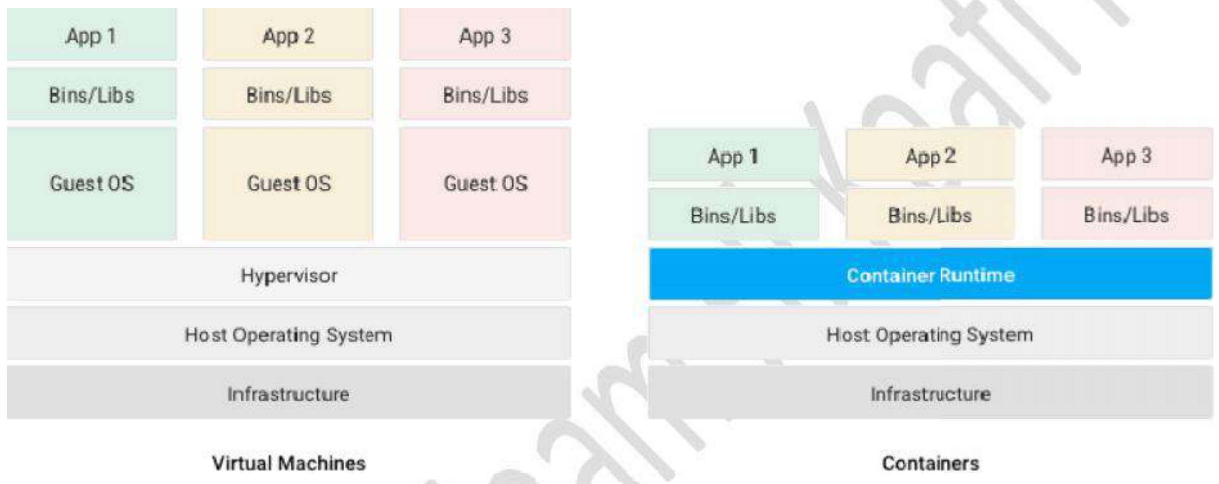
What are containers?

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run. This decoupling allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer’s personal laptop. Containerization provides a clean separation of concerns, as developers focus on their application logic and dependencies, while IT operations teams can focus on deployment and management without bothering with application details such as specific software versions and configurations specific to the app.

For those coming from virtualized environments, containers are often compared with virtual machines (VMs). You might already be familiar with VMs: a guest operating system such as Linux or Windows runs on top of a host operating system with virtualized access to the underlying hardware. Like virtual machines, containers allow you to package your application together with libraries and other dependencies, providing isolated environments for running your software services. As you’ll see below however, the similarities end here as containers offer a far more lightweight unit for developers and IT Ops teams to work with, carrying a myriad of benefits.



Why Containers?

Instead of vitalizing the hardware stack as with the virtual machines approach, containers virtualize at the operating system level, with multiple containers running atop the OS kernel directly. This means that containers are far more lightweight: they share the OS kernel, start much faster, and use a fraction of the memory compared to booting an entire OS.

There are many container formats available. Docker is a popular, open-source container format that is supported on Google Cloud Platform and by Google Kubernetes Engine.

Consistent Environment

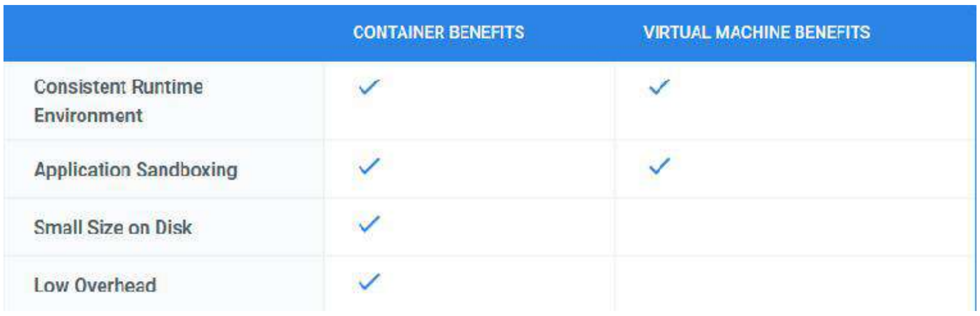
Containers give developers the ability to create predictable environments that are isolated from other applications. Containers can also include software dependencies needed by the application, such as specific versions of programming language runtimes and other software libraries. From the developer’s perspective, all this is guaranteed to be consistent no matter where the application is ultimately deployed. All this translates to productivity: developers and IT Ops teams spend less time debugging and diagnosing differences in environments, and more time shipping new functionality for users. And it means fewer bugs since developers can now make assumptions in dev and test environments they can be sure will hold true in production.

Run Anywhere

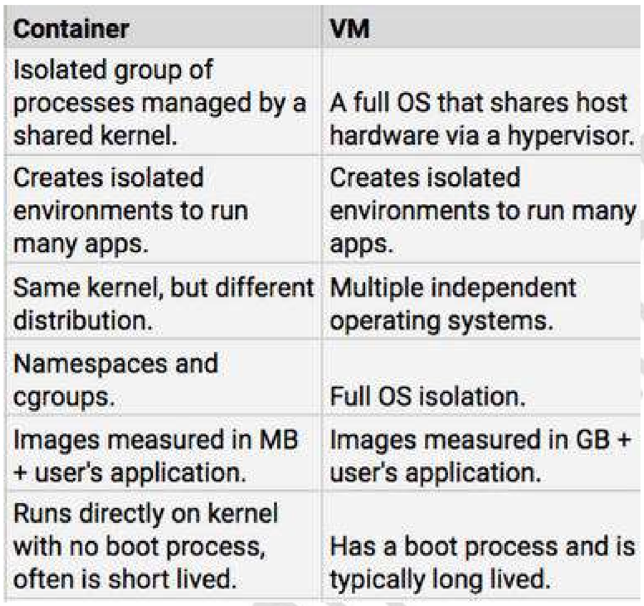
Containers are able to run virtually anywhere, greatly easing development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or bare metal; on a developer’s machine or in data centers on-premises; and of course, in the public cloud. The widespread popularity of the Docker image format for containers further helps with portability. Wherever you want to run your software, you can use containers.

Isolation

Containers virtualize CPU, memory, storage, and network resources at the OS-level, providing developers with a sandboxed view of the OS logically isolated from other applications.

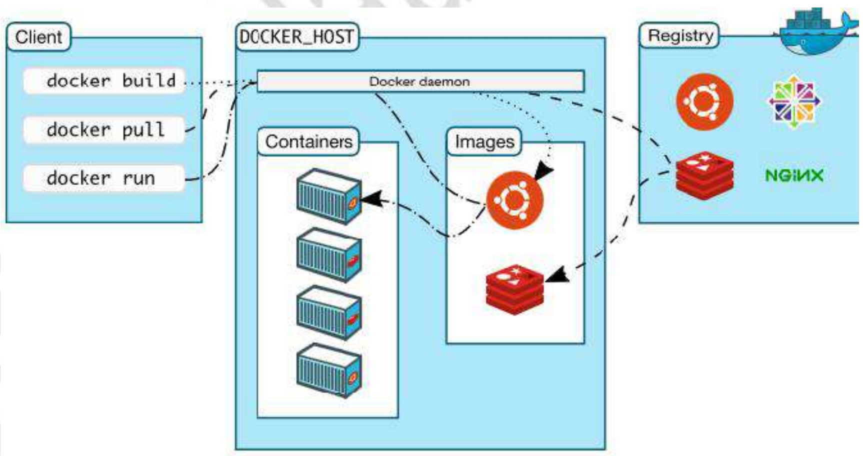


	CONTAINER BENEFITS	VIRTUAL MACHINE BENEFITS
Consistent Runtime Environment	✓	✓
Application Sandboxing	✓	✓
Small Size on Disk	✓	
Low Overhead	✓	



Container	VM
Isolated group of processes managed by a shared kernel.	A full OS that shares host hardware via a hypervisor.
Creates isolated environments to run many apps.	Creates isolated environments to run many apps.
Same kernel, but different distribution.	Multiple independent operating systems.
Namespaces and cgroups.	Full OS isolation.
Images measured in MB + user's application.	Images measured in GB + user's application.
Runs directly on kernel with no boot process, often is short lived.	Has a boot process and is typically long lived.

Docker Architecture

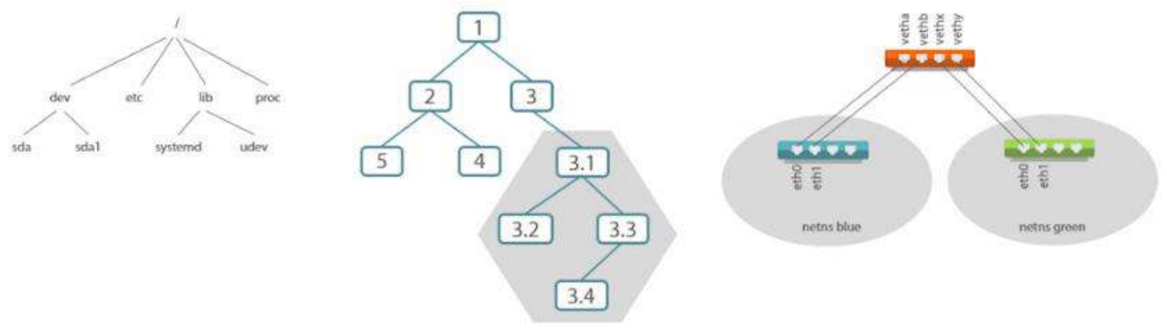


Docker hub is nothing but the remote registry to store images (public/private) if user want local one then they have to user registory2 where you can store their own images (private).

Example : Openjdk public image -
https://hub.docker.com/_/openjdk
Just run - docker pull openjdk

Here we are pulling this image and run as container where no need to install anything.
- Every container have its own hierarchy .

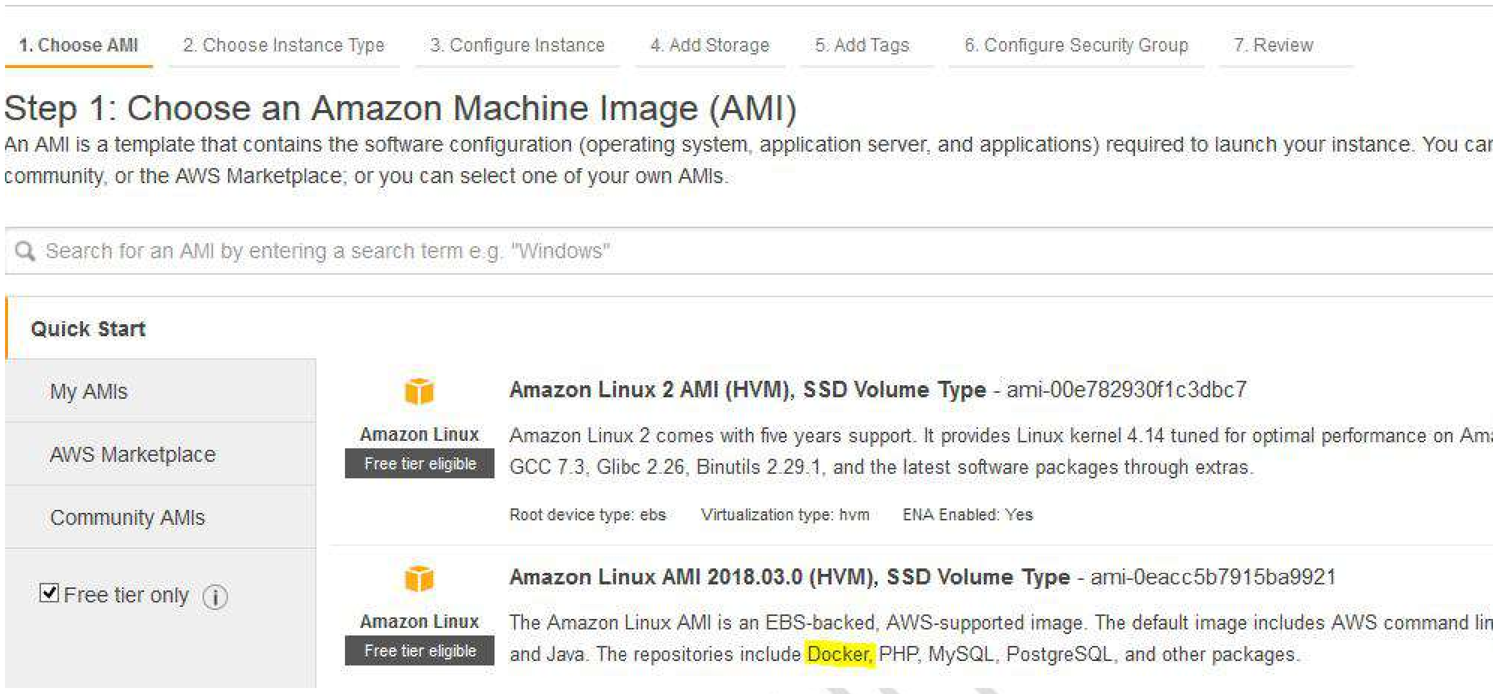
How Containers Work



DOCKER ON AWS

1. Login on your aws account and select free tier AMI with docker.

2. Select all default setting and until security group.



1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You car
community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Quick Start

My AMIs

AWS Marketplace

Community AMIs

☑ Free tier only

Amazon Linux — Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-00e782930f1c3dbc7

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Am
GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

