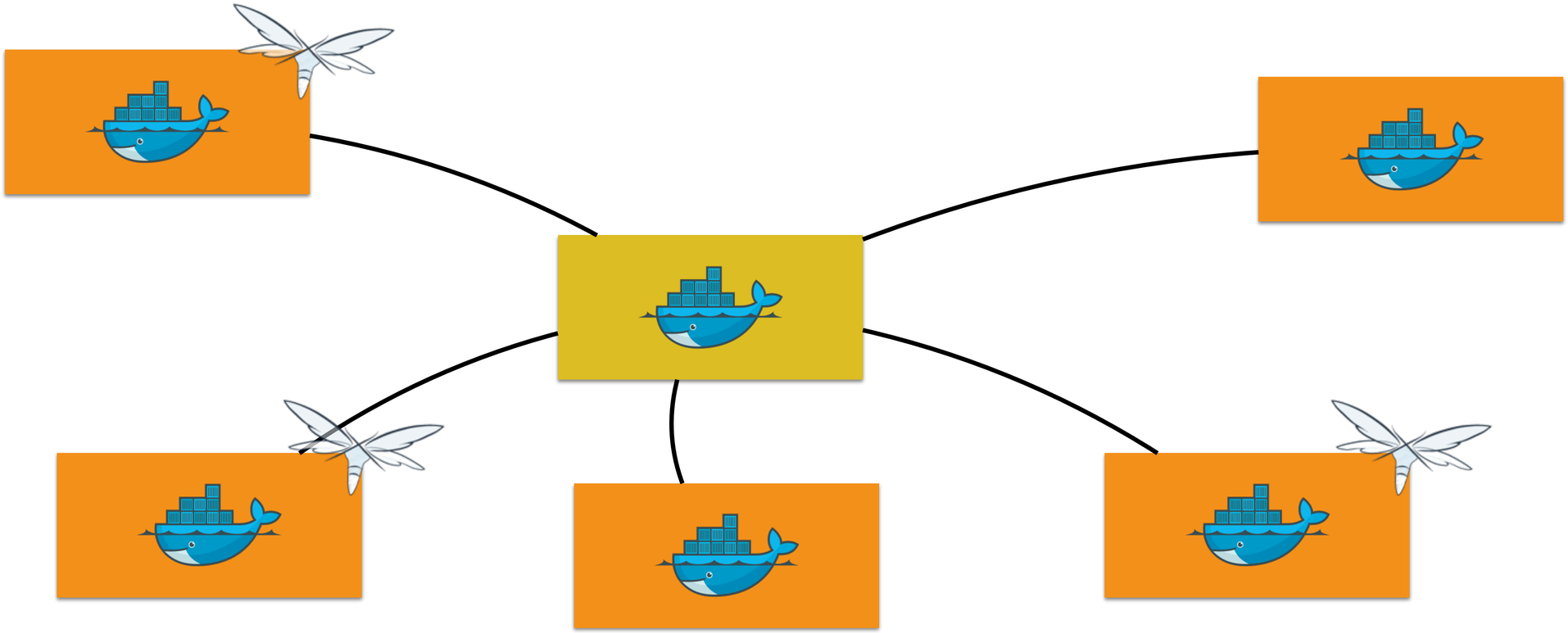
https://blog.online.net/2016/07/29/docker-swarm-an-analysis-of-a-very-large-scale-container-system/

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

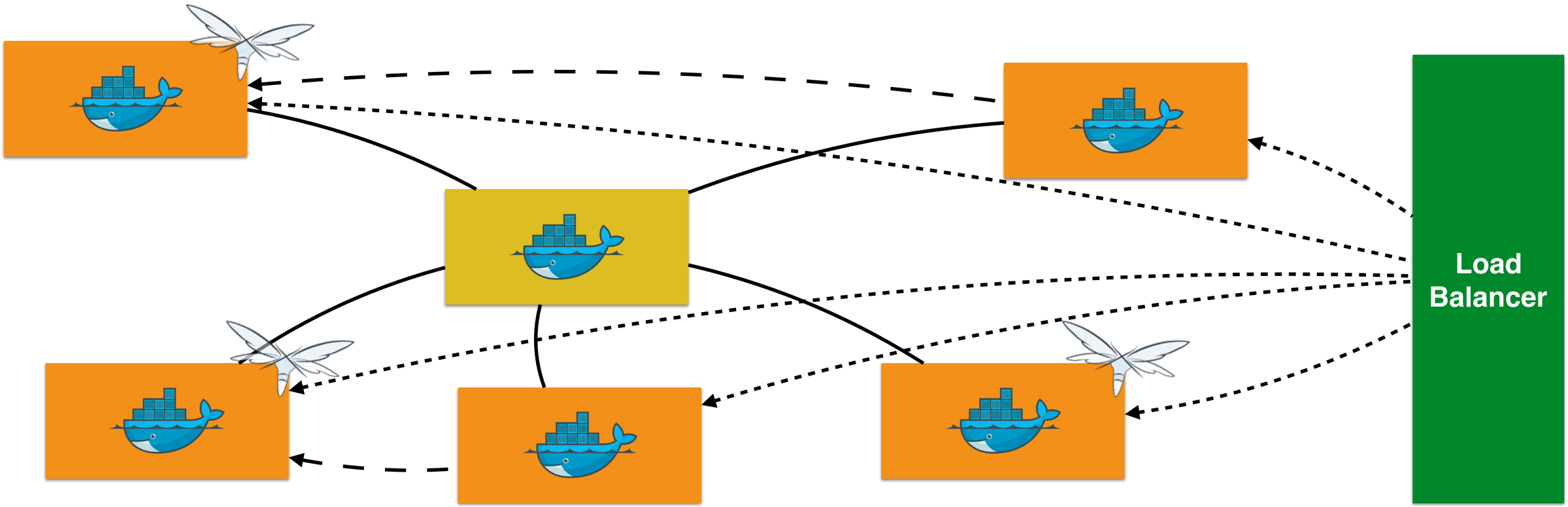


```
docker service create --replicas 3 --name web jboss/wildfly
```

Swarm Mode - Routing Mesh

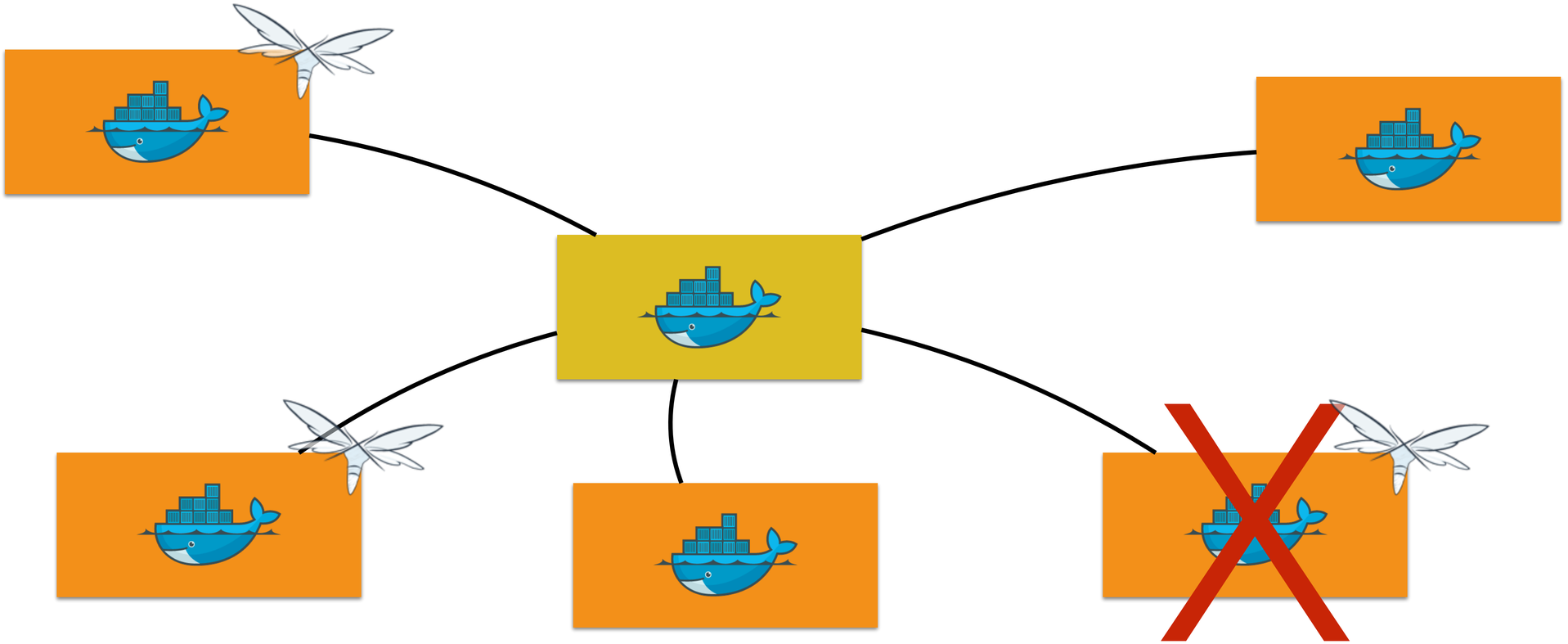
- Load balancers are host-aware, not container-aware
- Swarm mode introduces container-aware routing mesh
- Reroutes traffic from any host to a container
  - Reserves a Swarm-wide ingress port
  - Uses DNS-based service discovery

