Experiment no. 9

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

Theory:

1. Introduction to Docker and Containerization

Docker is a powerful open-source platform designed to automate the deployment, scaling, and management of applications using containerization. Containers are lightweight, standalone, and executable units that include everything needed to run an application—code, runtime, libraries, and system tools—ensuring consistent behavior across different environments.

Unlike traditional virtual machines (VMs), which bundle a full operating system along with the application, containers share the host OS kernel and isolate applications at the process level, making them faster and more efficient.

2. Why Docker?

Docker revolutionized DevOps and modern software development with its portability, simplicity, and speed. It allows developers to:

- Package once, run anywhere
- Isolate applications from each other
- Scale efficiently and deploy rapidly
- Ensure consistent environments from development to production

3. Docker Architecture

The Docker architecture follows a client-server model and includes several key components:

- Docker Client:

The user interface through which Docker commands are issued. Commands like docker run, docker build, and docker pull are sent to the Docker daemon.

- Docker Daemon (dockerd):

The background process that builds, runs, and manages containers. It listens to the Docker API and handles container operations.

- Docker Images:

Read-only templates used to create containers. An image can be based on another image and can include additional layers like libraries or application code.

- Docker Containers:

Running instances of Docker images. They are isolated but share the same kernel with the host and are lightweight.

- Docker Registry:

A centralized repository (like Docker Hub) from where Docker images can be stored and retrieved.

- Docker Engine:

A runtime that includes the Docker daemon, REST API, and CLI client.

4. Container Life Cycle in Docker

Understanding the container life cycle helps in managing containers effectively during development and deployment.

- Create: A container is created from an image but not yet running. (docker create)
- Start: The container begins running. (docker start)
- Pause/Unpause: The container's processes are paused/unpaused. (docker pause, docker unpause)
- Stop: The container is gracefully shut down. (docker stop)
- Kill: The container is forcefully stopped. (docker kill)
- Restart: The container is stopped and started again. (docker restart)
- Remove: The container is deleted. (docker rm)
- Containers can also exit automatically if their main process ends.

5. Common Docker Commands for Managing Images and Containers

- Image Commands:

docker pull <image> – Download an image from Docker Hub
docker images – List downloaded images
docker rmi <image> – Remove an image

- Container Commands:

docker run <image> – Create and start a container

docker ps – Show running containers
docker ps -a – Show all containers (running + stopped)
docker exec -it <container> bash – Access shell inside a container
docker stop <container> – Gracefully stop a container
docker rm <container> – Remove a stopped container

- Dockerfile & Image Building:

Create a Dockerfile describing the container instructions
Build using docker build -t <name>

6. Docker Use Cases in Real Projects

- Development: Consistent dev environments across teams
- Testing: Spin up isolated test environments with specific dependencies
- CI/CD: Containers are core to automated pipelines for testing, building, and deploying code
- Microservices: Run each service in its own container for better scalability and maintainability

Implementation:

```
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, mo
1:C 13 Mar 2024 03:19:03.928 # Warning: no config file specified, using the defa
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...
```

Output:

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

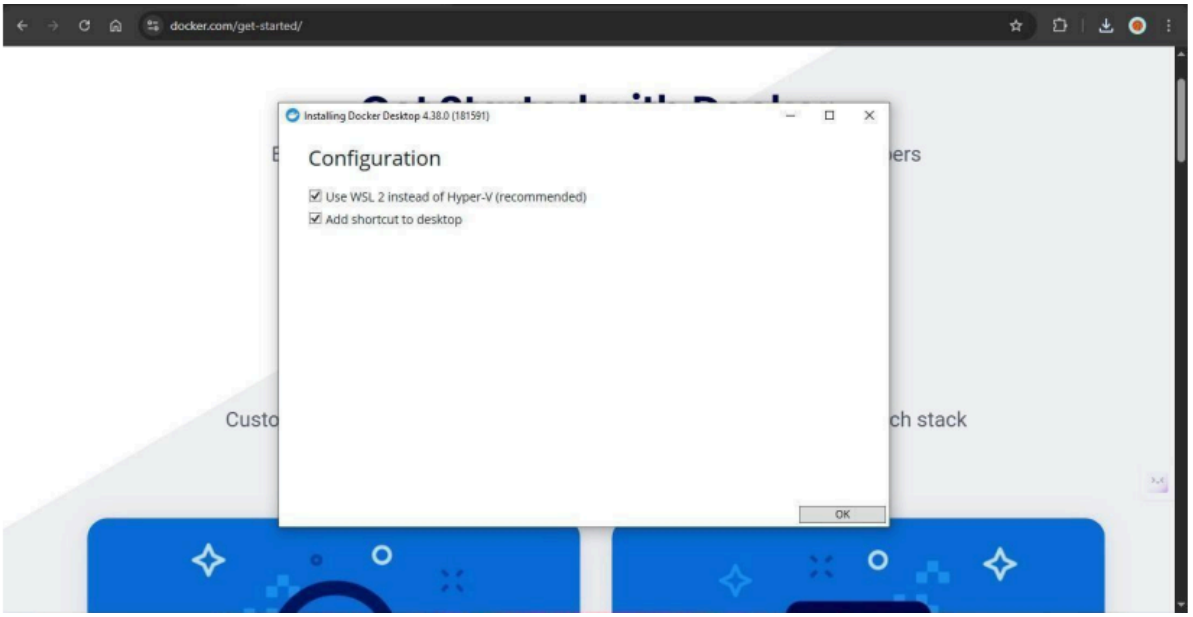
```
C:\Users\202>docker pull redis
Using default tag: latest

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>
```

