What’s new in Swarm 1.1

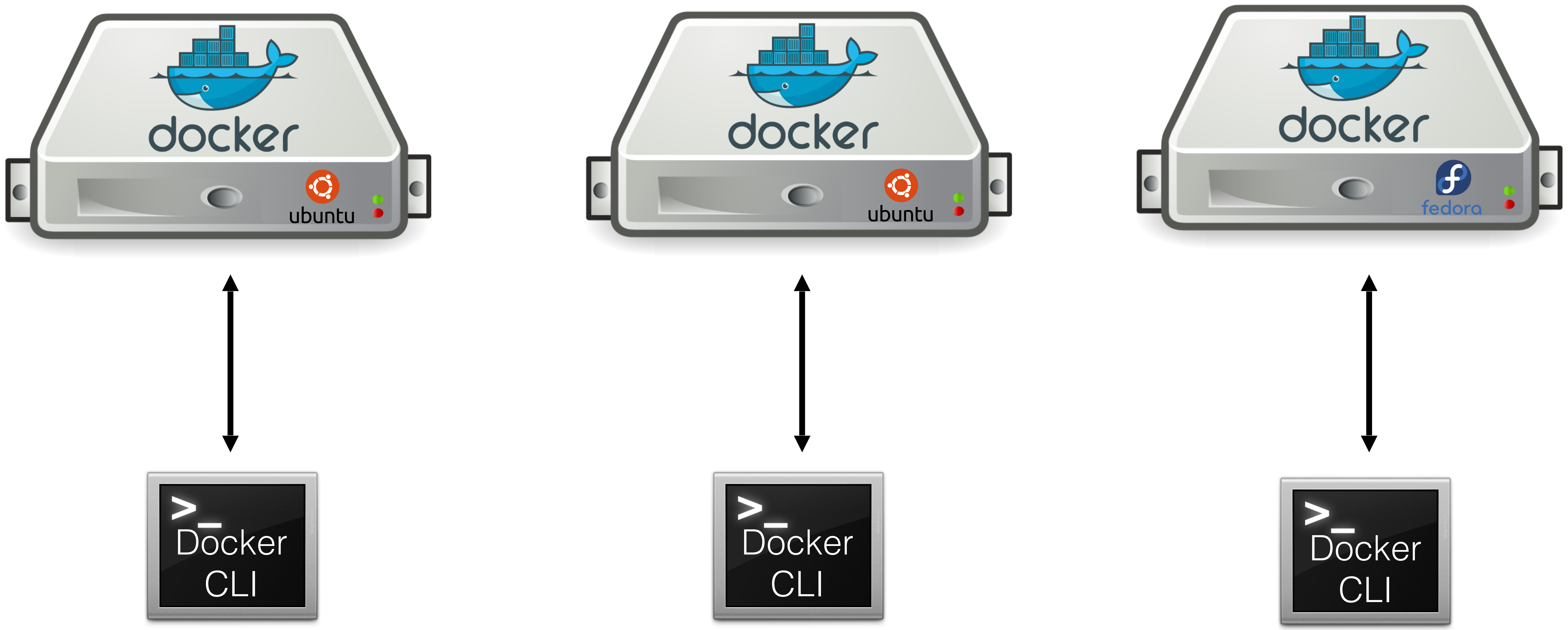
the Docker-native clustering system

Docker San Mateo meetup - 02/17/2016

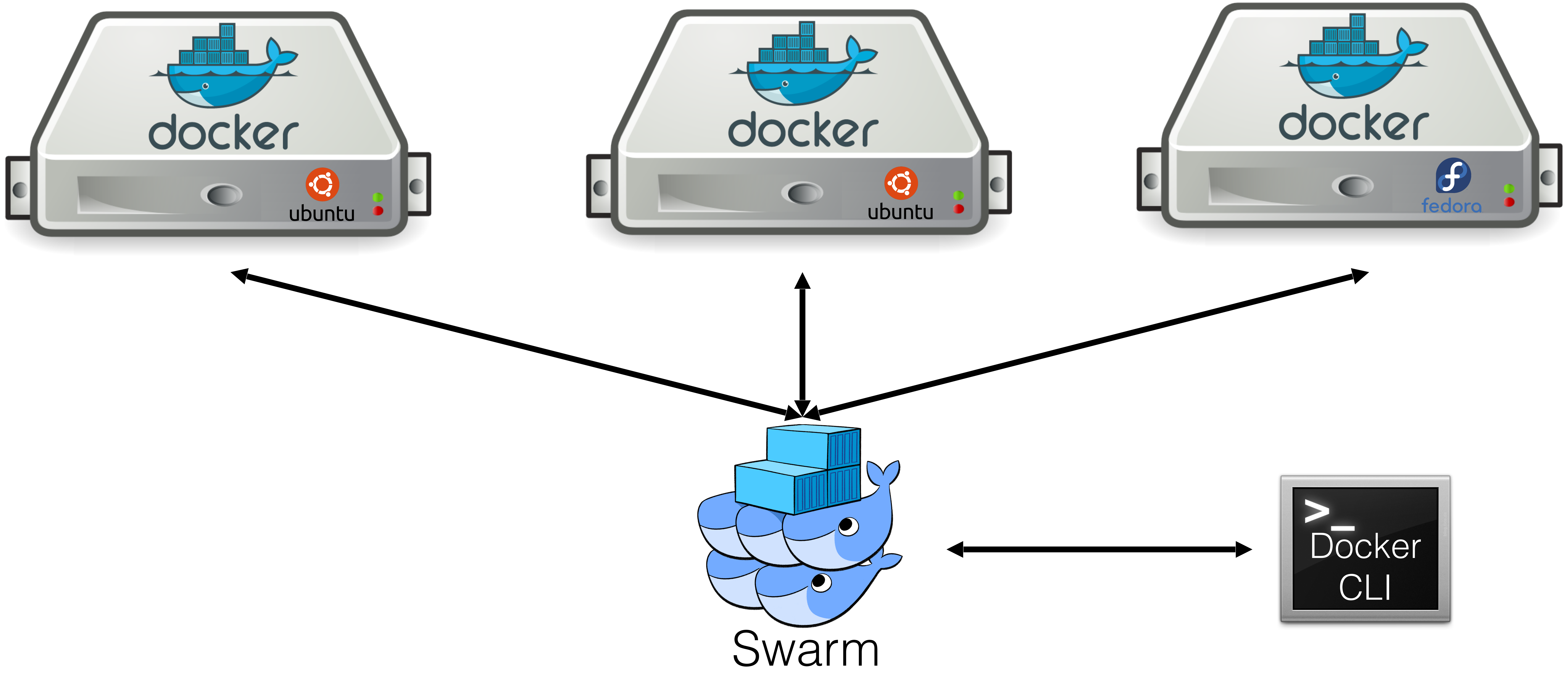
@vieux

Running containers on multiple hosts

Without Docker Swarm



With Docker Swarm



Swarm in a nutshell

- Exposes several Docker Engines as a single virtual Engine
- Serves the standard Docker API
- Extremely easy to get started
- Batteries included but swappable

Setup: Without networking

Setup using the hosted discovery service

/!\ Not to be used in production, for testing only /!

- Create a cluster:
```
$ swarm create
```
- Add nodes to a cluster:
```
$ swarm join  --advertise=<engine_ip>:<engine_port> token://<token>
```
- Start Swarm
```
$ swarm manage <...> token://<token>
```

Setup using your own K/V store

- Add nodes to a cluster:

  ```
  $ swarm join  --advertise=<engine_ip>:<engine_port> \
                            consul://<ip_consul>:<port_consul>
  ```

- Start Swarm

  ```
  $ swarm manage <...> consul://<ip_consul>:<port_consul>
  ```

You can also use etcd or zookeeper

Setup using a file (static list of nodes)

- Add nodes to the file:

```
$ echo 10.0.0.1:2375 > my_cluster
$ echo 10.0.0.2:2375 >> my_cluster
$ echo 10.0.0.3:2375 >> my_cluster
```

- Start Swarm

```
$ swarm manage <...> file://my_cluster
```

Setup: With networking

Setup using your own K/V store

- Configure networking on engine

```
$ docker daemon --cluster-advertise=<engine_ip>:<engine_port>  \
                          --cluster-store=consul://<ip_consul>:<port_consul>
```

- Start Swarm

```
$ swarm manage <…> --discovery-opt kv.path=docker/docker \
                                          consul://<ip_consul>:<port_consul>
```

You can also use etcd or zookeeper

Docker Swarm internals

Resource Management

- Memory
  ```
  $ docker run -m 1g ...
  ```
- CPU
  ```
  $ docker run -c 1 ...
  ```
- Ports
  ```
  $ docker run -p 80:80 ...
  ```

Constraints

- Standard constraints induced from docker info

```
docker run -e “constraint:operatingsystem==*fedora*” ...
docker run -e “constraint:storagedriver==*aufs*” ...
```

- Custom constraints with host labels

```
docker daemon --label “region=us-east”
docker run -e “constraint:region==us-east” ...
```

