Docker Swarm Practical: -

To initiate the docker swarm in manager server we use the command: -

docker swarm init

before initiating the swarm if we run “docker info | grep Swarm” we can see the output that swarm is inactive mode,

```
[root@master1 yum.repos.d]# docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[root@master1 yum.repos.d]# docker swarm init
Swarm initialized: current node (m7dayok5u5ohjmx081oii7waz) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-226ieqimldj1j9gy7tixs712h4g2t82s8dhsmcs7dzxphaliez-8678ua9ze2cjny0e0j5wknymv \
    172.31.25.196:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[root@master1 yum.repos.d]# docker info | grep -i swarm
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: active
[root@master1 yum.repos.d]#
```

Next, we can see the available instance in cluster by using the command “docker node ls”

Note: -

If manager status will be Leader means it is primary manager node, if status will be reachable means they are secondary manager node which could come up in case the primary manager node goes down, if manager status will be nothing means they are worker node.

```
[root@master1 yum.repos.d]# docker node ls
ID                            HOSTNAME          STATUS            AVAILABILITY        MANAGER STATUS
m7dayok5u5ohjmx081oii7waz *   master1           Ready             Active              Leader
[root@master1 yum.repos.d]# #Now adding worker node into this swarm by running above join command
[root@master1 yum.repos.d]# docker node ls
ID                            HOSTNAME          STATUS            AVAILABILITY        MANAGER STATUS
1uih64db2maj1jo1gey0un1oi     node3             Ready             Active
m7dayok5u5ohjmx081oii7waz *   master1           Ready             Active              Leader
qvi2cg79mhfk6xev08wcmhohj     node1             Ready             Active
u0su7t3x4zuca0gfk64tz2wva     node2             Ready             Active
[root@master1 yum.repos.d]#
```

After running “docker swarm init” in the manager node we will get the join token as output, running that token we can add any number of nodes into this cluster. Now we can get the token for worker node as well as manager node using the command: -

docker swarm join-token worker

docker swarm join-token manager

We can add the manager to the cluster by two ways, first is either by running this join token in any of the nodes, that nodes will become manager nodes, or other way is by promoting the existing worker node to manager node, using command “docker node promote node_id”, if we want to bring any

```
[root@master1 yum.repos.d]# #to see the join token
[root@master1 yum.repos.d]# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-226ieqimldj1j9gy7tixs712h4g2t82s8dhsmcs7dzxphaliez-8678ua9ze2cjny0e0j5wknymv \
    172.31.25.196:2377

[root@master1 yum.repos.d]# docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-226ieqimldj1j9gy7tixs712h4g2t82s8dhsmcs7dzxphaliez-4dhnh3gw5fqhll9c8nq6fd9mu \
    172.31.25.196:2377

[root@master1 yum.repos.d]#
```

To remove any node from node list which is down or inactive status we use the command: -

docker node rm node_name

** Note: - But if we remove any node from swarm cluster which is in running state, then it will immediately bring up new container in any of the host which is a part of swarm cluster.

If we want to leave the docker swarm cluster then use the command in the server which we want to move out of cluster: -

docker swarm leave

But if we run the same command in master then we will get warning alert that this is master server are you sure you want to move out of swarm, so we have to use –force along with command to move of swarm: -

docker swarm leave –force

- ➔ We can promote any worker node to master node using command promote, but the important point is promotion can be done for any worker node from master node only, means if we run this promote command from worker node we will get the error, so only manager node has access to promote any worker node as a part of master node, we uses below command:-
  docker node promote node_name

- ➔ If we want to make any manager node to worker node we can use the demote command, the node will be immediately converted to worker, use below command to do so: -
  docker node demote node_name

```
[root@master1 yum.repos.d]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
1uih64db2maj1jo1gey0un1oi     node3               Ready               Active
m7dayok5u5ohjmx081oii7waz *   master1             Ready               Active              Leader
qvi2cg79mhfk6xev08wcmhohj     node1               Ready               Active
u0su7t3x4zuca0gfk64tz2wva     node2               Ready               Active
[root@master1 yum.repos.d]# #now promoting node3 to master
[root@master1 yum.repos.d]# docker node promote node3
Node node3 promoted to a manager in the swarm.
[root@master1 yum.repos.d]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
1uih64db2maj1jo1gey0un1oi     node3               Ready               Active              Reachable
m7dayok5u5ohjmx081oii7waz *   master1             Ready               Active              Leader
qvi2cg79mhfk6xev08wcmhohj     node1               Ready               Active
u0su7t3x4zuca0gfk64tz2wva     node2               Ready               Active
[root@master1 yum.repos.d]# #now demoting node3 again to worker node
[root@master1 yum.repos.d]# docker node demote node3
Manager node3 demoted in the swarm.
[root@master1 yum.repos.d]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
1uih64db2maj1jo1gey0un1oi     node3               Ready               Active
m7dayok5u5ohjmx081oii7waz *   master1             Ready               Active              Leader
qvi2cg79mhfk6xev08wcmhohj     node1               Ready               Active
u0su7t3x4zuca0gfk64tz2wva     node2               Ready               Active
[root@master1 yum.repos.d]#
```

In docker we use run command to create any container likewise we use service in docker swarm, all the options which we used in docker run command we can use it with docker run.