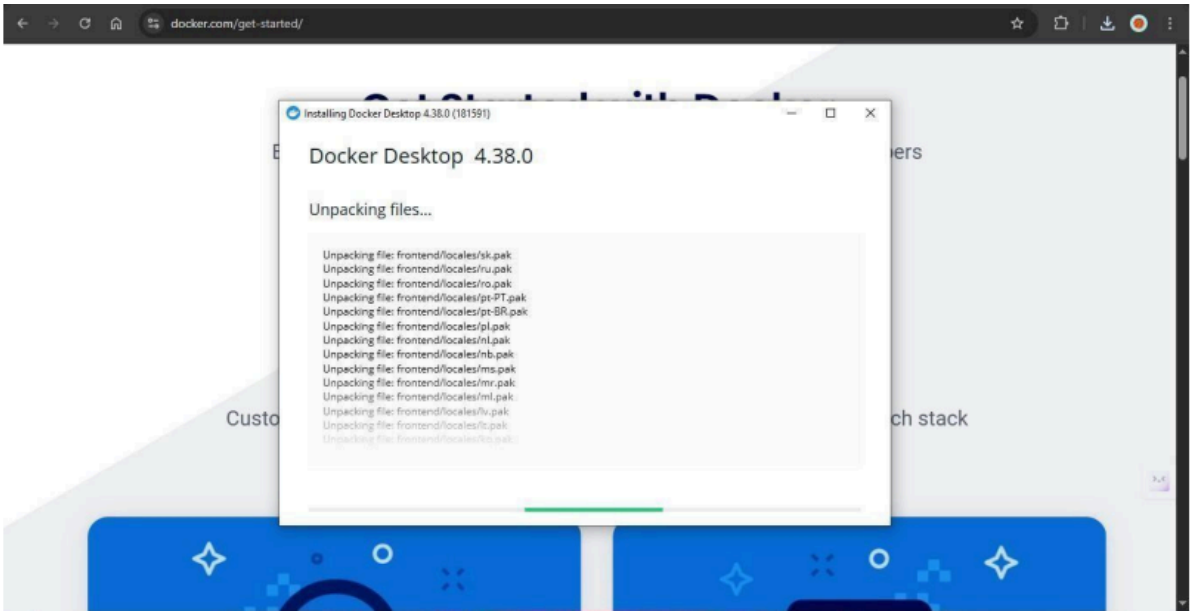
```
C:\Users\202>docker start 052aecb0ee88

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

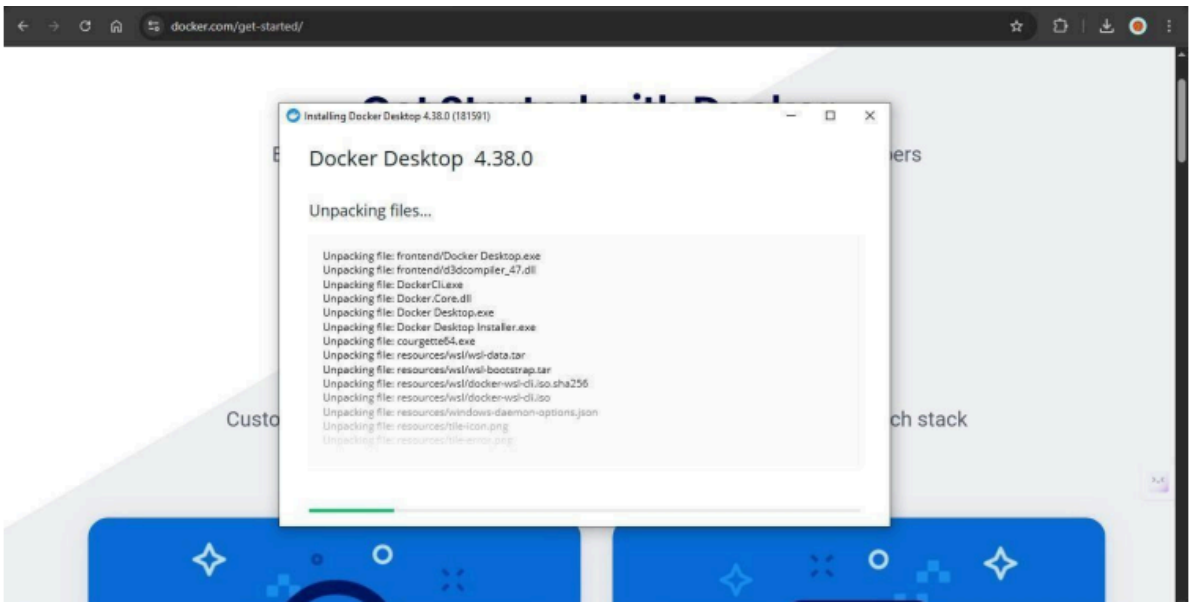
```
C:\Users\202>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
