Docker Swarm

Blacksburg Docker Meetup

November 9, 2016

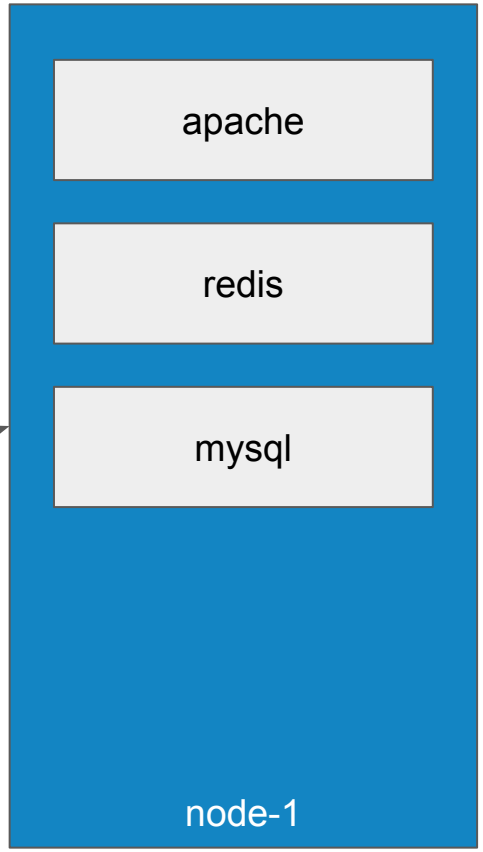
Quick review...

- What’s an image?
- What’s a container?
- What’s Docker Compose?

Docker Compose Shortcomings

- Only runs containers on a single host
  - Works fantastic locally (development)
  - Single host = not preferred for prod

```
version: '2'
services:
  apache:
    image: php:7.0-apache
  redis:
    image: redis:3.0
  mysql:
    image: mysql:5.7
```

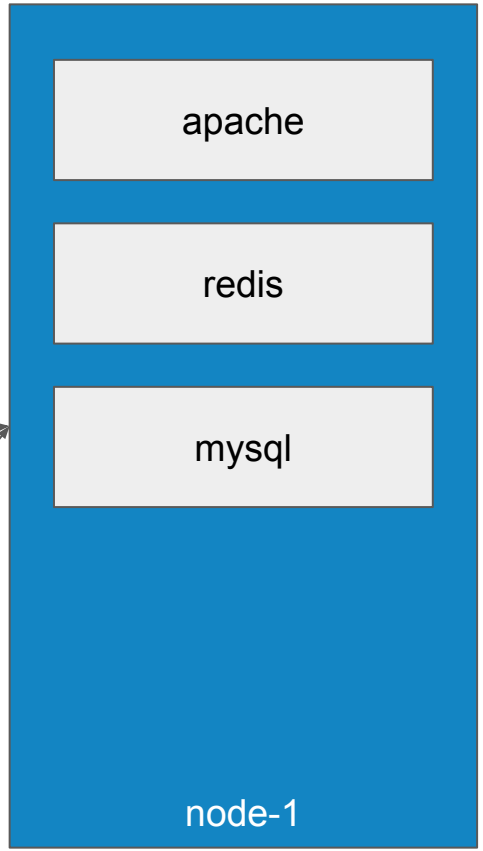


Quick V1 vs V2 Review...

- What's the difference??

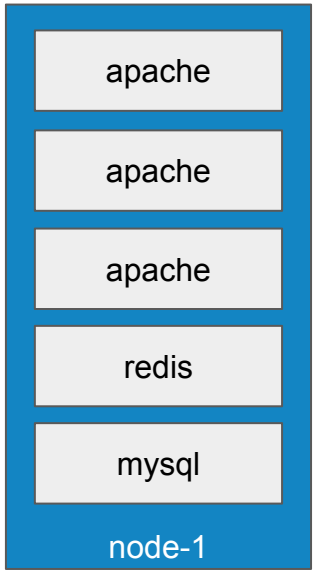
```yaml
version: '2'
services:
  apache:
    image: php:7.0-apache
  redis:
    image: redis:3.0
  mysql:
    image: mysql:5.7
```

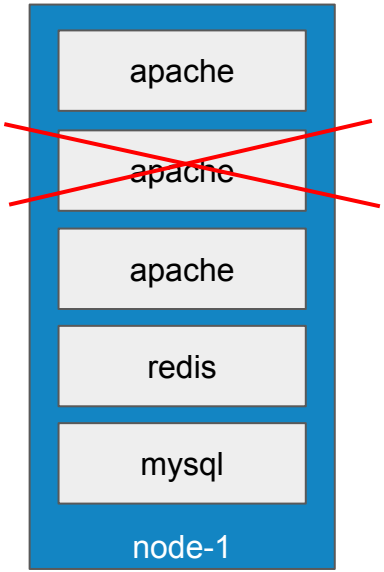
```yaml
apache:
  image: php:7.0-apache
redis:
  image: redis:3.0
mysql:
  image: mysql:5.7
```

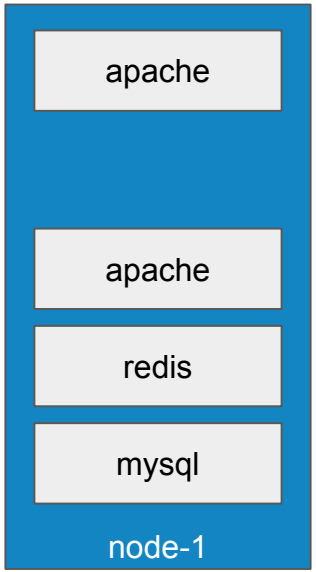


Docker Compose Shortcomings

- Purely “fire and forget”

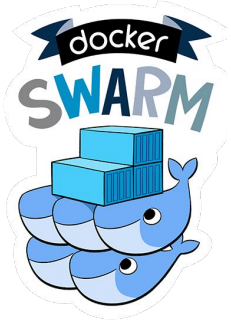







```
docker-compose scale apache=3
```

Enter the world of orchestration...

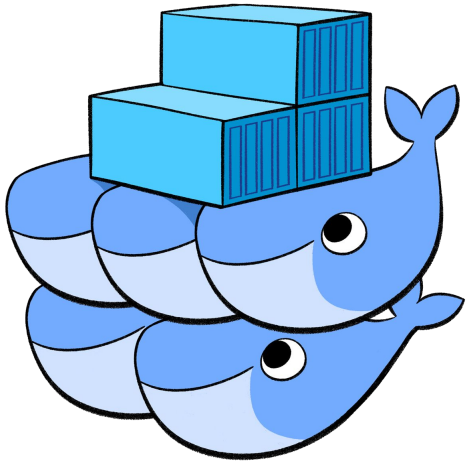


Introducing Docker Swarm

- Docker Swarm was pulled into the Docker Engine with Docker 1.12
  - There was a previous version of Swarm, but was a lot harder to use
- Completely optional to use
- BUT... comes with a ton of cool features
  - Swarm mode
  - Secure by default
  - Routing mesh
  - Service API

