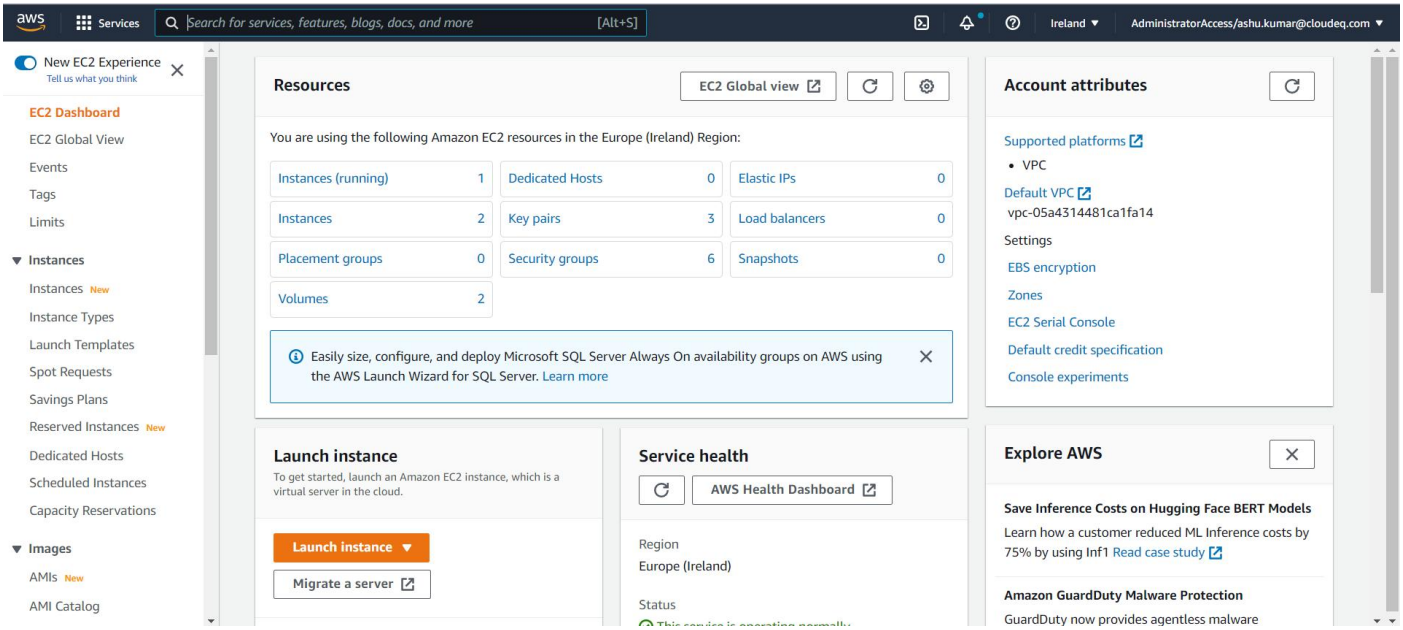
Batch-3

Assignmnet-1(Sprint-5)

ANSHUL KUMAR

Q1. What Is EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.



(Dashboard Of EC2)