In docker swarm we use –replicas to scale up out container to the desired count

We have two things here to scale up our container count: -

First is –replicas where we have to give the count so that the count will be maintained through out the lifecycle till swarm services will remain active, and 2nd option is "–mode global", this global mode will bring up only one container in all the hosts which is part of cluster and will maintain its state throughout the lifecycle. We can use global mode to run such container whose single instance we need in all our hosts eg. Monitoring tools.

Docker service example: -

docker service create --name my-tomcat --replicas 6 -p 1234:8080 tomcat

Once we run above command 6 tomcat containers will be created.

We can view the my_tomcat service using command: -

docker service ls

Then if we want to see in which server this container is running then we can use the command: -

docker service ps my_tomcat

it will show complete details that in which server our my_tomcat containers are running.

We can do all this practice in play with docker website, it provides us free access to run docker commands only limitation is the sessions remains active for 4 hrs.

we also have options with us to use constraints so to make choice that in which worker node the container should run, we add the constraints using the Labels. We can view the labels by running the command, “docker inspect node_name”, there we can find the term labels: -

```
[root@master1 ~]# docker inspect master1
[
    {
        "ID": "kvovq03lop7mcy75az6c1ffhf",
        "Version": {
            "Index": 610
        },
        "CreatedAt": "2019-06-22T13:04:56.956462781Z",
        "UpdatedAt": "2019-06-23T05:50:14.108932831Z",
        "Spec": {
            "Labels": {},
            "Role": "manager",
            "Availability": "drain"
        },
```

Here by can see the Labels clearly, inside Labels Availability is drain which I made manually, which means no container will run in my master node, by default container runs in master node as well, but using this constraint now it won’t run. Similarly we can add constraints of our wish as shown below: -

```
[root@master1 ~]# docker node update --label-add type=DevOpsG master1
master1
[root@master1 ~]# docker inspect master1
[
    {
        "ID": "kvovq03lop7mcy75az6c1ffhf",
        "Version": {
            "Index": 619
        },
        "CreatedAt": "2019-06-22T13:04:56.956462781Z",
        "UpdatedAt": "2019-06-23T07:02:58.060140739Z",
        "Spec": {
            "Labels": {
                "type": "DevOpsG"
            },
            "Role": "manager",
            "Availability": "drain"
        },
```

In below snap, one intresting thing to note is that we had run given the constraints that httpd should run in master server, but we haven’t got any error as already I made master as drain earlier, but in replicas we can see 0/2 which means service is created but the service can’t run any container, as “–availability = drain” in master. Not let’s make the “--availability = active” then review the changes.

```
[root@master1 ~]# docker service create --name myhttpd --replicas=2 --constraint node.labels.type==DevOpsG -p 1234:80 http
t8eshj152wx2kyd1xbsscn5hp
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
[root@master1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@master1 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@master1 ~]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
kr2as47xo2b6        my_nginu            replicated          1/1                 nginx:latest        *:1452->80/tcp
t8eshj152wx2        myhttpd             replicated          0/2                 httpd:latest        *:1234->80/tcp
[root@master1 ~]#
```

we can notice that when we made availability active immediately docker containers came up.

```
[root@master1 ~]# docker node update --availability active master1
master1
[root@master1 ~]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
kr2as47xo2b6        my_nginu            replicated          1/1                 nginx:latest        *:1452->80/tcp
t8eshj152wx2        myhttpd             replicated          2/2                 httpd:latest        *:1234->80/tcp
[root@master1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
7be1fe7283fa        httpd:latest        "httpd-foreground"  5 seconds ago       Up 3 seconds        80/tcp              myhttpd.1.kazg2
2btfac9yurmdx9
98e6f16dfaf9        httpd:latest        "httpd-foreground"  5 seconds ago       Up 3 seconds        80/tcp              myhttpd.2.eh6vd
oyxhri4r2gx5x9
[root@master1 ~]#
```

Don’t use P for port mapping which we were doing in running docker container, in swarm -P don’t assigns the port automatically, however we need to do the port mapping manually by using -p which is show in below snap: -

```
[root@master1 ~]# docker service create --name my-tommy --replicas 5 -P tomcat
unknown shorthand flag: 'P' in -P
See 'docker service create --help'.
[root@master1 ~]# docker service create --name my-tommy --replicas 5 -p 1234:8080 tomcat
rsjpymht4kxql2ykpay508ubn
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
[root@master1 ~]# docker service ls
ID             NAME        MODE         REPLICAS   IMAGE          PORTS
rsjpymht4kxq   my-tommy    replicated   5/5        tomcat:latest  *:1234->8080/tcp
[root@master1 ~]# docker service ps my-tommy
ID             NAME        IMAGE        NODE       DESIRED STATE  CURRENT STATE
   PORTS
smey4k5tt9yn   my-tommy.1  tomcat:latest node2     Running        Running 13 seconds ago

88665jmw8ye2   my-tommy.2  tomcat:latest master1   Running        Running 13 seconds ago

xbdde7ewjp9g   my-tommy.3  tomcat:latest node2     Running        Running 13 seconds ago

p77a9mdud8o7   my-tommy.4  tomcat:latest master1   Running        Running 13 seconds ago

zc0z4xyvfsxs   my-tommy.5  tomcat:latest node1     Running        Running 13 seconds ago

[root@master1 ~]#
```

