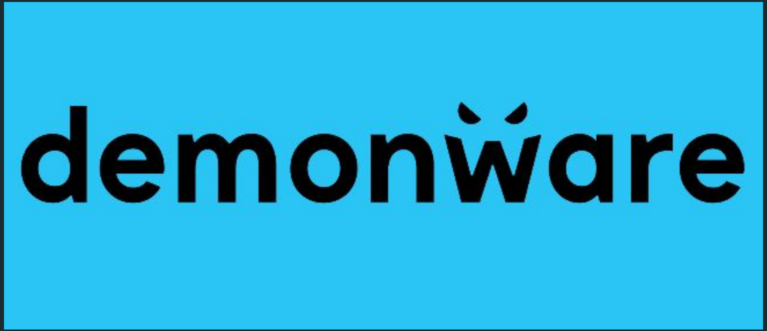
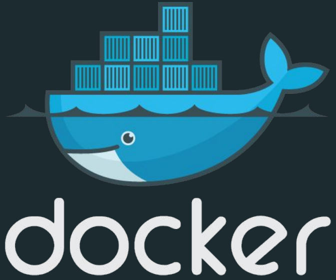
Docker and Jenkins Meetup

Sponsors





Agenda

- DockerCon 2016 Recap
    - What’s new in Docker 1.12
    - Orchestration Deep Dive
- Swarm Demo
- 5 minute break
- Jenkins Setup and Pipeline Demo

#DockerMeetup

#DockerDublin

#JenkinsMeetup

DockerCon 2016 Recap

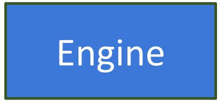
What’s new in Docker 1.12 ?

- Orchestration
  - Swarm Mode
  - Docker Services
  - Security
  - Routing mesh

What’s new from Docker Inc. ?

- Docker for X
  - Mac/Windows (public beta)
  - AWS
  - Azure
- Container Healthcheck
- Plugins improvements
- Docker Application Bundle

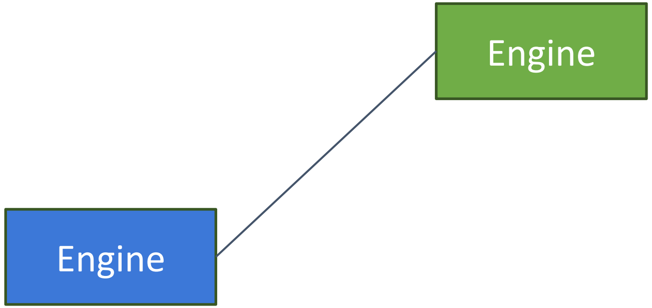
Swarm Mode



🟦

```
$ docker swarm init
```

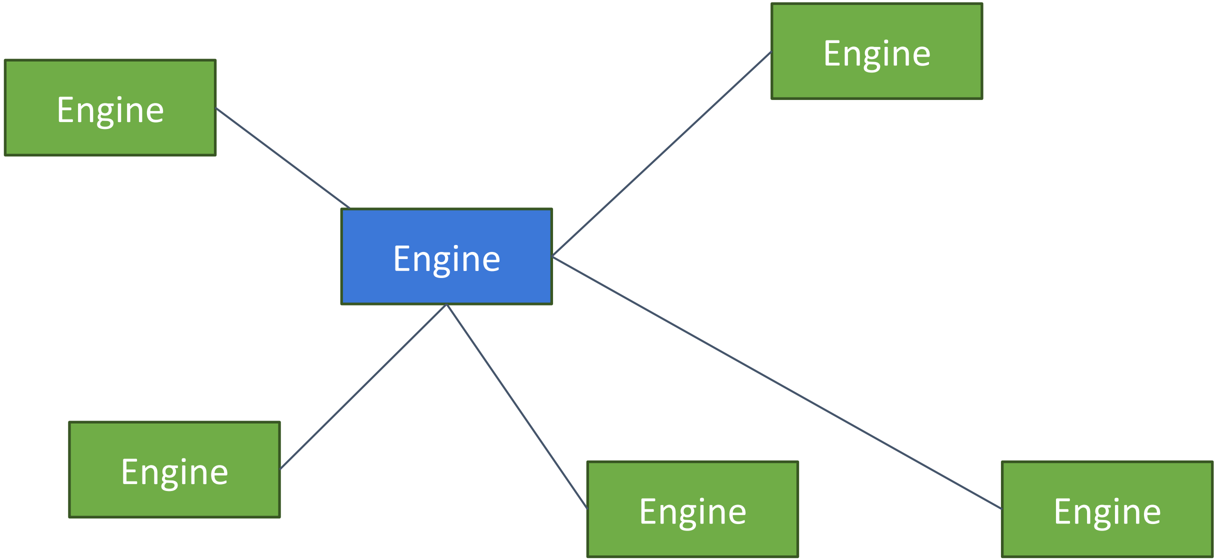
Swarm Mode



🟦 `$ docker swarm init`

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

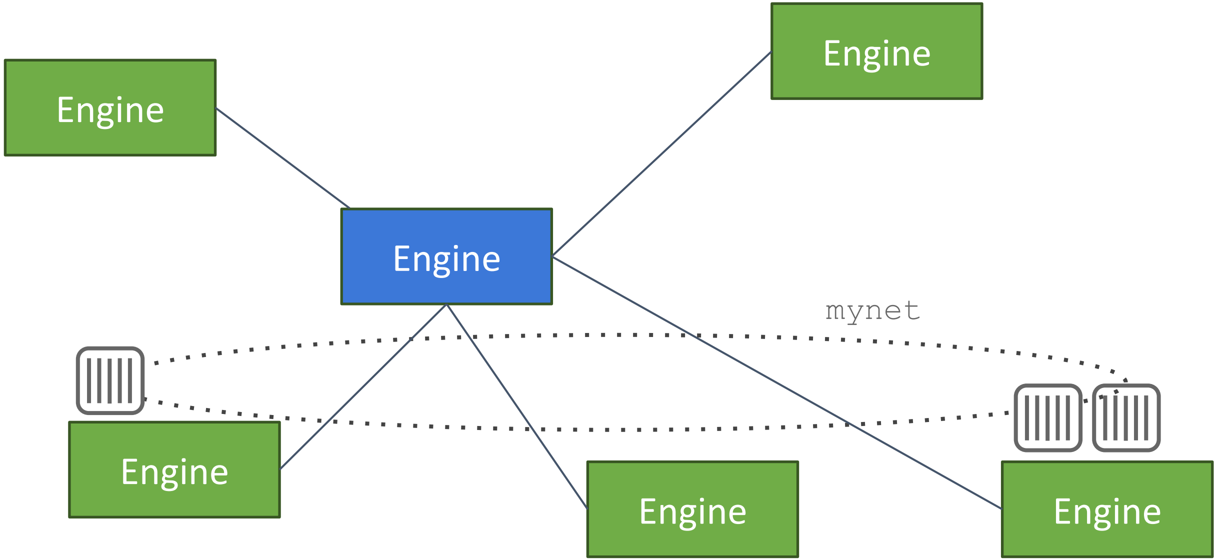
Swarm Mode



🟦 `$ docker swarm init`

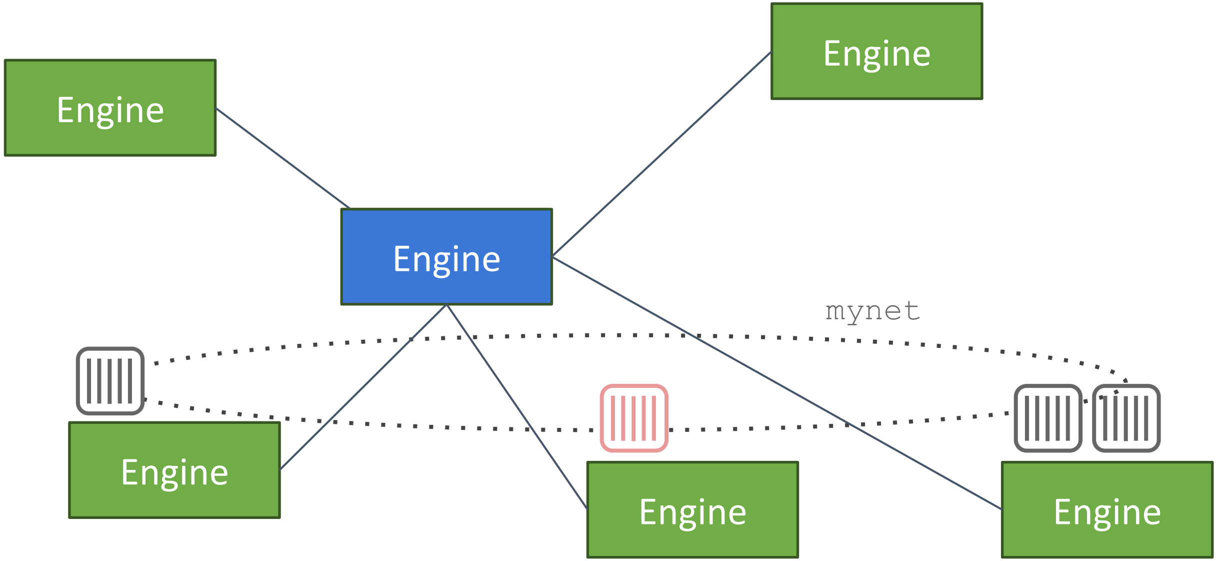
🟩 `$ docker swarm join <IP of manager>:2377`

