**Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers**

## Theory:

## What is Docker?

Docker is a platform that allows developers to build, package, and deploy applications in lightweight, portable containers. These containers include everything needed to run an application, such as code, runtime, system tools, libraries, and dependencies.

## Containerization Technology

Containers are isolated environments where applications run independently. Unlike traditional virtualization, which requires separate operating systems for each application, containers share the host OS kernel, making them faster and more efficient.

## Docker Architecture

Docker follows a client-server architecture consisting of the following key components:

**1. Docker Client**

- It is the command-line interface (CLI) that allows users to interact with Docker.
- Commands such as docker run, docker build, and docker stop are executed through the client.

**2. Docker Daemon (dockerd)**

- It runs in the background and manages Docker containers, images, volumes, and networks.
- It listens for requests from the Docker Client and executes commands.

**3. Docker Images**

- A Docker image is a read-only template containing the application code, libraries, and dependencies.

- Images are created using Dockerfiles, which define the steps to build an image.
- Images are stored in Docker Hub or private repositories.

**4. Docker Containers**

- A container is an instance of a Docker image running as an isolated process on a host machine.
- Containers are lightweight, portable, and can be started, stopped, or removed as needed.

**5. Docker Registry**

- It is a storage system for Docker images.
- The public registry, Docker Hub, provides access to a vast collection of pre-built images.
- Users can also create private registries for security and control.

## Docker Container Life Cycle

The **life cycle of a container** follows these steps:

1. **Create** – A container is created from an image using the docker create command.
2. **Start** – The container starts running using the docker start command.
3. **Run** – A new container can be started directly using docker run.
4. **Pause/Unpause** – Containers can be temporarily paused and resumed.
5. **Stop** – The container can be stopped using docker stop.
6. **Restart** – A stopped container can be restarted.
7. **Kill** – A container can be forcefully stopped using docker kill.
8. **Remove** – Containers that are no longer needed can be deleted using docker rm.

## Benefits of Docker

**1. Portability**

- Containers can run on any platform that supports Docker.
- Applications behave consistently across different environments.

**2. Efficiency**

- Containers share the host OS kernel, reducing overhead and improving performance.
- They consume fewer resources compared to virtual machines.

**3. Isolation**

- Each container runs in its own isolated environment, preventing dependency conflicts.

**4. Scalability**

- Applications can be scaled up quickly by launching multiple containers.
- Docker enables automatic load balancing in large-scale deployments.

**5. Consistency**

- Ensures that the application runs the same way in development, testing, and production.
- Eliminates the "works on my machine" problem.

## Docker Engine:

At the core of Docker is the Docker Engine, which is responsible for building, running, and managing containers. It consists of the Docker daemon, which manages containers, images, networks, and volumes, and the Docker client, which allows users to interact with the daemon through the Docker API.

## Docker Images:

Docker images are read-only templates used to create containers. They contain the application code, runtime, libraries, dependencies, and other files needed to run the application. Images are built using Dockerfiles, which are text files that define the steps needed to create the image.

## **OUTPUT**:-

```
C:\Users\202>docker run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
8a1e25ce7c4f: Pull complete
8ab039a68e51: Pull complete
2b12a49dcfb9: Pull complete
cdf9868f47ac: Pull complete
e73ea5d3136b: Pull complete
890ad32c613f: Pull complete
4f4fb700ef54: Pull complete
ba517b76f92b: Pull complete
Digest: sha256:7dd707032d90c6eaafd566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Downloaded newer image for redis:latest
1:C 13 Mar 2024 03:19:03.928 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 13 Mar 2024 03:19:03.928 * Redis version=7.2.4, bits=64, commit=00000000, mod
1:C 13 Mar 2024 03:19:03.928 # Warning: no config file specified, using the defau
s.conf
1:M 13 Mar 2024 03:19:03.929 * monotonic clock: POSIX clock_gettime
1:M 13 Mar 2024 03:19:03.929 * Running mode=standalone, port=6379.
1:M 13 Mar 2024 03:19:03.929 * Server initialized
1:M 13 Mar 2024 03:19:03.929 * Ready to accept connections tcp
1:signal-handler (1710300105) Received SIGINT scheduling shutdown...
1:M 13 Mar 2024 03:21:45.877 * User requested shutdown...
1:M 13 Mar 2024 03:21:45.877 * Saving the final RDB snapshot before exiting.
1:M 13 Mar 2024 03:21:45.887 * DB saved on disk
1:M 13 Mar 2024 03:21:45.887 # Redis is now ready to exit, bye bye...
```

```
C:\Users\202>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
redis        latest    170a1e90f843   2 months ago    138MB
```

```
C:\Users\202>docker pull redis
Using default tag: latest
latest: Pulling from library/redis
Digest: sha256:7dd707032d90c6eaafd566f62a00f5b0116ae08fd7d6cbbb0f311b82b47171a2
Status: Image is up to date for redis:latest
docker.io/library/redis:latest
```

```
C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS          PORTS      NAMES
052aecb0ee88   redis     "docker-entrypoint.s…"   About a minute ago   Up 4 seconds    6379/tcp   container121
1c4472744083   redis     "docker-entrypoint.s…"   6 minutes ago        Up 10 seconds   6379/tcp   modest_herschel

C:\Users\202>
```

```
C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS          PORTS      NAMES
052aecb0ee88   redis     "docker-entrypoint.s…"   About a minute ago   Up 4 seconds    6379/tcp   container121
1c4472744083   redis     "docker-entrypoint.s…"   6 minutes ago        Up 10 seconds   6379/tcp   modest_herschel

C:\Users\202>
C:\Users\202>
C:\Users\202>docker stop 052aecb0ee88
052aecb0ee88

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
1c4472744083   redis     "docker-entrypoint.s…"   9 minutes ago   Up 2 minutes   6379/tcp   modest_herschel
```

```
C:\Users\202>docker start 052aecb0ee88
052aecb0ee88

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS         PORTS      NAMES
052aecb0ee88   redis     "docker-entrypoint.s…"   5 minutes ago    Up 8 seconds   6379/tcp   container121
1c4472744083   redis     "docker-entrypoint.s…"   10 minutes ago   Up 3 minutes   6379/tcp   modest_herschel
```

```
C:\Users\202>docker rm 052aecb0ee88
052aecb0ee88
```

```
C:\Users\202>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
redis        latest    170a1e90f843   2 months ago    138MB
```