- Pin a container to a specific host

```
docker run –e “constraint:node==node-2” ...
```

Affinities

- Containers affinities

```
docker run --name web nginx
docker run -e “affinity:container==web” logger
```

- Containers Anti-affinities

```
docker run --name redis-master redis
docker run --name redis-slave -e “affinity:container!=redis*” ...
```

- Images affinities

```
docker run -e “affinity:image==redis” redis
```

Soft Affinities/Constraints

- Soft Contraints

```
docker run -e “constraint:operatingsystem==~*fedora*” …
docker run -e “constraint:region==~us-east” …
```

- Soft Affinities

```
docker run --name redis-master redis
docker run --name redis-slave -e “affinity:container!=~redis*” …
```

Swarm Scheduler

2 steps:

- 1- Apply filters to exclude nodes
  - ports
  - constraints
  - affinity
  - health
  - dependency
- 2- Use a strategy to rank and pick the best node
  - binpack
  - spread
  - random

New in Swarm 1.1

- **Improved node management**
- **Rescheduling** (EXPERIMENTAL)
- **New events**

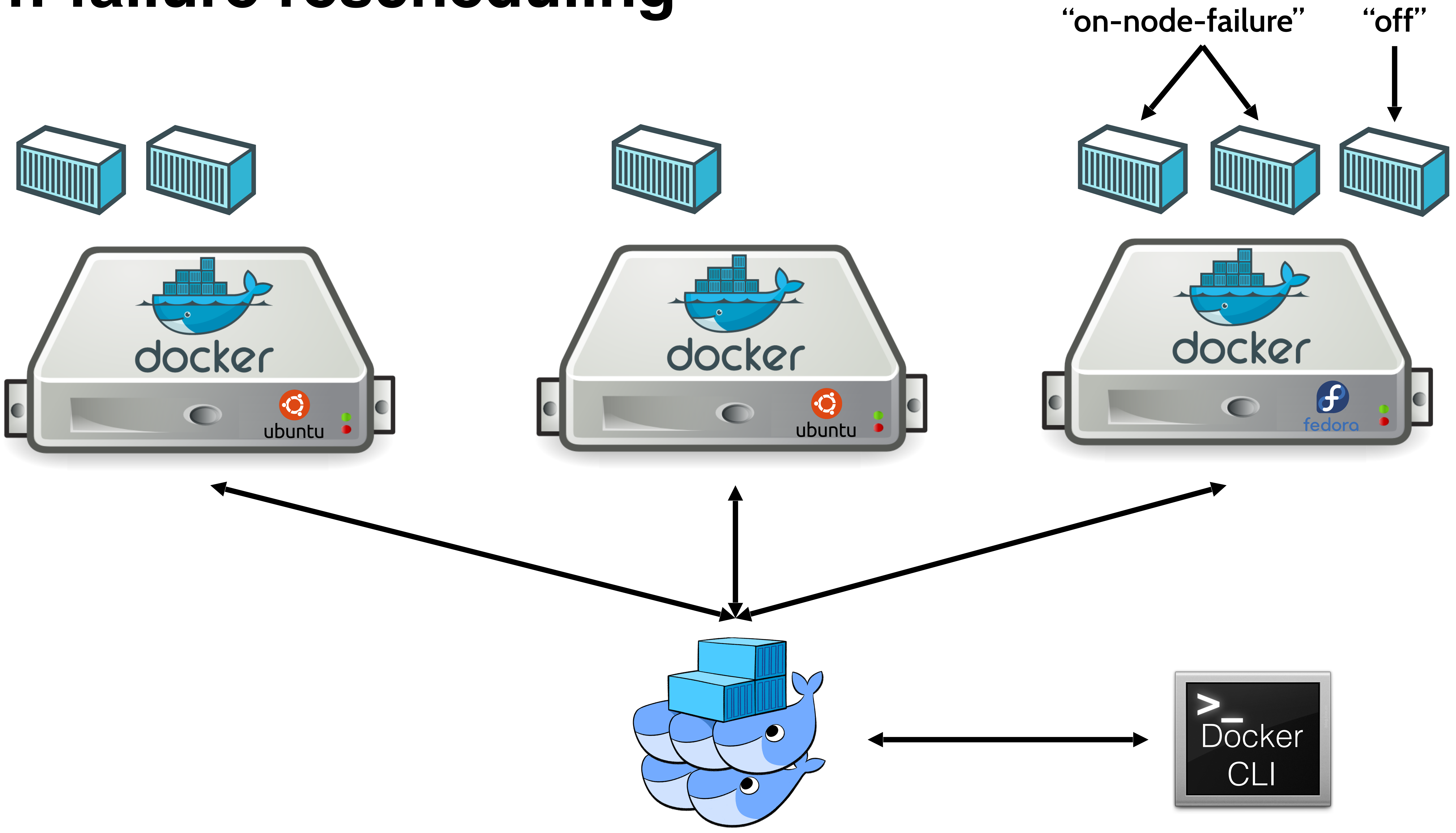
Improved node management: docker info

```
Nodes: 3
(unknown): 10.0.0.9:2375
  ...
  └ Status: Pending
node-1: 10.0.0.1:2375
  └ Status: Unhealthy
  ...
  └ Error: Cannot connect to the docker engine endpoint
  └ UpdatedAt: 2016-02-09T19:52:56Z
node-2: 10.0.0.0:2375
  └ Status: Healthy
  ...
  └ Labels: kernelversion=4.2.0-23-generic, operatingsystem=Ubuntu 14.04.3 LTS, ...
  └ Error: (none)
  └ UpdatedAt: 2016-02-09T19:52:56Z
```

