`docker service` is the new `docker run`

Getting Started with Docker Clustering

Mike Goelzer / mgoelzer@docker.com / @mgoelzer
Docker Inc.

docker

`docker service` is the new `docker run`

2013-14

```
docker run nginx
```

2014-15

```
docker run -p 3375:2375 swarm ;
docker run -H :3375 nginx
```

2016

Swarm Mode in Docker Engine

```
docker swarm init ;
docker service create nginx
```

docker

Features Walkthrough

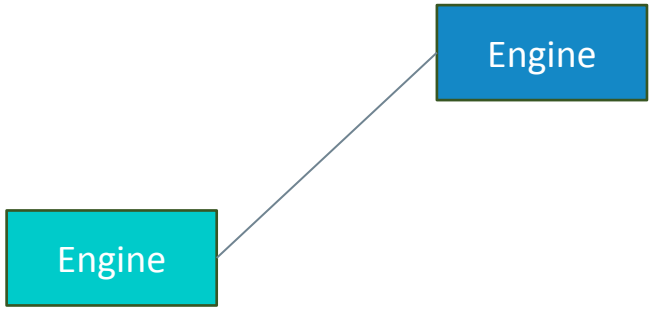
Swarm Mode

Engine

- `$ docker swarm init`

docker

Swarm Mode

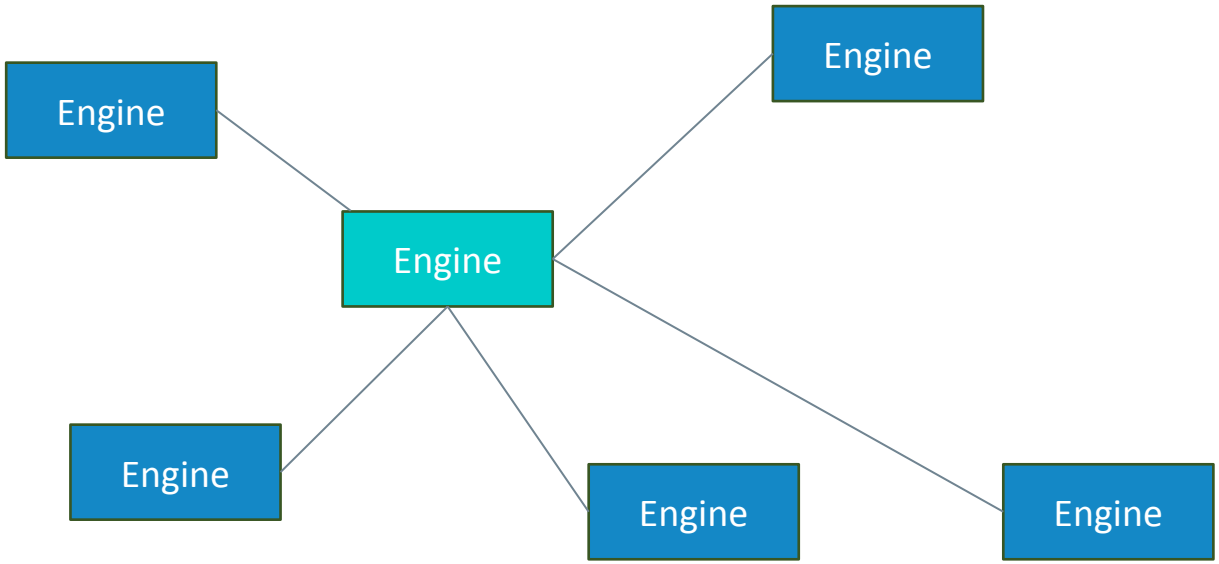



```
$ docker swarm init
```

```
$ docker swarm join <IP of manager>:2377
```

docker

Swarm Mode

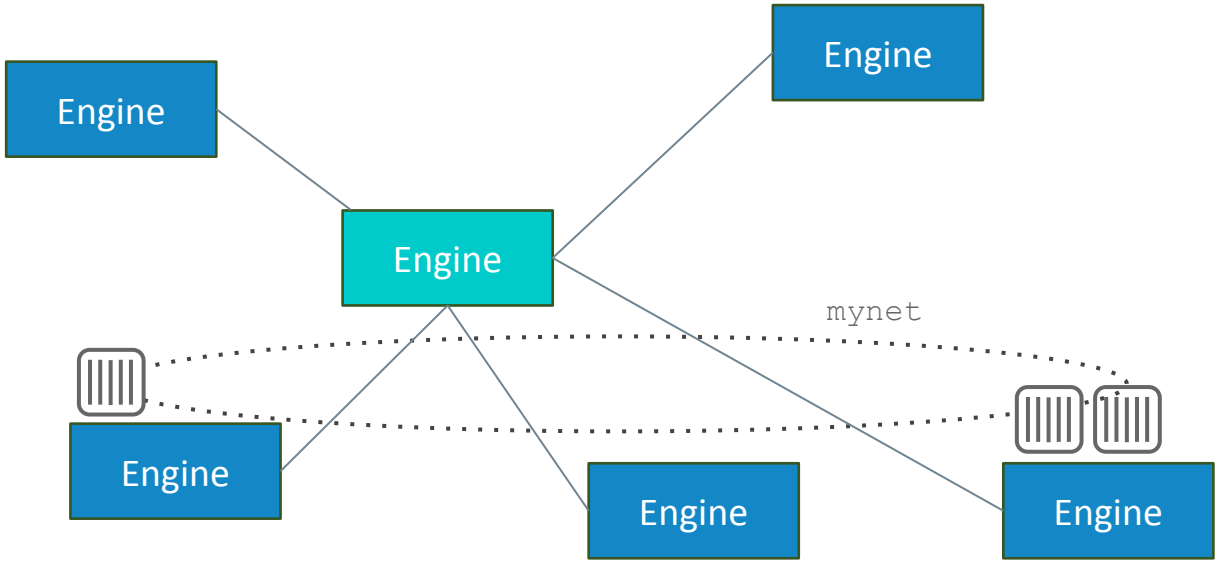



```
$ docker swarm init
```

```
$ docker swarm join <IP of manager>:2377
```

docker

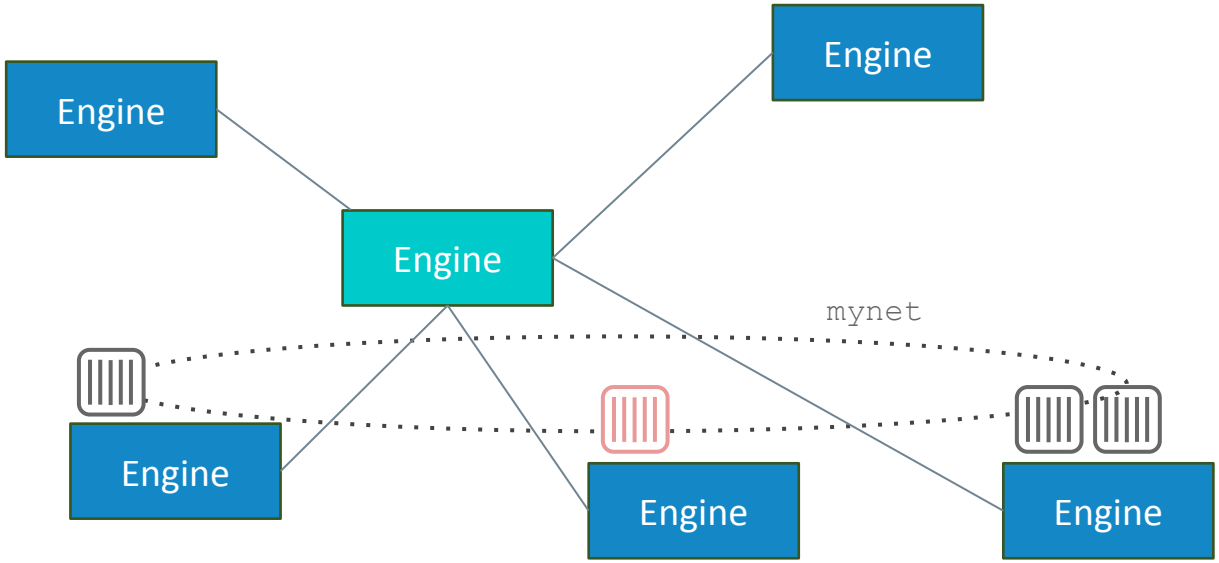
Services




```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 frontend:latest
```