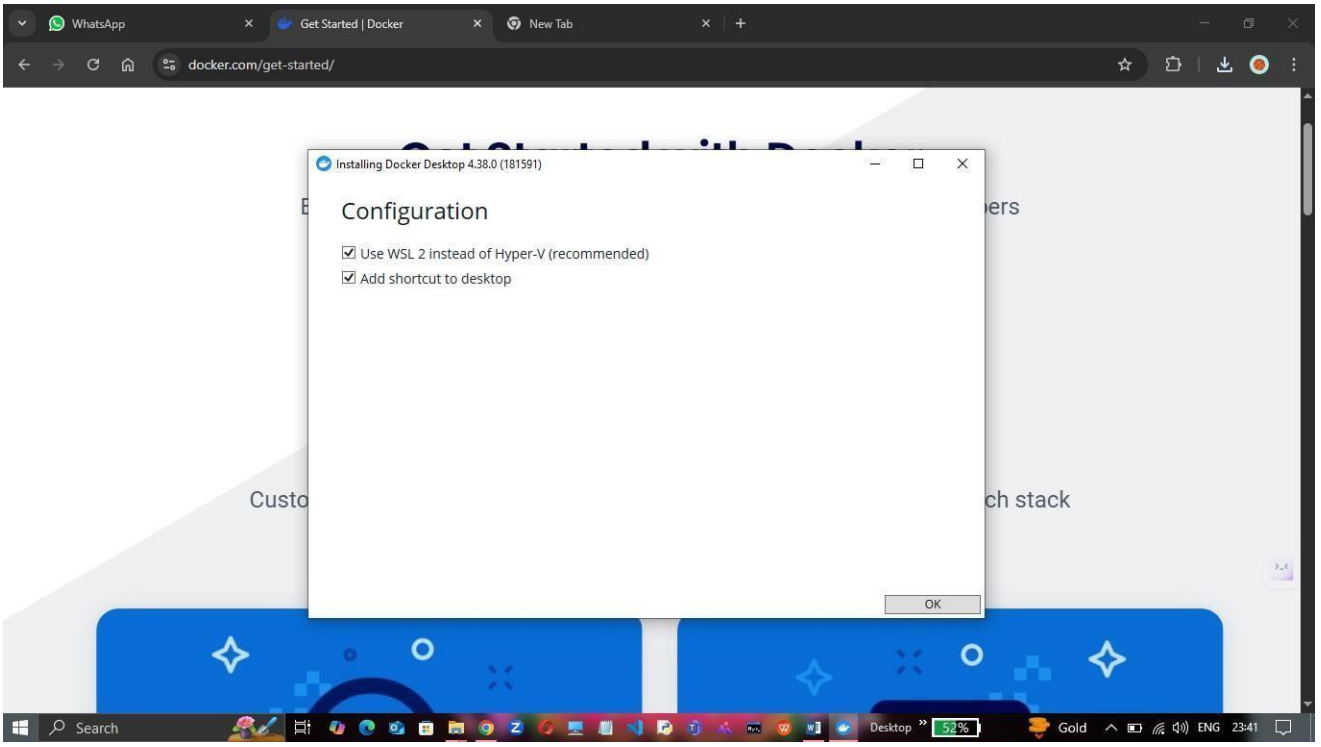
```
C:\Users\202>docker exec -d 1c4472744083 touch /tmp/execWorks

C:\Users\202>docker exec -it 1c4472744083 bash
root@1c4472744083:/data#
```

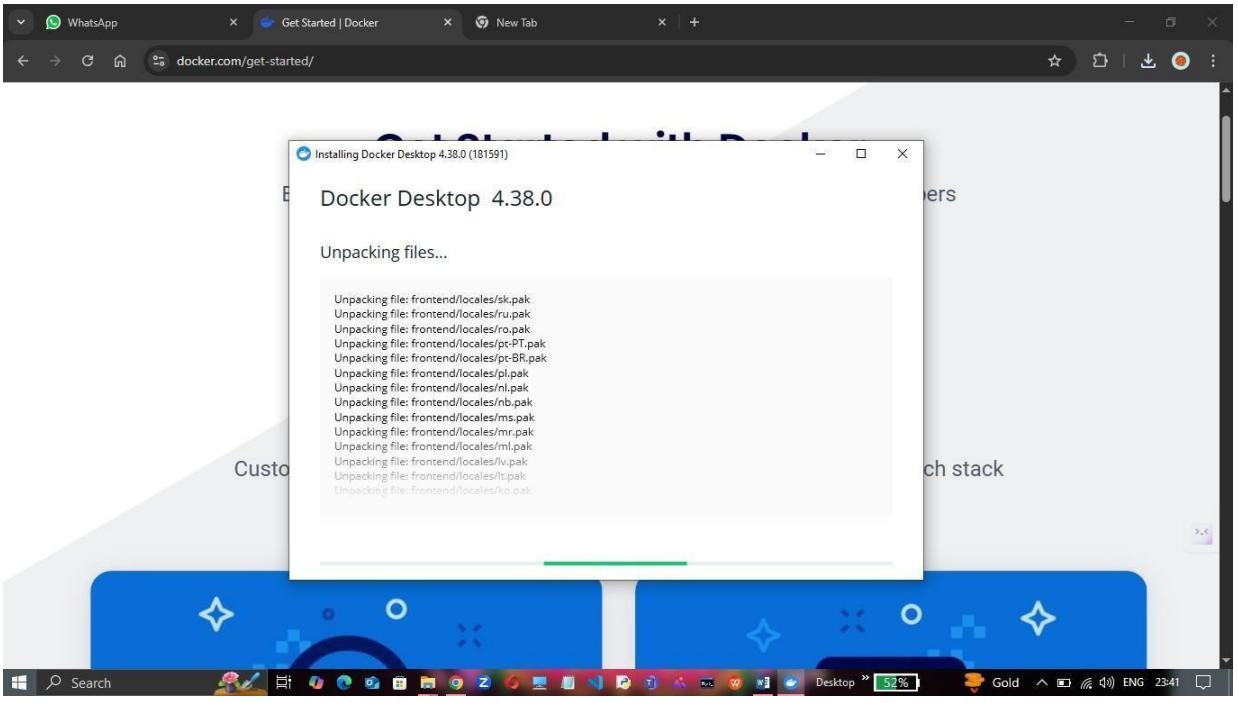
```
C:\Users\202>docker restart 1c4472744083
1c4472744083

C:\Users\202>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS         PORTS      NAMES
1c4472744083   redis     "docker-entrypoint.s…"   13 minutes ago   Up 3 seconds   6379/tcp   modest_herschel
```

