

Docker

☐ What Is Docker..??

- Docker is a centralized platform for packaging, deploying, and running applications. Before Docker, many users face the problem that a particular code is running in the developer's system but not in the user's system. So, the main reason to develop docker is to help developers to develop applications easily, ship them into containers, and can be deployed anywhere.
- Docker was firstly released in March 2013. It is used in the Deployment stage of the software development life cycle that's why it can efficiently resolve issues related to the application deployment.
- Docker is a container management service. The keywords of Docker are develop, ship and run anywhere.
- It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application works in any environment like development, test, or production.
- Docker includes components such as **Docker client, Docker server, Docker machine, Docker hub, Docker composes**, etc.

☐ Docker Container-

- Docker containers are the **lightweight** alternatives of the virtual machine. It allows developers to package up the application with all its libraries and dependencies, and ship it as a single package. The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications. It automatically generates storage and space according to the application requirement.

☐ Virtual Machine-

- A virtual machine is a software that allows us to install and use other operating systems (Windows, Linux, and Debian) simultaneously on our machine. The operating system in which virtual machine runs are called virtualized operating systems. These virtualized operating systems can run programs and preforms tasks that we perform in a real operating system.
- **Containers V/S Virtual Machine-**

Containers	Virtual Machine
Integration in a container is faster and cheap.	Integration in virtual is slow and costly.
No wastage of memory.	Wastage of memory.
It uses the same kernel, but different distribution.	It uses multiple independent operating systems.

☐ Docker Features-

- Docker provides lots of features, some major features which are given below.
 - Easy and Faster Configuration.
 - Increase productivity.
 - Application Isolation.
 - Swarm.
 - Routing Mesh.
 - Services.
 - Security Management.

1.Easy and Faster Configuration-

- This is a key feature of docker that helps us to configure the system easily and faster.
- We can deploy our code in less time and effort. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

2.Increase Productivity-

- By easing technical configuration and rapid deployment of application. No doubt it has increase productivity. Docker not only helps to execute the application in isolated environment but also it has reduced the resources.

3.Application Isolation-

- It provides containers that are used to run applications in isolation environment. Each container is independent to another and allows us to execute any kind of application.

4.Swarm-

- It is a clustering and scheduling tool for Docker containers. Swarm uses the Docker API as its front end, which helps us to use various tools to control it. It also helps us to control a cluster of Docker hosts as a single virtual host. It's a self-organizing group of engines that is used to enable pluggable backends.

5.Routing Mesh-

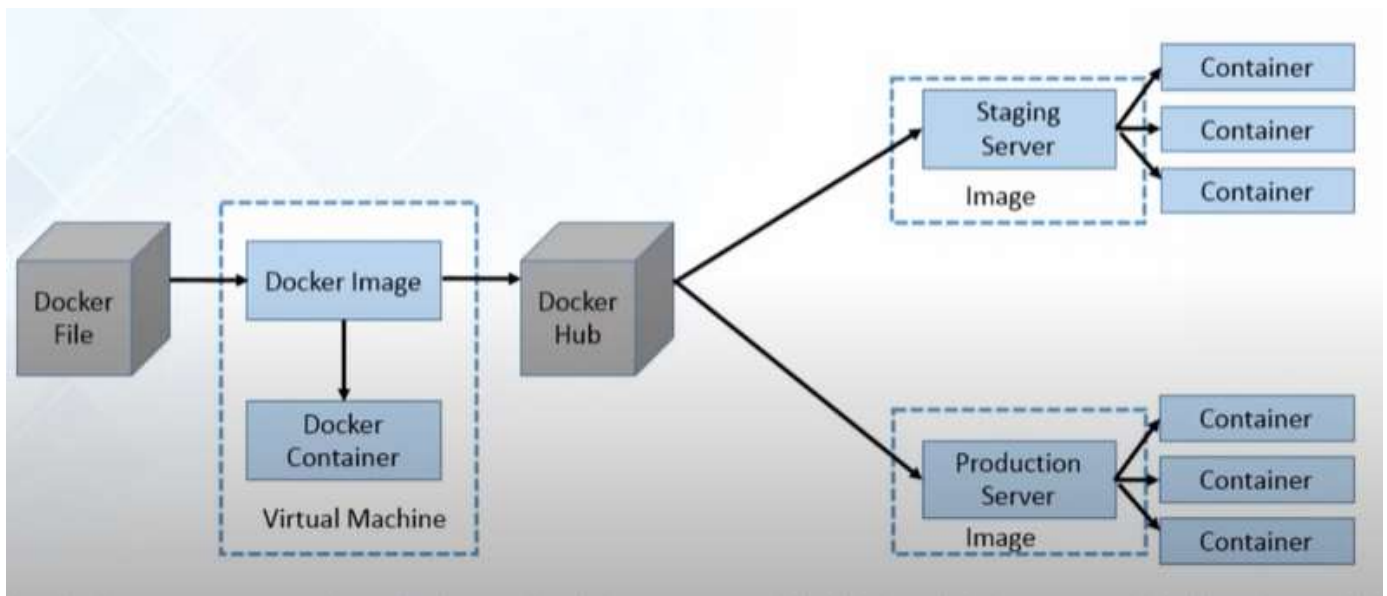
- It routes the incoming requests for published ports on available nodes to an active container. This feature enables the connection even if there is no task is running on the node.