Services



```
$ docker service create --replicas 3 --name frontend --network mynet
  --publish 80:80/tcp frontend_image:latest
```

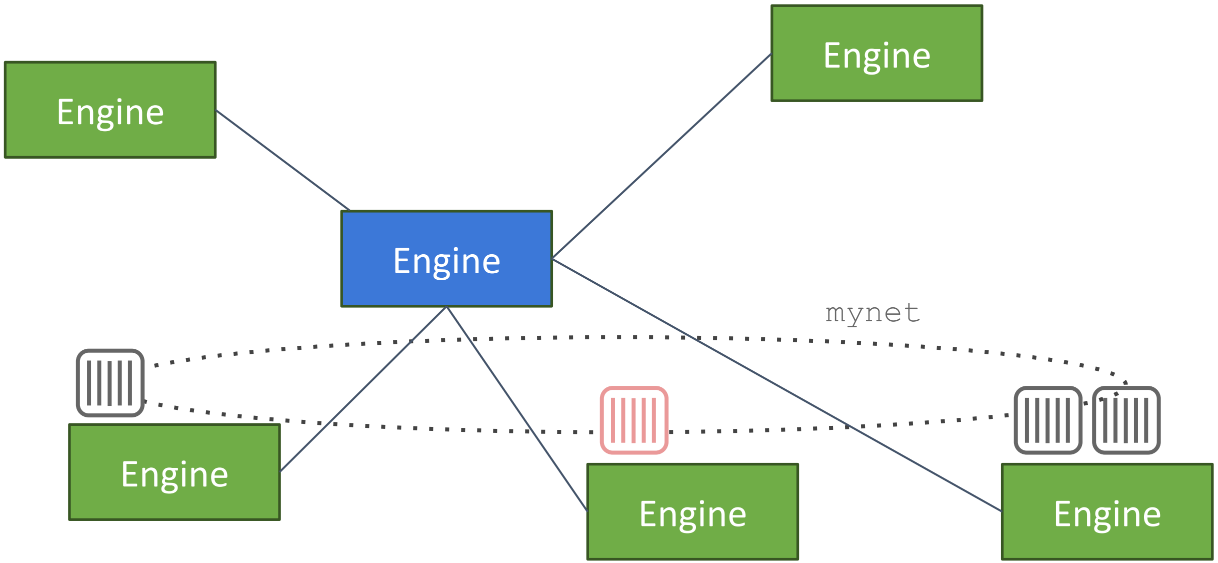
Services



```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

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

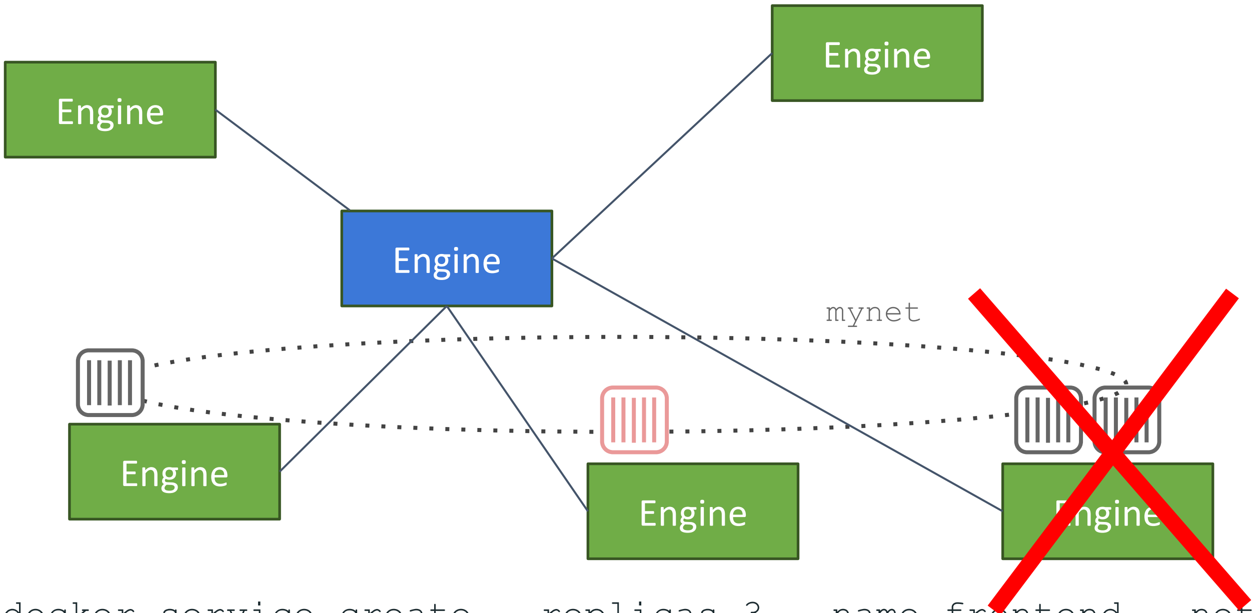
Node Failure



```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

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

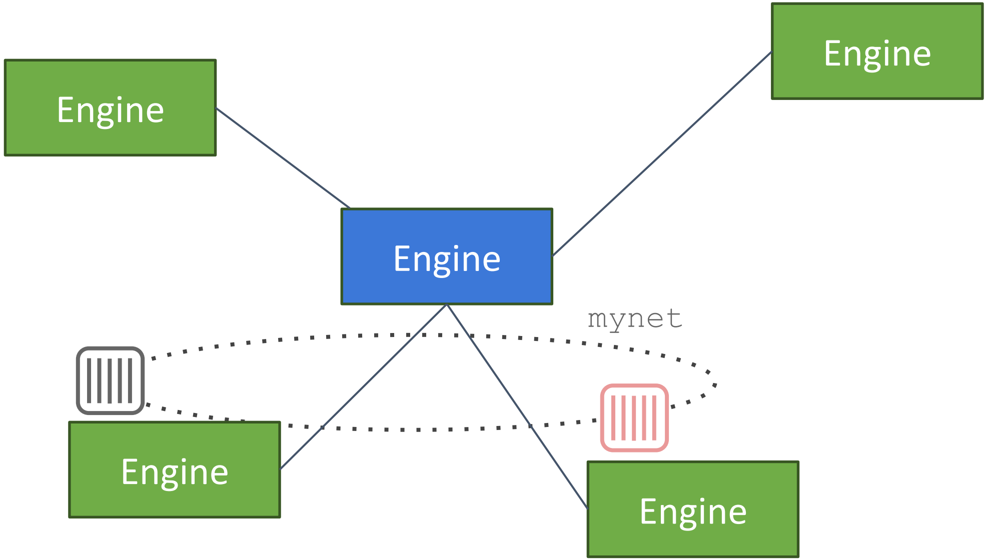
Node Failure



```
$ docker service create --replicas 3 --name frontend --network mynet
  --publish 80:80/tcp frontend_image:latest
```