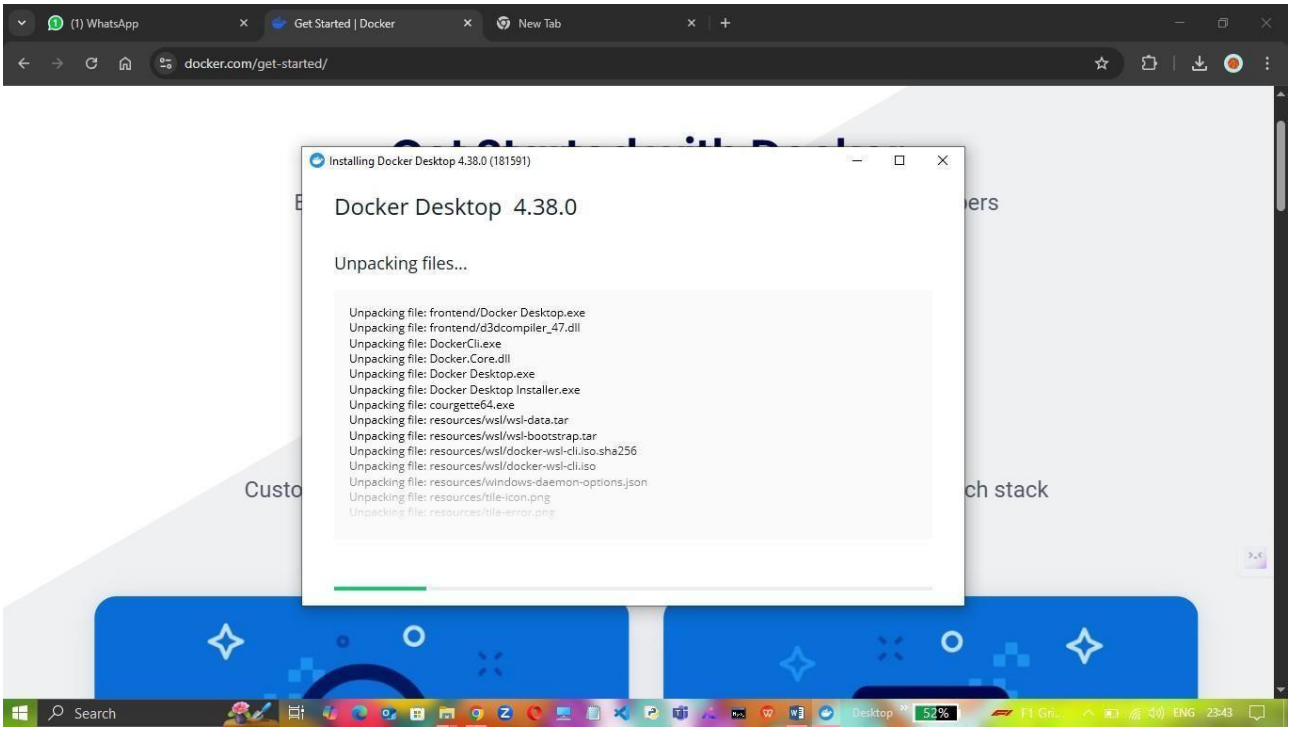
```
C:\Users\202>docker inspect 1c4472744083
[
    {
        "Id": "1c447274408314475b093dcbf93163064b819bdd9ad8378bb3a4fa847dc411d80",
        "Created": "2024-03-13T03:19:03.418741433Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 2112,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2024-03-13T03:32:13.750463204Z",
            "FinishedAt": "2024-03-13T03:32:13.145321277Z"
        },
        "Image": "sha256:170a1e90f8436daa6778aeea3926e716928826c215ca23a8dfd8055f663f9428",
        "ResolvConfPath": "/var/lib/docker/containers/1c447274408314475b093dcbf93163064b819bdd9a
```

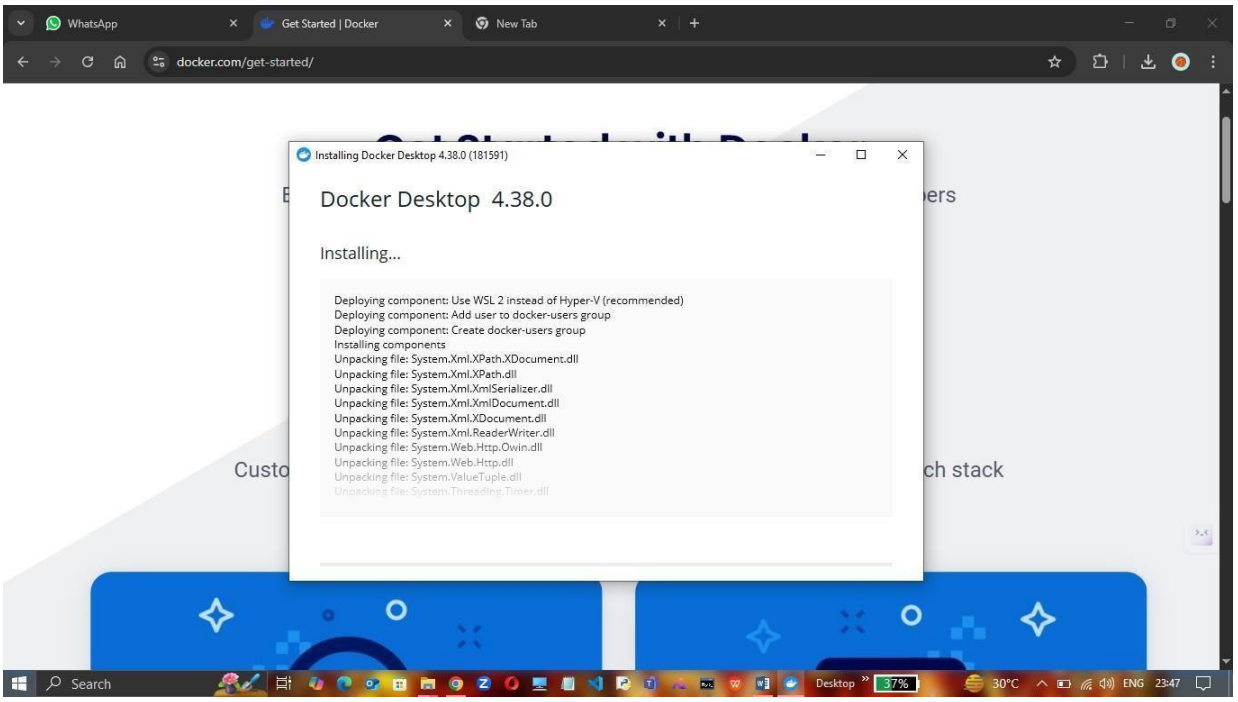
```
C:\Users\202>docker commit 1c4472744083 new_image_name:redis2
sha256:33e4284a7e92a4a1331555d01f6e078fc496e3a3ed8eb7f84f2678261ad07e83

C:\Users\202>docker images
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
new_image_name   redis2    33e4284a7e92   4 seconds ago    138MB
new_image_name   tag       61ab016507fa   36 seconds ago   138MB
redis            latest    170a1e90f843   2 months ago     138MB
```