Features of Amazon EC2

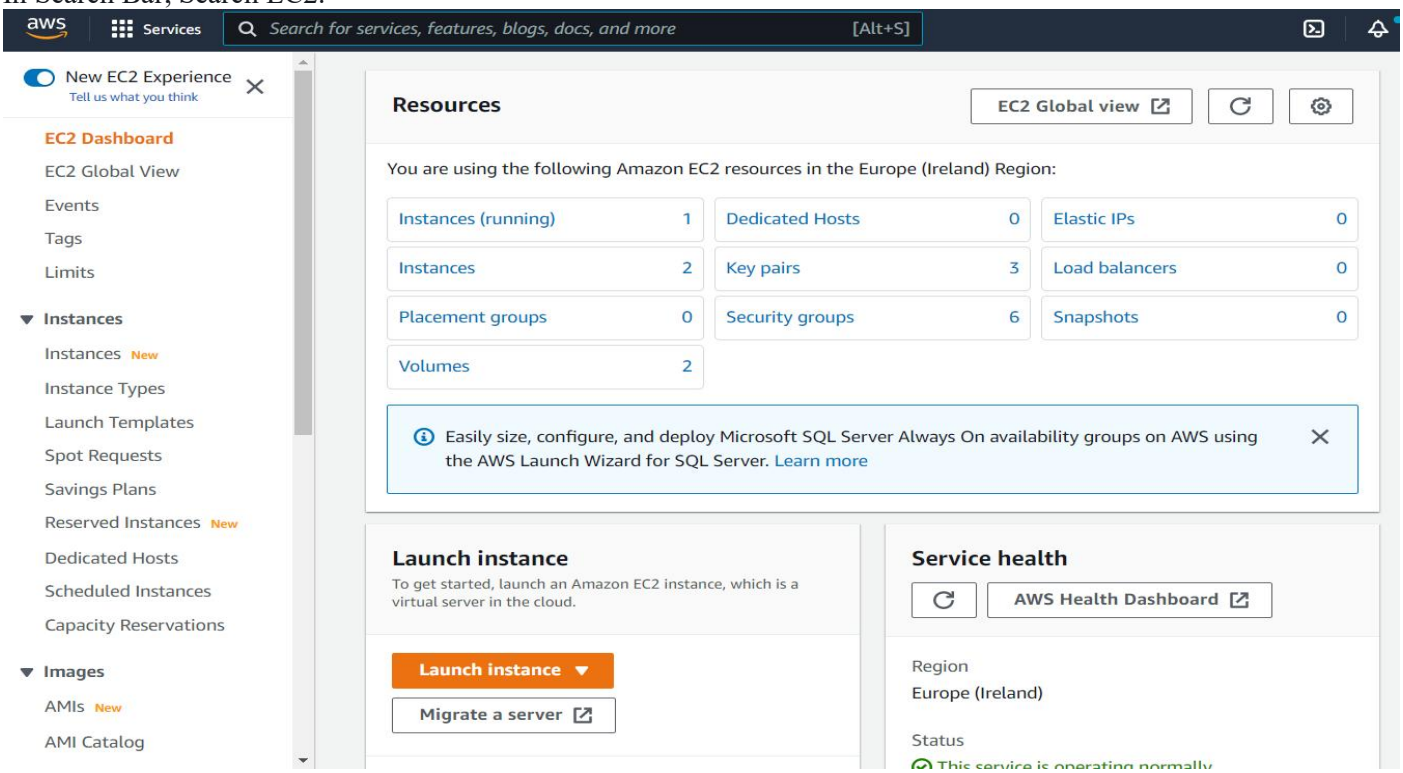
Amazon EC2 provides the following features:

1. Virtual computing environments, known as *instances*
2. Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
3. Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*.
4. Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
5. Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as *instance store volumes*
6. Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*

