Container Orchestration using Swarm

Presented by :
Suyash Tandon
Vinen Furtado

What is container orchestration ?

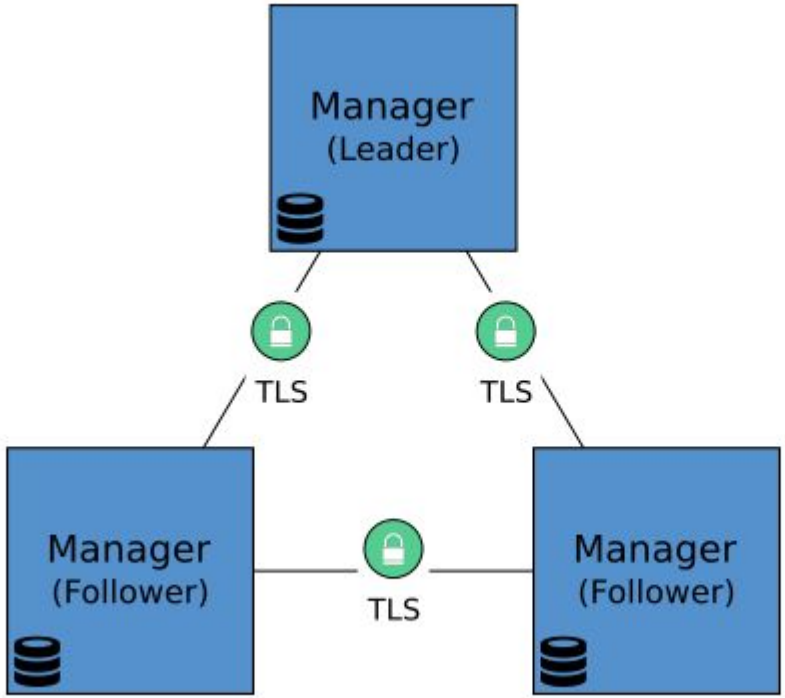
- Organizing components of applications at network level
- Container Orchestration allows user to control the containers, group them into clusters and coordinate the processes.
- So basically it’s a way to manage containers.

Why is container orchestration needed?

- Containers on a single docker host for development purposes is a easy to use and a straightforward experience.
- But, what about when we need to deploy our applications for usage? In a production environment?
- We use Container orchestration.
- Why?
- Difficult to have the ability to scale our services on a single host.

About Swarm architecture

- Docker Swarm is a clustering and scheduling tool for Docker containers
- Used to manage a cluster of Docker nodes as a single virtual system
- Swarm consists of manager node(s), worker node(s) and services
- A worker node will only get instructions but the manager node can give instructions


Manager
(Leader)
TLS
TLS
Manager
(Follower)
TLS
Manager
(Follower)

3 Parts of our Presentation

- *How Services can be scheduled on Docker Swarm?*
- *How services can be consumed on Docker Swarm?*
- *How updating services works in Docker Swarm? With a small Demo*

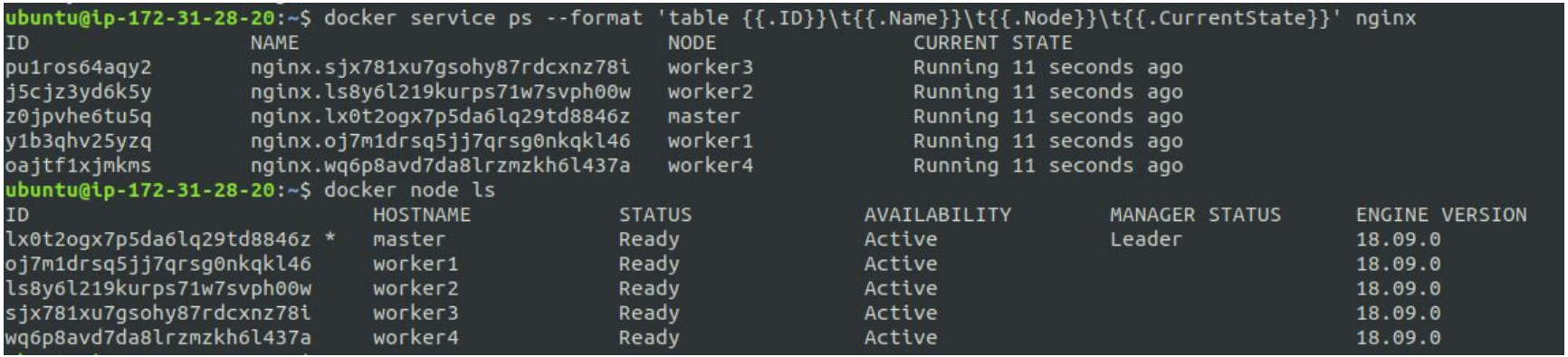
Services in docker swarm can run in 2 modes:

We have used the official NGINX docker image to create the nginx service.

- Default mode
  - have a specific number of replicated services(here 5)
  - Distributed among all the nodes to balance the load and for flexibility

```
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx
ID                  NAME                NODE                CURRENT STATE
igssmzpg01uy        nginx.1             worker3             Running 9 seconds ago
uj3ebqt7b215        nginx.2             worker2             Running 9 seconds ago
3kuk1hux067q        nginx.3             worker4             Running 9 seconds ago
rp5d130ch9wx        nginx.4             worker1             Running 9 seconds ago
r7h6pihifopu        nginx.5             master              Running 9 seconds ago
```

- Global mode
  - Schedules single task on every node
  - Good way to keep a track on which node the task is running



Scheduling services

Strategies

- Spread
  - Its a default strategy where in the service tasks are scheduled on the available node
  - Ensures that the tasks are evenly spread over the nodes and clusters
  - if a new node is created and if there are any tasks yet to be scheduled, then those tasks are scheduled on that node.

```
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx-01
ID                  NAME                NODE                CURRENT STATE
grbs7hw43ap7        nginx-01.1          worker2             Running 12 seconds ago
qurr8rwoxfat        nginx-01.2          worker1             Running 12 seconds ago
r49ixmd3oh1d        nginx-01.3          worker3             Running 12 seconds ago
```

```
ubuntu@ip-172-31-28-20:~$ docker service create --name nginx-02 nginx
u37ag0d3kajgc8tyzy0j93qto
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
ubuntu@ip-172-31-28-20:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
wv9a1xus5y2s        nginx-01            replicated          3/3                 nginx:latest
u37ag0d3kajg        nginx-02            replicated          1/1                 nginx:latest
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx-02
ID                  NAME                NODE                CURRENT STATE
2g0rwkwqmtvx        nginx-02.1          master              Running 23 seconds ago
```

- Specific node
  - Tasks can be scheduled on the node or cluster specified.

Reserve Resources

If the reserved memory of the scheduled task is greater than the memory of any node, that task will not be scheduled on that node.

It is automatically scheduled on a different node.

```
ubuntu@ip-172-31-28-20:~$ docker service create --name nginx-01 --reserve-memory 900Mb --replicas 3 nginx
09e2himy0p5aj17083js61ogt
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx-01
ID                  NAME                NODE                CURRENT STATE
lu8pneakqlfc        nginx-01.1          master              Running 21 seconds ago
vbo678khboxh        nginx-01.2          worker2             Running 21 seconds ago
8nqx150pzdqp        nginx-01.3          worker3             Running 21 seconds ago
ubuntu@ip-172-31-28-20:~$ docker service create --name nginx-02 --reserve-memory 200Mb --replicas 4 nginx
qby5mfq30fkrlfhdtvo4plf0m
overall progress: 4 out of 4 tasks
1/4: running   [==================================================>]
2/4: running   [==================================================>]
3/4: running   [==================================================>]
4/4: running   [==================================================>]
verify: Service converged
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx-01
ID                  NAME                NODE                CURRENT STATE
lu8pneakqlfc        nginx-01.1          master              Running 50 seconds ago
vbo678khboxh        nginx-01.2          worker2             Running 50 seconds ago
8nqx150pzdqp        nginx-01.3          worker3             Running 50 seconds ago
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx-02
ID                  NAME                NODE                CURRENT STATE
rews5eqvs6ty        nginx-02.1          worker1             Running 33 seconds ago
hth4mrs5wlg3        nginx-02.2          worker4             Running 33 seconds ago
pkxcs5ac0tu1        nginx-02.3          worker1             Running 33 seconds ago
e44adh0nkakw        nginx-02.4          worker4             Running 33 seconds ago
```

