Lesson 04 Demo 05

Converting an Application Deployment into Stack

Objective: To convert application deployment to Docker stack using docker-compose.yml for efficient management within Docker Swarm

Tools required: Docker

Prerequisites: Setup of a swarm mode

Steps to be followed:

1. Drain worker nodes
2. Start, check, and verify the registry service
3. Create a flask application
4. Create a requirement file
5. Create a Dockerfile and define the docker-compose configuration
6. Install docker-compose and start the application
7. Stop the application and push it to the registry
8. Deploy the stack and check its status
9. Test the application and remove the stack

Step 1: Drain worker nodes

1.1 Execute the following command to list all the nodes present in the swarm and ensure that all nodes are in active state:
sudo docker node ls

```
labsuser@ip-172-31-29-216:~$ sudo docker node ls
ID                            HOSTNAME           STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
i1w0ayok8416j8ts3ymcbdzbt     ip-172-31-26-147   Ready     Active                          19.03.12
tm977npw6l2a01mqm5grq9if1 *   ip-172-31-29-216   Ready     Active         Leader           20.10.2
xdeww81h036jheino68pc6tgc     ip-172-31-30-210   Ready     Active                          19.03.12
labsuser@ip-172-31-29-216:~$
```

1.2 Use the following command to drain the worker nodes:
sudo docker node update --availability drain hostname_Worker_Node

```
labsuser@ip-172-31-29-216:~$ sudo docker node update --availability drain ip-172-31-26-147
ip-172-31-26-147
labsuser@ip-172-31-29-216:~$ sudo docker node update --availability drain ip-172-31-30-210
ip-172-31-30-210
labsuser@ip-172-31-29-216:~$
```

Note: Run the cd command to return to the home directory

Step 2: Start, check, and verify the registry service

2.1 Use the following commands to start the registry as a service on the swarm:

**sudo docker service create --name registry **
--publish published=5000,target=5000 registry:2

```
labsuser@ip-172-31-29-216:~$ sudo docker service create --name registry \
> --publish published=5000,target=5000 registry:2
n4c9knwydx6qotd3rob9im25d
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
labsuser@ip-172-31-29-216:~$
```

2.2 List the running services to check the status of the registry service by executing the following command:

sudo docker service ls

```
labsuser@ip-172-31-29-216:~$ sudo docker service ls
ID             NAME       MODE         REPLICAS   IMAGE         PORTS
r08m4aan34mm   redis      replicated   3/3        redis:3.0.6
n4c9knwydx6q   registry   replicated   1/1        registry:2    *:5000->5000/tcp
labsuser@ip-172-31-29-216:~$
```

2.3 Execute the following command to check if the registry service is working:

curl http://localhost:5000/v2/

```
labsuser@ip-172-31-29-216:~$ curl http://localhost:5000/v2/
{}labsuser@ip-172-31-29-216:~$
```

The response, an empty JSON object **{}**, indicates that the Docker registry is up and running and supports the Docker registry HTTP API V2.

Step 3: Create a flask application

3.1 Create and move to a directory for the project by executing the following command:
mkdir stackdemo
cd stackdemo

```
labsuser@ip-172-31-29-216:~$ mkdir stackdemo
labsuser@ip-172-31-29-216:~$ cd stackdemo
labsuser@ip-172-31-29-216:~/stackdemo$
```

3.2 Create a file called **app.py** in the **stackdemo** directory by using the following command:
nano app.py

```
labsuser@ip-172-31-29-216:~/stackdemo$ nano app.py
labsuser@ip-172-31-29-216:~/stackdemo$
```

3.3 Add the following code in the **app.py** file:

```
from flask import Flask
from redis import Redis

app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
   count = redis.incr('hits')
   return 'Hello World! I have been seen {} times.\n'.format(count)

if __name__ == "__main__":
   app.run(host="0.0.0.0", port=8000, debug=True)
```

```
  GNU nano 2.9.3

from flask import Flask
from redis import Redis
app = Flask(__name__)
redis = Redis(host='redis', port=6379)
@app.route('/')
def hello():
    count = redis.incr('hits')
    return 'Hello World! I have been seen {} times.\n'.format(count)
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000, debug=True)
```

This Flask application uses Redis to count and display the number of times the root URL has been accessed.