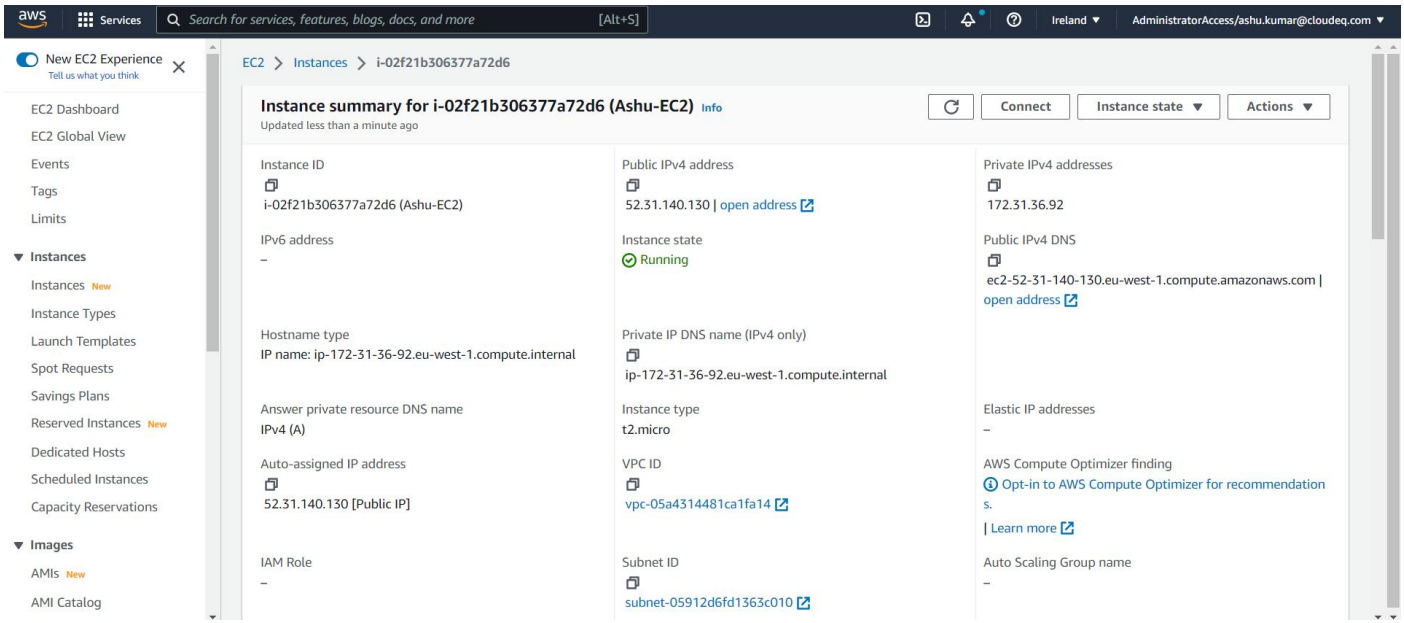
7. Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *Regions* and *Availability Zones*
8. A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
9. Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
10. Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
11. Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as *virtual private clouds* (VPCs)

Steps To Make An EC2 Instance

1. Open The AWS Portal.
2. In Search Bar, Search EC2.



3. Click On Instances → Click On Launch Instance.
4. Enter The Following Details(Like:- Name, Select Machine, Select Instance Type, Create/Select Key pair, Select Networking Settings)
5. Click On Launch Instance.