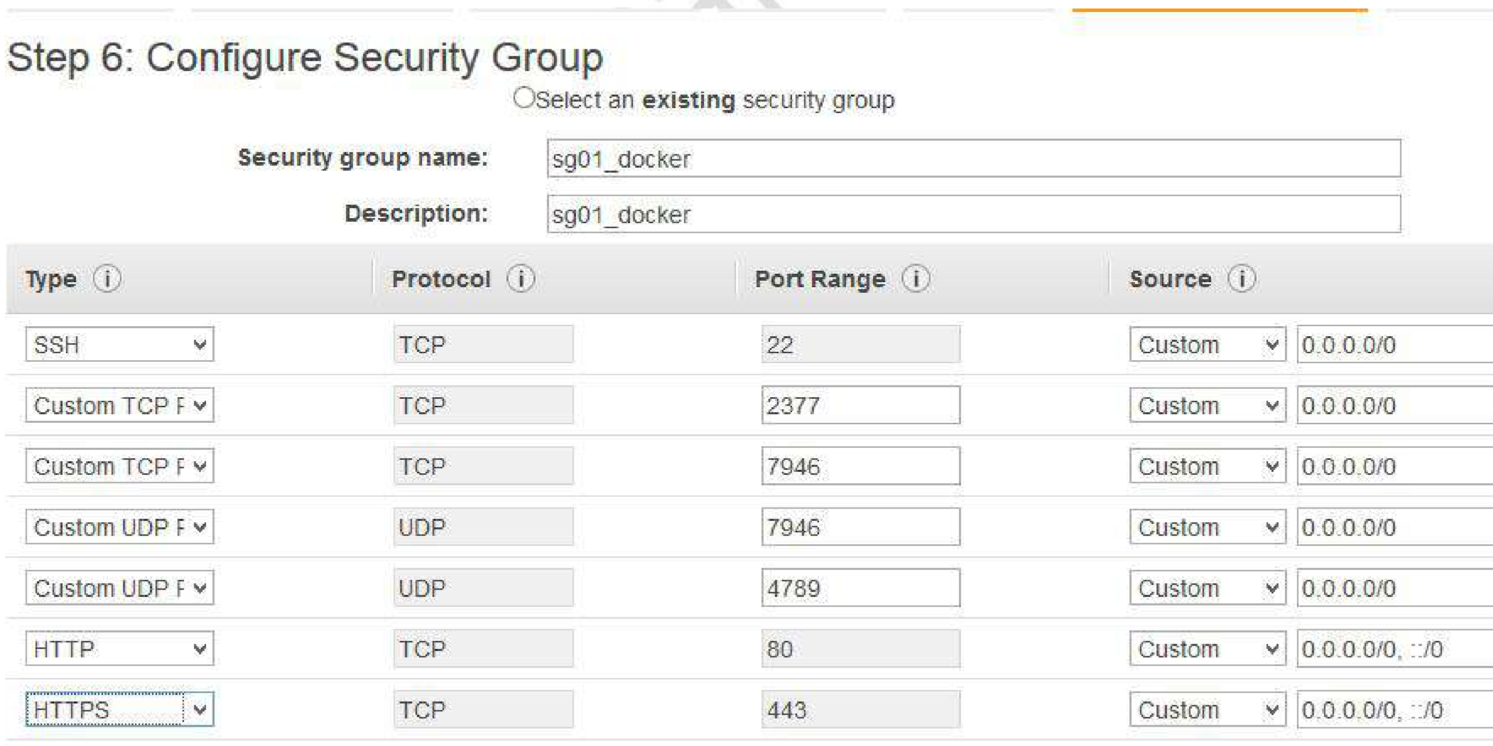
Amazon Linux — Free tier eligible

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0eacc5b7915ba9921

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command lin
and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

3. Create security group -

The following ports must be available. On some systems, these ports are open by default.
TCP port 2377 for cluster management communications
TCP and UDP port 7946 for communication among nodes
UDP port 4789 for overlay network traffic



Step 6: Configure Security Group

○ Select an **existing** security group

Security group name:	sg01_docker
Description:	sg01_docker

Type	Protocol	Port Range	Source	
SSH	TCP	22	Custom	0.0.0.0/0
Custom TCP F	TCP	2377	Custom	0.0.0.0/0
Custom TCP F	TCP	7946	Custom	0.0.0.0/0
Custom UDP F	UDP	7946	Custom	0.0.0.0/0
Custom UDP F	UDP	4789	Custom	0.0.0.0/0
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0

https://docs.docker.com/engine/swarm/swarm-tutorial/

4. Select key and launch the instance.

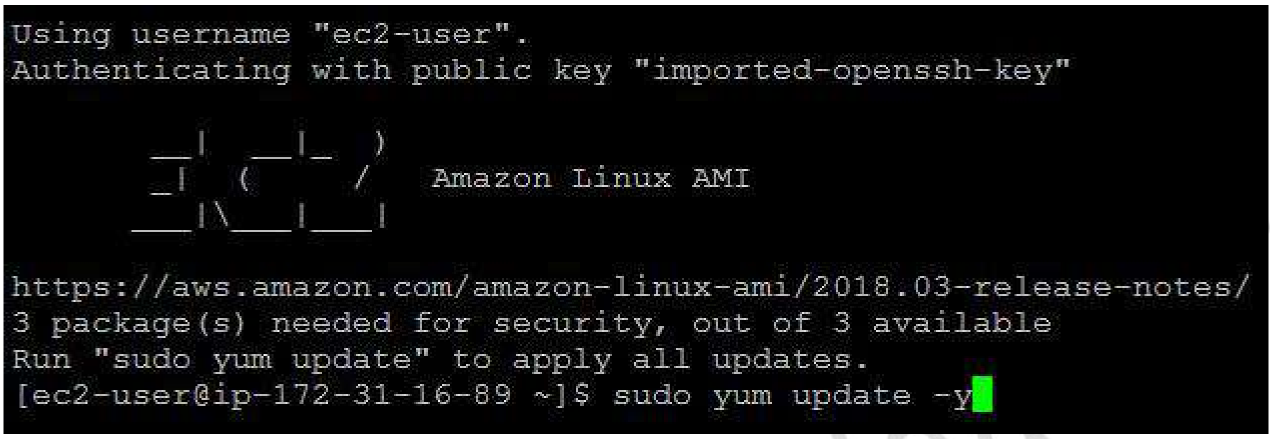
If it's your new setup then create new key and download it. To use these keys on putty have to convert from .pem to .ppk.

a. Download Putty and puttygen.

b. Use puttygen to convert .PEM file to .PPK file.
c. Start puttygen and select “Load”
d. Select your .PEM file.
e. Putty will convert the .PEM format to .PPK format.
===

1. **Update installed package on you instance.**

```
sudo yum update -y
```



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
3 package(s) needed for security, out of 3 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-16-89 ~]$ sudo yum update -y
```

2. **Install docker**

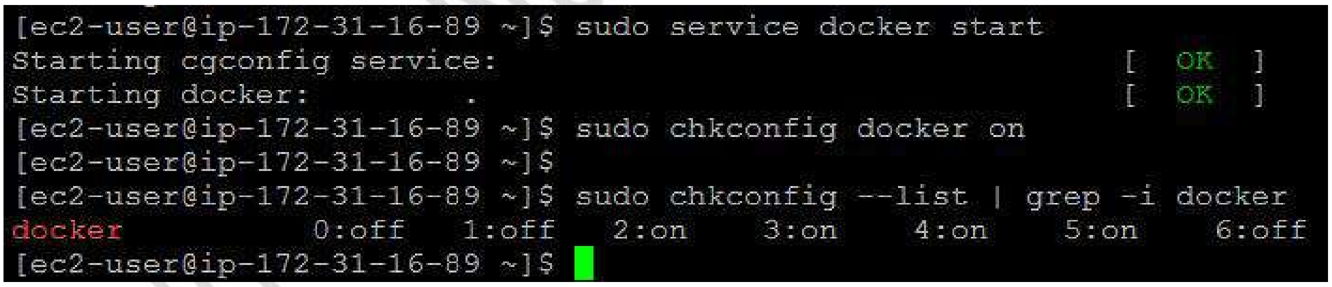
```
sudo yum install docker -y
```

```
[ec2-user@ip-172-31-16-89 ~]$ sudo yum install docker -y
```

Note: Where you have install docker its will be your docker host.

3. **Start docker services**

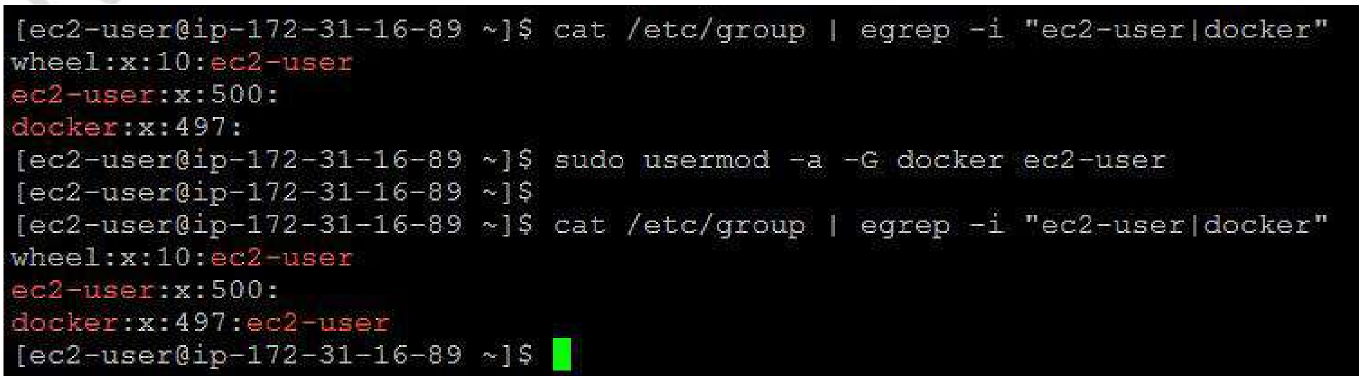
```
sudo service docker start; sudo chkconfig docker on; sudo chkconfig --list | grep -i docker
```



```
[ec2-user@ip-172-31-16-89 ~]$ sudo service docker start
Starting cgconfig service:                                 [  OK  ]
Starting docker:        .                                  [  OK  ]
[ec2-user@ip-172-31-16-89 ~]$ sudo chkconfig docker on
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ sudo chkconfig --list | grep -i docker
docker          0:off   1:off   2:on    3:on    4:on    5:on    6:off
[ec2-user@ip-172-31-16-89 ~]$
```

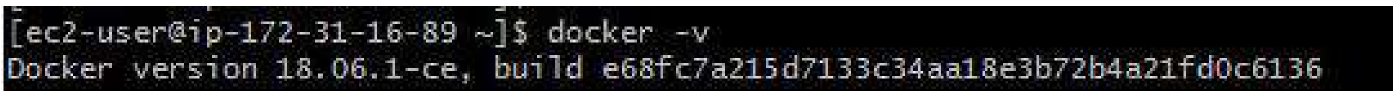
4. **Add ec2-user in docker group so we can run docker commands without sudo.**

```
cat /etc/group | egrep -i "ec2-user|docker"
sudo usermod -a -G docker ec2-user
cat /etc/group | egrep -i "ec2-user|docker"
```



```
[ec2-user@ip-172-31-16-89 ~]$ cat /etc/group | egrep -i "ec2-user|docker"
wheel:x:10:ec2-user
ec2-user:x:500:
docker:x:497:
[ec2-user@ip-172-31-16-89 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ cat /etc/group | egrep -i "ec2-user|docker"
wheel:x:10:ec2-user
ec2-user:x:500:
docker:x:497:ec2-user
[ec2-user@ip-172-31-16-89 ~]$
```

5. To check docker version and its build

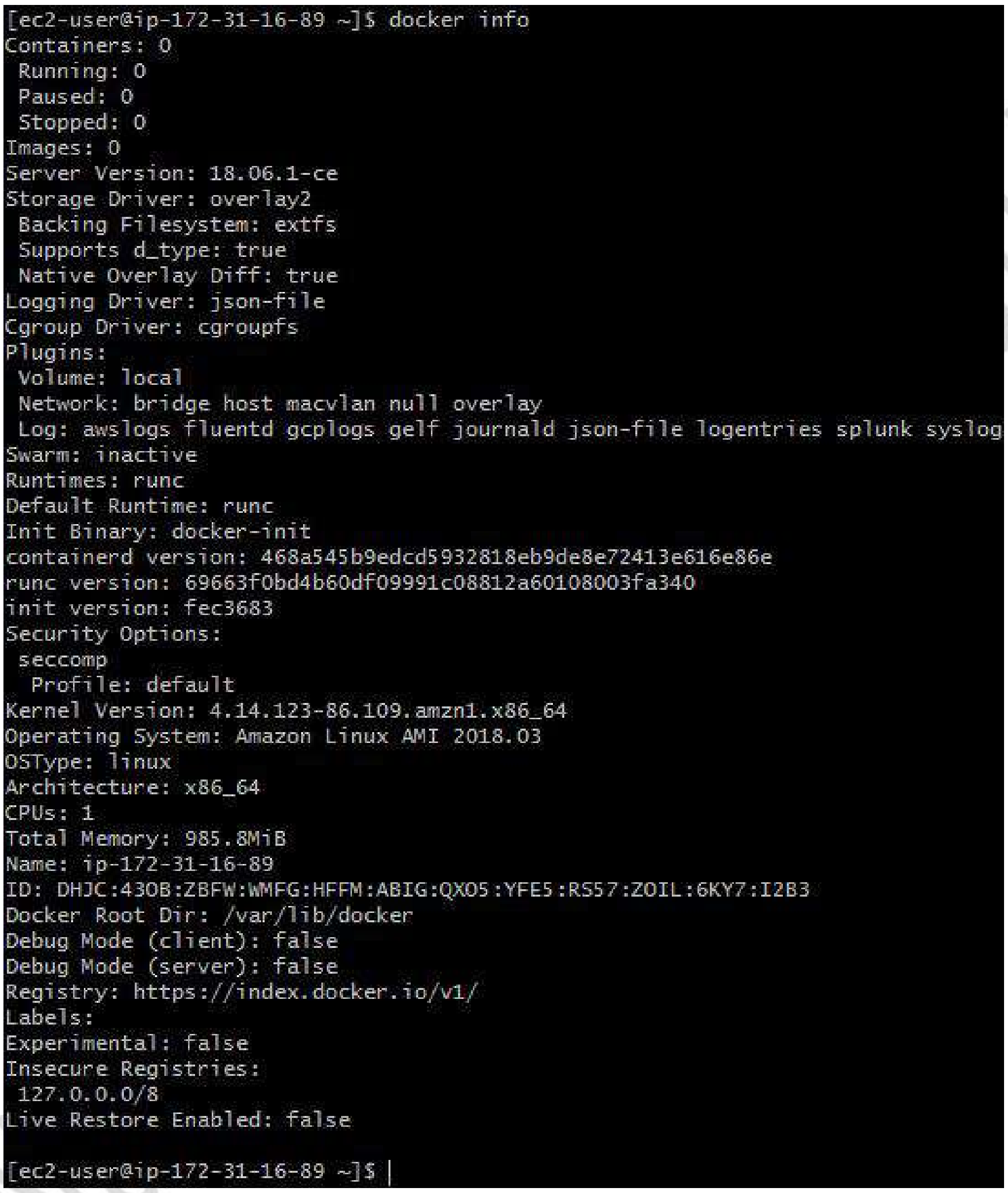


```
[ec2-user@ip-172-31-16-89 ~]$ docker -v
Docker version 18.06.1-ce, build e68fc7a215d7133c34aa18e3b72b4a21fd0c6136
```

Docker CE (Community Edition) is the simple classical OSS (Open Source Software) i.e. **Free**

Docker EE (Enterprise Edition) is Docker CE with certification on some systems and support by Docker i.e. **Licensed version and user have to buy it**

6. To check all information (version/container state/Images/Storage driver etc.)



