Dr. D. Y. Patil Pratishthan's

Institute for Advanced Computing &Software Development

IACSD

DevOps

INDEX

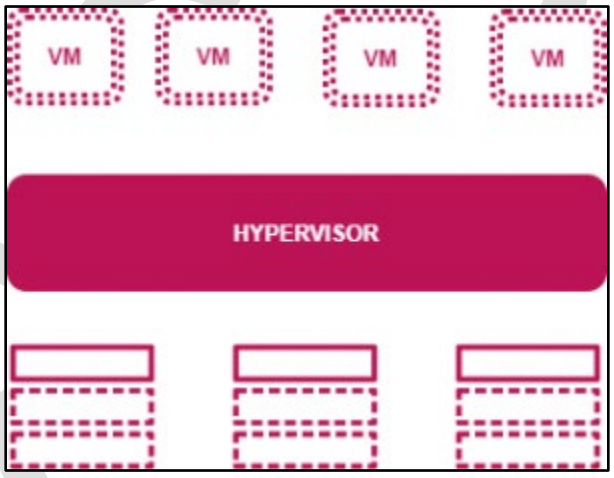
Introduction of virtualization

Virtualization is a technique to divide the computer resources logically. It's achieved by abstracting away the underlying complexity of resource segregation. Although an old technology, it's still a popular technique and highly relevant in this era of cloud computing.

Virtualization helps us to create software-based or virtual versions of a computer resource. These computer resources can include computing devices, storage, networks, servers, or even applications.

It allows organizations to partition a single physical computer or server into several virtual machines (VM). Each VM can then interact independently and run different operating systems or applications while sharing the resources of a single computer.

Hypervisor software facilitates virtualization. A hypervisor sits on top of an operating system but we can also have hypervisors that are installed directly onto the hardware. Hypervisors take physical resources and divide them up so that virtual environments can use them. When a user or program issues an instruction to the VM that requires additional resources from the physical environment, the hypervisor relays the request to the physical system and caches the changes.



A virtual machine created by a hypervisor function as a single data file, and we can move it from one computer to another, open it there, and it works the same as on any other machine. Thus, it provides a lot of flexibility and portability.

Virtualization types: type1, type2

Type 1 Hypervisors (Bare Metal)

A Type 1 hypervisor is installed directly on top of the physical machine. Type 1 hypervisors are also known as bare-metal hypervisors due to the nature of their installation type.

These categories of hypervisors are more popular and secure than the Type 2 hypervisors.

Type 1 hypervisors have a lower amount of latency and are the most used in the market. Some examples of these hypervisors are VMware ESXi, Microsoft Hyper-V, or open-source Kernel-based VMs (KVMs).

Type 2 Hypervisors (Hosted)

For Type 2 hypervisors, there is a layer of host OS that sits between the physical server and the hypervisor. For this reason, we call these hypervisors "hosted hypervisors".

They are less common and mostly used for end-user virtualization.

They are known to have more latency compared to Type 1 due to their hosted nature. Type 2 hypervisors include Oracle Virtual Box or VMware Workstation.

Benefits of Virtualization

Cost Savings: The ability to run multiple virtual machines in one piece of physical infrastructure drastically reduces the footprint and the associated cost. Moreover, as this consolidation is done at the core, we don't need to maintain as many servers. We also have a reduction in electricity consumption and the overall maintenance cost.

Agility and Speed: Spinning up a virtual machine is a straightforward and quick approach. It's a lot simpler than provisioning entirely new infrastructure. For instance, if we need a development/test region for a team, it's much faster to provision a new VM for the system administrators. Besides, with an automated process in place, this task is swift and similar to other routine tasks.

SAN

SAN is an abbreviation of the Storage Area Network. Storage Area Network is a dedicated, specialized, and high-speed network which provides block-level data storage. It delivers the shared pool of storage devices to more than one server.

The main aim of SAN is to transfer the data between the server and storage device. It also allows for transferring the data between the storage systems.

Storage Area networks are mainly used for accessing storage devices such as tape libraries and disk-based devices from the servers.

It is a dedicated network which is not accessible through the LAN. It consists of hosts, switches, and storage devices which are interconnected using the topologies, protocols, and technologies.

Advantages of SAN

- It is more scalable.
- Security is also a main advantage of SAN. If users want to secure their data, then SAN is a good option to use. Users can easily implement various security measures on SAN.
- Storage devices can be easily added or removed from the network. If users need more storage, then they simply add the devices.
- The cost of this storage network is low as compared to others.
- Another big advantage of using the SAN (Storage Area Network) is better disk utilization.

What is SDN?

Software-defined networking (SDN) is an approach to network virtualization and containerization that helps optimize network resources and quickly adapt networks to changing business needs, applications, and traffic. It works by separating the network's control and data planes to create a software-programmable infrastructure.

With SDN, the functions of network orchestration, management, analytics, and automation become the job of SDN controllers. The controllers can take advantage of the scale, performance, and availability of modern cloud computing and storage resources. Increasingly, SDN controllers are built on platforms with open standards and APIs, enabling them to orchestrate, manage, and control network equipment from different vendors.

SDN delivers many business benefits. Separation of the control and data transport layers increases flexibility and accelerates time-to-market for new applications. The ability to respond more swiftly to issues and outages improves network availability. Finally, programmability makes it easier for IT organizations to automate network functions and reduce operating costs.

SDN dovetails with another technology, Network Functions Virtualization (NFV). NFV offers the ability to virtualize appliance-based network functions such as firewalls, load balancers, and WAN accelerators. The centralized control that SDN provides can efficiently manage and orchestrate these virtual network functions (VNFs) enabled by NFV.

What is Cloud API?

A cloud application programming interface (cloud API) enables applications to communicate and transfer information to one another in the cloud. Cloud APIs essentially enable you to develop applications and services in the cloud. APIs also connect multiple clouds or connect cloud and on-premises apps.

Introduction to Cloud

Cloud computing is the delivery of computing resources *as a service*, meaning that the resources are owned and managed by the cloud provider rather than the end user. Those resources may include anything from browser-based software applications (such as Tik Tok or Netflix), third party data storage for photos and other digital media (such as iCloud or Dropbox), or third-party servers used to support the computing infrastructure of a business, research, or personal project.

Before the broad proliferation of cloud computing, businesses and general computer users typically had to buy and maintain the software and hardware that they wished to use. With the growing availability of cloud-based applications, storage, services, and machines, businesses and consumers now have access to a wealth of on-demand computing resources as internet-accessed services. Shifting from on-premise software and hardware to networked remote and distributed resources means cloud users no longer have to invest the labor, capital, or expertise required for buying and maintaining these computing resources themselves. This unprecedented access to computing resources has given rise to a new wave of cloud-based businesses, changed IT practices across industries, and transformed many everyday computer-assisted practices. With the cloud, individuals can now work with colleagues over video meetings and other collaborative platforms, access entertainment and educational content on demand, communicate with household appliances, hail a cab with a mobile device, and rent a vacation room in someone's house.

Defining Cloud Computing

The National Institute of Standards and Technology (NIST), a non-regulatory agency of the United States Department of Commerce with a mission to advance innovation, defines cloud computing as:

a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

NIST lists the following as the five essential characteristics of cloud computing:

- On-demand self-service: Cloud resources can be accessed or provisioned without human interaction. With this model, consumers can gain immediate access to cloud services upon signup. Organizations can also create mechanisms for allowing employees, customers, or partners to access internal cloud services on demand according to predetermined logics without needing to go through IT services.
- Broad network access: Users can access cloud services and resources through any device and in any networked location provided that they have permission.
- Resource pooling: Cloud provider resources are shared by multiple tenants while keeping the data of individual clients hidden from other clients.
- Rapid elasticity: Unlike on-premise hardware and software, cloud computing resources can be rapidly increased, decreased, or otherwise modified based on the cloud user's changing needs.
- Measured service: Usage of cloud resources is metered so that businesses and other cloud users need only pay for the resources they use in any given billing cycle.

