Create a Three Tier Application using Docker SOP

Cognizant

Table of Contents

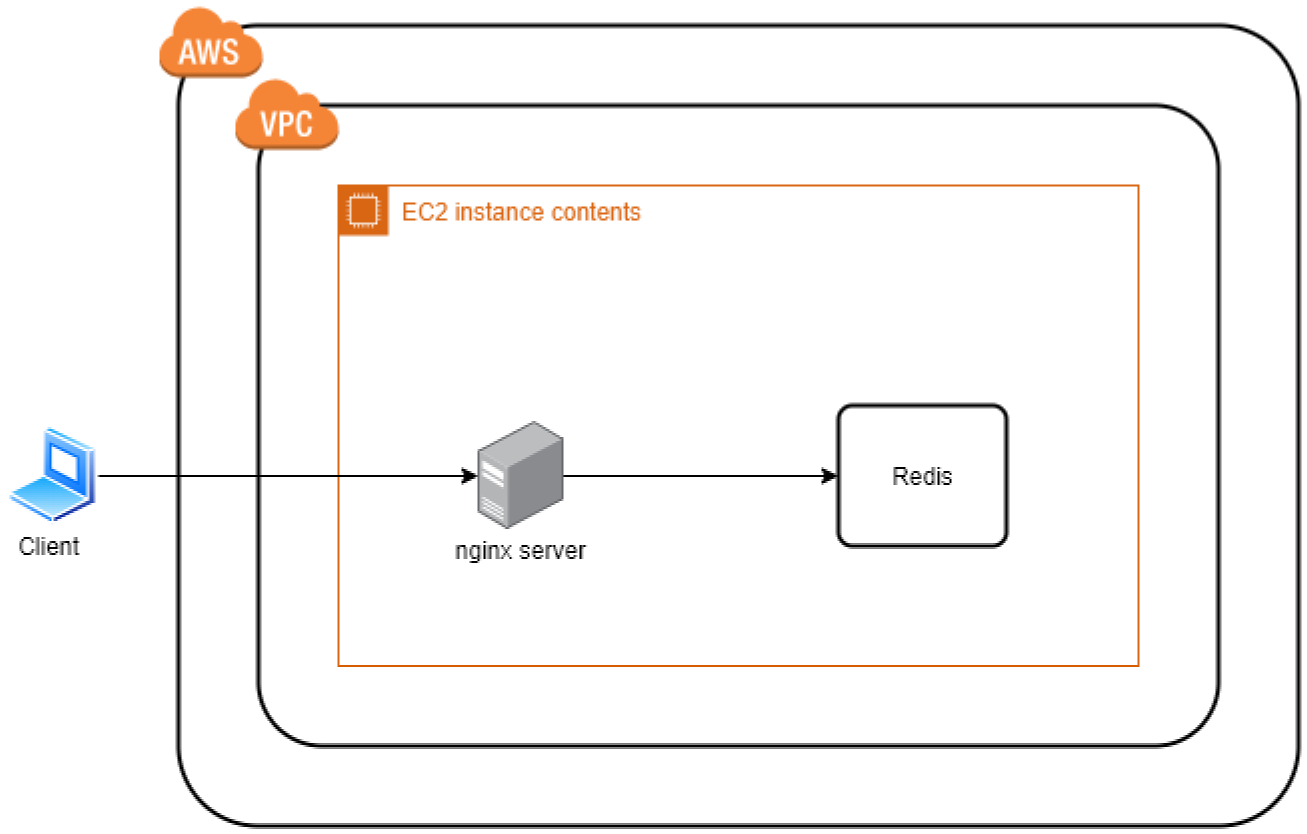
1. Create a Three Tier Application using Docker

1.1 Description

Docker is a great tool for running and orchestrating software. A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. To build a multi-tier application effectively we can docker compose tool. Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.
In this lab exercise, you will build a three-tier application using docker-compose tool.