```
[ec2-user@ip-172-31-16-89 ~]$ docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 18.06.1-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 468a545b9edcd5932818eb9de8e72413e616e86e
runc version: 69663f0bd4b60df09991c08812a60108003fa340
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.14.123-86.109.amzn1.x86_64
Operating System: Amazon Linux AMI 2018.03
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 985.8MiB
Name: ip-172-31-16-89
ID: DHJC:43OB:ZBFW:WMFG:HFFM:ABIG:QXO5:YFE5:RS57:ZOIL:6KY7:I2B3
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

[ec2-user@ip-172-31-16-89 ~]$ |
```

- overrelay2 is default storage driver for docker.

What is image?

The docker image contain the all grouping into application that making as image and that image going to run as a container.

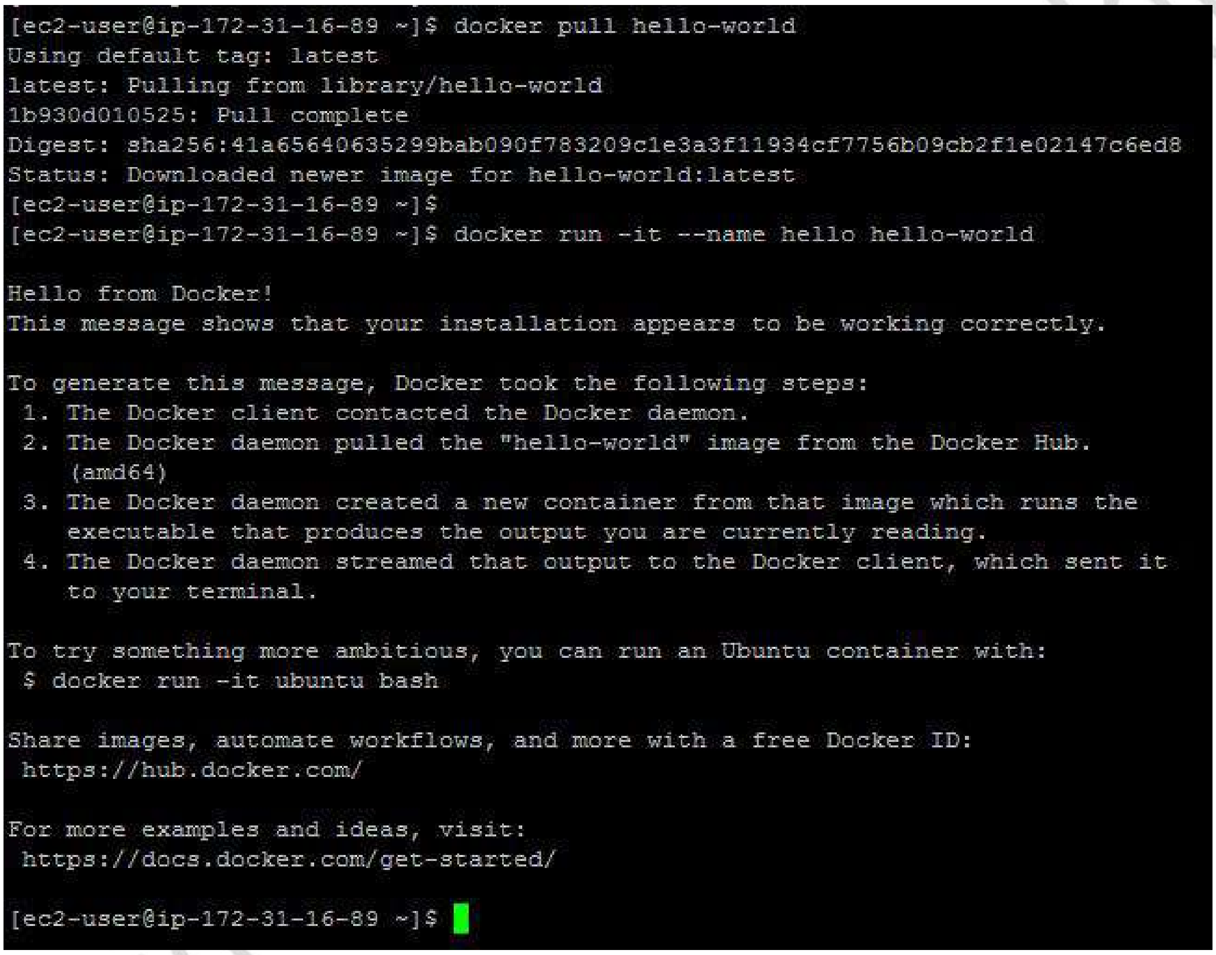
example: os + nginx this information added in docker file which create a simple image.

When we run image as container that time only it act as a real application. Images can't use directly.

==

To check the docker is working correctly or not.

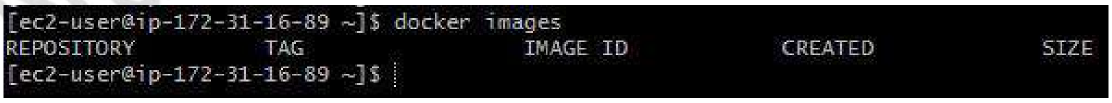
```
docker pull hello-world       #Its simple hello-world image
docker run -it --name hello hello-world   #It will only display the output.
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:41a65640635299bab090f783209c1e3a3f11934cf7756b09cb2f1e02147c6ed8
Status: Downloaded newer image for hello-world:latest
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ docker run -it --name hello hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[ec2-user@ip-172-31-16-89 ~]$
```

7. To check images in system or local images

docker images

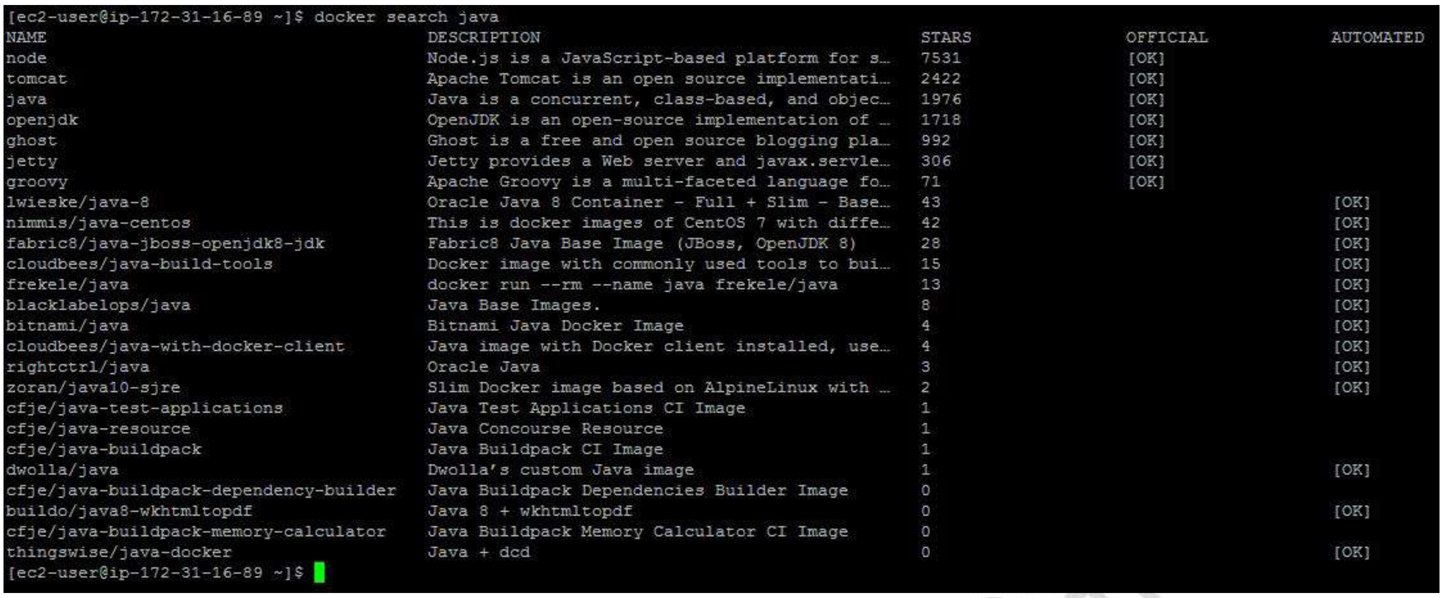


```
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[ec2-user@ip-172-31-16-89 ~]$
```

Repository -
Tag -
Image ID -
Created -
Size -

8. To search images which is available in docker hub publically.

docker search java (image name)

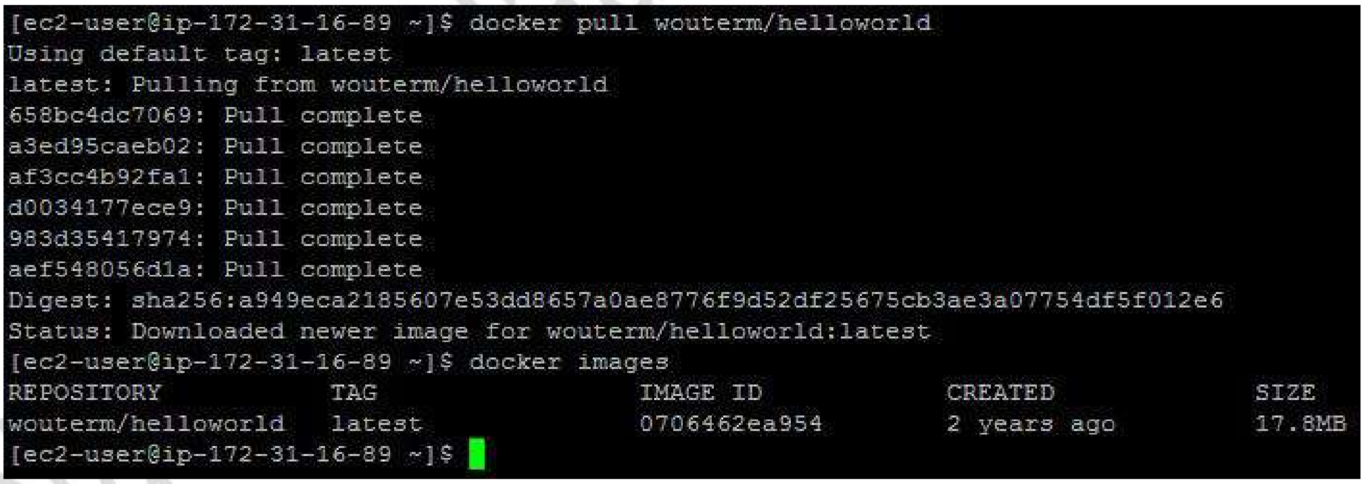


```
[ec2-user@ip-172-31-16-89 ~]$ docker search java
NAME                                     DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
node                                     Node.js is a JavaScript-based platform for s…   7531                [OK]
tomcat                                   Apache Tomcat is an open source implementati…   2422                [OK]
java                                     Java is a concurrent, class-based, and objec…   1976                [OK]
openjdk                                  OpenJDK is an open-source implementation of …   1718                [OK]
ghost                                    Ghost is a free and open source blogging pla…   992                 [OK]
jetty                                    Jetty provides a Web server and javax.servle…   306                 [OK]
groovy                                   Apache Groovy is a multi-faceted language fo…   71                  [OK]
lwieske/java-8                           Oracle Java 8 Container - Full + Slim - Base…   43                                      [OK]
nimmis/java-centos                       This is docker images of CentOS 7 with diffe…   42                                      [OK]
fabric8/java-jboss-openjdk8-jdk          Fabric8 Java Base Image (JBoss, OpenJDK 8)      28                                      [OK]
cloudbees/java-build-tools               Docker image with commonly used tools to bui…   15                                      [OK]
frekele/java                             docker run --rm --name java frekele/java        13                                      [OK]
blacklabelops/java                       Java Base Images.                               8                                       [OK]
bitnami/java                             Bitnami Java Docker Image                       4                                       [OK]
cloudbees/java-with-docker-client        Java image with Docker client installed, use…   4                                       [OK]
rightctrl/java                           Oracle Java                                     3                                       [OK]
zoran/java10-sjre                        Slim Docker image based on AlpineLinux with …   2                                       [OK]
cfje/java-test-applications              Java Test Applications CI Image                 1
cfje/java-resource                       Java Concourse Resource                         1
cfje/java-buildpack                      Java Buildpack CI Image                         1
dwolla/java                              Dwolla's custom Java image                      1                                       [OK]
cfje/java-buildpack-dependency-builder   Java Buildpack Dependencies Builder Image       0
buildo/java8-wkhtmltopdf                 Java 8 + wkhtmltopdf                            0                                       [OK]
cfje/java-buildpack-memory-calculator    Java Buildpack Memory Calculator CI Image       0
thingswise/java-docker                   Java + dcd                                      0                                       [OK]
[ec2-user@ip-172-31-16-89 ~]$
```

```
Name -
Description -
Stars -
Official -
Automated -
```

9. To pull the images

```
docker images
docker search helloworld
docker pull helloworld
docker images
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker pull wouterm/helloworld
Using default tag: latest
latest: Pulling from wouterm/helloworld
658bc4dc7069: Pull complete
a3ed95caeb02: Pull complete
af3cc4b92fa1: Pull complete
d0034177ece9: Pull complete
983d35417974: Pull complete
aef548056d1a: Pull complete
Digest: sha256:a949eca2185607e53dd8657a0ae8776f9d52df25675cb3ae3a07754df5f012e6
Status: Downloaded newer image for wouterm/helloworld:latest
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
wouterm/helloworld  latest              0706462ea954        2 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$
```

Note: If you won't' mentioned the version docker will download the latest version from docker public repository. you can find about version details in the tag. (tag=version)

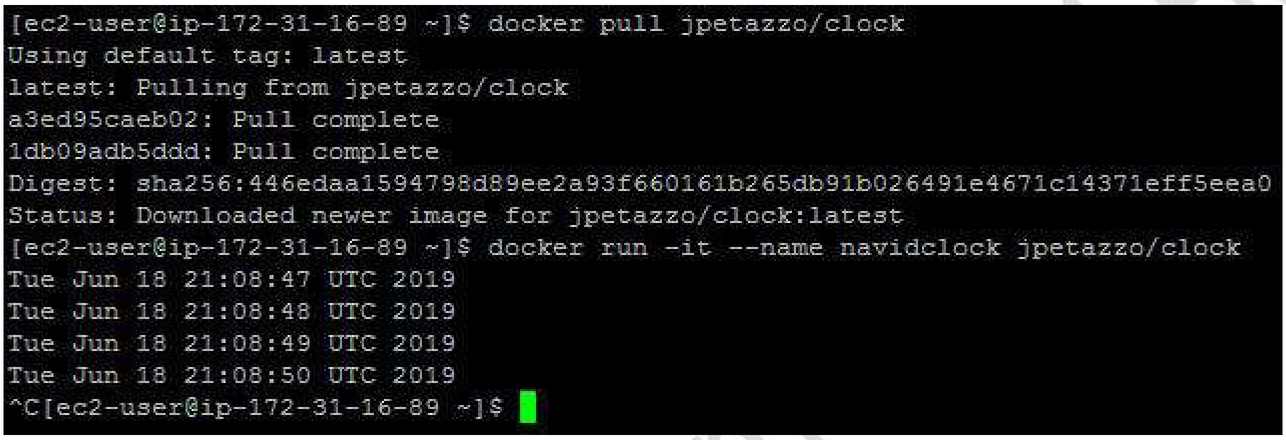
In above example