☐ Architecture Of Docker-

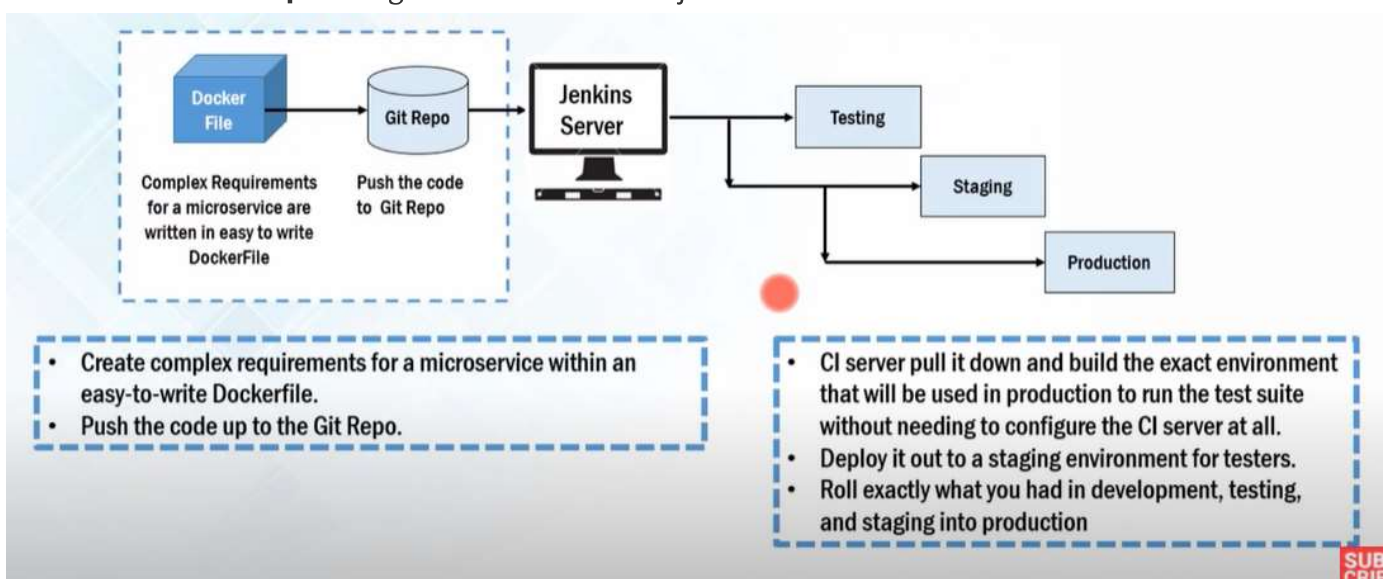
- **Docker File**-Developer writes a code that defines an application requirements or the dependencies in an easy to write docker file.
- **Docker Image**- Docker images is read only Templates.Docker file builds Docker Images and that images contains all the projects code, and contains all the dependencies required for particular application inside this image.
- **Docker Container**-Docker containers is nothing but the runtime instance of the Docker Image. It contains everything that is required to run an application or Micro service.
- **Docker Hub**-Docker hub git repository for docker images,it contains public as well as private repositories. So from public repositories you can pull images and also you can upload your images on Docker Hub.
- **Docker Daemon/Engine**-Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

- **Docker Registry-**

- Docker registry is a storage component for Docker Images.
- We can store images in either Public/Private repositories.
- Docker Hub is Docker's very own cloud repository.



- From Docker Hub Production Team or Testing Team can pull images and can build container.
- **Micro Service Architecture-** is an architectural style that structures an application as a collection of services that are
 - Highly maintainable and testable
 - Loosely coupled
 - Independently deployable
 - Organized around business capabilities
 - Owned by a small team
- The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.
- **Docker Example** integration with CI server jenkins-



- So in this way whatever requirement are there for your micro service is present throughout the Software Delivery Lifecycle.

1. Create EC2 Instance for Ubuntu-

- Click on launch EC2 instance,
- ### i) Choose an Amazon Machine image-
- Select **Ubuntu server 20.04 LTS**

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)

SUSE Linux

Free tier eligible

SUSE Linux Enterprise Server 15 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available.

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

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-04505e74c0741db8d (64-bit x86) / ami-0b49a4a6e8e22fa16 (64-bit Arm)

Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

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

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-0e472ba40eb589f49 (64-bit x86) / ami-0a940cb939351ccca (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

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

Cancel and Exit

64-bit (x86)

64-bit (Arm)

Select

64-bit (x86)

64-bit (Arm)

Select

64-bit (x86)

64-bit (Arm)

ii) Choose an Instance type-

Step 2: Choose an Instance Type

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-
<input checked="" type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-

iii) Configure Instance Details-

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing.

Number of instances

1

Launch into Auto Scaling Group

Purchasing option

☐ Request Spot instances

Network

vpc-0c231108277daecbd (default)

Create new VPC

Subnet

No preference (default subnet in any Availability Zone)

Create new subnet

Auto-assign Public IP

Use subnet setting (Enable)

Hostname type

Use subnet setting (IP name)

iv) Add Storage-

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)
Root	/dev/sda1	snap-0ebcdda21a8129fca	8	General Purpose SSD (gp2)	100 / 3000	N/A

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

v)Add Tags-

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value	Instances
(128 characters maximum)	(256 characters maximum)	
<input type="text" value="Name"/>	<input type="text" value="value"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

vi) Configure security Group-

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source
SSH	TCP	22	My IP 152.57.5.184/32
All TCP	TCP	0 - 65535	My IP 152.57.5.184/32

vii) Review and Create new Key pair-

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

Note: The selected key pair will be added to the set of keys authorized for this instance. [Learn more](#) about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair type