Swarm Mode: Routing Mesh

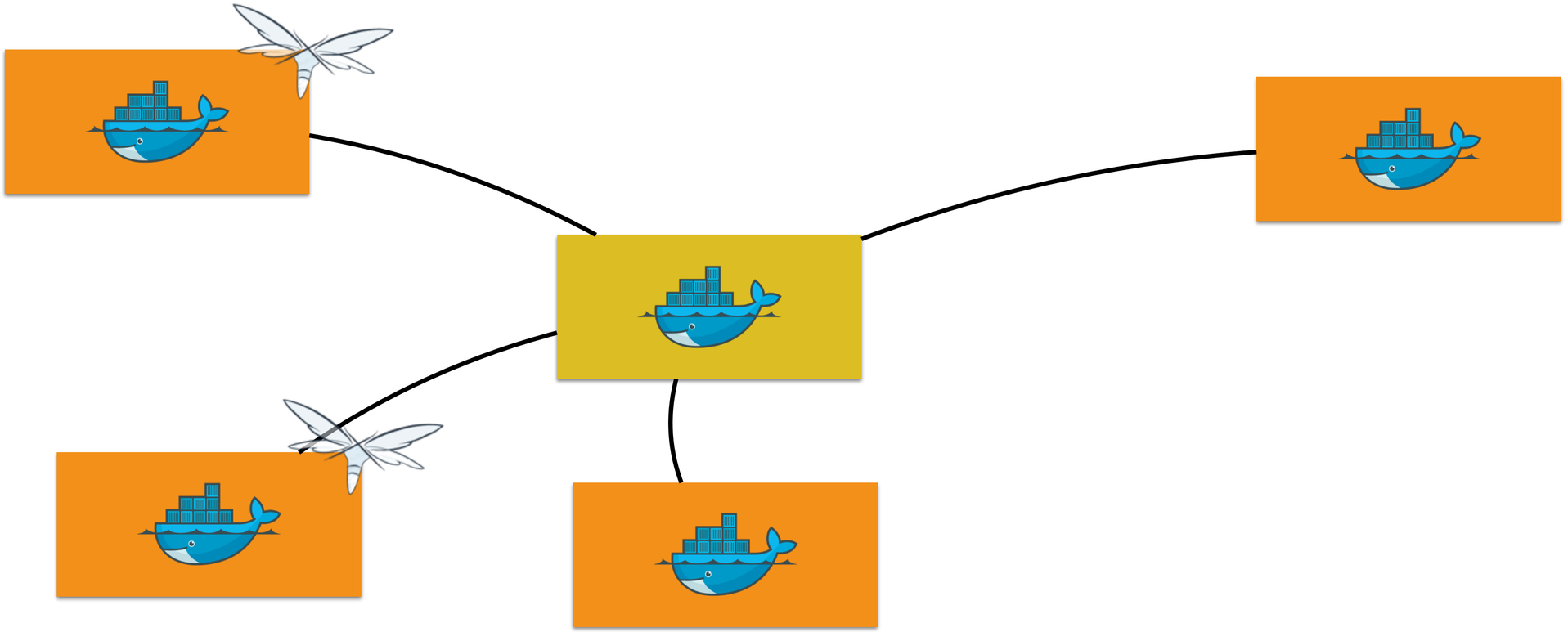



```
docker service create --replicas 3 --name web -p 8080:8080 jboss/
wildfly
```

