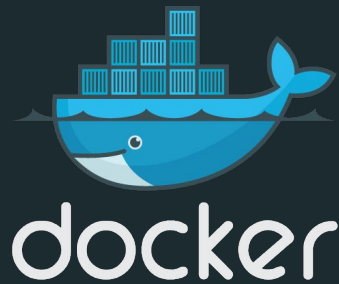


Docker Built-in Orchestration

Docker Madrid Meetup
July 26, 2016

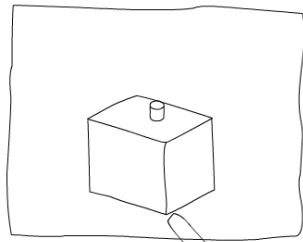
Pablo Chico de Guzmán



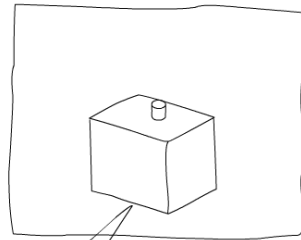
Build, Ship and Run

Any app, anywhere

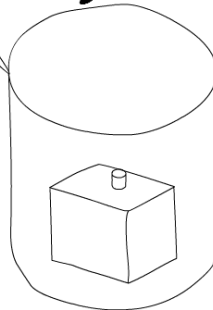
Development



Production



Registry



docker push

docker pull

What's Coming in Docker 1.12

- Docker for AWS/Azure
- Container Healthcheck
- Plugins improvements
- Docker Application Bundle
- **Orchestration**