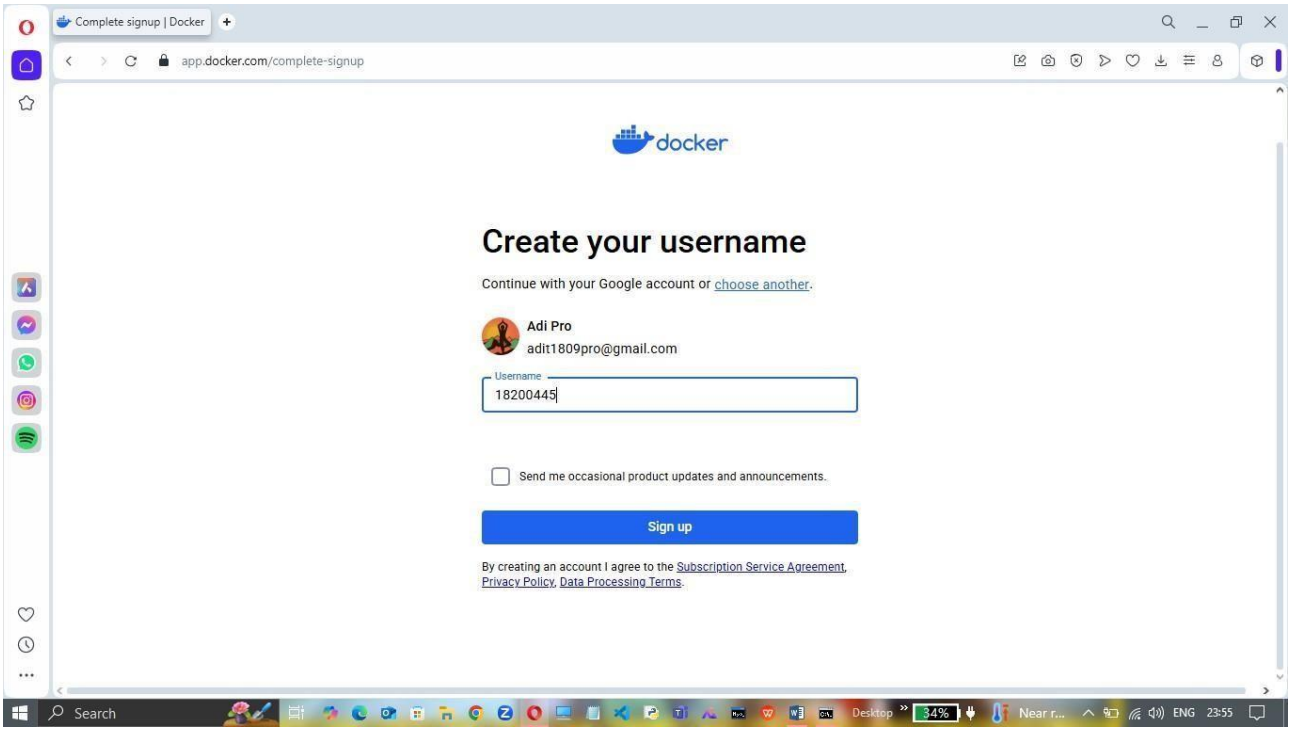
## SCREENSHOTS:



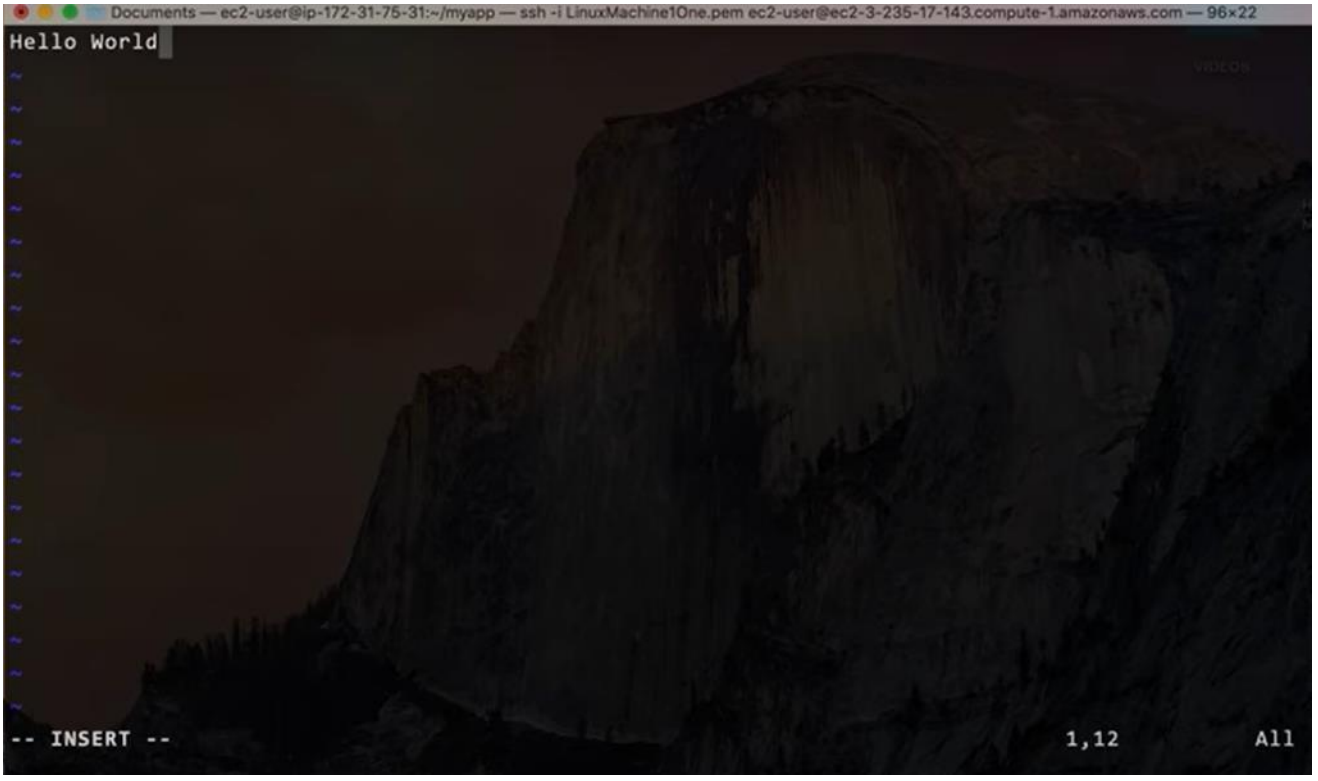



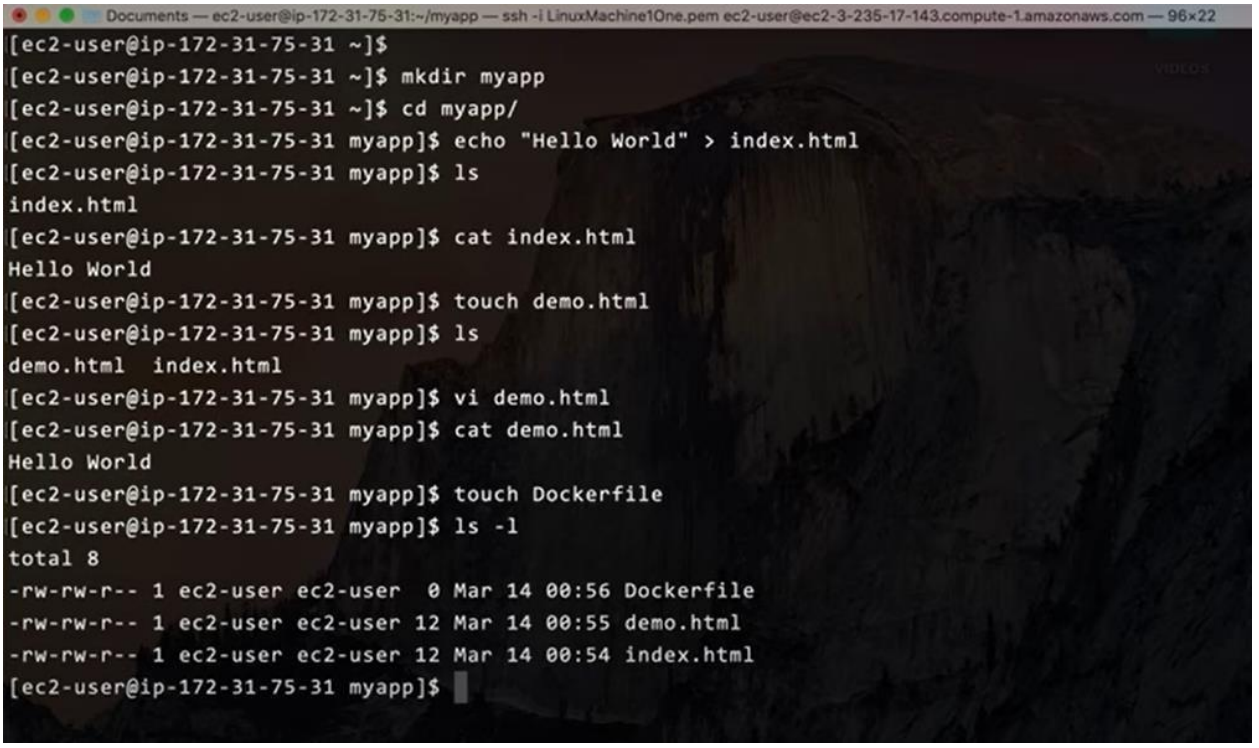

(1) WhatsApp
Get Started | Docker
New Tab
docker.com/get-started/
Installing Docker Desktop 4.38.0 (181591)
Docker Desktop 4.38.0
Unpacking files...
Unpacking file: frontend/Docker Desktop.exe
Unpacking file: frontend/d3dcompiler_47.dll
Unpacking file: DockerCli.exe
Unpacking file: Docker.Core.dll
Unpacking file: Docker Desktop.exe
Unpacking file: Docker Desktop Installer.exe
Unpacking file: courgette64.exe
Unpacking file: resources/wsl/wsl-data.tar
Unpacking file: resources/wsl/wsl-bootstrap.tar
Unpacking file: resources/wsl/docker-wsl-cli.iso.sha256
Unpacking file: resources/wsl/docker-wsl-cli.iso
Unpacking file: resources/windows-daemon-options.json
Unpacking file: resources/tile-icon.png
Search

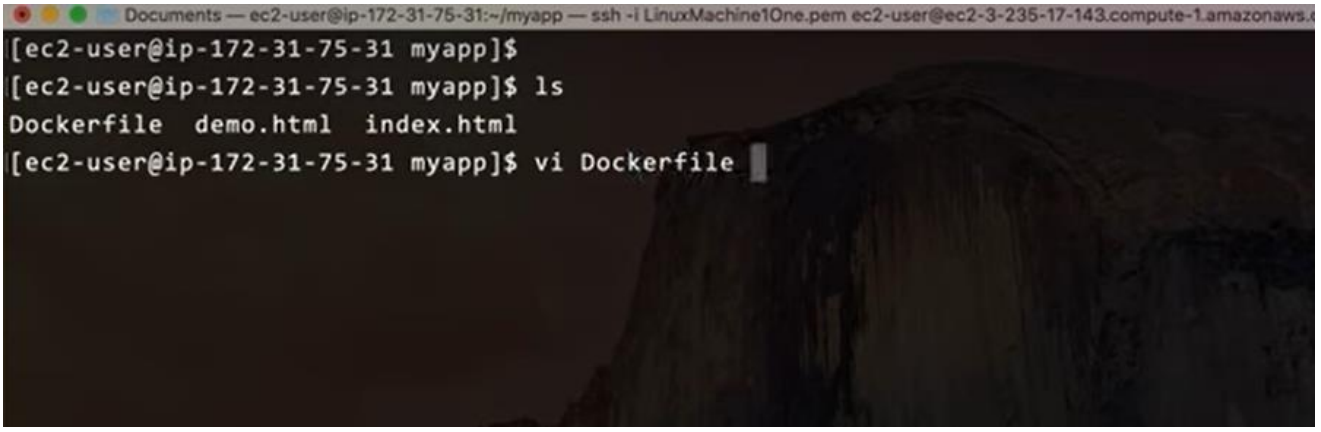

WhatsApp
Get Started | Docker
New Tab
docker.com/get-started/
Installing Docker Desktop 4.38.0 (181591)
Docker Desktop 4.38.0
Installing...
Deploying component: Use WSL 2 instead of Hyper-V (recommended)
Deploying component: Add user to docker-users group
Deploying component: Create docker-users group
Installing components
Unpacking file: System.Xml.XPath.XDocument.dll
Unpacking file: System.Xml.XPath.dll
Unpacking file: System.Xml.XmlSerializer.dll
Unpacking file: System.Xml.XmlDocument.dll
Unpacking file: System.Xml.XDocument.dll
Unpacking file: System.Xml.ReaderWriter.dll
Unpacking file: System.Web.Http.Owin.dll
Unpacking file: System.Web.Http.dll
Unpacking file: System.ValueTuple.dll
Search