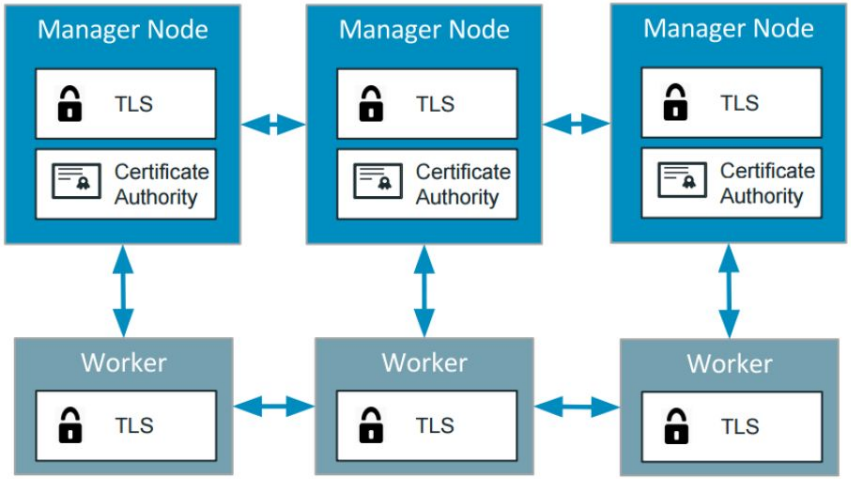
Swarm Mode

- Self-organizing and self-healing
- Completely decentralized architecture
  - No external data store required
  - No single points of failure



Secure by Default

- Cryptographic node identity
- Automatic encryption and mutual auth (TLS)
  - End-to-end TLS
- Automatic cert rotation
- External CA integration
- Built-in government-grade PKI



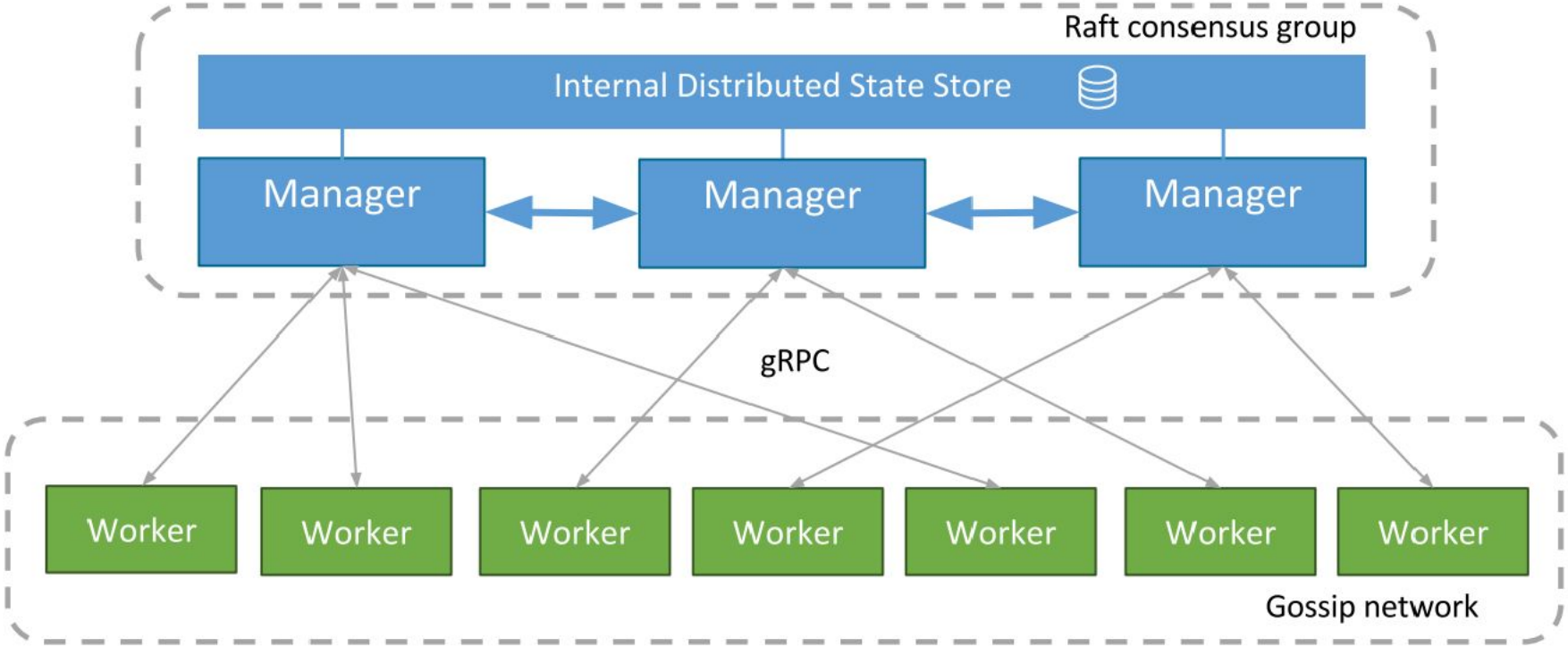
Routing Mesh

- Swarm-wide overlay network
- Container-native load-balancing
- DNS-based service discovery
- Kernel-only data path using IPVS
- Every node knows how to route to any other service in the swarm

Docker Service API

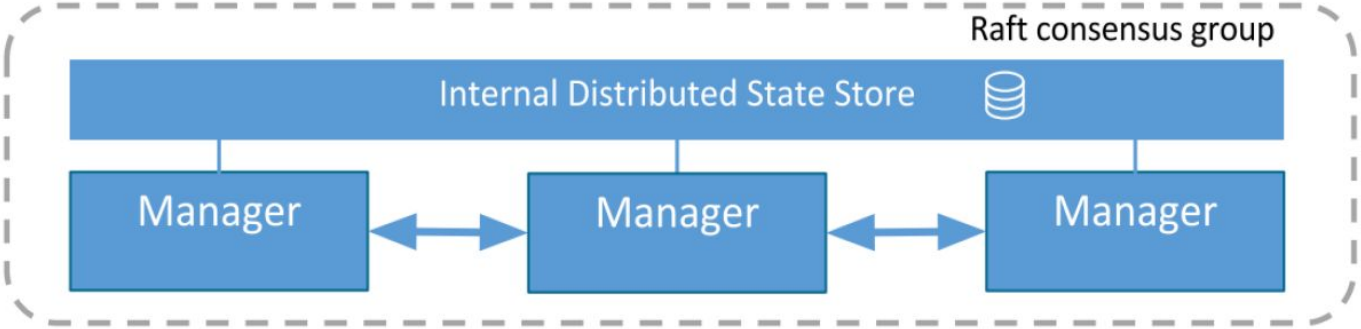
- New CLI commands to set desired state
- Allows for...
  - Scaling
  - Rolling updates
  - Advanced scheduling
  - App-specific healthchecks
  - Rescheduling on node failure

Internal Communications



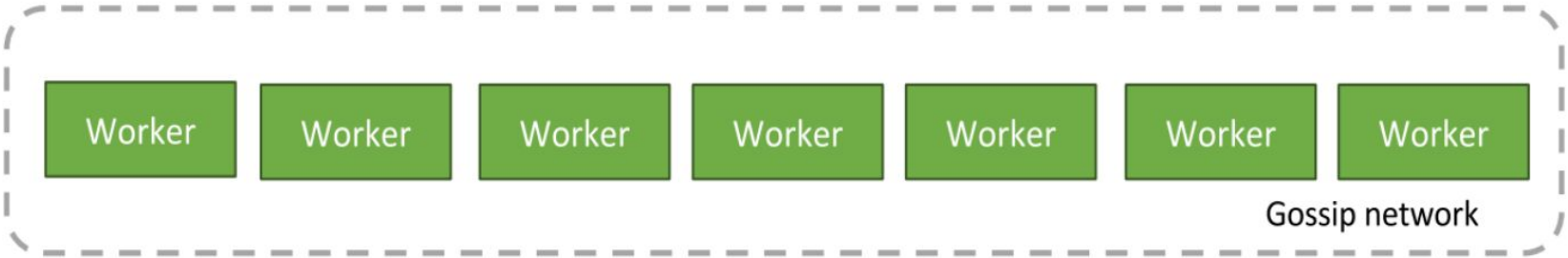
Quorum Layer (Managers)