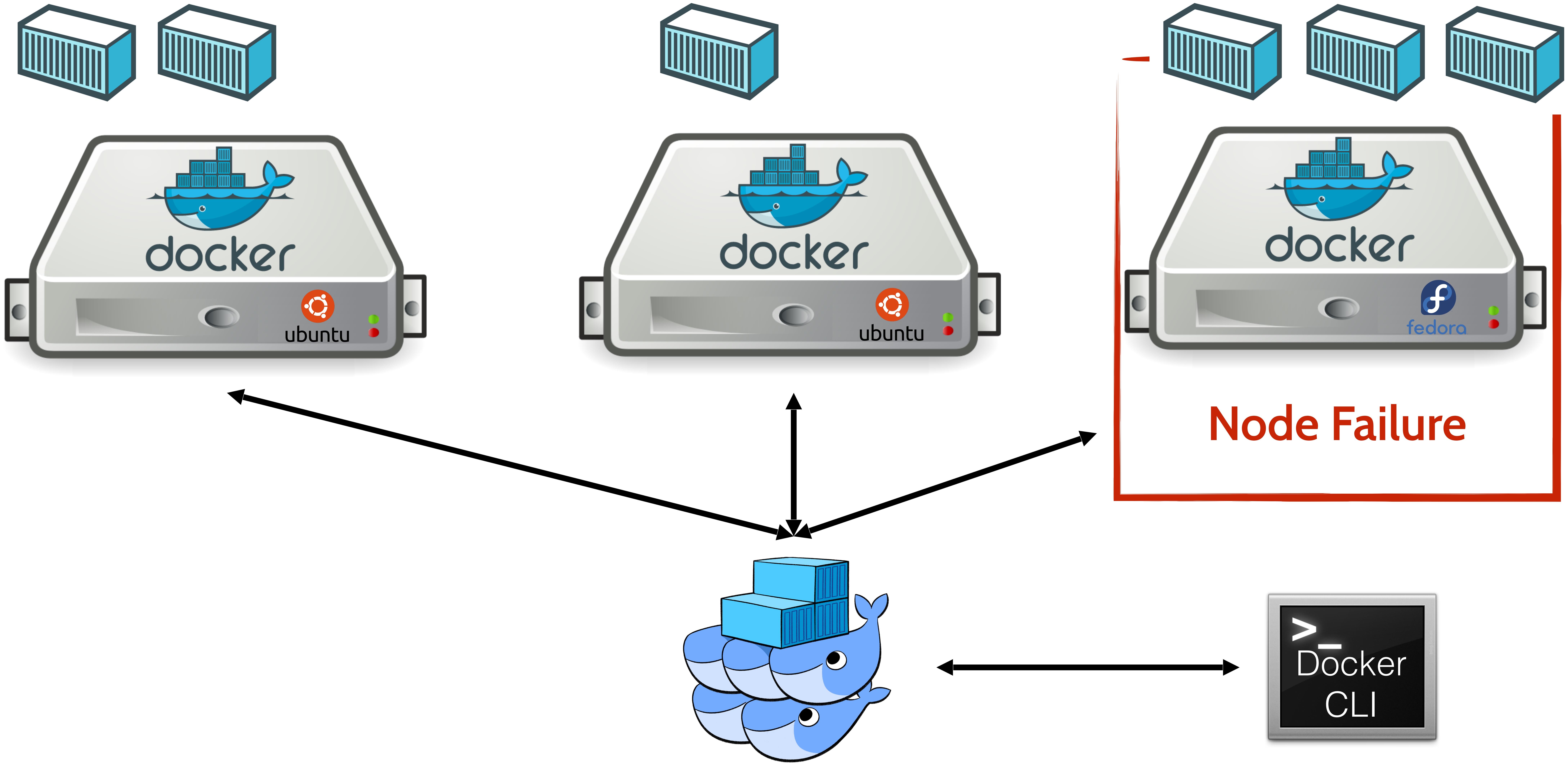
Rescheduling

- Experimental feature
  swarm --experimental manage …
- On node failure
  docker run -e “reschedule:on-node-failure” …