docker

Services

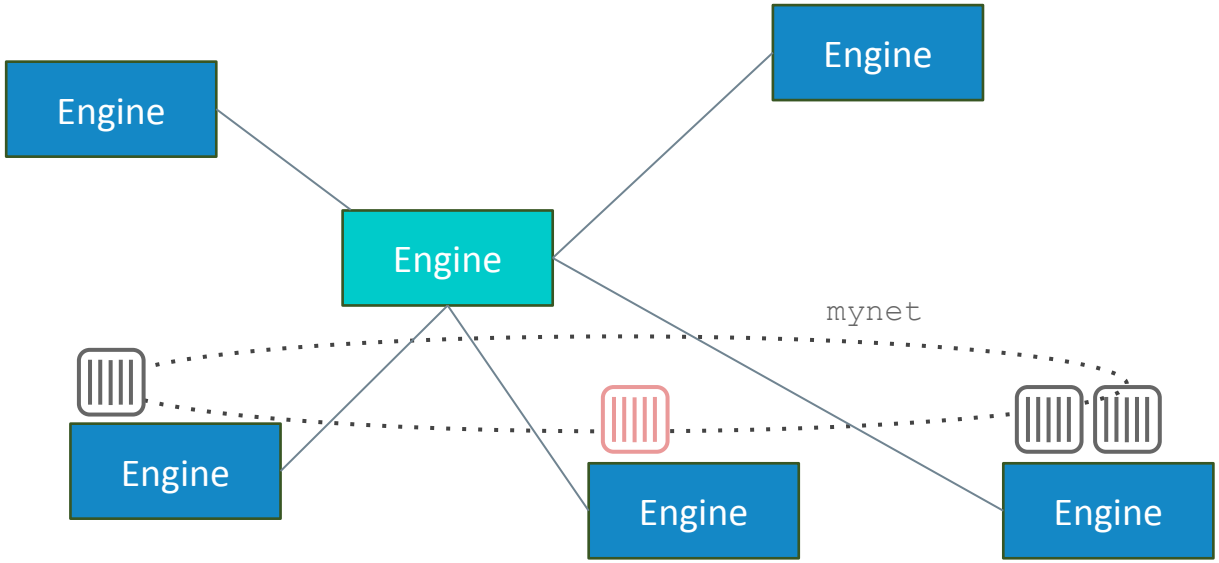



```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 frontend:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

docker

Node Failure

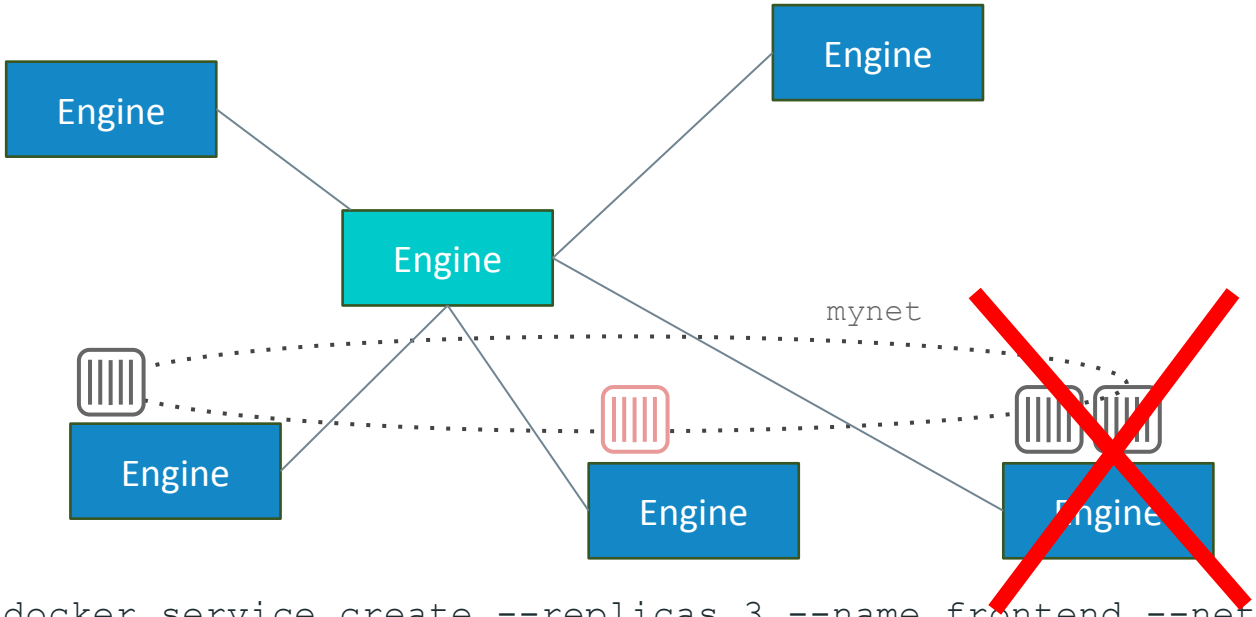



```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 frontend:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

docker

Node Failure

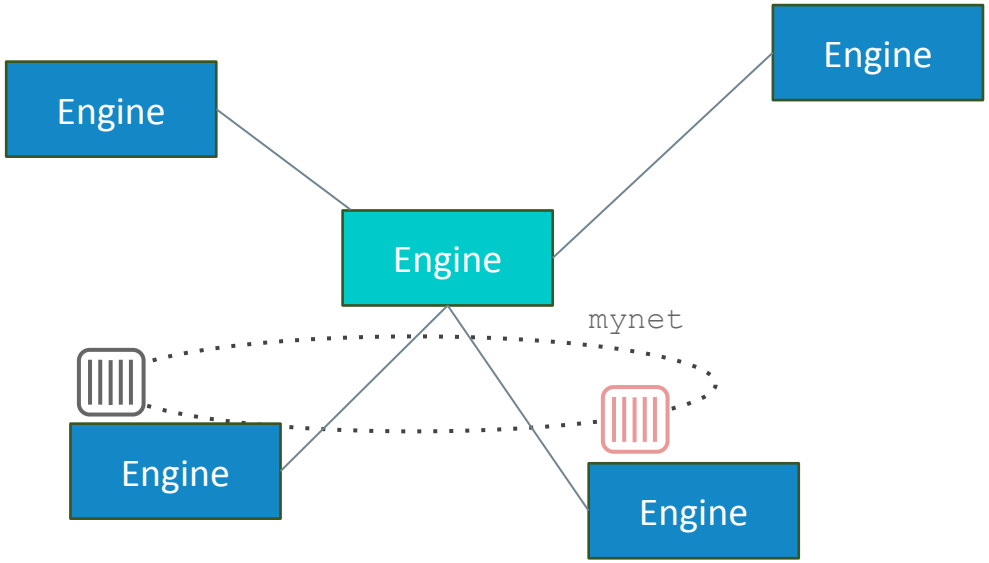



```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 frontend:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

docker

Desired State ≠ Actual State

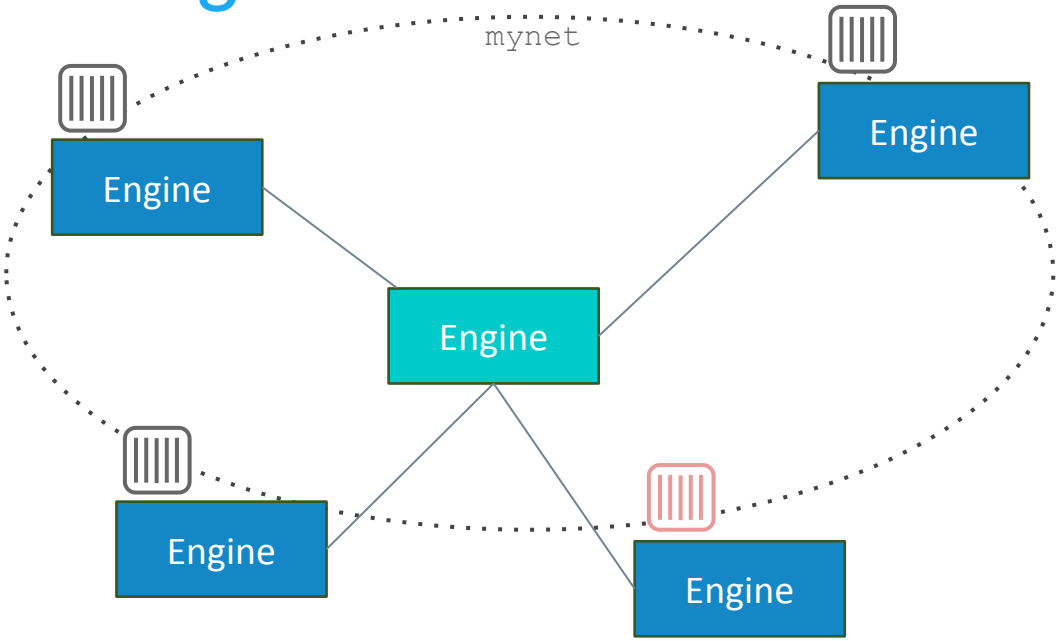