- Strongly consistent - holds desired state
- Simple to operate using Docker CLI
- Blazing fast (in-memory reads, domain specific indexing)
- Secure communications

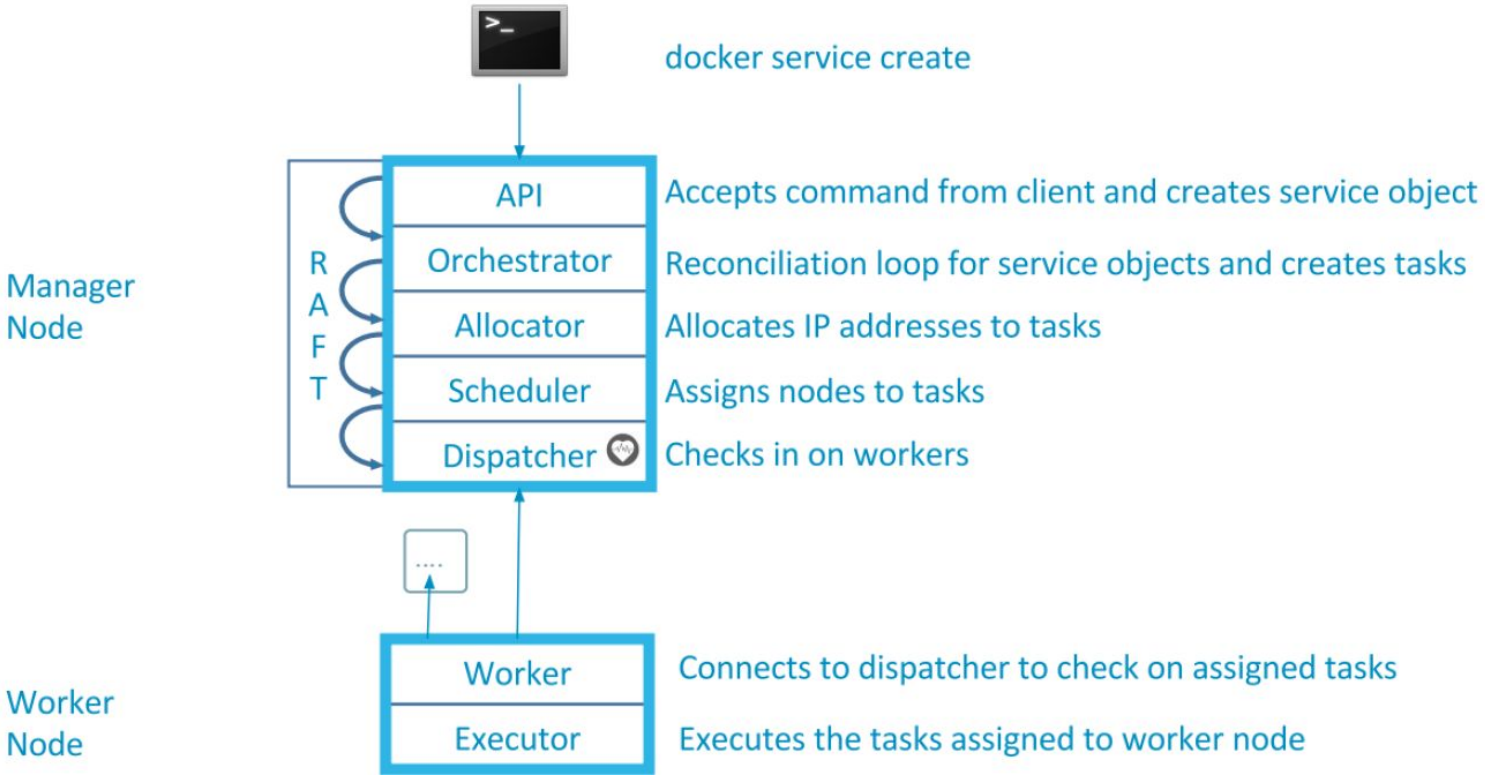


Work-to-Worker Gossip

- Eventually consistent: Routing mesh, load balancing rules
- High volume, P2P network
- Secure - symmetric encryption with key rotation

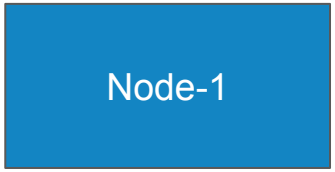


Under the Hood...



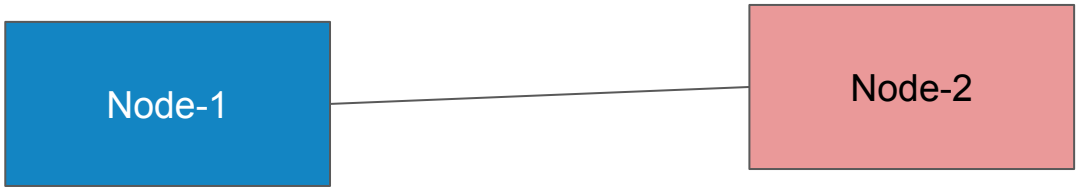
Let’s see it in action...