Here we had mounted the volume as well to show how swarm redirects to another server in the cluster and manages load balancing task. Now if we delete some of the running container immediately other container will come up in the cluster to maintain the desired state of cluster.

```
[root@master1 volumes]# docker volume create DevOpsG
DevOpsG
[root@master1 volumes]# ls
DevOpsG  metadata.db
[root@master1 volumes]# pwd
/var/lib/docker/volumes
[root@master1 volumes]# cd -
/root
[root@master1 ~]# docker service create --name my-nginx --replicas 6 --mount source=DevOpsG,target=/usr/share/nginx/html -p 1234:80 nginx
5sknud4qrvuj52h5tqr8zxw9w
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
[root@master1 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS         PORTS     NAMES
9e48b40f5003   nginx:latest   "nginx -g 'daemon ..."   6 seconds ago  Up 5 seconds   80/tcp    my-nginx.2.hioo
lq7gwzhigrpanj9cmlp
a2d3db5dd9ce   nginx:latest   "nginx -g 'daemon ..."   6 seconds ago  Up 5 seconds   80/tcp    my-nginx.5.xy20
81jyoyu5v8dn9s7dieg
```

In replicas we can use the mode as global also, if mode will be global then a single container will come up in all host server which are part of swarm cluster: -

```
[root@master1 _data]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
[root@master1 _data]# docker node ls | grep Ready
6krvx8bg0k4gdtydslxvq7dzv     node1               Ready               Active
kvovq03lop7mcy75az6c1ffhf *   master1             Ready               Active              Leader
r0eugdxoc1fbvtl8xjq5bajtz     node2               Ready               Active              Reachable
[root@master1 _data]# docker service create --name my-global --mode global -p 1234:80 nginx
u3d91kuj33b8vua22et9g46dm
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
[root@master1 _data]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
u3d91kuj33b8        my-global           global              3/17                nginx:latest        *:1234->80/tcp
```

In below snap we had increased the replicas count from 2 to 5 using update command:-

```
[root@node2 _data]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
[root@node2 _data]# docker service create --name my_tommy --replicas=2 -p 1234:8080 tomcat
ife7cbwfhtnd7y9rev2yak965
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
[root@node2 _data]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
ife7cbwfhtnd        my_tommy            replicated          2/2                 tomcat:latest       *:1234->8080/tcp
[root@node2 _data]# docker service ps my_tommy
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE           ERROR
fydbvm37yoju        my_tommy.1          tomcat:latest       node1               Running             Running 22 seconds ago
cvpn2d0rds37        my_tommy.2          tomcat:latest       node2               Running             Running 22 seconds ago
[root@node2 _data]# # now suppose i want to increase the replicas from 2 to 4, so use update command
[root@node2 _data]# docker service update --replicas=5 my_tommy
my_tommy
Since --detach=false was not specified, tasks will be updated in the background.
In a future release, --detach=false will become the default.
[root@node2 _data]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
ife7cbwfhtnd        my_tommy            replicated          5/5                 tomcat:latest       *:1234->8080/tcp
[root@node2 _data]#
```

How to make the master node to drain condition so that no container should run on it, by default all swarm server remains in active mode, we need to make our manager server in drain mode as show below:-

```
[root@master1 ~]# docker node inspect master1
[
    {
        "ID": "kvovq03lop7mcy75az6c1ffhf",
        "Version": {
            "Index": 625
        },
        "CreatedAt": "2019-06-22T13:04:56.956462781Z",
        "UpdatedAt": "2019-06-23T07:09:50.115142943Z",
        "Spec": {
            "Labels": {
                "type": "DevOpsG"
            },
            "Role": "manager",
            "Availability": "active"
```

We can see in previous snap, the availability contraints is active, we need to make it as drain as show below:-

```
[root@master1 ~]# docker node update --availability drain master1
master1
[root@master1 ~]# docker node inspect master1
[
    {
        "ID": "kvovq03lop7mcy75az6c1ffhf",
        "Version": {
            "Index": 695
        },
        "CreatedAt": "2019-06-22T13:04:56.956462781Z",
        "UpdatedAt": "2019-06-23T08:26:07.197218419Z",
        "Spec": {
            "Labels": {
                "type": "DevOpsG"
            },
            "Role": "manager",
            "Availability": "drain"
        },
```