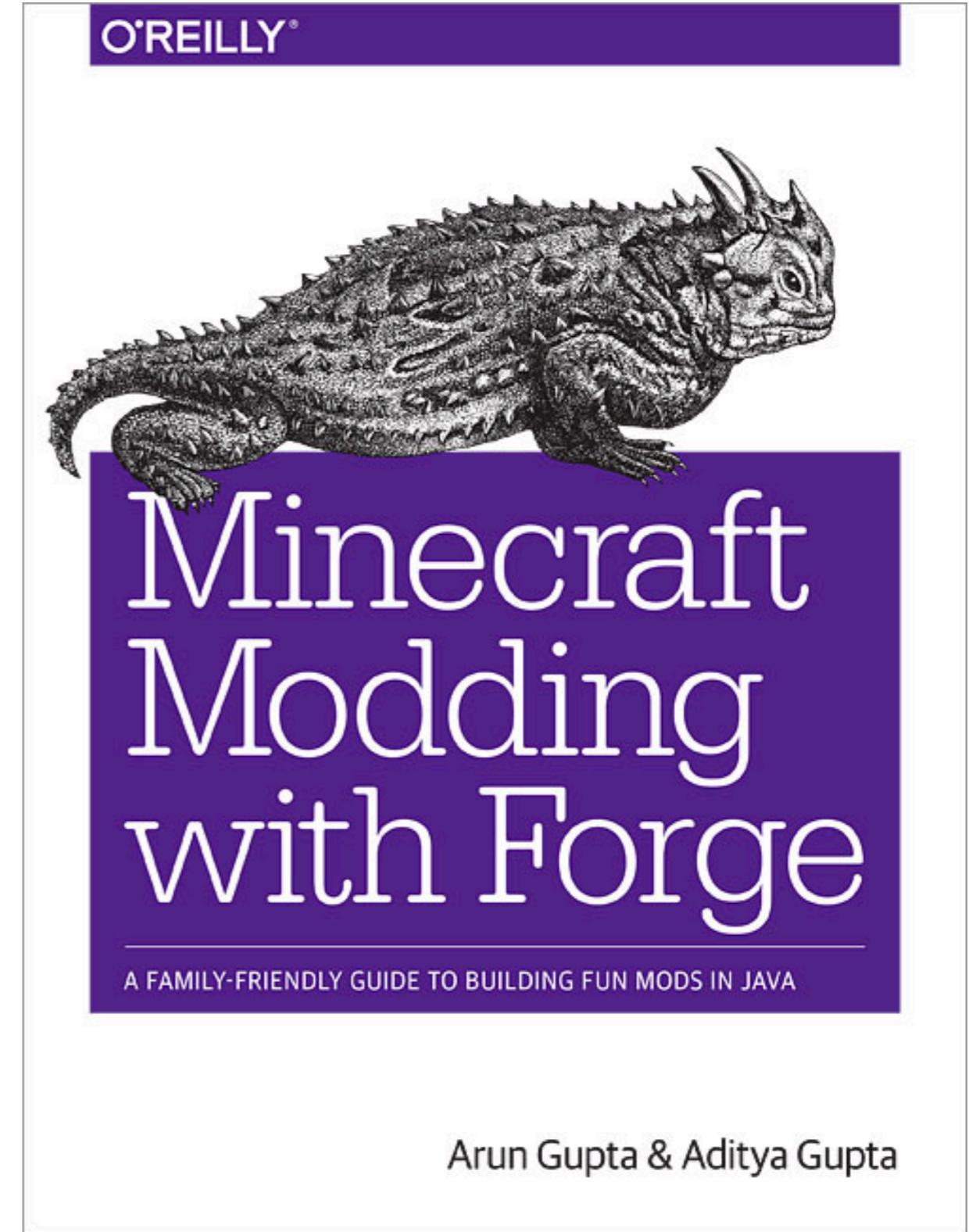


Docker Introduction



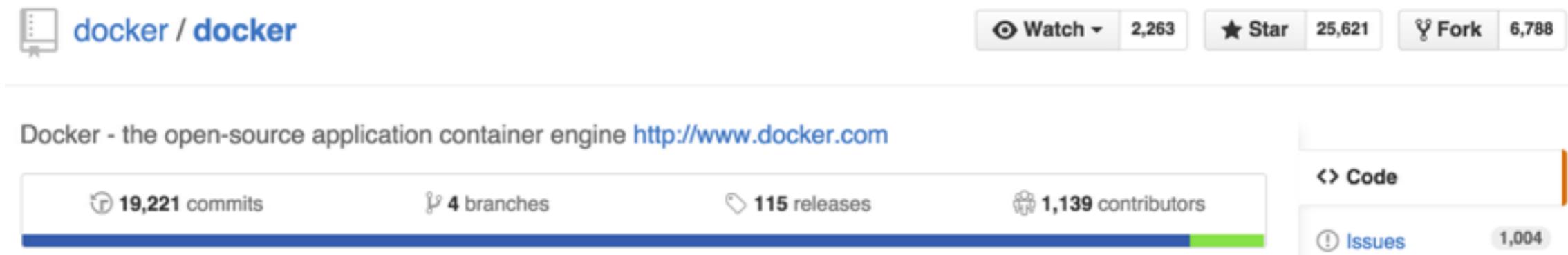
Arun Gupta

Vice President, Developer Advocacy
@arungupta, blog.arungupta.me
arun@couchbase.com



What is Docker?

- Open source project and company

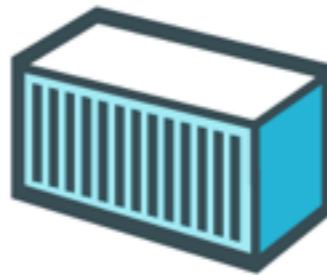


- Used to create containers for software applications
- Package Once Deploy Anywhere (PODA)



Build

Develop an app using Docker containers with
any language and any toolchain.