Cloud Delivery Models

Cloud resources are provided in a variety of different delivery models that offer customers different levels of support and flexibility.

Infrastructure as a Service (IaaS)

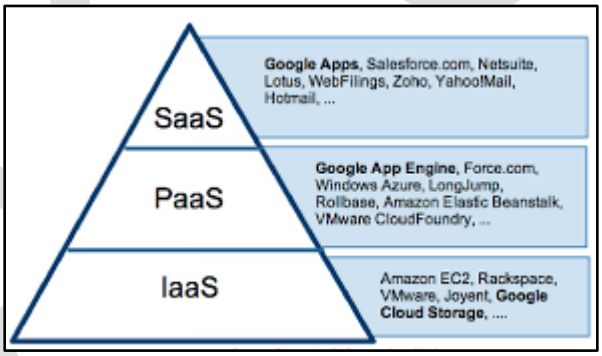
IaaS is the on-demand delivery of computing infrastructure, including operating systems, networking, storage, and other infrastructural components. Acting much like a virtual equivalent to physical servers, IaaS relieves cloud users of the need to buy and maintain physical servers while also providing the flexibility to scale and pay for resources as needed. IaaS is a popular option for businesses that wish to leverage the advantages of the cloud and have system administrators who can oversee the installation, configuration, and management of operating systems, development tools, and other underlying infrastructure that they wish to use. However, IaaS is also used by developers, researchers, and others who wish to customize the underlying infrastructure of their computing environment. Given its flexibility, IaaS can support everything from a company's computing infrastructure to web hosting to big data analysis.

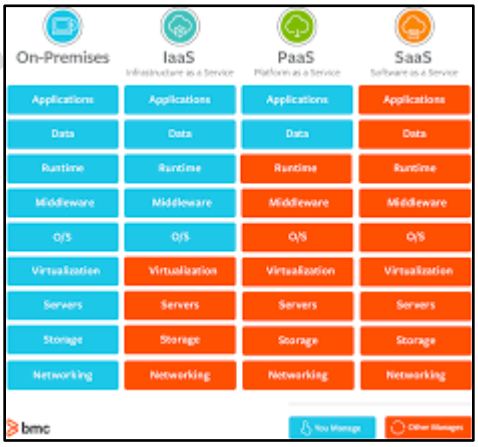
Platform as a Service (PaaS)

PaaS provides a computing platform where the underlying infrastructure (such as the operating system and other software) is installed, configured, and maintained by the provider, allowing users to focus their efforts on developing and deploying apps in a tested and standardized environment. PaaS is commonly used by software developers and developer teams as it cuts down on the complexity of setting up and maintaining computer infrastructure, while also supporting collaboration among distributed teams. PaaS can be a good choice for developers who don't have the need to customize their underlying infrastructure, or those who want to focus their attention on development rather than DevOps and system administration.

Software as a Service (SaaS)

SaaS providers are cloud-based applications that users access on demand from the internet without needing to install or maintain the software. Examples include GitHub, Google Docs, Slack, and Adobe Creative Cloud. SaaS applications are popular among businesses and general users given that they're often easy to adopt, accessible from any device, and have free, premium, and enterprise versions of their applications. Like PaaS, SaaS abstracts away the underlying infrastructure of the software application so that users are only exposed to the interface they interact with.





Cloud Environments

Cloud services are available as public or private resources, each of which serves different needs.

Public Cloud

The public cloud refers to cloud services (such as virtual machines, storage, or applications) offered publicly by a commercial provider to businesses and individuals. Public cloud resources are hosted on the commercial provider's hardware, which user's access through the internet. They are not always suitable for organizations in highly-regulated industries, such as healthcare or finance, as public cloud environments may not comply with industry regulations regarding customer data.

Private Cloud

The private cloud refers to cloud services that are owned and managed by the organization that uses them and available only to the organization's employees and customers. Private clouds allow organizations to exert greater control over their computing environment and their stored data, which can be necessary for organizations in highly-regulated industries. Private clouds are sometimes seen as more secure than public clouds as they are accessed through private networks and enable the organization to directly oversee their cloud security. Public cloud providers sometimes provide their services as applications that can be installed on private clouds, allowing organizations to keep their infrastructure and data on premise while taking advantage of the public cloud's latest innovations.

Hybrid Cloud and Multi Cloud

Many organizations use a hybrid cloud environment which combines public and private cloud resources to support the organization's computing needs while maintaining compliance with industry regulation. Multi Cloud environments are also common, which entail the use of more than one public cloud provider (for example, combining Amazon Web Services and DigitalOcean).

Benefits of Cloud Computing

Cloud computing offers a variety of benefits to individuals, businesses, developers, and other organizations. These benefits vary according to the cloud users goals and activities.

For Business and Industry

Prior to the proliferation of cloud computing, most businesses and organizations needed to purchase and maintain the software and hardware that supported their computing activities. As

cloud computing resources became available, many businesses began using them to store data, provide enterprise software, and deploy online products and services. Some of these cloud-based adoptions and innovations are industry-specific. In healthcare, many providers use cloud services that are specifically designed to store and share patient data or communicate with patients. In academia, educators and researchers use cloud-based teaching and research apps. But there are also a large number of general cloud-based tools that have been adopted across industries, such as apps for productivity, messaging, expense management, video conferencing, project management, newsletters, surveys, customer relations management, identity management, and scheduling. The rapid growth of cloud-based business apps and infrastructure shows that the cloud isn't just changing business IT strategy: it's a booming business in its own right.

Cloud-based technologies offer businesses several key advantages. First, they can help optimize IT costs. As businesses shift towards renting computing resources, they no longer have to invest as much in purchasing and maintaining on-premise IT infrastructure. Cloud computing is also enormously flexible, allowing businesses to rapidly scale (and only pay for) the computing resources they actually use. Cost, however, is not the only consideration that drives cloud adoption in business. Cloud-based technologies can help make internal IT processes more efficient as they can be accessed on demand by employees without needing to go through IT approval processes. Cloud-based apps can improve collaboration across a business as they allow for real-time communication and data sharing.

For Independent Developers

Computing resources that were once only affordable to large companies and organizations are now available on demand through an internet connection and at a fraction of their previous cost. In effect, independent developers can rapidly deploy and experiment with cloud-based apps. Cloud-based apps for sharing code (such as GitHub) have also made it easier for developers to build upon and collaborate on open source software projects. Additionally, cloud-based educational platforms and interactive coding tutorials have expanded access to developer education, enabling individuals without formal technical training to learn to code in their own time.

Altogether, these cloud-based computing and educational resources have helped lower the barriers to learning developer skills and deploying cloud-based apps. Formal training, company support, and massive amounts of startup capital are no longer necessary for individuals to experiment with creating and deploying apps, allowing for more individuals to participate in cloud development, compete with established industry players, and create and share apps as side projects.

For Researchers

As machine learning methods become increasingly important in scientific research, cloud computing has become essential to many scientific fields, including astronomy, physics, genomics, and artificial intelligence. The massive amount of data collected and analyzed in machine learning and other data-intensive research projects often require computing resources that scale beyond the capacity of hardware owned by an individual researcher or provisioned by the university. Cloud computing allows researchers to access (and only pay for) computing resources as their workloads require and allows for real-time collaboration with research partners across the globe. Without commercial cloud providers, a majority of academic machine learning research would be limited to individuals with access to university-provisioned, high-powered computing resources.

