CLOUD COMPUTING APPLICATIONS

Docker Swarm

Prof. Reza Farivar

Container Orchestration

- Many application consist of multiple components, that need to be distributed on more than one machine
- Using containers, we can have each component running in its own container
- Thousands of pre-built components available on public registries

Docker vs. Swarm

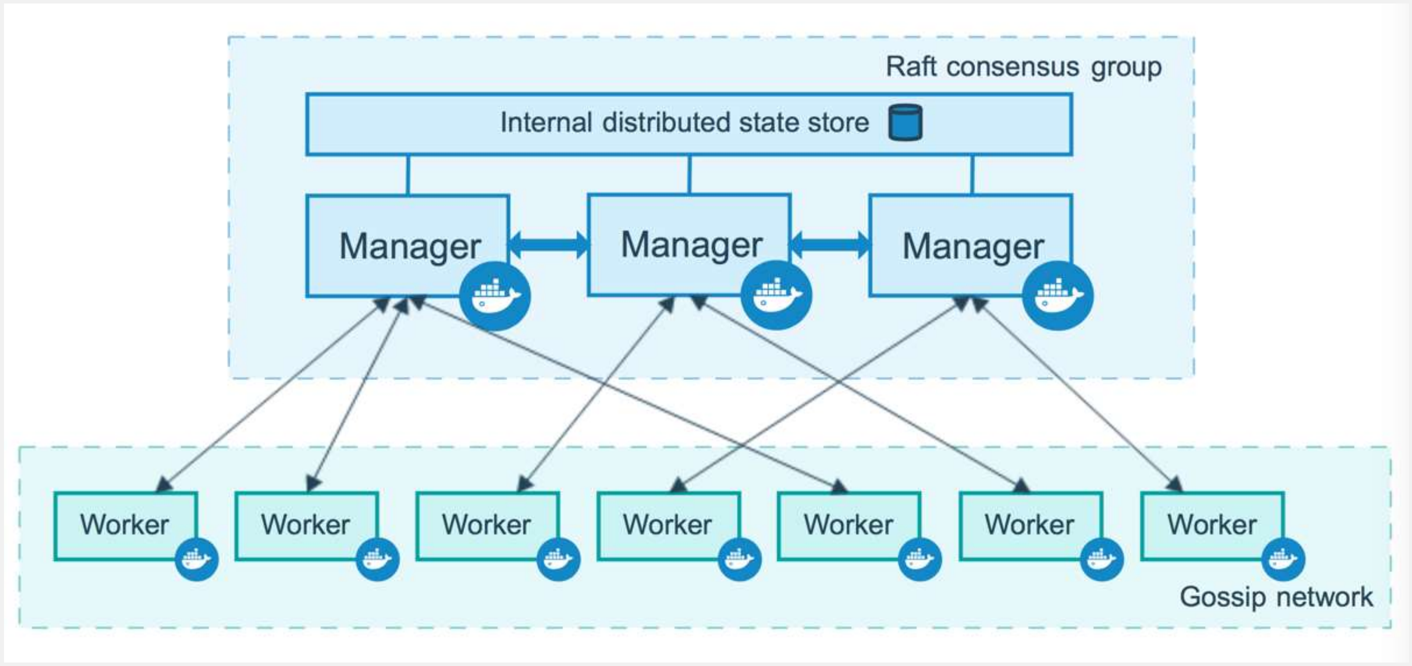
```
$ docker container create \
 --name my-nginx \
 --publish 80:80 \

 --network nginx-net \
 nginx
```

```
$ docker service create \
 --name my-nginx \
 --publish target=80,published=80 \
 --replicas=5 \
 --network nginx-net \
 nginx
```

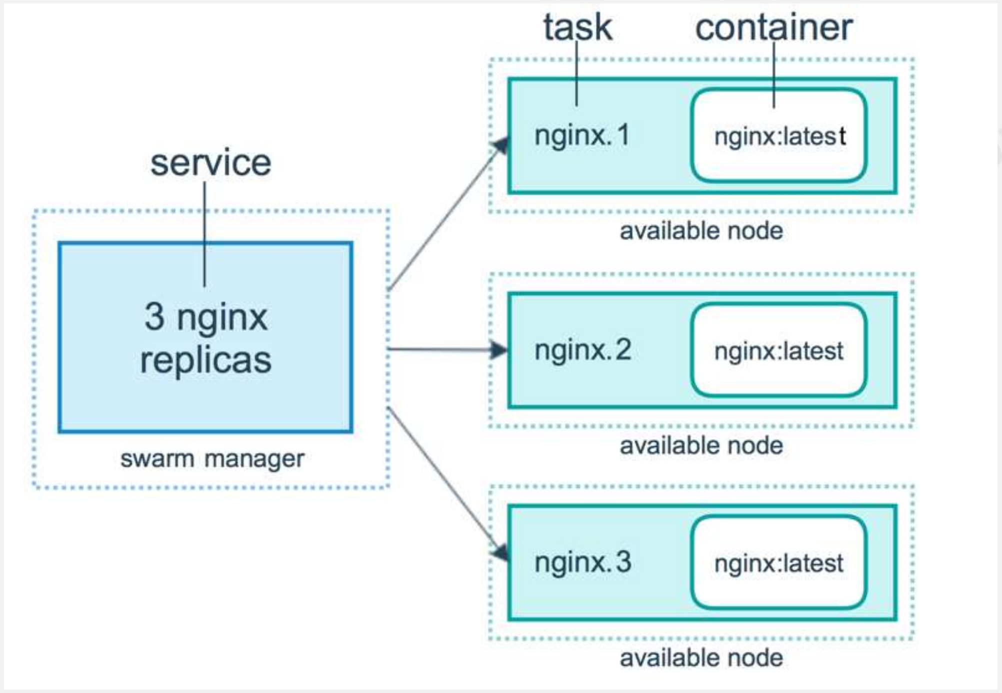
Swarm Services

- Swarm services use a *declarative* model, which means that you define the desired state of the service, and rely upon Docker to maintain this state.
- State
    - image name and tag
    - how many containers (tasks) in the service
    - ports exposed to clients outside the swarm