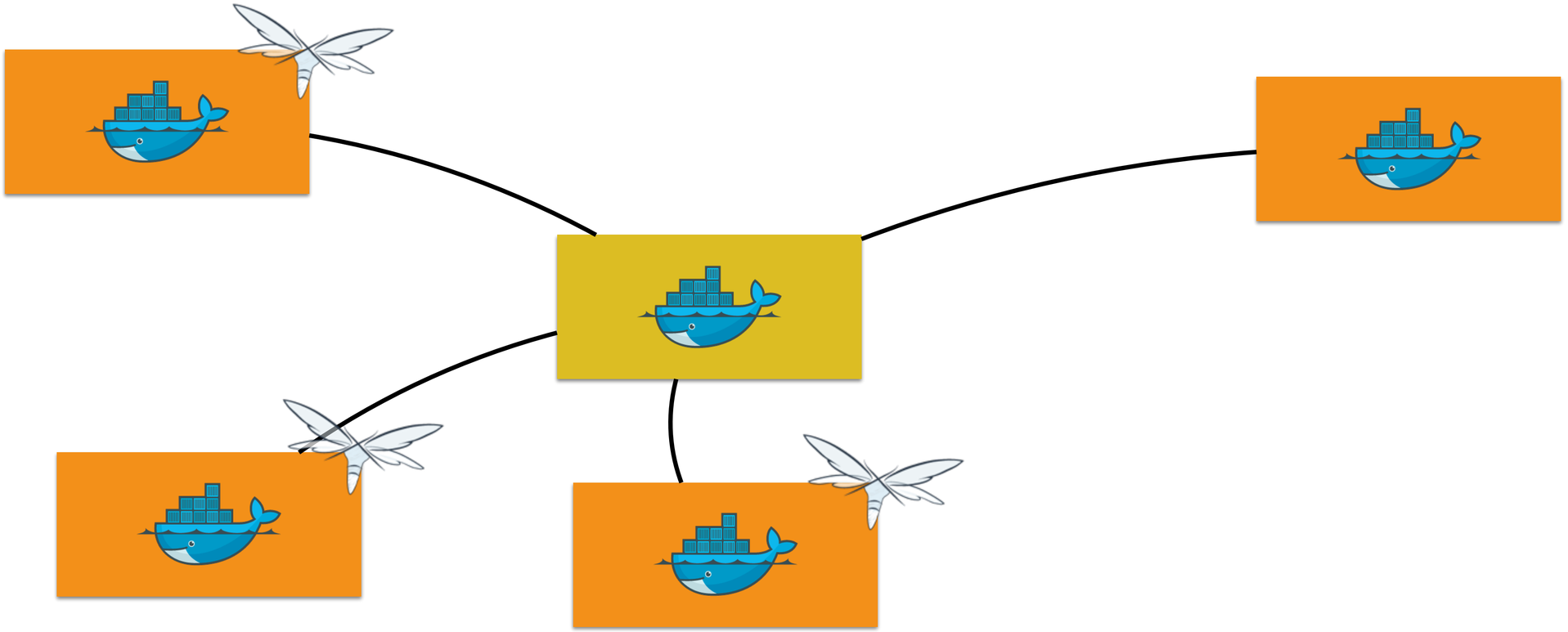
Swarm Mode: Node Failure



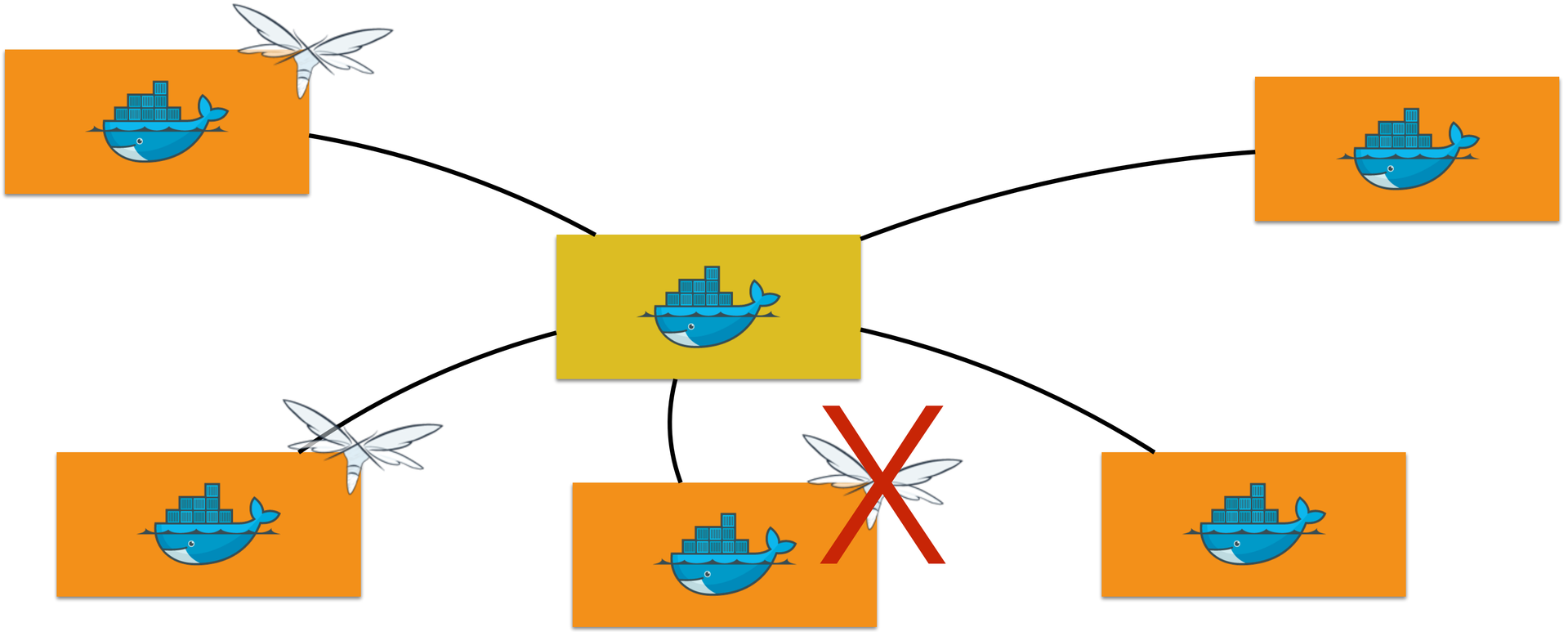
Swarm Mode: Desired != Actual



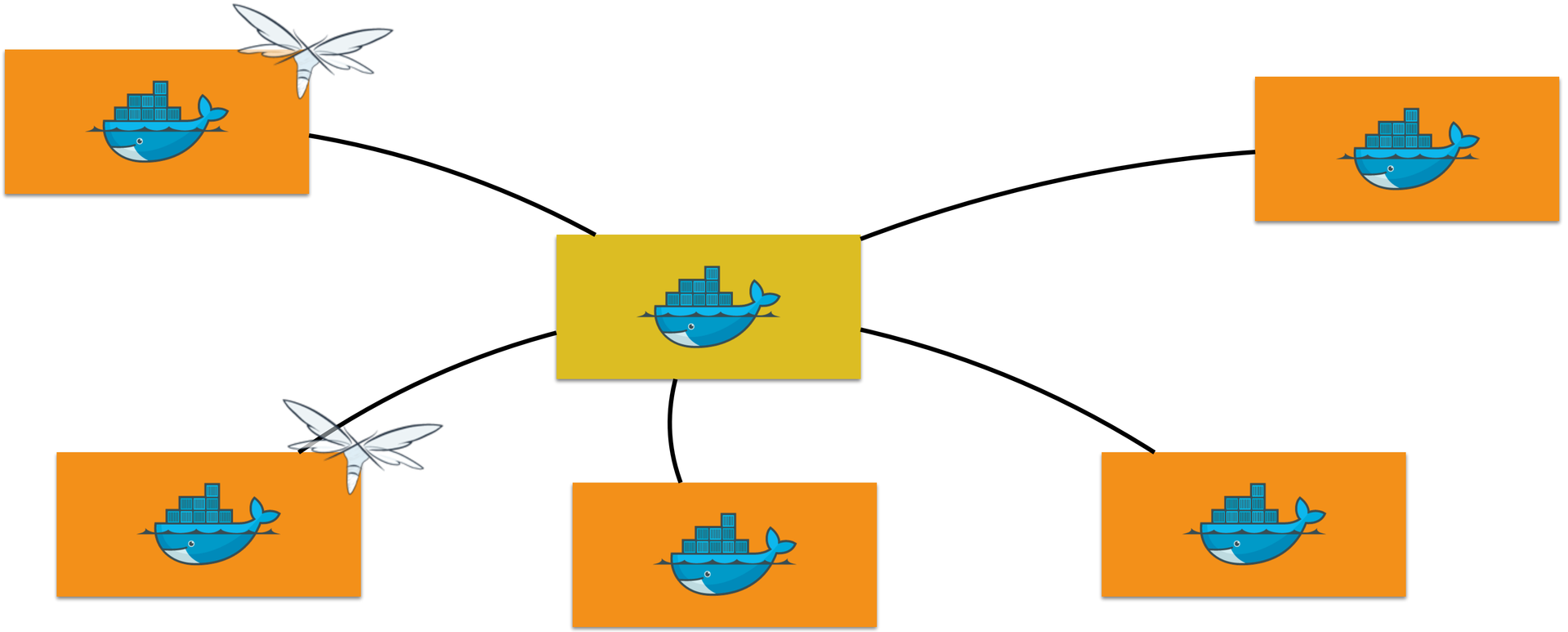
Swarm Mode: Reconcile



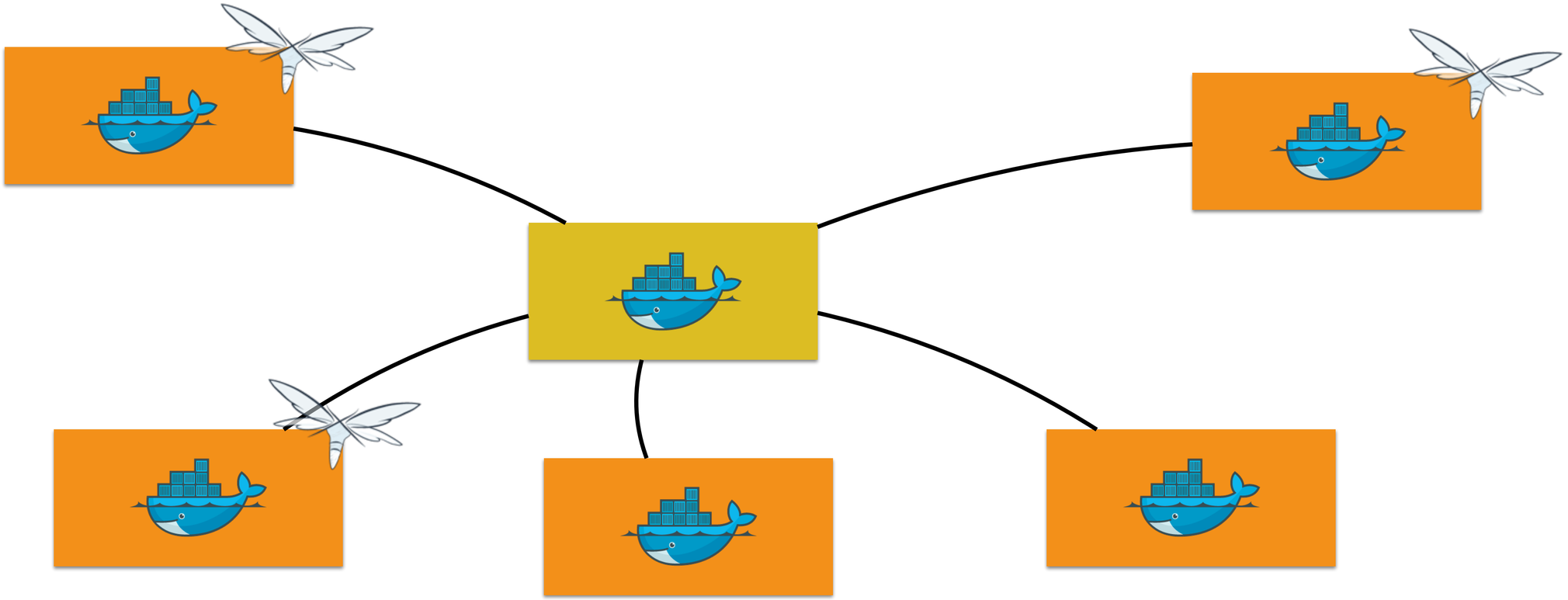
Swarm Mode: Container Failure



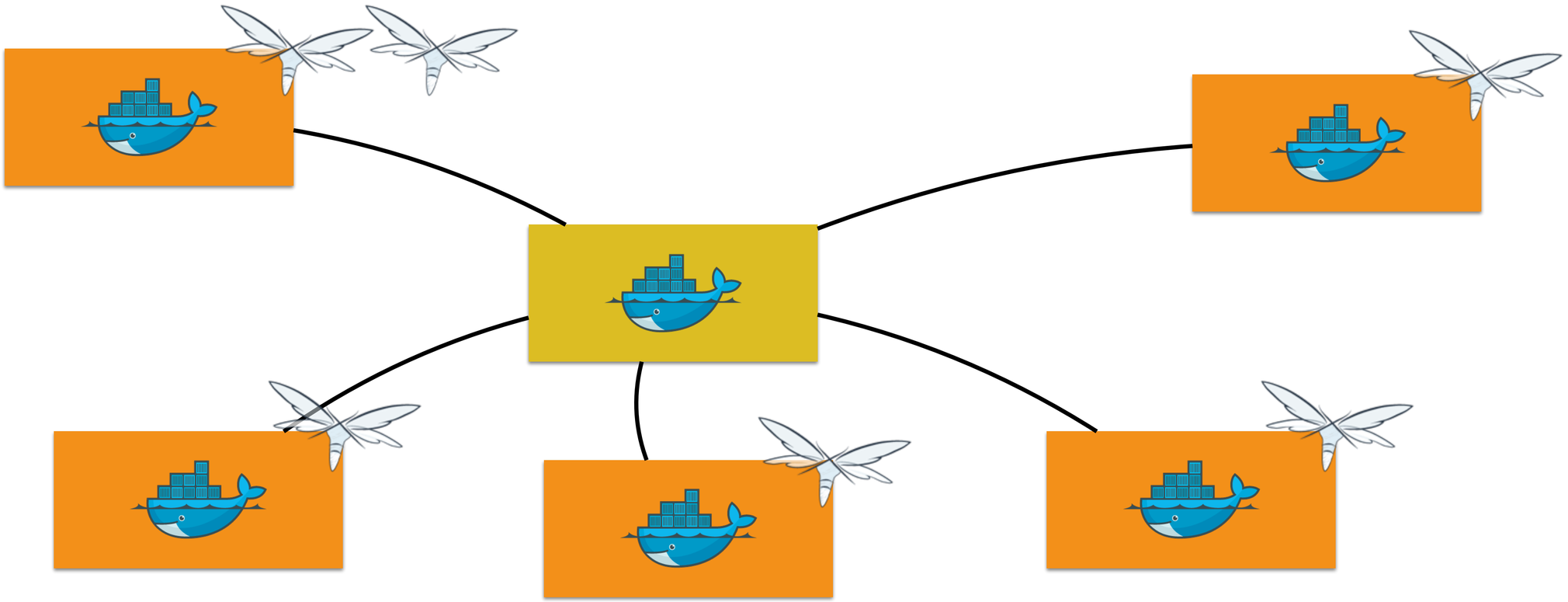
Swarm Mode: Desired != Actual



Swarm Mode: Reconcile