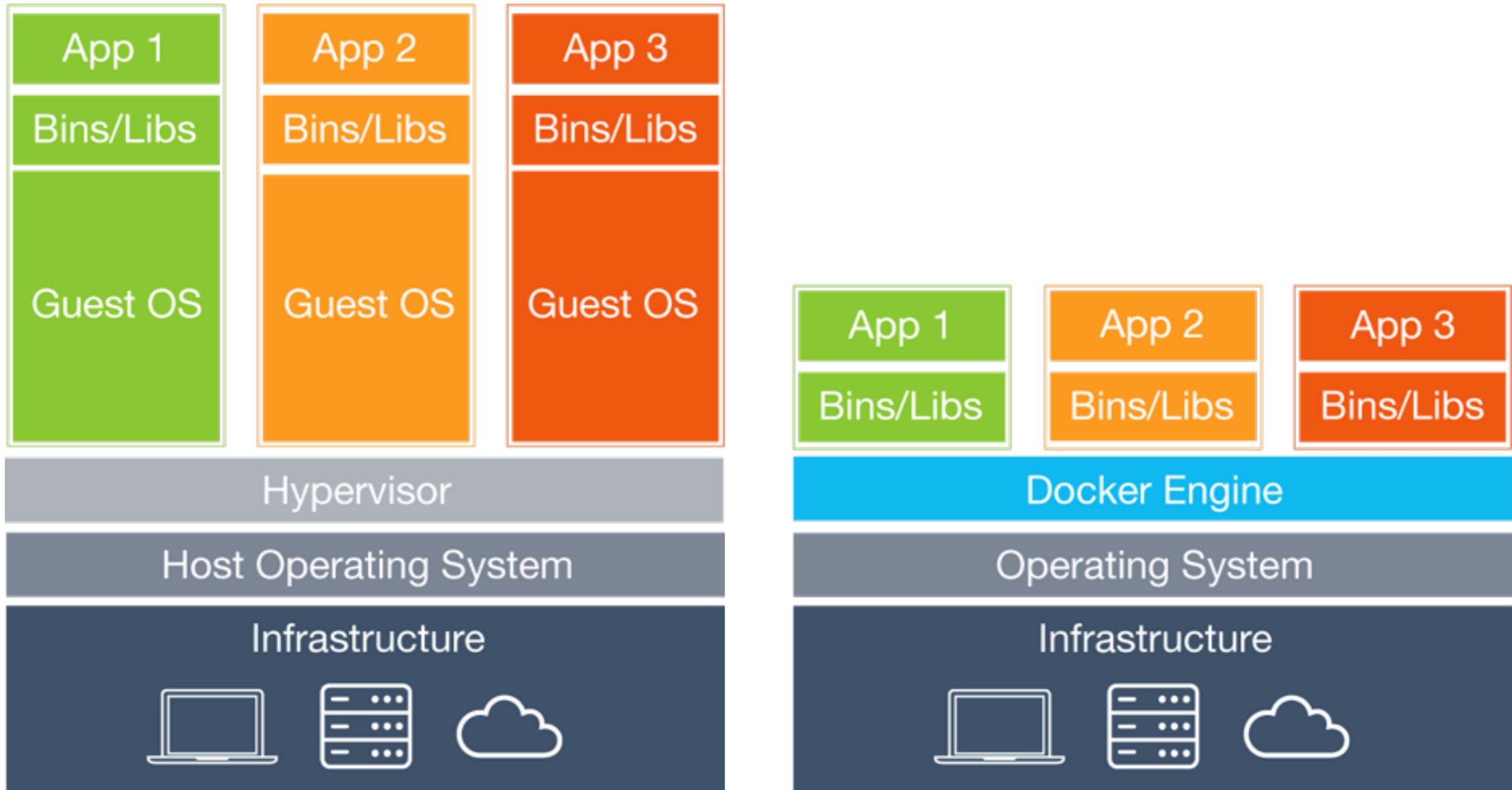
Ship

Ship the “Dockerized” app and dependencies
anywhere - to QA, teammates, or the cloud -
without breaking anything.



Run

Scale to 1000s of nodes, move between data
centers and clouds, update with zero
downtime and more.





Build

Develop an app using Docker containers with
any language and any toolchain.

- Image defined in text-based **Dockerfile**
- List of commands to build the image

```
FROM fedora:latest

CMD echo "Hello world"
```

```
FROM jboss/wildfly

RUN curl -L https://github.com/javaee-
samples/javaee7-hol/raw/master/solution/
movieplex7-1.0-SNAPSHOT.war -o /opt/jboss/
wildfly/standalone/deployments/
movieplex7-1.0-SNAPSHOT.war
```

Dockerfile reference

Usage

Format

 Environment replacement

 .dockerignore file

FROM

MAINTAINER

RUN

 Known issues (RUN)

CMD

LABEL

EXPOSE

ENV

ADD

COPY

ENTRYPOINT

 Exec form ENTRYPOINT example

 Shell form ENTRYPOINT example

VOLUME

USER

WORKDIR

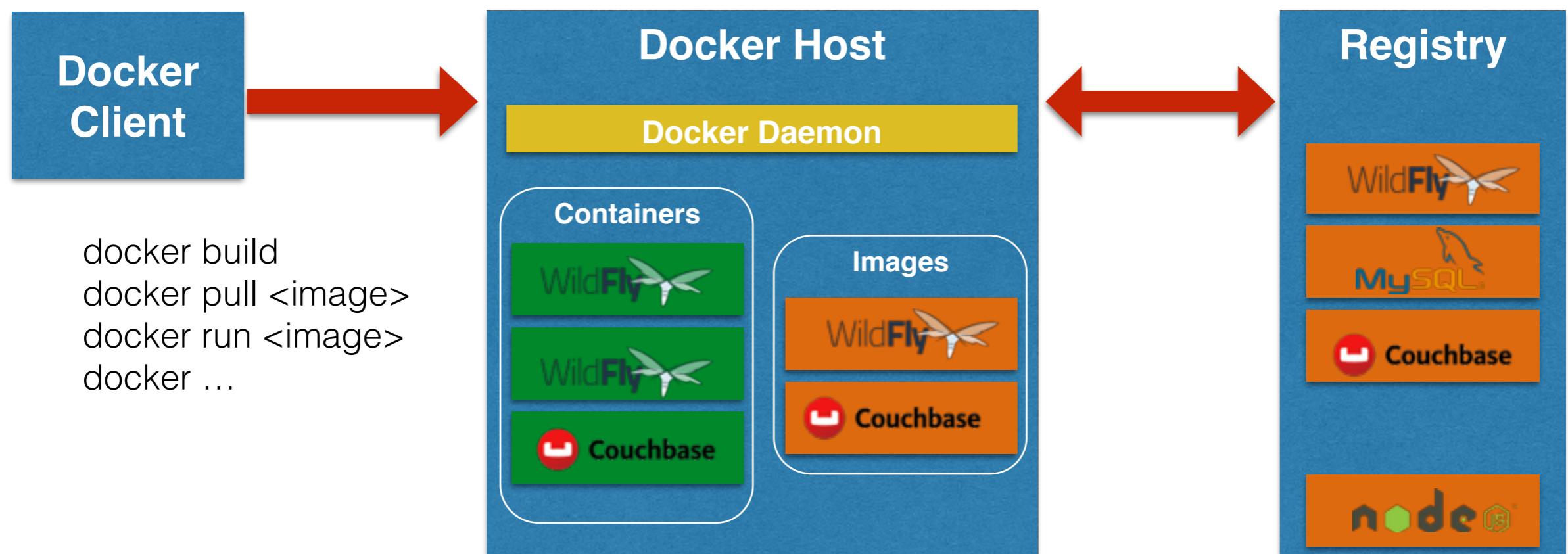
ARG

ONBUILD

STOP SIGNAL

Dockerfile examples

Docker Workflow



aufs

Image	jboss/wildfly
Image	jboss/base-jdk:8
Image	jboss/base
Base Image	centos:7
Bootfs/Kernel	