```
658bc4dc7069: Pull complete
a3ed95caeb02: Pull complete
af3cc4b92fa1: Pull complete
d0034177ece9: Pull complete
983d35417974: Pull complete
aef548056d1a: Pull complete
```

These are layers like caches if you already have it then next time when you run "docker pull <imagename> it won't download same layer. It simply uses layer which you have locally.

10. To run the container

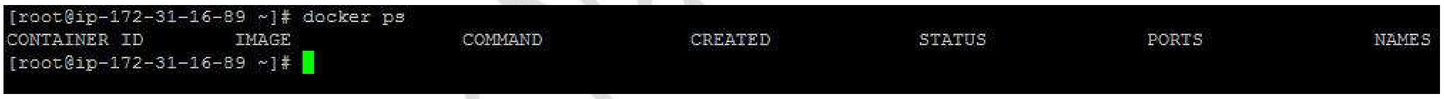
```
docker pull jpetazzo/clock
docker run -it --name navidclock jpetazzo/clock

OR

docker run -it --name jpetazzo/clock #Docker itself gives a name to container

docker run
      -i :  Input
      -t :  Terminal which you want to open.
   --name:  Name of the container " --name <containername>
 j*/clock:  Image which you want to use for container.

      a) To terminate running container Ctrl+c
      b) To exit from container without terminating it (run in background) Ctrl+p+q
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker pull jpetazzo/clock
Using default tag: latest
latest: Pulling from jpetazzo/clock
a3ed95caeb02: Pull complete
1db09adb5ddd: Pull complete
Digest: sha256:446edaa1594798d89ee2a93f660161b265db91b026491e4671c14371eff5eea0
Status: Downloaded newer image for jpetazzo/clock:latest
[ec2-user@ip-172-31-16-89 ~]$ docker run -it --name navidclock jpetazzo/clock
Tue Jun 18 21:08:47 UTC 2019
Tue Jun 18 21:08:48 UTC 2019
Tue Jun 18 21:08:49 UTC 2019
Tue Jun 18 21:08:50 UTC 2019
^C[ec2-user@ip-172-31-16-89 ~]$
```

11. To list the running containers

```
docker ps
```

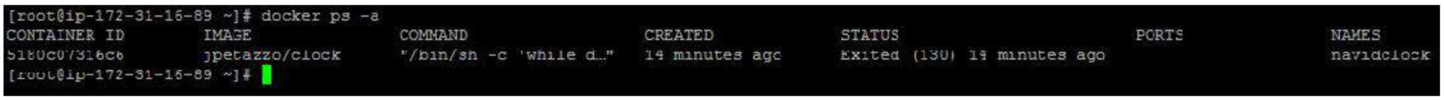


```
[root@ip-172-31-16-89 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@ip-172-31-16-89 ~]#
```

```
Container ID -
Image -
Command -
Created -
Status -
Ports -
Names -
```

12. To list all the containers (running/stopped etc)

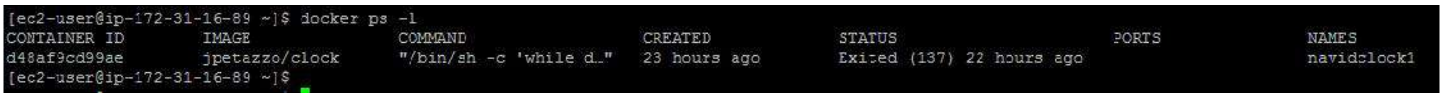
```
docker ps -a

OR

docker ps -a | grep -i exited (Stopped)
docker ps -a | grep -i up (running)
docker ps -a | grep -i container id
```



```
[root@ip-172-31-16-89 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS               NAMES
5180c07316c6        jpetazzo/clock      "/bin/sh -c 'while d…"   14 minutes ago      Exited (130) 14 minutes ago                      navidclock
[root@ip-172-31-16-89 ~]#
```

13. To list the latest container only [Show n last created containers (includes all states)]

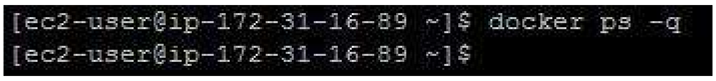
```
docker ps -l
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS               NAMES
d48af9cd99ae        jpetazzo/clock      "/bin/sh -c 'while d…"   23 hours ago        Exited (137) 22 hours ago                        navidclock1
[ec2-user@ip-172-31-16-89 ~]$
```

14. To list running container ID's only

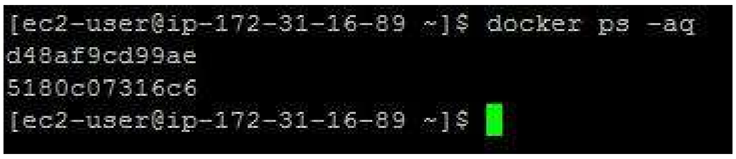
docker ps -q



```
[ec2-user@ip-172-31-16-89 ~]$ docker ps -q
[ec2-user@ip-172-31-16-89 ~]$
```

15. To list all container ID's (running/stopped etc)

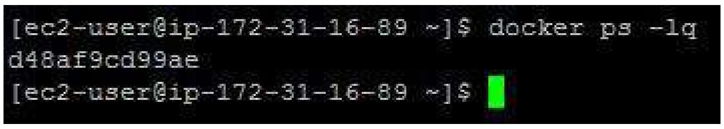
docker ps -aq



```
[ec2-user@ip-172-31-16-89 ~]$ docker ps -aq
d48af9cd99ae
5180c07316c6
[ec2-user@ip-172-31-16-89 ~]$
```

16. To list the latest container ID's only.

docker ps -lq

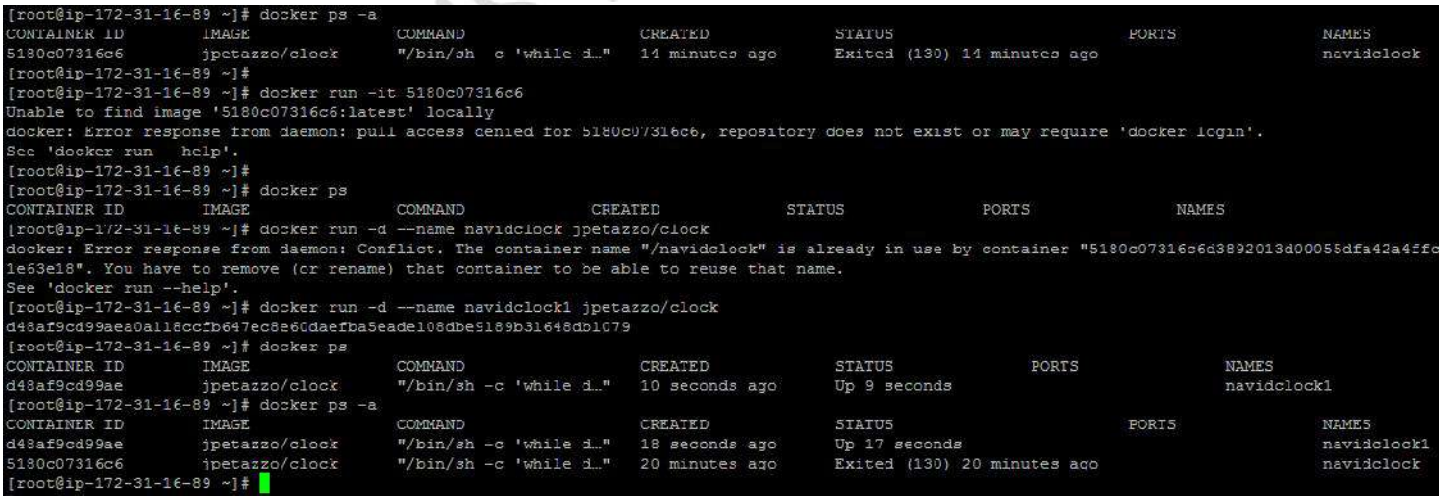


```
[ec2-user@ip-172-31-16-89 ~]$ docker ps -lq
d48af9cd99ae
[ec2-user@ip-172-31-16-89 ~]$
```

17. To run the container in background

```
docker ps -a
docker run -d --name navidclock jpetazzo/clock   ##It will show an error
docker run -d --name navidclock1 jpetazzo/clock ##Given unique name to container
docker ps
docker ps -a
```

-d: Detached mode
d48af9cd99aea0a118ccfb647ec8e60daefba5eade108dbe9189b31648db1079: Container running id

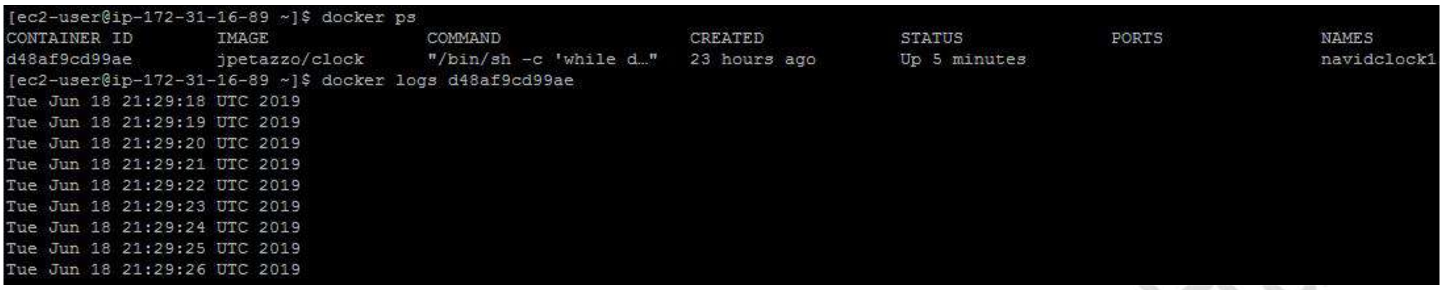


```
[root@ip-172-31-16-89 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS               NAMES
5180c07316c6        jpetazzo/clock      "/bin/sh -c 'while d…"   14 minutes ago      Exited (130) 14 minutes ago                      navidclock
[root@ip-172-31-16-89 ~]#
[root@ip-172-31-16-89 ~]# docker run -it 5180c07316c6
Unable to find image '5180c07316c6:latest' locally
docker: Error response from daemon: pull access denied for 5180c07316c6, repository does not exist or may require 'docker login'.
See 'docker run --help'.
[root@ip-172-31-16-89 ~]#
[root@ip-172-31-16-89 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@ip-172-31-16-89 ~]# docker run -d --name navidclock jpetazzo/clock
docker: Error response from daemon: Conflict. The container name "/navidclock" is already in use by container "5180c07316c6d3892013d00055dfa42a4ffc
1e63e18". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[root@ip-172-31-16-89 ~]# docker run -d --name navidclock1 jpetazzo/clock
d48af9cd99aea0a118ccfb647ec8e60daefba5eade108dbe9189b31648db1079
[root@ip-172-31-16-89 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
d48af9cd99ae        jpetazzo/clock      "/bin/sh -c 'while d…"   10 seconds ago      Up 9 seconds                            navidclock1
[root@ip-172-31-16-89 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS               NAMES
d48af9cd99ae        jpetazzo/clock      "/bin/sh -c 'while d…"   18 seconds ago      Up 17 seconds                                    navidclock1
5180c07316c6        jpetazzo/clock      "/bin/sh -c 'while d…"   20 minutes ago      Exited (130) 20 minutes ago                      navidclock
[root@ip-172-31-16-89 ~]#
```

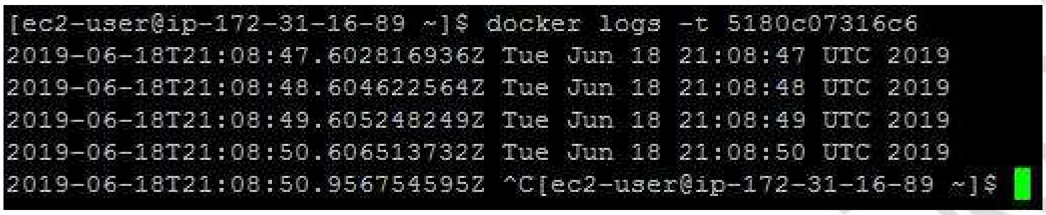
Note: Container name should be unique else it will show an error like above example.

18. To list the logs for container.