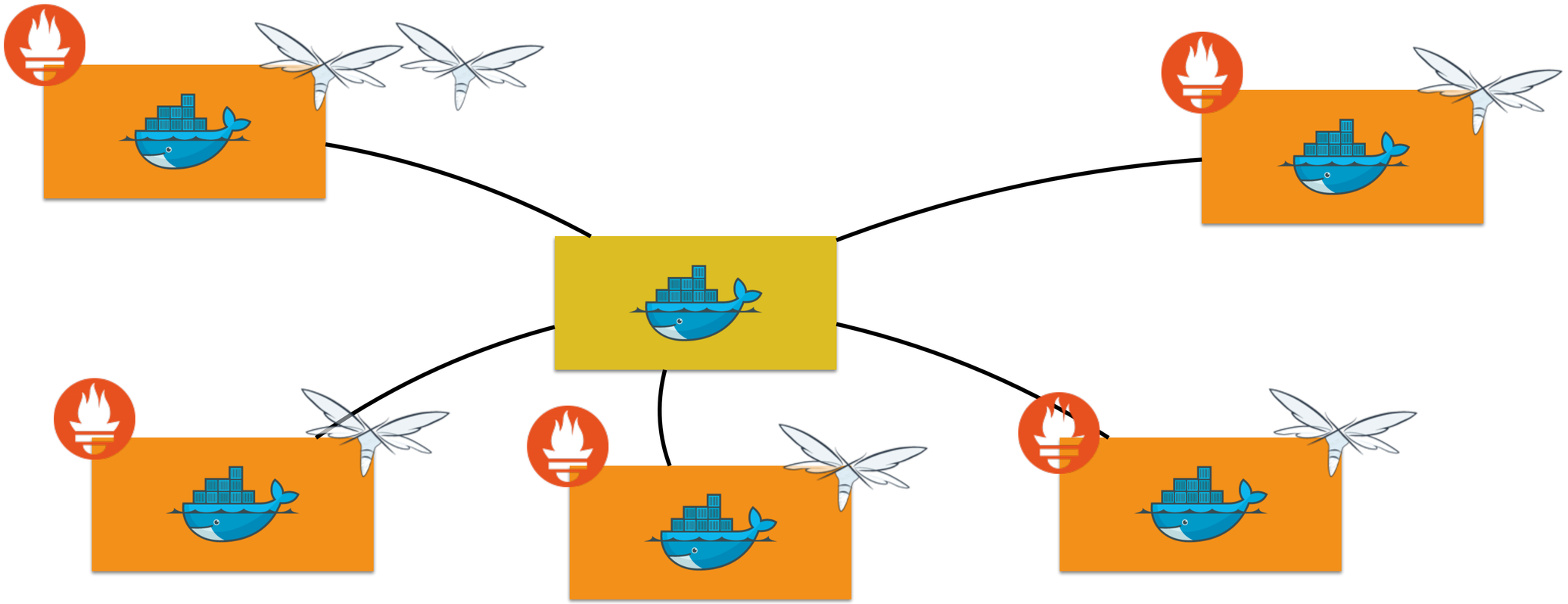


Swarm Mode: Scale



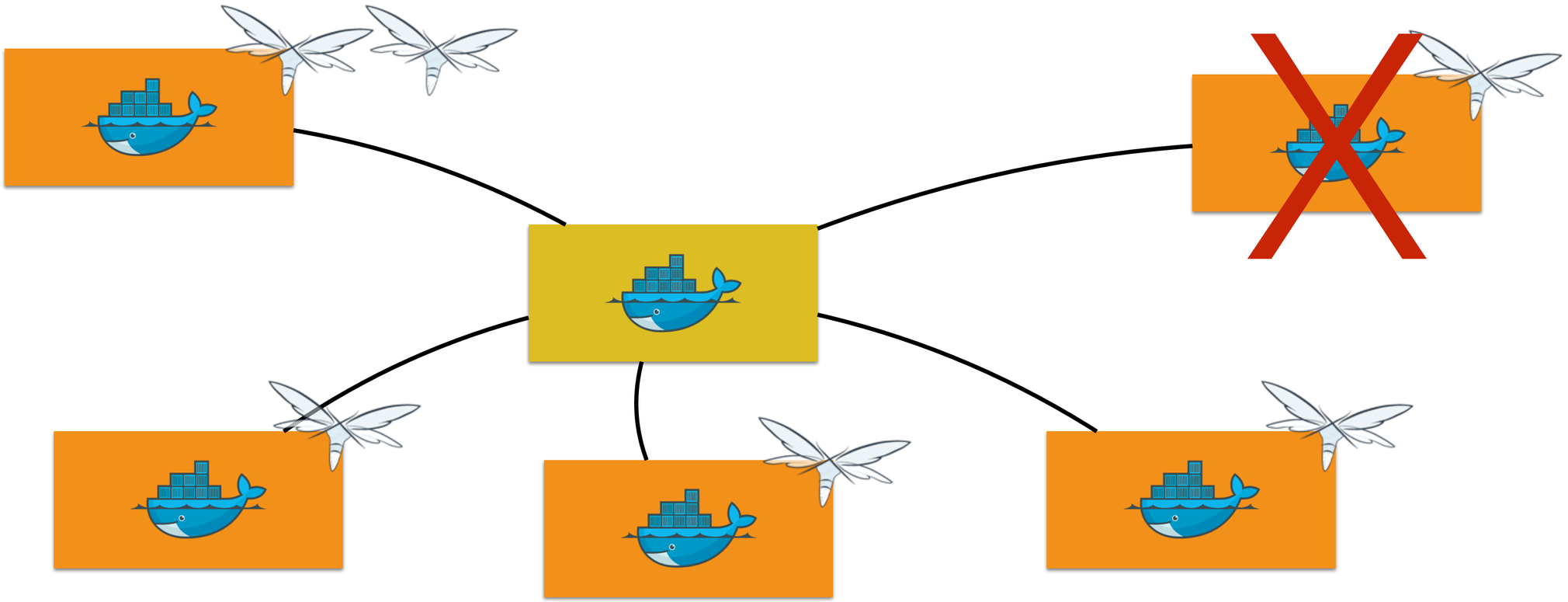
```
docker service scale web=6
```

Swarm Mode: Global Service



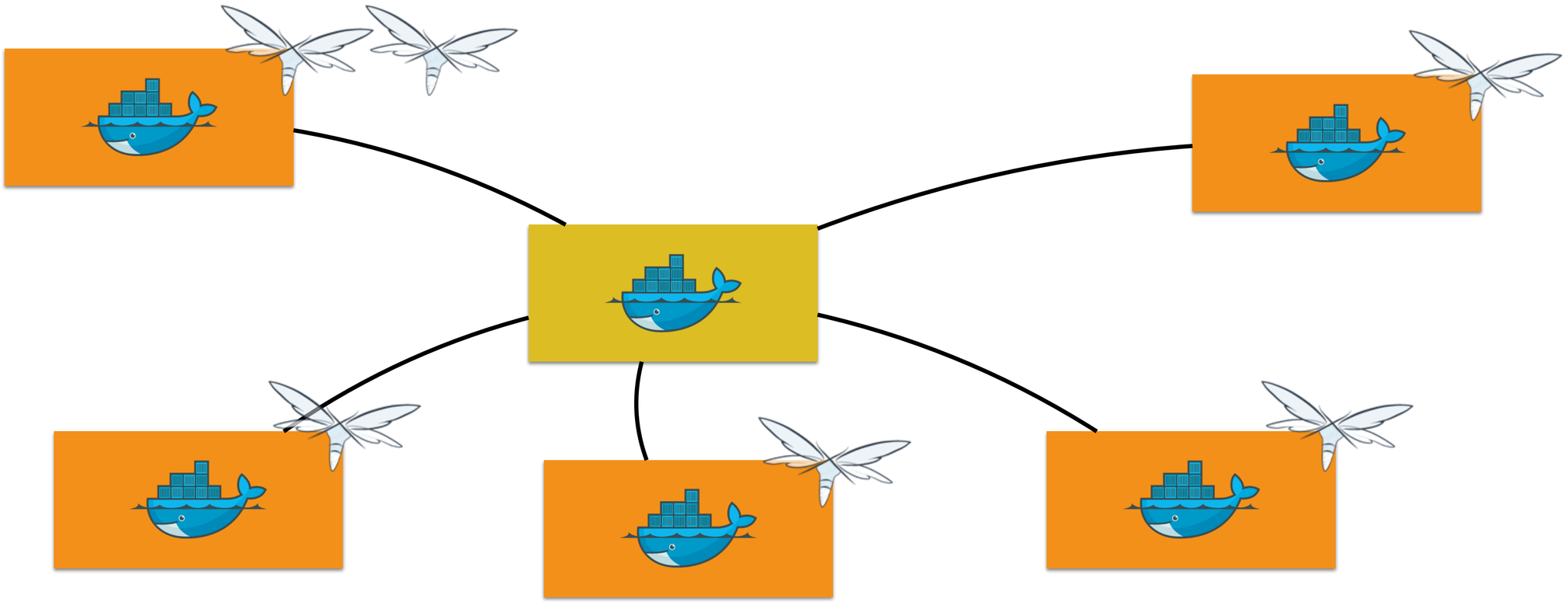
```
docker service create --mode=global --name=prom prom/prometheus
```

Swarm Mode: Pause Node



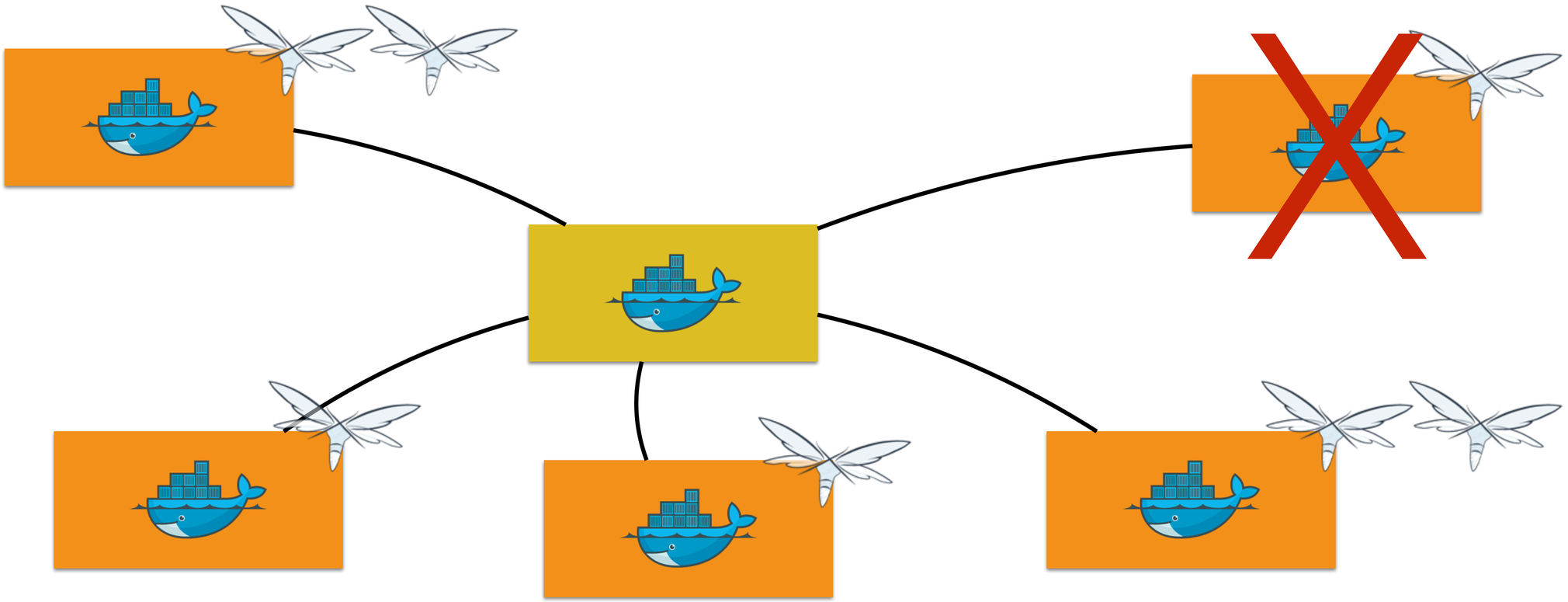
```
docker node update --availability pause <nodename>
```