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

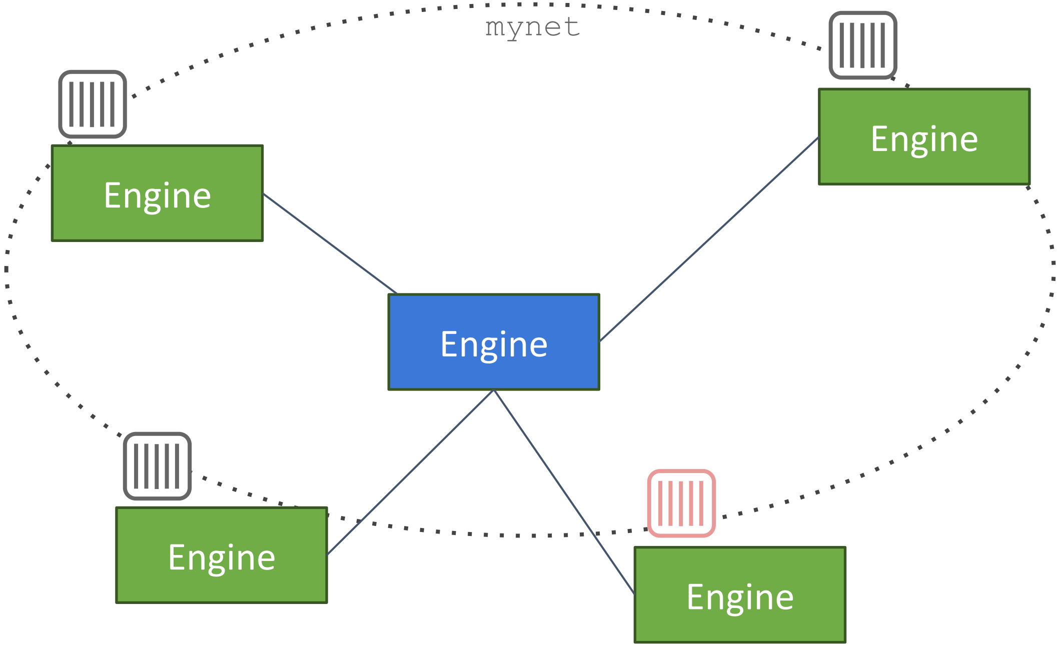
Desired State ≠ Actual State



```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp frontend_image:latest
```

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

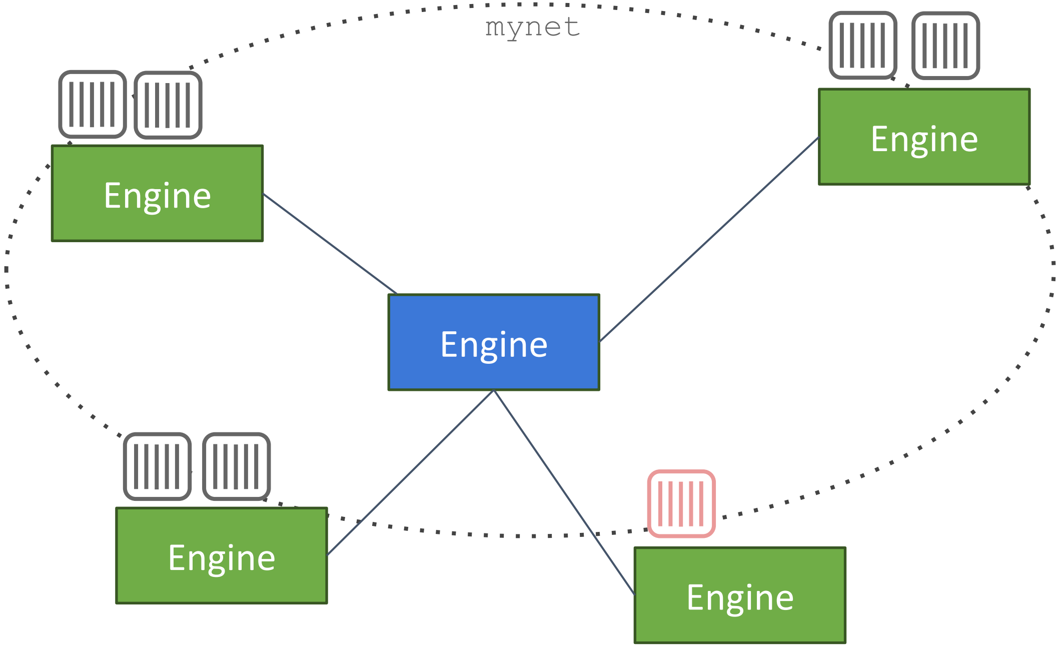
Converge Back to Desired State



```
$ docker service create --replicas 3 --name frontend --network mynet
  --publish 80:80/tcp frontend_image:latest
```

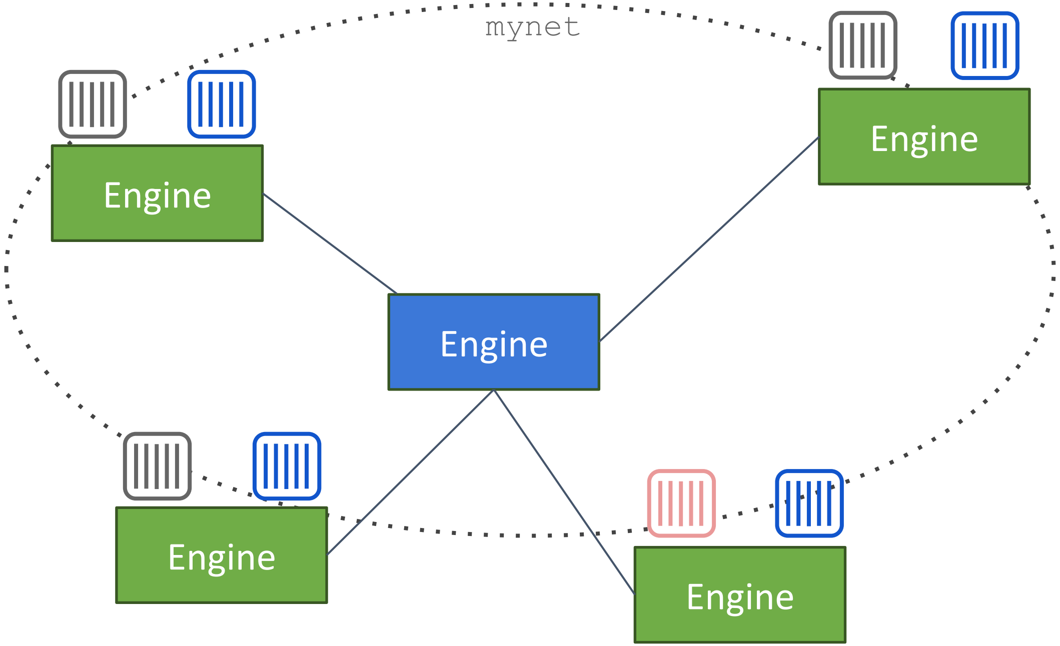
```
$ docker service create --name redis --network mynet redis:latest
```

Scaling



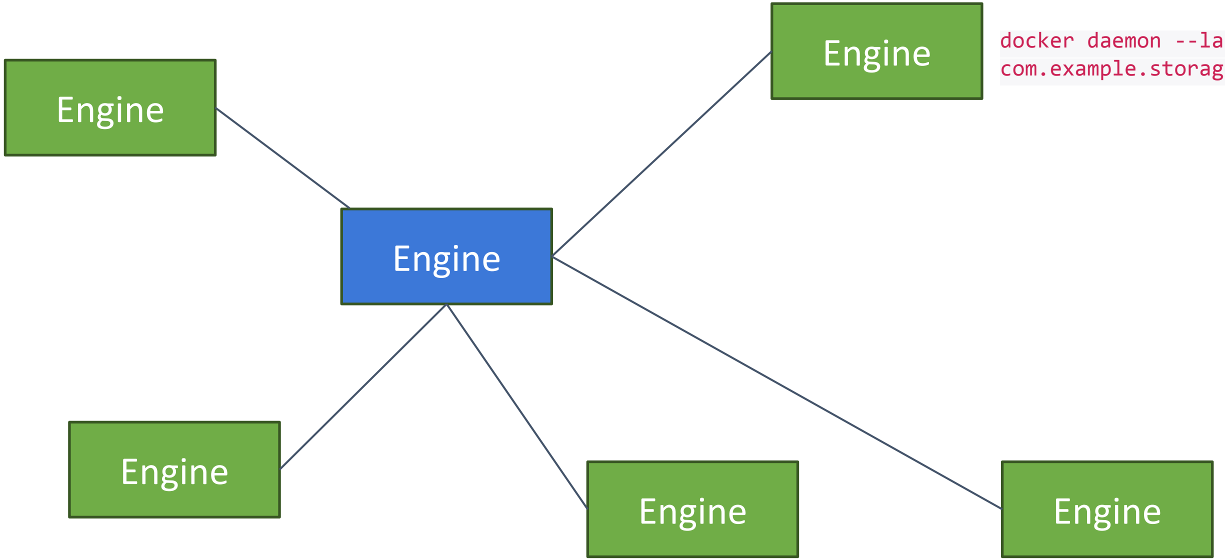
```
$ docker service scale frontend=6
```

