Lesson 07 Demo 03

Scaling Microservices with Docker Swarm

Objective: To demonstrate effective scaling of microservices using Docker Swarm, facilitating easy development, deployment, and management within a microservices architecture.

Tools required: Docker

Prerequisites: Docker must be installed on your system

Steps to be followed:

1. Initialize Docker Swarm
2. Define microservices in Docker Compose
3. Deploy microservices using the Docker Stack
4. Monitor and manage your microservices

Step 1: Initialize Docker Swarm

1.1 Run the following command to initialize Docker Swarm, if it is not already set up:
docker swarm init

```
labuser@ip-172-31-22-247:~$ docker swarm init
Swarm initialized: current node (uz6h72cd0mcwkh6j3rwivq1yv) is now a manager.

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

    docker swarm join --token SWMTKN-1-2vy9wwiku3w4dbszgtcn31v8k5na45j21atwky3hx
w7syod6he-845otecx5vx135y5zo9u31zh0 172.31.22.247:2377

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

Step 2: Define microservices in Docker Compose

2.1 Execute the following command to create a **docker-compose.yml** file:

vi docker-compose.yml

```
w7syod6he-845otecx5vx135y5zo9u31zh0 172.31.22.247:2377

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

labuser@ip-172-31-22-247:~$ vi docker-compose.yml
```

2.2 Add the following script in the **docker-compose.yml** file:

```
version: '3.8'
services:
 web-app:
  image: nginx
  ports:
   - "5001:5001"
  deploy:
   replicas: 3
   update_config:
    parallelism: 2
    delay: 10s
 api-service:
  image: redis
  ports:
   - "4001:4001"
  deploy:
   replicas: 2
   update_config:
    parallelism: 2
    delay: 10s
```

```
version: '3.8'
services:
  web-app:
    image: nginx
    ports:
      - "5001:5001"
    deploy:
      replicas: 3
      update_config:
        parallelism: 2
        delay: 10s
  api-service:
    image: redis
    ports:
      - "4001:4001"
    deploy:
      replicas: 2
      update_config:
        parallelism: 2
        delay: 10s
~
~
```

Step 3: Deploy microservices using the Docker Stack

3.1 Run the following command to deploy your microservices to the swarm:
docker stack deploy -c docker-compose.yml mystack

```
labuser@ip-172-31-22-247:~$ docker stack deploy -c docker-compose.yml mystack
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Updating service mystack_web-app (id: mypprdx6ktm2o7ufpuwxk8y1v)
image my-web-app:latest could not be accessed on a registry to record
its digest. Each node will access my-web-app:latest independently,
possibly leading to different nodes running different
versions of the image.

Updating service mystack_api-service (id: jpuhgwexfno9ekiznaq6yzp0x)
image my-api-service:latest could not be accessed on a registry to record
its digest. Each node will access my-api-service:latest independently,
possibly leading to different nodes running different
versions of the image.
```

Step 4: Monitor and manage your microservices

4.1 Run the following command to manage your microservices, including scaling up or down:
docker service scale mystack_web-app=1
docker service scale mystack_api-service=1

```
labuser@ip-172-31-22-247:~$ docker service scale mystack_web-app=1
mystack_web-app scaled to 1
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service mystack_web-app converged
```

```
labuser@ip-172-31-22-247:~$ docker service scale mystack_api-service=1
mystack_api-service scaled to 1
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service mystack_api-service converged
```

4.2 Run the following command to check the status of the microservices:
docker service ls

```
labuser@ip-172-31-22-247:~$ docker service ls
ID             NAME                  MODE         REPLICAS   IMAGE               PORTS
wt3py2fwq7wf   mern_web-app          replicated   0/3        my-web-app:latest   *:5000->5000/tcp
jpuhgwexfno9   mystack_api-service   replicated   1/1        redis:latest        *:4001->4001/tcp
mypprdx6ktm2   mystack_web-app       replicated   1/1        nginx:latest        *:5001->5001/tcp
labuser@ip-172-31-22-247:~$
```

4.3 Run the following command to deploy microservices to the Swarm:
docker stack deploy -c docker-compose.yml mystack

```
labuser@ip-172-31-22-247:~$ docker stack deploy -c docker-compose.yml mystack
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Updating service mystack_web-app (id: mypprdx6ktm2o7ufpuwxk8y1v)
Updating service mystack_api-service (id: jpuhgwexfno9ekiznaq6yzp0x)
labuser@ip-172-31-22-247:~$
```

By following these steps, you have successfully scaled your services using Docker Swarm, ensuring high availability and effective load distribution across your microservices architecture. This setup also facilitates easy updates and management of services through Docker Compose.