1.2 Architecture Diagram

The diagram below displays a visual representation of the application architecture:



1.3 Lab Steps

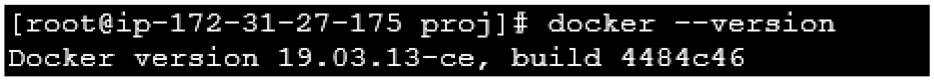
Follow the steps outlined below to achieve the objective of this lab exercise:

1. Log in to AWS console with the credentials provided.

1.3.1 Create Ec2 Instance and install docker

1. Navigate to EC2 console and launch an Amazon Linux 2 instance.
2. Once instance is launched and is running ssh into the instance either using PuTTY or directly from the console using Session Manager by attaching **CCL-EC2-Role**.
3. Install docker if not installed:

```
• yum update -y
• sudo yum install -y docker
• docker -version
```



```
[root@ip-172-31-27-175 proj]# docker --version
Docker version 19.03.13-ce, build 4484c46
```

```
• service docker status
• sudo service docker start
```

4. Install Docker compose from the following link: https://docs.docker.com/compose/install/

1.3.2 Create Docker compose file

1. Create a directory and inside that, perform the following steps.
2. In this docker compose file we are having a web frontend, nginx reverse proxy, redis.
3. Add the following files in the folder:
   a. **vi docker-compose.yml** - You can find that proxy, web and redis have been created:

```
proxy:
    image: jwilder/nginx-proxy
    ports:
        - "8080:80"
    volumes:
        - /var/run/docker.sock:/tmp/docker.sock:ro
web:
    build: .
    ports:
        - "5000"
    volumes:
        - .:/code
    links:
        - redis
    environment:
        - VIRTUAL_HOST=192.168.99.100
        - VIRTUAL_PORT=5000
redis:
    image: redis
```

b. **vi Dockerfile** - This Dockerfile is used to build app.py

```
FROM python:3.8
ADD . /code
WORKDIR /code
RUN pip install -r
requirements.txt
CMD python app.py
```

c. **vi app.py**

```
from flask import Flask
from redis import Redis
import os
import socket
app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    redis.incr('hits')
    return 'Hello World! I have been seen %s times. Run on %s.' %
(redis.get('hits'), socket.gethostname())

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

d. **vi requirements.txt**

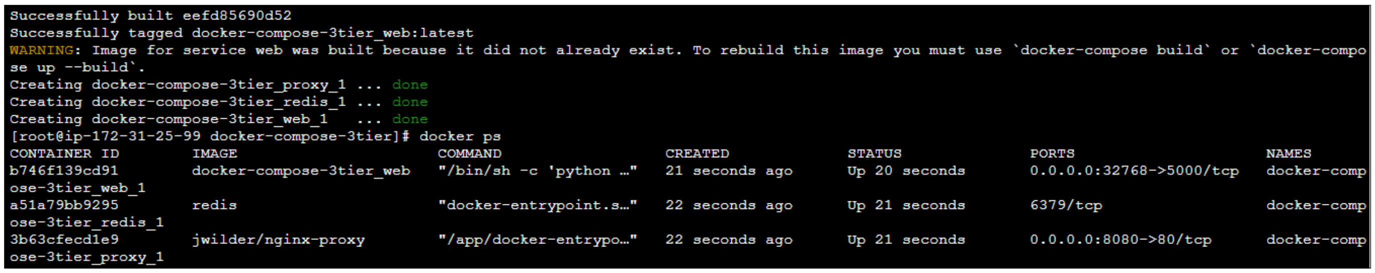
```
flask
redis
```

4. Create the multi-tier application using docker compose:

```
docker-compose up -d
```

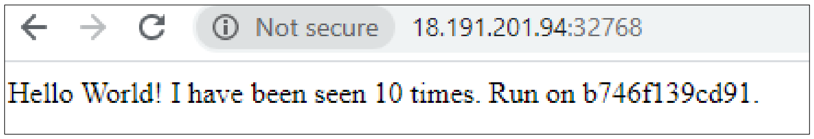
5. Once it is successful, you can verify using the following command:

```
docker-compose ps
```

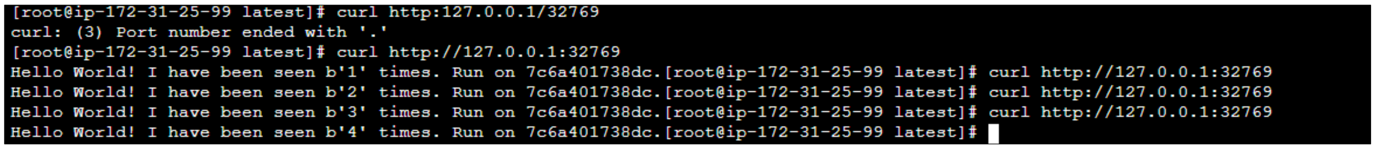


```
Successfully built eefd85690d52
Successfully tagged docker-compose-3tier_web:latest
WARNING: Image for service web was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compo
se up --build`.
Creating docker-compose-3tier_proxy_1 ... done
Creating docker-compose-3tier_redis_1 ... done
Creating docker-compose-3tier_web_1   ... done
[root@ip-172-31-25-99 docker-compose-3tier]# docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS              PORTS                     NAMES
b746f139cd91        docker-compose-3tier_web   "/bin/sh -c 'python …"   21 seconds ago      Up 20 seconds       0.0.0.0:32768->5000/tcp   docker-comp
ose-3tier_web_1
a51a79bb9295        redis                      "docker-entrypoint.s…"   22 seconds ago      Up 21 seconds       6379/tcp                  docker-comp
ose-3tier_redis_1
3b63cfecd1e9        jwilder/nginx-proxy        "/app/docker-entrypo…"   22 seconds ago      Up 21 seconds       0.0.0.0:8080->80/tcp      docker-comp
ose-3tier_proxy_1
```

6. Open the necessary ports that are visible from the ps command on your EC2 security group. Browse your instance ip with respective port number:



Hello World! I have been seen 10 times. Run on b746f139cd91.



```
[root@ip-172-31-25-99 latest]# curl http:127.0.0.1/32769
curl: (3) Port number ended with '.'
[root@ip-172-31-25-99 latest]# curl http://127.0.0.1:32769
Hello World! I have been seen b'1' times. Run on 7c6a401738dc.[root@ip-172-31-25-99 latest]# curl http://127.0.0.1:32769
Hello World! I have been seen b'2' times. Run on 7c6a401738dc.[root@ip-172-31-25-99 latest]# curl http://127.0.0.1:32769
Hello World! I have been seen b'3' times. Run on 7c6a401738dc.[root@ip-172-31-25-99 latest]# curl http://127.0.0.1:32769
Hello World! I have been seen b'4' times. Run on 7c6a401738dc.[root@ip-172-31-25-99 latest]#
```

7. To stop the application:

```
docker-compose down.
```

8. Now you can delete local image and try to pull your image from dockerhub.

1.4 *Troubleshooting*

S. No	Problem	Solution
1	Docker compose installation failed	Make sure that you made it executable, if the problem still persists install command completion mentioned in the installation doc
2	Docker compose build failed	Make sure that all files syntax is correctly given.

1.5 *Supporting References*

Refer the below links for additional information:

1. https://docs.docker.com/compose/#:~:text=Compose%20is%20a%20tool%20for,the%20services%20from%20your%20configuration.&text=Run%20docker%2Dcompose%20up%20and,and%20runs%20your%20entire%20app.
2. https://docs.tibco.com/pub/mash-local/4.1.1/doc/html/docker/GUID-BD850566-5B79-4915-987E-430FC38DAAE4.html
3. https://docs.docker.com/compose/install/