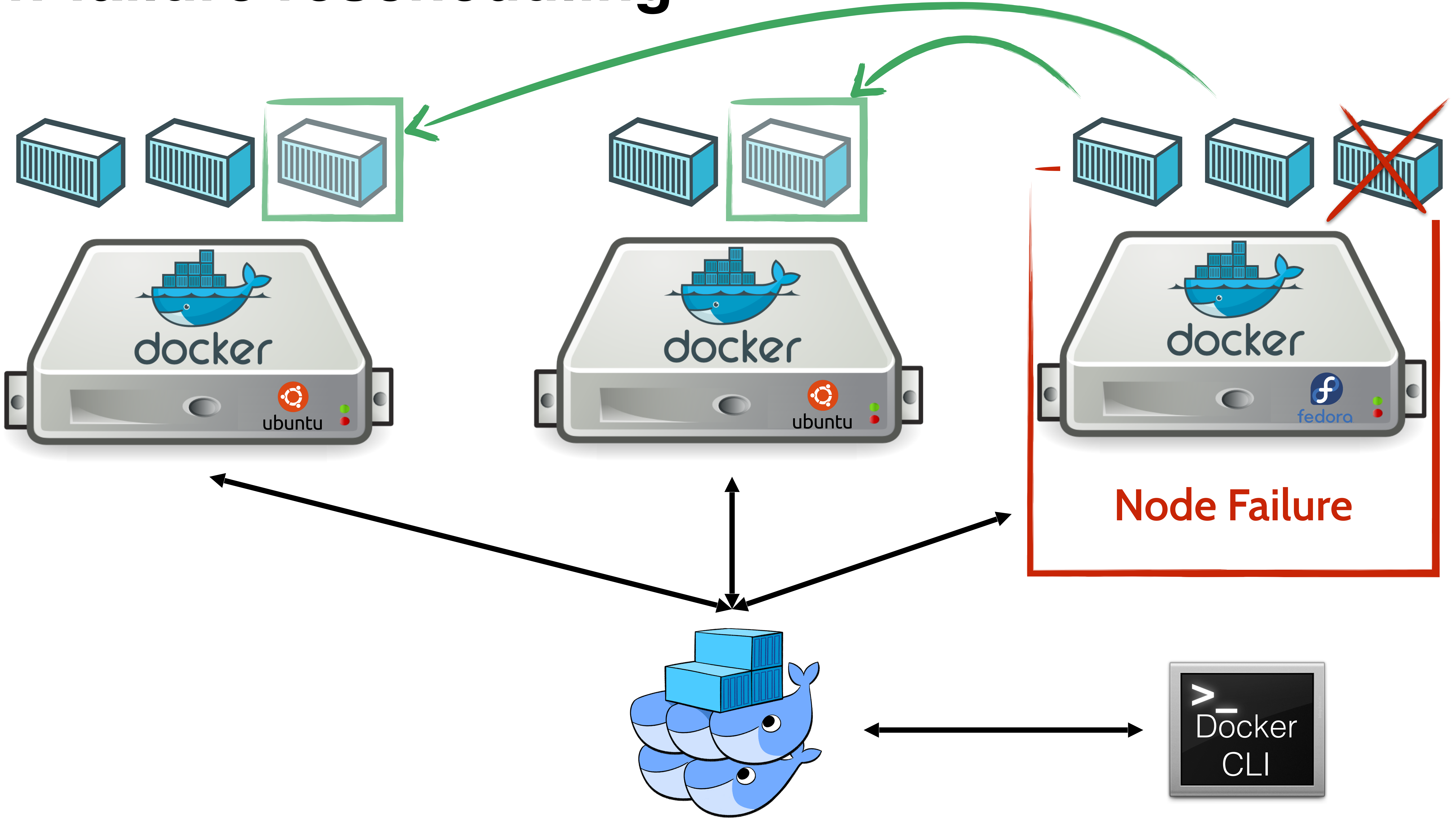
On-failure rescheduling



On-failure rescheduling



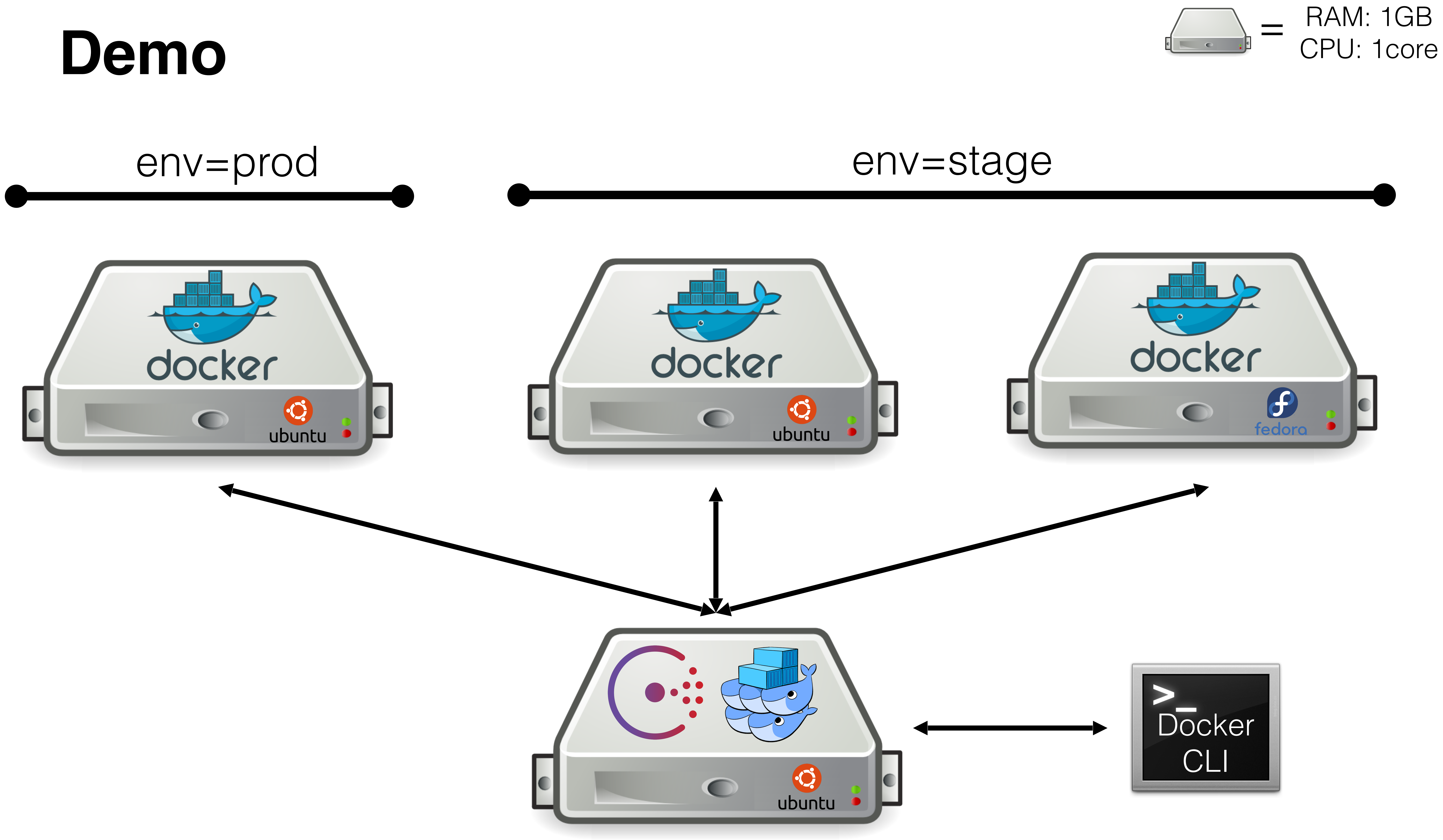
On-failure rescheduling



New events

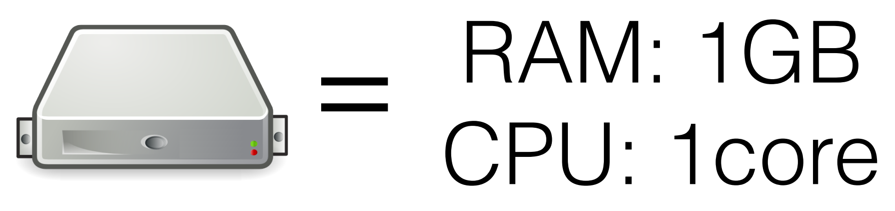
```
$ docker events
…
2016-02-09T12:02:01 container create XXX (com.docker.swarm.id=YYY, image=busybox, node.addr=10.0.0.1:2375, node.name=node-1)
…
2016-02-09T12:02:01 network connect ZZZ (node.name=node-1, type=bridge, container=XXX, name=bridge, node.addr=10.0.0.1:2375,)
…
2016-02-09T12:03:10 swarm engine_connect  (node.name=node-2, node.addr=10.0.0.2:2375)
…
```