```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 frontend:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

docker

Converge Back to Desired State

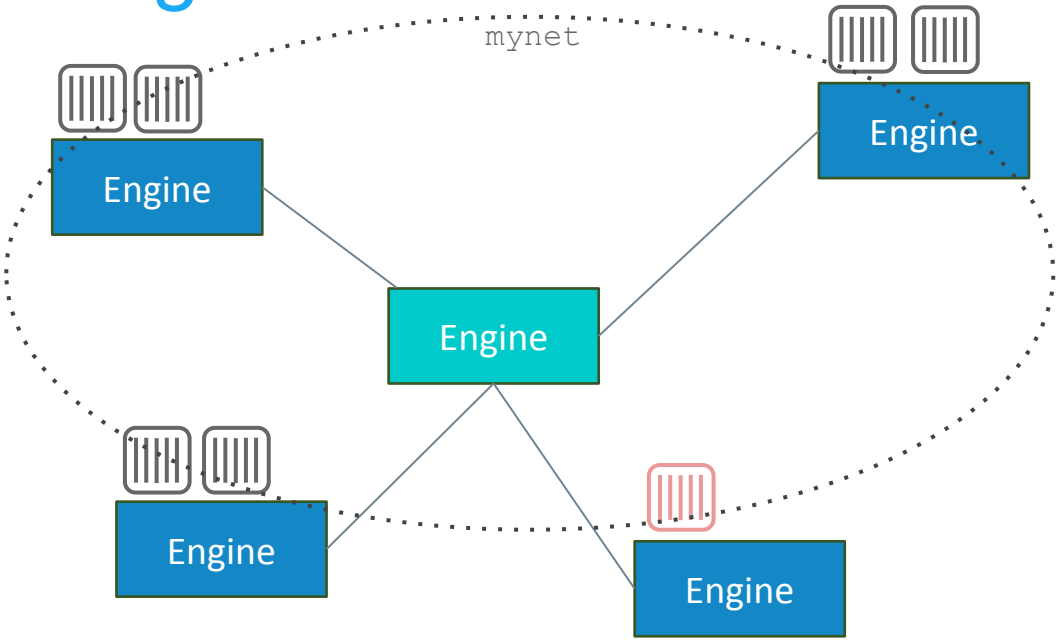



```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 frontend:latest
```

```
$ docker service create --name redis --network mynet redis:latest
```

docker

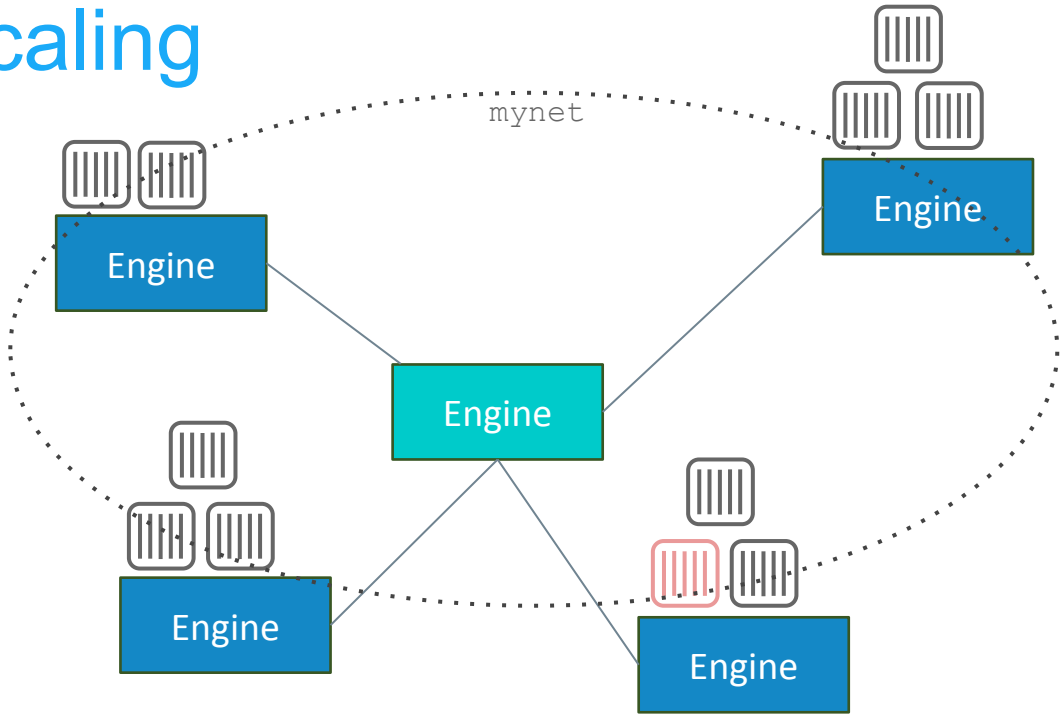
Scaling




```
$ docker service update --replicas 6 frontend
```

docker

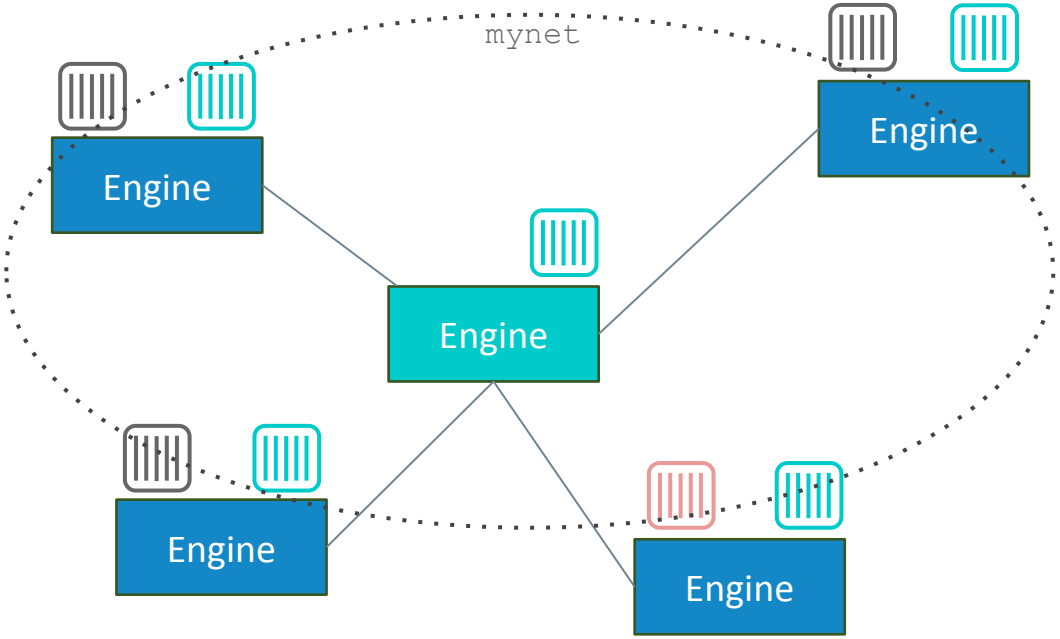
Scaling




```
$ docker service update --replicas 10 frontend
```