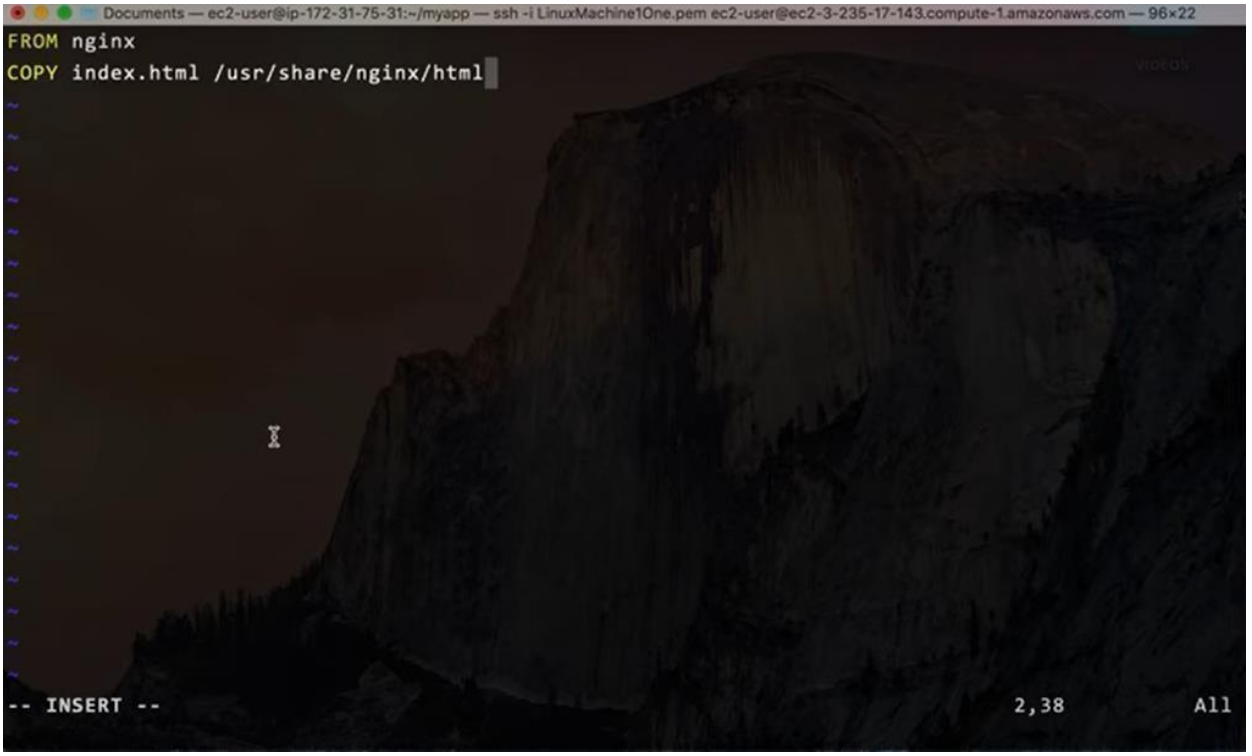

Complete signup | Docker
app.docker.com/complete-signup
docker
Create your username
Continue with your Google account or choose another.
Adi Pro
adit1809pro@gmail.com
Username
18200445
Send me occasional product updates and announcements.
Sign up
By creating an account I agree to the Subscription Service Agreement, Privacy Policy, Data Processing Terms.
Search


Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 ~]$
[ec2-user@ip-172-31-75-31 ~]$ mkdir myapp
[ec2-user@ip-172-31-75-31 ~]$ cd myapp/
[ec2-user@ip-172-31-75-31 myapp]$ echo "Hello World" > index.html
[ec2-user@ip-172-31-75-31 myapp]$ ls
index.html
[ec2-user@ip-172-31-75-31 myapp]$ cat index.html
Hello World
[ec2-user@ip-172-31-75-31 myapp]$ touch demo.html
[ec2-user@ip-172-31-75-31 myapp]$ ls
demo.html  index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi demo.html

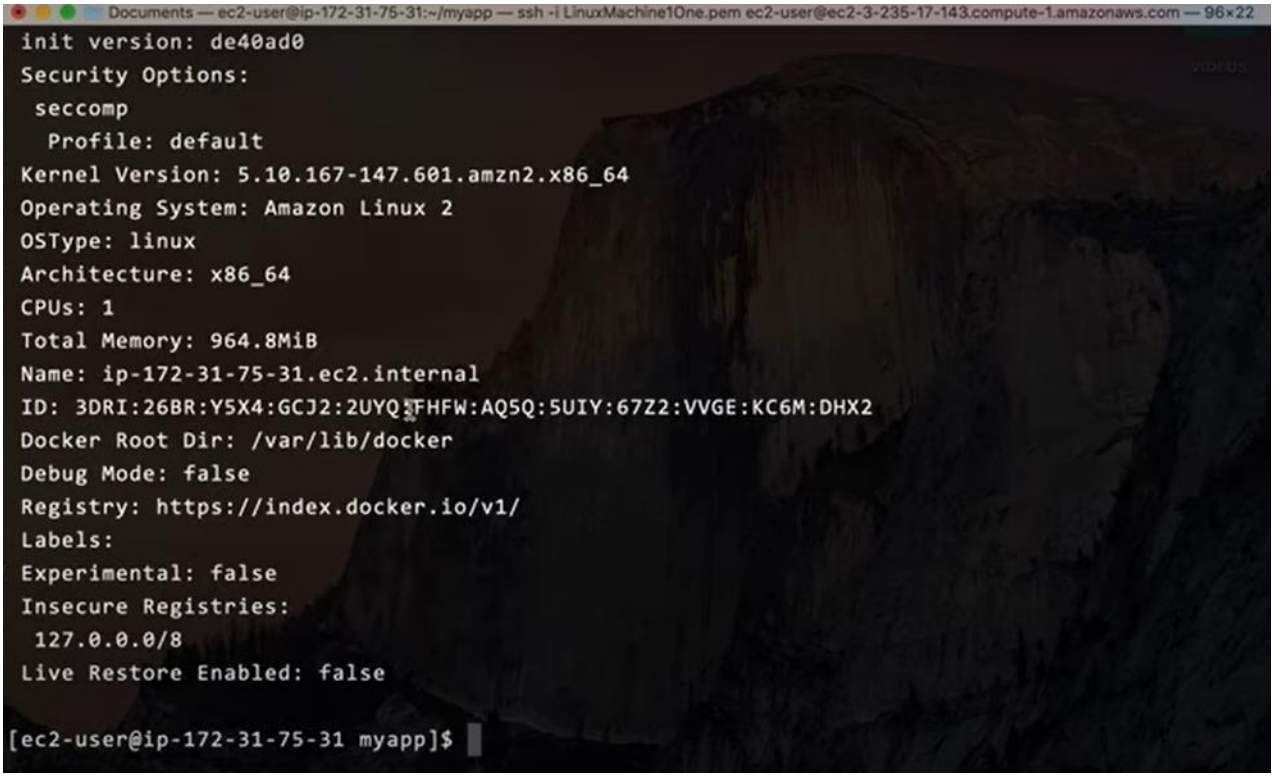

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
Hello World
-- INSERT --
1,12
All