Swarm Mode

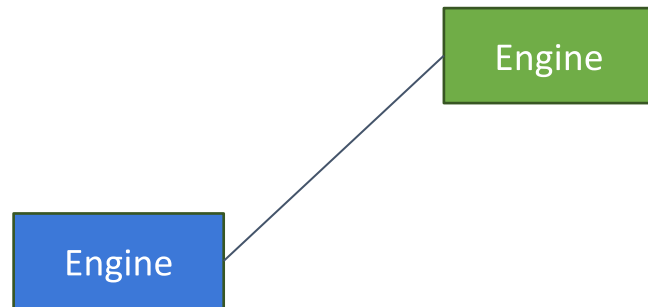



Engine



```
$ docker swarm init
```

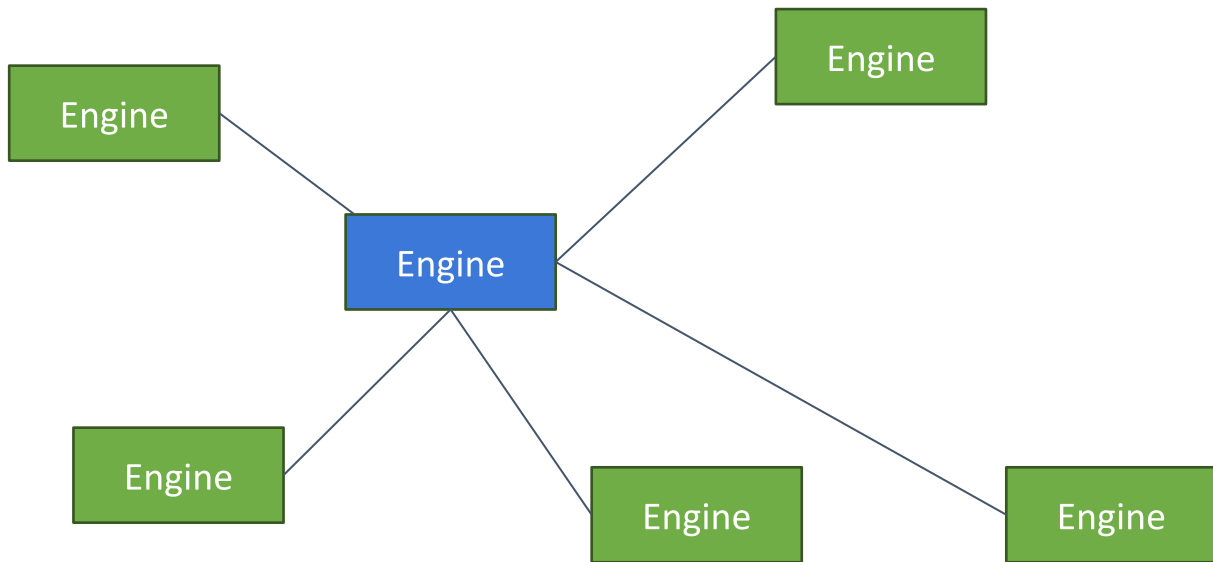
Swarm Mode




 `$ docker swarm init`

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

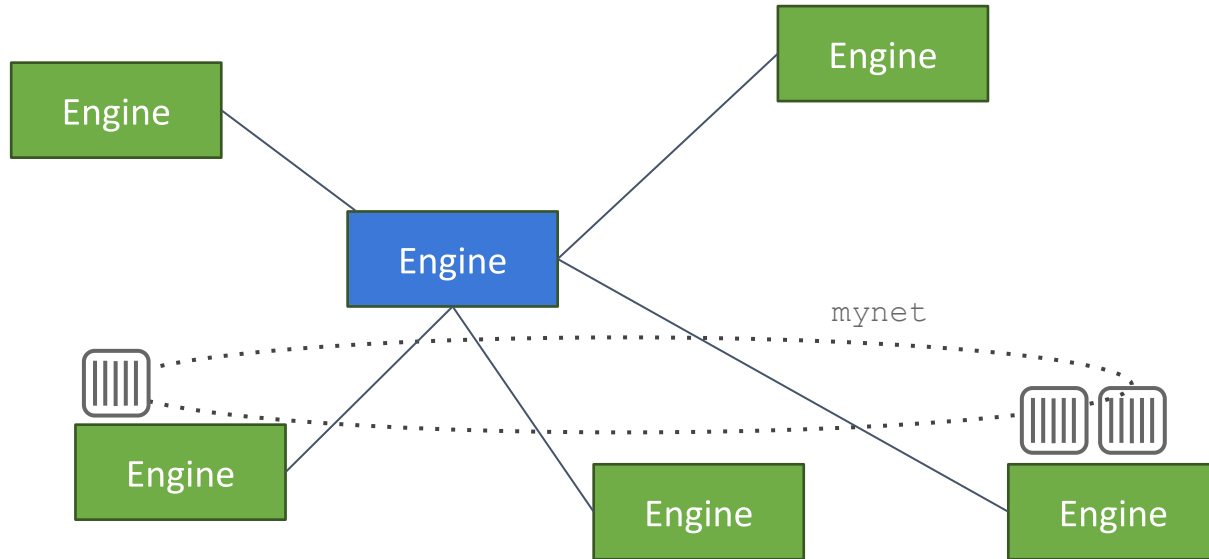
Swarm Mode



 `$ docker swarm init`

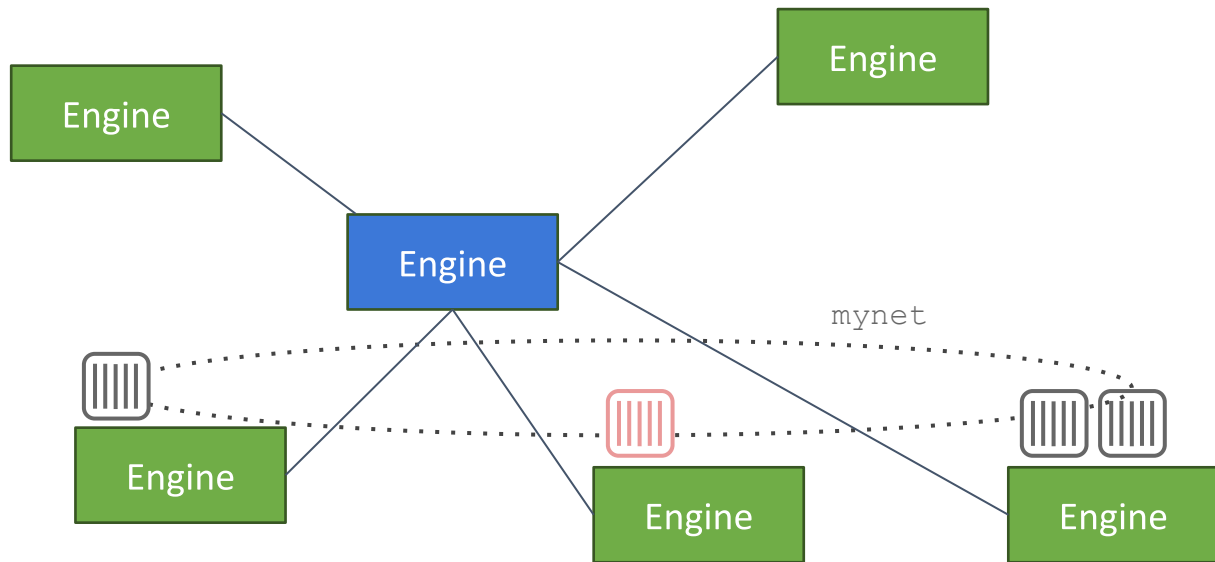
 `$ docker swarm join <IP of manager>:2377`