redis        latest    170a1e90f843   2 months ago    138MB
```

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

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

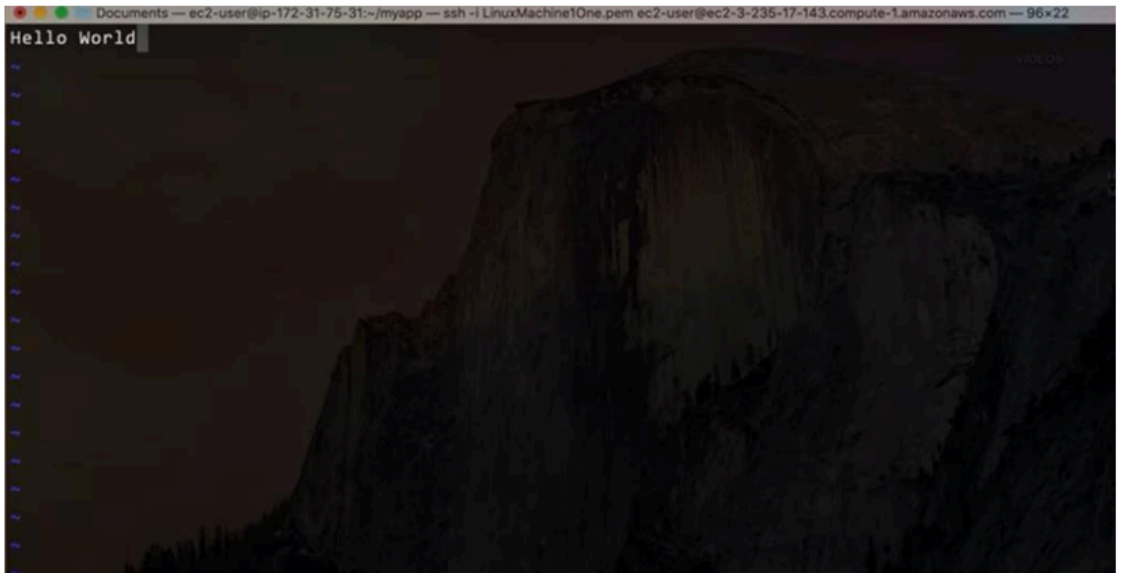
```
C:\Users\202>docker restart 1c4472744083
1c4472744083