docker

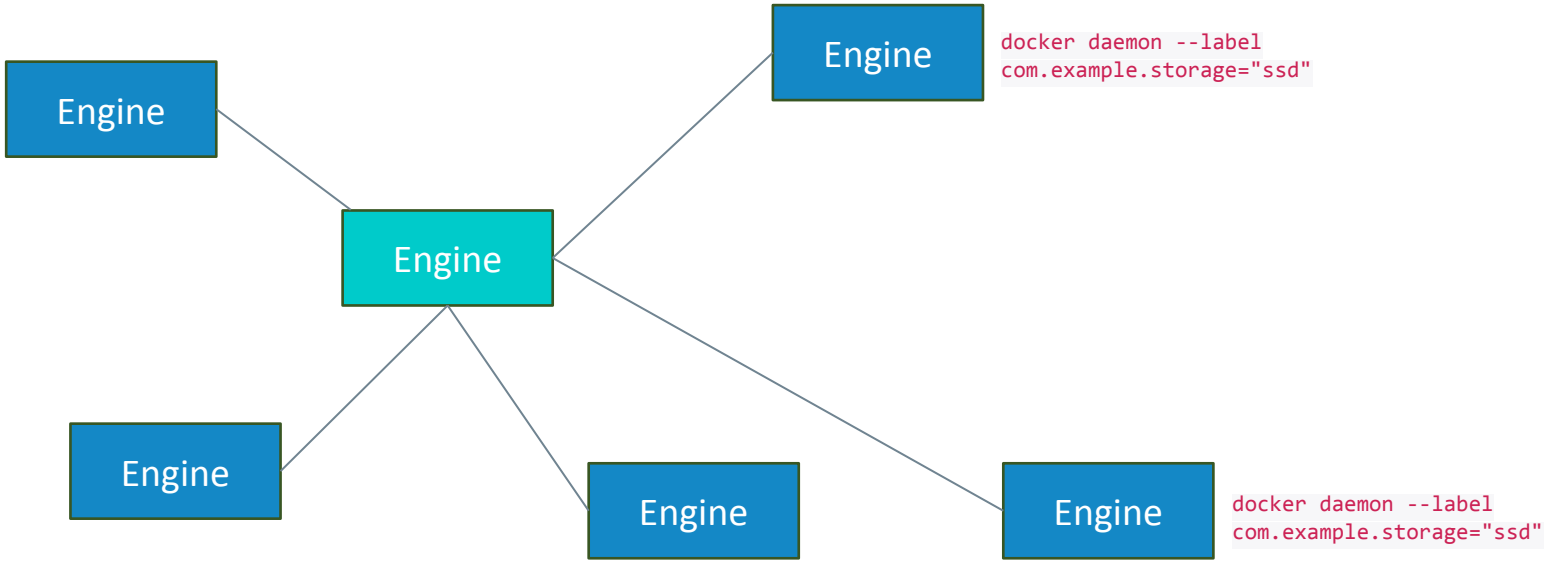
Global Services




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

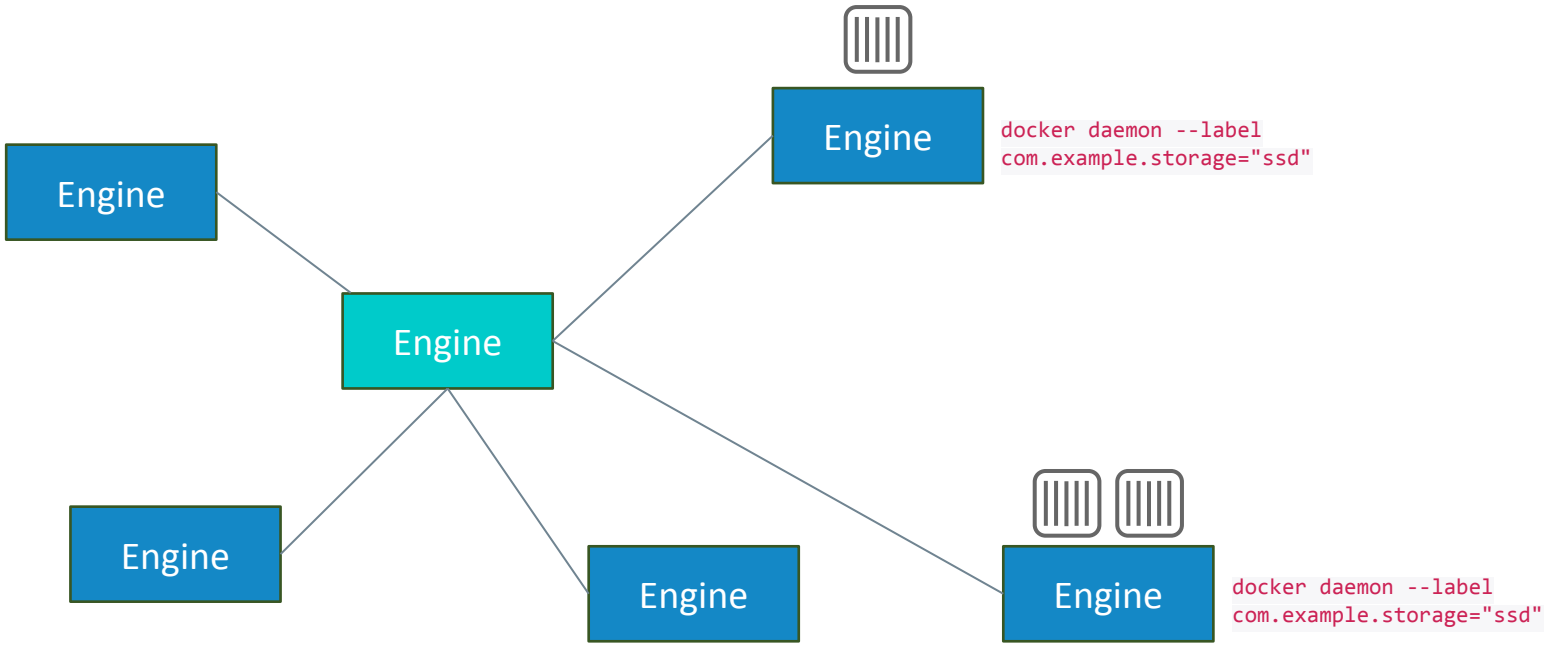
docker

Constraints



docker

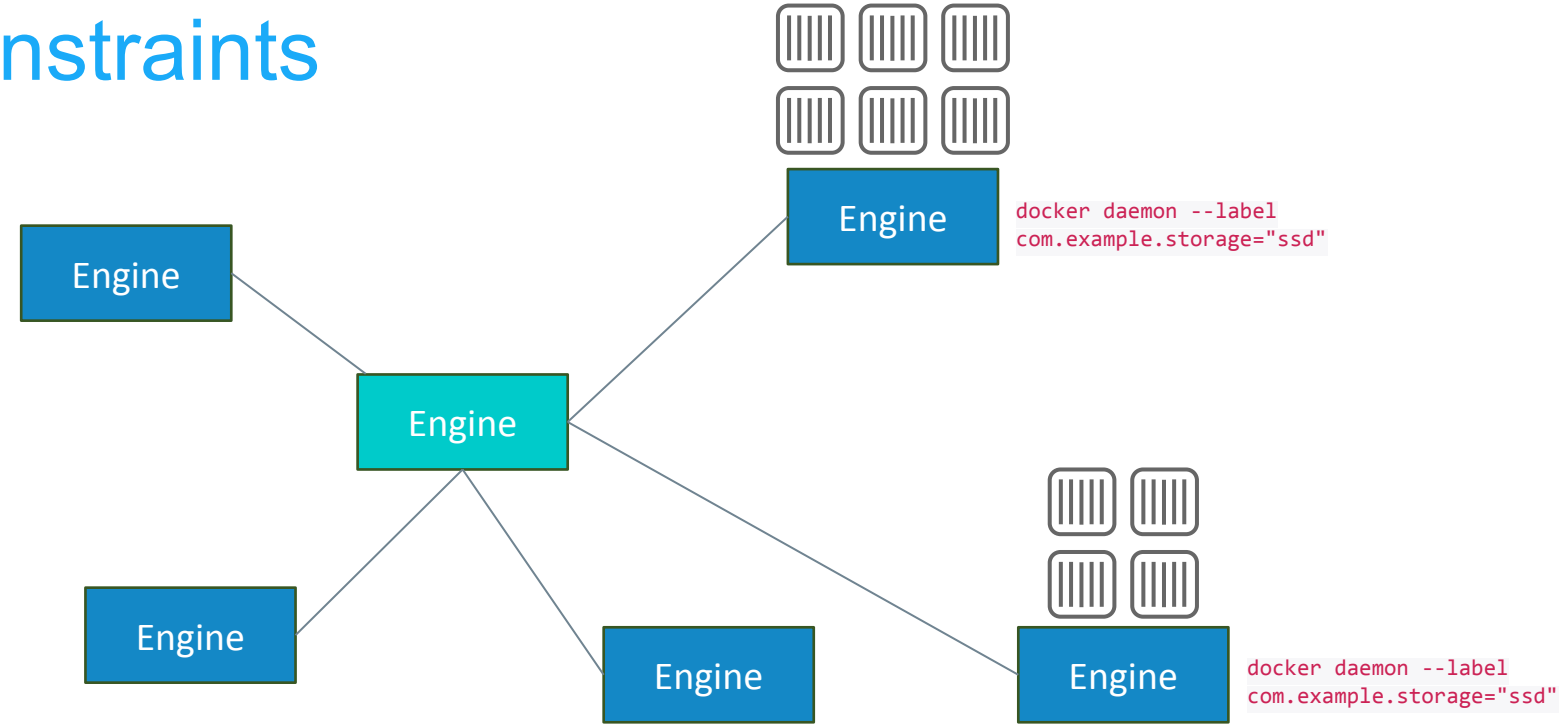
Constraints




```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 --constraint engine.labels.com.example.storage==ssd
frontend:latest
```