Swarm Mode: Active Node



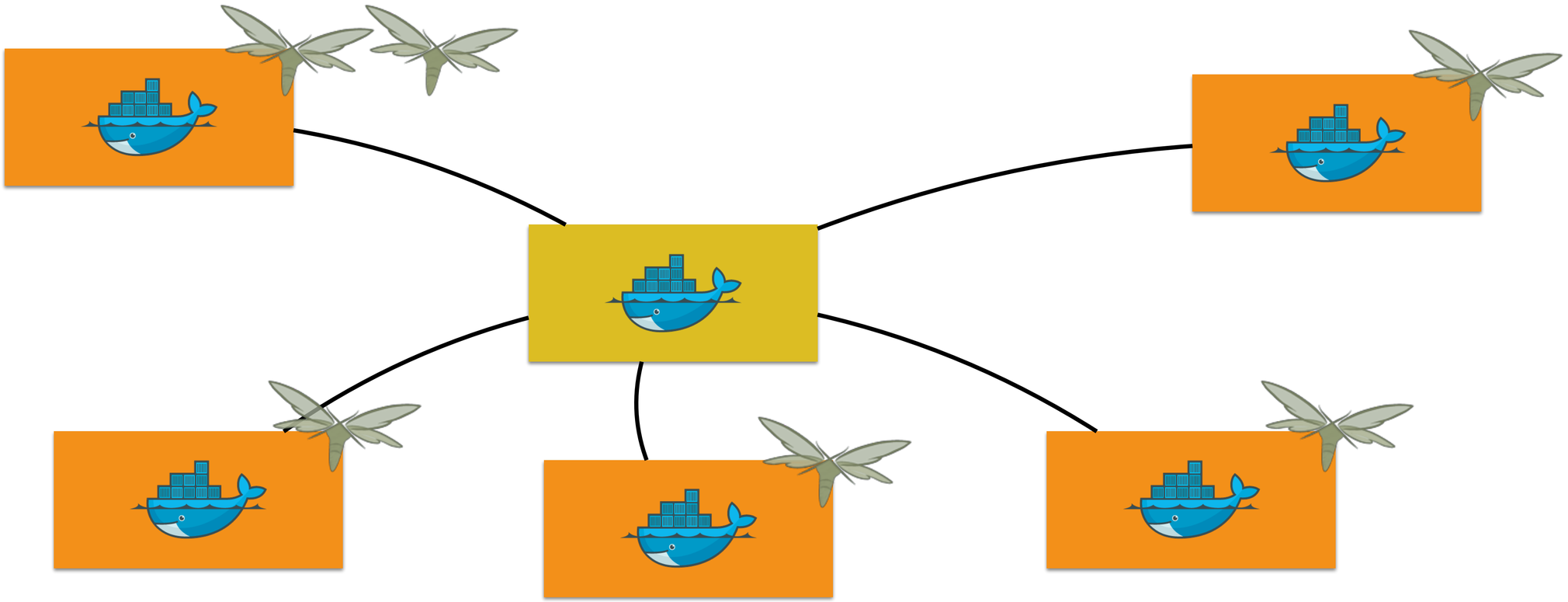
```
docker node update --availability active <nodename>
```

Swarm Mode: Drain Node



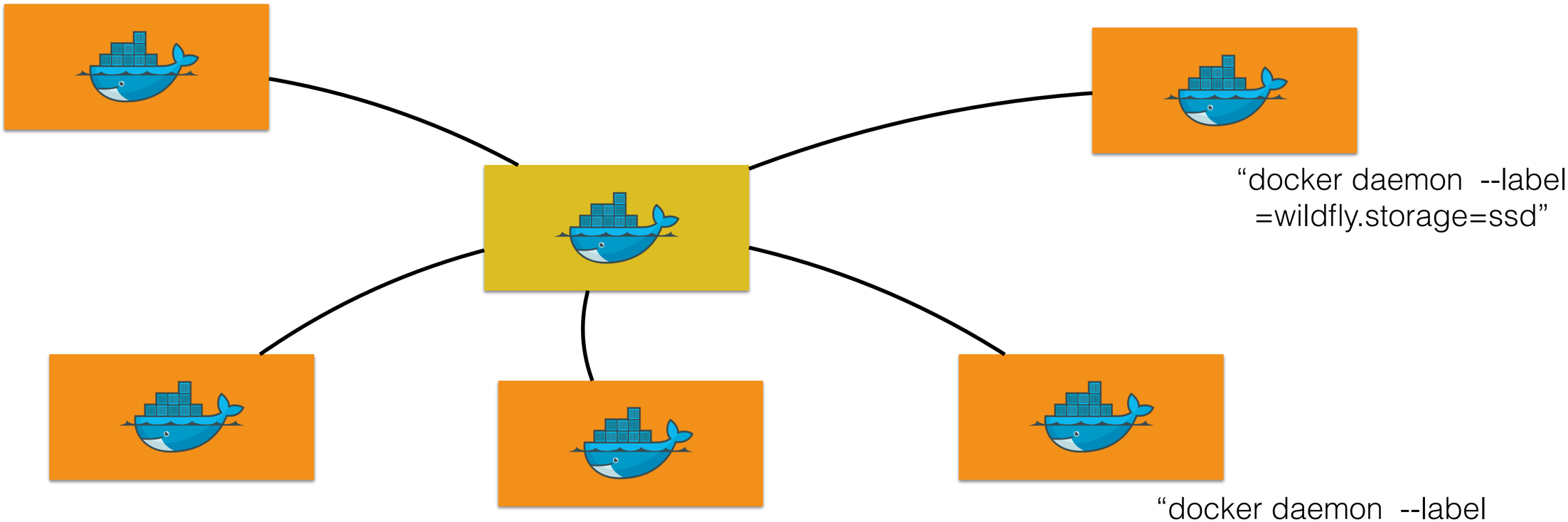
```
docker node update --availability drain <nodename>
```

Swarm Mode: Rolling Updates



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

Swarm Mode: Label

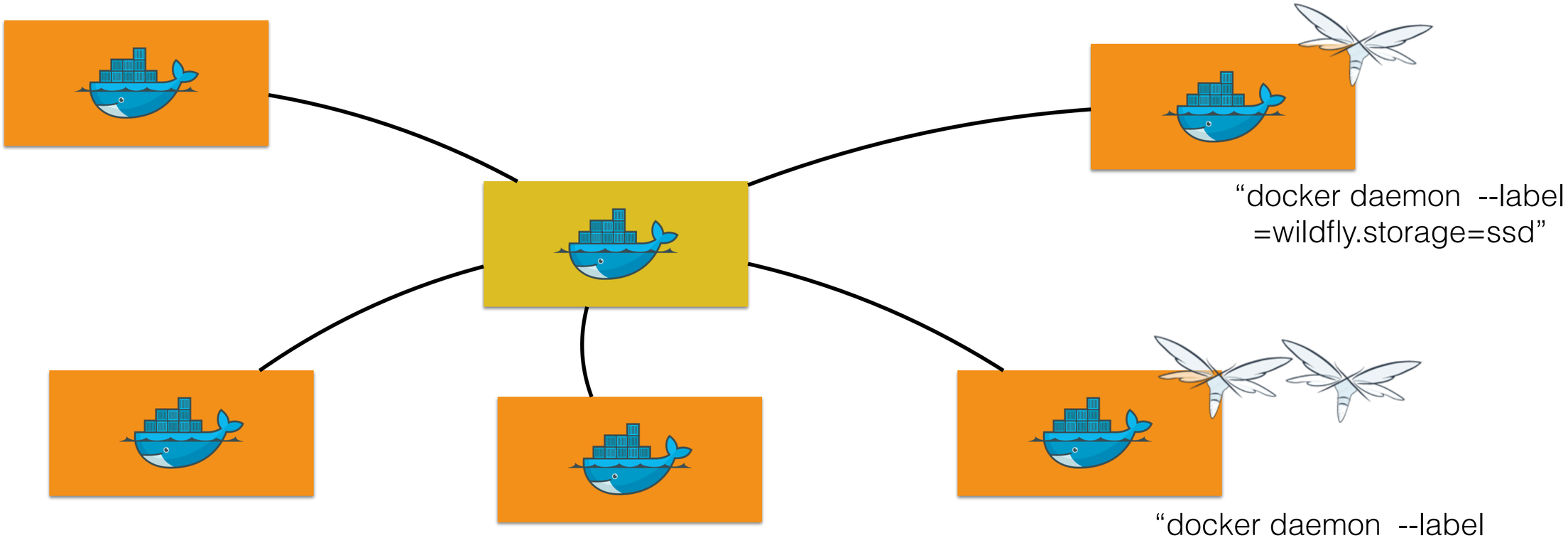


“docker daemon --label =wildfly.storage=ssd”

“docker daemon --label =wildfly.storage=ssd”