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
```

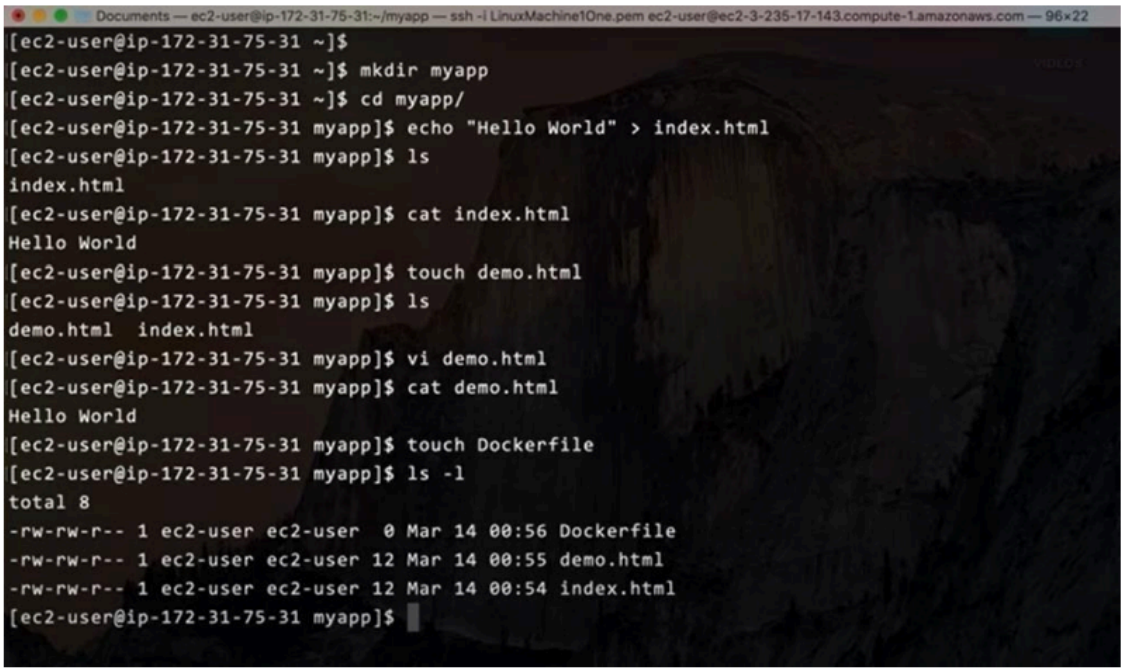

docker.com/get-started/
Installing Docker Desktop 4.38.0 (181591)
Configuration
Use WSL 2 instead of Hyper-V (recommended)
Add shortcut to desktop
OK

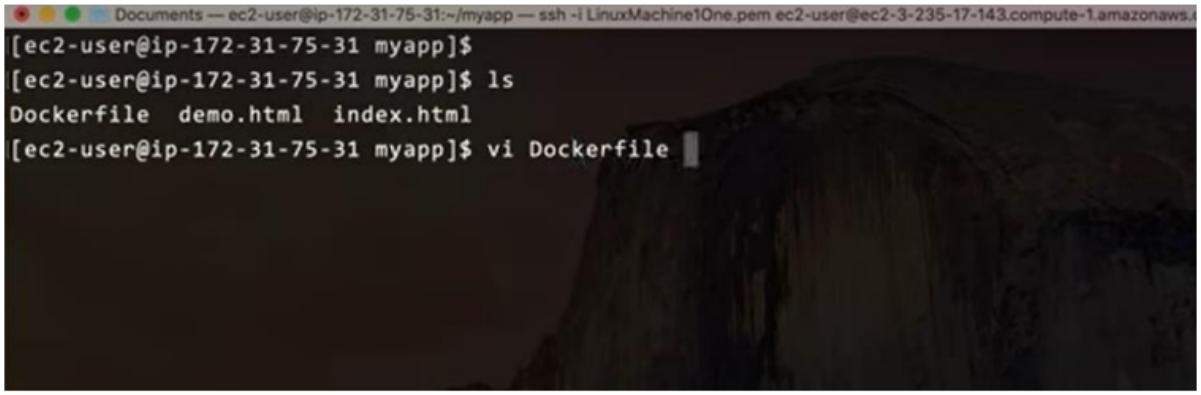

docker.com/get-started/
Installing Docker Desktop 4.38.0 (181591)
Docker Desktop 4.38.0
Unpacking files...
Unpacking file: frontend/locales/sk.pak
Unpacking file: frontend/locales/ru.pak
Unpacking file: frontend/locales/ro.pak
Unpacking file: frontend/locales/pt-PT.pak
Unpacking file: frontend/locales/pt-BR.pak