☒ RSA ☐ ED25519

Key pair name

[Download Key Pair](#)



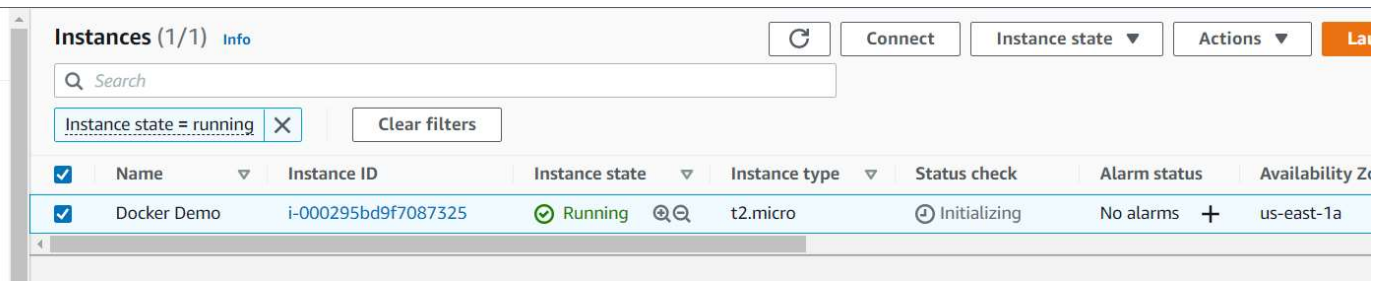
You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

[Cancel](#)

[Launch Instances](#)

- Download Key pair and launch instance.

- Go to ES2 instance,you can find running instance



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input checked="" type="checkbox"/>	Docker Demo	i-000295bd9f7087325	Running	t2.micro	Initializing	No alarms	us-east-1a

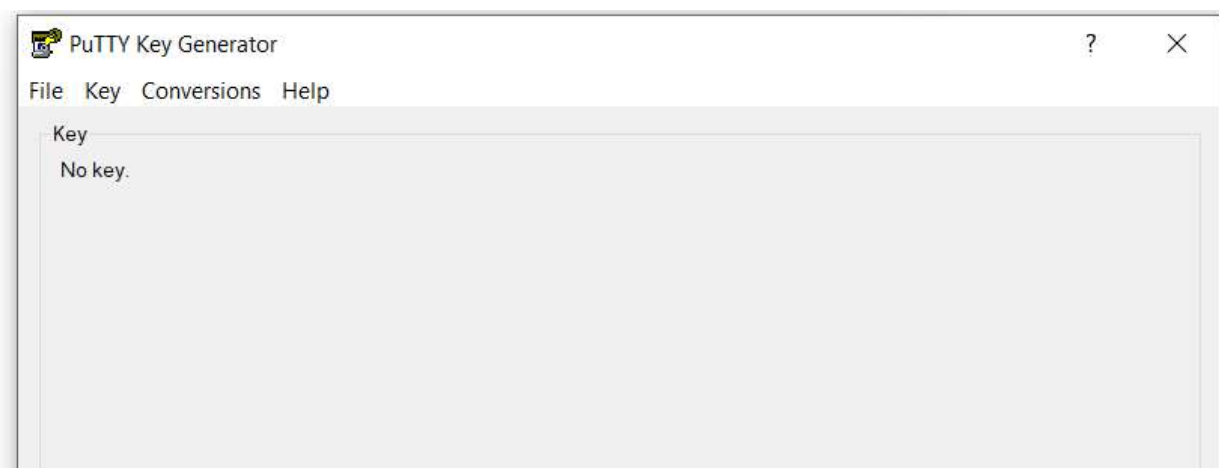
- We successfully created linux platform.

2.Connect to your Linux instance from Windows -

- An [Amazon Elastic Compute Cloud \(EC2\)](#) instance is a virtual server that runs applications on the Amazon Web Services (AWS) infrastructure.
- AWS is a robust and ever-evolving cloud computing platform, while EC2 is a service that enables company subscribers to run application programs in a computing environment. The EC2 will host an almost infinite number of virtual machines.
- To meet customer needs, Amazon offers a variety of instances with different CPU, memory, storage, and networking resource configurations. Each type is also available in two sizes to accommodate varying workloads.
- You can connect to your AWS EC2 instance in one of the following ways,
 1. Using PuTTY
 2. Using Git Bash

A)Connect EC2 Instance using PuTTY-

- PuTTY is a terminal emulator, serial console, and network file transfer program that is free and open-source. SCP, SSH, Telnet, rlogin, and raw socket connections are among the network protocols it supports.
- Now, to connect to your ec2 instance using PuTTY, you need to have a .ppk file. When you create an ec2 instance on AWS, a .pem file is downloaded. We need to convert the .pem file to a .ppk file and using this, connect to our instance through PuTTY. Follow the below steps to create a .ppk file from a .pem file,
 1. Open **PuTTYgen**
 2. Click on **Load**



Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

Parameters

Type of key to generate:
☒ RSA ☐ DSA ☐ ECDSA ☐ EdDSA ☐ SSH-1 (RSA)

Number of bits in a generated key:

3. Select .pem file-

4. Click on save private Key-

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADHN2JsoCv
+Wsn4aJFylfjUHjORCd03MsfWuuXoDTurs7+SMW1SSuKP/GfqnuEeYRvu3JnfVYzjzv73VEpiEMhpAh01Z0avEGNcpx4J
PAPBDgubsk
+WcyTXd5AB0QEMZn0CY023mbtj5poMxHNL5MYMsmKitybvN0Zms3SM2tnBRG65zzocPG7VJ0w4BnuzJ2D4Qf2C0s84
vA4aTUmWy005jr2RwbRYKnfsRSNX6/jz+wWnndujTlcUk6b+nnuynEgdHW/Ben/k9lBdxg
```

Key fingerprint: Warning

Key comment:

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

Parameters

Type of key to generate:
☒ RSA ☐ DSA ☐ ECDSA ☐ EdDSA ☐ SSH-1 (RSA)

Number of bits in a generated key:

5. You will get a warning message asking you to save the key without a passphrase. Click on **Yes**.

6. Provide the name and location for the file and click on **Save**.

- We have successfully generated a .ppk file from a .pem file. Now we will use this .ppk file to connect to our ec2 instance.

1. Open **PuTTY**

2. Provide the public IP address of your ec2 instance in the **Host Name** section

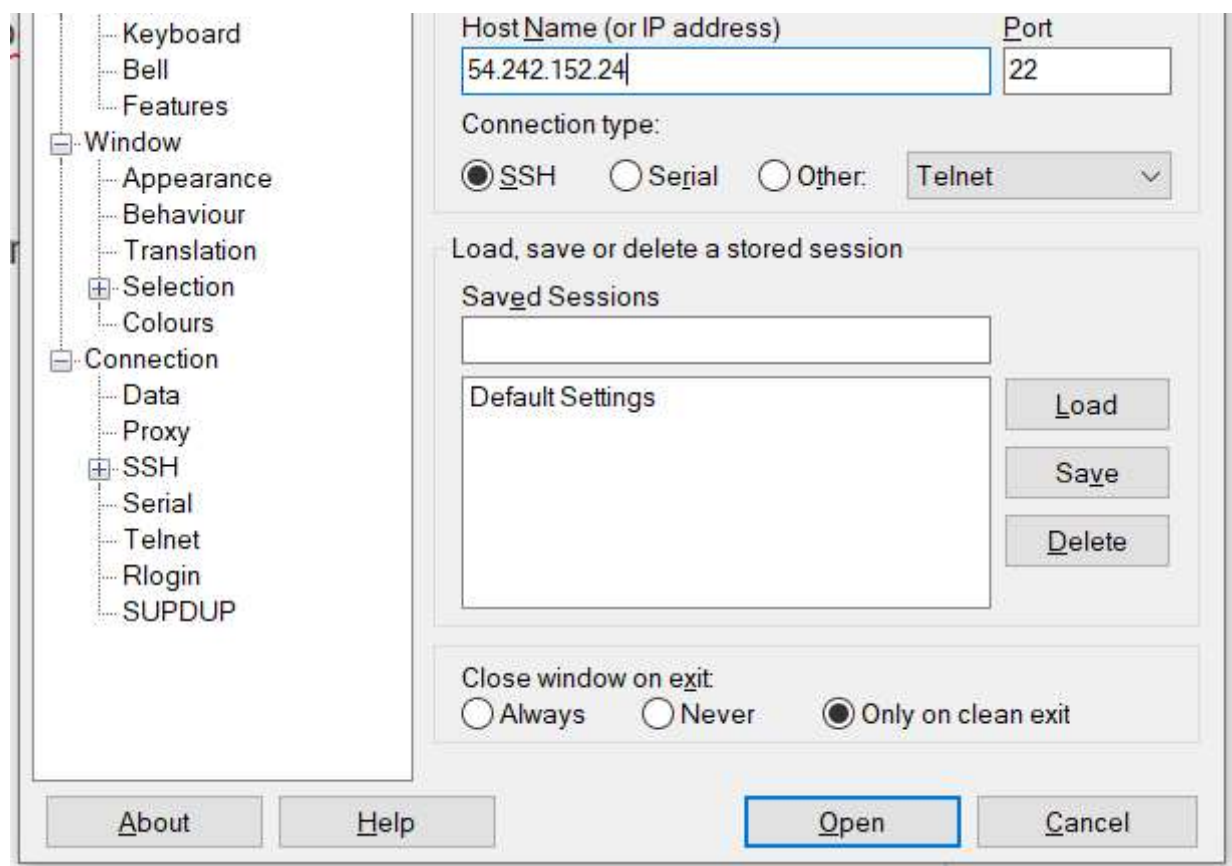
PuTTY Configuration

Category:

- Session
- Logging
- Terminal

Basic options for your PuTTY session

Specify the destination you want to connect to



3. On the left menu, expand **SSH** and click on **Auth**

4. Click on **Browse** and open the .ppk file

5. Click on **Open**. You will get a warning message again. Click on **Yes**

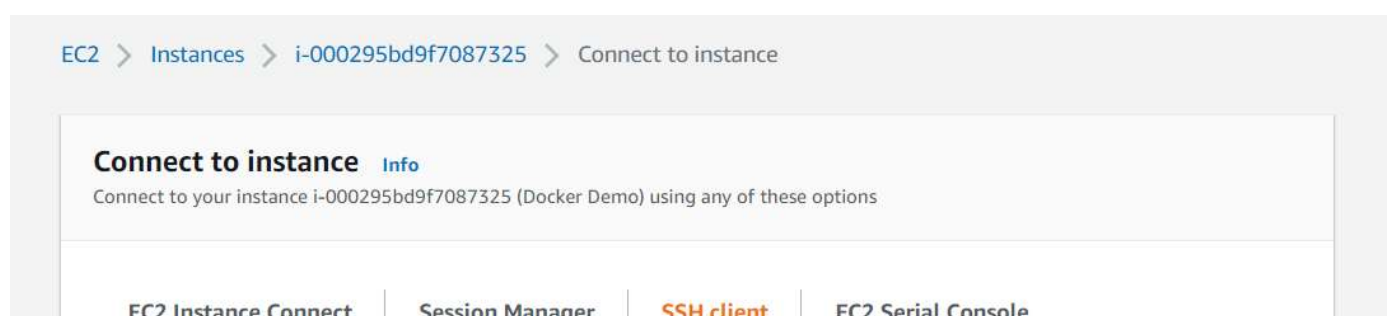
6. Provide the username as per your instance

Done. We have successfully connected to our AWS ec2 instance using PuTTY.

B)Connect to EC2 Instance using GitBash-

- Git Bash is a program that adds an emulation layer to Microsoft Windows environments, allowing users to use Git command-line. It's similar to a kit that installs some basic bash utilities on a Windows computer. It provides a command-line interface on Windows that enable us to use all Git features as well as most basic UNIX commands.

1. Open GitBash where .pem file is located and Execute the below command to connect to your aws instance,



Instance ID

 i-000295bd9f7087325 (Docker Demo)


1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is linux.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.


 `chmod 400 linux.pem`

4. Connect to your instance using its Public DNS:

 `ec2-54-242-152-24.compute-1.amazonaws.com`

Example:

 `ssh -i "linux.pem" ubuntu@ec2-54-242-152-24.compute-1.amazonaws.com`

 **Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

- Click on highlighted text and simply just copy into your GitBash command line interface.

☐ Install Docker-

1. Go on Root user-

- `sudo su` is a command which is used to provide the privileges to the root level.
- Syntax-

`sudo su`