- Zone based
  - In this the nodes can be clustered which can be called a zone
  - Scheduling services only on that zone is zone based scheduling.
  - If there are zones created and spread scheduling is used, then the zone will be considered as a single unit node
  - The service tasks scheduled on that zone will be distributed evenly among the nodes on the zone.

```
ubuntu@ip-172-31-28-20:~$ docker node update --label-add 'com.acme.zone=a' master
master
ubuntu@ip-172-31-28-20:~$ docker node update --label-add 'com.acme.zone=a' worker1
worker1
ubuntu@ip-172-31-28-20:~$ docker node update --label-add 'com.acme.zone=b' worker2
worker2
ubuntu@ip-172-31-28-20:~$ docker node update --label-add 'com.acme.zone=c' worker3
worker3
ubuntu@ip-172-31-28-20:~$ docker node update --label-add 'com.acme.zone=d' worker4
worker4
```

```
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx
ID                  NAME                NODE                CURRENT STATE
4fg17bq1ibqx        nginx.1             worker2             Running 10 seconds ago
133qbxfc5uyd        nginx.2             worker4             Running 10 seconds ago
vk7areq0tmty        nginx.3             worker4             Running 10 seconds ago
p32jicg0i5ei        nginx.4             worker4             Running 10 seconds ago
7ig4qu0tr56i        nginx.5             worker2             Running 10 seconds ago
e7i03x6frnvi        nginx.6             worker2             Running 10 seconds ago
sjs758tx8ulh        nginx.7             master              Running 10 seconds ago
uuhgr5kkv1td        nginx.8             worker1             Running 10 seconds ago
8pv3oxppb37m        nginx.9             worker3             Running 10 seconds ago
sf0d2gszc8xw        nginx.10            worker1             Running 10 seconds ago
qfy03li0tz5u        nginx.11            worker4             Running 10 seconds ago
y9vzaqtm707l        nginx.12            worker3             Running 10 seconds ago
3kic64rw52b0        nginx.13            worker3             Running 10 seconds ago
q9f8r193muc6        nginx.14            worker3             Running 10 seconds ago
mx8qj52nrcrg        nginx.15            master              Running 10 seconds ago
d6v8c4h3el0l        nginx.16            worker2             Running 10 seconds ago
```

- Rescheduling on failure
  - If a node fails, the state of the tasks scheduled on that node and also the current state of the running process is maintained and is scheduled on another node.

```
ubuntu@ip-172-31-28-20:~$ docker service ps --format 'table {{.ID}}\t{{.Name}}\t{{.Node}}\t{{.CurrentState}}' nginx
ID                  NAME                NODE                CURRENT STATE
kg9d5lv92awg        nginx.1             master              Running 23 seconds ago
zdkhw4pd5u81        nginx.2             worker4             Running 23 seconds ago
yovg123zuhct        nginx.3             worker2             Running 23 seconds ago
0qtlcivddplo        nginx.4             worker3             Running 23 seconds ago
rbz43xodi8ts        nginx.5             master              Running 7 seconds ago
tn4z42jjm0qt         \_ nginx.5         worker1             Shutdown 8 seconds ago
ubuntu@ip-172-31-28-20:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
lx0t2ogx7p5da6lq29td8846z *   master              Ready               Active              Leader              18.09.0
oj7m1drsq5jj7qrsg0nkqkl46     worker1             Ready               Drain                                   18.09.0
ls8y6l219kurps71w7svph00w     worker2             Ready               Active                                  18.09.0
sjx781xu7gsohy87rdcxnz78i     worker3             Ready               Active                                  18.09.0
wq6p8avd7da8lrzmzkh6l437a     worker4             Ready               Active                                  18.09.0
```

Consuming services

- Internal service consumption
  - Virtual ip
    - When a new service is created, each service is given a virtual IP address through which it can be accessed,
  - Embedded DNS server
    - Remembering the IP addresses is difficult, as a result swarm makes use of embedded DNS server
    - A query is fired from the service to the DNS server

Load balancing

- External service consumption
  - Routing mesh
    - A service’s port is published on all the nodes in the cluster.
    - Whenever there is an external request, the request is redirected to any free node that has the service’s port, even if the node is not running a service task.
    - The swarm then makes use of the routing mesh to redirect the request to the desired node.
  - Host mode
    - The task’s port is mapped directly on the node’s port and all the service request are redirected on that port.