Global Services



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

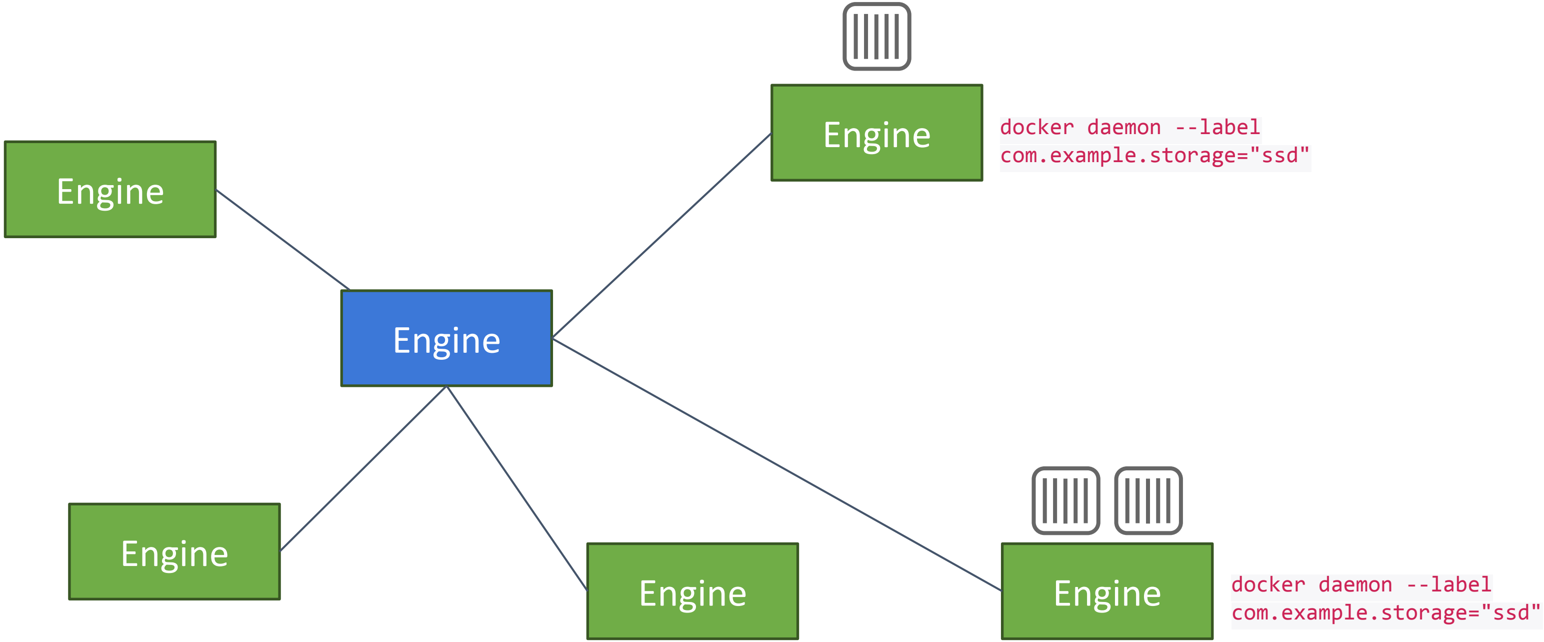
Constraints



```
docker daemon --label
com.example.storage="ssd"
```

```
docker daemon --label
com.example.storage="ssd"
```

Constraints

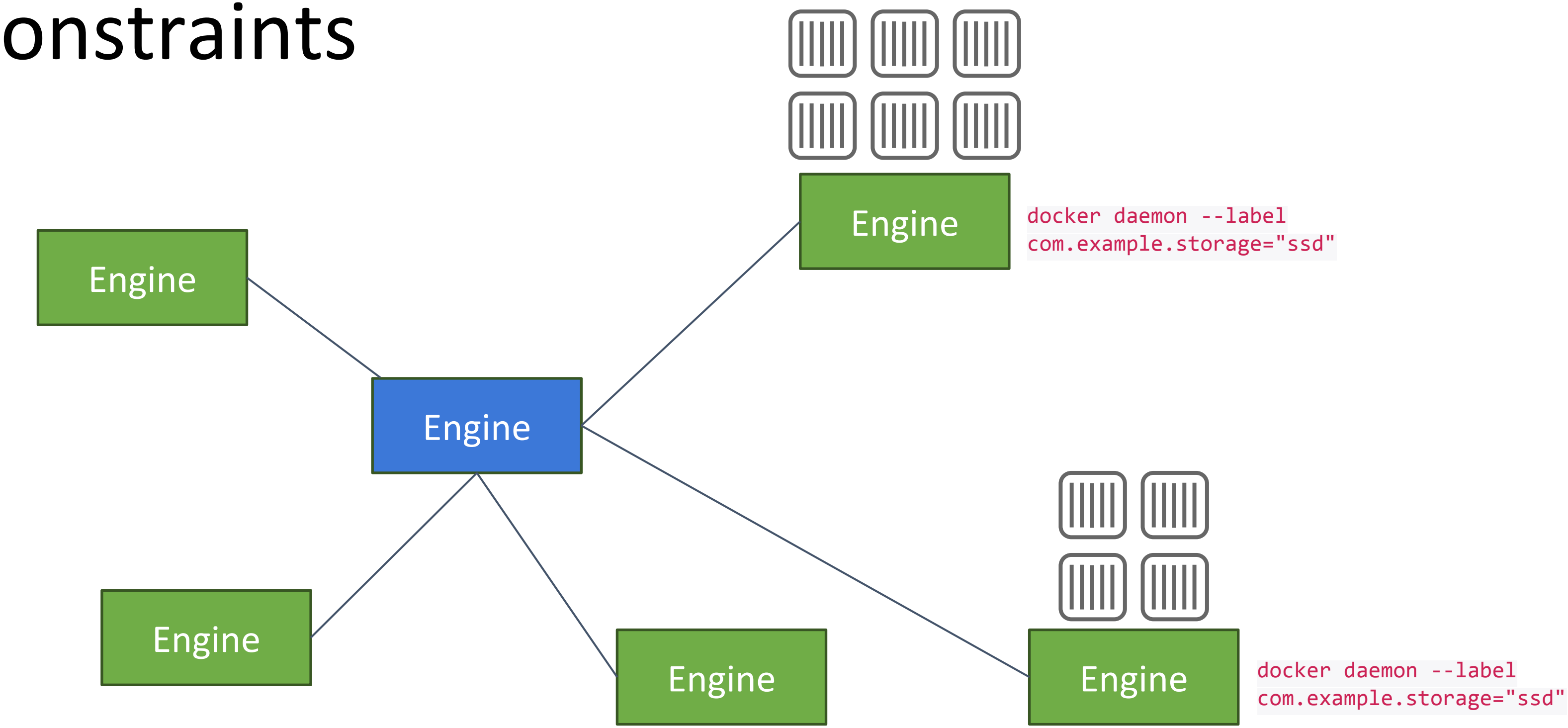


```
docker daemon --label
com.example.storage="ssd"
```

```
docker daemon --label
com.example.storage="ssd"
```

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

Constraints



```
docker daemon --label
com.example.storage="ssd"
```

```
docker daemon --label
com.example.storage="ssd"
```

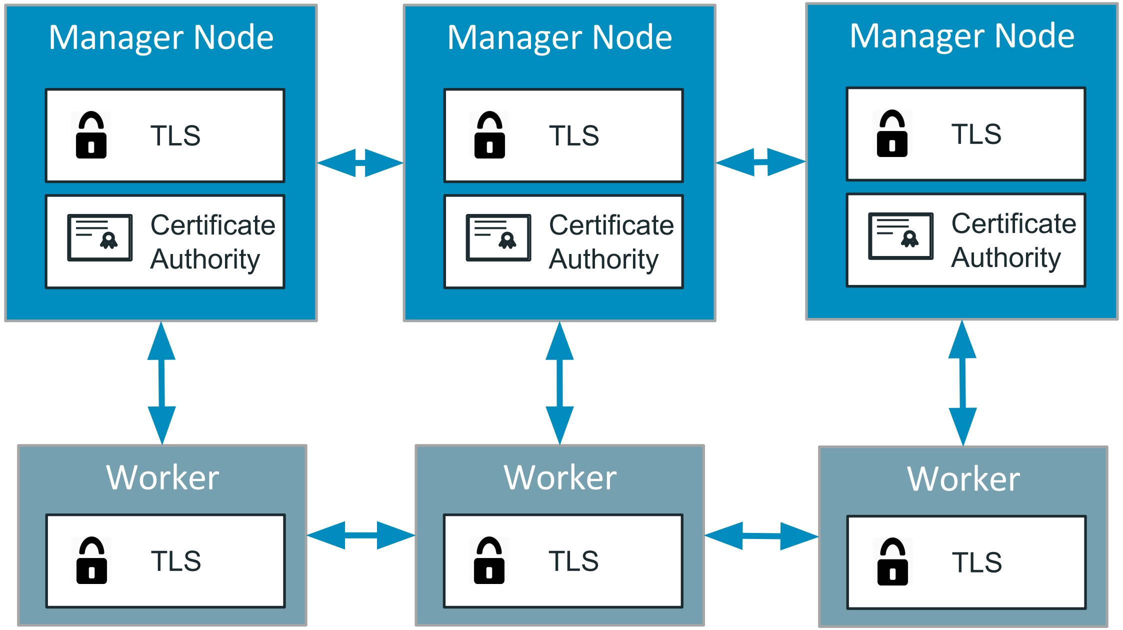
```
$ docker service create --replicas 3 --name frontend --network mynet
 --publish 80:80/tcp --constraint engine.labels.com.example.
storage==ssd frontend_image:latest
$ docker service scale frontend=10
```