Services




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

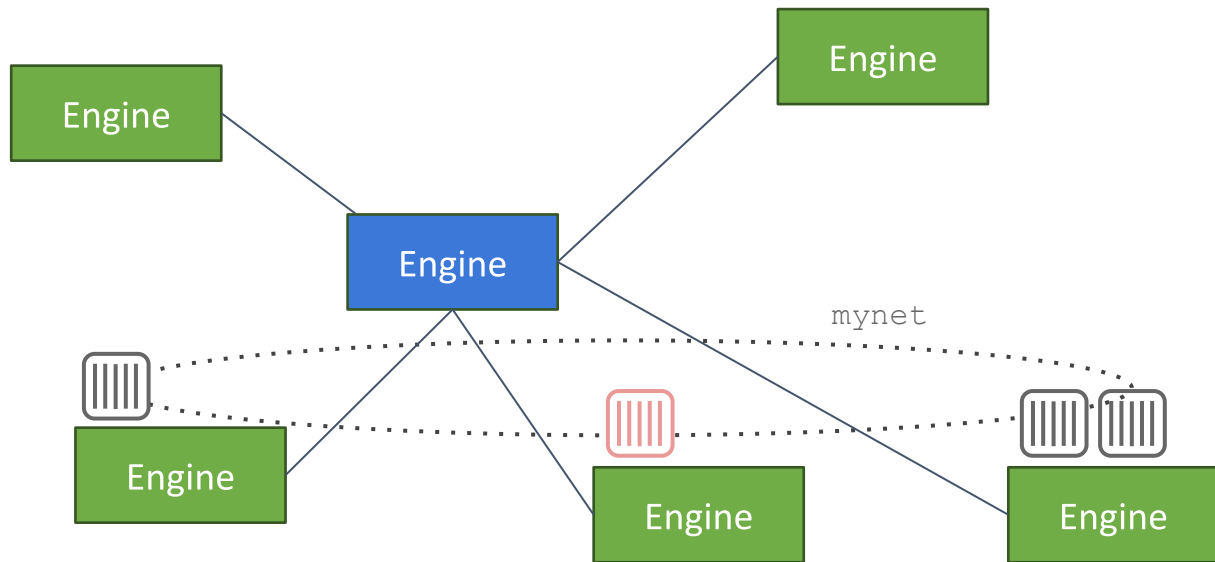
Services





 `$ 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`

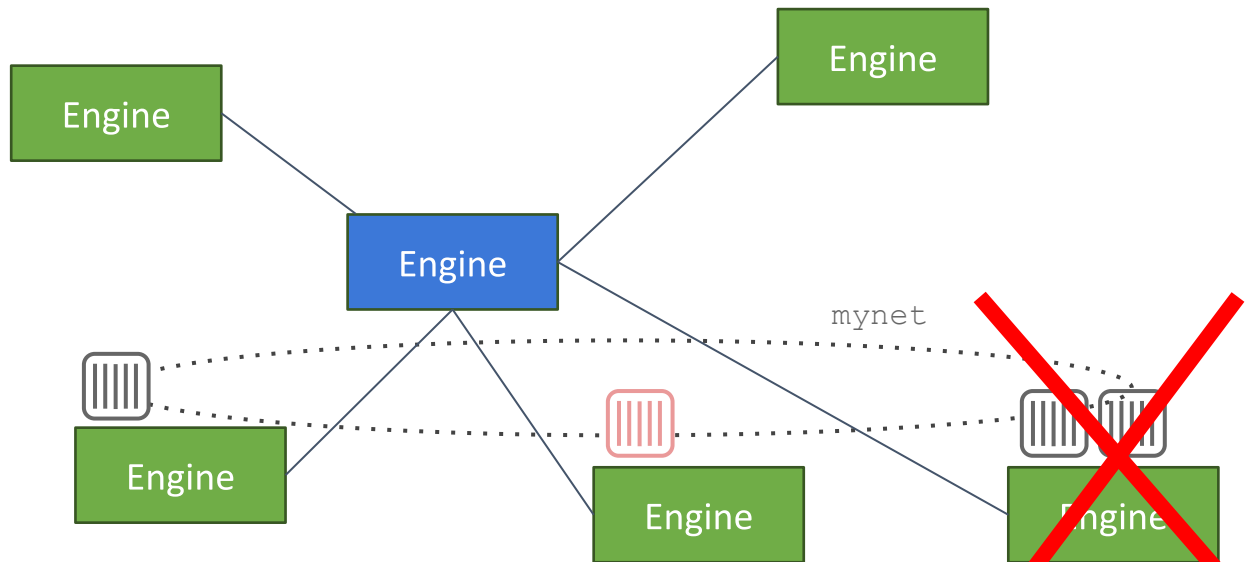
Node Failure





 `$ 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`

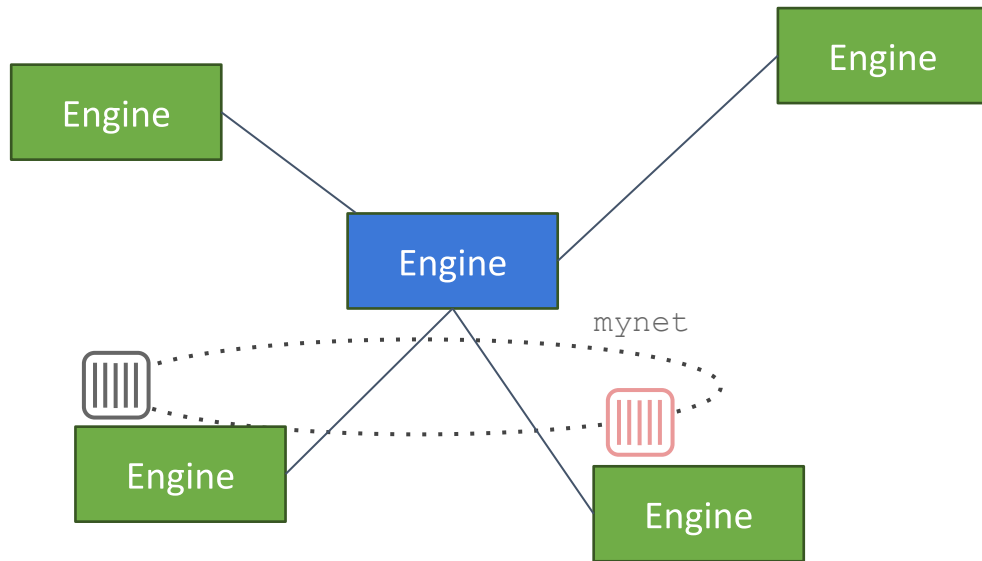
Node Failure