```
docker ps OR docker ps -a (Select container ID for which you want to see the logs)
docker logs <container ID>

OR

docker logs -t <container ID>
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
d48af9cd99ae        jpetazzo/clock      "/bin/sh -c 'while d…"   23 hours ago        Up 5 minutes                            navidclock1
[ec2-user@ip-172-31-16-89 ~]$ docker logs d48af9cd99ae
Tue Jun 18 21:29:18 UTC 2019
Tue Jun 18 21:29:19 UTC 2019
Tue Jun 18 21:29:20 UTC 2019
Tue Jun 18 21:29:21 UTC 2019
Tue Jun 18 21:29:22 UTC 2019
Tue Jun 18 21:29:23 UTC 2019
Tue Jun 18 21:29:24 UTC 2019
Tue Jun 18 21:29:25 UTC 2019
Tue Jun 18 21:29:26 UTC 2019
```



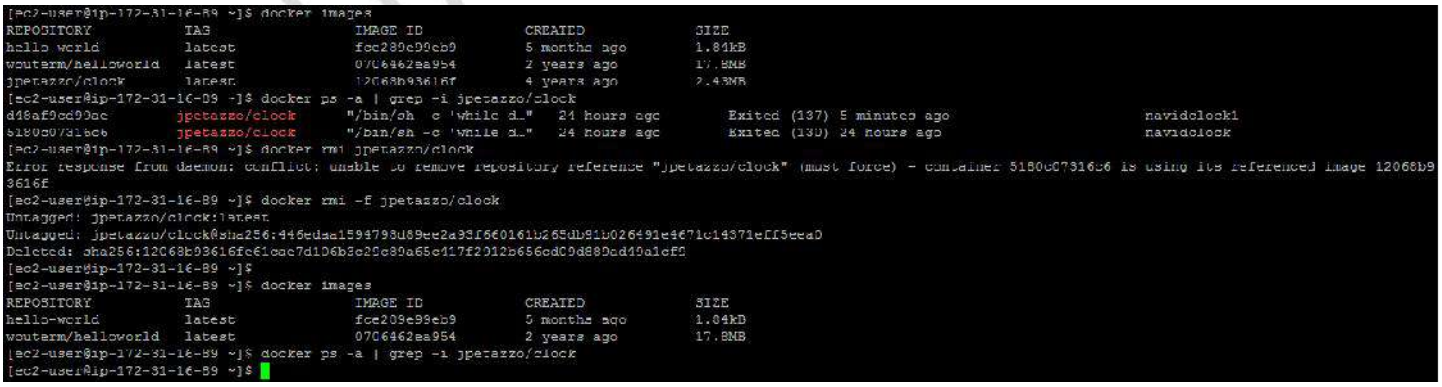
```
[ec2-user@ip-172-31-16-89 ~]$ docker logs -t 5180c07316c6
2019-06-18T21:08:47.602816936Z Tue Jun 18 21:08:47 UTC 2019
2019-06-18T21:08:48.604622564Z Tue Jun 18 21:08:48 UTC 2019
2019-06-18T21:08:49.605248249Z Tue Jun 18 21:08:49 UTC 2019
2019-06-18T21:08:50.606513732Z Tue Jun 18 21:08:50 UTC 2019
2019-06-18T21:08:50.956754595Z ^C[ec2-user@ip-172-31-16-89 ~]$
```

19. To list real time logs from container.

```
docker ps or docker ps -a (Select container ID for which you want to see the logs)

docker logs -f <container ID>
```
#Live logs can't be capture in artifact ... :-)

20. To delete the image

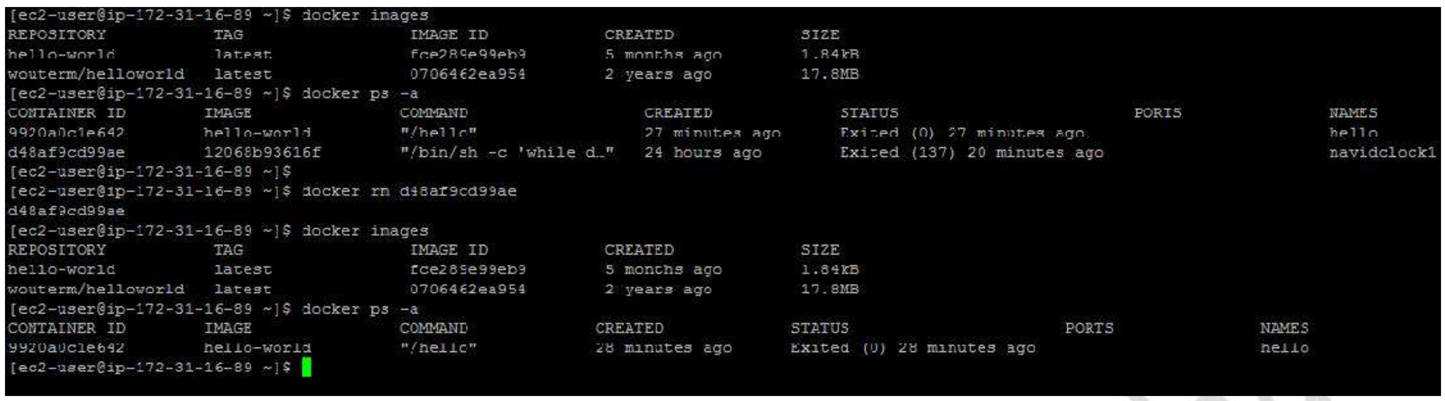
```
docker images
docker ps -a | grep -i <imagename>  #To list container associated with image
docker rmi <imagename> # If any container associated with image it will show an error

docker rmi -f <imagename> #It will forcefully remove the images & container associate it

docker images
docker ps -a | grep -i <imagename>  #To list container associated with image
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY            TAG                 IMAGE ID            CREATED             SIZE
hello-world           latest              fce289e99eb9        5 months ago        1.84kB
wouterm/helloworld    latest              0706462ea954        2 years ago         17.8MB
jpetazzo/clock        latest              12068b93616f        4 years ago         2.43MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a | grep -i jpetazzo/clock
d48af9cd99ae        jpetazzo/clock      "/bin/sh -c 'while d…"   24 hours ago        Exited (137) 5 minutes ago                       navidclock1
5180c07316c6        jpetazzo/clock      "/bin/sh -c 'while d…"   24 hours ago        Exited (130) 24 hours ago                        navidclock
[ec2-user@ip-172-31-16-89 ~]$ docker rmi jpetazzo/clock
Error response from daemon: conflict: unable to remove repository reference "jpetazzo/clock" (must force) - container 5180c07316c6 is using its referenced image 12068b9
3616f
[ec2-user@ip-172-31-16-89 ~]$ docker rmi -f jpetazzo/clock
Untagged: jpetazzo/clock:latest
Untagged: jpetazzo/clock@sha256:446edaa1594798d89ee2a93f6e0161b265db91b026491e4e71c14371eff5eea0
Deleted: sha256:12068b93616fc61cac7d106b3c29c89a65c417f2912b656cd09d889ad49a1cf2
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY            TAG                 IMAGE ID            CREATED             SIZE
hello-world           latest              fce289e99eb9        5 months ago        1.84kB
wouterm/helloworld    latest              0706462ea954        2 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a | grep -i jpetazzo/clock
[ec2-user@ip-172-31-16-89 ~]$
```

21. To delete container (only delete the container not the image)

```
docker ps -a
docker rm <container ID>
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
wouterm/helloworld  latest              0706462ea954        2 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                         PORTS               NAMES
9920a0c1e642        hello-world         "/hello"                 27 minutes ago      Exited (0) 27 minutes ago                          hello
d48af9cd99ae        12068b93616f        "/bin/sh -c 'while d…"   24 hours ago        Exited (137) 20 minutes ago                        navidclock1
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ docker rm d48af9cd99ae
d48af9cd99ae
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
wouterm/helloworld  latest              0706462ea954        2 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
9920a0c1e642        hello-world         "/hello"            28 minutes ago      Exited (0) 28 minutes ago                       hello
[ec2-user@ip-172-31-16-89 ~]$
```

===

What is tutum?

https://hub.docker.com/u/tutum

In the past, a lot of Docker users have resorted to patching together brittle custom scripts to deploy their apps to production. Tutum removes the need for this with their “Orchestration-as-a-Service.”

Docker Acquires Tutum To Help IT Teams Deploy and Manage Production Apps

Tutum were early members of the Docker community. Their contributions are many, including curating popular images on Docker Hub, leading Docker meetups in NYC and Madrid, participating in all three DockerCons, and contributing to the Docker project.
===

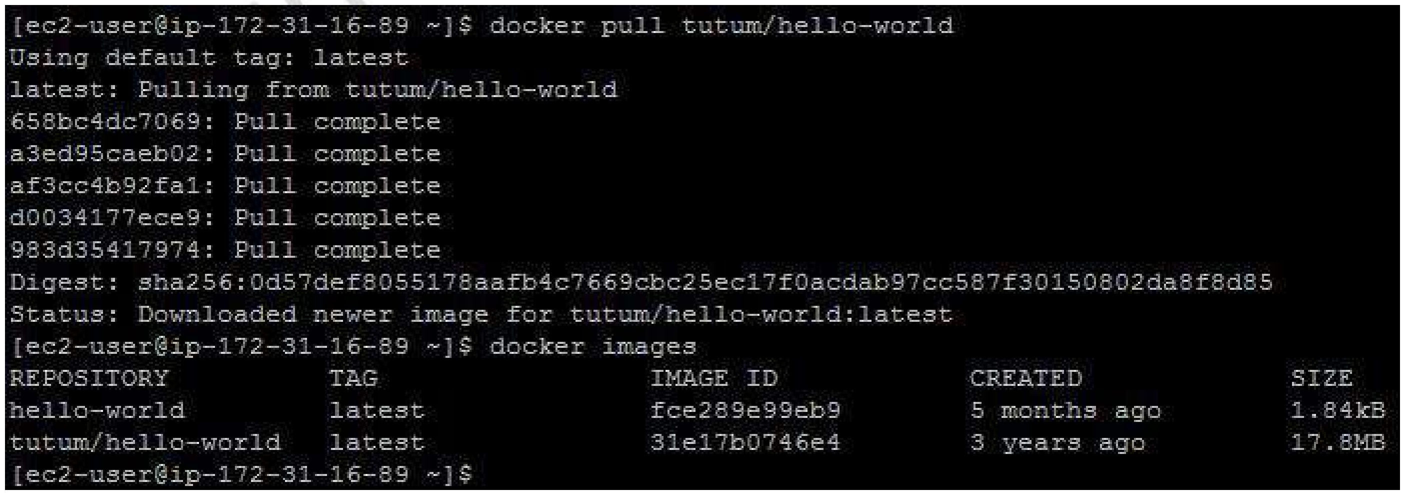
<u>Deploy web application on docker using Port forwarding</u>

- Download web application/images from https://hub.docker.com/r/tutum/hello-world/

<u>What is port forwarding?</u>

To make a port available to services outside of Docker, or to Docker containers which are not connected to the container's network, use the --publish or -p flag. This creates a firewall rule which maps a container port to a port on the Docker host.

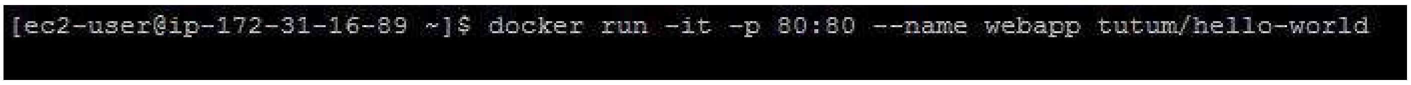
1. docker pull tutum/hello-world



```
[ec2-user@ip-172-31-16-89 ~]$ docker pull tutum/hello-world
Using default tag: latest
latest: Pulling from tutum/hello-world
658bc4dc7069: Pull complete
a3ed95caeb02: Pull complete
af3cc4b92fa1: Pull complete
d0034177ece9: Pull complete
983d35417974: Pull complete
Digest: sha256:0d57def8055178aafb4c7669cbc25ec17f0acdab97cc587f30150802da8f8d85
Status: Downloaded newer image for tutum/hello-world:latest
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
tutum/hello-world   latest              31e17b0746e4        3 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$
```

```
2. docker run -it -p <exposed port>:<internal port> --name <container name> <imagename>
```

Example 01:



```
[ec2-user@ip-172-31-16-89 ~]$ docker run -it -p 80:80 --name webapp tutum/hello-world
```

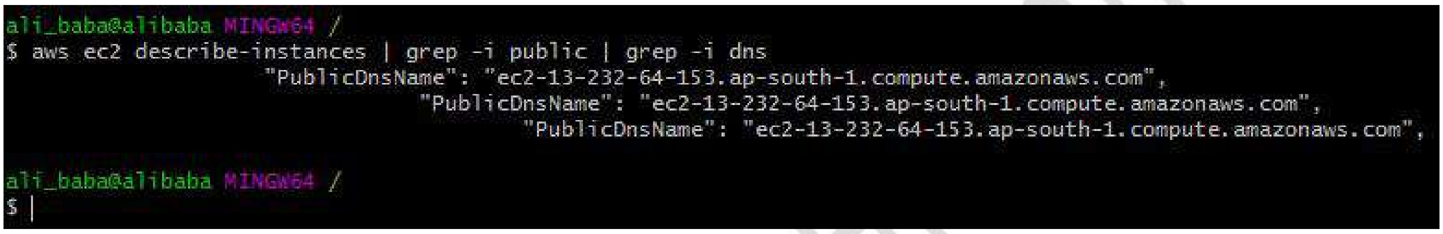
Note: you will not get prompt unless until you access it.

```
-i: Interactive
-t: Terminal
-p: port
80: External port which is expose and access by public network.
80: It's internal port not exposed and can't be access by public network.
```

Note: Exposed Port details should be in both inbound and outbound for Security group.

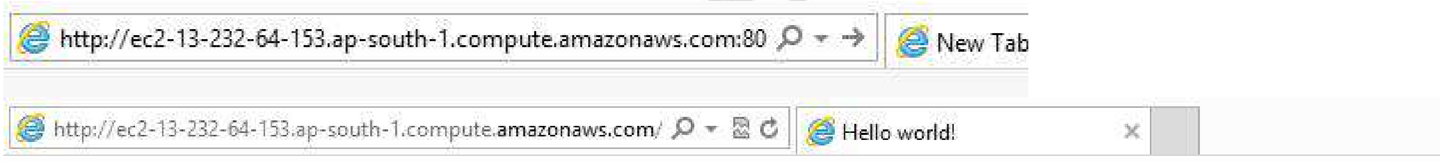
Inbound is enough .------> Pending to check

Select the public DNS name from GUI console/CLI and enter it in browser with :80 (Exposed port for this example)

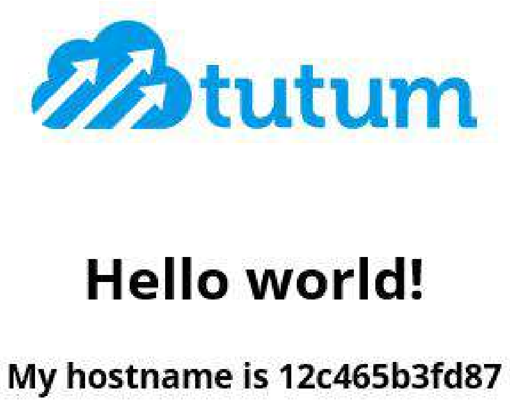