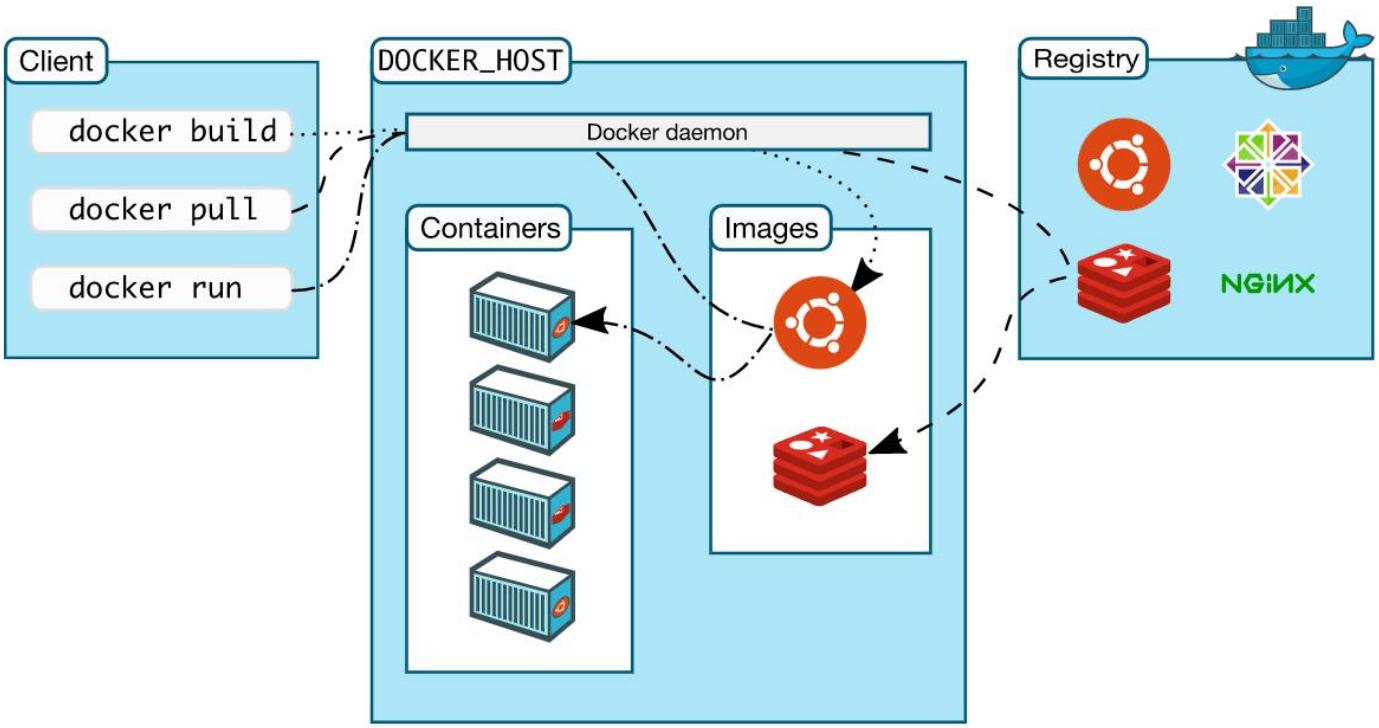

aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
AdministratorAccess/ashu.kumar@cloudeq.com
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs New
AMI Catalog
EC2 > Instances > i-02f21b306377a72d6
Instance summary for i-02f21b306377a72d6 (Ashu-EC2) Info
Updated less than a minute ago
Connect
Instance state
Actions
Instance ID
i-02f21b306377a72d6 (Ashu-EC2)
Public IPv4 address
52.31.140.130 | open address
Private IPv4 addresses
172.31.36.92
IPv6 address
–
Instance state
Running
Public IPv4 DNS
ec2-52-31-140-130.eu-west-1.compute.amazonaws.com | open address
Hostname type
IP name: ip-172-31-36-92.eu-west-1.compute.internal
Private IP DNS name (IPv4 only)
ip-172-31-36-92.eu-west-1.compute.internal
Answer private resource DNS name
IPv4 (A)
Instance type
t2.micro
Elastic IP addresses
–
Auto-assigned IP address
52.31.140.130 [Public IP]
VPC ID
vpc-05a4314481ca1fa14
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
| Learn more
IAM Role
–
Subnet ID
subnet-05912d6fd1363c010
Auto Scaling Group name
–

Q2. What Is Docker?

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production

Docker Architecture

Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



Q3. What Is Virtual Machine?

A Virtual Machine (VM), on the other hand, is created to perform tasks that if otherwise performed directly on the host environment, may prove to be risky. VMs are isolated from the rest of the system; the software inside the virtual machine cannot tamper with the host computer. Therefore, implementing tasks such as accessing virus-infected data and testing of operating systems are done using virtual machines. We can define a virtual machine as:

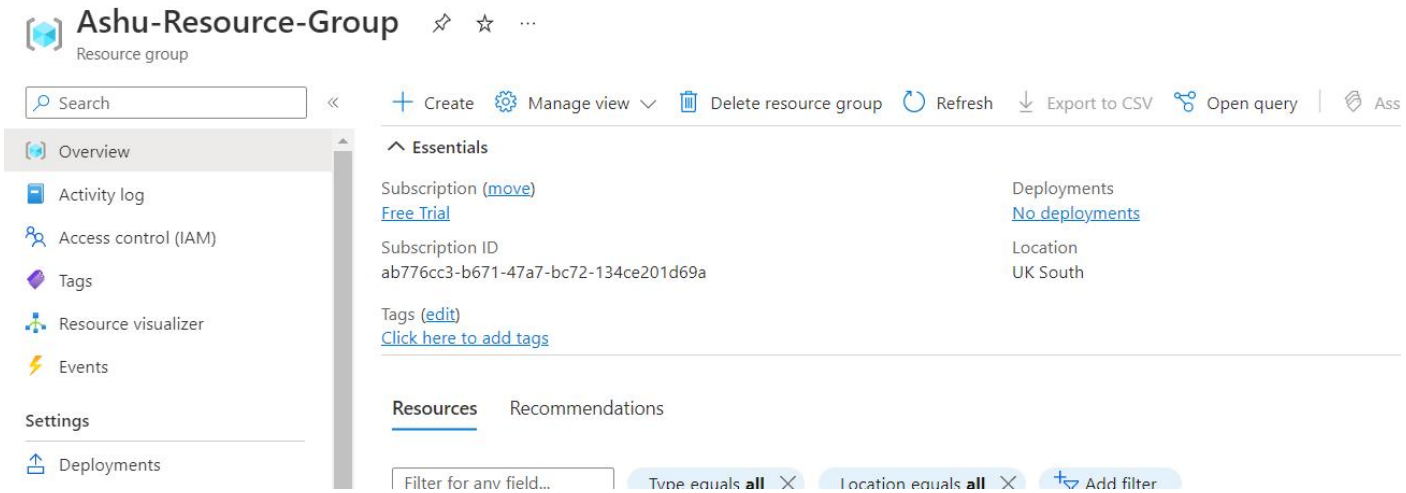
A virtual machine is a computer file or software usually termed as a guest, or an image that is created within a computing environment called the host.

VMs are broadly divided into two categories depending upon their use:

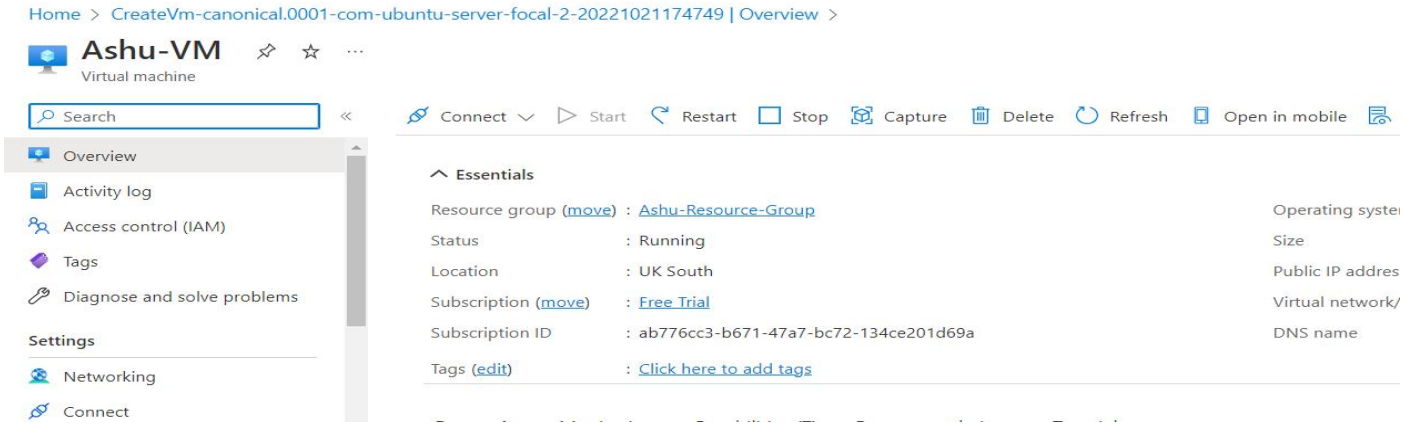
1. System Virtual Machines: A platform that allows multiple VMs, each running with its copy of the operating system to share the physical resources of the host system. Hypervisor, which is also a software layer, provides the virtualization technique. The hypervisor executes at the top of the operating system or the hardware alone.
2. Process Virtual Machine: Provides a platform-independent programming environment. The process virtual machine is designed to hide the information of the underlying hardware and operating system and allows the program to execute in the same manner on every given platform.