docker

Constraints




```
$ docker service create --replicas 3 --name frontend --network mynet
 -p 8080:80 --constraint engine.labels.com.example.storage==ssd
frontend:latest
$ docker service update --replicas 10 frontend
```

docker

Container Health Check in `Dockerfile`

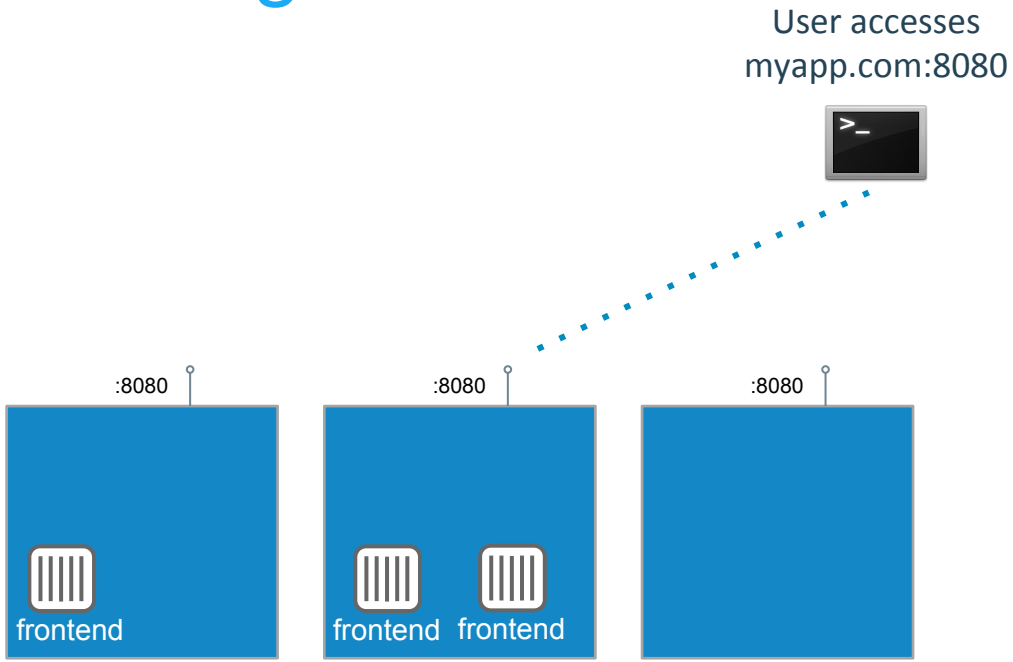
```
HEALTHCHECK --interval=5m --timeout=3s
  --retries 3
  CMD curl -f http://localhost/ || exit 1
```

Check web server every 5 minutes, require < 3 sec latency.

>= 3 consecutive failures sets unhealthy state

Coming soon: health checks in official images

Routing Mesh

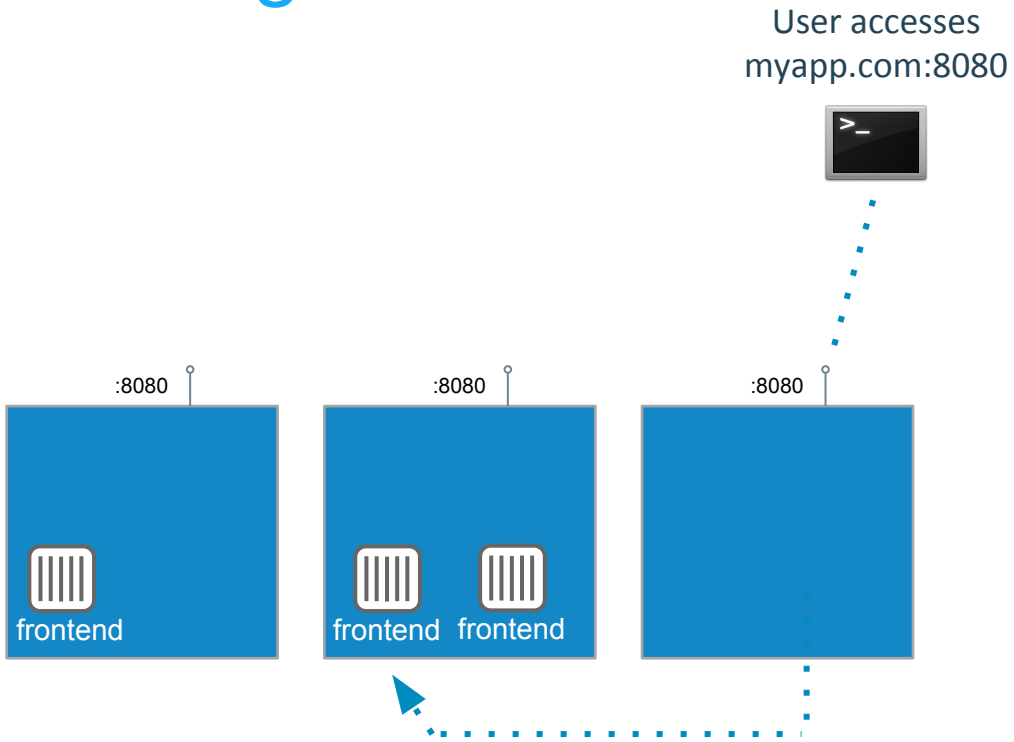


- Operator reserves a swarm-wide ingress port (8080) for `myapp`
- Every node listens on 8080
- Container-aware routing mesh can transparently reroute traffic from Worker3 to a node that is running container
- Built in load balancing into the Engine
- DNS-based service discovery

```
$ docker service create --replicas 3 --name frontend --network mynet
  -p 8080:80 frontend:latest
```