Nodes on Docker Swarm



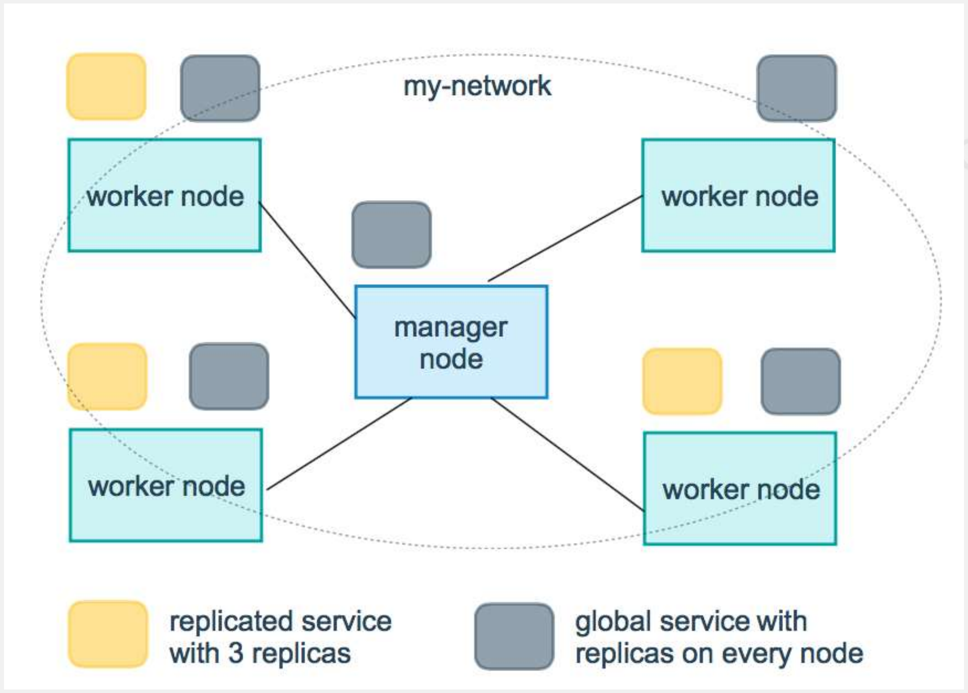
Services on Docker Swarm



Swarm Task States

- Docker lets you create services, which can start tasks.
- A service is a description of a desired state, and a task does the work.
- Work is scheduled on swarm nodes in this sequence
  1. Create a service by using `docker service create`.
  2. The request goes to a Docker manager node.
  3. The Docker manager node schedules the service to run on particular nodes.
  4. Each service can start multiple tasks.
  5. Each task has a life cycle, with states like `NEW`, `PENDING`, and `COMPLETE`

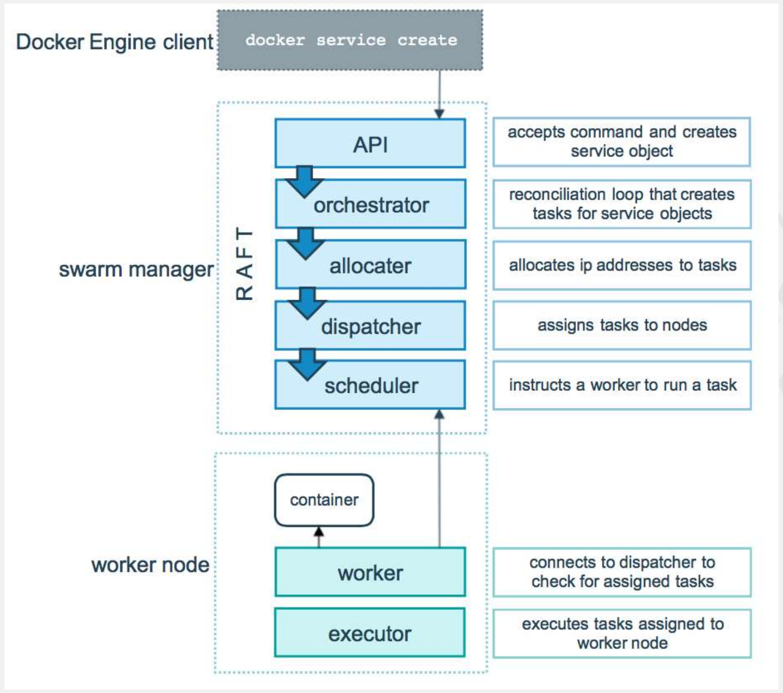
Replicated and Global Tasks


my-network
worker node
worker node
manager node
worker node
worker node
replicated service with 3 replicas
global service with replicas on every node

Example: Run a three-task Nginx service on 10-node swarm

- `$ docker service create --name my_web --replicas 3 --publish published=8080,target=80 nginx`
- Three tasks run on up to three nodes.
- You don’t need to know which nodes are running the tasks; connecting to port 8080 on **any** of the 10 nodes connects you to one of the three nginx tasks.
  - Routing mesh

Tasks and Scheduling



Docker Engine client
docker service create
swarm manager
RAFT
API
accepts command and creates service object
orchestrator
reconciliation loop that creates tasks for service objects
allocater
allocates ip addresses to tasks
dispatcher
assigns tasks to nodes
scheduler
instructs a worker to run a task
worker node
container
worker
connects to dispatcher to check for assigned tasks
executor
executes tasks assigned to worker node