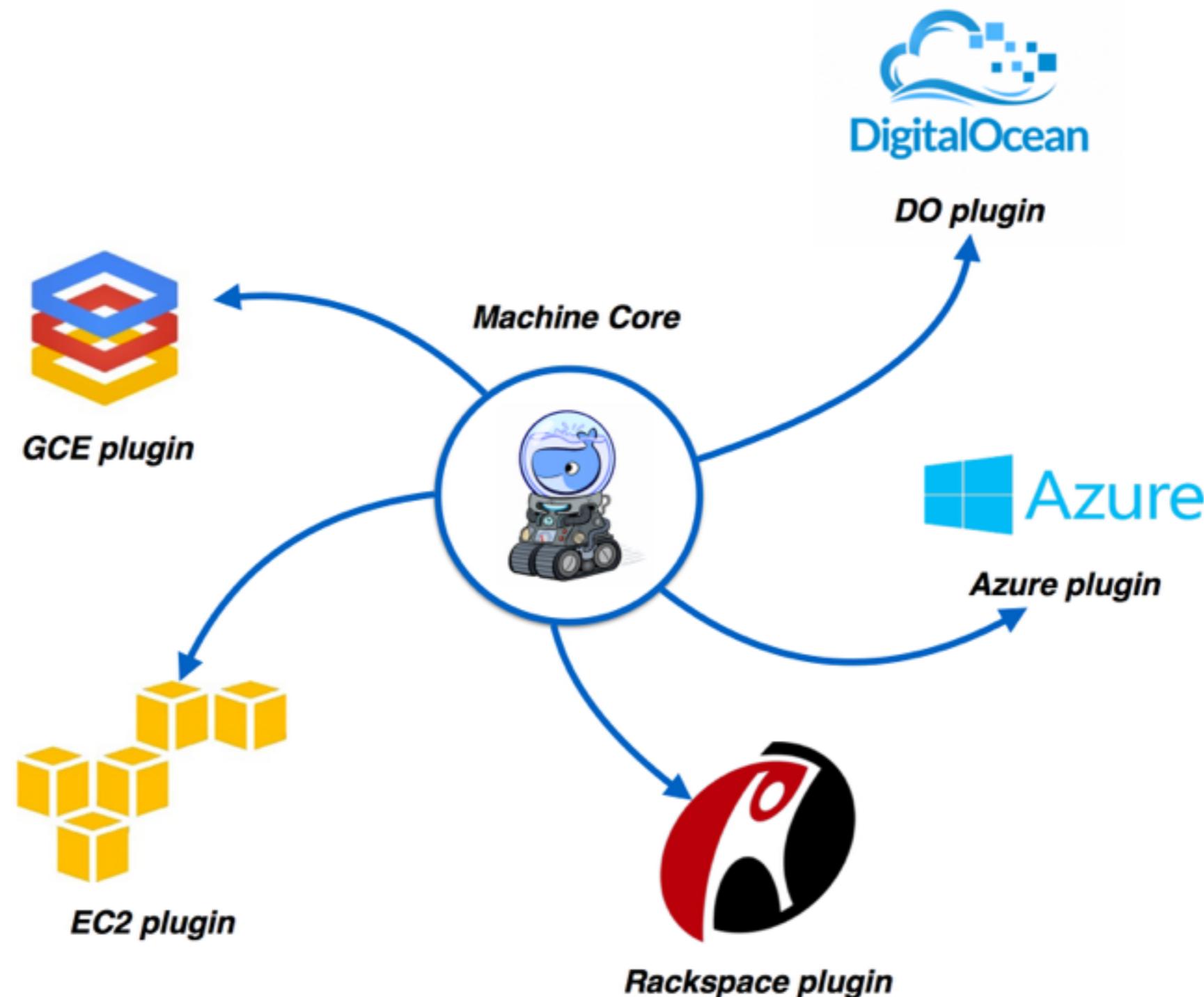
Docker Machine

- Create Docker Host on computer or cloud provider

```
docker-machine create --driver=virtualbox  
myhost
```

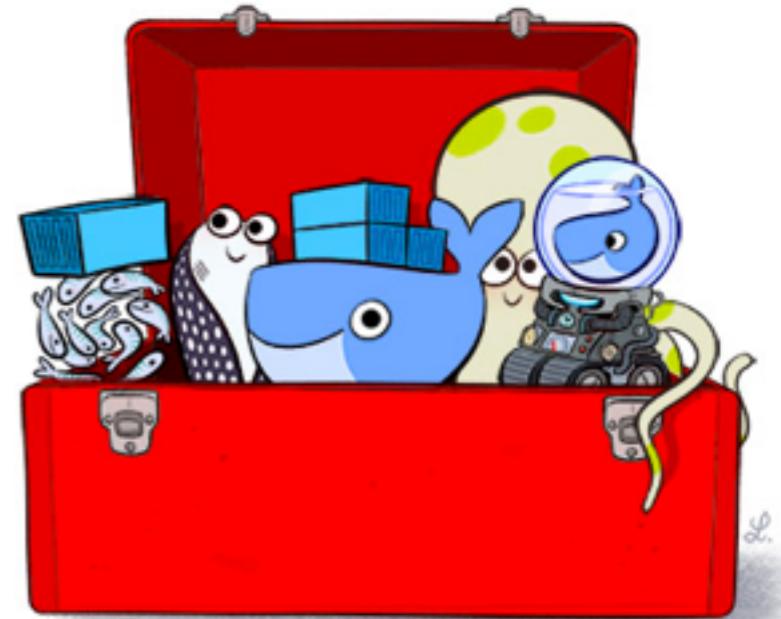
- Configure Docker client to talk to host
- Create and pull images
- Start, stop, restart containers
- Upgrade Docker
- Not recommended for production yet

Docker Machine Providers



Docker Toolbox

- Docker Client 1.9.0
- Docker Machine 0.5.0
- Docker Compose 1.5.0 (~~Mac only~~)
- Docker Kitematic 0.9.3
- Boot2Docker ISO 1.9.0
- Virtualbox 5.0.8





```
docker run  
--name mysqlDb  
-e MYSQL_USER=mysql  
-e MYSQL_PASSWORD=mysql  
-e MYSQL_DATABASE=sample  
-e MYSQL_ROOT_PASSWORD=supersecret  
-d  
mysql
```

```
docker run  
--name mywildfly  
--link mysqlDb:db  
-p 8080:8080  
-d  
arungupta/wildfly-mysql-javaee7
```

Docker Compose

- Defining and running multi-container applications
- Configuration defined in one or more files
 - `docker-compose.yml` (default)
 - `docker-compose.override.yml` (default)
 - Multiple files specified using `-f`
 - All paths relative to base configuration file
- Great for dev, staging, and CI

docker-compose.yml

```
mysqldb:  
  image: mysql  
  environment:  
    MYSQL_DATABASE: sample  
    MYSQL_USER: mysql  
    MYSQL_PASSWORD: mysql  
    MYSQL_ROOT_PASSWORD: supersecret  
  
mywildfly:  
  image: arungupta/wildfly-mysql-javaee7  
  links:  
    - mysqldb:db
```

Links Environment Variables

```
data-source add
--name=mysqlDS
--driver-name=mysql
--jndi-name=java:jboss/datasources/
ExampleMySQLDS
--connection-url=jdbc:mysql://
${DB_PORT_3306_TCP_ADDR}:
${DB_PORT_3306_TCP_PORT}/sample?
useUnicode=true&characterEncoding=
UTF-8
--user-name=mysql
--password=mysql
--use-ccm=false
--max-pool-size=25
--blocking-timeout-wait-millis=5000
--enabled=true
```

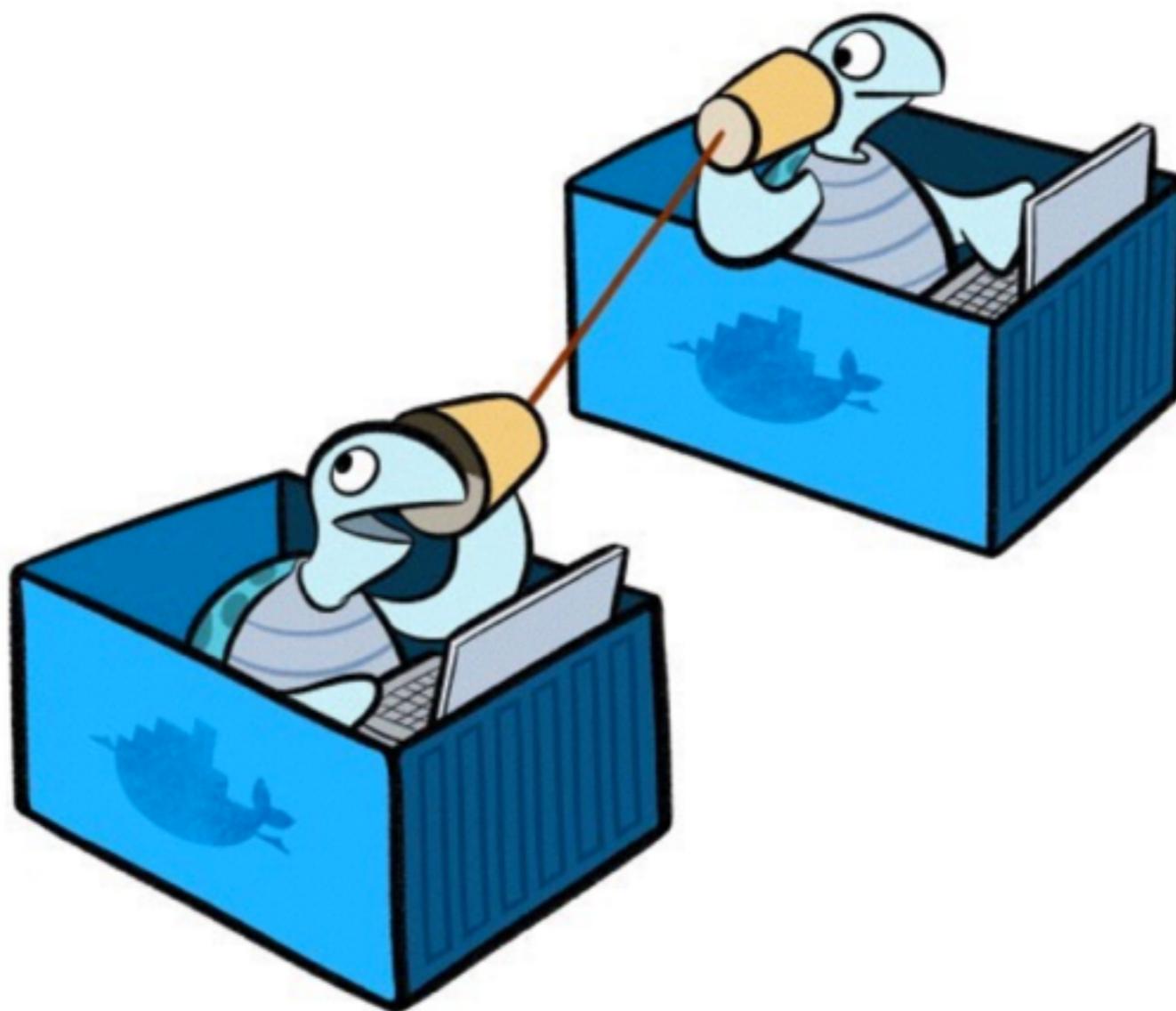
```
<connection url>jdbc:mysql://
172.17.0.2:3306/sample?
useUnicode=true&characterEncod
ing=UTF-8</connection>
```