Swarm initialization



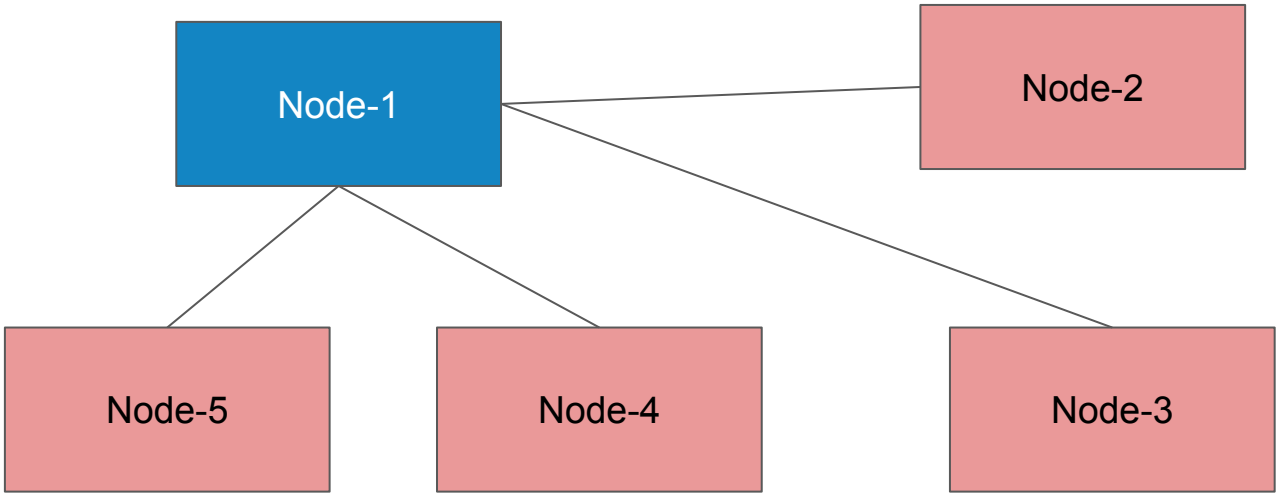

```
docker swarm init
```

Adding to the cluster



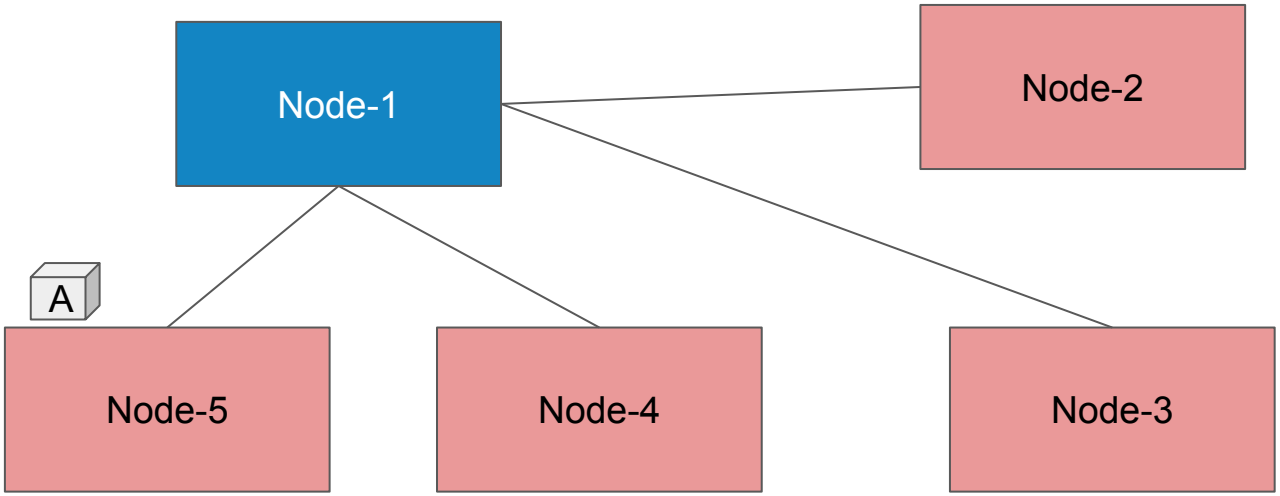

```
docker swarm join <IP of node-1>:2377
```

Adding to the cluster



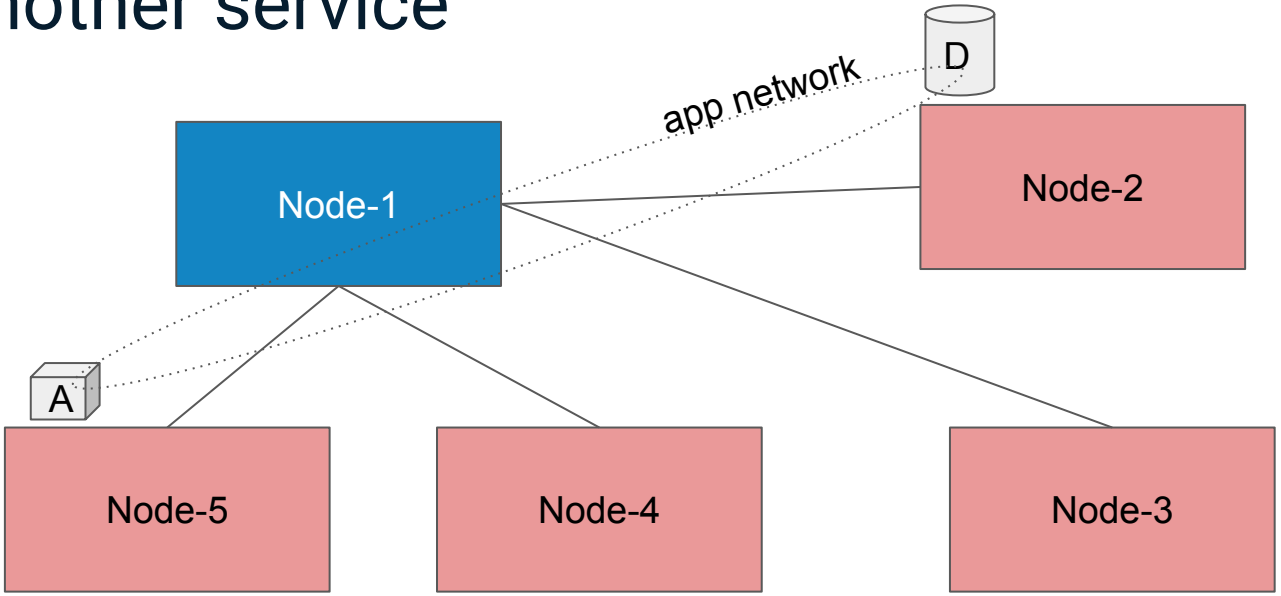

```
docker swarm join <IP of node-1>:2377
```

Starting a service



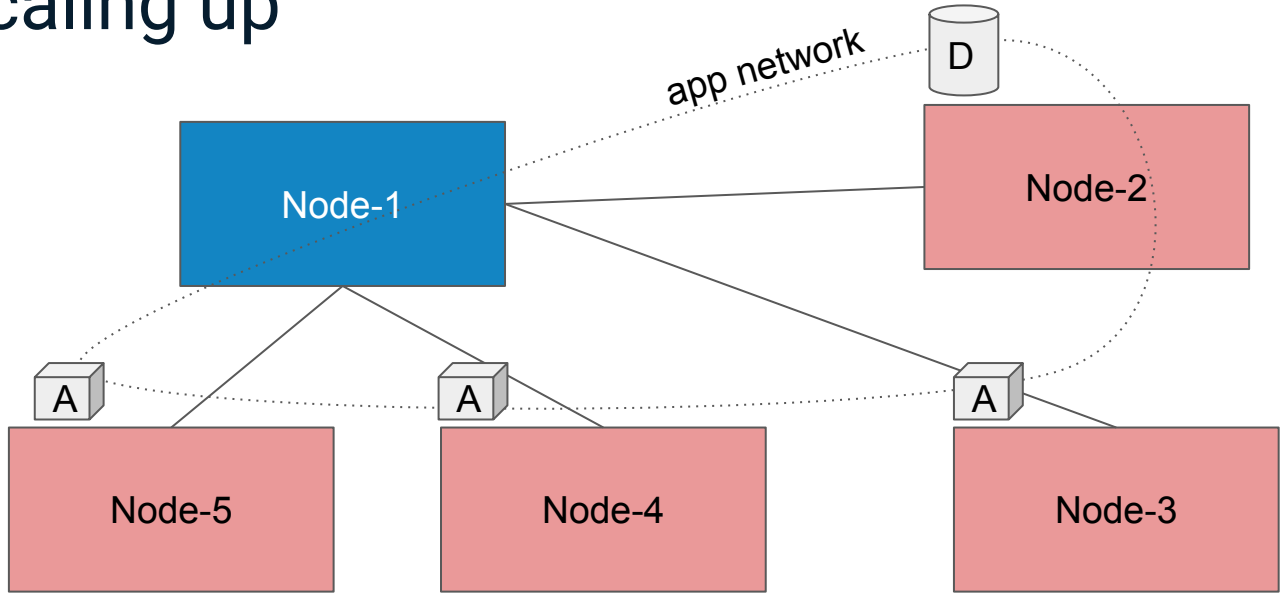

```
docker service create --name app --network app -p 80:80 myapp
```