docker

Routing Mesh: Published Ports

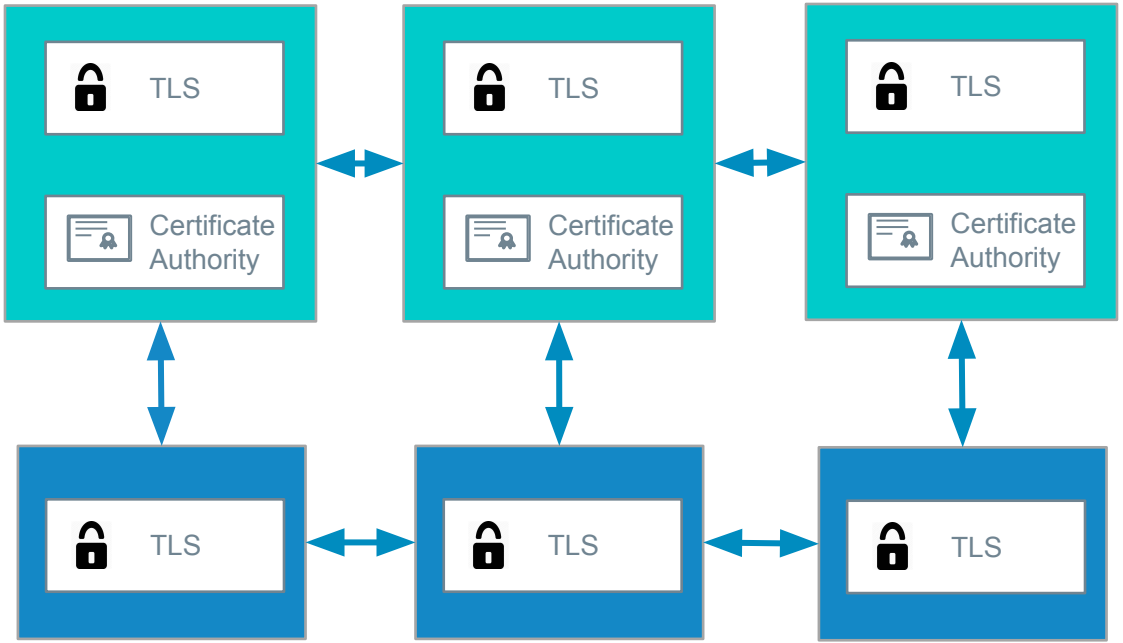


- Operator reserves a swarm-wide ingress port (8080) for `myapp`
- Every node listens on 8080
- Container-aware routing mesh can transparently reroute traffic from third node to a node that is running container
- Built in load balancing into the Engine
- DNS-based service discovery

```
$ docker service create --replicas 3 --name frontend --network mynet
  -p 8080:80 frontend_image:latest
```

docker

Secure by default with end-to-end encryption



- Out-of-the-box TLS encryption and mutual auth
- Automatic cert rotation
- External or self-signed root CA
- Cryptographic node identity

docker

Scale: 2,000 Nodes and Counting

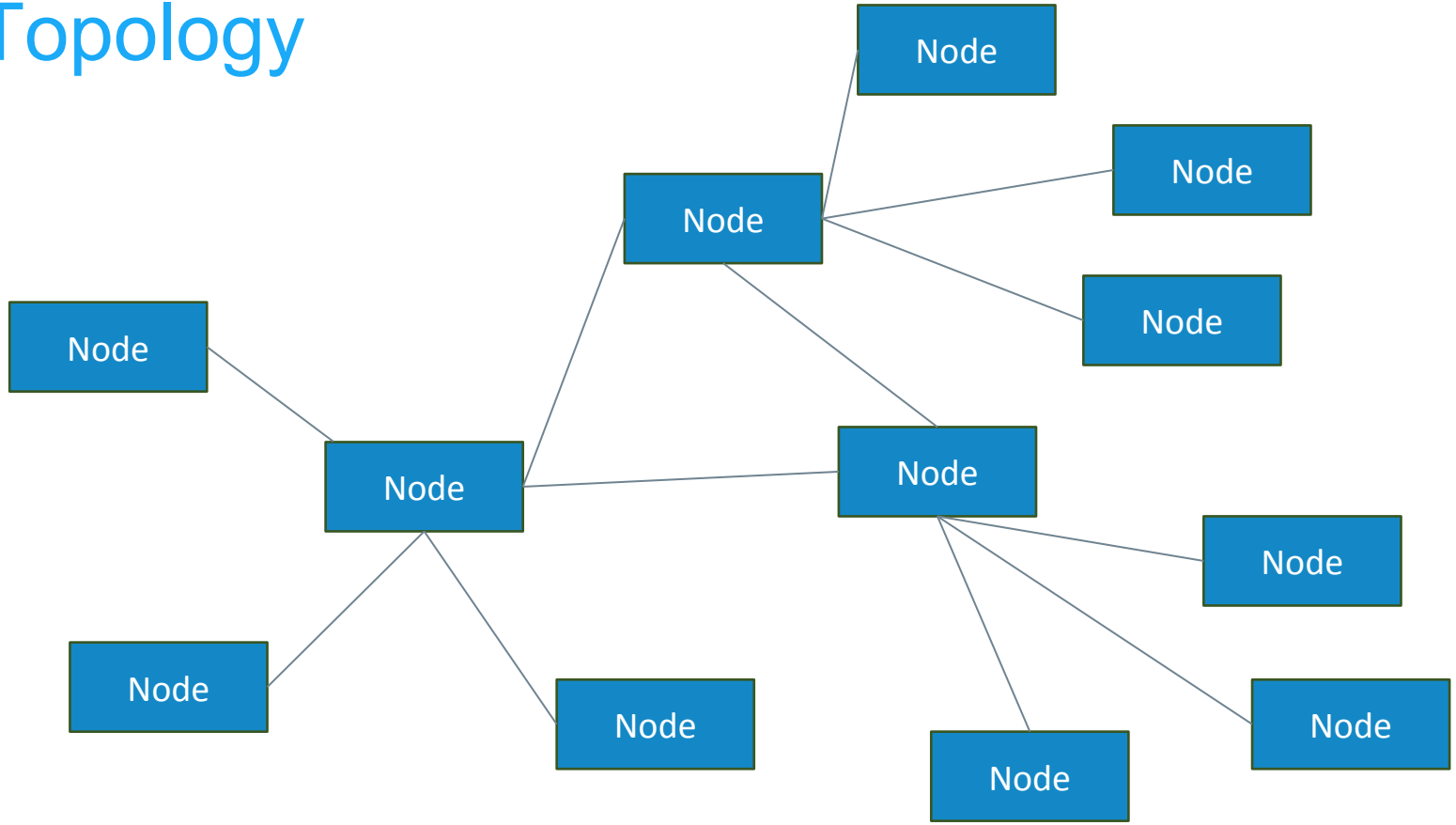
- For now: community testing, crowd-sourced nodes, not funded by Docker
- Credit to: Chanwit Kaewkasi, Suranaree University of Technology (SUT), Thailand

@chanwit

- Results:
  - 2,384 nodes
  - 96,287 containers
  - Manager CPU/memory ≲15%
  - Test stopped because 3rd-party monitoring failed
- https://github.com/swarm2k/swarm2k

