Docker Task - 2

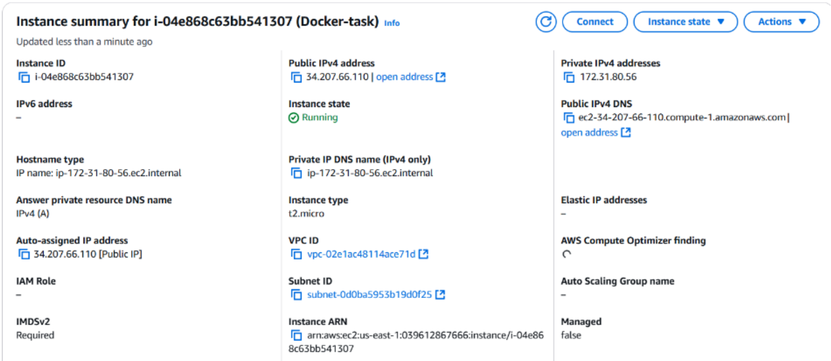
TASKS

Work Flow:

- Create an EC2 instance with the help of AWS Management Console with linux OS of required configuration and also update an security group with port number 80 (http).
- Now, Connect an EC2 instance with an help of Windows Terminal or Gitbash or Vbox.
- To connect an EC2 instance the command is:
    - ssh -i “**key_file**” ec2-user@“**Public_IP_address**”

Key_file --- Key file of the instance with the extension .pem

Public_IP_address --- Public IP address of the instance.



1. **Create a custom docker image for nginx and deploy it using docker compose, where the volume bind mount should be at /var/opt/nginx location. Push the created custom docker image to your docker-hub.**

Step 1: Install an Docker & Docker Compose in an EC2 instance

Install An Docker :

✓ To install an docker in linux machine, the command is:

- **sudo yum install docker**

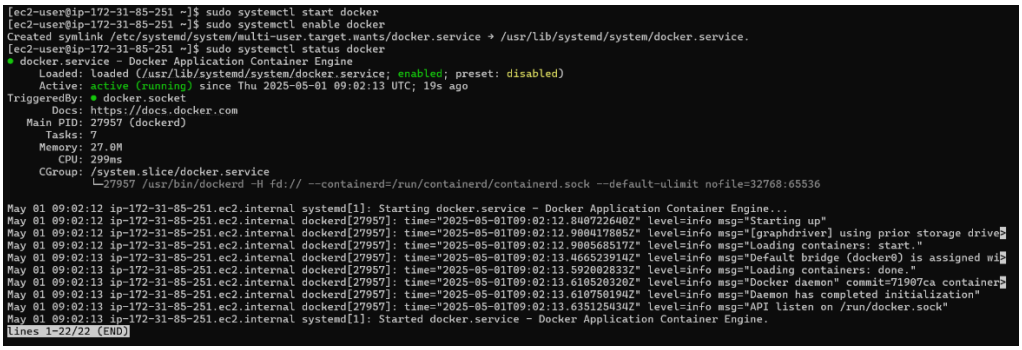
```
[ec2-user@ip-172-31-85-251 ~]$ sudo yum install docker
Amazon Linux 2023 Kernel Livepatch repository                                   133 kB/s |  16 kB     00:00
Dependencies resolved.
=====================================================================================================================
 Package                       Architecture        Version                             Repository              Size
=====================================================================================================================
Installing:
 docker                        x86_64              25.0.8-1.amzn2023.0.3               amazonlinux             45 M
Installing dependencies:
 container-selinux             noarch              3:2.233.0-1.amzn2023                amazonlinux             55 k
 containerd                    x86_64              1.7.27-1.amzn2023.0.2               amazonlinux             37 M
 iptables-libs                 x86_64              1.8.8-3.amzn2023.0.2                amazonlinux            401 k
 iptables-nft                  x86_64              1.8.8-3.amzn2023.0.2                amazonlinux            183 k
 libcgroup                     x86_64              3.0-1.amzn2023.0.1                  amazonlinux             75 k
 libnetfilter_conntrack        x86_64              1.0.8-2.amzn2023.0.2                amazonlinux             58 k
 libnfnetlink                  x86_64              1.0.1-19.amzn2023.0.2               amazonlinux             30 k
 libnftnl                      x86_64              1.2.2-2.amzn2023.0.2                amazonlinux             84 k
 pigz                          x86_64              2.5-1.amzn2023.0.3                  amazonlinux             83 k
 runc                          x86_64              1.2.4-1.amzn2023.0.1                amazonlinux            3.4 M

Transaction Summary
=====================================================================================================================
Install  11 Packages
```