Note: Press **Ctrl+O** to save the file. Then, press **Enter** and **Ctrl+X** to exit.

Step 4: Create a requirement file

4.1 Use the following command to create and open a **requirements.txt**:

nano requirements.txt

```
labsuser@ip-172-31-29-216:~/stackdemo$ nano requirements.txt
labsuser@ip-172-31-29-216:~/stackdemo$
```

4.2 Add the following text in the **requirements.txt** file:

flask

redis

```
GNU nano 2.9.3

flask
redis
```

Note: Press **Ctrl+O** to save the file. Then press **Enter** and **Ctrl+X** to exit.

Step 5: Create a Dockerfile and define the docker-compose configuration

5.1 Use the following command to create a Dockerfile:

nano Dockerfile

```
labsuser@ip-172-31-29-216:~/stackdemo$ nano Dockerfile
labsuser@ip-172-31-29-216:~/stackdemo$
```

5.2 Add the following code in the Dockerfile:

FROM python:3.4-alpine

ADD . /code

WORKDIR /code

RUN pip install -r requirements.txt

CMD ["python", "app.py"]

```
GNU nano 2.9.3

FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

Note: Press **Ctrl+O** to save the file. Then press **Enter** and **Ctrl+X** to exit.

5.3 Use the following command to create the **docker-compose.yml** file:

nano docker-compose.yml

```
labsuser@ip-172-31-29-216:~/stackdemo$ nano docker-compose.yml
labsuser@ip-172-31-29-216:~/stackdemo$
```

5.4 Add the following code in the **docker-compose.yml** file:

```
version: "3.3"
services:
  web:
    image: 127.0.0.1:5000/stackdemo
    build: .
    ports:
      - "8000:8000"
  redis:
    image: redis:alpine
```

```
  GNU nano 2.9.3

version: "3.3"

services:
  web:
    image: 127.0.0.1:5000/stackdemo
    build: .
    ports:
      - "8000:8000"
  redis:
    image: redis:alpine
```

Note: Press Ctrl+O to save the file. Then press Enter and Ctrl+X to exit.

Step 6: Install docker-compose and start the application

6.1 Use the following commands to install **docker-compose**:

sudo curl -L "https://github.com/docker/compose/releases/download/
1.29.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

```
labsuser@ip-172-31-41-11:~$ sudo curl -L "https://github.com/docker/compose/releases/download/\
> 1.29.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   633  100   633    0     0   7033      0 --:--:-- --:--:-- --:--:--  7033
100 12.1M  100 12.1M    0     0  19.5M      0 --:--:-- --:--:-- --:--:-- 19.5M
```

6.2 To grant executable permission and see the version, use the following commands:

sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version

```
labsuser@ip-172-31-41-11:~$ sudo chmod +x /usr/local/bin/docker-compose
labsuser@ip-172-31-41-11:~$ docker-compose --version
docker-compose version 1.29.1, build c34c88b2
labsuser@ip-172-31-41-11:~$
```

6.3 Start **docker-compose** using the following command:
sudo docker-compose up -d

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "stackdemo_default" with the default driver
Building web
Step 1/5 : FROM python:3.4-alpine
3.4-alpine: Pulling from library/python
8e402f1a9c57: Pull complete
cda9ba2397ef: Pull complete
aafecf9bbbfd: Pull complete
bc2e7e266629: Pull complete
e1977129b756: Pull complete
Digest: sha256:c210b660e2ea553a7afa23b41a6ed112f85dbce25cbcb567c75dfe05342a4c4b
Status: Downloaded newer image for python:3.4-alpine
 ---> c06adcf62f6e
Step 2/5 : ADD . /code
 ---> 0fa406645a78
Step 3/5 : WORKDIR /code
 ---> Running in 2b265002d534
Removing intermediate container 2b265002d534
 ---> 68a225fbb883
Step 4/5 : RUN pip install -r requirements.txt
 ---> Running in 1b06f1ff080b
```

```
Step 5/5 : CMD ["python", "app.py"]
 ---> Running in 5e63adebed6a
Removing intermediate container 5e63adebed6a
 ---> 3e943d52e720

Successfully built 3e943d52e720
Successfully tagged 127.0.0.1:5000/stackdemo:latest
WARNING: Image for service web was built because it did not already exist. To rebuild t
Pulling redis (redis:alpine)...
alpine: Pulling from library/redis
801bfaa63ef2: Pull complete
9a8d0188e481: Pull complete
8a3f5c4e0176: Pull complete
3f7cb00af226: Pull complete
e421f2f8acb5: Pull complete
f41cc3c7c3e4: Pull complete
Digest: sha256:2cd821f730b90a197816252972c2472e3d1fad3c42f052580bc958d3ad641f96
Status: Downloaded newer image for redis:alpine
Creating stackdemo_web_1   ... done
Creating stackdemo_redis_1 ... done
```