```
ubuntu@ip-172-31-21-9:~$ sudo su
root@ip-172-31-21-9:/home/ubuntu#
```

2.Update packages-

- Syntax-

`sudo apt-get update`

```
root@ip-172-31-21-9:/home/ubuntu# sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
```

3.Install Docker -

- Syntax-

`apt install docker.io`

```
root@ip-172-31-21-9:/home/ubuntu# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
```

4.Check version of Docker-

- Syntax-

docker --version

```
root@ip-172-31-21-9:/home/ubuntu# docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
root@ip-172-31-21-9:/home/ubuntu#
```

5. Check current operating system-

- Syntax-

cat /etc/os-release

```
[ec2-user@ip-172-31-93-86 ~]$ cat/etc/os-release
-bash: cat/etc/os-release: No such file or directory
[ec2-user@ip-172-31-93-86 ~]$ cat /etc/os-release
NAME="Amazon Linux"
VERSION="2"
ID="amzn"
ID_LIKE="centos rhel fedora"
VERSION_ID="2"
PRETTY_NAME="Amazon Linux 2"
```

☐ Pull Image from Docker Hub and Create Container-

1. Check images in your Local -

- Syntax-

docker images

```
root@ip-172-31-34-146:~# docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
root@ip-172-31-34-146:~#
```

2. Pull image from Docker Hub-

- Syntax-

docker pull <image_name>

```
root@ip-172-31-21-9:/home/ubuntu# docker pull amazonlinux
Using default tag: latest
latest: Pulling from library/amazonlinux
bda57ff2d0d8: Pull complete
Digest: sha256:ff9ec1b0de6572fd14c6bdbda585828257d72dfea74b2b1
Status: Downloaded newer image for amazonlinux:latest
```

3. Check Docker status-

- Syntax-

service docker status

```
root@ip-172-31-21-9:/home/ubuntu# service docker status
● docker.service - Docker Application Container Engine
```

```
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor pre
Active: active (running) since Sat 2022-03-05 08:19:38 UTC; 13min ago
Docs: https://docs.docker.com
Main PID: 14470 (dockerd)
Tasks: 10
```

4.Create Container-

- By default it will check is there image available in local directory, if not then it will download from github.
- this command will convert image into container, also start container and we will go inside the container.
- Syntax-

`docker run -it --name <Name of Container> <Name of Image>`

```
root@ip-172-31-21-9:/home/ubuntu# docker run -it --name Demo amazonlinux
bash-4.2#
```

5.Exit from container-

- Syntax-

`exit`

```
bash-4.2# exit
exit
root@ip-172-31-21-9:/home/ubuntu#
```

6.Check Container status-

a)Existing Container-

- It will show running and stopped container.
- Syntax-

`docker ps -a`

```
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS
d142443c8d66   amazonlinux    "/bin/bash"             4 minutes ago   Exited (0) 2 minu
root@ip-172-31-21-9:/home/ubuntu#
```

b)Running Container-

- Syntax-

`docker ps`

```
root@ip-172-31-21-9:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS         NAMES
root@ip-172-31-21-9:/home/ubuntu#
```

7.Start Docker Container-

- This command is only for start container and container will reflect in running container status.
- syntax-

`docker start <container Id>`

```
root@ip-172-31-21-9:/home/ubuntu# docker start d142443c8d66
d142443c8d66
```

```
d142443c8d66
root@ip-172-31-21-9:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
d142443c8d66   amazonlinux    "/bin/bash"            20 minutes ago Up
root@ip-172-31-21-9:/home/ubuntu#
```

8. Stop Docker Container-

- Syntax-

docker stop <Container_ID>