Steps To Create A Virtual Machine

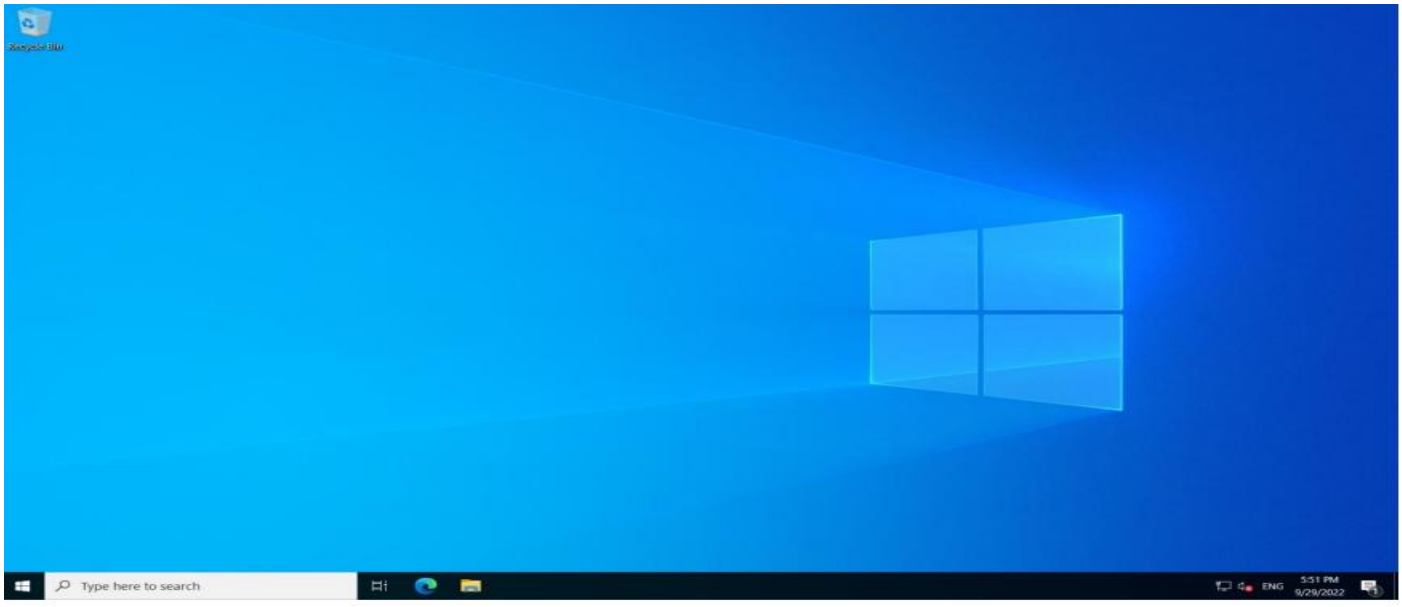
1. Open The Azure Portal.
2. Make A Resource Group.



3. After Making Resource Group Go To Search Bar And Search Virtual Machine
4. Click On Create Button.
5. Enter The Details(Like:- Select Resource Group, Name, Select Machine, Select Size, Enter I'd & Password)
6. Click Next.
7. Select Disk Type → Click Next.
8. Click On Review + Create.



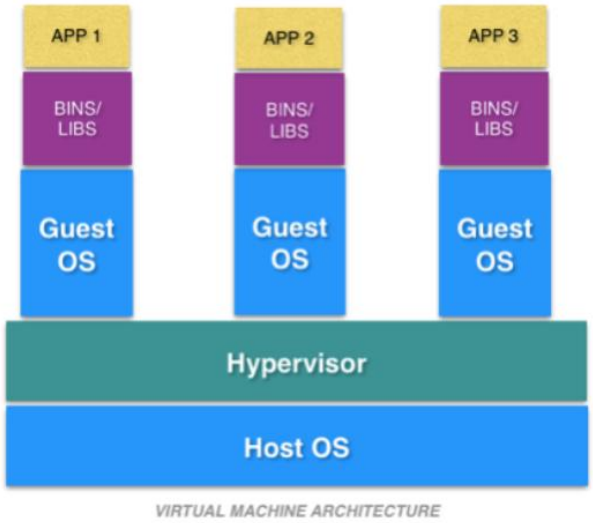
9. Click On Connect → Enter I'd & Password.



