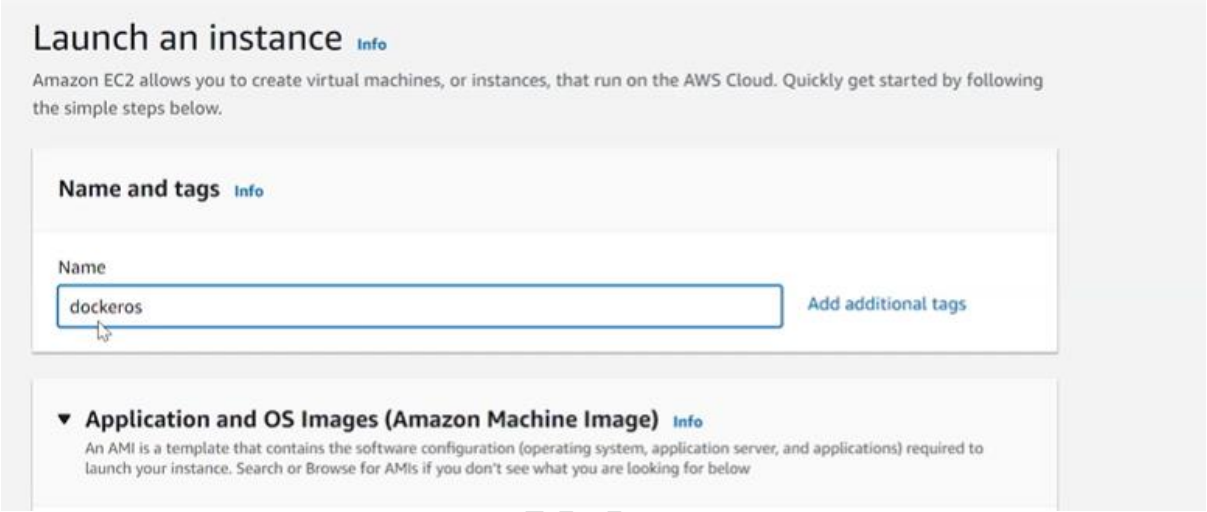
Summary

Session No – 19

- Launching amazon Linux instance



- The operating system contains two things physical hardware & OS on the top of which we are launching the docker engine
- On the top of the docker engine, we launch the container & on the top of the docker container we can launch the docker engine from which we can launch the container this is called docker inside docker (DIND)
- Whenever we need multiple docker engines may be for testing purpose rather than launching multiple nodes we can use the concept of docker inside docker.
- On the top of the docker container, we launch the docker engine to launch multiple containers which work as a real instance, the same setup we simulate with the help of docker inside docker. This method or approach is not recommended in the real production environment because if the base system goes down entire setup will go down
- Installing docker & starting docker services

- Whenever we start the docker service it is a kind of process or engine & if one process wants to communicate with another process it uses a unique socket
- If we check the status with the **systemctl status docker** command behind the scene it launches the program or process & in this process docker start one socket because of which the docker process has the capability to communicate with other processes

```
[root@ip-172-31-46-115 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-11-30 16:00:45 UTC; 1min 11s ago
     Docs: https://docs.docker.com
  Process: 3875 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 3874 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 3878 (dockerd)
    Tasks: 7
   Memory: 21.5M
   CGroup: /system.slice/docker.service
           └─3878 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Nov 30 16:00:44 ip-172-31-46-115.ap-south-1.compute.internal dockerd[3878]: time="2022-11-30T16:00:44.592207335Z" level=info msg=...rpc
Nov 30 16:00:44 ip-172-31-46-115.ap-south-1.compute.internal dockerd[3878]: time="2022-11-30T16:00:44.708976312Z" level=warning m...ht"
Nov 30 16:00:44 ip-172-31-46-115.ap-south-1.compute.internal dockerd[3878]: time="2022-11-30T16:00:44.709003907Z" level=warning m...ce"
Nov 30 16:00:44 ip-172-31-46-115.ap-south-1.compute.internal dockerd[3878]: time="2022-11-30T16:00:44.709220764Z" level=info msg=...t."
Nov 30 16:00:44 ip-172-31-46-115.ap-south-1.compute.internal dockerd[3878]: time="2022-11-30T16:00:44.932253430Z" level=info msg=...ss"
```