 `$ 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`

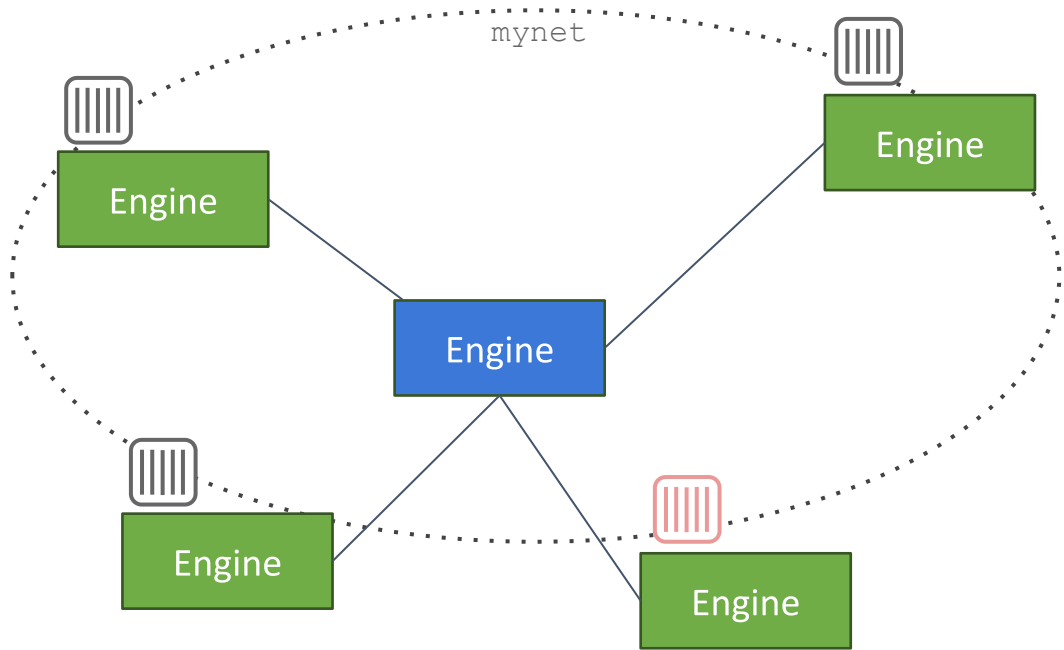
Desired State \neq Actual State



 \$ 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

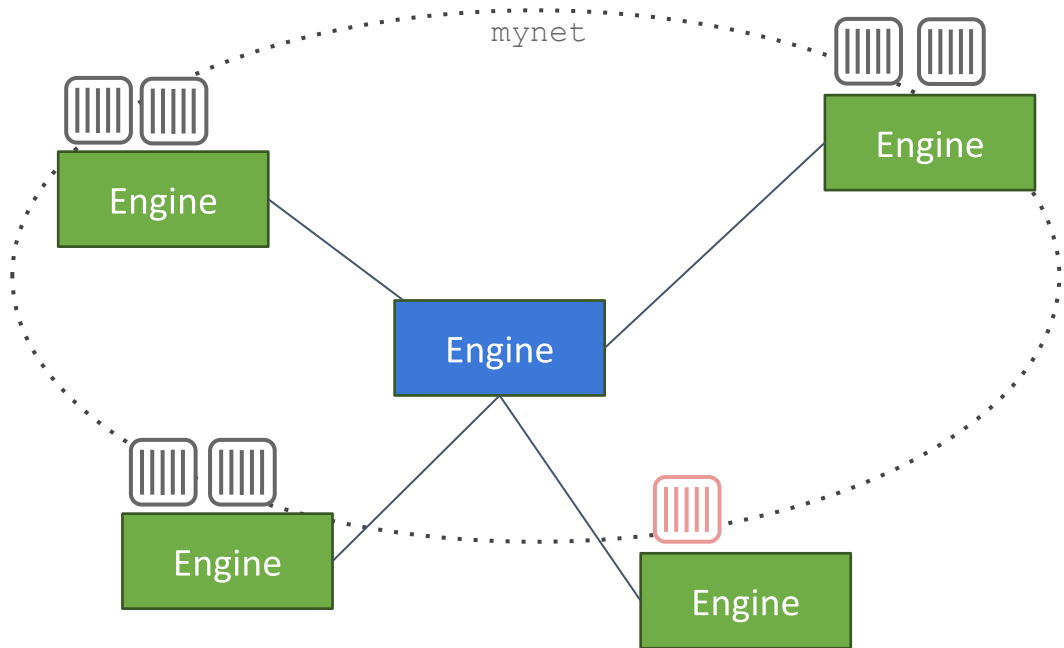


```
$ docker service create --replicas 3 --name frontend --network mynet  
--publish 80:80/tcp frontend_image:latest
```