✓ To start and enable an docker service, The command is:

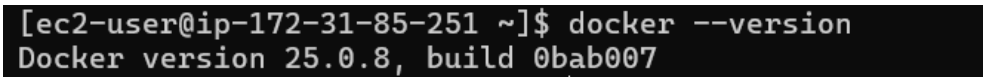
- **sudo systemctl start docker**
- **sudo systemctl enable docker**

- ✓ To check the status of the docker service, The command is:
    - **sudo systemctl status docker**



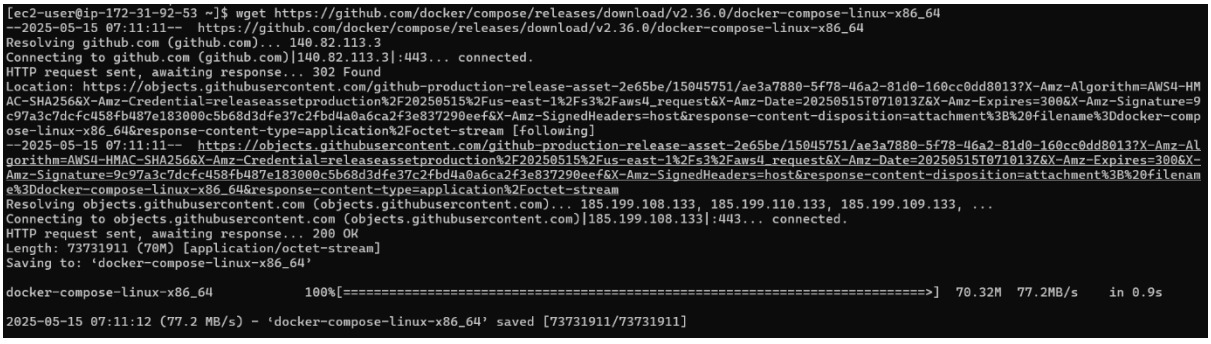
- ✓ To add an ec2-user to docker group, the command is:
    - **sudo usermod -aG docker ec2-user**

- ✓ To check an version of the docker and to verify an installation, the command is:
    - **docker --version**



Install an Docker Compose:

- ✓ Now, We have to install an docker-compose file we can get the docker compose file in github, we can copy the link and paste in an below command formate, The command is:
    - **wget <docker-compose-file>**



- ✓ To add some permission to that docker-compose file, The command is:
    - **Chmod +x <docker-compose-file>**

- ✓ And now copy that file to the default location of the docker compose file, The command is:
    - **Cp <docker-compose-file> /usr/local/bin/<docker-compose-file>**

- ✓ To check an version of docker-compose file, The command is:
    - **docker-compose --version**

```
[ec2-user@ip-172-31-92-53 ~]$ sudo su
[root@ip-172-31-92-53 ec2-user]# cp docker-compose-linux-x86_64 /usr/local/bin/docker-compose
[root@ip-172-31-92-53 ec2-user]# docker-compose --version
Docker Compose version v2.36.0
```

Step 2: Create a project directory with necessary files.