Another service



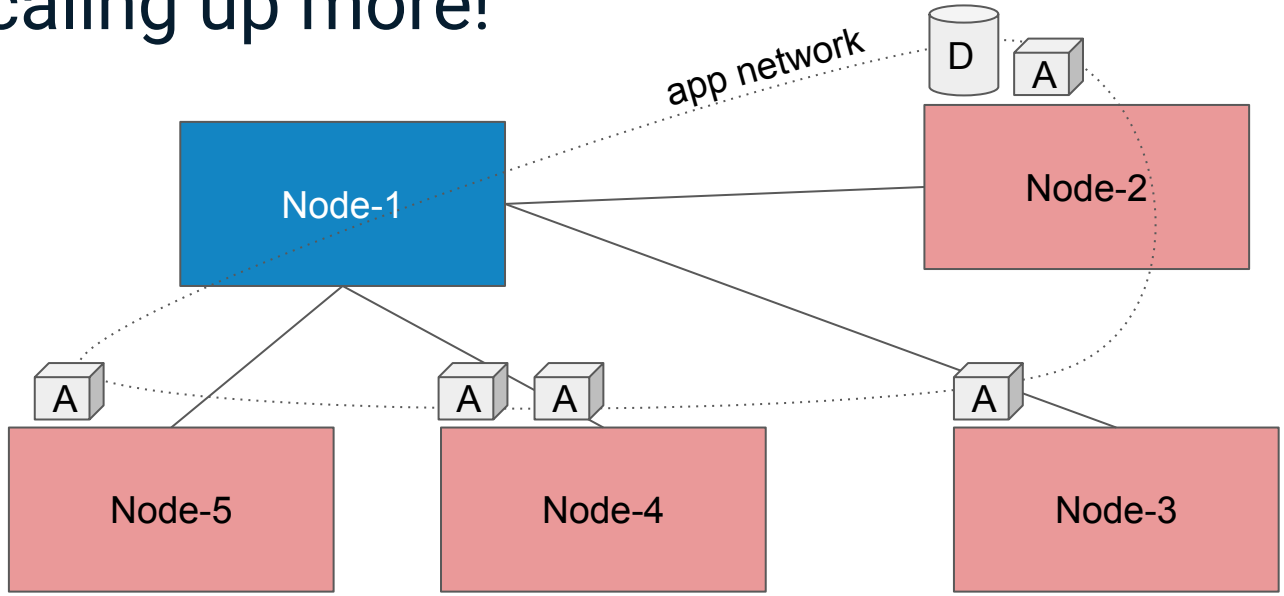

```
docker service create --name db --network app mysql
```

Scaling up



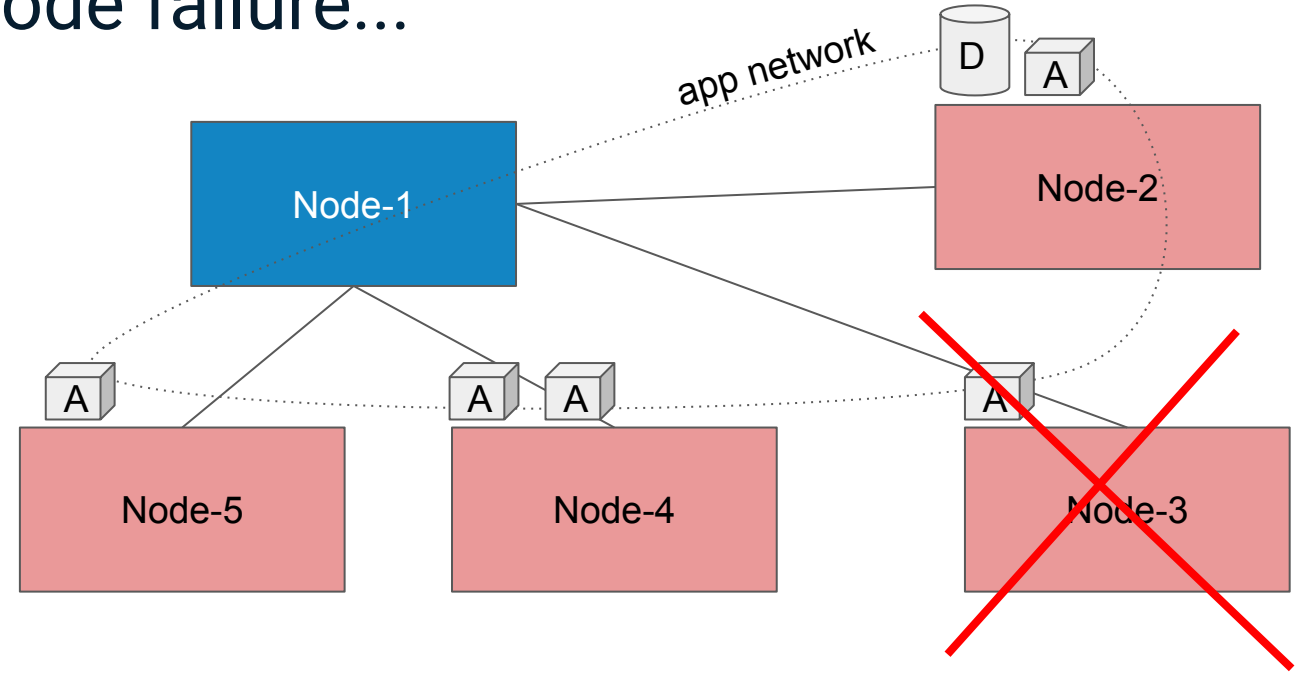

```
docker service scale app=3
```

Scaling up more!