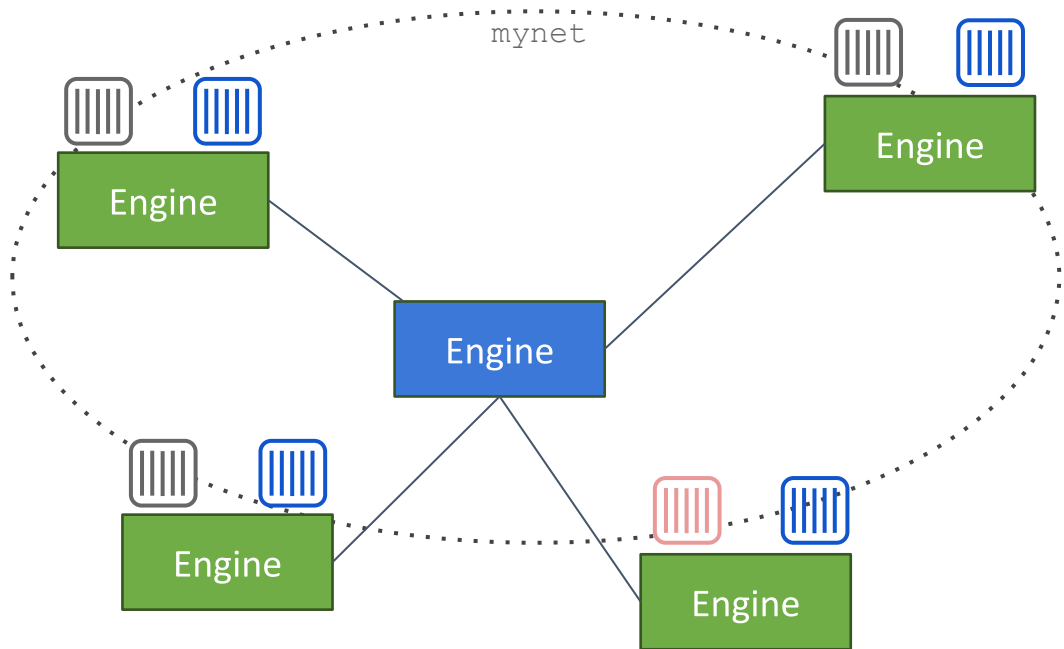
```
$ docker service create --name redis --network mynet redis:latest
```


Scaling



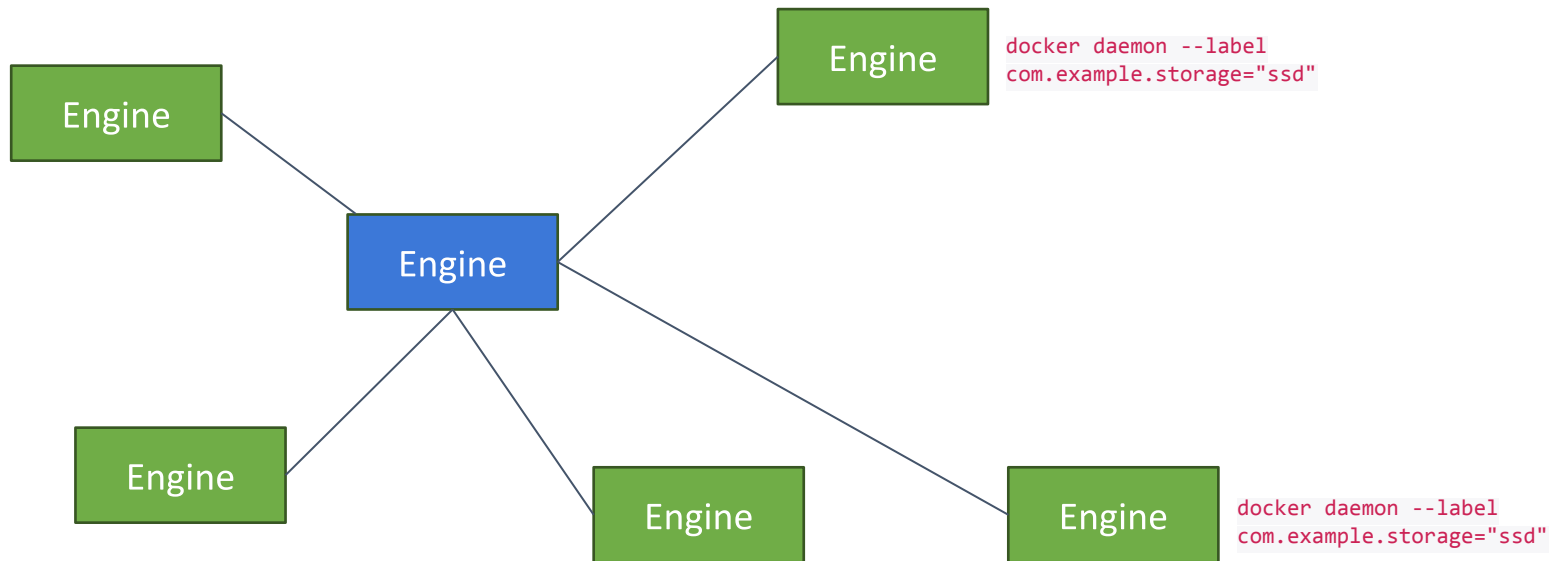
```
$ docker service scale frontend=6
```