For Educators and Students

Cloud computing has also provided students with tools for supplementing their education and opportunities to put their technical skills into practice as they learn. Cloud-based apps for sharing, teaching, and collaborating on code and data (such as GitHub and Jupyter Notebooks) enable students to learn technical skills in a hands-on manner by studying, deploying, and contributing to open source software and research projects relevant to their field or professional aspirations. And just like independent developers, students are able to use cloud computing resources to share their code and apps with the public and reap the satisfaction of understanding the real-world application of their skills.

Students, researchers, and educators can also take advantage of cloud computing resources to support personalized academic infrastructure and practice greater control over their computing environments. Some academics prefer this approach as it lets them pick which applications they use, customize the functionality and design of these tools, and limit or prohibit the collection of data. There are also a growing number of cloud-based applications developed specifically for academic purposes that supplement or provide alternatives to traditional academic IT offerings. Voyant Tools offers students and researchers a code-free method for providing textual analysis on documents of their choosing and The HathiTrust provides access to its digital collection of millions of volumes. Reclaim Hosting, Commons in a Box, the Modern Language Humanities Commons, and Manifold offer educational, publishing, and networking tools designed specifically for academic communities.

For Community Infrastructure

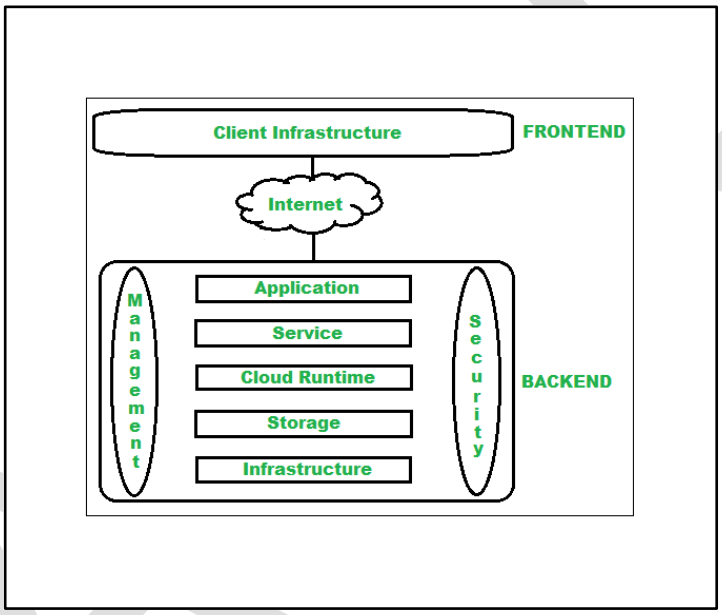
Some individuals and communities choose to install and manage their own cloud-based software to serve community needs and values, customize functionality, protect user data, and have more control over their computing environment. Open source software, such as social media tools like Mastodon, video conferencing software like Jitsi, collaborative text editors like Etherpad, and

web chat tools like Rocket Chat, provide alternatives to SaaS platforms that often limit user's control, privacy, and oversight over their computing environment. While often requiring more administrative work than SaaS applications or social media platforms, some communities prefer these options given ethical concerns about the use of personal data and company practices with popular platforms and SaaS applications.

Cloud Computing Architecture:

The cloud architecture is divided into 2 parts i.e.

1. Frontend
2. Backend



Architecture of cloud computing is the combination of both SOA (Service Oriented Architecture) and EDA (Event Driven Architecture). Client infrastructure, application, service, runtime cloud, storage, infrastructure, management and security all these are the components of cloud computing architecture.

1. Frontend:

Frontend of the cloud architecture refers to the client side of cloud computing system. Means it contains all the user interfaces and applications which are used by the client to access the cloud computing services/resources. For example, use of a web browser to access the cloud platform.

- **Client Infrastructure –** Client Infrastructure is a part of the frontend component. It contains the applications and user interfaces which are required to access the cloud platform.

- In other words, it provides a GUI (Graphical User Interface) to interact with the cloud.

2. Backend:

Backend refers to the cloud itself which is used by the service provider. It contains the resources as well as manages the resources and provides security mechanisms. Along with this, it includes huge storage, virtual applications, virtual machines, traffic control mechanisms, deployment models, etc.

1. **Application**
   Application in backend refers to a software or platform to which a client accesses. Means it provides the service in the backend as per the client requirement.

2. **Service**
   Service in backend refers to the major three types of cloud based services like SaaS, PaaS and IaaS. Also manages which type of service the user accesses.

3. **RuntimeCloud-**
   Runtime cloud in backend provides the execution and Runtime platform/environment to the Virtual machine.

4. **Storage**
   Storage in the backend provides flexible and scalable storage service and management of stored data.

5. **Infrastructure**
   Cloud Infrastructure in backend refers to the hardware and software components of cloud like it includes servers, storage, network devices, virtualization software etc.

6. **Management**
   Management in backend refers to management of backend components like application, service, runtime cloud, storage, infrastructure, and other security mechanisms etc.

7. **Security**
   Security in the backend refers to implementation of different security mechanisms in the backend for secure cloud resources, systems, files, and infrastructure to end-users.

8. **Internet**
   Internet connection acts as the medium or a bridge between frontend and backend and establishes the interaction and communication between frontend and backend.

Benefits of Cloud Computing Architecture:

- Makes the overall cloud computing system simpler.
- Improves data processing requirements.
- Helps in providing high security.
- Makes it more modularized.
- Results in better disaster recovery.
- Gives good user accessibility.
- Reduces IT operating costs.

What is AWS?

AWS stands for Amazon Web Services.

The AWS service is provided by Amazon that uses distributed IT infrastructure to provide different IT resources available on demand. It provides different services such as infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS). Amazon launched AWS, a cloud computing platform to allow the different organizations to take advantage of reliable IT infrastructure.

Uses of AWS

- A small manufacturing organization uses their expertise to expand their business by leaving their IT management to the AWS.
- A large enterprise spread across the globe can utilize the AWS to deliver the training to the distributed workforce.
- An architecture consulting company can use AWS to get the high-compute rendering of construction prototypes.
- A media company can use the AWS to provide different types of content such as ebox or audio files to the worldwide files.

What is Amazon 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.Features of Amazon EC2 Amazon EC2 provides the following features:

- Virtual computing environments, known as instances
- 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)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as instance store volumes
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
- 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)

Following are the list of important terms needs to know before creating EC2 instances:

➢ Amazon Machine Image (AMI)

➢ Instance Type

➢ Network

➢ Subnet

➢ Public IP

➢ Elastic IP

➢ Private IP

➢ Placement Group

➢ Root Volume

➢ Security Group

➢ Key Pair

Amazon Machine Image: An Amazon Machine Image (AMI) Provides the information required to launch an instance. An AMI Includes the following, one or more Elastic Block Store snapshot, a template for the root volume of the instance (for example Operating system, software, configurations etc.)

Instance Type: Instance types comprise varying combinations of CPU, Memory, storage & Networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications.

Subnet: Subnet is a sub network in your virtual network of your Amazon Network. By default there is one subnet per availability zone.

Public IP: A public IP is an IP Address which can be used to access the internet and allow communication over the internet. Public IP will be assigned by amazon and it is dynamic. If you stop and start your EC2 instance, The public IP will change.

Elastic IP(EIP): Elastic IP is a kind of Fixed Public IP address which we can attach to our Instances. Elastic IP will not change if we stop & Start our EC2 instances. We need to request an EIP from Amazon and it will be free if we attach to any instances, if you keep this EIP unused in your account then it will be charged after the initial 1st hour.

Private IP: Private IP can be used to establish the communication within the same network only, Private (internal) addresses are not routed on the Internet and no traffic can be sent to them from the Internet, meaning no internet access will be available over private addresses.

Placement group: is a logical grouping of instances within a single availability zone. AWS provides three

types of placement groups

➢ Cluster

➢ Partition

➢ Spread

Cluster: A cluster placement group is a logical grouping of instances within a single Availability Zone. Instances in the same cluster placement group enjoy a higher per-flow throughput limit for TCP/IP traffic and are placed in the same high-bisection bandwidth segment of the network.