- If the socket is not there docker engine will not be able to do anything with the containers from the images
- Because the base system has the socket we are eligible to run the docker engine

```
[root@ip-172-31-46-115 ~]# cd /run/containerd/
[root@ip-172-31-46-115 containerd]# ls
containerd.sock  containerd.sock.ttrpc  io.containerd.runtime.v1.linux  io.containerd.runtime.v2.task  s
[root@ip-172-31-46-115 containerd]# ls -l
total 0
srw-rw---- 1 root root  0 Nov 30 16:00 containerd.sock
srw-rw---- 1 root root  0 Nov 30 16:00 containerd.sock.ttrpc
drwx--x--x 2 root root 40 Nov 30 16:00 io.containerd.runtime.v1.linux
drwx--x--x 3 root root 60 Nov 30 16:03 io.containerd.runtime.v2.task
drw------- 2 root root 60 Nov 30 16:03 s
[root@ip-172-31-46-115 containerd]#
```

- If we launch the container & go inside the **/run** folder there is no socket file available there so even if we install docker inside the container we will not be able to start the docker services with the systemctl command

```
[root@4534b16284c0 /]# cd /run/
[root@4534b16284c0 run]# ls
console  cryptsetup  faillock  lock  log  sepermit  setrans  systemd  user  utmp
[root@4534b16284c0 run]# yum  install docke
```

```
[root@4534b16284c0 ~]# systemctl start docker
Failed to get D-Bus connection: Operation not permitted
[root@4534b16284c0 ~]#
```

- Whenever we run the systemctl command behind the scene it runs the program called dockerd to start the services

```
[root@ip-172-31-46-115 containerd]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-11-30 16:00:45 UTC; 6min ago
     Docs: https://docs.docker.com
  Process: 3875 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 3874 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 3878 (dockerd)
    Tasks: 9
   Memory: 239.7M
   CGroup: /system.slice/docker.service
           └─3878 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536
```

- Even if we use the **dockerd** command to start the services it will fail because there is no socket available

```
[root@4534b16284c0 ~]# dockerd
WARN[0000] could not change group /var/run/docker.sock to docker: group docker not found
INFO[0000] libcontainerd: new containerd process, pid: 156
Error starting daemon: operation not permitted
[root@4534b16284c0 ~]#
```

- On the base system, docker is working because we have a socket available

```
[root@ip-172-31-46-115 containerd]# pwd
/run/containerd
[root@ip-172-31-46-115 containerd]# ls
containerd.sock  containerd.sock.ttrpc  io.containerd.runtime.v1.linux  io.containerd.runtime.v2.task  s
[root@ip-172-31-46-115 containerd]# ls -l
total 0
srw-rw---- 1 root root  0 Nov 30 16:00 containerd.sock
srw-rw---- 1 root root  0 Nov 30 16:00 containerd.sock.ttrpc
drwx--x--x 2 root root 40 Nov 30 16:00 io.containerd.runtime.v1.linux
drwx--x--x 3 root root 60 Nov 30 16:03 io.containerd.runtime.v2.task
drw------- 2 root root 60 Nov 30 16:15 s
[root@ip-172-31-46-115 containerd]# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS          PORTS     NAMES
8f7eef9d761b   ubuntu:14.04   "/bin/bash"   12 minutes ago   Up 12 minutes             os1
[root@ip-172-31-46-115 containerd]#
```

- The socket is just the file so we can share this with the container from our base system

```
[root@ip-172-31-46-115 containerd]# docker run -it --name myd1  -v /run/containerd/:/run/containerd/   centos:7
[root@93f5d7a72e2f /]# cd /run/containerd/
[root@93f5d7a72e2f containerd]# ls
containerd.sock  containerd.sock.ttrpc  io.containerd.runtime.v1.linux  io.containerd.runtime.v2.task  s
[root@93f5d7a72e2f containerd]# yum install docker -y
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
```