```
ali_baba@alibaba MINGW64 /
$ aws ec2 describe-instances | grep -i public | grep -i dns
                    "PublicDnsName": "ec2-13-232-64-153.ap-south-1.compute.amazonaws.com",
                                "PublicDnsName": "ec2-13-232-64-153.ap-south-1.compute.amazonaws.com",
                                        "PublicDnsName": "ec2-13-232-64-153.ap-south-1.compute.amazonaws.com",

ali_baba@alibaba MINGW64 /
$ |
```

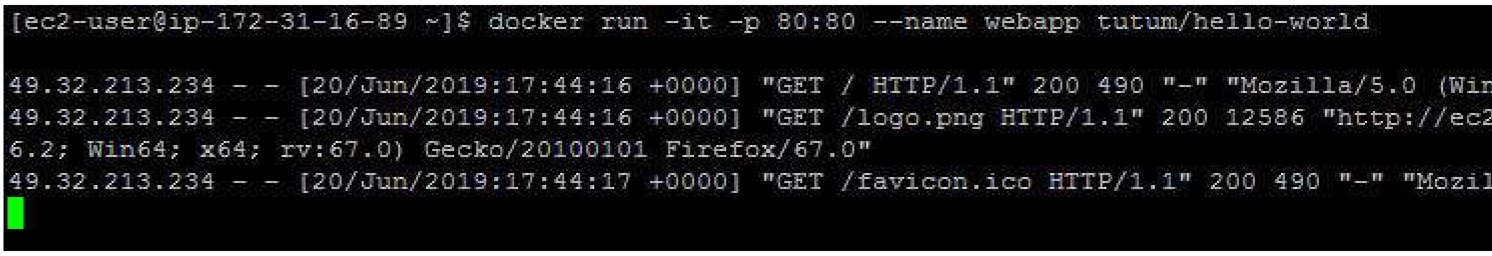
Enter the public dnsname with exposed port 80 -



http://ec2-13-232-64-153.ap-south-1.compute.amazonaws.com:80
New Tab
http://ec2-13-232-64-153.ap-south-1.compute.amazonaws.com/
Hello world!



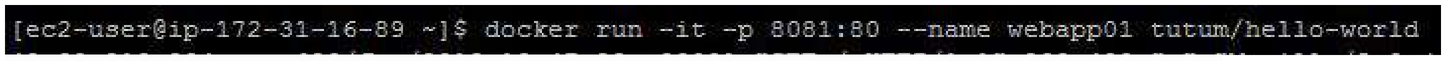
Once you started the browsing you. you will started to get logs on console side -



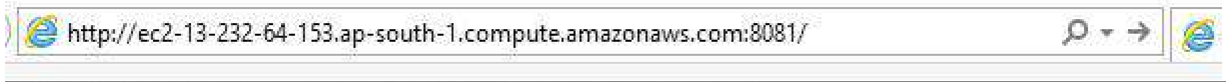
```
[ec2-user@ip-172-31-16-89 ~]$ docker run -it -p 80:80 --name webapp tutum/hello-world

49.32.213.234 - - [20/Jun/2019:17:44:16 +0000] "GET / HTTP/1.1" 200 490 "-" "Mozilla/5.0 (Win
49.32.213.234 - - [20/Jun/2019:17:44:16 +0000] "GET /logo.png HTTP/1.1" 200 12586 "http://ec2
6.2; Win64; x64; rv:67.0) Gecko/20100101 Firefox/67.0"
49.32.213.234 - - [20/Jun/2019:17:44:17 +0000] "GET /favicon.ico HTTP/1.1" 200 490 "-" "Mozil
```

Example 02: exposed port 8081.

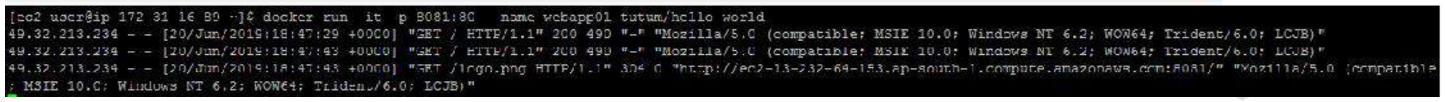
```
docker run -it -p 8081:80 --name webapp01 tutum/hello-world
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker run -it -p 8081:80 --name webapp01 tutum/hello-world
```



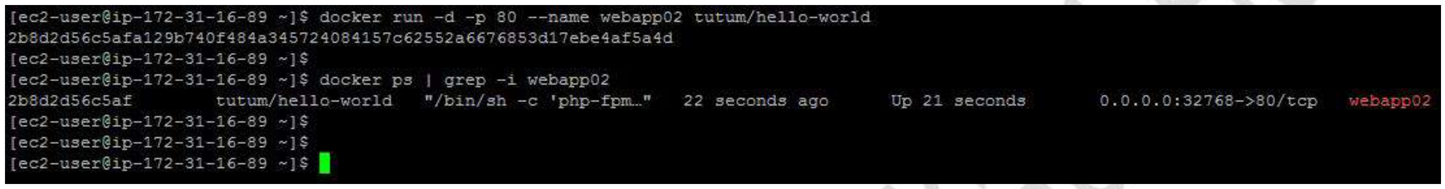
http://ec2-13-232-64-153.ap-south-1.compute.amazonaws.com:8081/



```
[ec2 user@ip 172 31 16 89 ~]$ docker run  it  p 8081:80   name webapp01 tutum/hello world
49.32.213.234 - - [20/Jun/2019:18:47:29 +0000] "GET / HTTP/1.1" 200 490 "-" "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; LCJB)"
49.32.213.234 - - [20/Jun/2019:18:47:43 +0000] "GET / HTTP/1.1" 200 490 "-" "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; LCJB)"
49.32.213.234 - - [20/Jun/2019:18:47:43 +0000] "GET /logo.png HTTP/1.1" 304 0 "http://ec2-13-232-64-153.ap-south-1.compute.amazonaws.com:8081/" "Mozilla/5.0 (compatible
; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; LCJB)"
```

Example 03: Without any exposed port

```
docker run -d -p 80 --name webapp02 tutum/hello-world
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker run -d -p 80 --name webapp02 tutum/hello-world
2b8d2d56c5afa129b740f484a345724084157c62552a6676853d17ebe4af5a4d
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ docker ps | grep -i webapp02
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   22 seconds ago      Up 21 seconds       0.0.0.0:32768->80/tcp   webapp02
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$
```

Note:

In above example we won't mentioned the exposed port so docker will assign one port for us. If you tried to access it via browser page will not displayed as our SG don't have 32768 port in its rules. Update the SG so it will work.
===

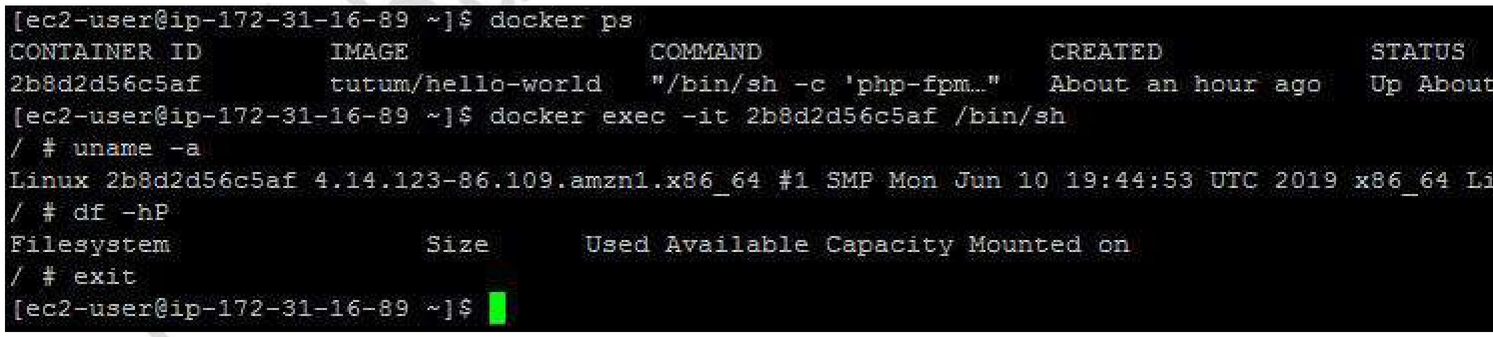
To interact with docker container/ To login in container

Docker exec only interacts with containers that are actively running.

If there is an existing container that was started headless (such as by docker-compose), you can easily drop into a shell to check on the state of things:

```
docker ps
docker exec -it CONTAINER COMMAND
```

To exit from running container - run "exit" command



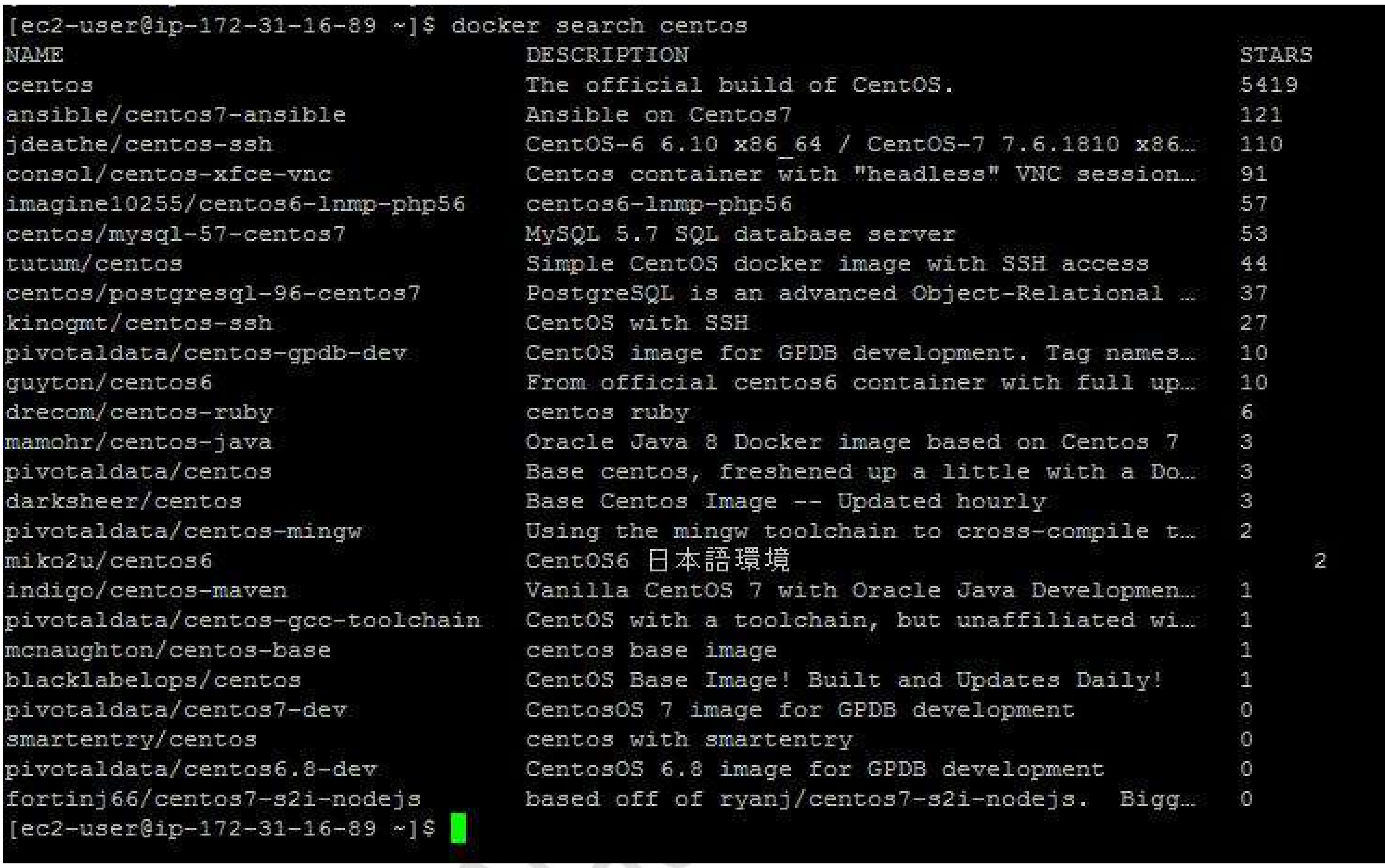
```
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   About an hour ago   Up About
[ec2-user@ip-172-31-16-89 ~]$ docker exec -it 2b8d2d56c5af /bin/sh
/ # uname -a
Linux 2b8d2d56c5af 4.14.123-86.109.amzn1.x86_64 #1 SMP Mon Jun 10 19:44:53 UTC 2019 x86_64 Li
/ # df -hP
Filesystem                Size      Used Available Capacity Mounted on
/ # exit
[ec2-user@ip-172-31-16-89 ~]$
```

===

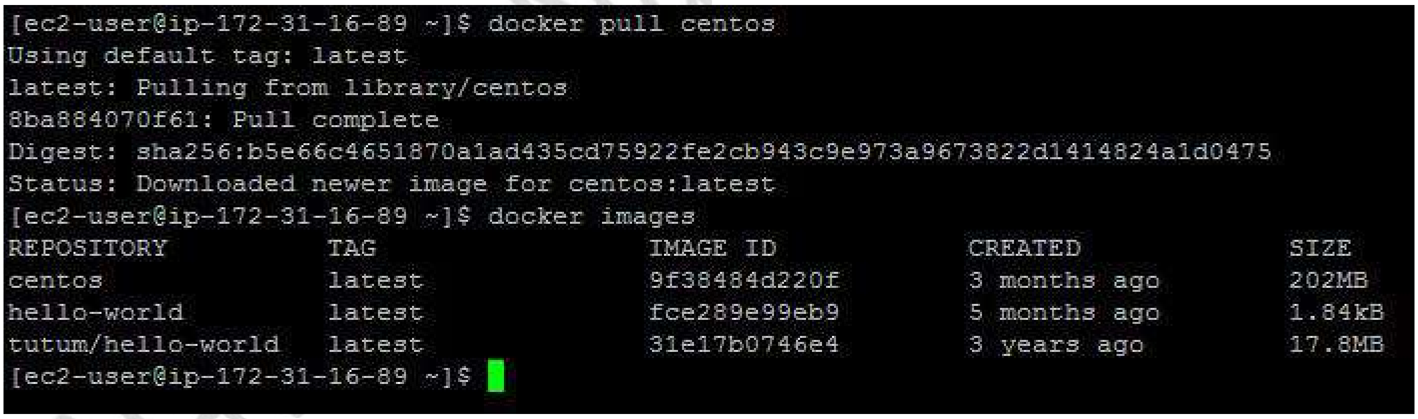
Install packages on container and create custom image

1. Download image from docker hub ex: centos