- ✓ Create one directory, inside that directory we are going to create an html file, so to create an one directory, The command is:
  - **mkdir nginx-project**

- ✓ To move inside that directory, The command is:
  - **cd nginx-project**

- ✓ Inside nginx-project directory, create an another directory which was named as html, so to create an one directory, The command is:
  - **mkdir html**

```
[ec2-user@ip-172-31-80-56 ~]$ mkdir nginx-project
[ec2-user@ip-172-31-80-56 ~]$ ls
docker-compose-linux-x86_64  nginx-project
[ec2-user@ip-172-31-80-56 ~]$ cd nginx-project
[ec2-user@ip-172-31-80-56 nginx-project]$ mkdir html
[ec2-user@ip-172-31-80-56 nginx-project]$ ls
html
[ec2-user@ip-172-31-80-56 nginx-project]$
```

- ✓ Now, create an html file inside html directory which should be named with “index.html”, The command is:
  - **echo ‘<h1>Hello From Custom NGNIX!</h1>’ > html/index.html**

```
[ec2-user@ip-172-31-80-56 nginx-project]$ echo '<h1>Hello From Custom NGNIX!</h1>' > html/index.html
[ec2-user@ip-172-31-80-56 nginx-project]$ cd html
[ec2-user@ip-172-31-80-56 html]$ ls
index.html
[ec2-user@ip-172-31-80-56 html]$ cat index.html
<h1>Hello From Custom NGNIX!</h1>
[ec2-user@ip-172-31-80-56 html]$
```

Step 3: Build a custom Nginx Docker image.

- ✓ Create one Dockerfile and write an command to perform an required task, The command is:
  - **touch Dockerfile**

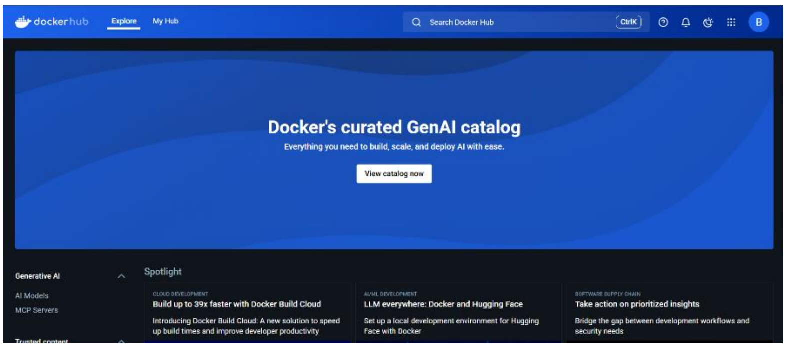
- ✓ Open that docker file and write an command to perform an required task which as given below, The command is:
  - **vi Dockerfile**

```
FROM nginx:latest
COPY html /usr/share/nginx/html
~
~
```

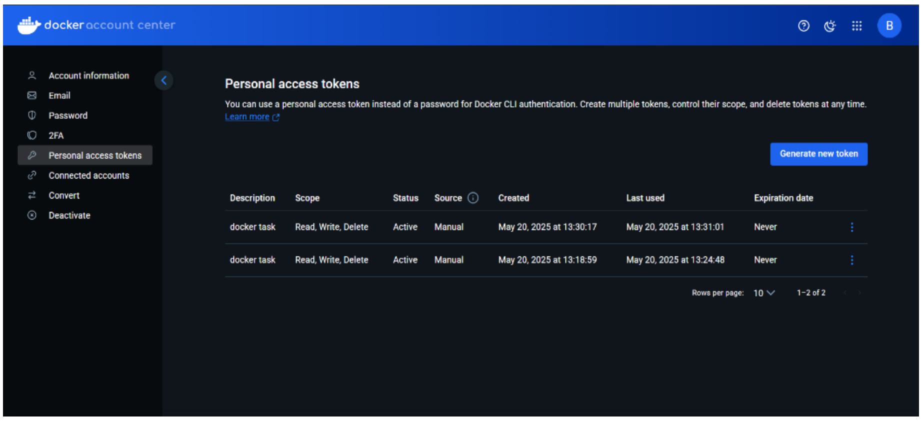
DOCKERFILE LINES	EXPLAINATION
FROM nginx:latest	Specifies the **base image** to use for the container.
COPY html /usr/share/nginx/html	Copies the entire html folder from your local project directory into the container's default NGINX web root.