- When we install docker software it gives us two commands dockerd & docker. The Docker command always contacts to dockerd command ie is the reason docker is a client-side program & dockerd is a server-side program
- Pulling docker images

```
[root@ip-172-31-46-115 containerd]# docker pull docker
Using default tag: latest
latest: Pulling from library/docker
c158987b0551: Pull complete
93eea1d5d8e5: Pull complete
6984615ae184: Pull complete
3e603bb99416: Pull complete
8102f44846c1: Extracting [==================>                                ]  6.062MB/14.77MB
65281e1e3f66: Download complete
d36e18e3aa06: Download complete
a2ec3bdf2755: Download complete
```

- Launching the container from the docker image and mounting the socket

```
[root@ip-172-31-46-115 containerd]# docker run -dit  --name os2  -v /run/containerd/containerd.sock:/var/run/docker.sock  docker
c95f77c85e9f9fc57d2f0de37619c12b56063c9573c473152d699c8c4e03be73
[root@ip-172-31-46-115 containerd]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
c95f77c85e9f   docker         "docker-entrypoint.s…"   3 seconds ago    Up 2 seconds              os2
8f7eef9d761b   ubuntu:14.04   "/bin/bash"              23 minutes ago   Up 23 minutes             os1
[root@ip-172-31-46-115 containerd]#
```

- Entering inside the container with the exec command

```
[root@ip-172-31-46-115 containerd]# docker exec -it os2   sh
/ #
/ #
/ #
```

- There is a conflict between the version running on the base system which is not compatible with the version we are running inside the container

```
~ #
~ # docker ps
Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": net/http: HTTP/1.x transport connection broken: malformed HTTP response
"\x00\x00\x06\x04\x00\x00\x00\x00\x00\x00\x05\x00\x00@\x00".
* Are you trying to connect to a TLS-enabled daemon without TLS?
~ # docker images
Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/json": net/http: HTTP/1.x transport connection broken: malformed HTTP response "\x
0\x00\x06\x04\x00\x00\x00\x00\x00\x00\x05\x00\x00@\x00".
* Are you trying to connect to a TLS-enabled daemon without TLS?
~ #
```

- Sharing the base system socket with the container is not secure
- Another method to launch docker inside docker is to create the socket
- In the container we can install the docker & dockerd is the program that will start the docker services but it is not allowing us due to some security reasons

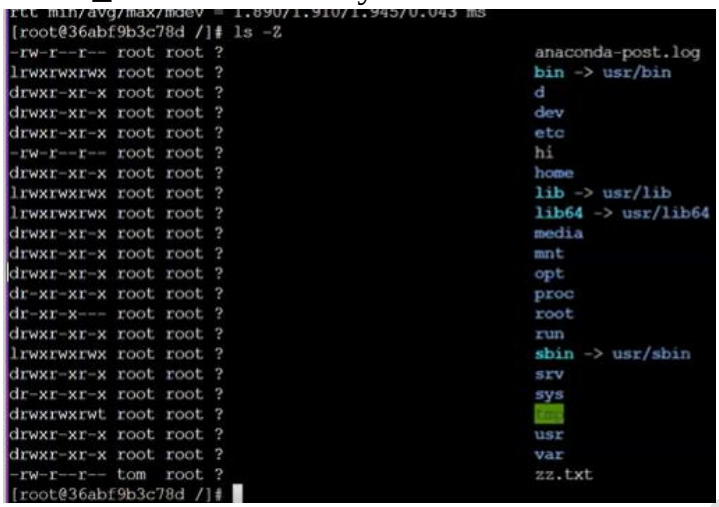
```
[root@674a9adc15ca /]# dockerd
WARN[0000] could not change group /var/run/docker.sock to docker: group docker not found
INFO[0000] libcontainerd: new containerd process, pid: 149
Error starting daemon: operation not permitted
[root@674a9adc15ca /]#
```