Global Services

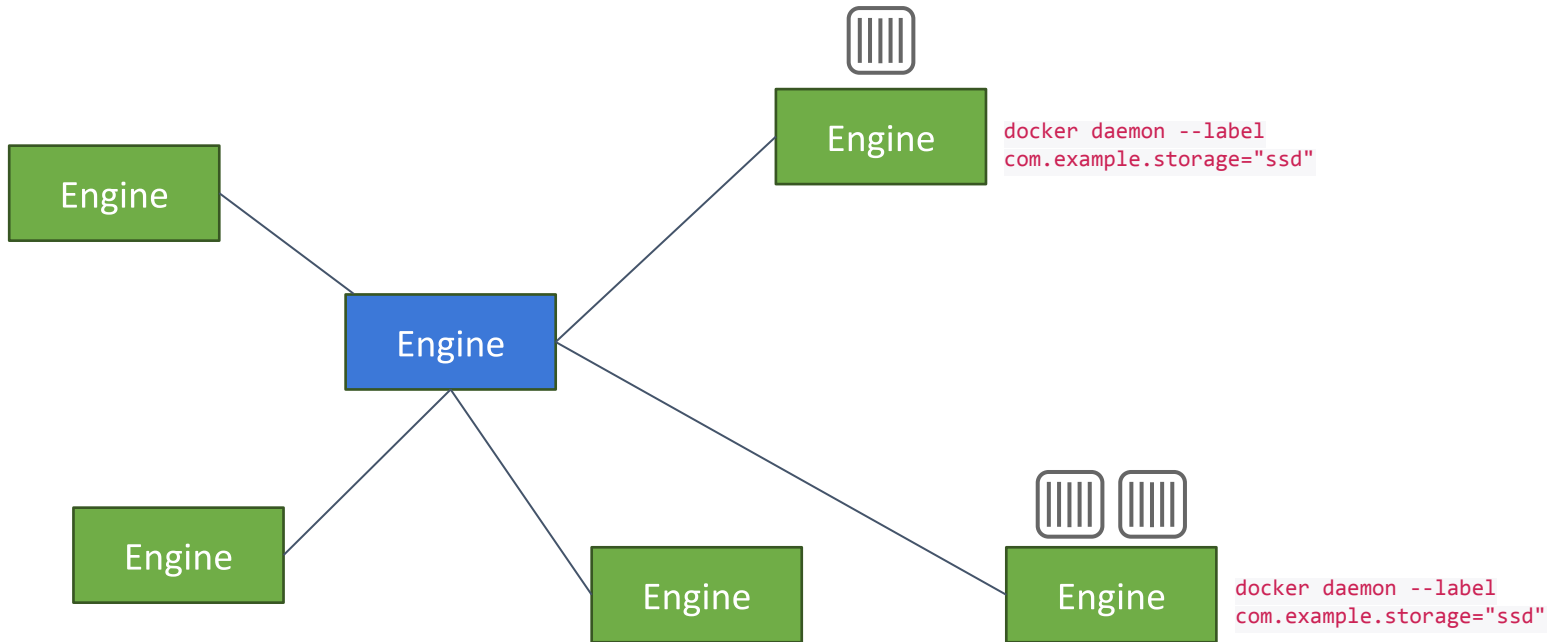


 `$ docker service create --mode=global --name prometheus prom/prometheus`

Constraints

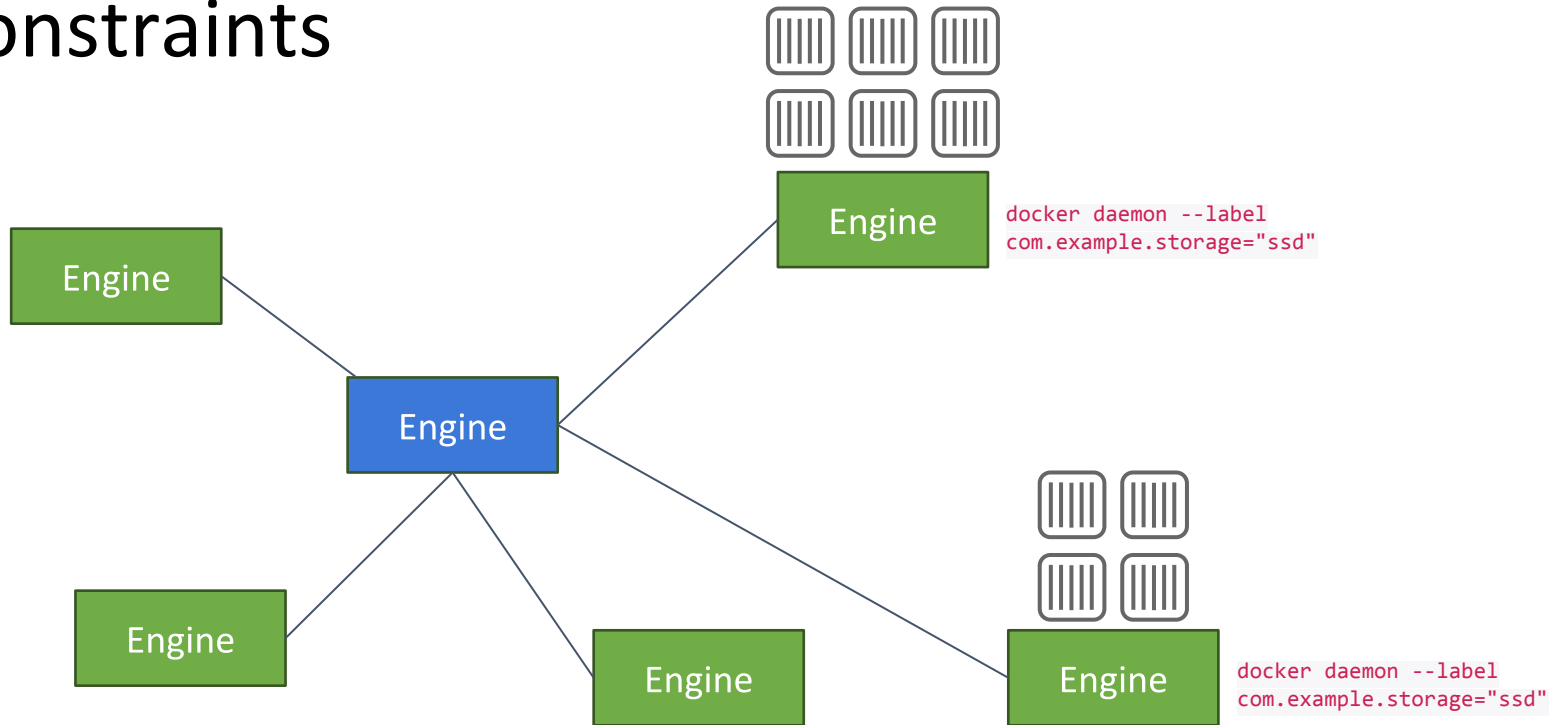


Constraints