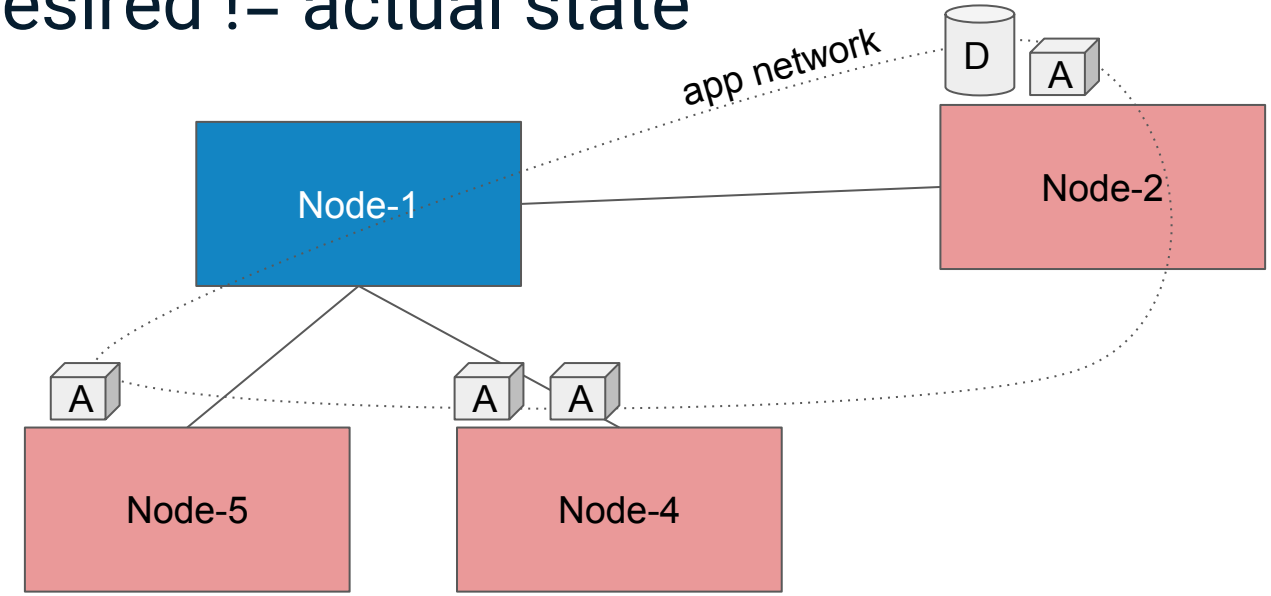



```
docker service scale app=5
```

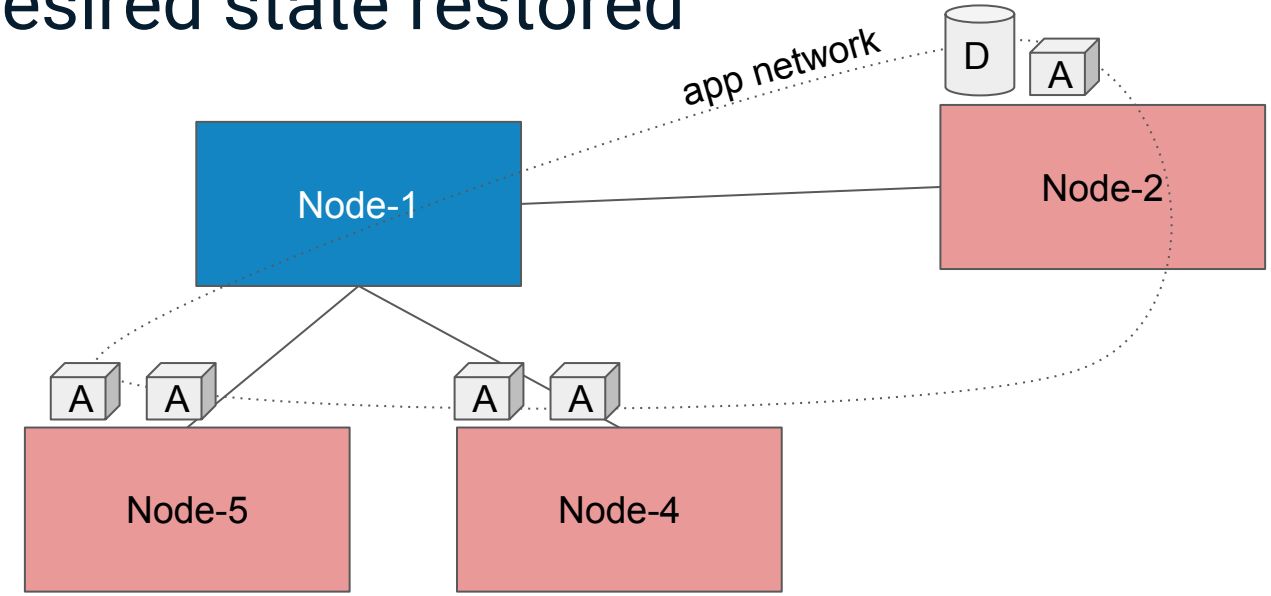
Node failure...