Unpacking file: frontend/locales/pl.pak
Unpacking file: frontend/locales/nl.pak
Unpacking file: frontend/locales/nb.pak
Unpacking file: frontend/locales/ms.pak
Unpacking file: frontend/locales/mr.pak
Unpacking file: frontend/locales/ml.pak
Unpacking file: frontend/locales/lv.pak

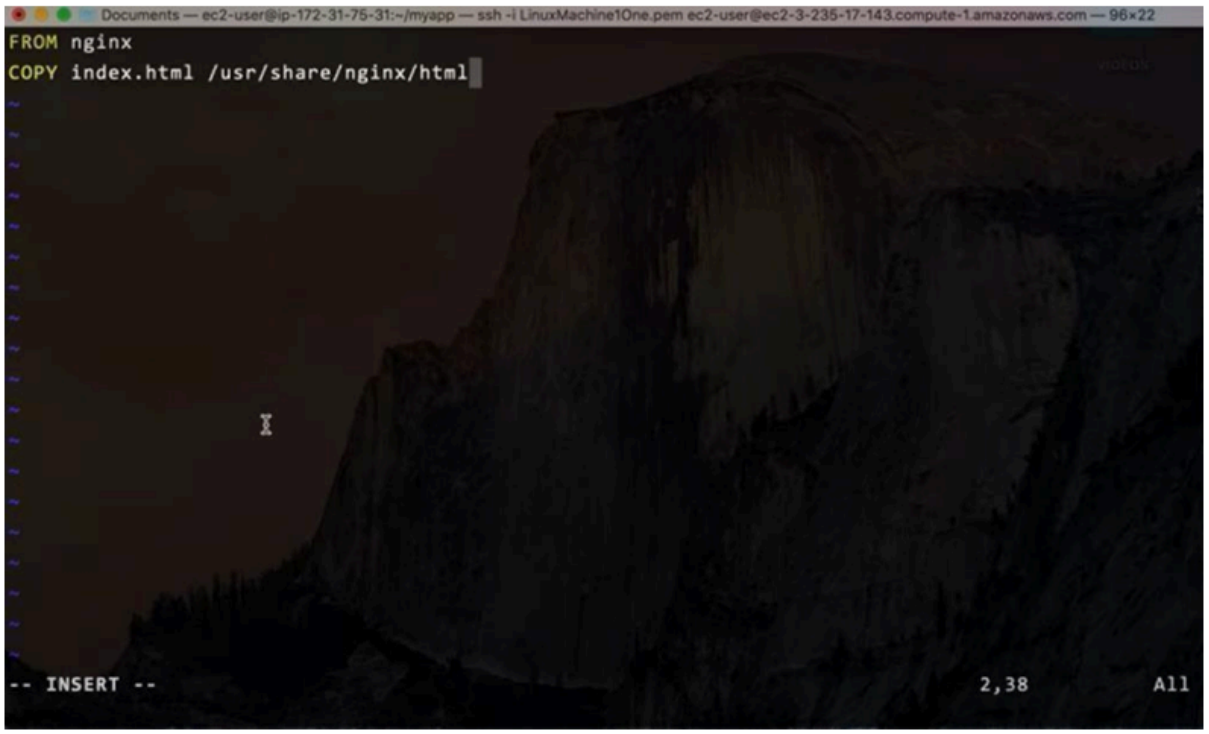

docker.com/get-started/
Installing Docker Desktop 4.38.0 (181591)
Docker Desktop 4.38.0
Unpacking files...
Unpacking file: frontend/Docker Desktop.exe
Unpacking file: frontend/d3dcompiler_47.dll
Unpacking file: DockerCli.exe
Unpacking file: Docker.Core.dll
Unpacking file: Docker Desktop.exe