Couchbase Cluster using Docker Compose

```
couchbase1:  
  image: couchbase/server  
  volumes:  
    - ~/couchbase/node1:/opt/couchbase/var  
couchbase2:  
  image: couchbase/server  
  volumes:  
    - ~/couchbase/node2:/opt/couchbase/var  
couchbase3:  
  image: couchbase/server  
  volumes:  
    - ~/couchbase/node3:/opt/couchbase/var  
  ports:  
    - 8091:8091  
    - 8092:8092  
    - 8093:8093  
    - 11210:11210
```



Multi Host Networking



Multi-host Networking

- Create virtual networks and attach containers
 - Software defined networking
- **Bridge** network spans single host
- **Overlay** network spans multiple hosts
- Works with Swarm and Compose
 - Experimental in Compose
- Pluggable: Calico, Cisco, Weave, . . .

Networking vs Links

- Connect containers to each other across different physical or virtual hosts
- Containers can be easily stopped, started and restarted w/o disrupting connection to other containers
- Can be created in any order

Links

```
mysqldb:  
  image: mysql  
  environment:  
    MYSQL_DATABASE: sample  
    MYSQL_USER: mysql  
    MYSQL_PASSWORD: mysql  
    MYSQL_ROOT_PASSWORD: supersecret  
mywildfly:  
  image: arungupta/wildfly-mysql-javaee7  
  links:  
    - mysqldb:db
```

Networking

```
mysqldb:  
  container_name: "db"  
  image: mysql:latest  
  environment:  
    MYSQL_DATABASE: sample  
    MYSQL_USER: mysql  
    MYSQL_PASSWORD: mysql  
    MYSQL_ROOT_PASSWORD: supersecret  
mywildfly:  
  image: arungupta/wildfly-mysql-javaee7  
  environment:  
    - MYSQL_URI=db:3306  
  ports:  
    - 8080:8080
```

Networking Environment Variables

```
data-source add  
--name=mysqlDS  
--driver-name=mysql  
--jndi-name=java:jboss/datasources/  
ExampleMySQLDS  
--connection-url=jdbc:mysql://  
$MYSQL_URI/sample?  
useUnicode=true&characterEncoding=UTF-8  
--user-name=mysql  
--password=mysql  
--use-ccm=false  
--max-pool-size=25  
--blocking-timeout-wait-millis=5000  
--enabled=true
```

```
<connection-url>jdbc:mysql://db:3306/  
sample?  
useUnicode=true&characterEncoding=UTF-8</connection-url>
```

Overriding Services in Docker Compose

```
mywildfly:  
  image: jboss/wildfly  
  ports:  
    - 8080:8080
```

docker-compose.yml

```
mywildfly:  
  ports:  
    - 9080:8080
```

docker-compose.override.yml

`docker-compose up -d`

```
mycouchbase:  
  container_name: "db-dev"  
  image: couchbase/server  
  ports:  
    - . . .
```

```
mywildfly:  
  image: arungupta/wildfly-couchbase-javaee7  
  environment:  
    - COUCHBASE_URL=db-dev:3093  
  ports:  
    - 8080:8080
```

docker-compose.yml

docker-compose up -d

```
mywildfly:  
  environment:  
    - COUCHBASE_URL=db-prod:3093  
  ports:  
    - 80:8080  
mycouchbase:  
  container_name: "db-prod"
```