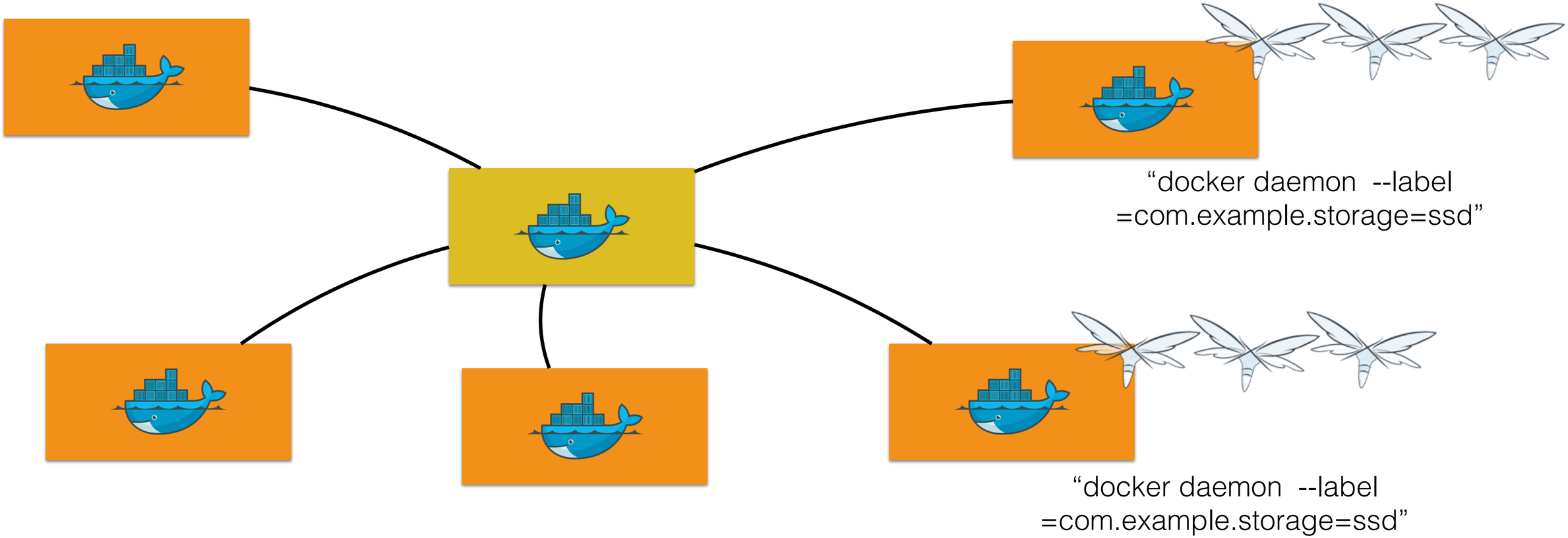
```
DOCKER_OPTS="--label=wildfly.storage=ssd"
```

Swarm Mode: Constraints



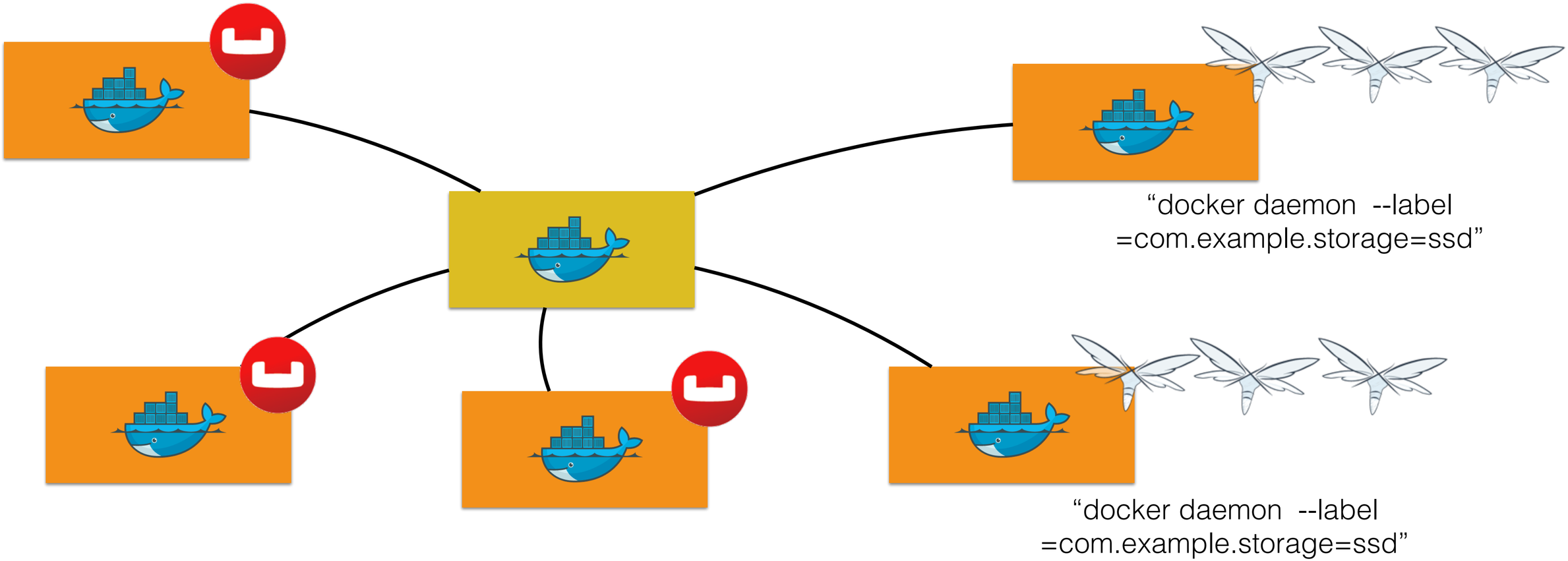

```
docker service create --replicas=3 --name=web --constraint
engine.labels.wildfly.storage==ssd jboss/wildfly
```