Unpacking 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

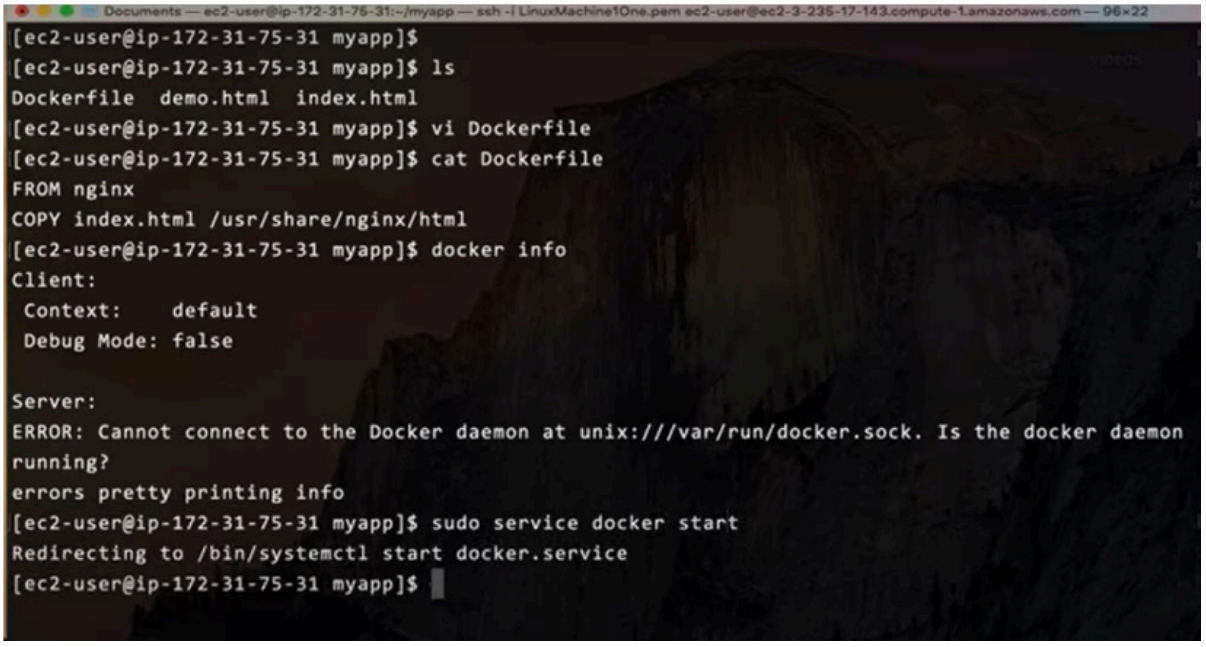

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 ~]$
[ec2-user@ip-172-31-75-31 ~]$ mkdir myapp
[ec2-user@ip-172-31-75-31 ~]$ cd myapp/
[ec2-user@ip-172-31-75-31 myapp]$ echo "Hello World" > index.html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

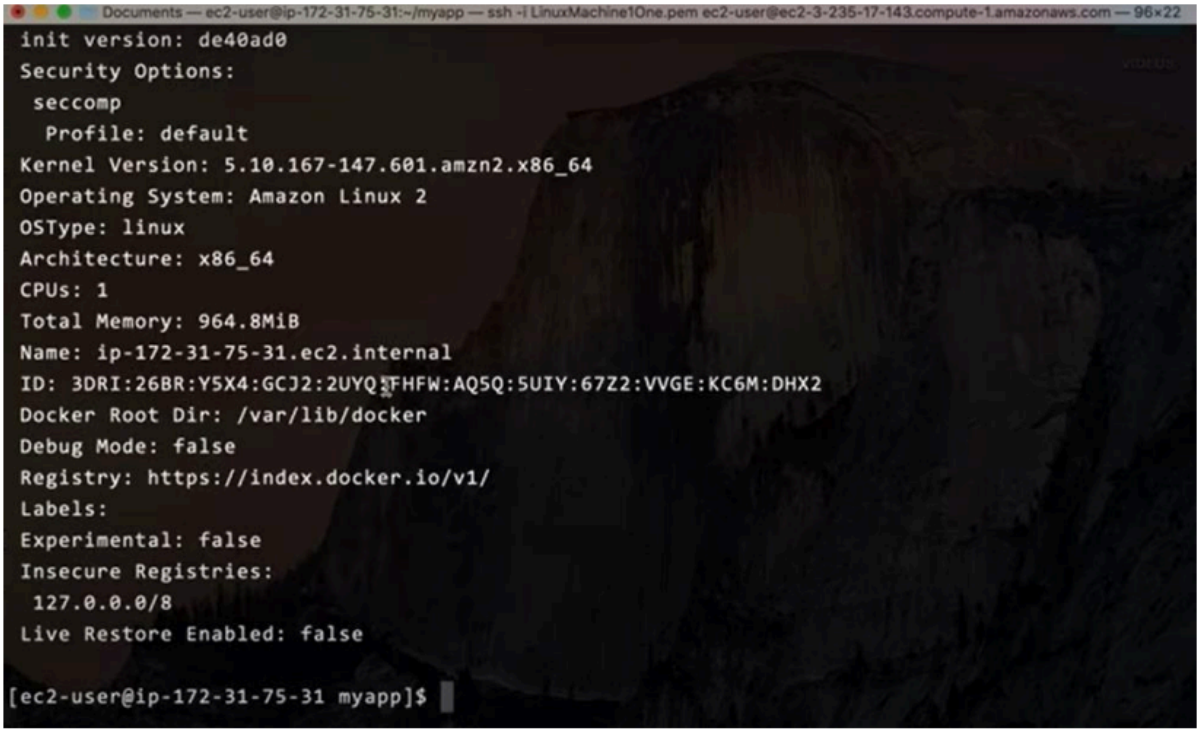

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
Hello World

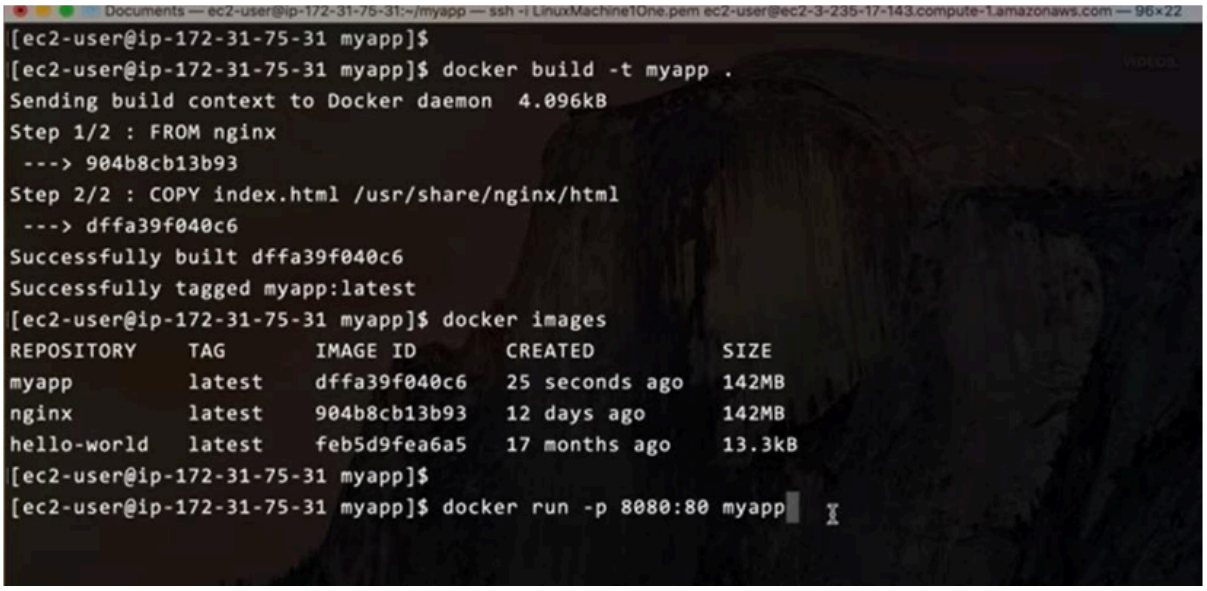

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 ~]$
[ec2-user@ip-172-31-75-31 ~]$ mkdir myapp
[ec2-user@ip-172-31-75-31 ~]$ cd myapp/