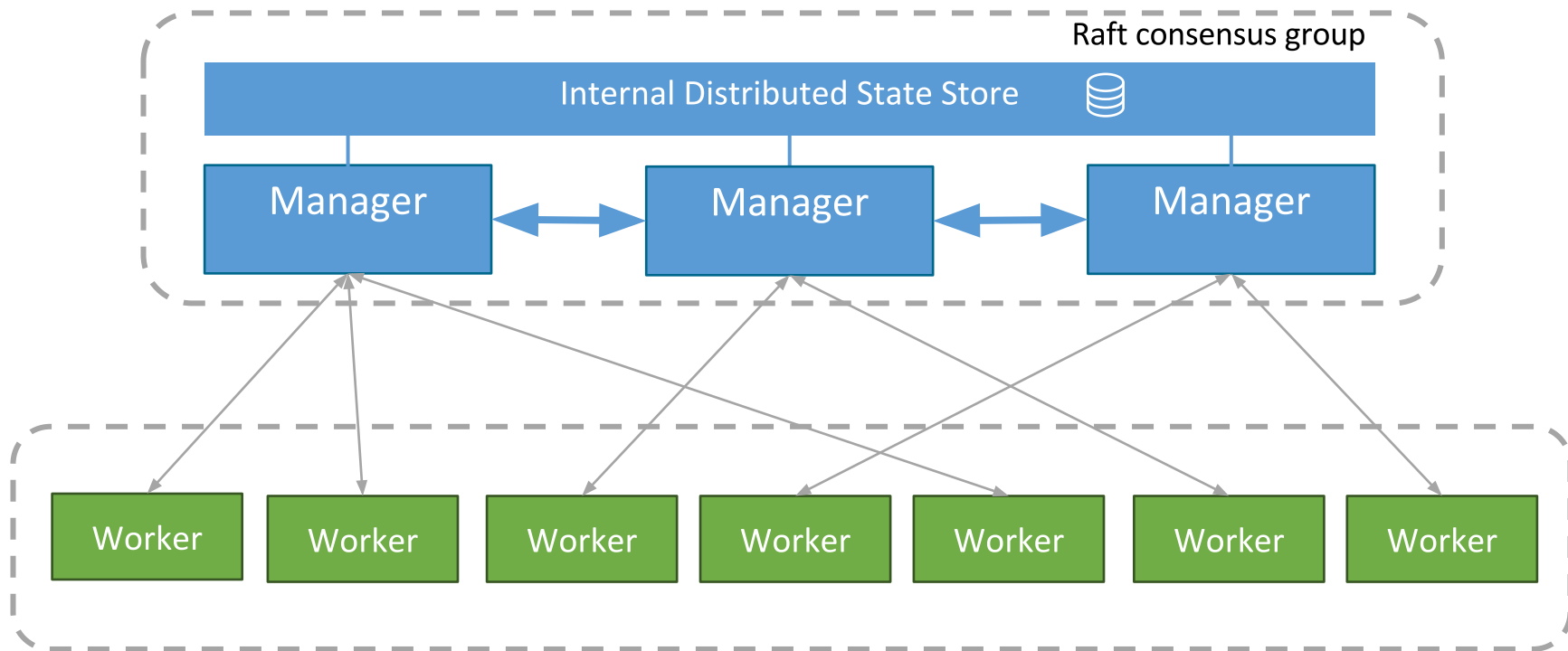
```
$ docker service create --replicas 3 --name frontend --network mynet  
--publish 80:80/tcp --constraint engine.labels.com.example.  
storage==ssd frontend_image:latest
```