```
root@ip-172-31-21-9:/home/ubuntu# docker stop d142443c8d6
d142443c8d6
root@ip-172-31-21-9:/home/ubuntu#
root@ip-172-31-21-9:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
root@ip-172-31-21-9:/home/ubuntu#
```

9. Go in Container-

- We cant go inside the container although we have start it .
- Syntax

docker attach <Container Id>

```
root@ip-172-31-21-9:/home/ubuntu# docker attach d142443c8d66
bash-4.2#
```

10. Create file in container-

- Syntax-

touch <file Name>

```
bash-4.2# touch demo.txt
bash-4.2# ls
bin    demo.txt  etc      lib      local    mnt      proc     run      srv      tmp      var
boot  dev              home    lib64   media   opt      root     sbin     sys      usr
bash-4.2#
```

- Add content in new created file-

Syntax-

vi <file Name>

For exit pop window , : wq

- Read content from file-

syntax-

cat <file name>

```
bash-4.2# vi demo.txt
bash-4.2# cat demo.txt
This is demo file
bash-4.2#
```


☐ Create Image from Container-

1. Exit from container-

```
bash-4.2# exit
exit
root@ip-172-31-21-9:/home/ubuntu# |
```

2. Create Image of Docker Container-

- Syntax-

docker commit <container id> <image name>

```
root@ip-172-31-21-9:/home/ubuntu# docker commit d142443c8d66 myimage
sha256:544f8963ead538a69188e4b2458db66a5282522b8818b86bd609ad9c1d67e2a3
root@ip-172-31-21-9:/home/ubuntu# docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
myimage         latest      544f8963ead5   13 seconds ago 164MB
amazonlinux     latest      6862f029c6f5   29 hours ago   164MB
root@ip-172-31-21-9:/home/ubuntu# |
```

☐ Create EC2 instance for Windows-

1.Choose an Amazon Machine Image-

- Choose Microsoft Windows server 2019 Base

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Add Tags6. Configure Security Group7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

Q windows

Search by Systems Manager parameter

AWS Launch Wizard for SQL Server offers an easy way to size, configure, and deploy Microsoft SQL Server Always On availability groups. Use AWS Launch Wizard for this launch

Quick Start (19)

My AMIs (0)

AWS Marketplace (938)

Community AMIs (10323)

☐ Free tier only

Windows

Free tier eligible

Microsoft Windows Server 2019 Base - ami-0c19f80dba70861db

Microsoft Windows 2019 Datacenter edition. [English]

Root device type: ebsVirtualization type: hvmENA Enabled: Yes

Select

64-bit (x86)

Windows

Free tier eligible

Microsoft Windows Server 2019 Base with Containers - ami-0d9de6258e941b4f4

Microsoft Windows 2019 Datacenter edition with Containers. [English]

Root device type: ebsVirtualization type: hvmENA Enabled: Yes

Select

64-bit (x86)

2.Choose an Instance type-

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by:

All instance families

Current generation

[Show/Hide Columns](#)

Currently selected: t2.nano (- ECUs, 1 vCPUs, 2.4 GHz, -, 0.5 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available
<input checked="" type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-
<input type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-

3.Configure Instance Details-

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access man

Number of instances ⓘ 1 Launch into Auto Scaling Group ⓘ

Purchasing option ⓘ ☐ Request Spot instances

Network ⓘ vpc-0c231108277daecbd (default) ↕ Create new VPC

Subnet ⓘ No preference (default subnet in any Availability Zone) ↕ Create new subnet

Auto-assign Public IP ⓘ Use subnet setting (Enable) ↕

Hostname type ⓘ Use subnet setting (IP name) ↕

DNS Hostname ⓘ ☒ Enable IP name IPv4 (A record) DNS requests
☒ Enable resource-based IPv4 (A record) DNS requests

4.Add storage-

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ
Root	/dev/sda1	snap-0ce4ad4a65b3d72e7	30	General Purpose SSD (gp2) ▾	100 / 3000	N/A

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

5. Add tags-

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key ⓘ (128 characters maximum)	Value ⓘ (256 characters maximum)	Instances ⓘ	Volumes ⓘ
--------------------------------	----------------------------------	-------------	-----------

Name test ☒ ☒

Add another tag (Up to 50 tags maximum)

6.Configure Security Group-

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name: launch-wizard-2

Description: launch-wizard-2 created 2022-03-05T16:00:06.848+05:30

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ
RDP ▾	TCP	3389	Custom ▾ 0.0.0.0/0

Add Rule

7. Review instance Launch

7. Review Instance Launch-

Step 7: Review Instance Launch

▼ AMI Details



Microsoft Windows Server 2019 Base - ami-0c19f80dba70861db

Free tier
eligible

Microsoft Windows 2019 Datacenter edition. [English]

Root Device Type: ebs Virtualization type: hvm

If you plan to use this AMI for an application that benefits from Microsoft License Mobility, fill out the [License Mobility Form](#). Don't show me this again

▼ Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Availab
t2.nano	-	1	0.5	EBS only	-

▼ Security Groups

Security group name launch wizard 2

- Download Key pair-

Select an existing key pair or create a new key pair



A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

Note: The selected key pair will be added to the set of keys authorized for this instance. [Learn more about removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair type

☒ RSA ☐ ED25519

Key pair name

windows

Download Key Pair



You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

8. Download RDP file-

EC2 > Instances > i-0b435ab3a3447be41 > Connect to instance

Connect to instance [Info](#)


Connect to your instance i-0b435ab3a3447be41 (test) using any of these options

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download remote desktop file](#)

When prompted, connect to your instance using the following details:

Public DNS

 ec2-3-82-69-152.compute-1.amazonaws.com

User name

 Administrator

Password [Get password](#)

- After that click on "Get Password" and upload .pem file.

EC2 > Instances > i-0b435ab3a3447be41 > Get Windows password

Get Windows password [Info](#)

Retrieve and decrypt the initial Windows administrator password for this instance.

To decrypt the password, you will need your key pair for this instance.



Key pair associated with this instance
windows

Browse to your key pair:

 [Browse](#)

Or copy and paste the contents of the key pair below:

- Copy created password .

9. Run Test.rdp file from Downloads-

← → ↕ ⬇ This PC > Downloads

Quick access

- Desktop
- Downloads
- Documents
- Pictures
- Python Class lect
- Machine Learning
- SQL
- 3. Identity Access M

Today (8)



test.rdp

Yesterday (4)



Remote Desktop Connection

The publisher of this remote connection can't be identified. Do you want to connect anyway?

This remote connection could harm your local or remote computer. Do not connect unless you know where this connection came from or have used it before.



Publisher: **Unknown publisher**
Type: Remote Desktop Connection
Remote computer: ec2-3-82-69-152.compute-1.amazonaws.com

☐ Don't ask me again for connections to this computer



wsupdate_...
(1).msi

- Paste copied password in pop up window. and your virtual windows operating system will run.

☐ Docker File

- A Dockerfile is a text document that contains commands that are used to assemble an image.

We can use any command that call on the command line. Docker builds images automatically by reading the instructions from the Docker-file.

- The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

☐ **Docker File Instructions-**

- The instructions are not case-sensitive but you must follow conventions which recommend to use uppercase.
- Docker runs instructions of Docker file in top to bottom order. The first instruction must be **FROM** in order to specify the Base Image.
- A statement begin with # treated as a comment. You can use RUN, CMD, FROM, EXPOSE, ENV etc instructions in your Dockerfile.

1.FROM-

- This instruction is used to set the Base Image for the subsequent instructions. A valid Dockerfile must have FROM as its first instruction.
- Example-
FROM ubuntu

2.LABEL-

- We can add labels to an image to organize images of our project. We need to use LABEL instruction to set label for the image.
- Example-
LABEL vebdrol ='JavaTpoint'

3.RUN-

- This instruction is used to execute any command of the current image.
- Example-
RUN /bin/bash -c 'source \$HOME/.bashrc; echo \$HOME'

4.CMD-

- This is used to execute application by the image. We should use CMD always in the following form
- Example-
CMD ["executable", "param1", "param2"?]
- This is preferred way to use CMD. There can be only one CMD in a Dockerfile. If we use more than one CMD, only last one will execute.

5.COPY-

- This instruction is used to copy new files or directories from source to the filesystem of the container at the destination.
- Example-
COPY abc/ /xyz
- The source path must be inside the context of the build. We cannot COPY ../something /something because the first step of a docker build is to send the context directory (and subdirectories) to the docker daemon.
- If source is a directory, the entire contents of the directory are copied including filesystem metadata.

6.WORKDIR-

- The WORKDIR is used to set the working directory for any RUN, CMD and COPY instruction that follows it in the Dockerfile. If work directory does not exist, it will be created by default.
- We can use WORKDIR multiple times in a Dockerfile.
- Example-
WORKDIR /var/www/html

- Example-

1.Create Docker-file