[ec2-user@ip-172-31-75-31 myapp]$ echo "Hello World" > index.html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

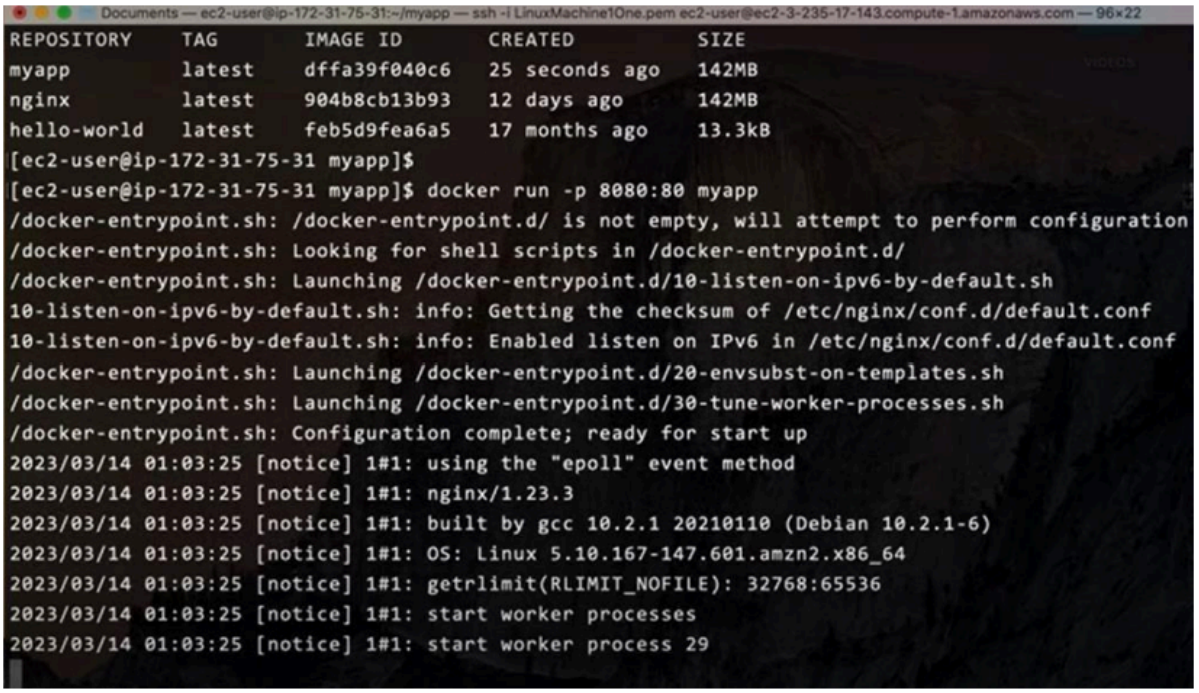

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ ls
Dockerfile demo.html index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi Dockerfile


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

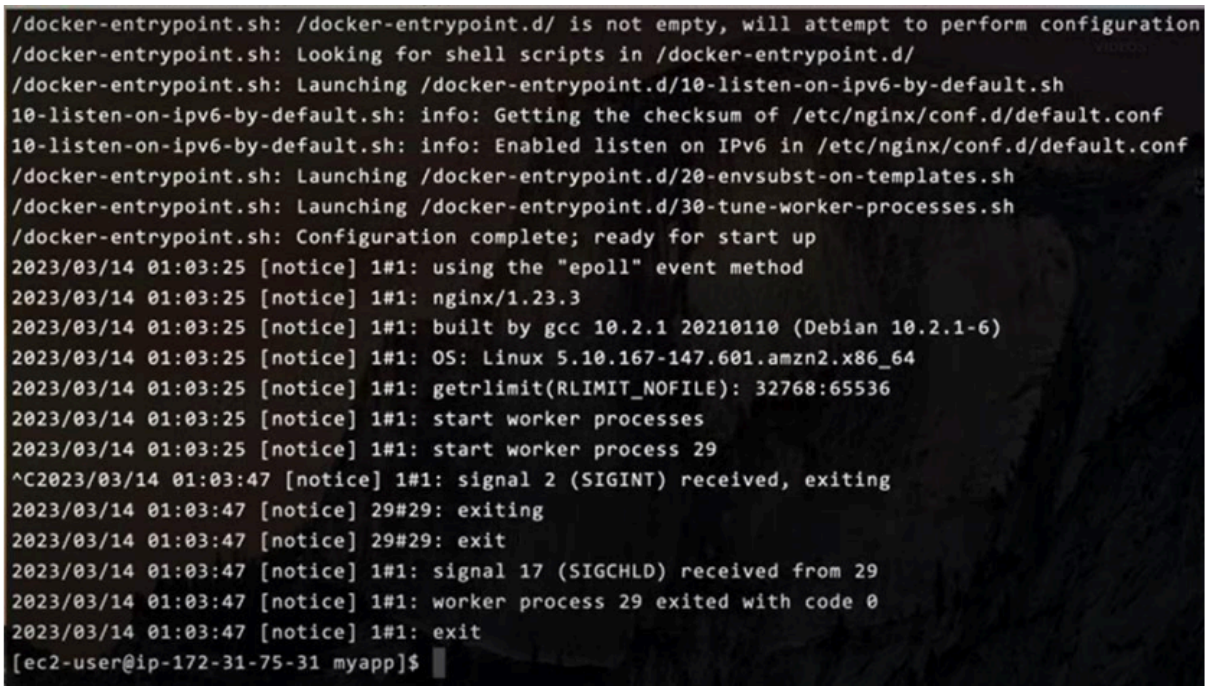

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ ls
Dockerfile demo.html index.html
[ec2-user@ip-172-31-75-31 myapp]$ vi Dockerfile
[ec2-user@ip-172-31-75-31 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]$