- In the Linux operating system root is the administrator of the system which has unlimited power.
- The interesting thing is even though we are a root user in the container still we are not able to change the hostname

```
[root@36abf9b3c78d /]# hostname  vimal.lw.com
hostname: you must be root to change the host name
[root@36abf9b3c78d /]#
[root@36abf9b3c78d /]# whoami
root
[root@36abf9b3c78d /]#
```

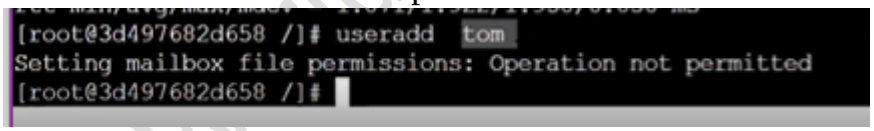
- Inside the container even if we log in with the root user we are unable to change the hostname & one of the facilities we can do even if we are a root user is to create the socket
- Whenever we launch the container, it is restricted with some limited power which is called privileges because the main use of the container is to run the program. Privileges are also called capabilities
- There are a lot of capabilities we can give to the containers or we can remove from the container

- If we try to run the **ls -Z** command inside the container to see the SELinux permission to the file that is also not allowed inside the container because SELinux capabilities are disabled with the **MAC_OVERRIDE** keyword



- If we want to allow the capability to a container we have to use the **--cap-add** keyword similarly to remove the capabilities we have to use the **--cap-drop** keyword
- --**privileged** keyword to give all the capabilities to the container
- Removing & adding capabilities from the container

- Now if we try to create the user in the container it will not be permitted because we removed the capabilities



- **/dev** directory in Linux contains all the information of base system devices

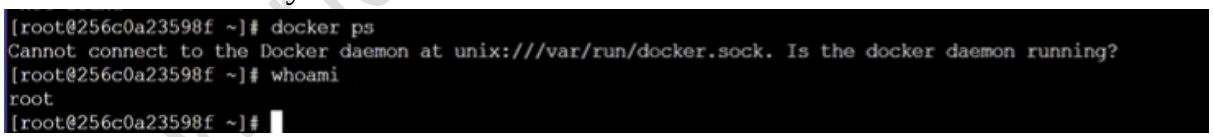
- Privilege gives the capability to access all the base system devices which are very dangerous

- Launching a container with all the capabilities

- Installing docker in the container

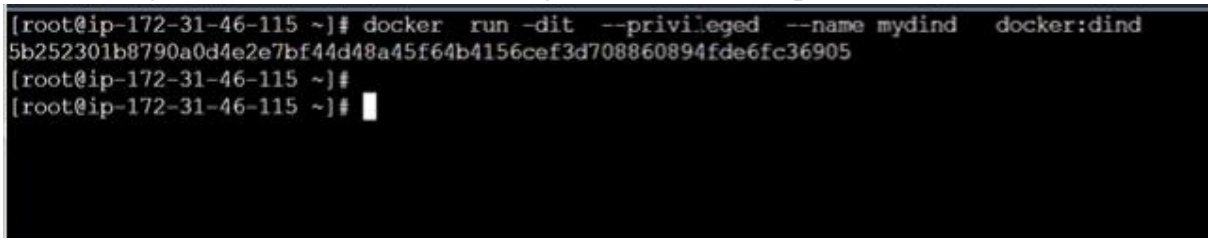
- Starting docker services with dockerd. The warning is due to we need some extra groups & overlay divers which we can install manually

- Docker ps is not running because we need some extra packages which we can install manually



- Pulling the image which has all the packages downloaded which are required to run docker in docker

- Launching a container from the image with all the capabilities



- Inside the container, we can run all the docker client commands

- From the base system, we can not see the containers which are running inside the container
- The difference between the first method & second method is in the second method we can not see the containers which are running inside the container from the base system