Swarm Mode: Constraints




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

Swarm Mode: Constraints




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

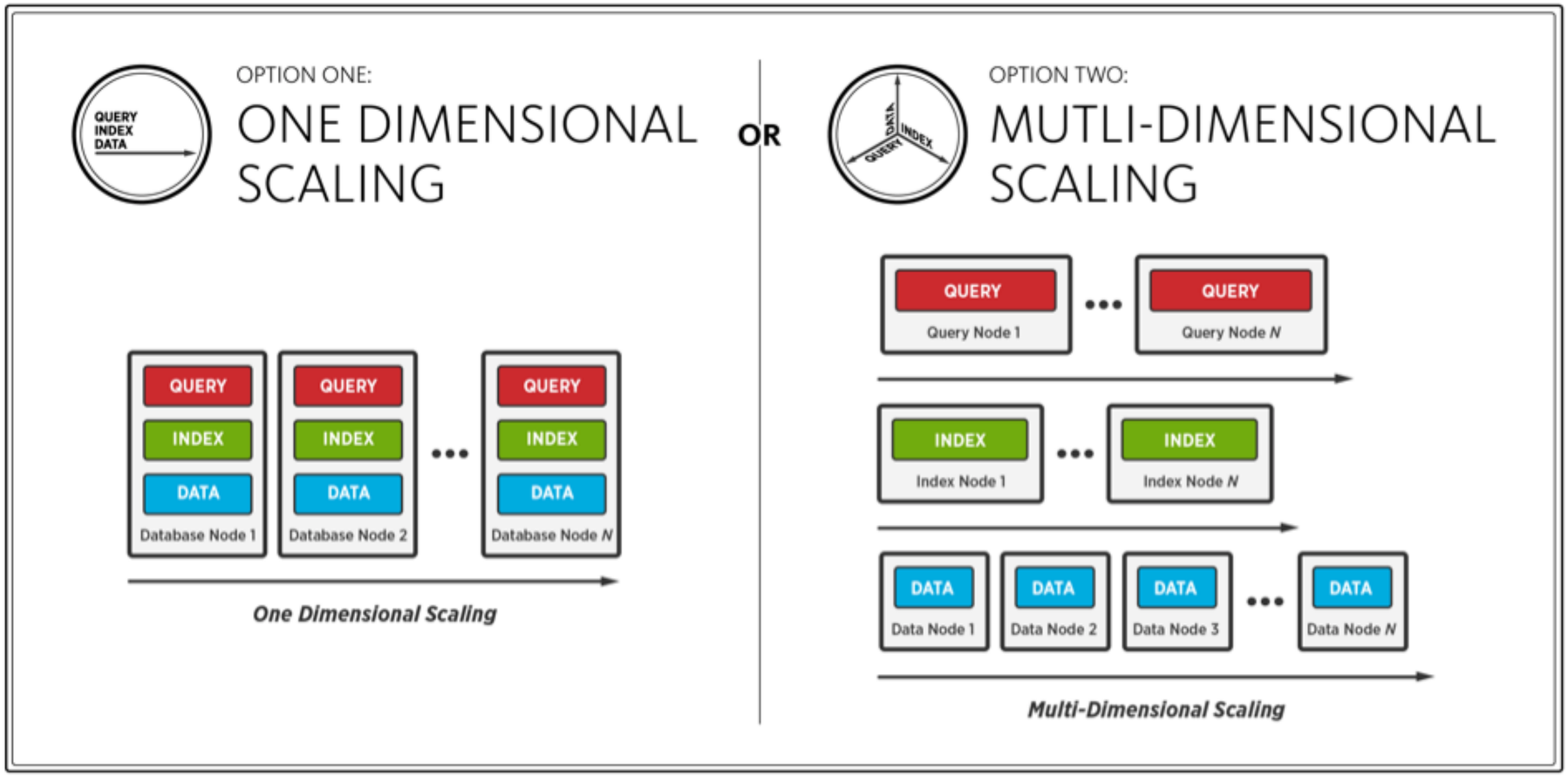
Scheduling Backends using Filters

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

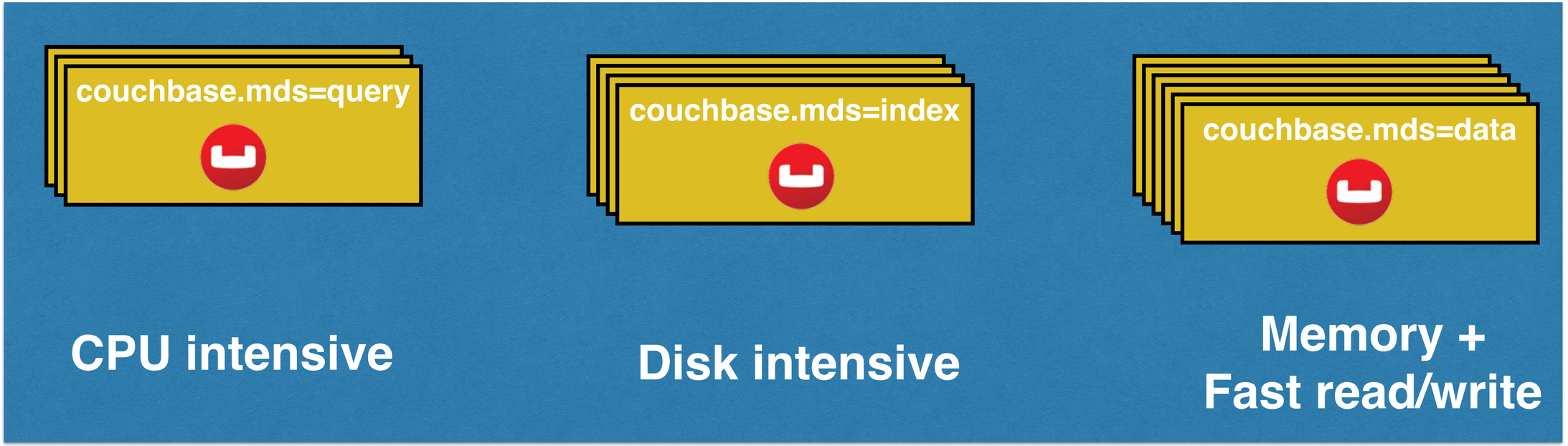
Node		
Constraint	Default or custom tags	`node`, `operatinsystem`, `kernelversion`, …
Health	Schedule containers on healthy nodes only	
Container Slots	Maximum number of containers on a node	`--labels containerslots=3`
Container		
Affinity	“Attraction” between containers	`-e affinity:container=<name>/<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.

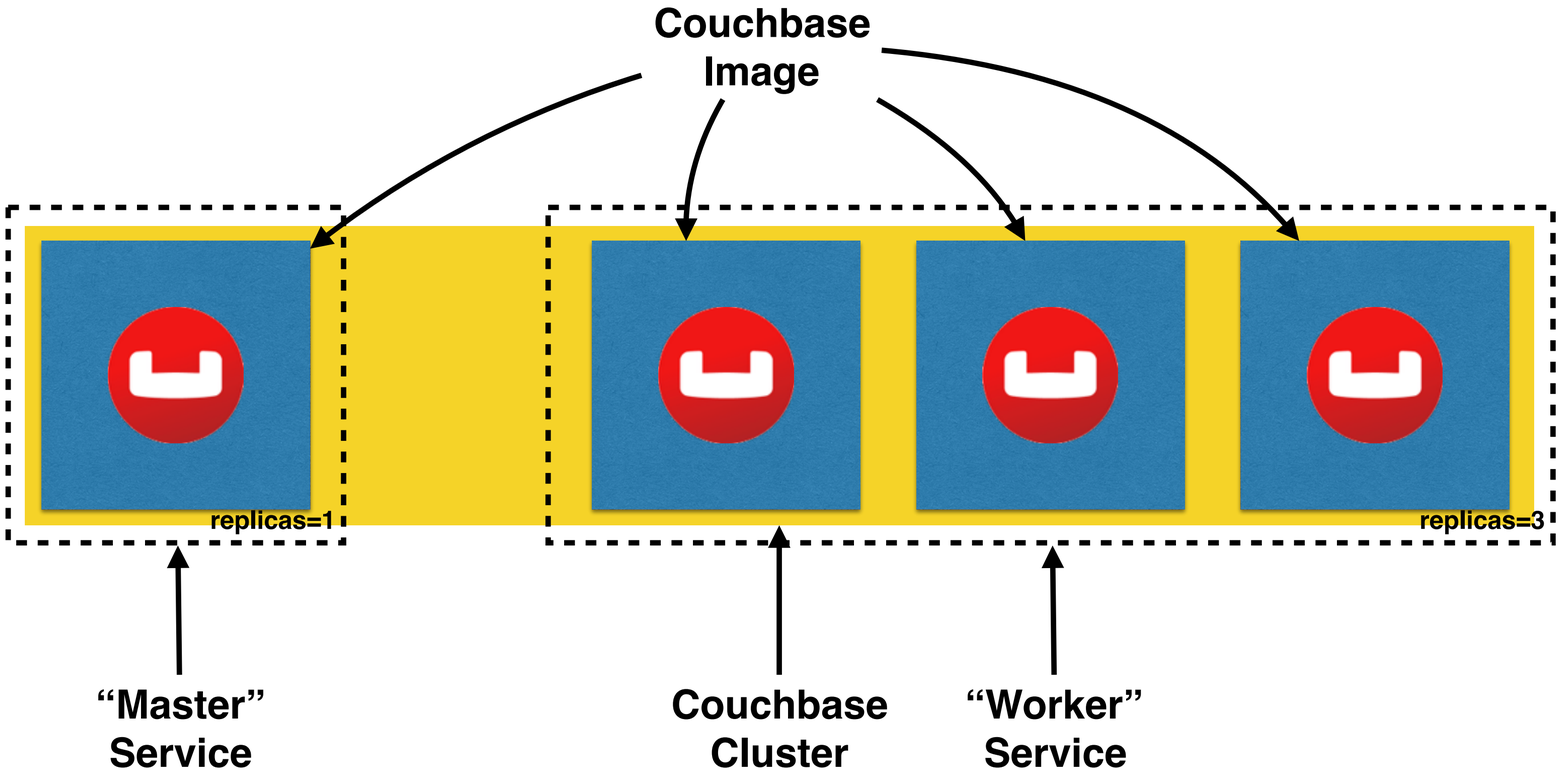


Optimal Utilization of Resources



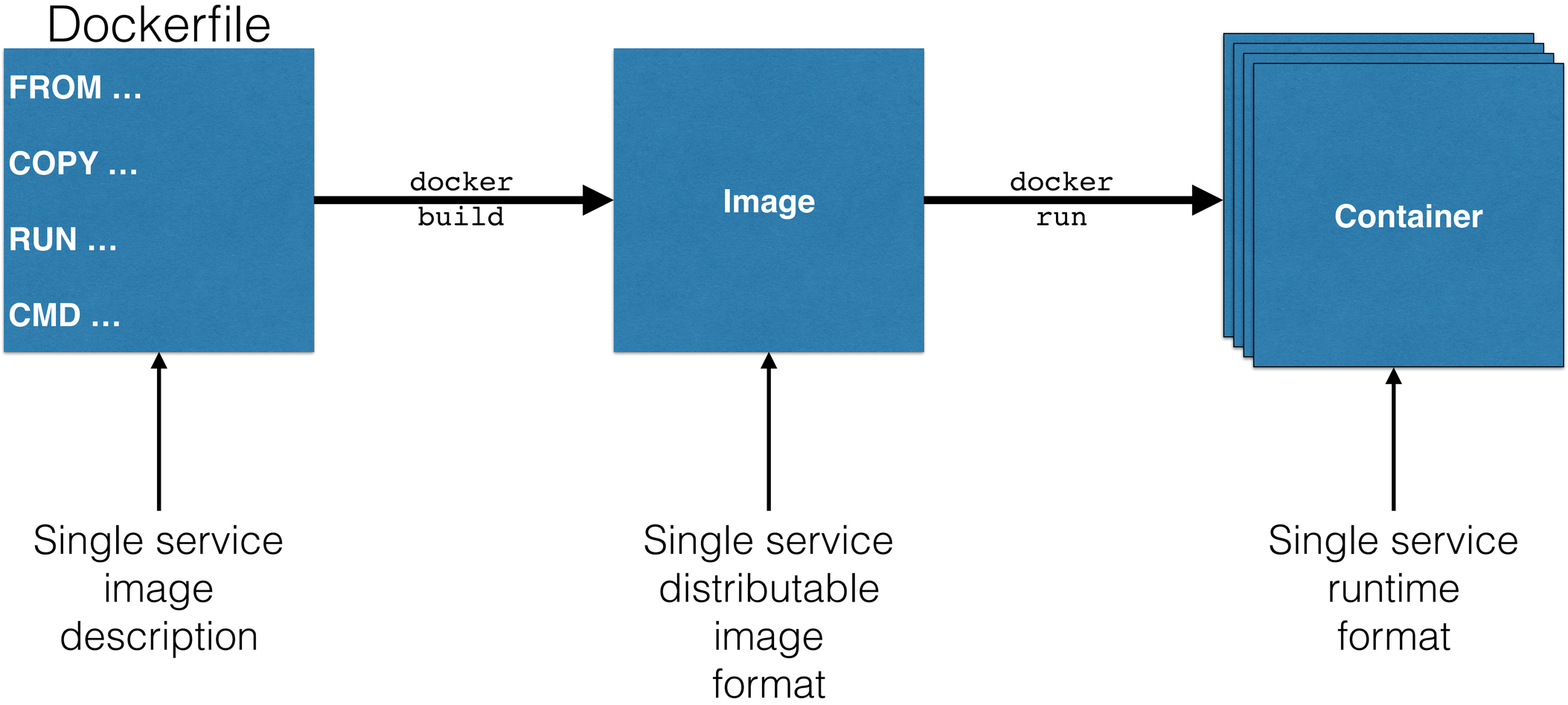
- **Attach labels**: `DOCKER_OPTS="--label.couchbase.mds=data"`
- **Run Containers**: `docker service create --constraint engine.labels.couchbase.mds==index couchbase`