How to Try Docker 1.12

- Mac/Windows: https://www.docker.com/products/docker
- Linux: https://github.com/docker/docker/releases
- AWS/Azure Editions Beta: https://www.docker.com/products/docker
- Bleeding edge (docker:master binaries from CI): https://master.dockerproject.org/

- Keynote demo: https://www.youtube.com/watch?v=Q1jSDyZ4Org

Secure by default with end to end encryption

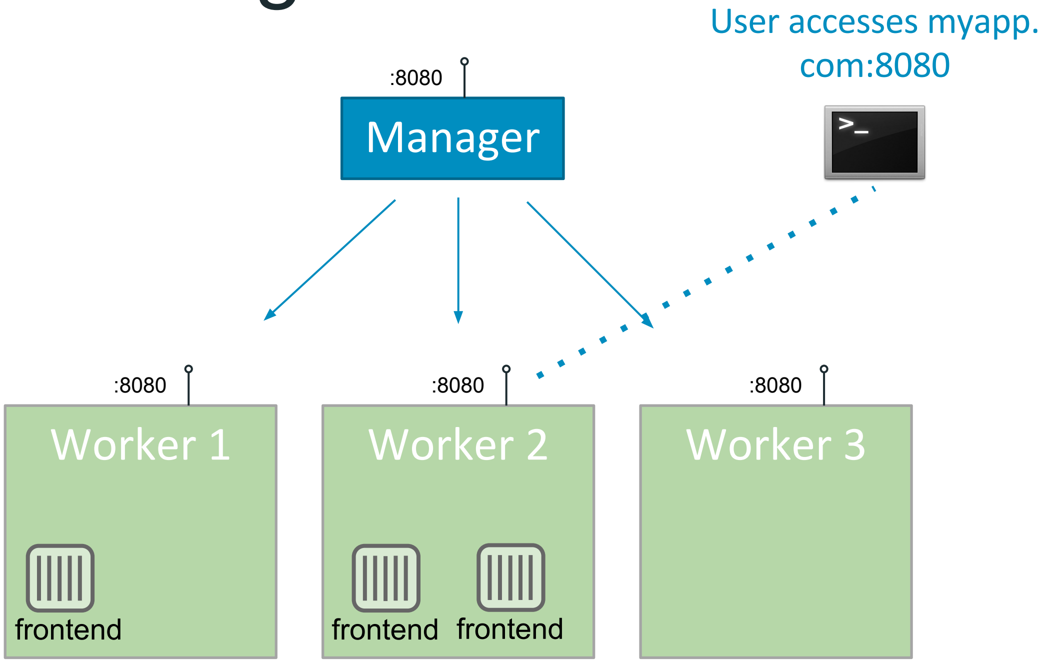


- Cryptographic node identity
- Automatic encryption and mutual auth (TLS)
- Automatic cert rotation
- External CA integration

Swarm mode orchestration is optional

- You don’t have to use it
- 1.12 is fully backwards compatible
- Will not break existing deployments and scripts

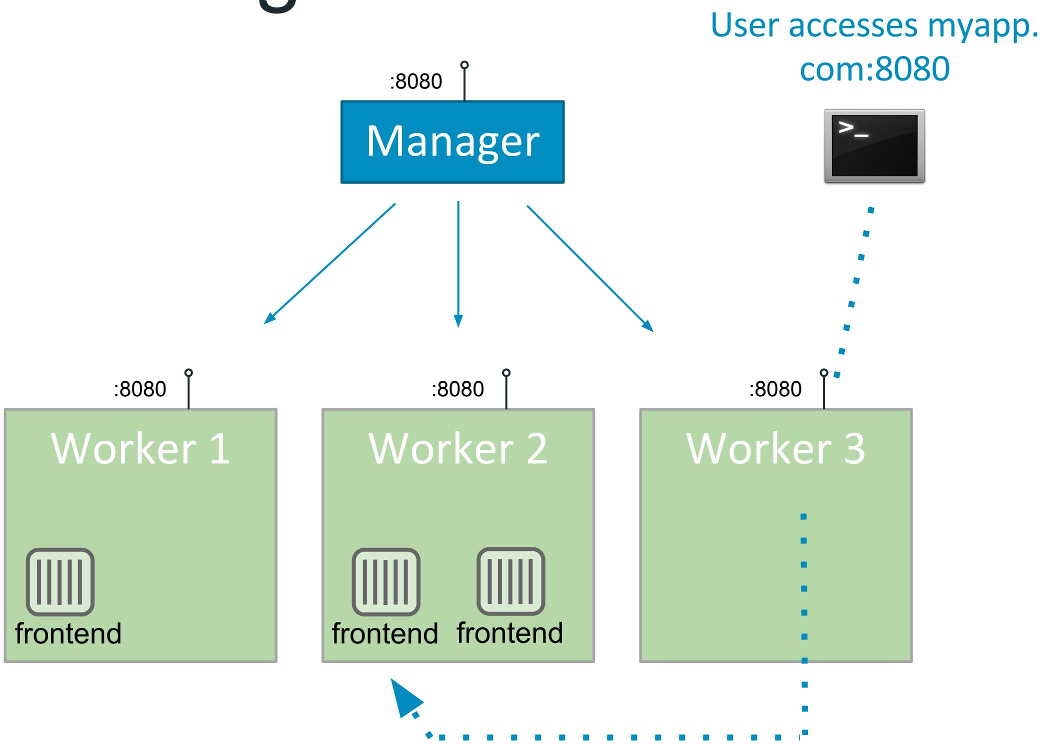
Routing Mesh



- 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
 --publish 8080:80/tcp frontend_image:latest
```

Routing Mesh: Published Ports



User accesses myapp.com:8080

:8080

Manager

:8080

Worker 1

frontend

:8080

Worker 2

frontend

frontend

:8080

Worker 3

- 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
 --publish 8080:80/tcp frontend_image:latest
```

Container Health Check in `Dockerfile`

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

Checks every 5 minutes that web server can return index page within 3 seconds.

Three consecutive failures puts container in an unhealthy state.

Daemonless containers

Upgrade Docker, keep containers alive

```
docker daemon --live-restore
```

(https://github.com/docker/docker/pull/23213)

Plugin Permissions Model

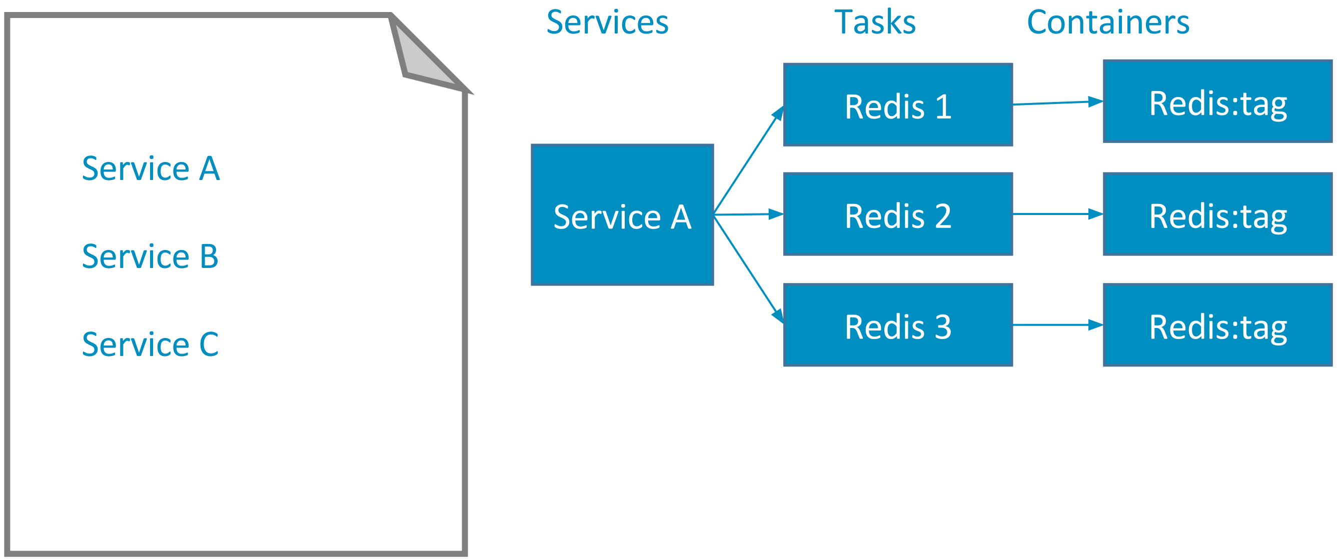
```
$ docker plugin install tiborvass/no-remove
Plugin "tiborvass/no-remove:latest" requested
the following privileges:
 - Networking: host
 - Mounting host path: /data
Do you grant the above permissions? [y/N]
```