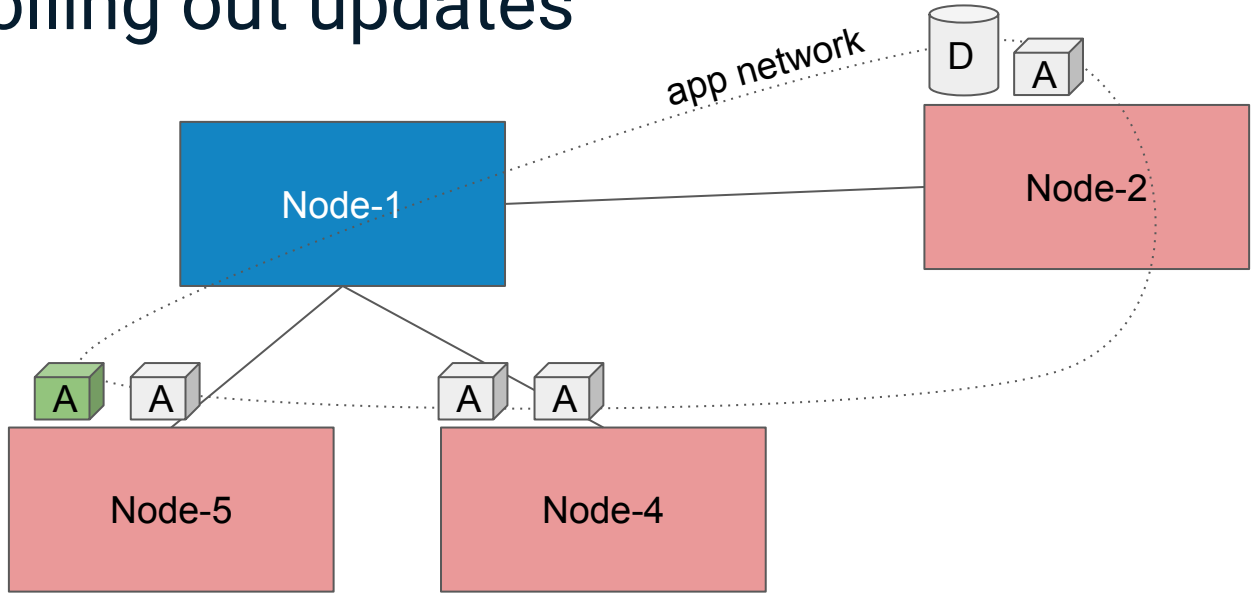
Desired != actual state



Desired state restored

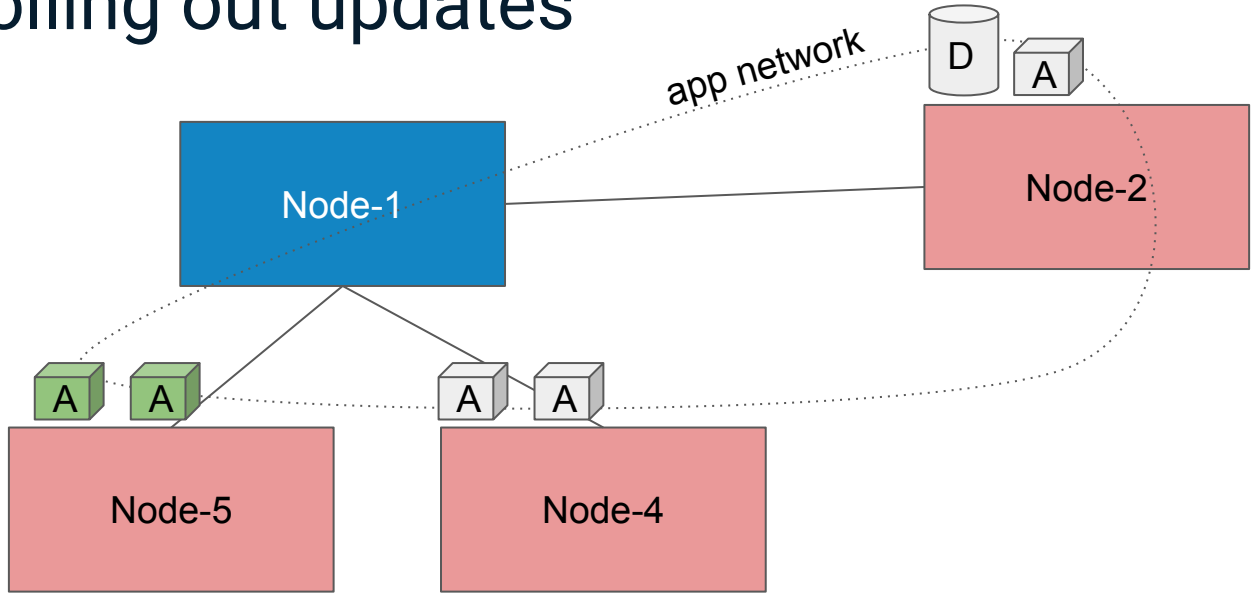


Rolling out updates



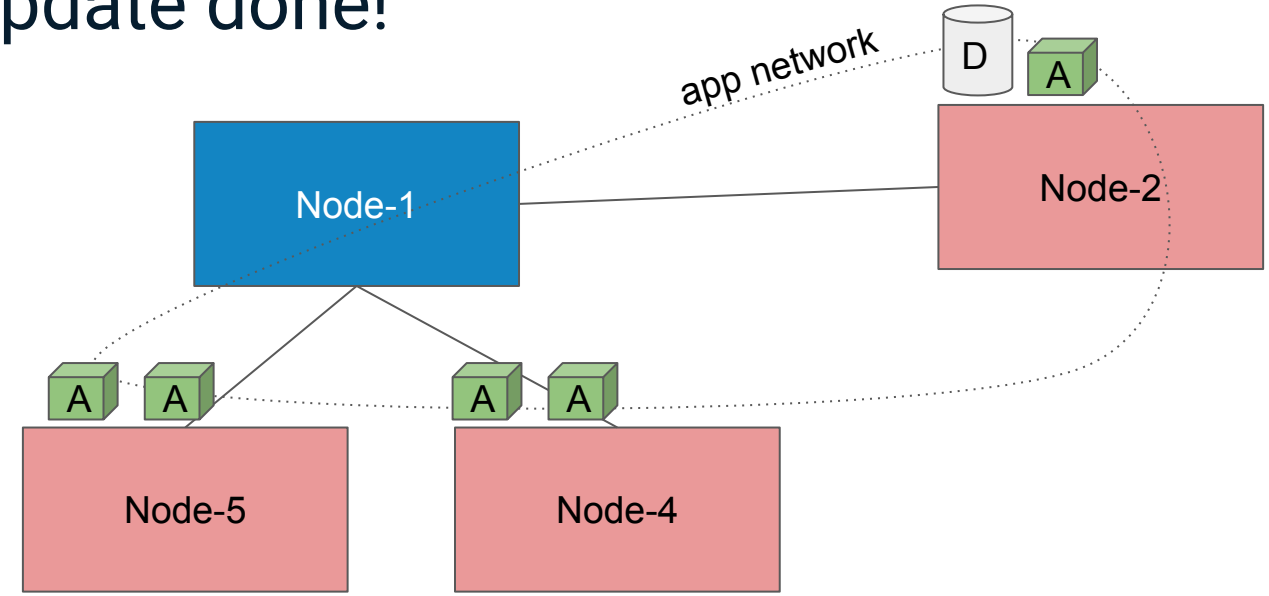

```
docker service update --image=app:1.1 --update-delay 10s app
```

Rolling out updates

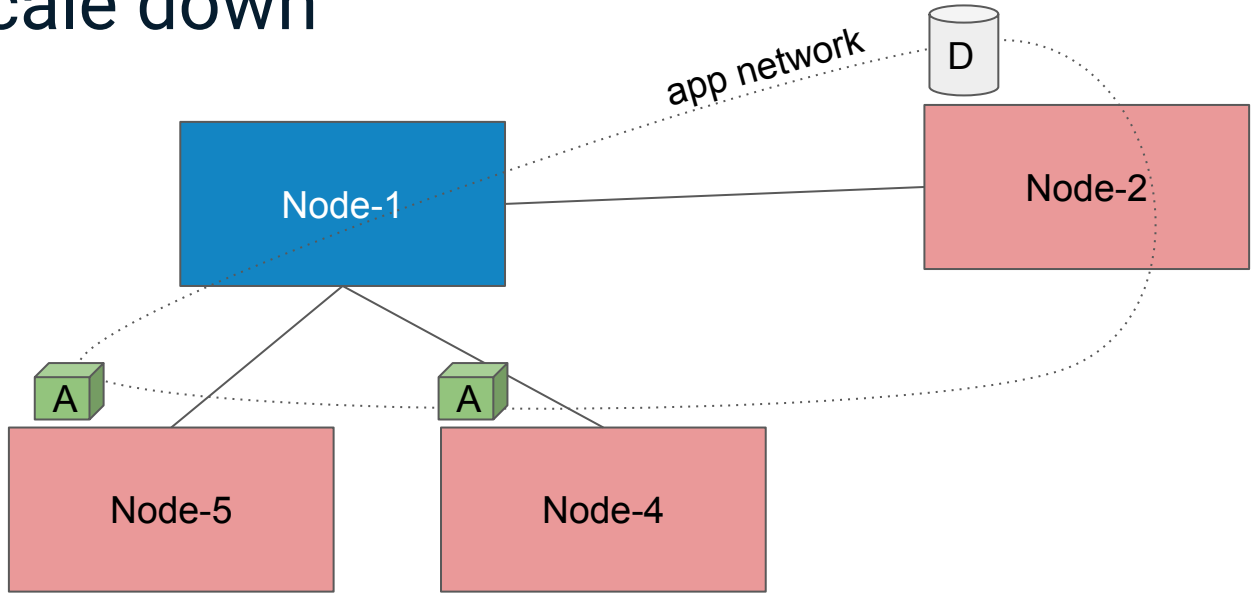


Updates one, waits until it’s in RUNNING state, then waits 10 seconds to move on to next task

Update done!