6.4 Use the following commands to check whether the application is running:

sudo docker-compose ps

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker-compose ps
      Name                     Command               State           Ports
-------------------------------------------------------------------------------
stackdemo_redis_1   docker-entrypoint.sh redis ...   Up      6379/tcp
stackdemo_web_1     python app.py                    Up      0.0.0.0:8000->8000/tcp
```

6.5 Verify if the application is running by using the following command:

curl http://localhost:8000

```
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 1 times.
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 2 times.
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 3 times.
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 4 times.
```

Step 7: Stop the application and push it to the registry

7.1 Bring the application down by using the following command:
sudo docker-compose down –volumes

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker-compose down --volumes
Stopping stackdemo_redis_1 ... done
Stopping stackdemo_web_1   ... done
Removing stackdemo_redis_1 ... done
Removing stackdemo_web_1   ... done
Removing network stackdemo_default
labsuser@ip-172-31-29-216:~/stackdemo$
```

7.2 Push the application to the registry by using the following command:
sudo docker-compose push

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker-compose push
Pushing web (127.0.0.1:5000/stackdemo:latest)...
The push refers to repository [127.0.0.1:5000/stackdemo]
d1d576b560d4: Pushed
89a7a541a462: Pushed
62de8bcc470a: Pushed
58026b9b6bf1: Pushed
fbe16fc07f0d: Pushed
aabe8fddede5: Pushed
bcf2f368fe23: Pushed
latest: digest: sha256:a401b863a190326c74d858c0b19dd735bf669054e5afc4e5c567dd3d05b30bf6 size: 1790
```

Step 8: Deploy the stack and check its status

8.1 Use the following command to deploy a Docker stack using the services defined in the **docker-compose.yml** file:
sudo docker stack deploy --compose-file docker-compose.yml stackdemo

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker stack deploy --compose-file docker-compose.yml stackdemo
Ignoring unsupported options: build

Creating network stackdemo_default
Creating service stackdemo_web
Creating service stackdemo_redis
```

8.2 Run the following command to check the running status of the stack:
sudo docker stack services stackdemo

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker stack services stackdemo
ID             NAME             MODE         REPLICAS   IMAGE                            PORTS
lqyhnc7brrvr   stackdemo_redis  replicated   1/1        redis:alpine
zmp3rb0ri9it   stackdemo_web    replicated   1/1        127.0.0.1:5000/stackdemo:latest  *:8000->8000/tcp
labsuser@ip-172-31-29-216:~/stackdemo$
```

Step 9: Test the application and remove the stack

9.1 Test the app again with the following curl command:
curl http://localhost:8000
curl http://ip-172-31-26-147:8000

```
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 1 times.
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 2 times.
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://localhost:8000
Hello World! I have been seen 3 times.
labsuser@ip-172-31-29-216:~/stackdemo$ curl http://ip-172-31-26-147:8000
Hello World! I have been seen 4 times.
```

Note: In step 7, while starting the **docker-compose** if you get an error showing the port is already assigned, run the command **sudo docker ps** and kill the container with the same port, and then proceed.

9.2 Use the following command to bring the stack down:
sudo docker stack rm stackdemo

```
labsuser@ip-172-31-29-216:~/stackdemo$ sudo docker stack rm stackdemo
Removing service stackdemo_redis
Removing service stackdemo_web
Removing network stackdemo_default
```

By following these steps, you have successfully transitioned the application deployment into a Docker stack, leveraging the docker-compose.yml file for enhanced management and scalability within the Docker swarm environment.