Distributed Application Bundle (.dab) declares a stack

Experimental! The bundle is a multi-services distributable image format

https://github.com/docker/docker/blob/master/experimental/docker-stacks-and-bundles.md

Distributed Application Bundle (.dab) declares a stack



Distributed Application Bundle

DAB - Producing a bundle

```
$ docker-compose bundle
WARNING: Unsupported key 'network_mode' in services.nsqd - ignoring
WARNING: Unsupported key 'links' in services.nsqd - ignoring
WARNING: Unsupported key 'volumes' in services.nsqd - ignoring
[...]
Wrote bundle to vossibility-stack.dab
```

DAB - Deploying a bundle

```
$ docker deploy vossibility-stack
Loading bundle from vossibility-stack.dab
Creating service vossibility-stack_elasticsearch
Creating service vossibility-stack_kibana
Creating service vossibility-stack_logstash
Creating service vossibility-stack_vossibility-collector
```

How to Try Docker 1.12

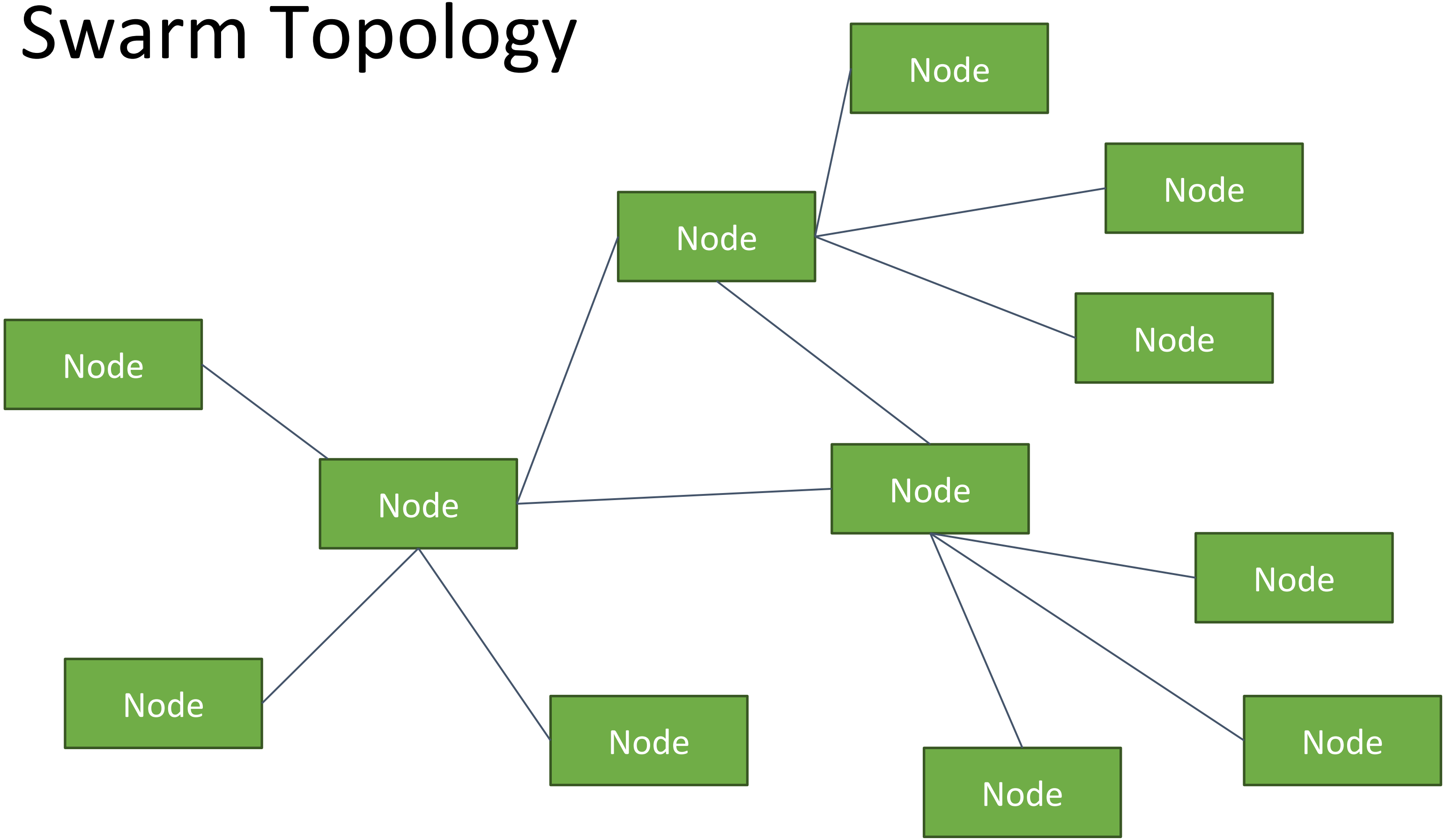
- Mac/Windows: https://www.docker.com/products/docker
- Linux: https://github.com/docker/docker/releases
- AWS/Azure Editions Beta: https://www.docker.com/products/docker
- Bleeding edge (docker:master binaries from CI): https://master.dockerproject.org/

- Good quick overview: https://www.youtube.com/watch?v=Q1jSDyZ4Org

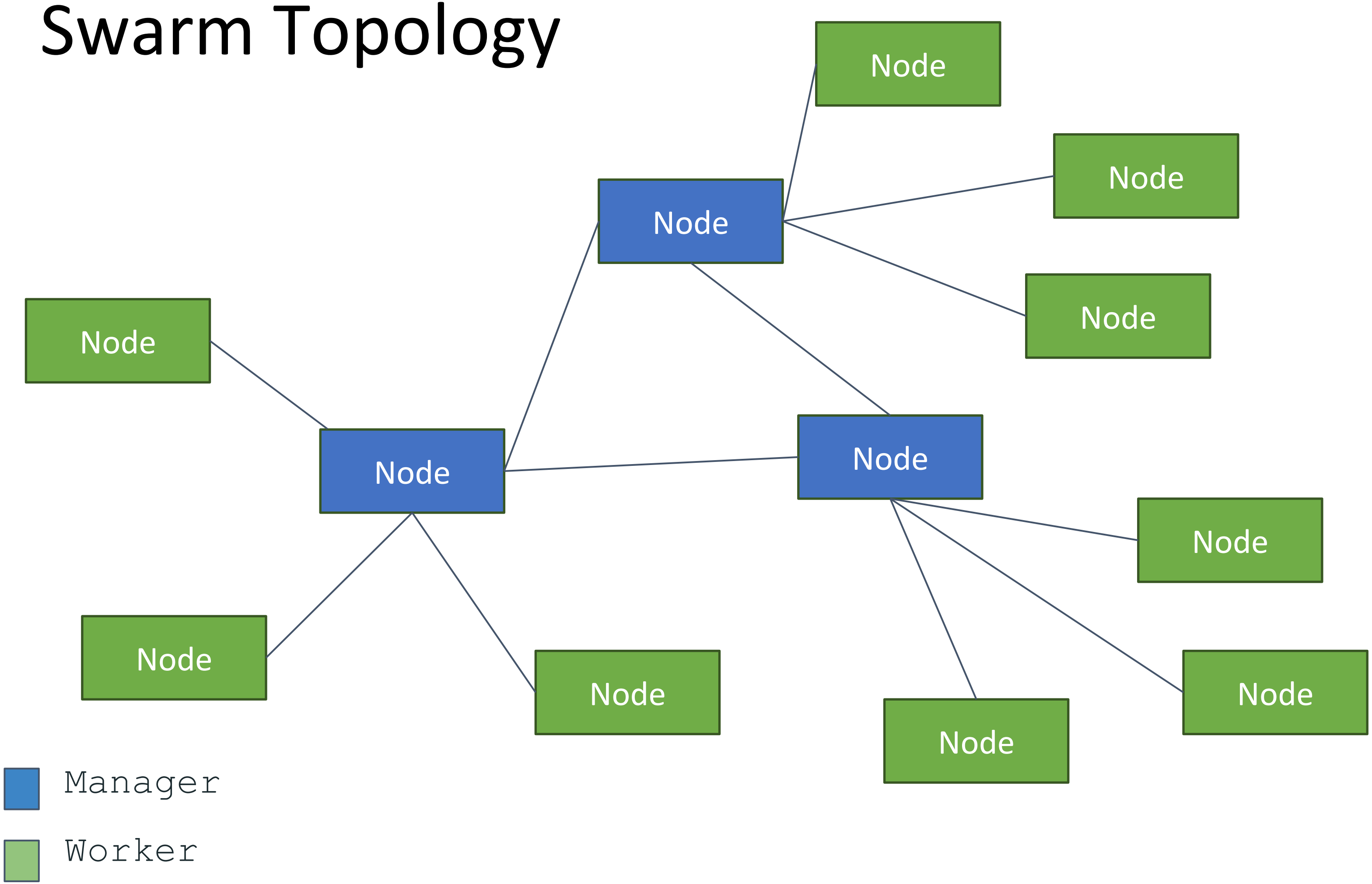
Orchestration Deep Dive

(To be covered in Demo)