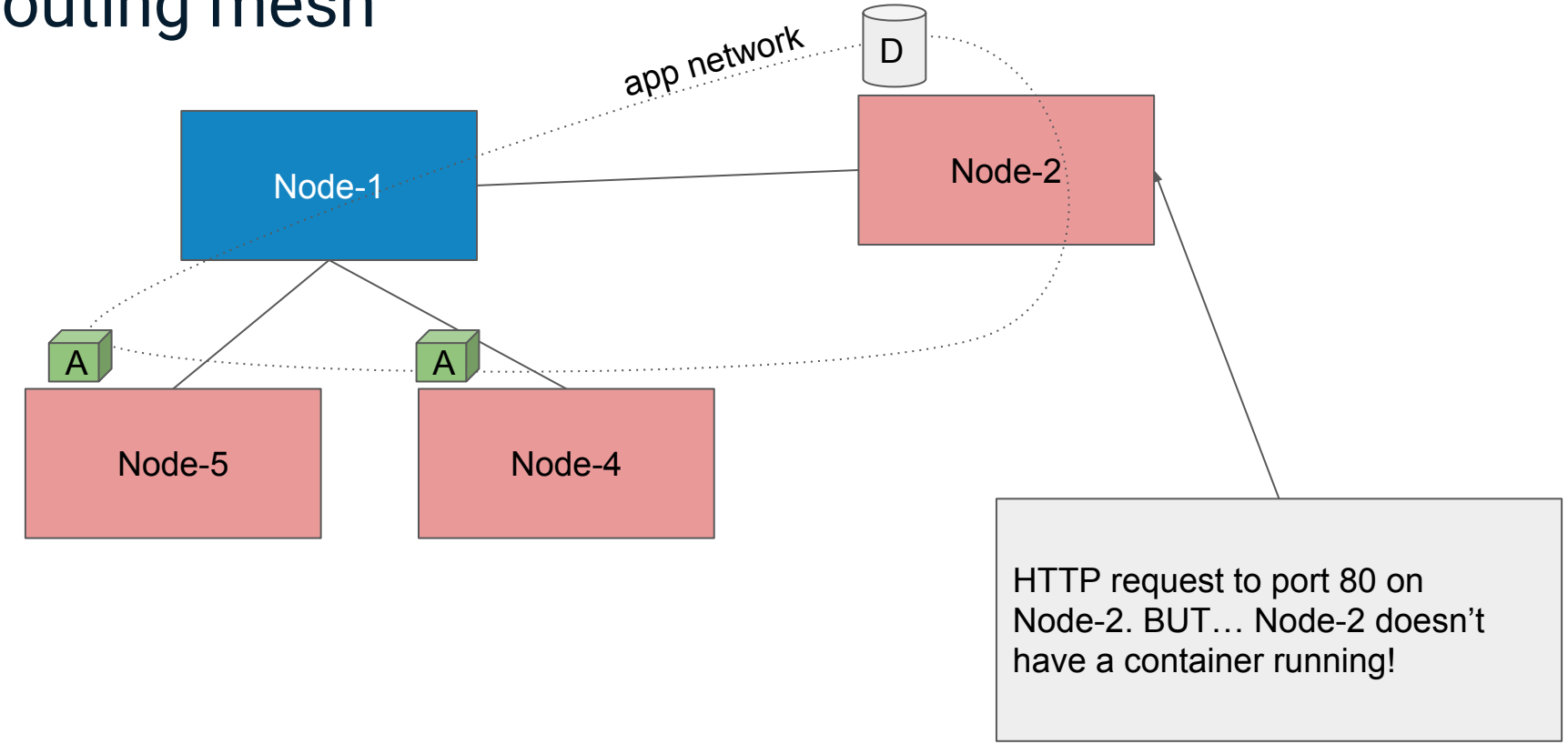


Scale down

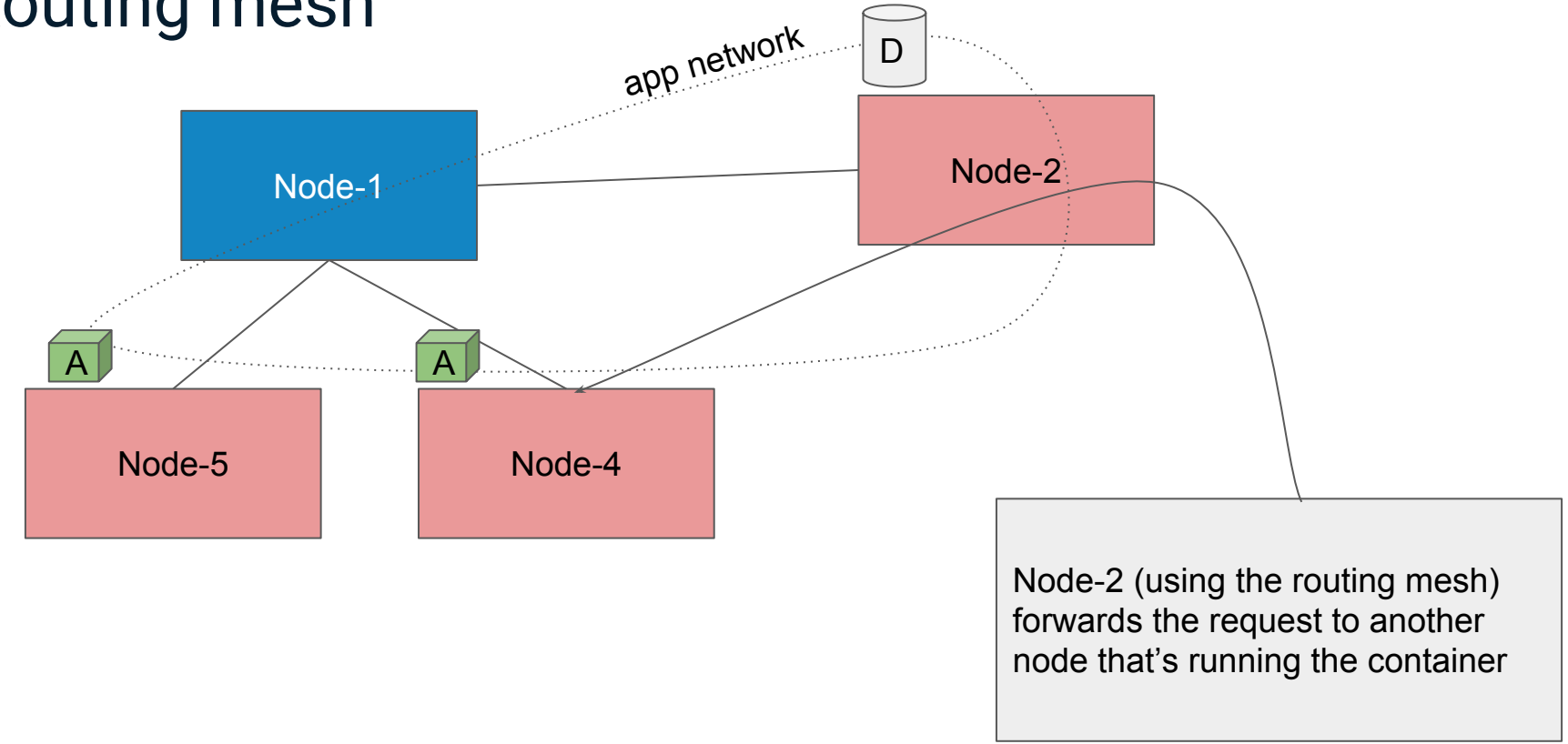



```
docker service scale app=2
```

Routing mesh

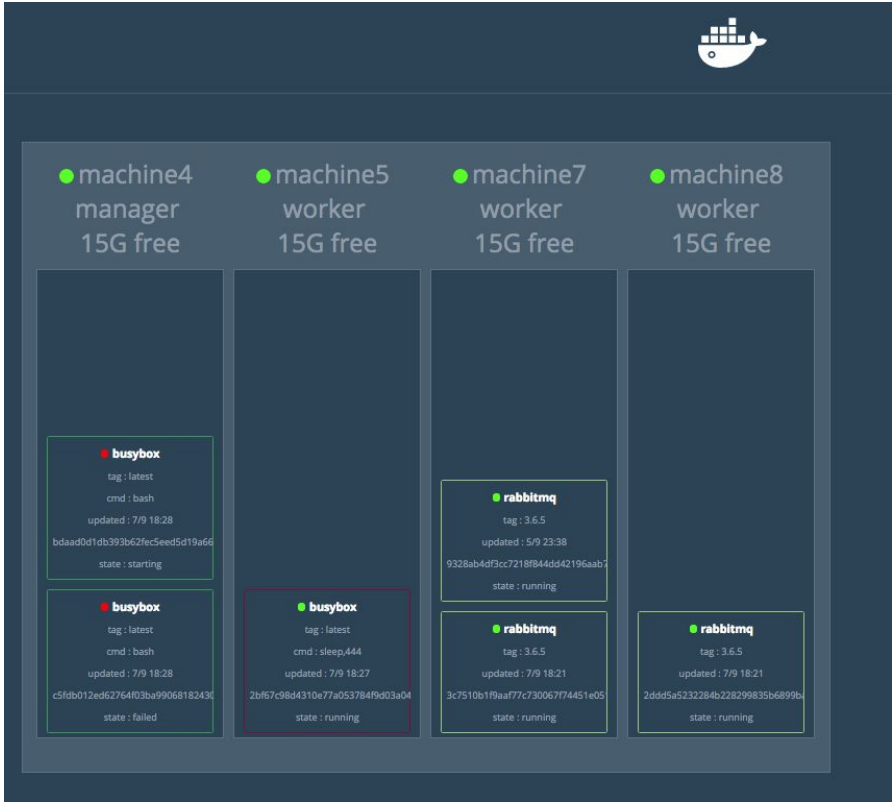


Routing mesh



Swarm Visualizer

- Mano Marks (works at Docker) created a tool to help visualize what's going on in a Swarm
- https://github.com/ManoMarks/docker-swarm-visualizer



Swarm recap

- Built-in orchestration that overcomes “fire and forget”
- Requires user to specify a desired state
  - Will use scheduling to get to desired state
  - Self-heals in case of node failure to restore desired state
- Routing mesh ensures traffic gets to a container
  - Allows multiple tasks to run on same machine, even when exposing the same port
- Secure by default

Questions?