Module-4: Orchestration in Docker

Demo Document - 5

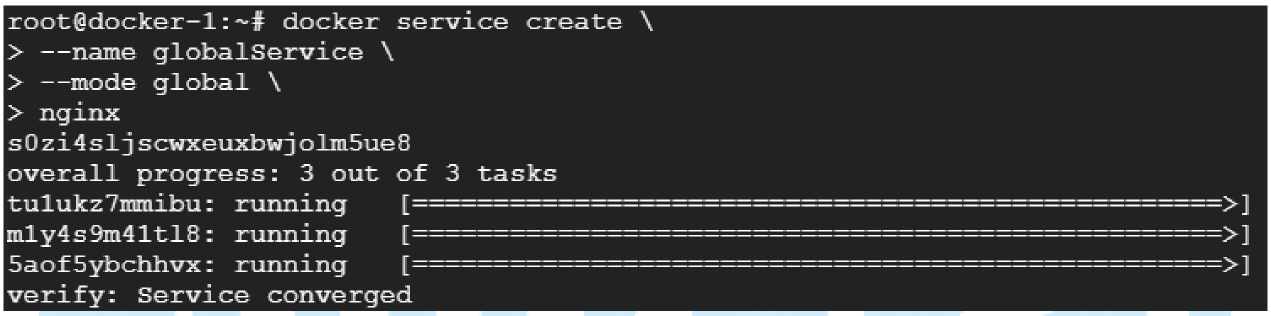
DEMO-5: Service Placement

Note: All commands are executed as root.

Global Service

1. So far, we have only deployed replicated service. To deploy a global service, change the mode to global

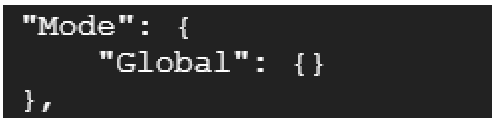
```
$ docker service create \
  --name <serviceName> \
  --mode global \
  <imageName>
```



```
root@docker-1:~# docker service create \
> --name globalService \
> --mode global \
> nginx
s0zi4sljscwxeuxbwjolm5ue8
overall progress: 3 out of 3 tasks
tu1ukz7mmibu: running   [==================================================>]
m1y4s9m41tl8: running   [==================================================>]
5aof5ybchhvx: running   [==================================================>]
verify: Service converged
```

2. Verify using the inspect command

```
$ docker service inspect <serviceName>
```

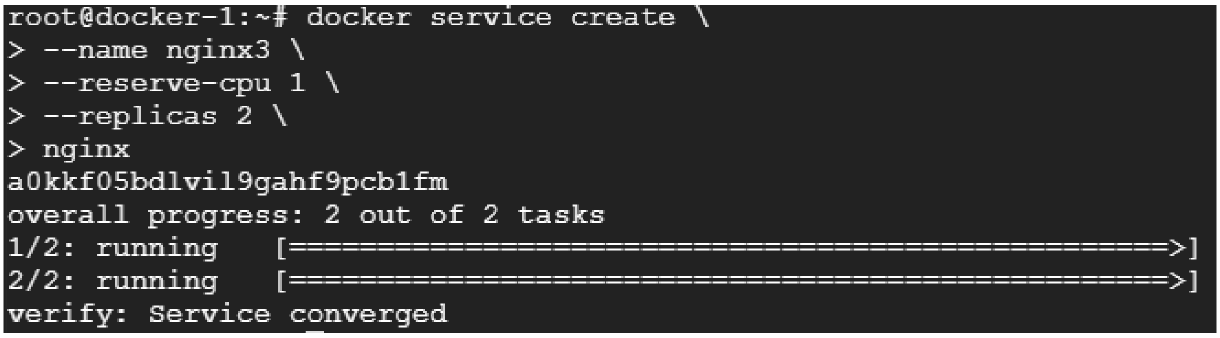


```
"Mode": {
    "Global": {}
},
```

Resource Constraints

1. To put resource constraints on a service use the --reserve-cpu or --reserve-memory flags

```
$ docker service create \
  --name <serviceName> \
  --reserve-cpu 1 \
  <imageName>
```