Swarm Topology

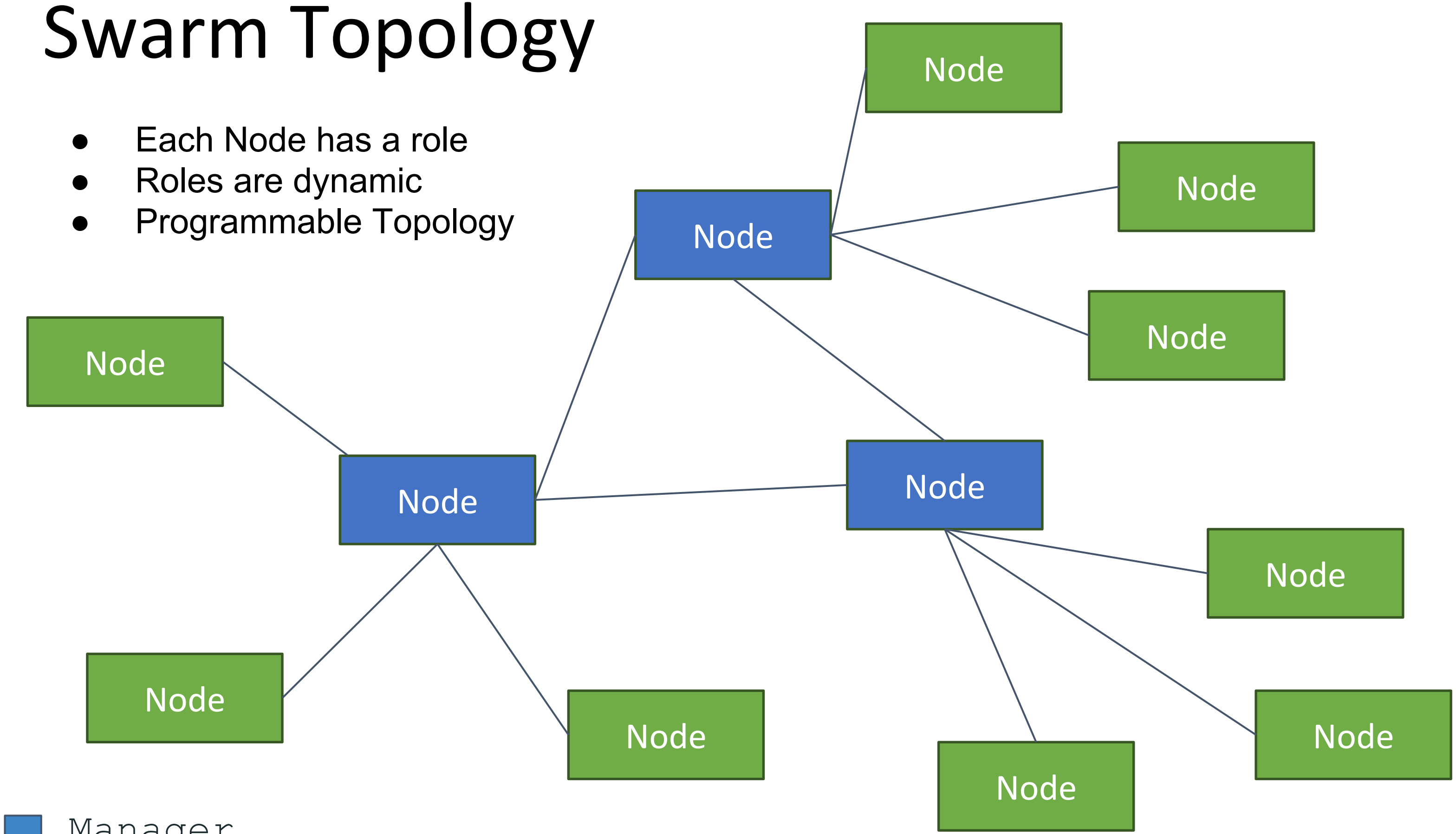


Swarm Topology



Swarm Topology

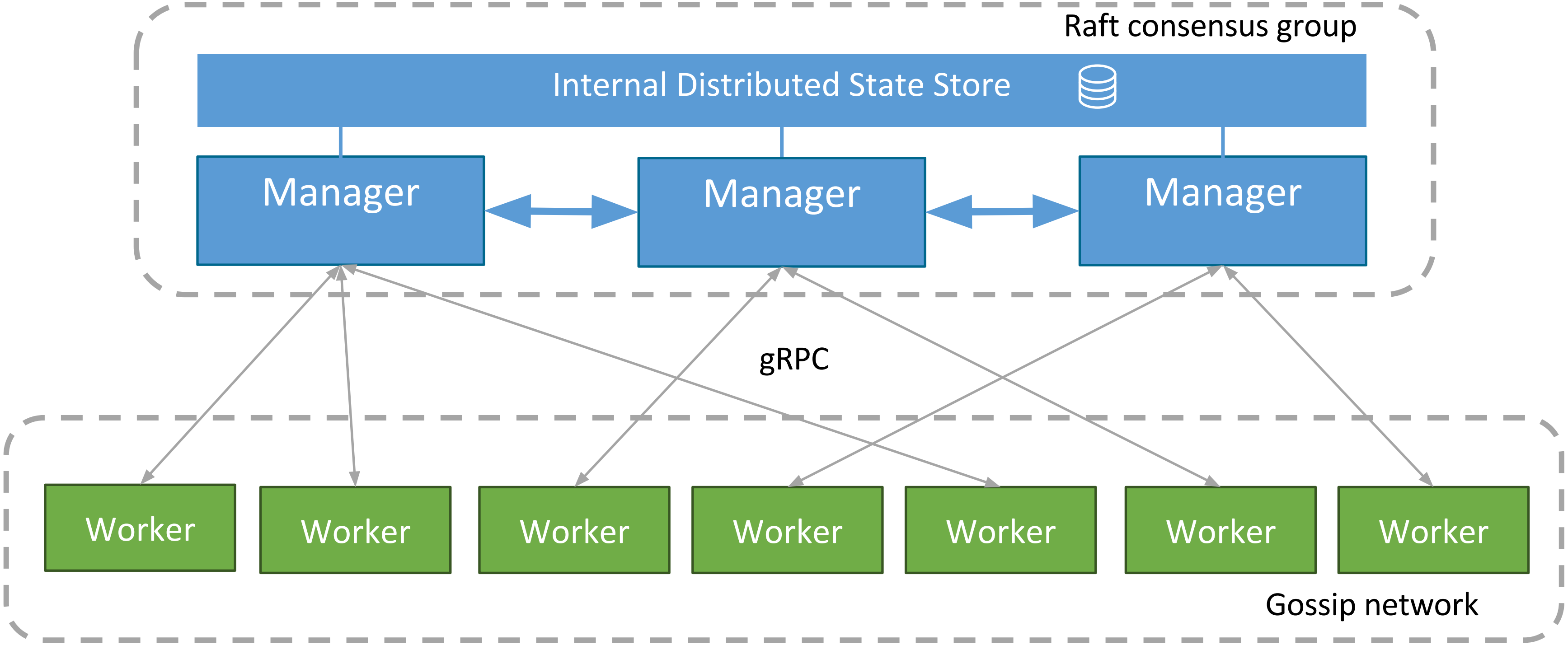
- Each Node has a role
- Roles are dynamic
- Programmable Topology