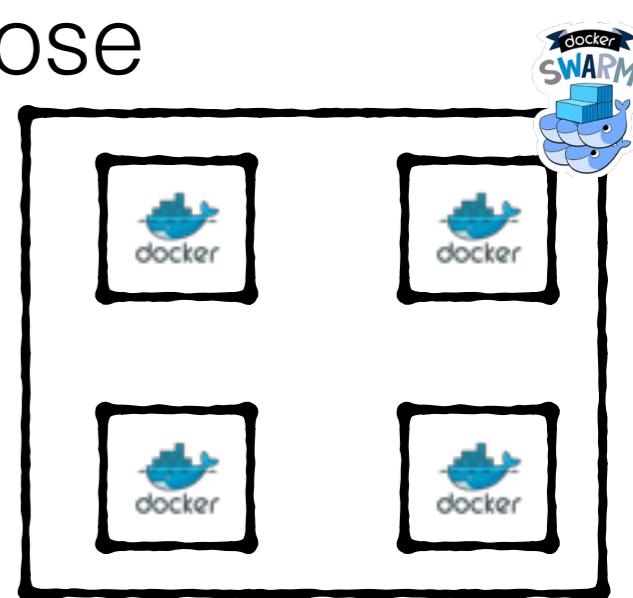
docker-compose-prod.yml

docker-compose up
-f docker-compose.yml
-f docker-compose-prod.yml
-d

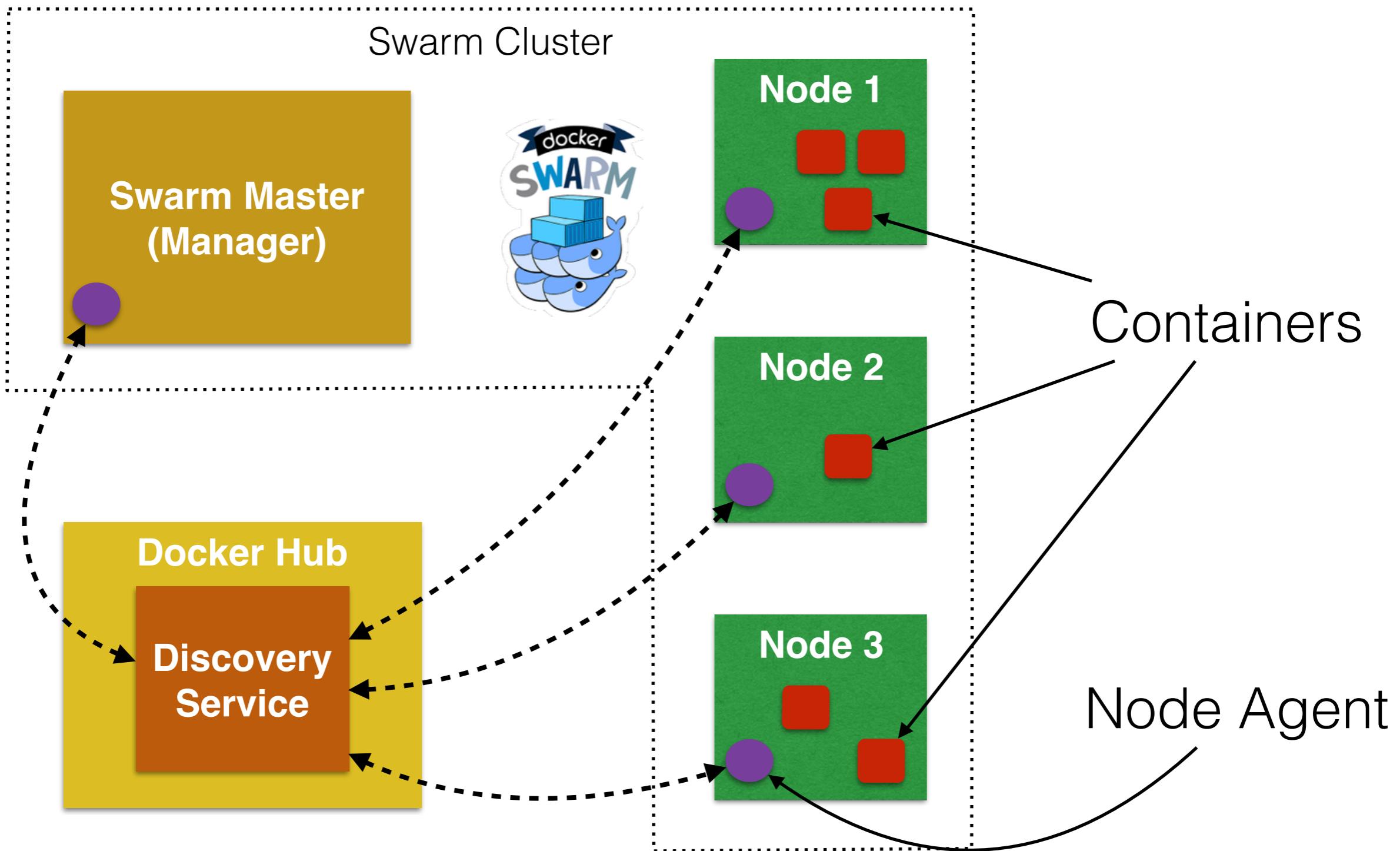


Docker Swarm

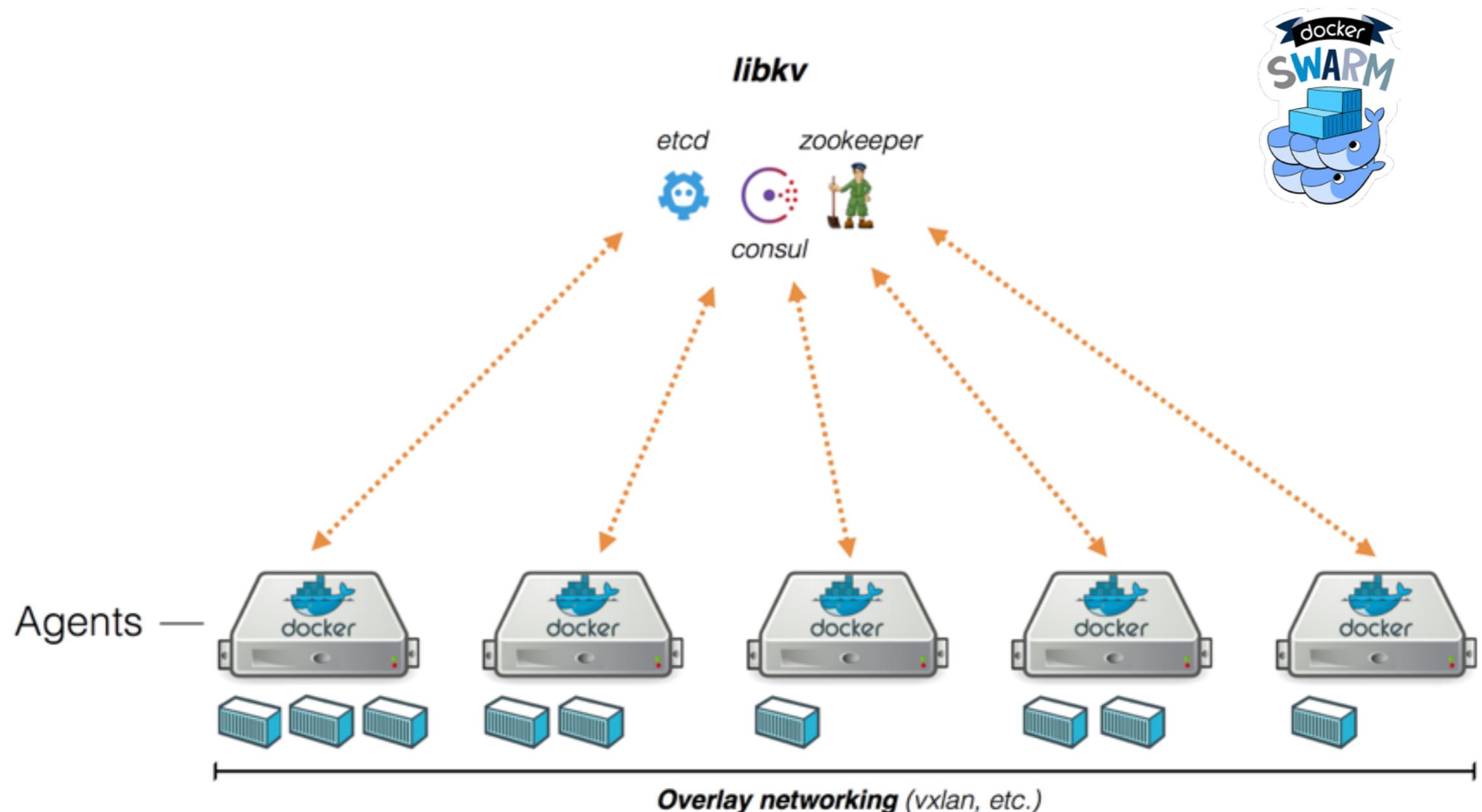
- Native clustering for Docker
- Provides a unified interface to a pool of Docker hosts
- Fully integrated with Machine and Compose
- Serves the standard Docker API
- 1.0.0 - Ready for production



Stress tested on 1000 EC2 nodes, ~30k containers



Multiple Discovery Backends



Node Discovery

```
docker-machine create  
--engine-opt --cluster-advertise=<value>  
--engine-opt --cluster-store=<value>  
--engine-opt -cluster-store-opt=<value>  
-d=virtualbox  
myMachine
```