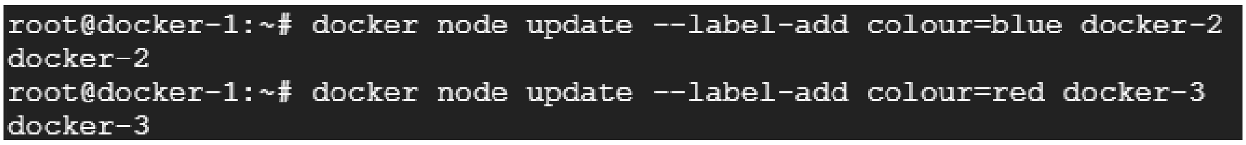


```
root@docker-1:~# docker service create \
> --name nginx3 \
> --reserve-cpu 1 \
> --replicas 2 \
> nginx
a0kkf05bdlvil9gahf9pcb1fm
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
```

Placement Constraint

1. To demonstrate a placement constraint, we will first add labels to our worker node

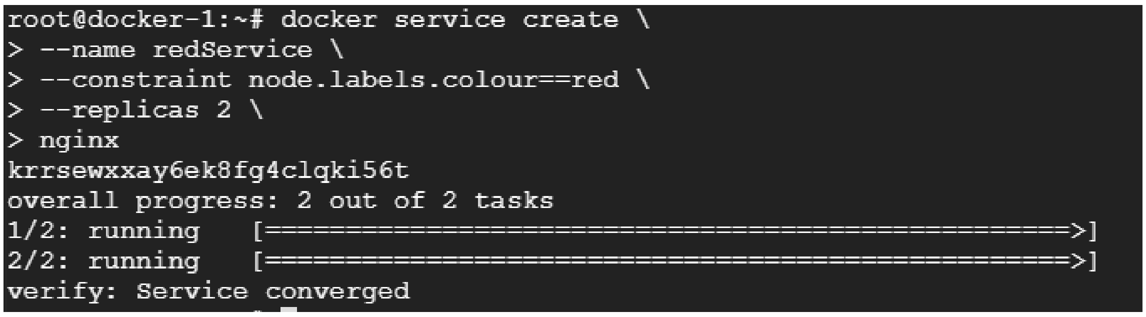
```
$ docker node update --label-add <key>=<value> <nodeName>
```



```
root@docker-1:~# docker node update --label-add colour=blue docker-2
docker-2
root@docker-1:~# docker node update --label-add colour=red docker-3
docker-3
```

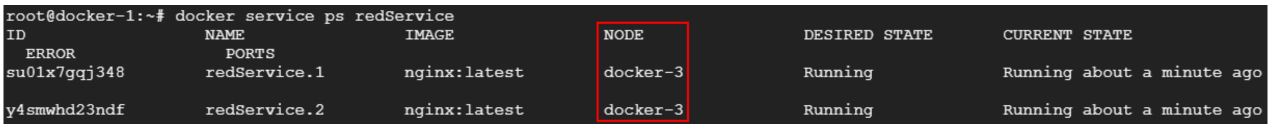
2. Now, deploy a service with a constraint that only lets it create tasks on a node with label red

```
$ docker service create \
  --name <serviceName> \
  --constraint node.labels.colour==red \
  <imageName>
```



```
root@docker-1:~# docker service create \
> --name redService \
> --constraint node.labels.colour==red \
> --replicas 2 \
> nginx
krrsewxxay6ek8fg4clqki56t
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
```