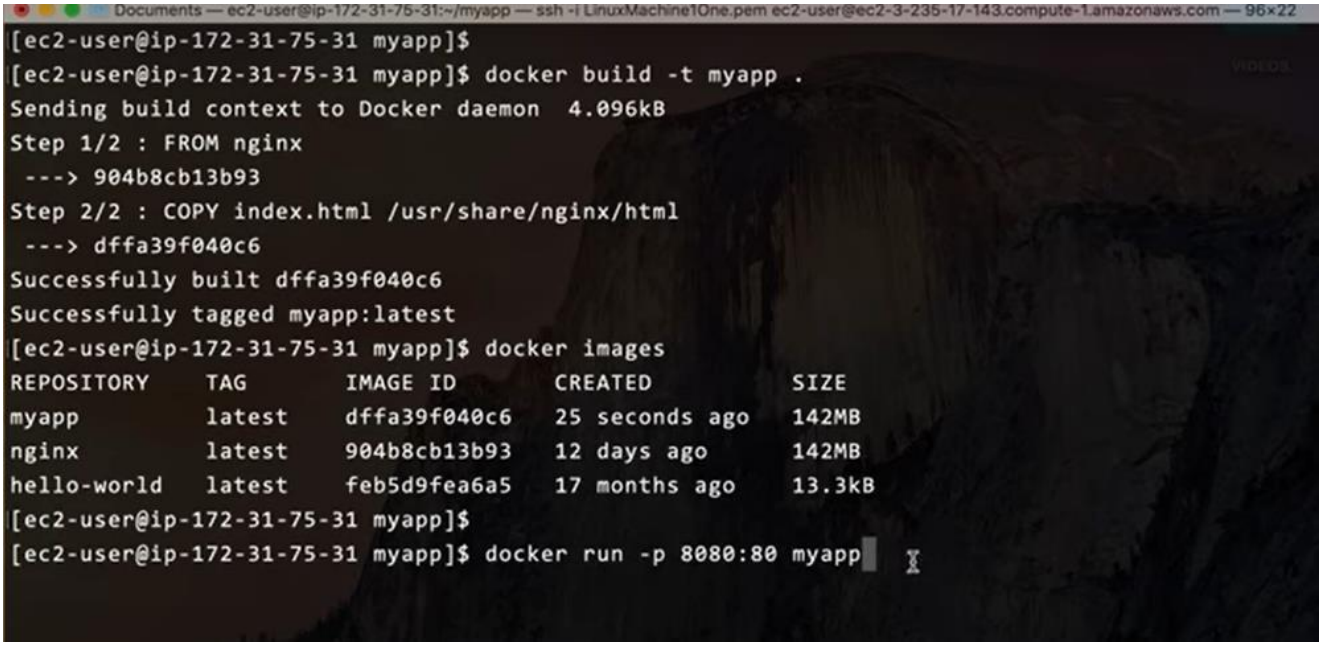

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 ~]$
[ec2-user@ip-172-31-75-31 ~]$ mkdir myapp
[ec2-user@ip-172-31-75-31 ~]$ cd myapp/
[ec2-user@ip-172-31-75-31 myapp]$ echo "Hello World" > index.html
[ec2-user@ip-172-31-75-31 myapp]$ ls
index.html
[ec2-user@ip-172-31-75-31 myapp]$ cat index.html
Hello World
[ec2-user@ip-172-31-75-31 myapp]$ touch demo.html
[ec2-user@ip-172-31-75-31 myapp]$ ls
demo.html index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi demo.html
[ec2-user@ip-172-31-75-31 myapp]$ cat demo.html
Hello World
[ec2-user@ip-172-31-75-31 myapp]$ touch Dockerfile
[ec2-user@ip-172-31-75-31 myapp]$ ls -l
total 8
-rw-rw-r-- 1 ec2-user ec2-user 0 Mar 14 00:56 Dockerfile
-rw-rw-r-- 1 ec2-user ec2-user 12 Mar 14 00:55 demo.html
-rw-rw-r-- 1 ec2-user ec2-user 12 Mar 14 00:54 index.html
[ec2-user@ip-172-31-75-31 myapp]$

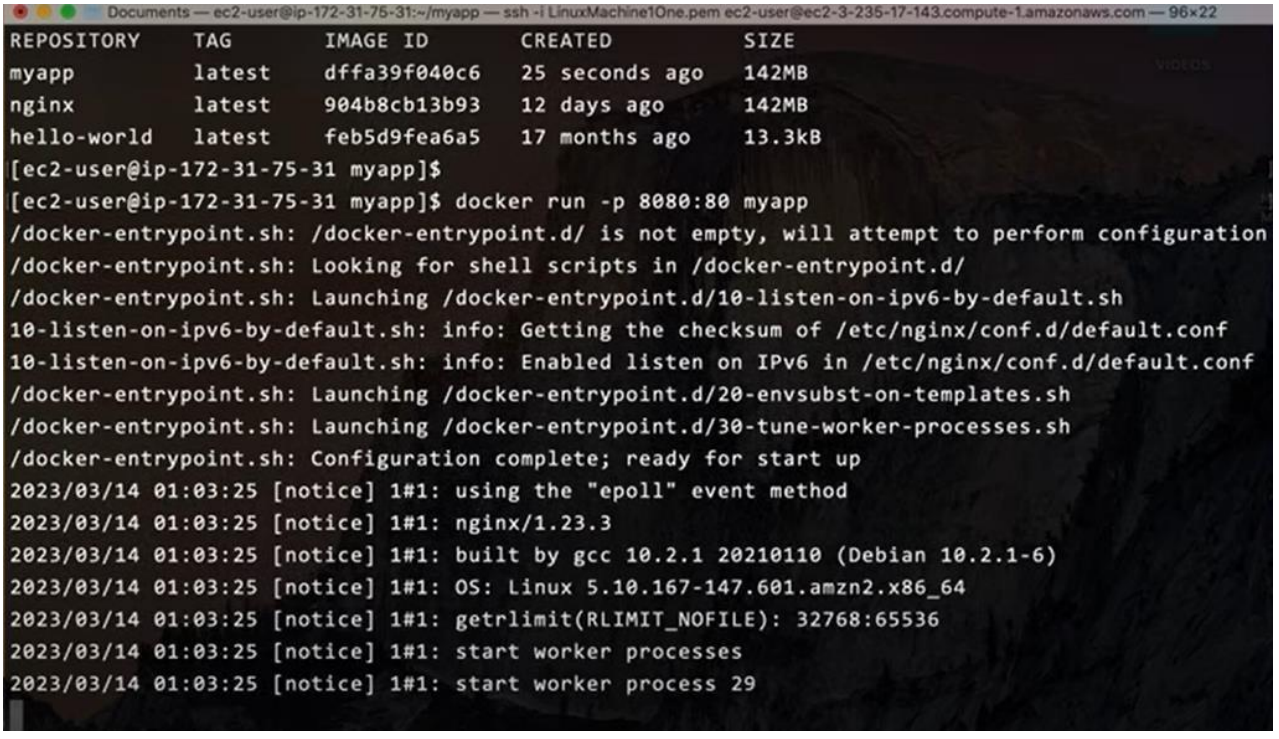

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ ls
Dockerfile  demo.html  index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi Dockerfile