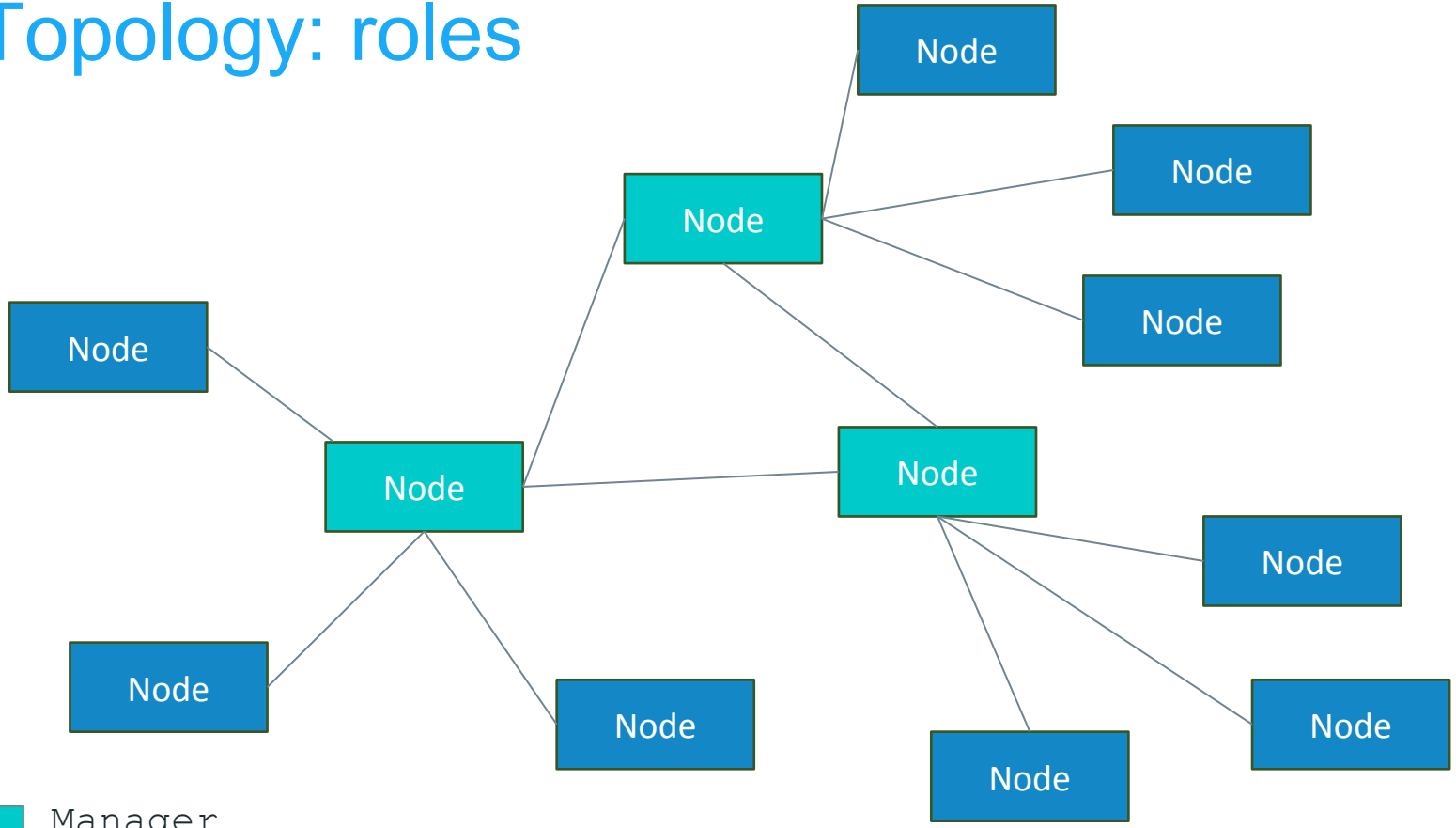
Deep Dive: Topology

Topology



docker

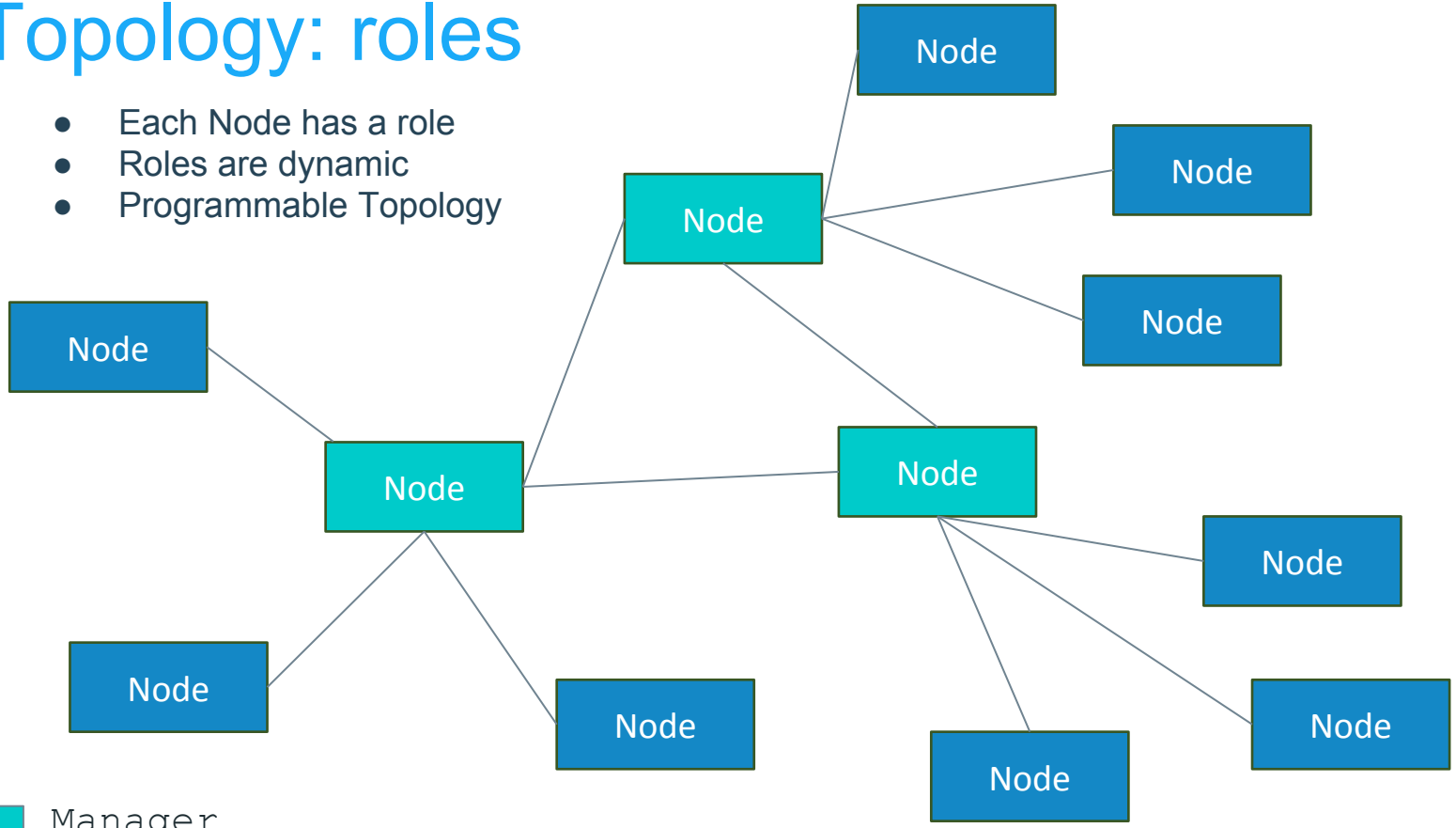
Topology: roles



docker

Topology: roles

- Each Node has a role
- Roles are dynamic
- Programmable Topology

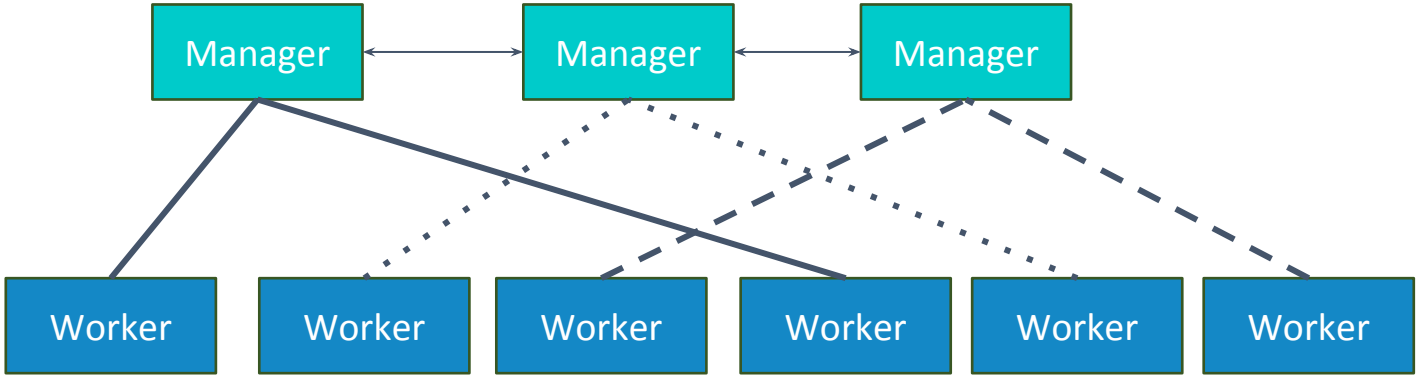


Manager

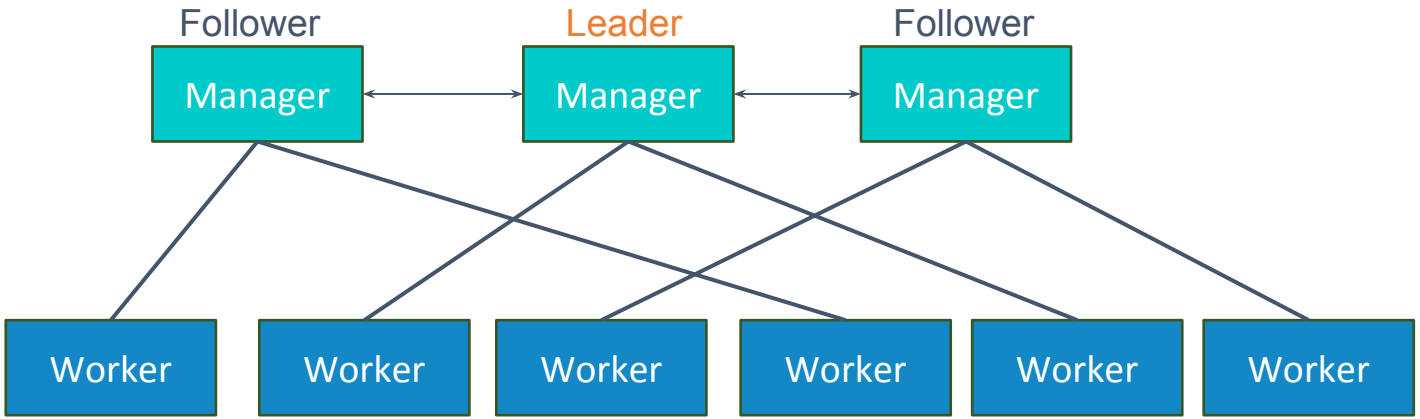
Worker

docker

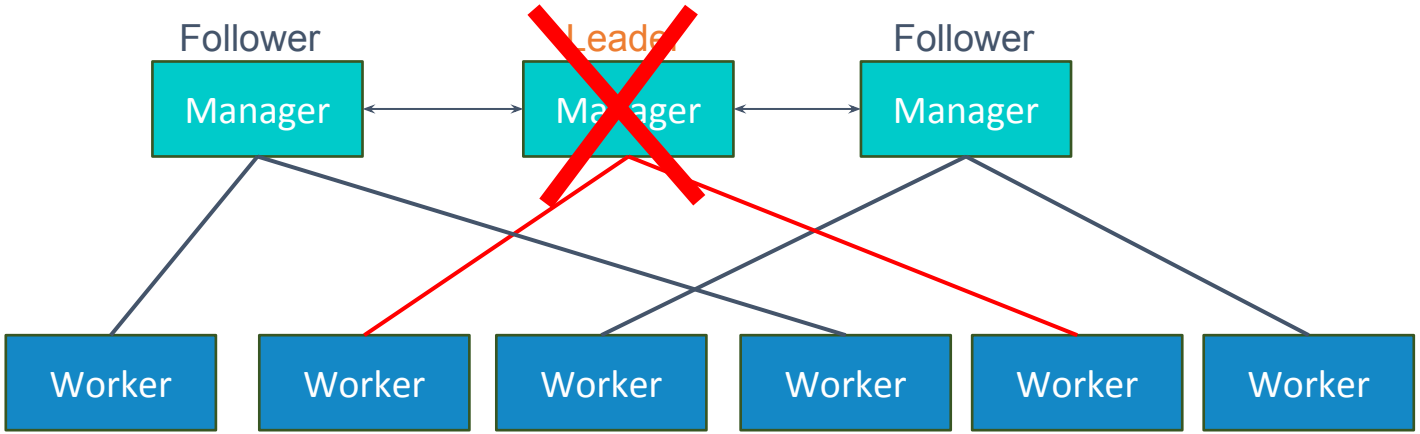