The following image shows instances that are placed into a cluster placement group.

Partition Placement Group: Partition placement groups help reduce the likelihood of correlated

Hardware failures for your application. When using partition placement groups, Amazon EC2 divides each group into logical segments called partitions. Amazon EC2 ensures that each partition within a placement group has its own set of racks. Each rack has its own network and power source. No two partitions within a placement group share the same racks, allowing you to isolate the impact of hardware failure within your application.

Spread: A spread placement group is a group of instances that are each placed on distinct racks, with each rack having its own network and power source.

The following image shows seven instances in a single Availability Zone that are placed into a spread placement group. The seven instances are placed on seven different racks. Spread placement groups are recommended for applications that have a small number of critical instances that should be kept separate from each other. A spread placement group can span multiple Availability Zones in the same Region. You can have a maximum of seven running instances per Availability Zone per group.

Root Volume: The storage which we used to install the Operating system for instance is called as root volume (Ex: C:\ Drive). The following volume types are supported as root volumes: General purpose SSD, Provisioned IOPS SSD, and Magnetic.

Security Group: A Security group acts as a virtual firewall for your instance to control incoming & Outgoing traffic. Security groups to be attached and we can attach 5 security groups to each instance.

Following are the basic characteristics of a security groups:

➢ you can specify allow rules, but not deny rules

➢ you can specify separate rules for inbound and outbound traffic.

➢ Security group rules enable you to filter traffic based on protocols and port numbers.

➢ Security groups are stateful — if you send a request from your instance, the response traffic for that request is allowed to flow in regardless of inbound security group rules.

➢ when you create a new security group, it has no inbound rules.

➢ by default, a security group includes an outbound rule that allows all outbound traffic.

➢ by default we can create 2500 security groups per region, can be increased upto 5000 per region

➢ 60 inbound and outbound rules per security group

Key Pair: Key pair is a combination of public key and private key which can be used to encrypt and decrypt the data, is a set of security credentials that you use to prove your identity when connecting to an instance. Amazon EC2 stores the public key and the user stores the private key.

Amazon EC2 Instance Types:

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases.

Instance types comprise varying combinations of CPU, memory, storage, and networking capacity each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

1. General Purpose

2. Compute Optimized

3. Accelerated Computing (GPU Optimized)

4. Memory Optimized

5. Storage Optimized

1. **General Purpose:**

General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads. These instances are ideal for applications that use these resources in equal proportions such as web servers and code repositories.

Ex: Mac, T4g, T3, T3a, T2, M6g, M5, M5a, M5n, M5zn, M4, A1

2. **Compute Optimized:**

Compute Optimized instances are ideal for compute bound applications that benefit from high performance processors. Instances belonging to this family are well suited for batch processing workloads, media transcoding, high performance web servers, high performance computing

Ex: C6g, C6gn, C5, C5a, C5n, C4

3. **Memory Optimized:**

Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory. Use case: Memory-intensive applications such as open-source databases, in-memory caches, and real time big data analytics

Ex: R6g, R5, R5a, R5b, R5n, R4, X2gd, X1e, X1, u, Z1d

4. **Accelerated Computing:**

Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating point number calculations, graphics processing, or data pattern matching,more efficiently than is possible in software running on CPUs.UseCase: Machine learning, high performance computing, computational fluid dynamics, computational finance, seismic analysis, speech recognition, autonomous vehicles, and drug discovery.

Ex: P4, P3, P2, Inf1, G4dn, G3, F1

5. **Storage Optimized:**

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications.

Ex: I3, I3en, D2, D3, D3en, H1

AWS Lambda

- AWS Lambda is used to execute backend code without worrying about the underlying architecture, you just upload the code and it runs, it's that simple!
- AWS Lambda is a compute service offered by Amazon.
- Amazon explains AWS Lambda (λ) as a 'serverless' compute service, meaning the developers don't have to worry about which AWS resources to launch, or how will they manage them, they just put the code on lambda and it runs, it's that simple! It helps you to focus on core-competency i.e. App Building or the code.
- Lambda is used to encapsulate Data centers, Hardware, Assembly code/Protocols, high-level languages, operating systems, AWS APIs.
- Lambda is a compute service where you can upload your code and create the Lambda function.
- Lambda takes care of provisioning and managing the servers used to run the code.
- While using Lambda, you don't have to worry about scaling, patching, operating systems, etc.
- But, if your code will be running for hours, and you expect a continuous stream of requests, you should probably go with EC2, because the architecture of Lambda is for a sporadic kind of workload, wherein there will be some quiet hours and some spikes in the no. of requests as well.
- For example, logging the email activity for say a small company, would see more activity during the day than in the night, also there could be days when there are less emails to be processed, and sometimes the whole world could start emailing you! In both cases, Lambda is at your service.
- Considering this use case for a big social networking company, where the emails are never ending because it has a huge user base, Lambda may not be the apt choice.

AWS Lambda manages all the administration such as

- Provisioning & capacity of compute fleet that offers a balance of memory, CPU, network & other resources
- Server & OS maintenance
- High availability & automatic scaling
- Monitoring fleet health

- Applying security patches
- Deploying your code
- Monitoring & logging your lambda function
- AWS Lambda runs your code on high availability compute infrastructure
- It execute your code only when needed & scales automatically from few requests per day to thousands per second
- You pay only for compute time you consume - no charge when your code is not running
- All you need to do is supply your code in form of one or more lambda functions to AWS Lambda, in one of the language that lambda supports & service can run the code on your behalf
- Typically the lifecycle of AWS Lambda based application includes authoring code, deploying code to AWS lambda & then monitoring & troubleshooting
- This is in exchange for flexibility which means that you cannot login to compute instances or customize the operating system or language runtime
- If you do not want to manage your own compute you can use EC2 or Elastic Beanstalk

Accessing Lambda

You can create, invoke, and manage your Lambda functions using any of the following interfaces:

- AWS Management Console – Provides a web interface for you to access your functions. For more information, see Lambda console.
- AWS Command Line Interface (AWS CLI) – Provides commands for a broad set of AWS services, including Lambda, and is supported on Windows, macOS, and Linux. For more information, see Using Lambda with the AWS CLI.
- AWS SDKs – Provide language-specific APIs and manage many of the connection details, such as signature calculation, request retry handling, and error handling. For more information, see AWS SDKs.
- AWS Cloud Formation – Enables you to create templates that define your Lambda applications. For more information, see AWS Lambda applications. AWS Cloud Formation also supports the AWS Cloud Development Kit (AWS CDK).
- AWS Server less Application Model (AWS SAM) – Provides templates and a CLI to configure and manage AWS server less applications. For more information, see SAM CLI.

Parameters	AWS Lambda	AWS EC2
Definition	AWS Lambda is a Platform as a Service (PaaS). It helps you to run and execute your backend code.	AWS EC2 Is an Infrastructure as a Service (laaS). It provides virtualized computing resources.
Flexibility	Does not offer any flexibility to log in to compute instances. It allows you to choose a customized operating system or language runtime.	Offers the flexibility to select the variety of instances, custom operating systems, security patches, and network, etc.
Installation process	You need to select your environment where you want to run the code and push the code into AWS Lambda.	For the first time in EC2, you have to choose the OS and install all the software required and then push your code in EC2.
Environment restrictions	It is restricted to a few languages like nodejs, java, Go, C#, powershell, python, Ruby.	No environmental restrictions. You can run any code or language.
Push	Write your code & push the code to AWS Lambda	For first time in EC2 you gave to choose OS & install all the software required & then push your code in EC2
Login	You cannot login to compute instances or customize the operating system or language runtime	You can select variety of OS, instance types, network & security patches, RAM & CPU etc

Bill	You pay only for compute time you consume - no charge when your code is not running	Pay per second or hour. Even if you are not using that EC2 instance bill is generated

How does Lambda work?