Updating Services

3 modes:

- **Service Updates**
- **Image Updates**
- **Rollback Updates**

Service Updates

Using the **docker service update --help** command, we can update just about any aspect of our services.

We tried updating the ENV variable of the service. Added the NODE ID.

```
ubuntu@ip-172-31-28-20:~/src/dockercoins$ docker container inspect --format '{{json .Config.Env}}' $(docker container ls -lq) | jq '.'
[
  "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
]
```

After updating:

```
ubuntu@ip-172-31-28-20:~/src/dockercoins$ docker container inspect --format '{{json .Config.Env}}' $(docker container ls -lq) | jq '.'
[
  "NODE_ID=sjx781xu7gsohy87rdcxnz78i",
  "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
]
```

Updating Images

One of the more obvious service updates that might be required is to update the image that the service is based on.

```
ubuntu@ip-172-31-28-20:~/src/dockercoins$ docker service create --detach=false --name nginxhello --publish published=8080,target=80 --mode global nbrown/nginxhello:1.12.1
vbfyr78itgqk9jsr0ugh23kuo
overall progress: 5 out of 5 tasks
wq6p8avd7da8: running   [==================================================>]
lx0t2ogx7p5d: running   [==================================================>]
ls8y6l219kur: running   [==================================================>]
sjx781xu7gso: running   [==================================================>]
oj7m1drsq5jj: running   [==================================================>]
verify: Service converged
ubuntu@ip-172-31-28-20:~/src/dockercoins$ docker service inspect --format '{{.Spec.TaskTemplate.ContainerSpec.Image}}' nginxhello
nbrown/nginxhello:1.12.1@sha256:7b6aab518939e8dff1224e74f6e1f58381b0c4a110f465c4fffdcb69a885a824
ubuntu@ip-172-31-28-20:~/src/dockercoins$ (docker service update --quiet --detach=true --image $IMAGE \
> > --update-delay 15s --update-order "start-first" nginxhello &) > /dev/null 2>&1 && \
> > watch -n 0.1 "docker service ps --filter "desired-state=running" --format '$FORMAT' nginxhello"
-n: command not found
ubuntu@ip-172-31-28-20:~/src/dockercoins$ (docker service update --quiet --detach=true --image $IMAGE > --update-delay 15s --update-order "start-first" nginxhello &) > /dev
/null 2>&1 && > watch -n 0.1 "docker service ps --filter "desired-state=running" --format '$FORMAT' nginxhello"
-n: command not found
ubuntu@ip-172-31-28-20:~/src/dockercoins$ (docker service update --quiet --detach=true --image $IMAGE nginxhello &) > /dev/null 2>&1 && \
> watch -n 1 "docker service ps --filter "desired-state=running" --format '$FORMAT' nginxhello"
ubuntu@ip-172-31-28-20:~/src/dockercoins$ docker service inspect --format '{{.Spec.TaskTemplate.ContainerSpec.Image}}' nginxhello
nbrown/nginxhello:1.13.5@sha256:c0f272bd3b059efa32125abb061aa94935737e87ddbcd2015fa9f2feddd3d9bc
```

Controlled Service Updates

We might, for example, want to update all of the tasks simultaneously, or two at a time, rather than the default behaviour of one at a time.

We can enforce a rollback when an update fails , instead of pause the update, which is the default behaviour.

DEMO

Rolling Back the Updates

If we hit a problem with an update, we can automatically rollback the update, based on the 'update failure action' policy.

We have simulated an update failure, by simply stopping one of the containers that make up the service's tasks.

DEMO

What we learnt?

- How service scheduling works and various ways in which a services are scheduled.
- Understand load balancing and how its performed.
- Different ways of updating the already existing services.

Thank you!