High Availability

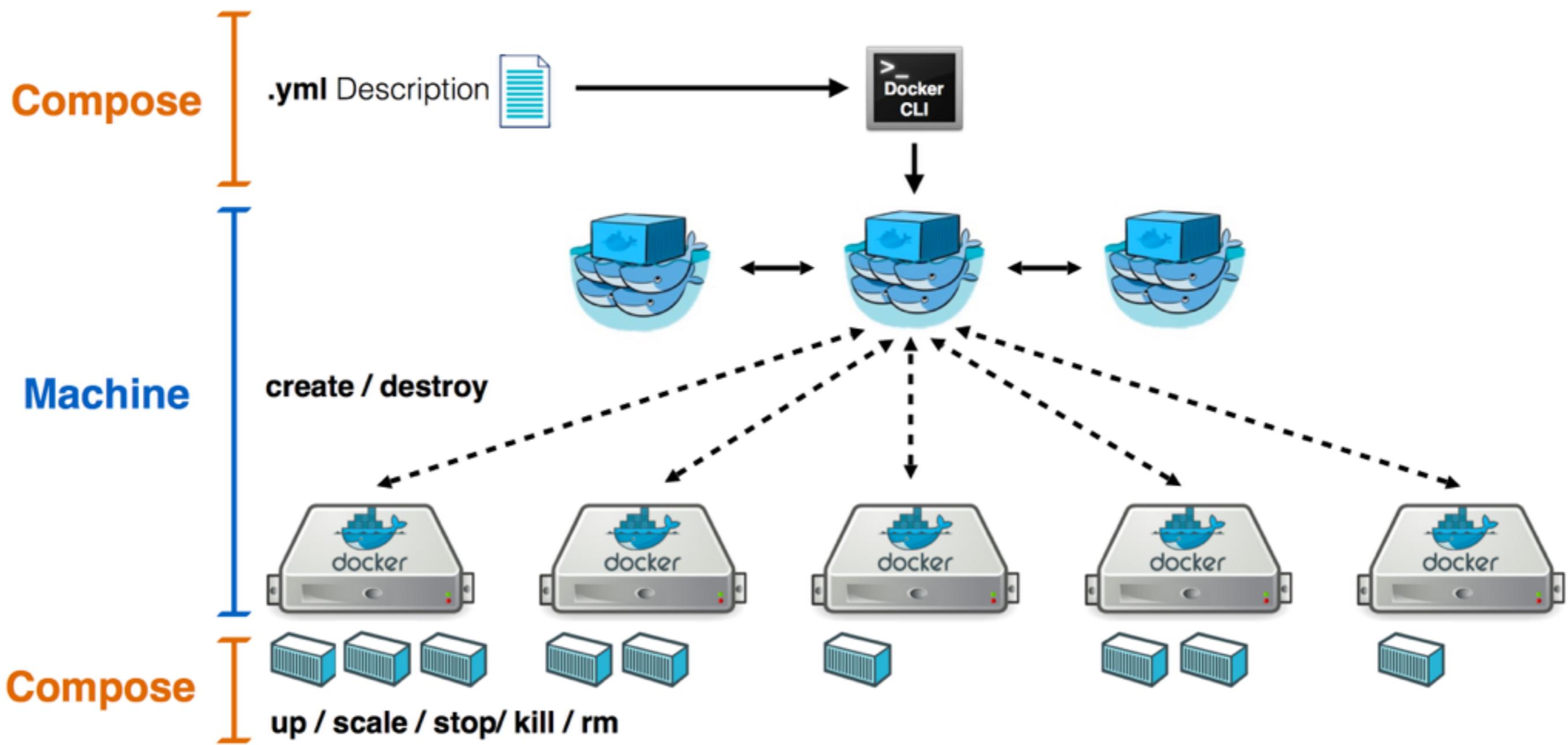
- Handle the failover of a manager instance
- **Primary** manager, multiple **replica** instances
- Requests to replica are proxied to primary
- `docker run swarm manage`
 - `--replication`: Enable Swarm manager replication
 - `--replication-ttl "30s"`: Leader lock release time on failure
 - `--advertise`, `--addr`: Address of the swarm manager joining the cluster



Scheduling Backends

- Based on CPU (**-c**), RAM (**-m**), number of containers
- **spread (default)**: node with least number of running containers
- **binpack**: node with most number of running containers
- **random**: mostly for debugging
- Specified using **–strategy**
- API for pluggable backends (e.g. Mesos) coming

Machine + Swarm + Compose



Persistent Storage

- Data volumes - used to persist data independent of container's lifecycle
- Multiple plugins: Flocker, Ceph, . . .

```
docker volume --help
```

Usage: docker volume [OPTIONS] [COMMAND]

Manage Docker volumes

Commands:

create
inspect
ls
rm

Create a volume
Return low-level information on a volume
List volumes
Remove a volume

Persistent Storage

Create a volume

```
docker volume create --name=data data
```

Run a container with the volume

```
docker run -it -v data:/opt/  
couchbase/var couchbase/server
```