1. First you upload your code in one or more lambda function
2. AWS Lambda then execute the code on your behalf
3. After the code is invoked, lambda automatically take care of provisioning & managing the required server

Important terms in AWS Lambda:

★ **Function:** A function is a resource that you can invoke to run your code in AWS Lambda. A function has code that processes events & runtime that passes request & responses between lambda & function code

★ **Runtime:** it allows lambda functions in different languages to run in the same base execution environment. Runtime sits in between lambda service & your function code, relaying invocation events context information & responses between two.

★ **Event:** It is JSON formatted document that contains data the function to process

★ **Trigger/Event source:** AWS service such as Amazon SNS or a custom service that triggers your function & executes its logic

★ **Downstream resource:** AWS service such as DynamoDB tables or S3 buckets that your lambda function calls once it is triggered.

★ **Concurrency:** number of requests that your function is serving at any given time.

When Lambda triggers, you can use AWS Lambda to run your in response to -

- Events such as changes to data in an Amazon S3 bucket or Amazon DynamoDB table.
- To run your code in response to HTTP request using Amazon API gateway
- With these capabilities, you can use lambda to easily build data processing triggers for AWS services like Amazon S3 & Amazon DynamoDB, process streaming data stored in kiness or create your own backend that operates at AWS scale performance & security

  Example of S3

➢ The user create object in bucket

➢ Amazon S3 detects the object create event

- ➢ Amazon S3 invokes your lambda functions using the permission provided by the execution role.
- ➢ Amazon function knows which lambda function to invoke based on event source mapping that is stored in bucket notification configuration

AWS Lambda function configuration:

- Lambda function contains code & associated dependencies
- In addition lambda function also have configuration information associated with it
- Initially you specify configuration information when you create a lambda function
- Lambda provides API for you to update some of configuration data

Lambda function configuration information includes following key elements:

- Compute resources that you need. You only specify amount of memory you want to allocate from your lambda function
- AWS Lambda, allocated CPU power proportional to the memory by using the same ratio as general purpose Amazon EC2 instance type such as M3 type
- You can update configuration & request additional memory in 64MB increments from **128 MB to 3008 MB**
- Functions larger than 1536 MB allocated multiple CPU threads

Maximum execution time out

- You pay for AWS resources that are used to run your lambda function
- To prevent your lambda function to run indefinitely, you specify a timeout
- When specified timeout is reached, AWS lambda terminated your lambda function
- Default is 3 seconds & maximum is 900 seconds (15 minutes)

IAM Role

This is the role that lambda assumes when it executes lambda function on your behalf

AWS Lambda function - services it can access

- AWS service or non-AWS services
- AWS services running in AWS VPC (e.g. Redshift, elastic cache, RDS instance)
- Non AWS services running on EC2 instances in an AWS VPC
- AWS Lambda run your function code securely within a VPC default
- However to enable your lambda function to access resources inside your private VPC, you must provide additional VPC specific configuration information that includes VPC subnet ID & security group ID's

Different ways to invoke function

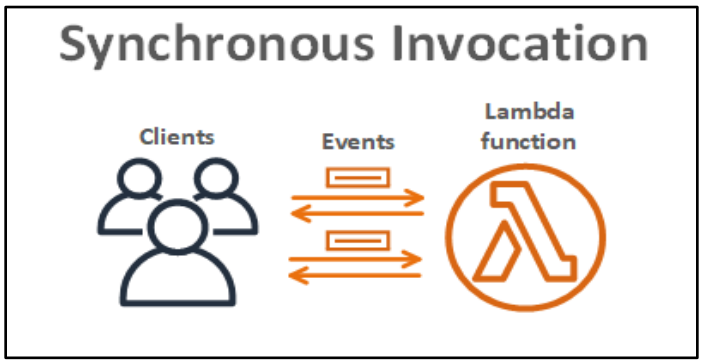
1. Synchronous invoke (Push)
2. Asynchronous invoke (Event)
3. Poll based invoke (Poll based)

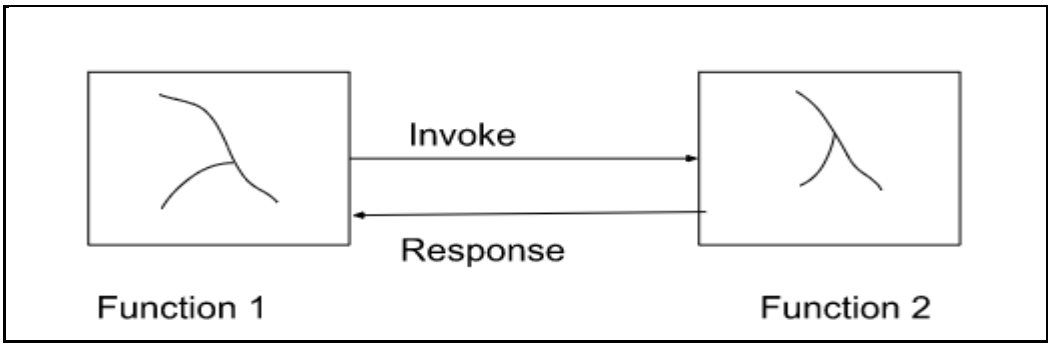
1. Synchronous invoke (Push)

Synchronous invoke are the most straightforward way to invoke your lambda function

In this model, your functions executes immediately when you perform the lambda invoke API call

Invocation flag specifies value of 'Request Response'

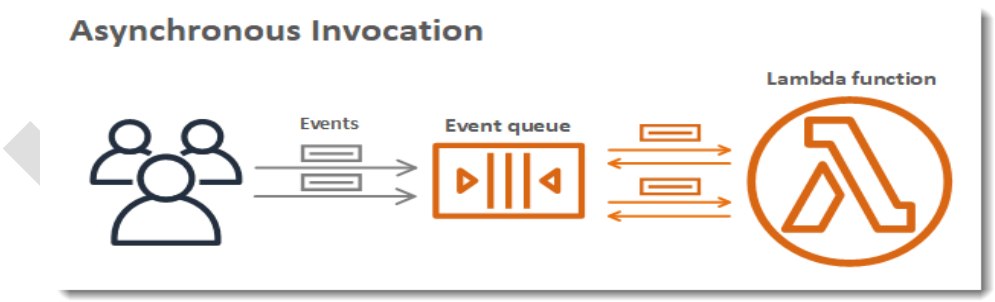


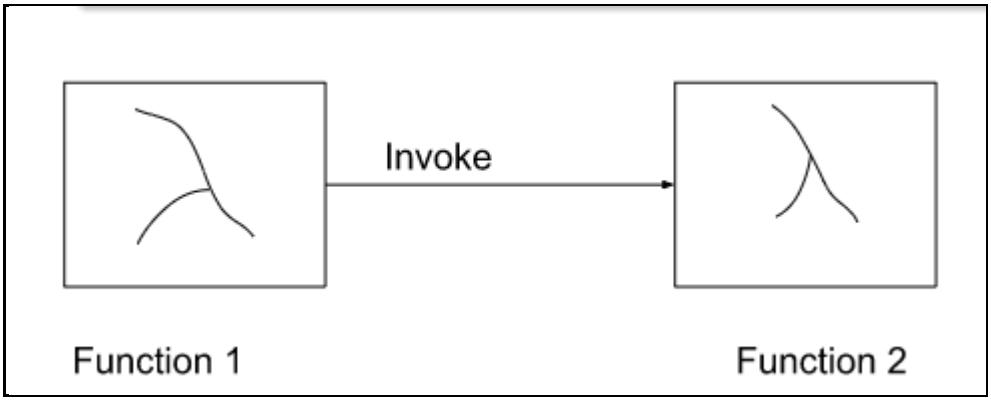


- Here is list of service that invokes lambda function synchronously
    - Elastic load balancer
    - Amazon Cognito
    - Cloudfront
    - API gateway
    - Amazon Lex
    - Kinesis data firehose