Constraints

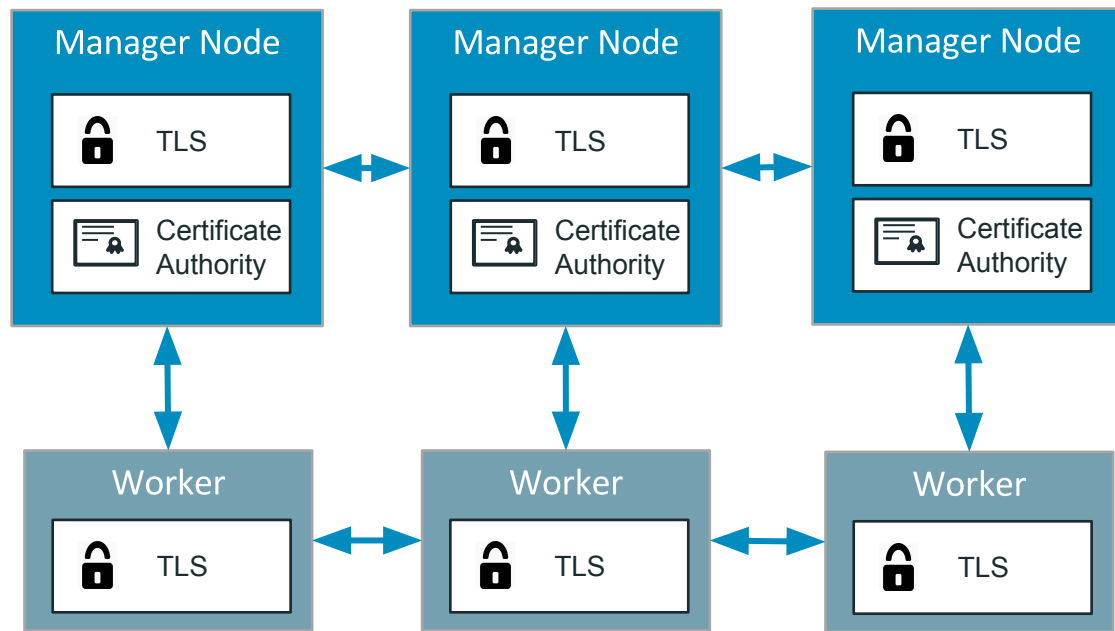


```
$ docker service create --replicas 3 --name frontend --network mynet  
--publish 80:80/tcp --constraint engine.labels.com.example.  
storage==ssd frontend_image:latest  
$ docker service scale frontend=10
```

Docker Swarm Communication Internals



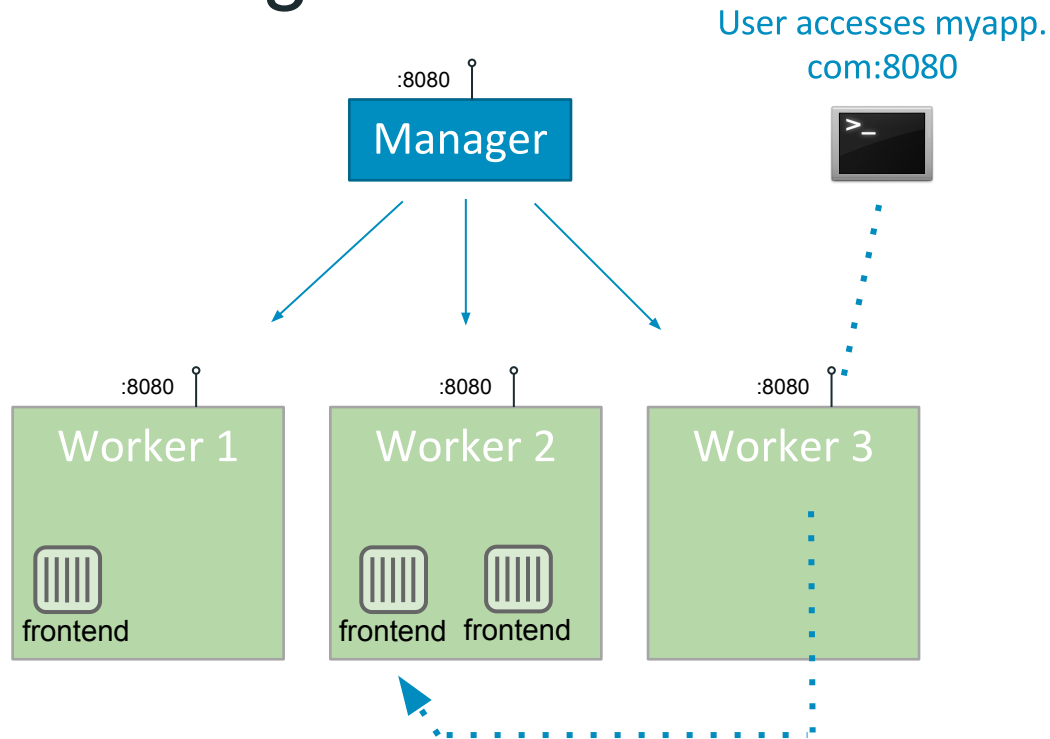
Secure by default with end to end encryption



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



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
```



Questions?

Pablo Chico de Guzmán
pchico83@gmail.com / @chico_de_guzman