We can see that both the tasks were deployed on the red node i.e. docker-3



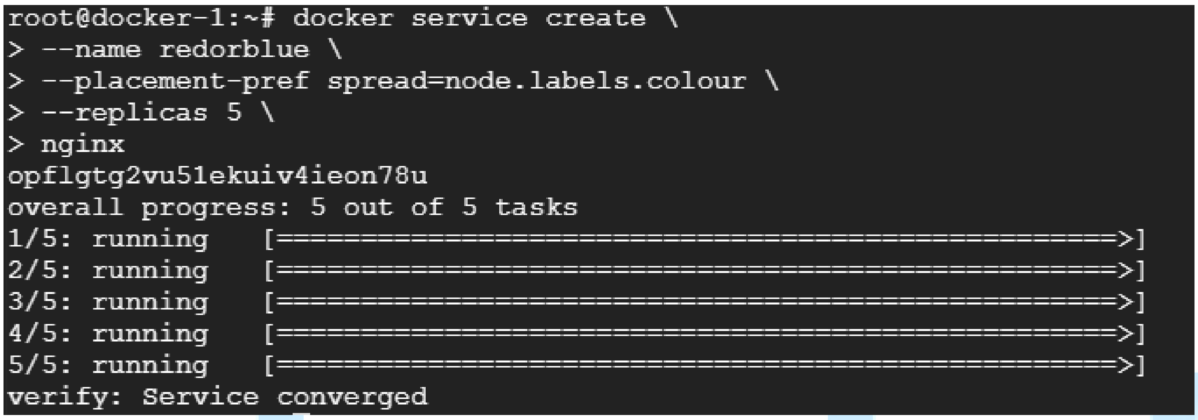
```
root@docker-1:~# docker service ps redService
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
  ERROR               PORTS
su01x7gqj348        redService.1        nginx:latest        docker-3            Running             Running about a minute ago

y4smwhd23ndf        redService.2        nginx:latest        docker-3            Running             Running about a minute ago
```

Placement Preference

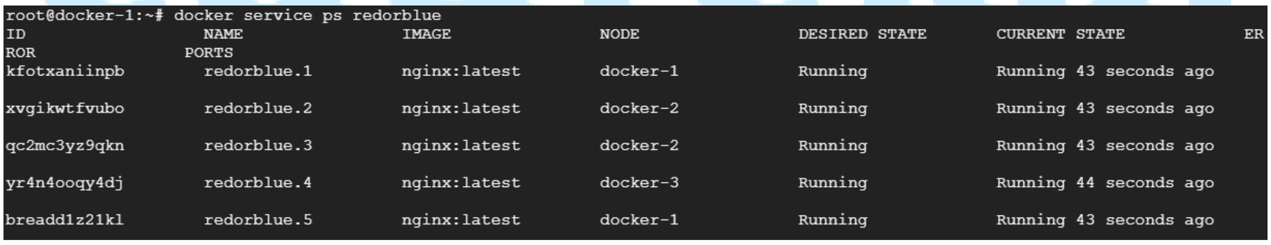
1. Continuing from the previous example, we will deploy a service which prefers to be deployed on a node with a colour label. The strategy used for preference is spread.

```
$ docker service create \
  --name <serviceName> \
  --placement-pref spread=node.labels.colour \
  --replicas 5 \
  <imageName>
```



```
root@docker-1:~# docker service create \
> --name redorblue \
> --placement-pref spread=node.labels.colour \
> --replicas 5 \
> nginx
opflgtg2vu51ekuiv4ieon78u
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service converged
```

2. We can see that tasks were spread evenly across the cluster with some even deployed on the node docker-1 because this is flag is not strictly enforced



```
root@docker-1:~# docker service ps redorblue
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ER
ROR               PORTS
kfotxaniinpb        redorblue.1         nginx:latest        docker-1            Running             Running 43 seconds ago

xvgikwtfvubo        redorblue.2         nginx:latest        docker-2            Running             Running 43 seconds ago

qc2mc3yz9qkn        redorblue.3         nginx:latest        docker-2            Running             Running 43 seconds ago

yr4n4ooqy4dj        redorblue.4         nginx:latest        docker-3            Running             Running 44 seconds ago

breadd1z21kl        redorblue.5         nginx:latest        docker-1            Running             Running 43 seconds ago
```