```
root@ip-172-31-89-134:/home/ubuntu# ls
root@ip-172-31-89-134:/home/ubuntu# vi Dockerfile
root@ip-172-31-89-134:/home/ubuntu# ls
Dockerfile
root@ip-172-31-89-134:/home/ubuntu# cat Dockerfile
#This is a sample Image
FROM ubuntu
MAINTAINER demousr@gmail.com

RUN apt-get update
RUN apt-get install -y python
RUN apt-get install -y pandas
RUN apt-get install -y numpy
CMD ["echo", "Image created"]
```

2. Create Image from Docker-file

- Syntax-
docker build -t <docker_Image_Name> <path of Docker-File>

```
root@ip-172-31-89-134:/home/ubuntu# docker build -t dockerimage .
Sending build context to Docker daemon 13.31kB
Step 1/7 : FROM ubuntu
---> 2b4cba85892a
Step 2/7 : MAINTAINER demousr@gmail.com
---> Using cache
---> bedadd5ba421
Step 3/7 : RUN apt-get update
---> Using cache
---> 0a505ccfde26
Step 4/7 : RUN apt-get install -y python
```

- Docker file is in current directory so give '.' while writing command.

3. Now create Container from new created Docker image

```
root@ip-172-31-89-134:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
<none>        <none>    f46c2fd199f2   16 minutes ago 143MB
ubuntu        latest    2b4cba85892a   4 days ago     72.8MB
centos        latest    5d0da3dc9764   5 months ago   231MB
root@ip-172-31-89-134:/home/ubuntu# docker run -it --name container1 f46c2fd199f2
root@d54bf7678a55:/#
```

- New container created ,we can find that,

```
root@d54bf7678a55:/# exit
```



```

root@d54b17678a55:/# exit
exit
root@ip-172-31-89-134:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@ip-172-31-89-134:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
d54bf7678a55   f46c2fd199f2   "bash"    2 minutes ago   Exited (0) 14 seconds ago   frosty_b
r1
8092237f33d8   f46c2fd199f2   "/bin/sh -c 'apt-get..." 20 minutes ago   Exited (100) 20 minutes ago   frosty_b
lackwell

```

- Docker container is running now