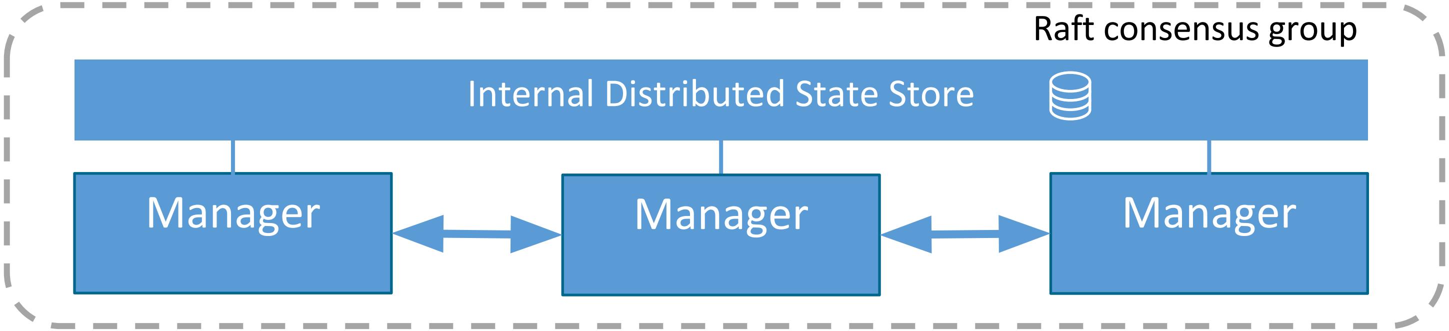
🟦 Manager

🟩 Worker

Docker Swarm Communication Internals

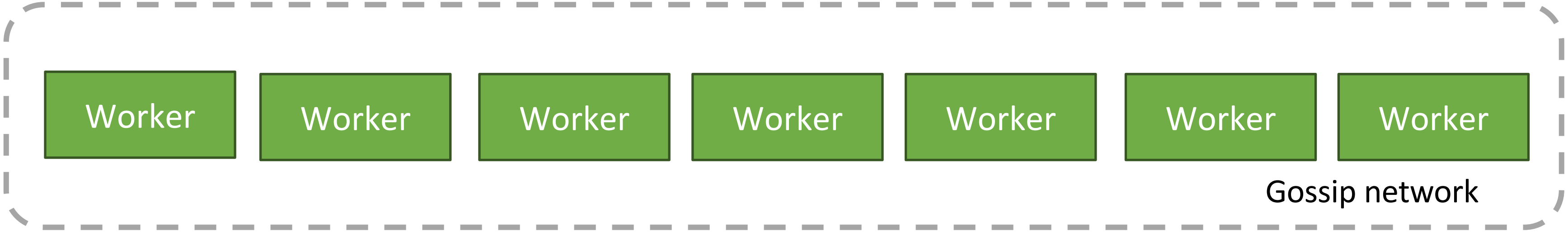


Quorum Layer



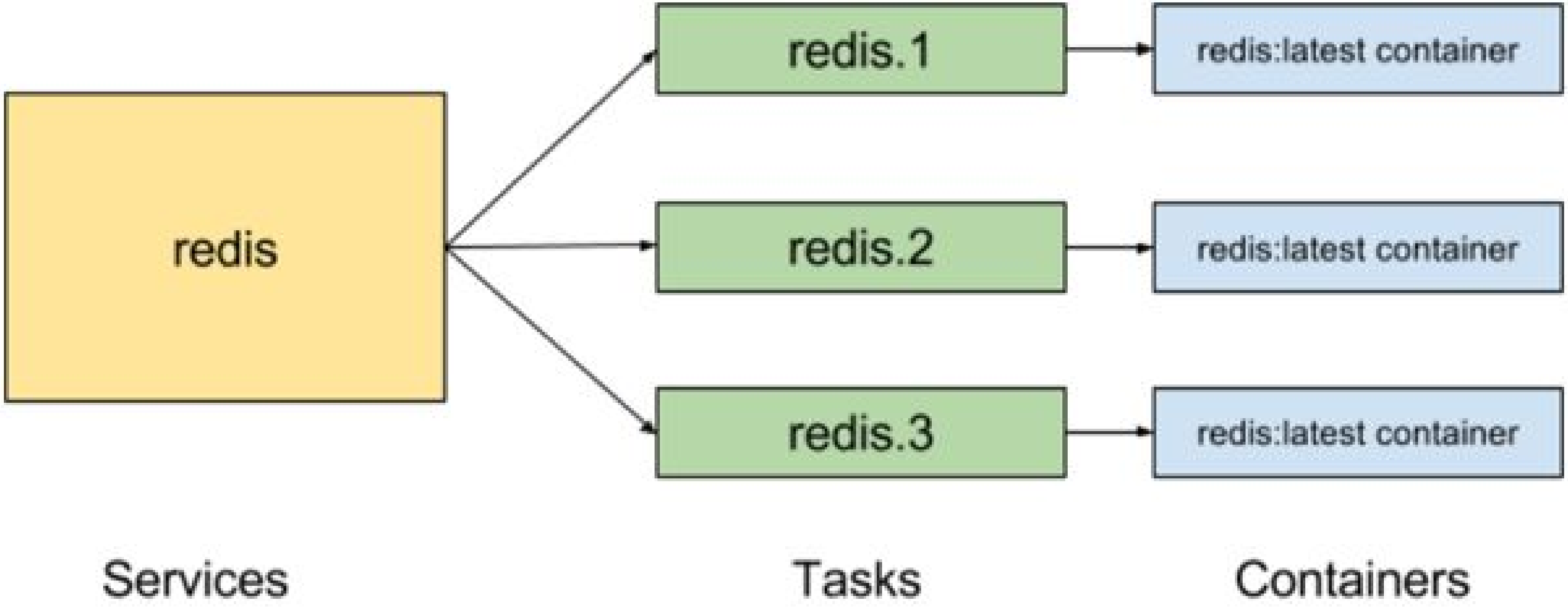
- Strongly consistent: Holds desired state
- Simple to operate
- Blazing fast (in-memory reads, domain specific indexing, ...)
- Secure

Worker-to-Worker Gossip

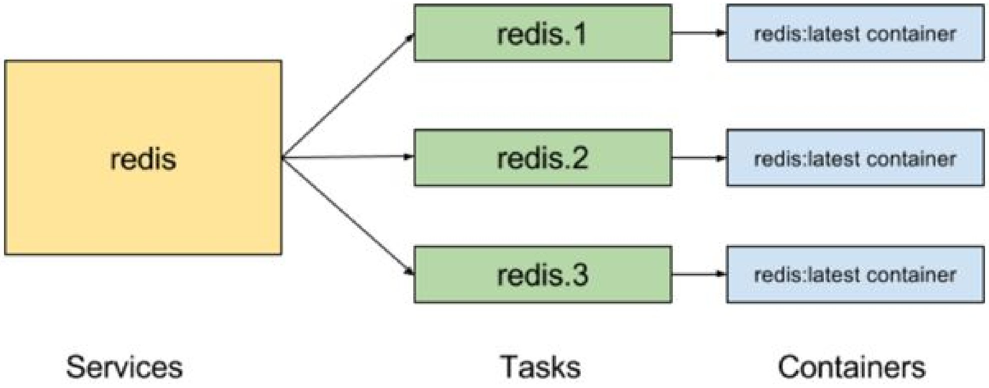


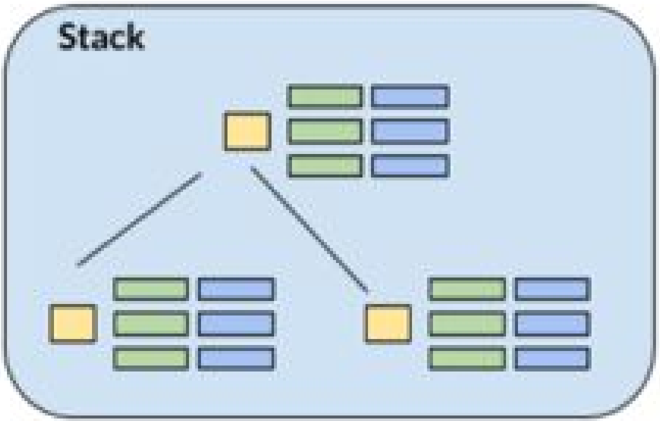
- Eventually consistent: Routing mesh, load balancing rules, ...
- High volume, p2p network between workers
- Secure: Symmetric encryption with key rotation in Raft

Services


redis
redis.1
redis.2
redis.3
redis:latest container
redis:latest container
redis:latest container
Services
Tasks
Containers

Services are grouped into stacks





#HackHarassment

demonware

docker

DockerCon Blog

http://2016.dockercon.com/blog

Docker for Mac and Windows

https://docs.docker.com/docker-for-mac/
https://docs.docker.com/docker-for-windows/

Docker for AWS and Azure

blog.docker.com/2016/06/docker-datacenter-aws-azure-cloud/

# Docker Swarm 2000

#DockerSwarm2000

Questions ?

Thomas Shaw / @tomwillfixit / @demonware

Gianluca Arbezzano / @gianarb