2. Asynchronous Invocation

- Lambda places event in queue & returns success response without additional information
- Lambda queues the event for processing & returns a response immediately
- You can configure lambda to send an invocation record to another service like SQS, SNS, lambda & event bridge
- Here is list of service that invokes lambda function asynchronously
    - Amazon S3
    - Amazon SNS
    - SES
    - Cloudformation
    - Cloudwatch logs
    - Cloudwatch events
    - AWS Codecommit
    - AWS Config



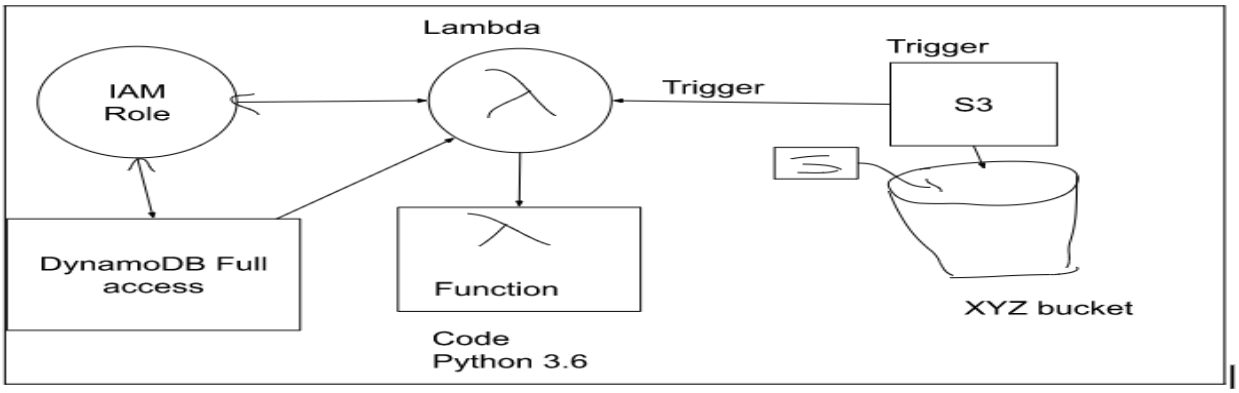


Does not wait for function 2 to finish or response from function 2

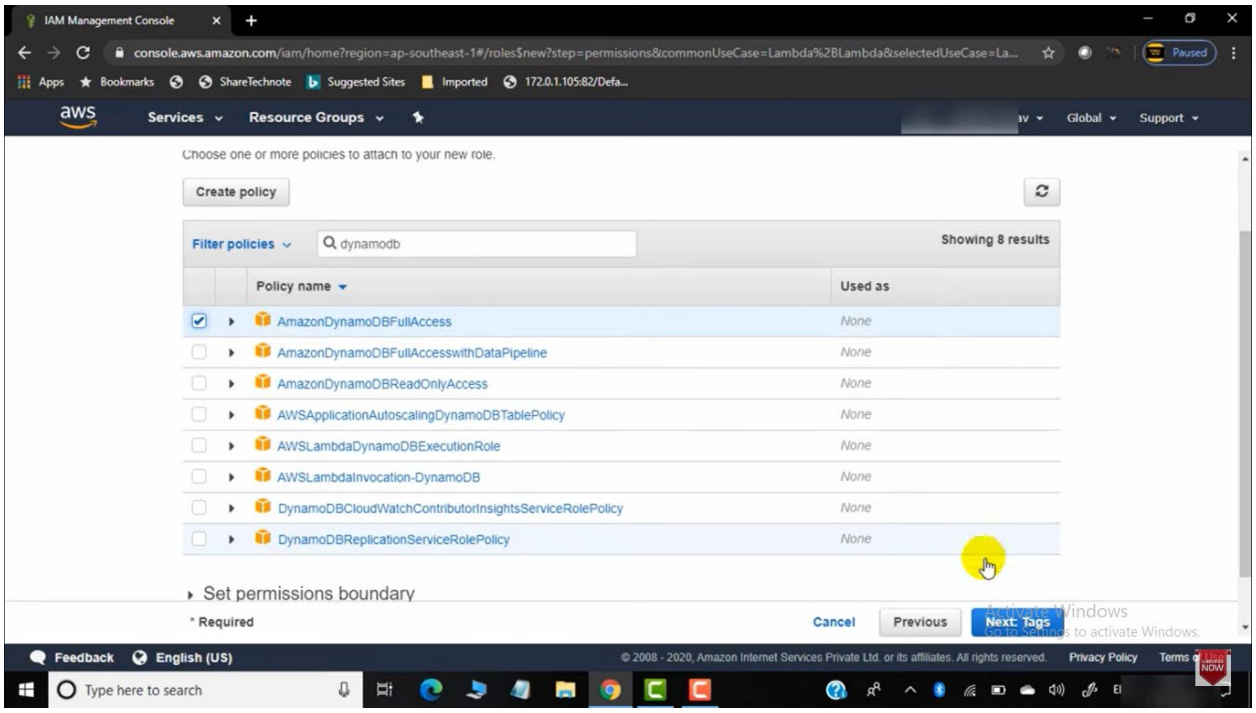
3. Poll based invocation

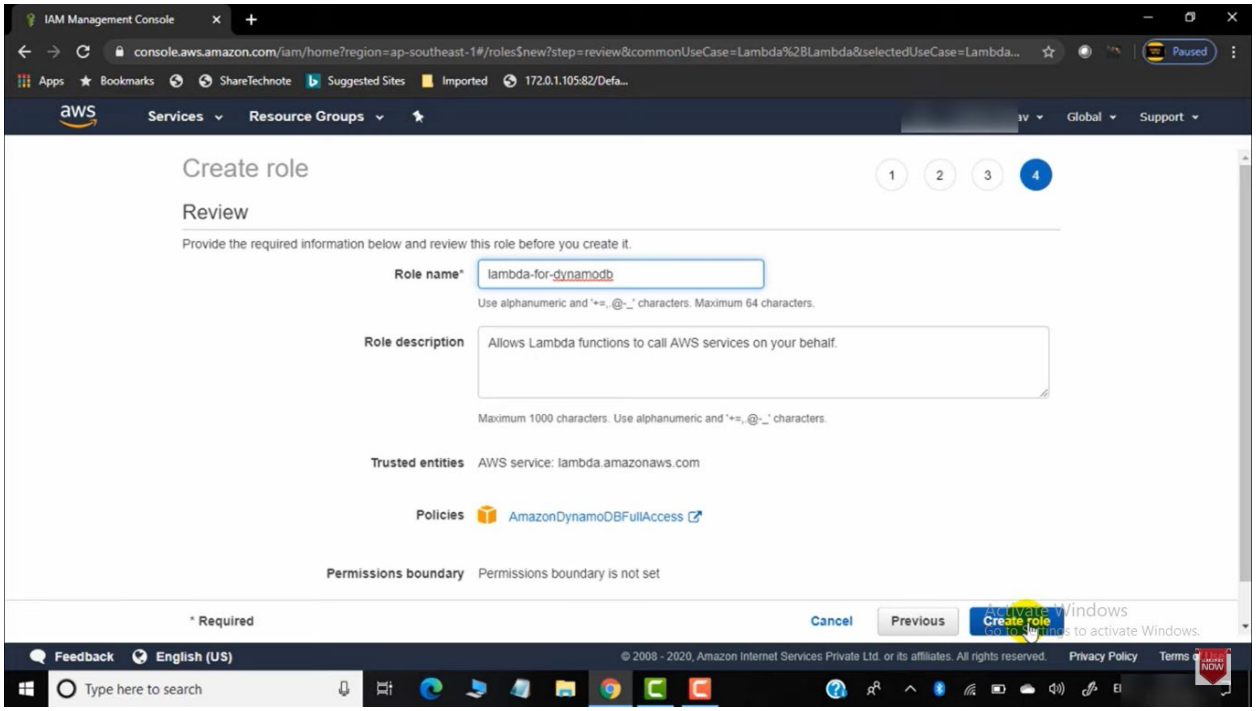
- It is designed to allow you to integrate with AWS stream & queue based service with no code or server management
- Lambda will poll following service on your behalf , retrieve records & invoke your function
- The following are supported service
    - Amazon kinesis
    - Amazon SQS
    - Amazon DynamoDB streams