Step 4: Create an Docker Hub Account.

- ✓ Before building an docker image we have to create an personal account in docker hub, then only we can able to login to docker hub and can able to push our custom image to docker hub publicly.

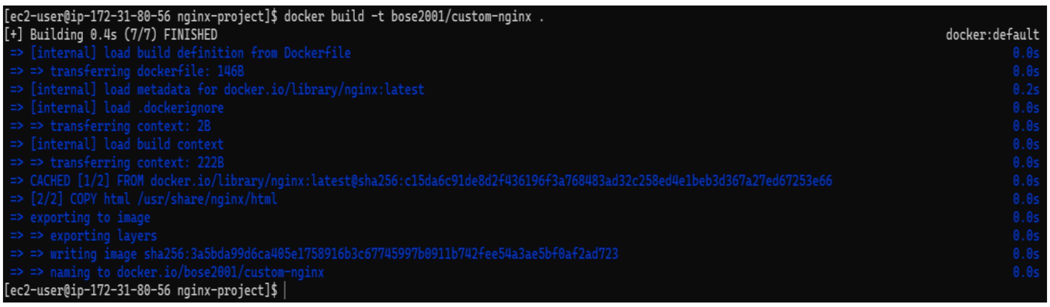


- ✓ And also generate an personal access token for your account ,because while login to your account we have to provide an token as an password.



Step 5: Build a custom Nginx Docker image.

- ✓ To build an custom docker image for an nginx which was create by you, The command is:
  - **docker build -t bose2001/custom-nginx**



- ✓ This command creates a custom Nginx Docker image by packaging your website files inside the official Nginx image, enabling you to deploy your custom content in a containerized environment.

Step 6: Push the image to Docker Hub.

- ✓ First we have to login to our docker hub account then only we can able to push our custom image to an docker hub.

- ✓ To Login to our docker hub account from linux machine, The command is:
  - **docker login**

```
[ec2-user@ip-172-31-80-56 nginx-project]$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to
create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using
 SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: bose2001
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-80-56 nginx-project]$
```

- ✓ Now we can provide our login details, and then you have logged in your docker hub account.

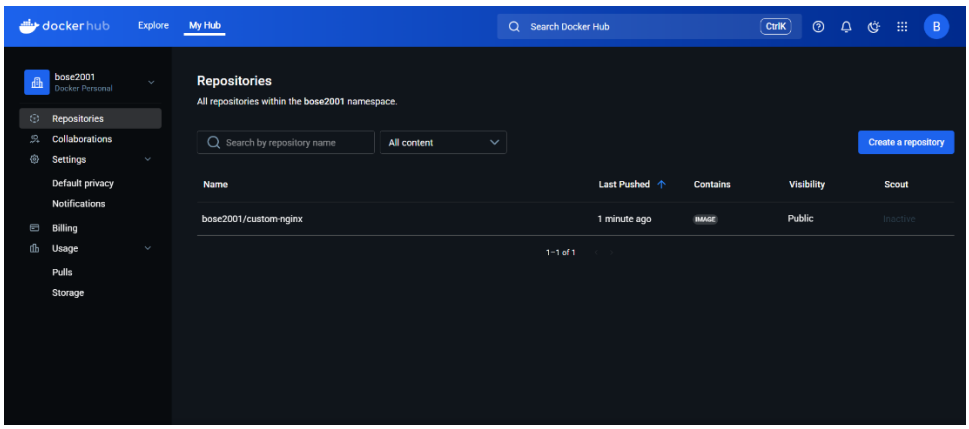
- ✓ Now you have to push your custom docker image to your Docker hub, The command is:

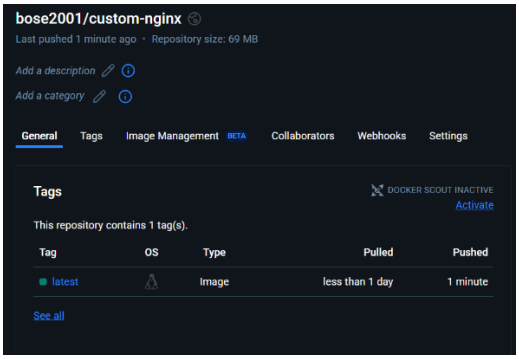
  - **docker push bose2001/custom-nginx**

```
[ec2-user@ip-172-31-80-56 nginx-project]$ docker push bose2001/custom-nginx
Using default tag: latest
The push refers to repository [docker.io/bose2001/custom-nginx]
e7c06c7f7ab8: Pushed
8030dd26ec5d: Mounted from library/nginx
d842334334437: Mounted from library/nginx
f8455d4eb3ff: Mounted from library/nginx
286733b13b0f: Mounted from library/nginx
46a24b5c31d8: Mounted from library/nginx
84accda66bf0: Mounted from library/nginx
6c4c763d22d0: Mounted from library/nginx
latest: digest: sha256:b14afda01d506d3ef3e0844e4eacac818dca7bc4a78ea82dbcf3db5854212501 size: 1985
[ec2-user@ip-172-31-80-56 nginx-project]$
```

- ✓ To Verify the upload, Go to https://hub.docker.com/repositories

- ✓ You should see the custom-nginx image under your Docker Hub account.





Step 7: Create a Docker Compose file .

- ✓ Create one Docker compose file and write an command to perform an required task, The command is:
  - **touch docker-compose.yml**

- ✓ Open that docker compose file and write an command to perform an required task which as given below, The command is:
  - **vi docker-compose.yml**

```
version: '3'
services:
  web:
    image: bose2001/custom-nginx
    ports:
      - "80:80"
    volumes:
      - /var/opt/nginx:/usr/share/nginx/html
~
~
~
```

LINE	DESCRIPTION
version: '3'	Specifies the Docker Compose file format version (v3).
services:	Defines the list of services (containers) to run.
web:	Names the service "web" (you can choose any name).
image: yourdockerhubusername/ custom-nginx	Uses the custom NGINX image you pushed to Docker Hub.
ports:	Maps container ports to the host.
- "80:80"	Maps port 80 on the host to port 80 in the container (for HTTP access).
volumes:	Defines file system mount points between host and container.
- /var/opt/nginx:/usr/share/ nginx/html	Binds the host directory /var/opt/nginx to NGINX's web root in the container.

Step 8: Create the bind mount directory.

- ✓ Create an directory to mount the custom directory and -p use to create an parent directory if does not exist, The command is:
  - **sudo mkdir -p /var/opt/nginx**

- ✓ Open that html file and write an html code to display in an web browser, The command is:
  - **sudo vi /var/opt/nginx/index.html**

- ✓ The file is now created at /var/opt/nginx/index.html with your custom HTML content.

- ✓ We going to write one custom html file.

```
<h1>Hello from /var/opt/nginx bind mount!</h1>
~
```

Step 9: Deploy the application with Docker Compose.

- ✓ Now, after creating all three file such as html file, docker file and docker compose file. Now build and run that docker compose file to build and run the docker compose file, The command is:
  - **docker-compose up -d --build**

```
[ec2-user@ip-172-31-80-56 nginx-project]$ docker-compose up -d --build
WARN[0000] /home/ec2-user/nginx-project/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential con
fusion
[+] Running 1/1
 ✔Container nginx-project-web-1  Started                                                                                                    0.7s
[ec2-user@ip-172-31-80-56 nginx-project]$
```

Step 10: Access the application via browser.

- ✓ Once the build is gets succeed , go to aws console and copy the instance Public DNS and paste it into the browser, where you can see the content which was written in the html document and also mount with an blind mount.

Hello from /var/opt/nginx bind mount!

********** **TASK COMPLETED** **********