```

root@ip-172-31-89-134:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
d54bf7678a55   f46c2fd199f2   "bash"    12 minutes ago   Exited (0) 6 seconds ago   frosty_b
r1
8092237f33d8   f46c2fd199f2   "/bin/sh -c 'apt-get..." 30 minutes ago   Exited (100) 30 minutes ago   frosty_b
lackwell
7abe02251336   0a505ccfde26   "/bin/sh -c 'apt-get..." 35 minutes ago   Exited (1) 35 minutes ago   frosty_b
artz

```

- We have make container with three methods-
 1. Pulling image from Docker Hub ----- docker pull <image name>
 2. Searching image in local directory, if available then will create container otherwise pull from Docker Hub----- docker run -it -- name <container_name> <image_name>
 3. With the image created from docker file.

☐ Interview Questions and Answers-

1. What is Docker?

Criteria	Docker	Virtual Machines
Use of OS	All containers share the host OS	Each VM runs on its own OS
Startup time	Very fast	Slow
Isolation	Process-level isolation	Full isolation
Security	Low	High

- We can define Docker as a containerization platform that combines all our applications in a package so that we have all the dependencies to run our applications in any environment. This means, our application will run seamlessly on any environment, and this makes it easy for having a product-ready application. What Docker does is wrap the software needed in a file system that has everything for running the code, providing the runtime and all the necessary libraries and system tools. Containerization technology like Docker will share the same operating system kernel with the machine, and due to this it is extremely fast. This means that we have to run Docker only at the beginning and after that, since our OS is already running, we will have a smooth and seamless process.

2. Define Containerization.

- Containerization is a form of virtualization through which applications are run in containers (isolated user spaces) all using a shared OS. It packs or encapsulates software code and all its dependencies for it to run in a consistent and uniform manner on any infrastructure.

3. What is the benefit of using a Docker over a Hypervisor?

- The benefit of Docker is that it is a lightweight containerization technology that does not require a full operating system to run. It is designed to be used on top of a Linux operating system, and it allows you to run multiple containers on a single host. This means that you can run multiple applications on a single host, and each application can have its own dependencies and configuration. This is a significant benefit over a hypervisor, which requires a full operating system to run on each virtual machine. Additionally, Docker containers are much faster to start and stop than virtual machines, and they are much easier to manage and deploy.

- Though Docker and Hypervisor might do the same job overall, there are many differences between them in terms of how they work. Docker can be thought of as lightweight since it uses very few resources and also the host kernel rather than creating it like a Hypervisor.

4. What are the unique features of Docker over other containerization technologies?

- Some of the most important and unique features of Docker are as follows:
- We can run our Docker container either on our PC or on our enterprise IT system.
- Along with the Docker Hub, which is a repository of all containers, we can deploy and download all our applications from a central location.
- We can even share our applications with the containers that we create.

5. What is a Docker image?

- A Docker image helps in creating Docker containers. We can create the Docker image with the build command; due to this, it creates a container that starts when it begins to run. All the Docker images are stored in the Docker registry such as the public Docker registry. These have minimal amounts of layers within the image so that there is a minimum amount of data on the network.

6. What is a Docker container?

- It is a comprehensive set of applications including all its dependencies which share the same OS kernel, along with the other containers running in separate processes within the operating system in a user space. Docker is not tied to any IT infrastructure, and thus it can run on any computer system or on the cloud. We can create a Docker container using Docker images and then run it, or we can use the images that are already created in the Docker Hub. To simplify things, let's say that the Docker containers are just runtime instances of the Docker image.

7. What is a Docker Hub?

- We can think of Docker Hub as a cloud registry that lets us link the code repositories, create the images, and test them. We can also store our pushed images, or we can link to the Docker Cloud, so that the images can be deployed to the host. We have a centralized container image discovery resource that can be used for the collaboration of our teams, automating the workflow and distribution, and changing management by creating the development pipeline.

8. What is a Docker Swarm?

- We can think of a Docker Swarm as the way of orchestrating the Docker containers. We will be able to implement Dockers in a cluster. We can convert our Docker pools into a single Docker Swarm for easy management and monitoring.

9. What is the use of a Dockerfile?

- A Dockerfile is a set of specific instructions that we need to pass on to Docker so that the images can be built. We can think of the Dockerfile as a text document which has all the commands that are needed for creating a Docker image. We can create an automated build that lets us execute multiple command lines one after the other.

10. What is Docker Compose?

- Docker Compose defines and runs multi-container Docker applications. With Compose, a YAML file is used to configure an application's services. All the services from the configuration can be created and started with a single command.

11. Why use Docker?

Following are the reasons why one should use Docker:

- It allows the use of system resources more efficiently
- Software delivery cycles are faster with it
- Application portability is possible and easy
- It is great for the microservices architecture

12. What are the drawbacks of Docker?

Docker has a few drawbacks as listed below:

- No storage option
- Poor monitoring
- Unable to automatically reschedule inactive nodes
- Has a complicated automatic horizontal scaling setup

13. What is Docker Engine?

- Docker Engine is an open-source containerization technology that facilitates the development, assembling, shipping, and running of applications with the help of the following components:
- Docker Daemon
- Docker Engine REST API
- Docker CLI

14. What are registries?

- Docker registries provide locations for storing and downloading images. There are two types of registries
- Public registry
- Private registry
- Public registries include Docker Hub and Docker Cloud.

15. What are Docker namespaces?

- When a container is run, Docker creates a set of isolated workspaces for the container called namespaces.

☐ **Interview Notes-**