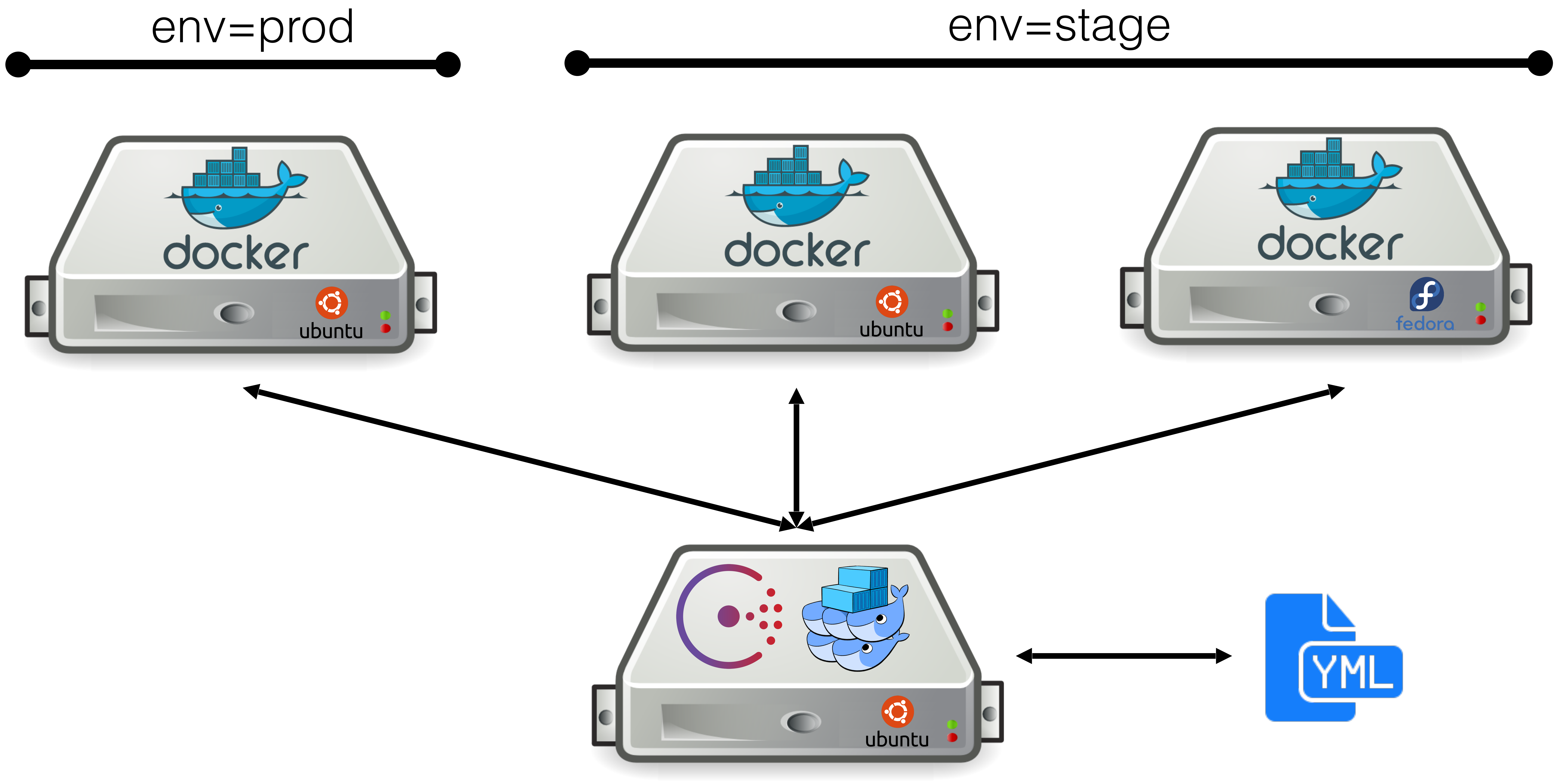
Demo

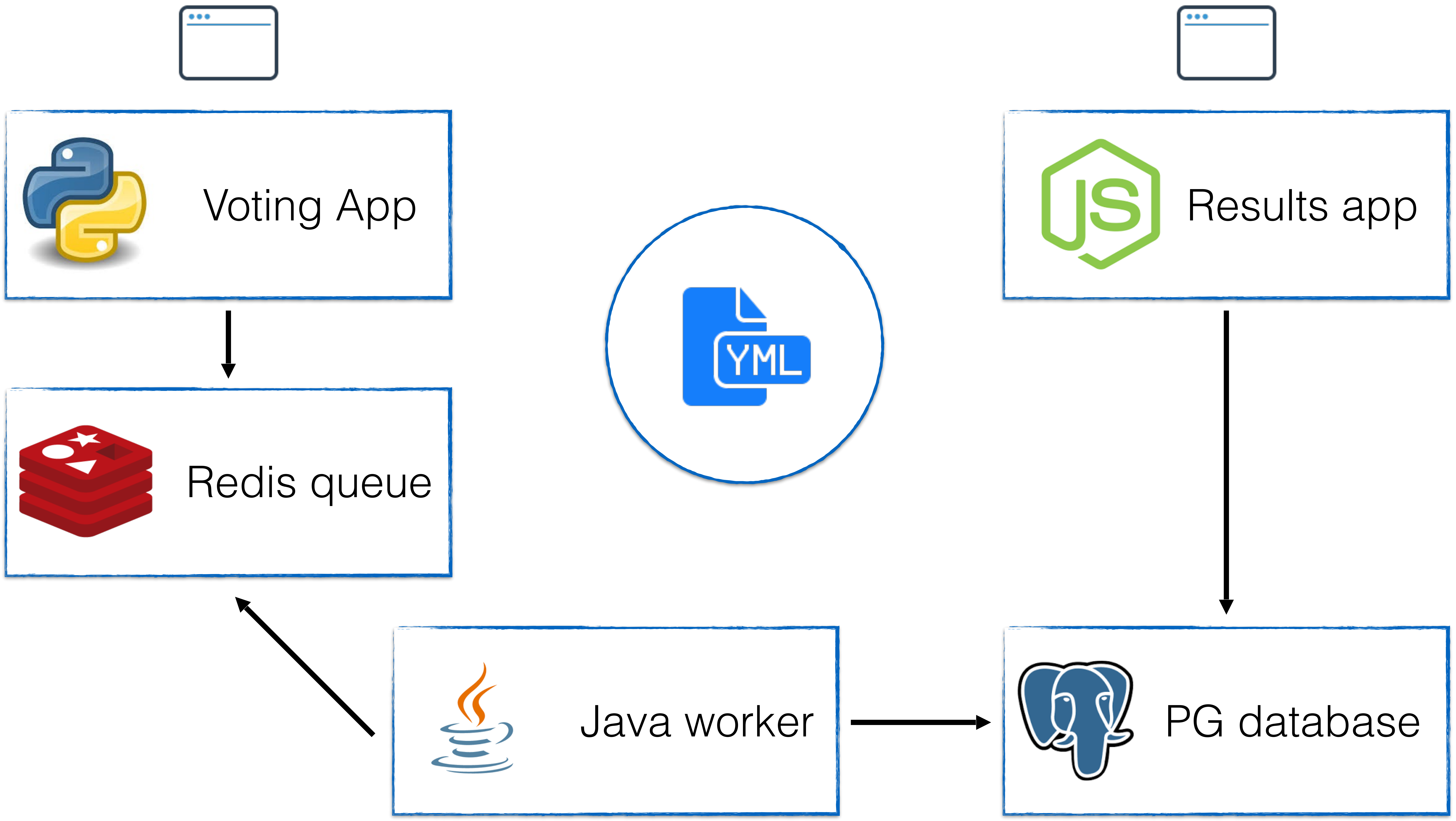


= RAM: 1GB
CPU: 1core
env=prod
env=stage
docker
ubuntu
docker
ubuntu
docker
fedora
ubuntu
>_
Docker
CLI

Demo


=
RAM: 1GB
CPU: 1core


env=prod
env=stage
docker
ubuntu
docker
ubuntu
docker
fedora
ubuntu
YML


Voting App
Redis queue
YML
Results app
Java worker
PG database

"vote" network
10.0.1.0/24
"result" network
10.0.2.0/24
Voting App
Redis queue
YML
Results app
Java worker
PG database

Thank You. Questions?

http://github.com/docker/swarm

#docker-swarm on freenode

@vieux