Topology: scaling model



Topology: High Availability

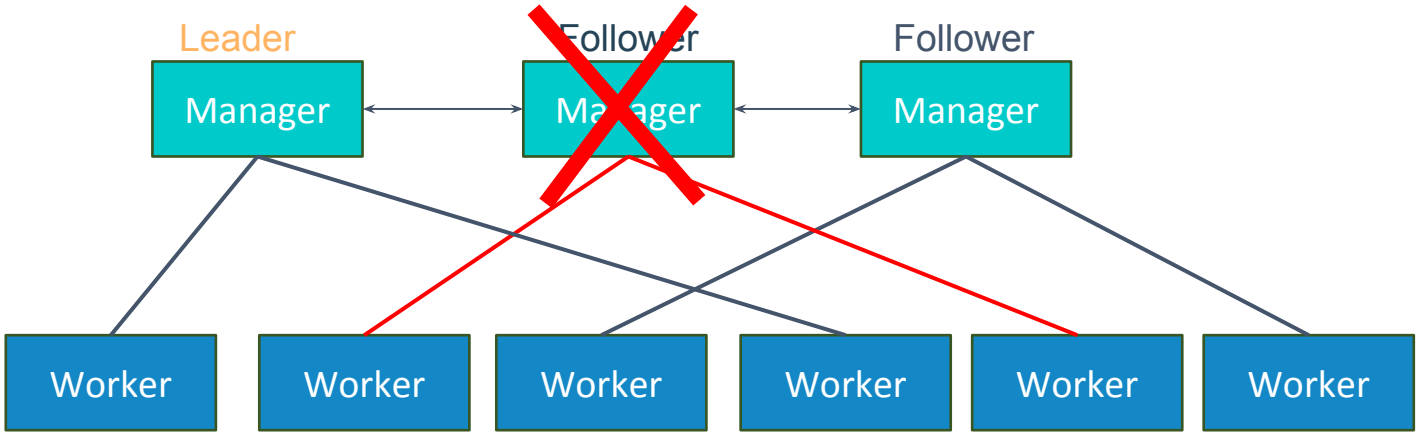


Topology: High Availability

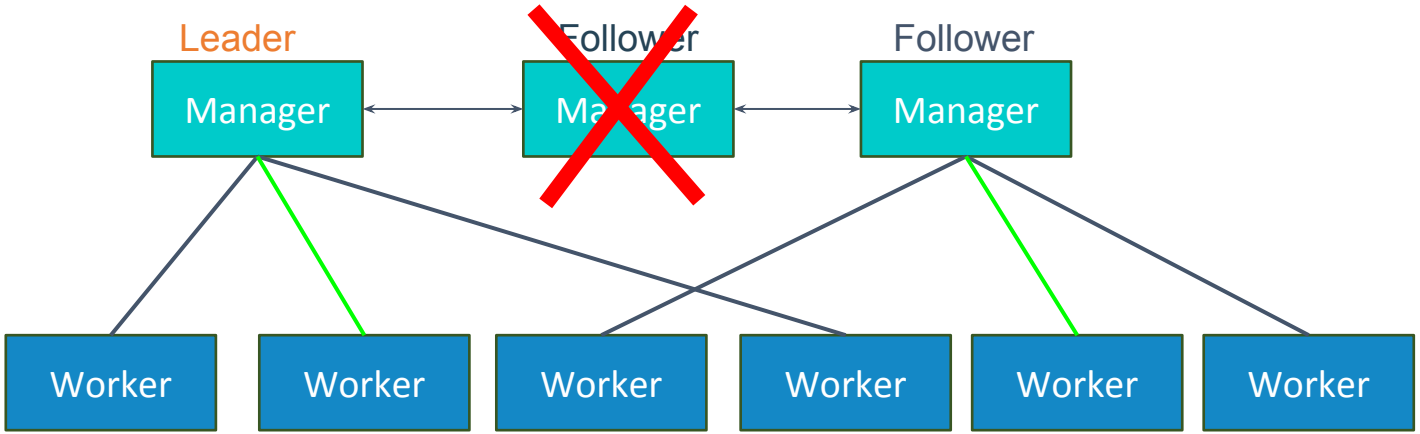


docker

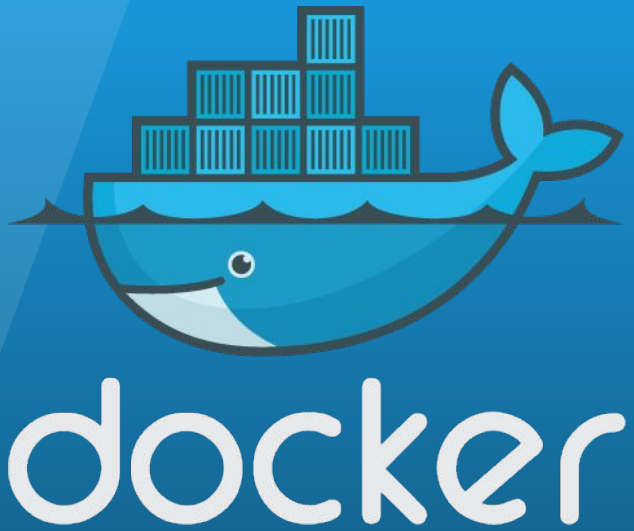
Topology: High Availability



Topology: High Availability



DEMO



Victor Vieux
vieux@docker.com / @vieux

Mike Goelzer
mgoelzer@docker.com / @mgoelzer