```
docker search <imagename>
docker pull <imagename>
```



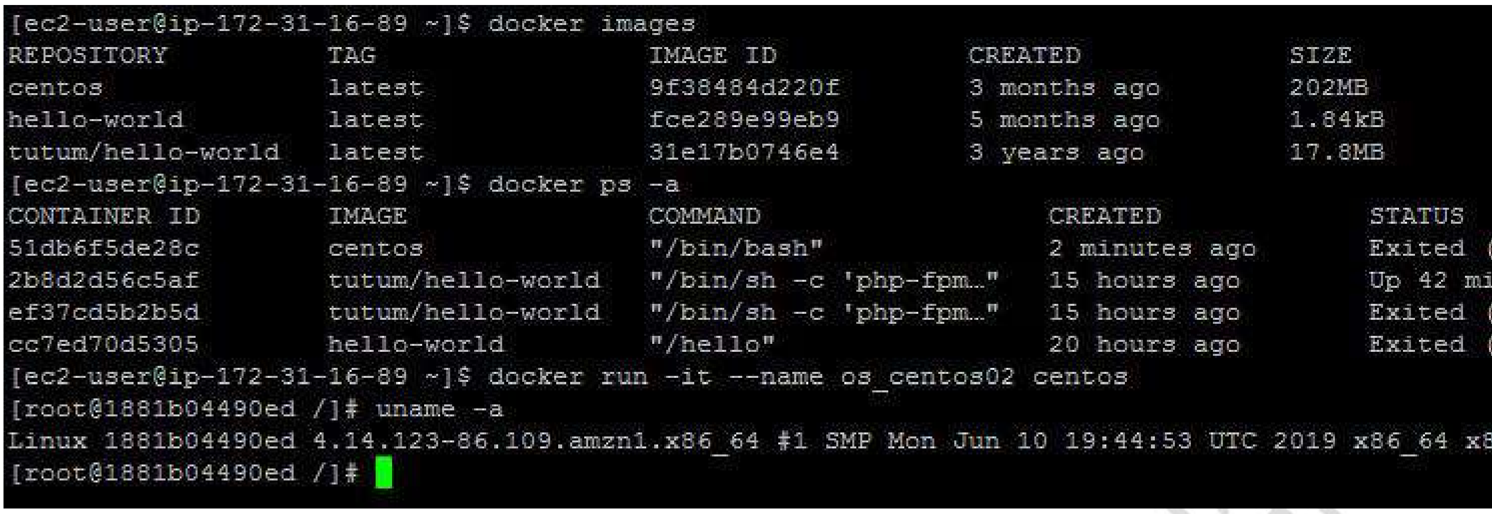
```
[ec2-user@ip-172-31-16-89 ~]$ docker search centos
NAME                               DESCRIPTION                                     STARS
centos                             The official build of CentOS.                   5419
ansible/centos7-ansible            Ansible on Centos7                              121
jdeathe/centos-ssh                 CentOS-6 6.10 x86_64 / CentOS-7 7.6.1810 x86…   110
consol/centos-xfce-vnc             Centos container with "headless" VNC session…   91
imagine10255/centos6-lnmp-php56    centos6-lnmp-php56                              57
centos/mysql-57-centos7            MySQL 5.7 SQL database server                   53
tutum/centos                       Simple CentOS docker image with SSH access      44
centos/postgresql-96-centos7       PostgreSQL is an advanced Object-Relational …   37
kinogmt/centos-ssh                 CentOS with SSH                                 27
pivotaldata/centos-gpdb-dev        CentOS image for GPDB development. Tag names…   10
guyton/centos6                     From official centos6 container with full up…   10
drecom/centos-ruby                 centos ruby                                     6
mamohr/centos-java                 Oracle Java 8 Docker image based on Centos 7    3
pivotaldata/centos                 Base centos, freshened up a little with a Do…   3
darksheer/centos                   Base Centos Image -- Updated hourly             3
pivotaldata/centos-mingw           Using the mingw toolchain to cross-compile t…   2
miko2u/centos6                     CentOS6 日本語環境                                   2
indigo/centos-maven                Vanilla CentOS 7 with Oracle Java Developmen…   1
pivotaldata/centos-gcc-toolchain   CentOS with a toolchain, but unaffiliated wi…   1
mcnaughton/centos-base             centos base image                               1
blacklabelops/centos               CentOS Base Image! Built and Updates Daily!     1
pivotaldata/centos7-dev            CentosOS 7 image for GPDB development           0
smartentry/centos                  centos with smartentry                          0
pivotaldata/centos6.8-dev          CentosOS 6.8 image for GPDB development         0
fortinj66/centos7-s2i-nodejs       based off of ryanj/centos7-s2i-nodejs.  Bigg…   0
[ec2-user@ip-172-31-16-89 ~]$
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker pull centos
Using default tag: latest
latest: Pulling from library/centos
8ba884070f61: Pull complete
Digest: sha256:b5e66c4651870a1ad435cd75922fe2cb943c9e973a9673822d1414824a1d0475
Status: Downloaded newer image for centos:latest
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              9f38484d220f        3 months ago        202MB
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
tutum/hello-world   latest              31e17b0746e4        3 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$
```

2. Run the container with -it option as os image won't worked with -d option.

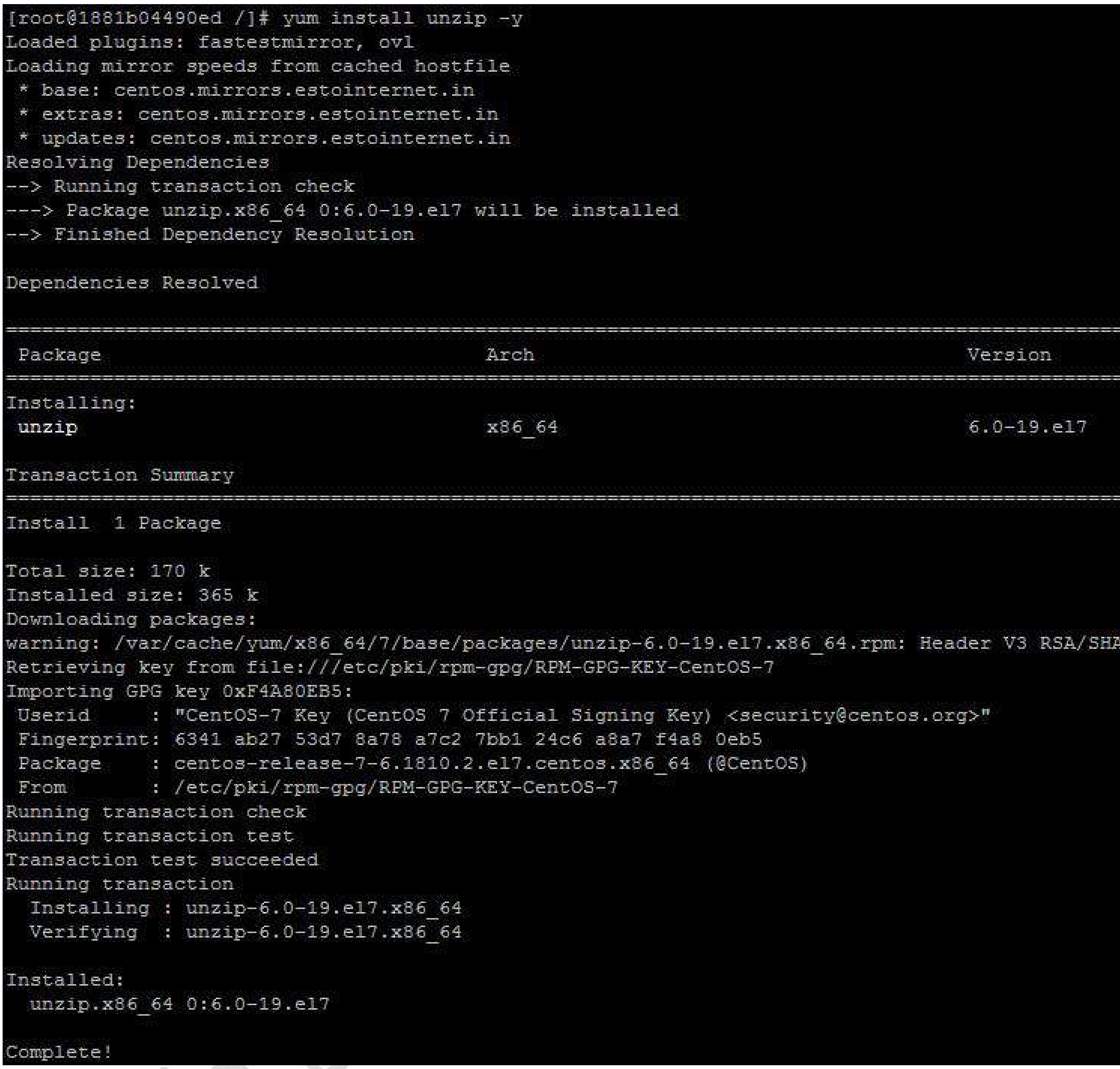
```
docker run -it --name <containername> <image>
      root@containerid#uname -a     ------> You will be in container
```



```
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              9f38484d220f        3 months ago        202MB
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
tutum/hello-world   latest              31e17b0746e4        3 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
51db6f5de28c        centos              "/bin/bash"              2 minutes ago       Exited (
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   15 hours ago        Up 42 mi
ef37cd5b2b5d        tutum/hello-world   "/bin/sh -c 'php-fpm…"   15 hours ago        Exited (
cc7ed70d5305        hello-world         "/hello"                 20 hours ago        Exited (
[ec2-user@ip-172-31-16-89 ~]$ docker run -it --name os_centos02 centos
[root@1881b04490ed /]# uname -a
Linux 1881b04490ed 4.14.123-86.109.amzn1.x86_64 #1 SMP Mon Jun 10 19:44:53 UTC 2019 x86_64 x8
[root@1881b04490ed /]#
```

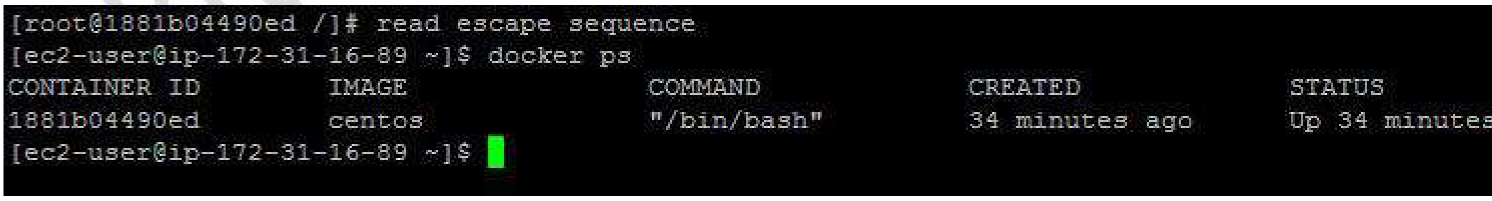
3. Install package on container ex:ungzip like we install on our vm machine.

```
yum update -y
yum install unzip -y
```



```
[root@1881b04490ed /]# yum install unzip -y
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: centos.mirrors.estointernet.in
 * extras: centos.mirrors.estointernet.in
 * updates: centos.mirrors.estointernet.in
Resolving Dependencies
--> Running transaction check
---> Package unzip.x86_64 0:6.0-19.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=============================================================================================
 Package                              Arch                                 Version
=============================================================================================
Installing:
 unzip                                x86_64                               6.0-19.el7

Transaction Summary
=============================================================================================
Install  1 Package

Total size: 170 k
Installed size: 365 k
Downloading packages:
warning: /var/cache/yum/x86_64/7/base/packages/unzip-6.0-19.el7.x86_64.rpm: Header V3 RSA/SHA
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Importing GPG key 0xF4A80EB5:
 Userid     : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
 Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
 Package    : centos-release-7-6.1810.2.el7.centos.x86_64 (@CentOS)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : unzip-6.0-19.el7.x86_64
  Verifying  : unzip-6.0-19.el7.x86_64

Installed:
  unzip.x86_64 0:6.0-19.el7

Complete!
```

4. If you want run container in background then exit from container without stopping it ctrl+p+q.

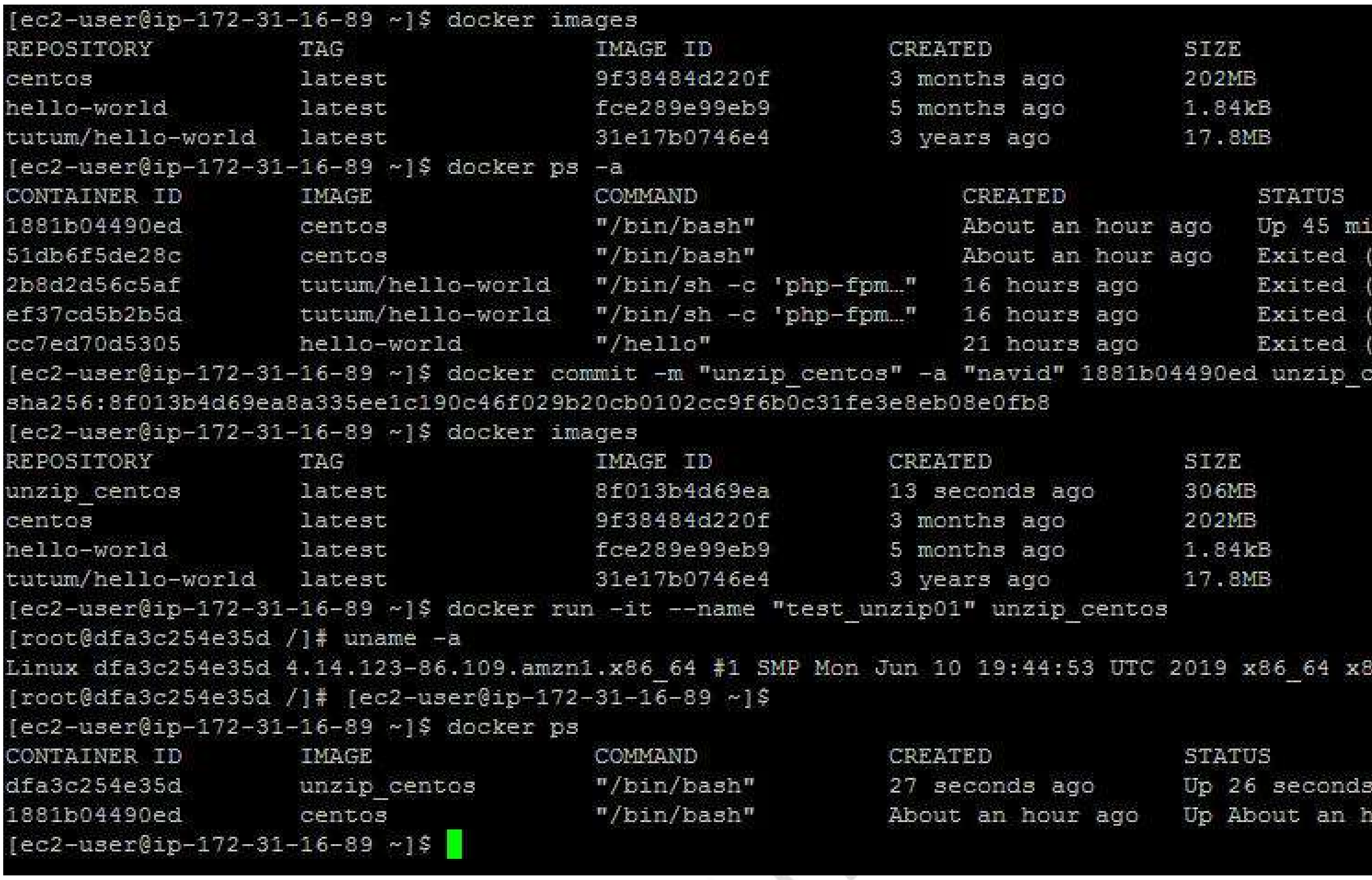