Setup S3 trigger with Lambda

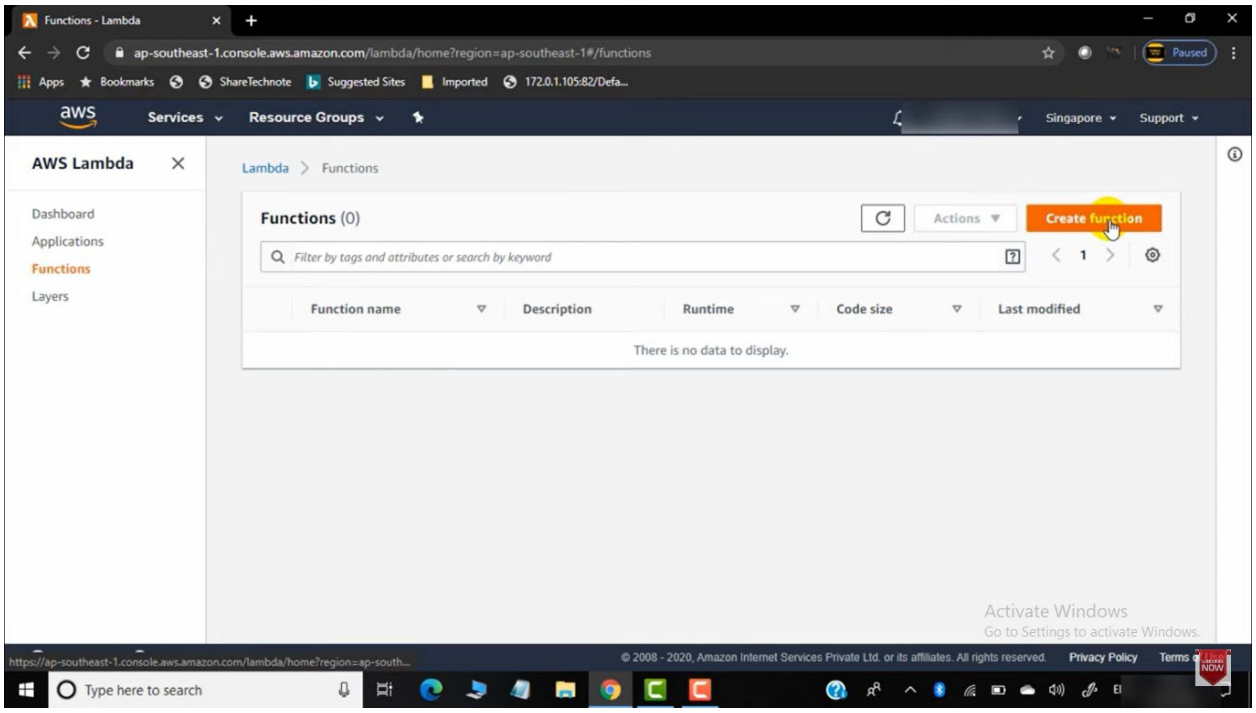


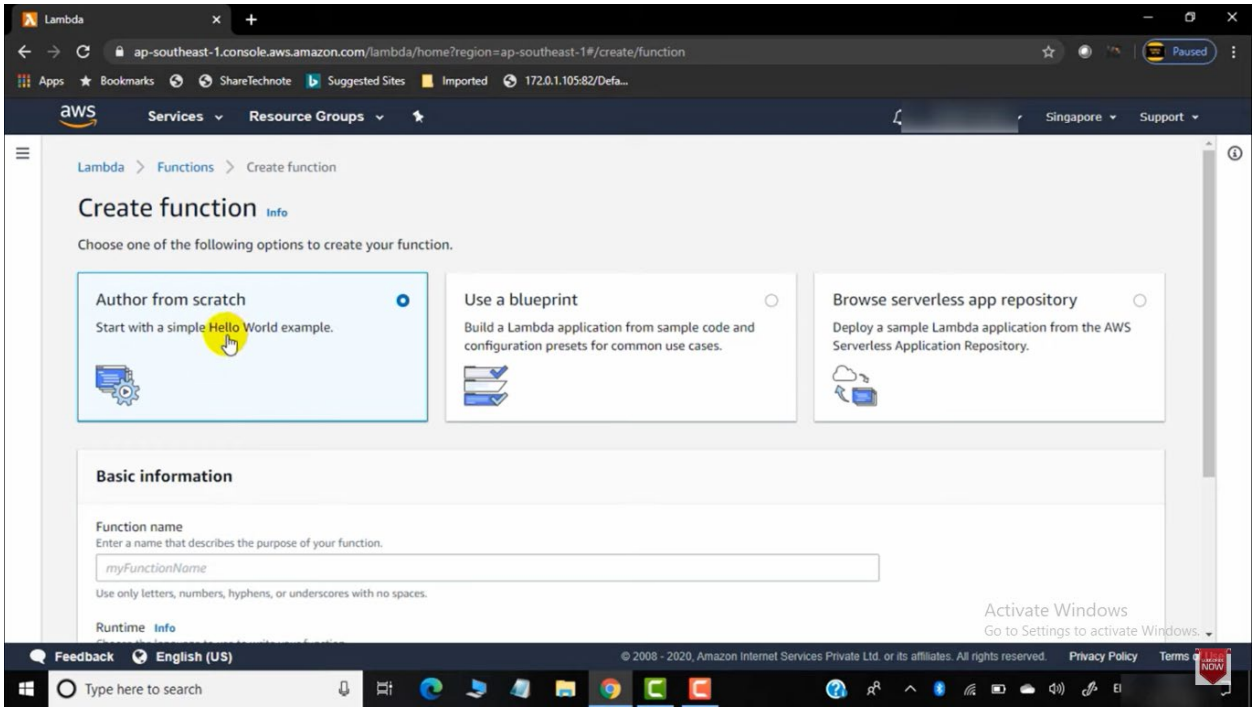
In IAM create a role for DynamoDB full access