docker network

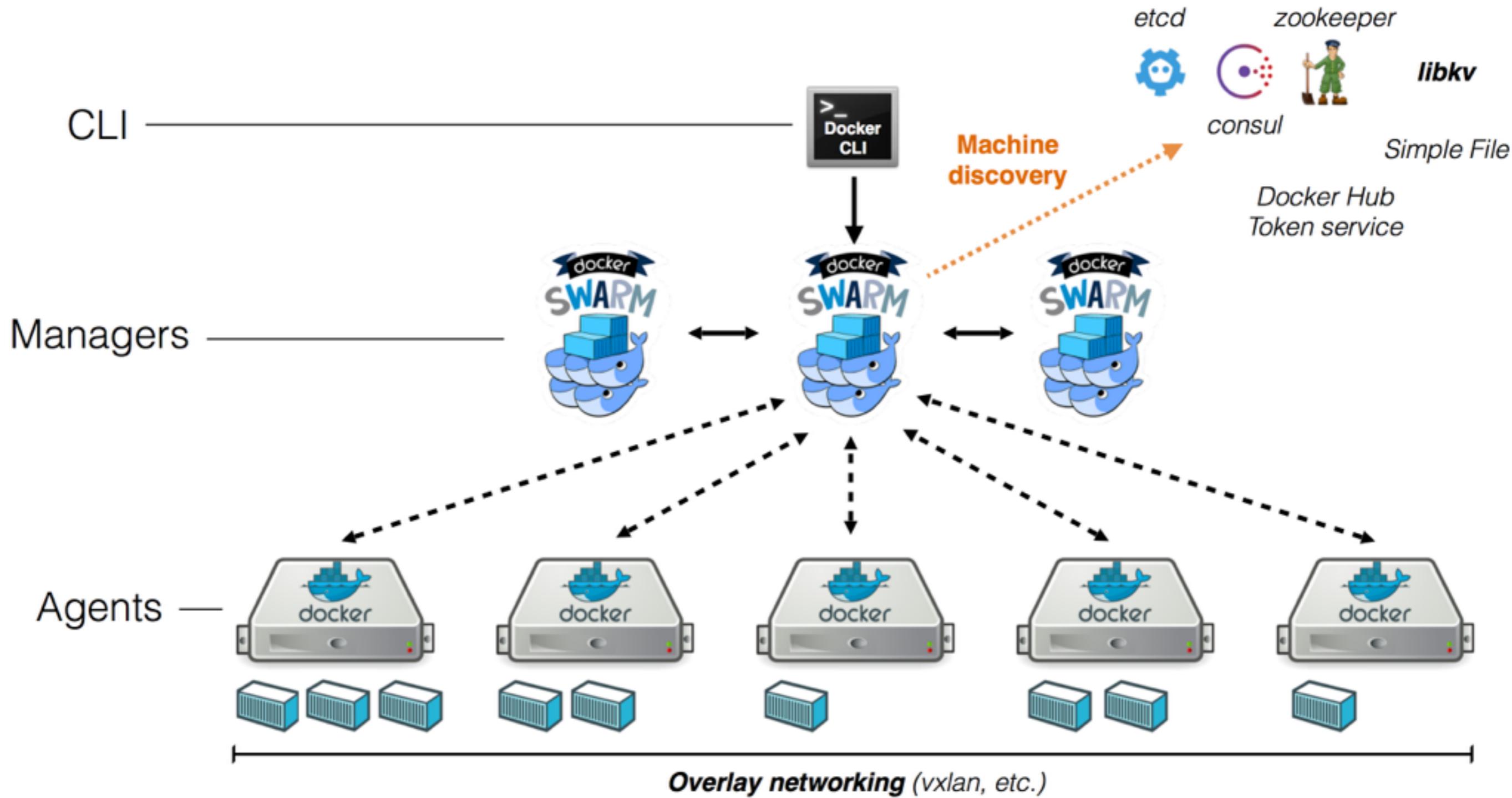
```
docker network --help
```

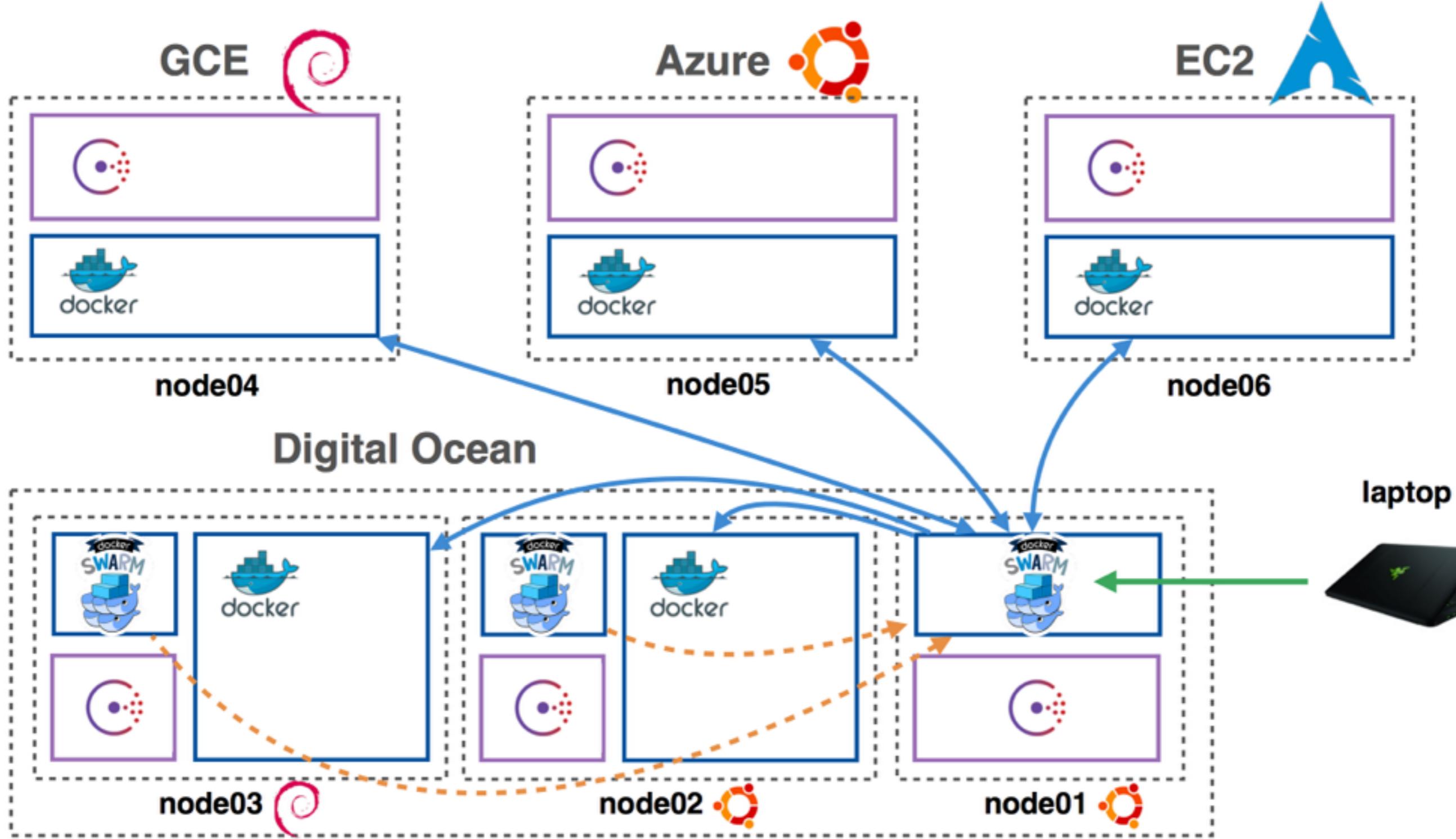
Usage: docker network [OPTIONS] COMMAND [OPTIONS]

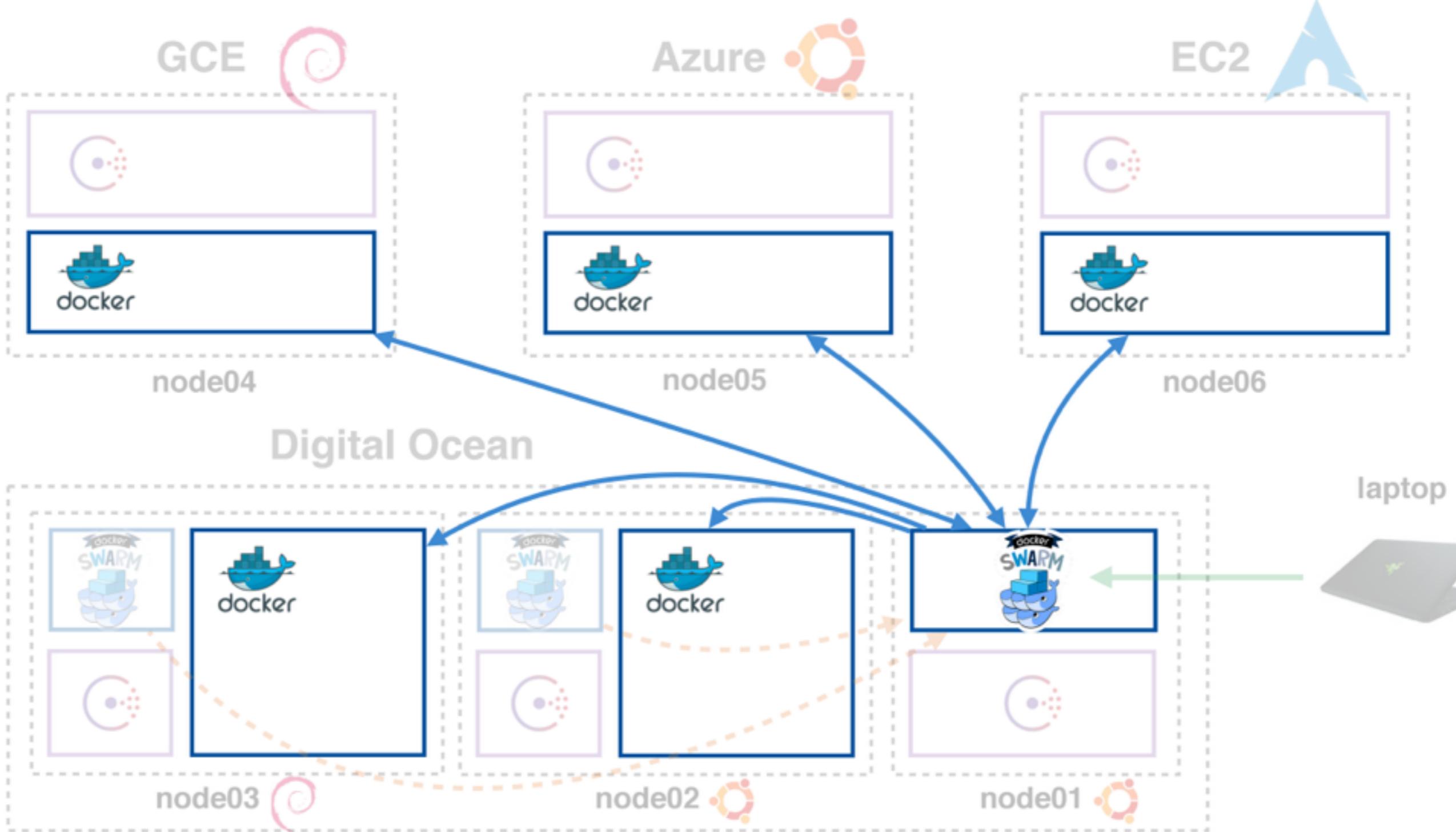
Commands:

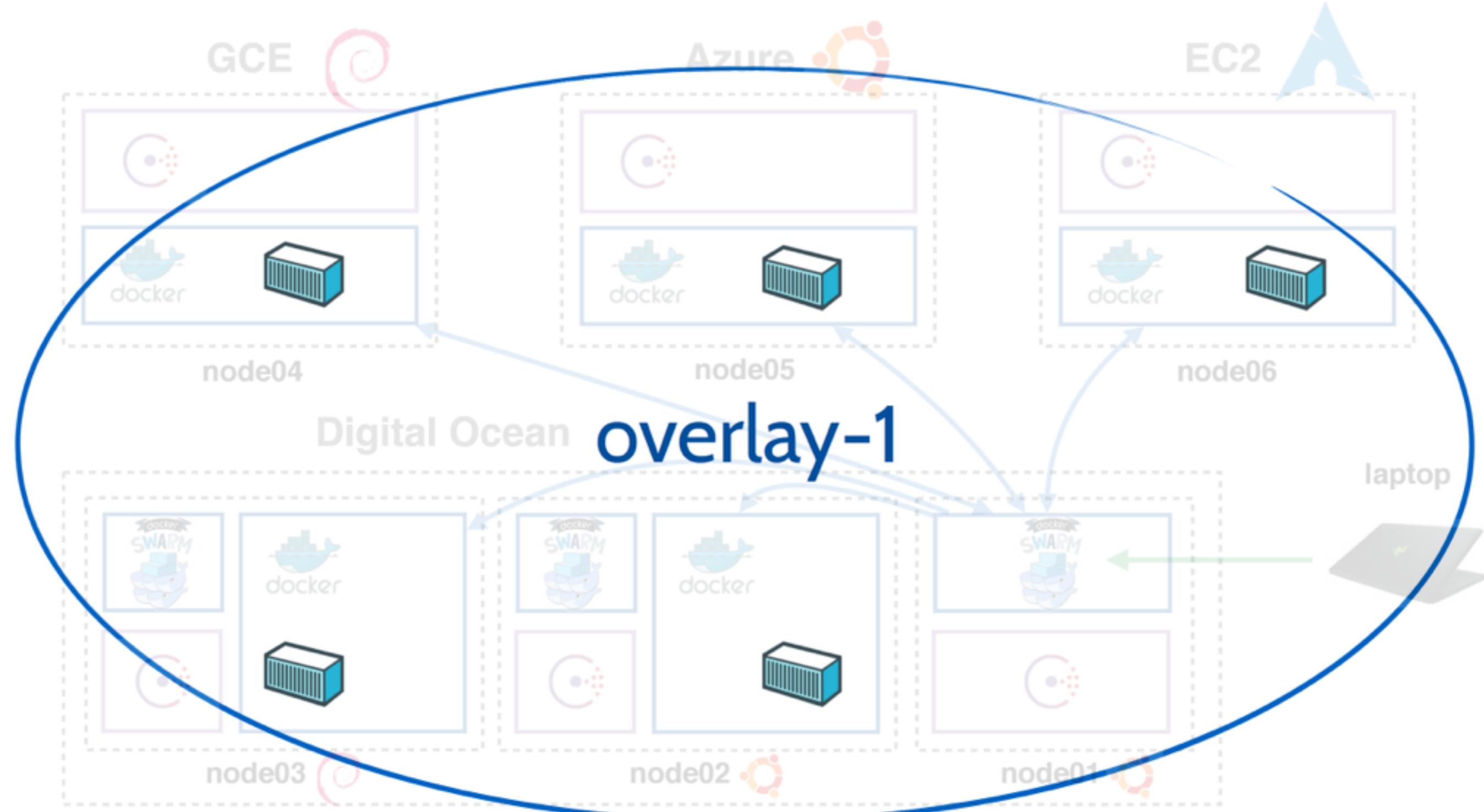
- disconnect**
- inspect**
- ls**
- rm**
- create**
- connect**

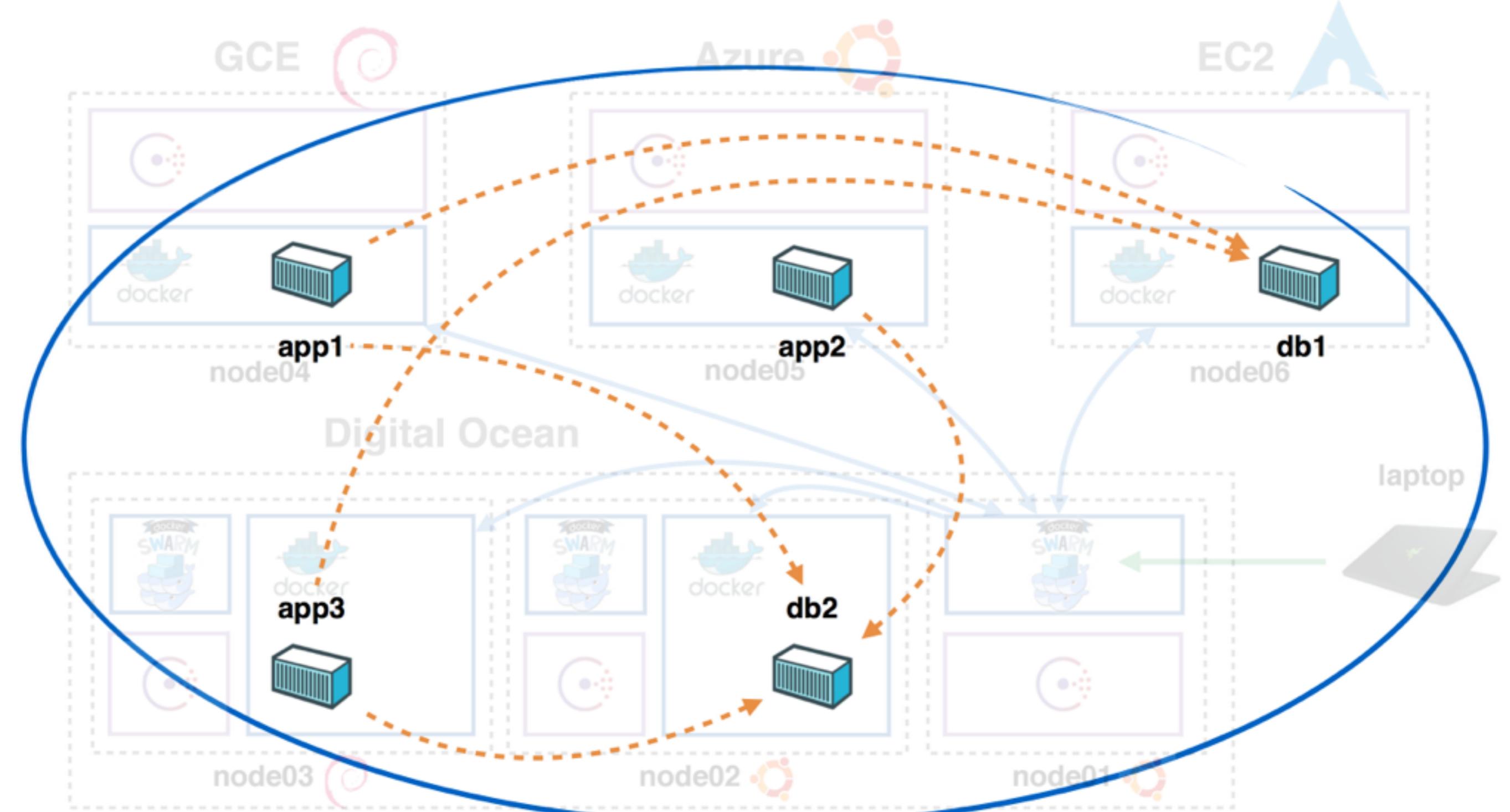
- Disconnect container from a network**
- Display detailed network information**
- List all networks**
- Remove a network**
- Create a network**
- Connect container to a network**











References

- github.com/javaee-samples/docker-java