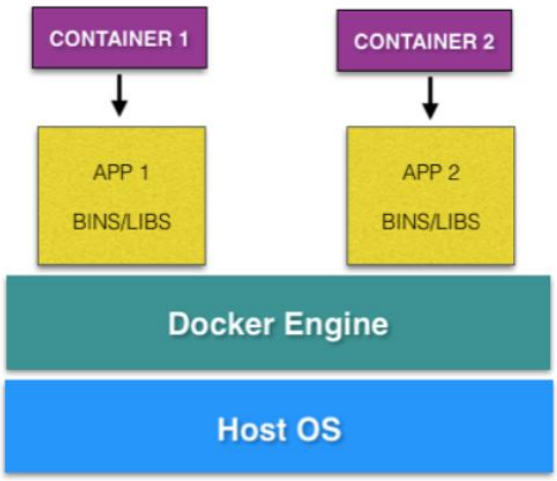
Q4. Difference Between Docker V/S Virtual Machine?

1. OS Support And Architecture

The main difference between Docker and VMs lies in their architecture, demonstrated below.



VIRTUAL MACHINE ARCHITECTURE



DOCKER ARCHITECTURE

VMs have the host OS and guest OS inside each VM. A guest OS can be any OS, like Linux or Windows, irrespective of the host OS. In contrast, Docker containers host on a single physical server with a host OS, which shares among them. Sharing the host OS between containers makes them light and increases the boot time. Docker containers are considered suitable to run multiple applications over a single OS kernel; whereas, virtual machines are needed if the applications or services required to run on different OS.

2. Security

The second difference between VMs and Docker is that Virtual Machines are stand-alone with their kernel and security features. Therefore, applications needing more privileges and security run on virtual machines.

On the flip side, providing root access to applications and running them with administrative premises is not recommended in the case of Docker containers because containers share the host kernel. The container technology has access to the kernel subsystems; as a result, a single infected application is capable of hacking the entire host system.

3. Portability

Another relevant Docker vs Virtual Machine difference is about portability: VMs are isolated from their OS, and so they are not ported across multiple platforms without incurring compatibility issues. At the development level, if an application is to be tested on different platforms, then Docker containers must be considered.

Docker container packages are self-contained and can run applications in any environment, and since they don't need a guest OS, they can be easily ported across different platforms. Docker containers can be easily deployed in servers since containers being lightweight can be started and stopped in very less time compared to virtual machines.

4. Performance

The last main Docker vs VM difference refers to performance: Virtual Machines are more resource-intensive than Docker containers as the virtual machines need to load the entire OS to start. The lightweight architecture of Docker containers is less resource-intensive than virtual machines.

In the case of a virtual machine, resources like CPU, memory, and I/O may not be allocated permanently to containers — unlike in the case of a Docker container, where the resource usage works with the load or

traffic. Scaling up and duplicating a Docker container is simple and easy as compared to a virtual machine because there is no need to install an operating system in them.

Apart from the major differences between Docker and VMs, some other ones are summarized below:

	Docker	Virtual Machines (VMs)
Boot-Time	Boots in a few seconds.	It takes a few minutes for VMs to boot.
Runs on	Dockers make use of the execution engine.	VMs make use of the hypervisor.
Memory Efficiency	No space is needed to virtualize, hence less memory.	Requires entire OS to be loaded before starting the surface, so less efficient.
Isolation	Prone to adversities as no provisions for isolation systems.	Interference possibility is minimum because of the efficient isolation mechanism.
Deployment	Deploying is easy as only a single image, containerized can be used across all platforms.	Deployment is comparatively lengthy as separate instances are responsible for execution.
Usage	Docker has a complex usage mechanism consisting of both third party and docker managed tools.	Tools are easy to use and simpler to work with.