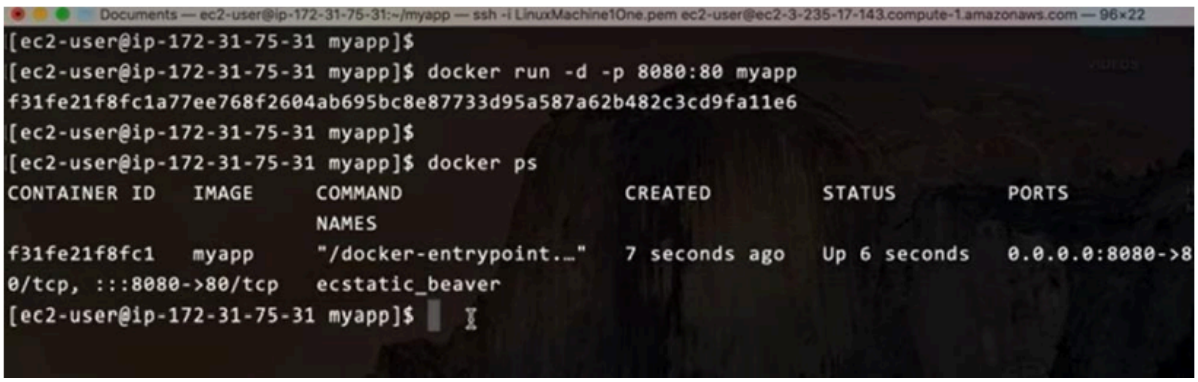

Documents — ec2-user@ip-172-31-75-31:~/myapp — ssh -i LinuxMachine1One.pem ec2-user@ec2-3-235-17-143.compute-1.amazonaws.com — 96×22
[ec2-user@ip-172-31-75-31 myapp]$
[ec2-user@ip-172-31-75-31 myapp]$ docker build -t myapp .
Sending build context to Docker daemon 4.096kB
Step 1/2 : FROM nginx
---> 904b8cb13b93
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]$
```

Conclusion: Hence we have successfully understood Docker Architecture and Container Life Cycle, installed Docker and executed docker commands to manage images and interacted with containers