Couchbase Cluster using Docker Services



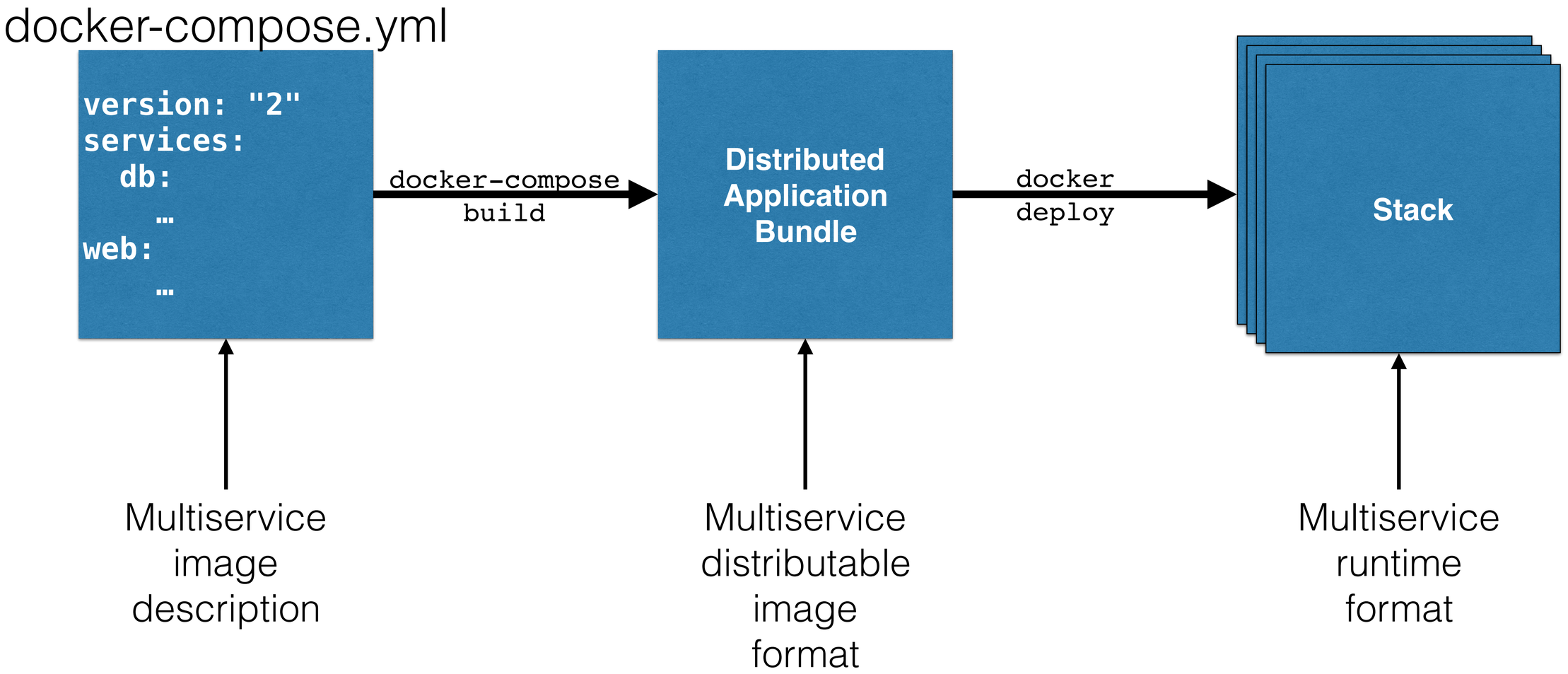
http://blog.couchbase.com/2016/september/docker-service-swarm-mode-couchbase-cluster

Docker Lifecycle



Experimental feature

Distributed Application Bundle

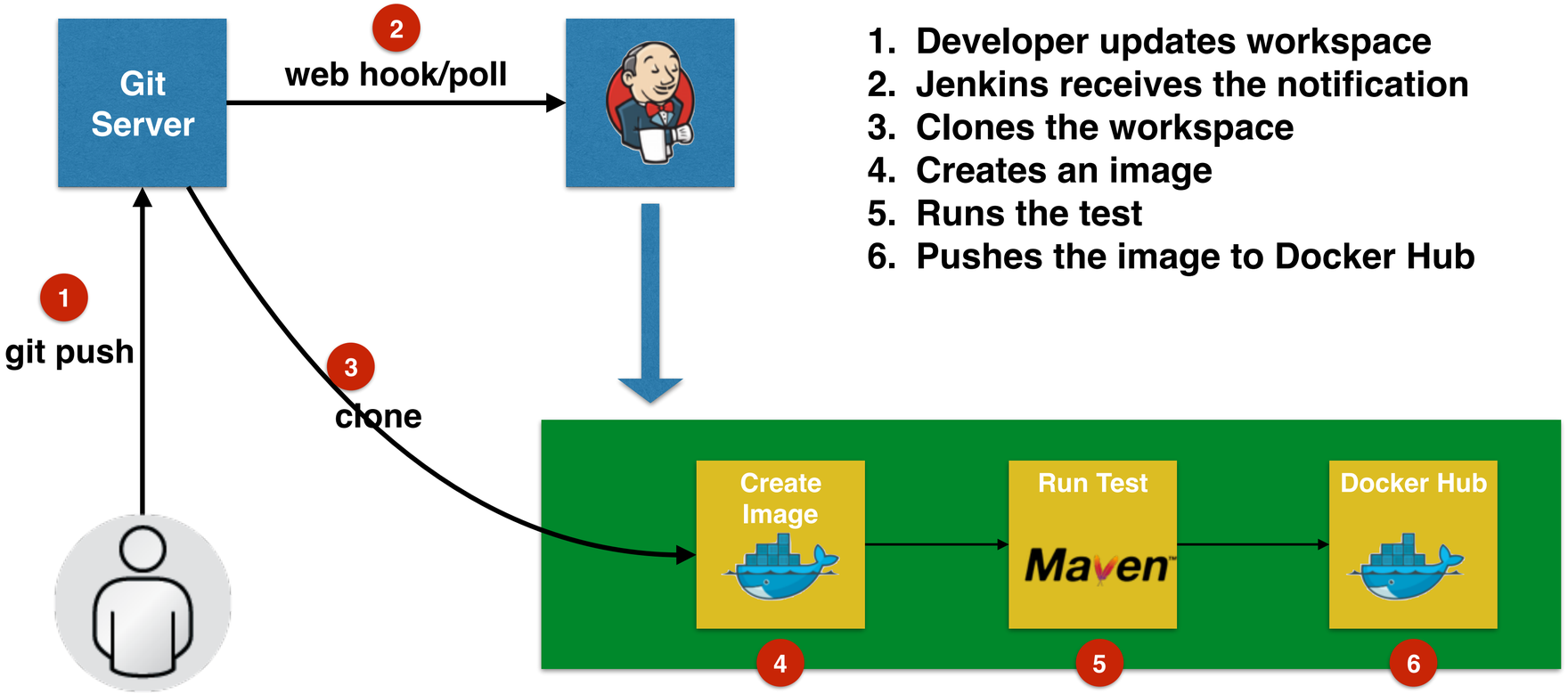


NOTE: Docker Engine and Registry do not support distribution of DABs

Monitoring Docker Containers

- **`docker stats`** command
  - LogEntries
- Service logs: **`docker service logs <service>`**
- Prometheus endpoint - New in Docker 1.13
- Docker Remote API: **`/container/{container-name|cid}/stats`**
- Docker Universal Control Plane
- cAdvisor
  - Prometheus
  - InfluxDB

CI/CD with Docker + Jenkins



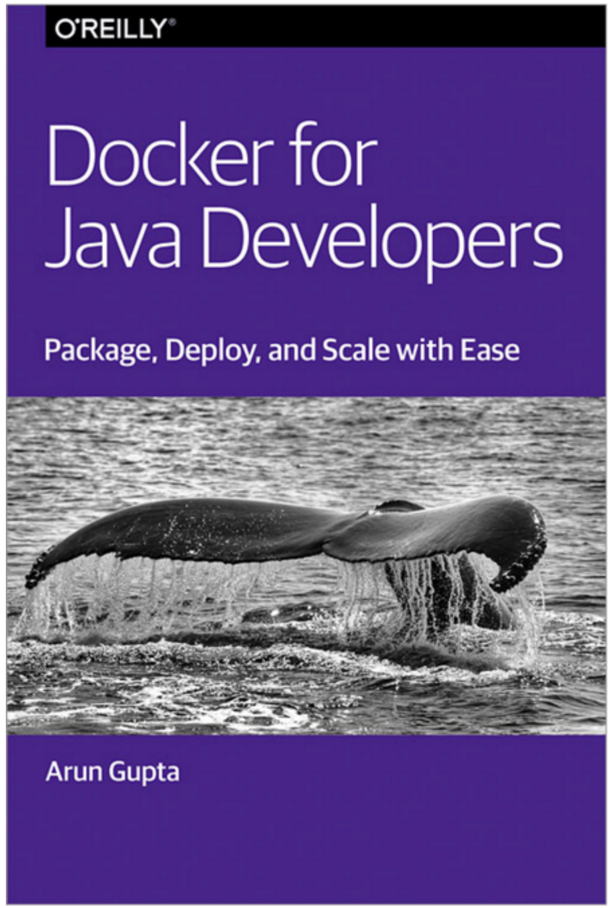
1. Developer updates workspace
2. Jenkins receives the notification
3. Clones the workspace
4. Creates an image
5. Runs the test
6. Pushes the image to Docker Hub

http://blog.couchbase.com/2016/september/deployment-pipeline-docker-jenkins-java-couchbase

Docker Support in Java IDEs

https://github.com/arun-gupta/docker-java-ides

bit.ly/dockerjava



bit.ly/kubejava

References

- Slides: github.com/docker/labs/tree/master/slides
- Workshop: github.com/docker/labs/tree/master/java
- Docs: docs.docker.com