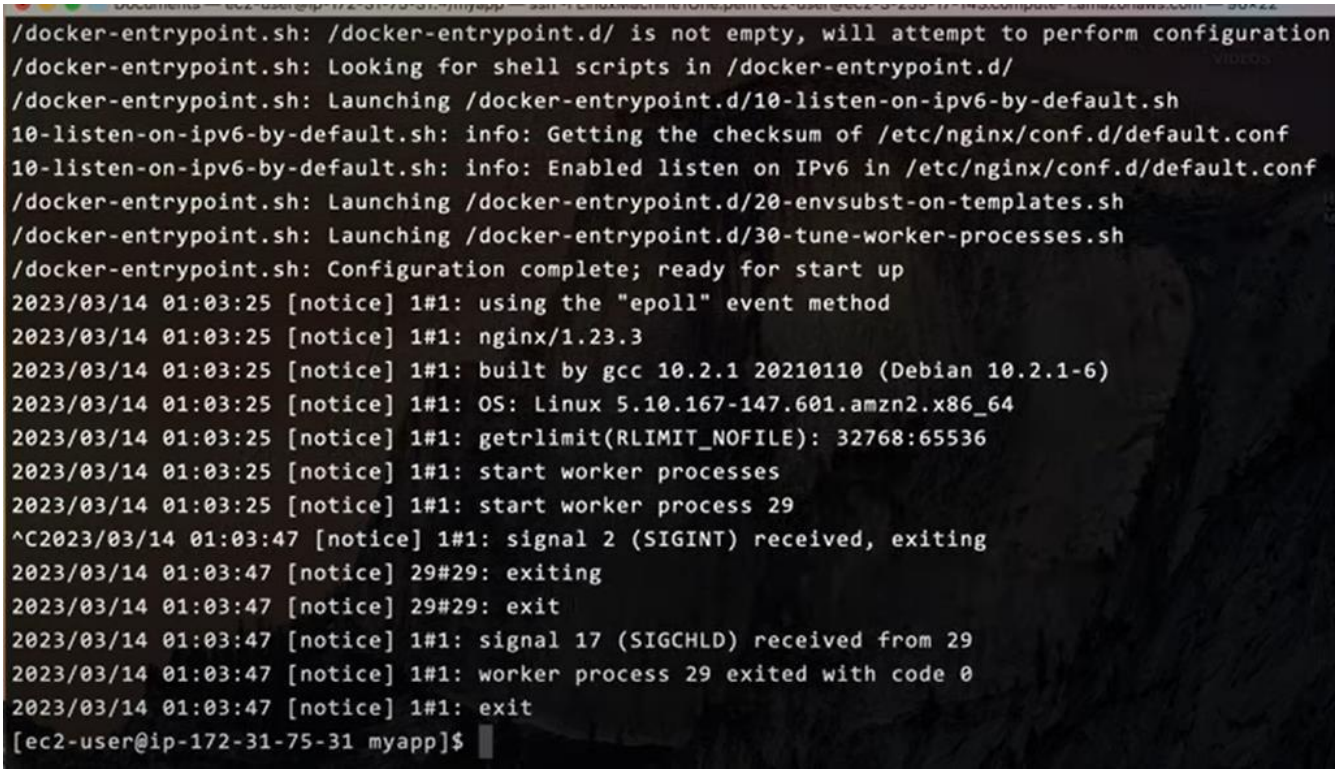

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
FROM nginx
COPY index.html /usr/share/nginx/html
-- INSERT --
2,38
All


Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ ls
Dockerfile  demo.html  index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi Dockerfile
[ec2-user@ip-172-31-75-31 myapp]$ cat Dockerfile
FROM nginx
COPY index.html /usr/share/nginx/html
[ec2-user@ip-172-31-75-31 myapp]$ docker info
Client:
 Context:    default
 Debug Mode: false
Server:
ERROR: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
errors pretty printing info
[ec2-user@ip-172-31-75-31 myapp]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-75-31 myapp]$

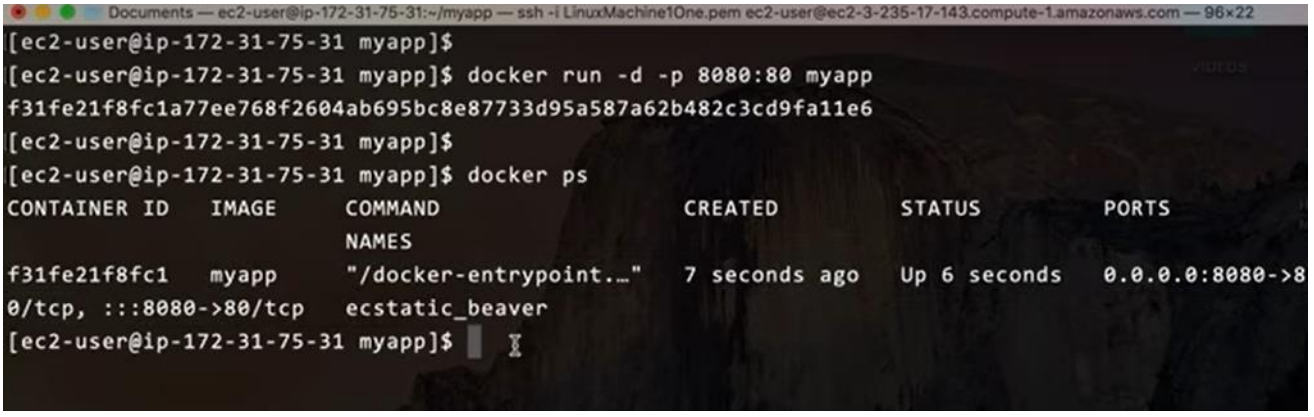

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 5.10.167-147.601.amzn2.x86_64
 Operating System: Amazon Linux 2
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 964.8MiB
 Name: ip-172-31-75-31.ec2.internal
 ID: 3DRI:26BR:Y5X4:GCJ2:2UYQ:FHFW:AQ5Q:5UIY:67Z2:VVGE:KC6M:DHX2
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false
[ec2-user@ip-172-31-75-31 myapp]$


Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ docker build -t myapp .
Sending build context to Docker daemon 4.096kB
Step 1/2 : FROM nginx
---> 904b8cb13b93
Step 2/2 : COPY index.html /usr/share/nginx/html
---> dffa39f040c6
Successfully built dffa39f040c6
Successfully tagged myapp:latest
[ec2-user@ip-172-31-75-31 myapp]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
myapp latest dffa39f040c6 25 seconds ago 142MB
nginx latest 904b8cb13b93 12 days ago 142MB
hello-world latest feb5d9fea6a5 17 months ago 13.3kB
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ docker run -p 8080:80 myapp


Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
REPOSITORY TAG IMAGE ID CREATED SIZE
myapp latest dffa39f040c6 25 seconds ago 142MB
nginx latest 904b8cb13b93 12 days ago 142MB
hello-world latest feb5d9fea6a5 17 months ago 13.3kB
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ docker run -p 8080:80 myapp
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/03/14 01:03:25 [notice] 1#1: using the "epoll" event method
2023/03/14 01:03:25 [notice] 1#1: nginx/1.23.3
2023/03/14 01:03:25 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/03/14 01:03:25 [notice] 1#1: OS: Linux 5.10.167-147.601.amzn2.x86_64
2023/03/14 01:03:25 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 32768:65536
2023/03/14 01:03:25 [notice] 1#1: start worker processes
2023/03/14 01:03:25 [notice] 1#1: start worker process 29



```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/03/14 01:03:25 [notice] 1#1: using the "epoll" event method
2023/03/14 01:03:25 [notice] 1#1: nginx/1.23.3
2023/03/14 01:03:25 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/03/14 01:03:25 [notice] 1#1: OS: Linux 5.10.167-147.601.amzn2.x86_64
2023/03/14 01:03:25 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 32768:65536
2023/03/14 01:03:25 [notice] 1#1: start worker processes
2023/03/14 01:03:25 [notice] 1#1: start worker process 29
^C2023/03/14 01:03:47 [notice] 1#1: signal 2 (SIGINT) received, exiting
2023/03/14 01:03:47 [notice] 29#29: exiting
2023/03/14 01:03:47 [notice] 29#29: exit
2023/03/14 01:03:47 [notice] 1#1: signal 17 (SIGCHLD) received from 29
2023/03/14 01:03:47 [notice] 1#1: worker process 29 exited with code 0
2023/03/14 01:03:47 [notice] 1#1: exit
[ec2-user@ip-172-31-75-31 myapp]$
```



```
Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ docker run -d -p 8080:80 myapp
f31fe21f8fc1a77ee768f2604ab695bc8e87733d95a587a62b482c3cd9fa11e6
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
                         NAMES
f31fe21f8fc1   myapp     "/docker-entrypoint.…"   7 seconds ago   Up 6 seconds   0.0.0.0:8080->8
0/tcp, :::8080->80/tcp   ecstatic_beaver
[ec2-user@ip-172-31-75-31 myapp]$
```

**<u>Conclusion</u>: Thus, we have successfully installed Docker and execute docker commands to manage images and interact with containers.**