```
[root@1881b04490ed /]# read escape sequence
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
1881b04490ed        centos              "/bin/bash"         34 minutes ago      Up 34 minutes
[ec2-user@ip-172-31-16-89 ~]$
```

5. Create custom image

Here we are creating centos custom image with unzip package. Container state is not matter here.

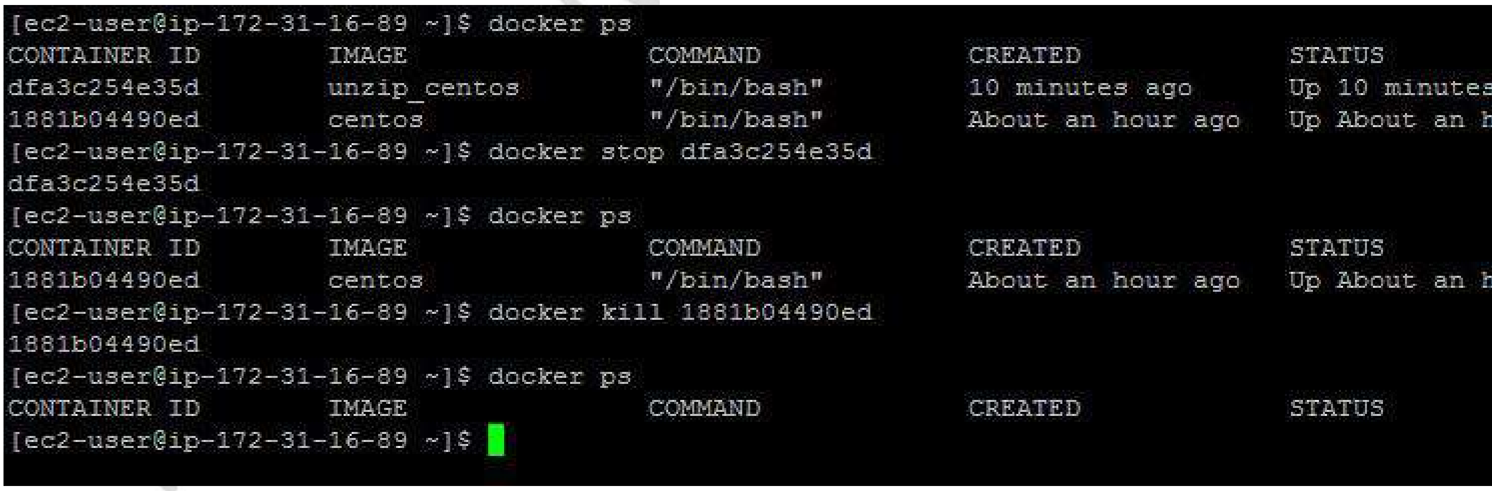
```
docker commit -m "<comment>" -a "<authername>" <containerid_f_image> <custom_image_name>
```


```
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              9f38484d220f        3 months ago        202MB
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
tutum/hello-world   latest              31e17b0746e4        3 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
1881b04490ed        centos              "/bin/bash"              About an hour ago   Up 45 mi
51db6f5de28c        centos              "/bin/bash"              About an hour ago   Exited (
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   16 hours ago        Exited (
ef37cd5b2b5d        tutum/hello-world   "/bin/sh -c 'php-fpm…"   16 hours ago        Exited (
cc7ed70d5305        hello-world         "/hello"                 21 hours ago        Exited (
[ec2-user@ip-172-31-16-89 ~]$ docker commit -m "unzip_centos" -a "navid" 1881b04490ed unzip_c
sha256:8f013b4d69ea8a335ee1c190c46f029b20cb0102cc9f6b0c31fe3e8eb08e0fb8
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
unzip_centos        latest              8f013b4d69ea        13 seconds ago      306MB
centos              latest              9f38484d220f        3 months ago        202MB
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
tutum/hello-world   latest              31e17b0746e4        3 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker run -it --name "test_unzip01" unzip_centos
[root@dfa3c254e35d /]# uname -a
Linux dfa3c254e35d 4.14.123-86.109.amzn1.x86_64 #1 SMP Mon Jun 10 19:44:53 UTC 2019 x86_64 x8
[root@dfa3c254e35d /]# [ec2-user@ip-172-31-16-89 ~]$
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
dfa3c254e35d        unzip_centos        "/bin/bash"         27 seconds ago      Up 26 seconds
1881b04490ed        centos              "/bin/bash"         About an hour ago   Up About an h
[ec2-user@ip-172-31-16-89 ~]$
```

==

To stop container

```
docker stop <containerid>     #Gracefully stop
docker kill <containerid>     #Will not wait to complete any container process. Its
                               forcefully stop the container
```


```
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
dfa3c254e35d        unzip_centos        "/bin/bash"         10 minutes ago      Up 10 minutes
1881b04490ed        centos              "/bin/bash"         About an hour ago   Up About an h
[ec2-user@ip-172-31-16-89 ~]$ docker stop dfa3c254e35d
dfa3c254e35d
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
1881b04490ed        centos              "/bin/bash"         About an hour ago   Up About an h
[ec2-user@ip-172-31-16-89 ~]$ docker kill 1881b04490ed
1881b04490ed
[ec2-user@ip-172-31-16-89 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
[ec2-user@ip-172-31-16-89 ~]$
```

==

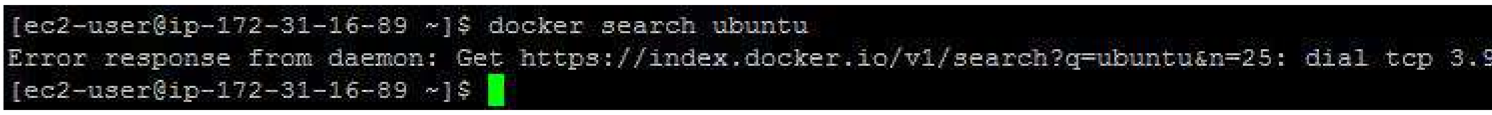
What is docker hub ?
Docker Hub is a cloud-based repository in which Docker users and partners create, test, store and distribute container images. Through Docker Hub, a user can access public, open source image repositories, as well as use a space to create their own private repositories, automated build functions, webhooks and work groups.

===

Troubleshooting:

Scenario 01:

Error response from daemon: Get https://index.docker.io/v1/search?q=ubuntu&n=25: dial tcp 3.91.211.1:443: i/o timeout

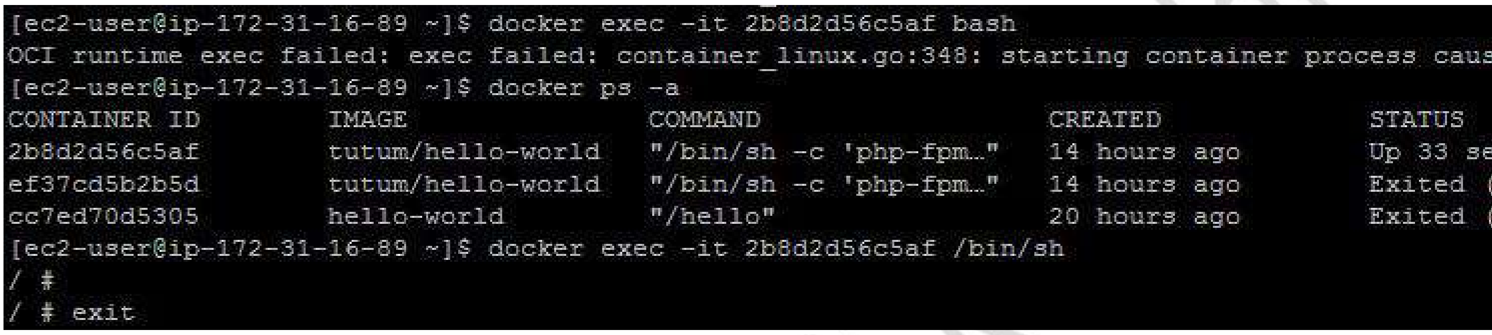


```
[ec2-user@ip-172-31-16-89 ~]$ docker search ubuntu
Error response from daemon: Get https://index.docker.io/v1/search?q=ubuntu&n=25: dial tcp 3.9
[ec2-user@ip-172-31-16-89 ~]$
```

Solution: Check security group rules.
===

Scenario 02:

OCI runtime exec failed: exec failed: container_linux.go:348: starting container process caused "exec:

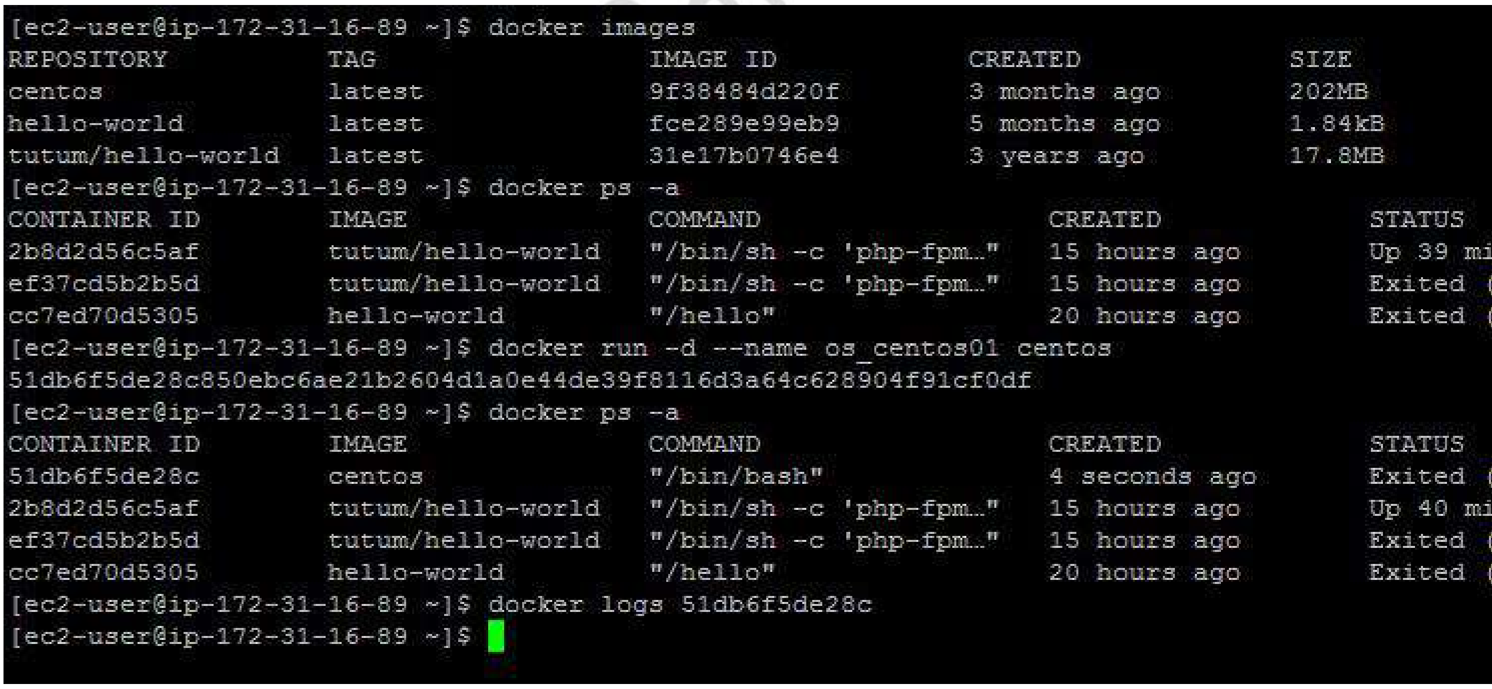


```
[ec2-user@ip-172-31-16-89 ~]$ docker exec -it 2b8d2d56c5af bash
OCI runtime exec failed: exec failed: container_linux.go:348: starting container process caus
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   14 hours ago        Up 33 se
ef37cd5b2b5d        tutum/hello-world   "/bin/sh -c 'php-fpm…"   14 hours ago        Exited (
cc7ed70d5305        hello-world         "/hello"                 20 hours ago        Exited (
[ec2-user@ip-172-31-16-89 ~]$ docker exec -it 2b8d2d56c5af /bin/sh
/ #
/ # exit
```

Solution: You might need to run use /bin/bash or /bin/sh, depending on the shell in your container.
===

Scenario 03:

OS image stop itself and not showing any logs.



```
[ec2-user@ip-172-31-16-89 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              9f38484d220f        3 months ago        202MB
hello-world         latest              fce289e99eb9        5 months ago        1.84kB
tutum/hello-world   latest              31e17b0746e4        3 years ago         17.8MB
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   15 hours ago        Up 39 mi
ef37cd5b2b5d        tutum/hello-world   "/bin/sh -c 'php-fpm…"   15 hours ago        Exited (
cc7ed70d5305        hello-world         "/hello"                 20 hours ago        Exited (
[ec2-user@ip-172-31-16-89 ~]$ docker run -d --name os_centos01 centos
51db6f5de28c850ebc6ae21b2604d1a0e44de39f8116d3a64c628904f91cf0df
[ec2-user@ip-172-31-16-89 ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
51db6f5de28c        centos              "/bin/bash"              4 seconds ago       Exited (
2b8d2d56c5af        tutum/hello-world   "/bin/sh -c 'php-fpm…"   15 hours ago        Up 40 mi
ef37cd5b2b5d        tutum/hello-world   "/bin/sh -c 'php-fpm…"   15 hours ago        Exited (
cc7ed70d5305        hello-world         "/hello"                 20 hours ago        Exited (
[ec2-user@ip-172-31-16-89 ~]$ docker logs 51db6f5de28c
[ec2-user@ip-172-31-16-89 ~]$
```

Solution: OS image not worked with -d option, instead of that use -it.
===

===

===

===

===

===

===

===

===

===

===

Simpleway:

1. Containers are isolated but share OS and appropriate bin/libraries but vm can't.

2. Docker hub is nothing but the remote registry to store images if user want local one then they have to user **registory2** where you can store their own images.

3. Public images are those which are on docker hub publically. Custom images which are created by you it might be public or private.

4. What is tag? If you

5. figlet is for large o/p.

Thank You ... This is incomplete document (v0.1)