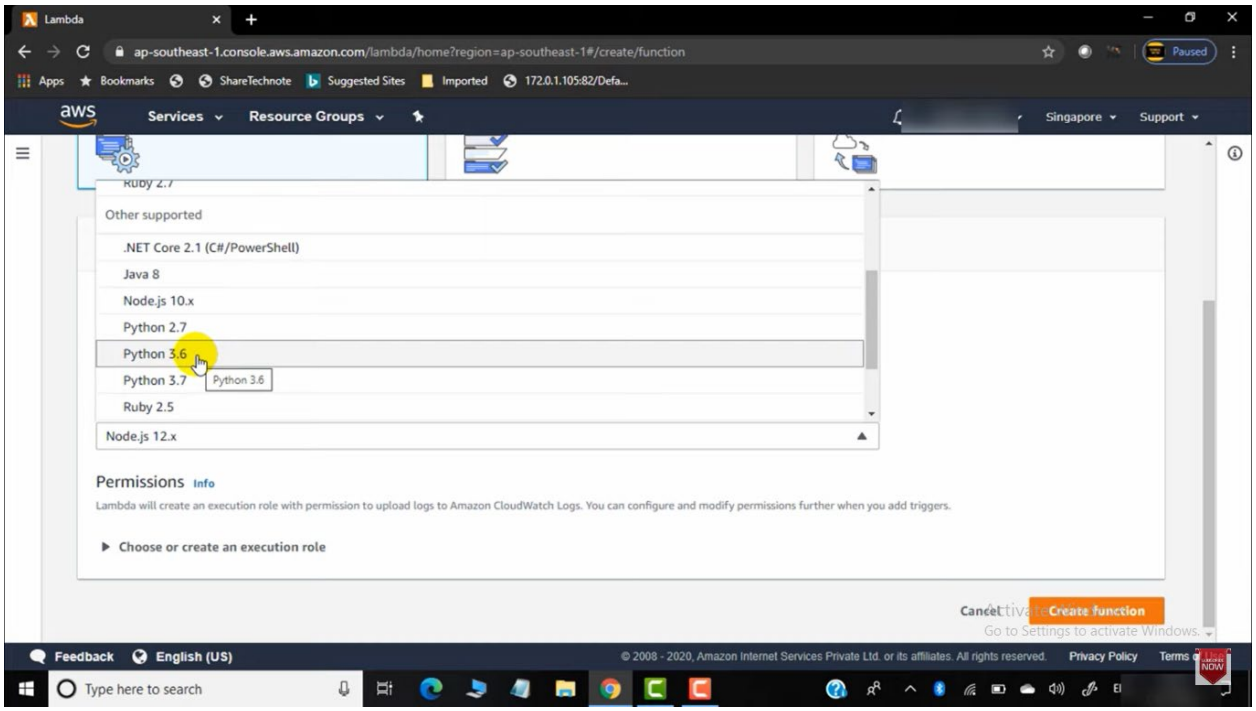


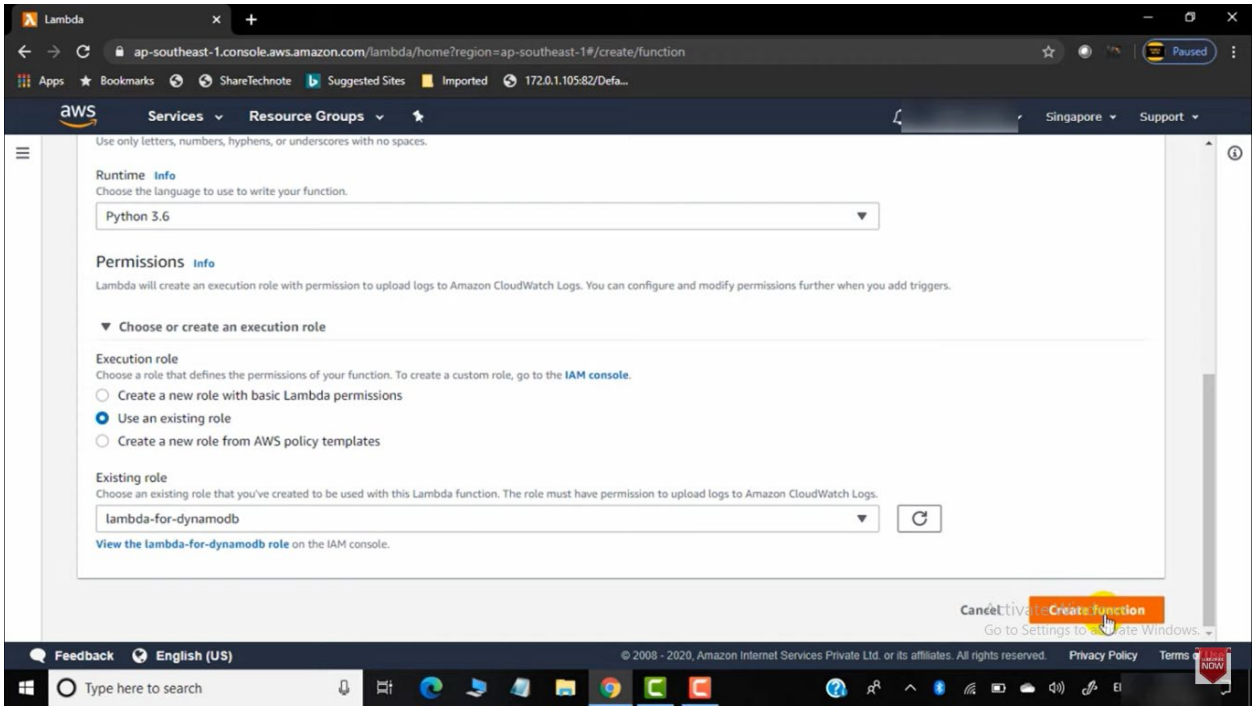


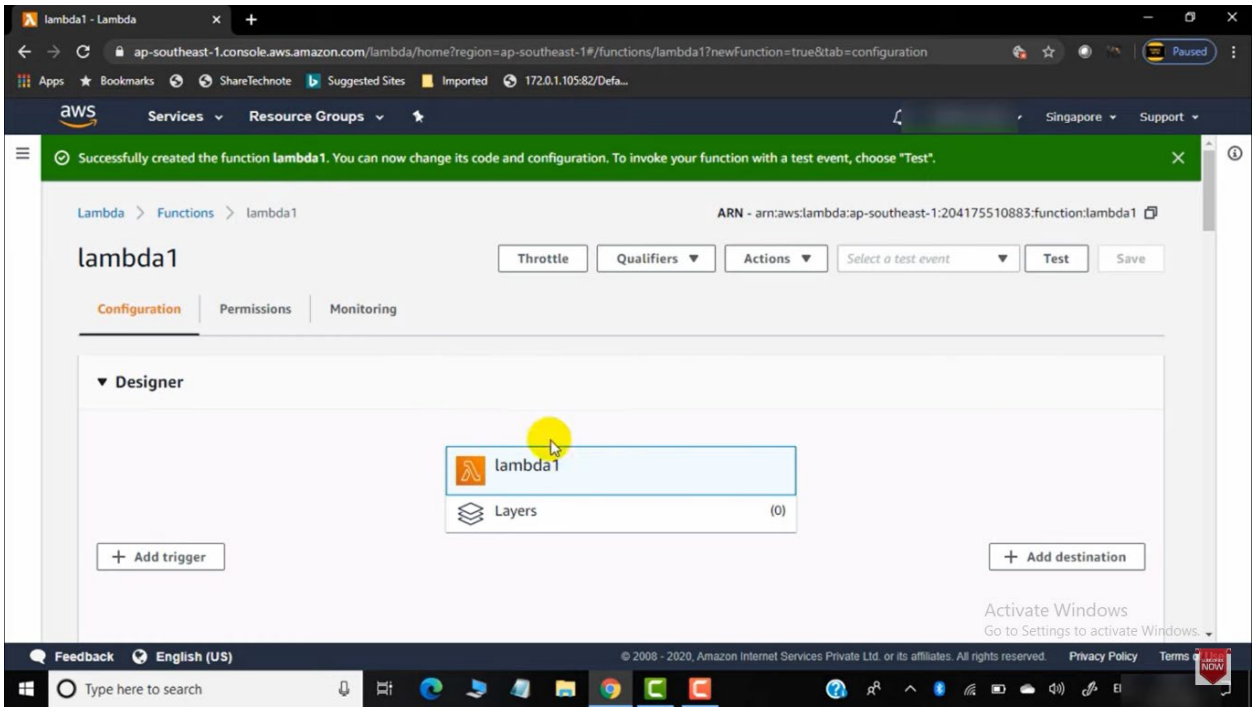
In AWS Lambda create a function

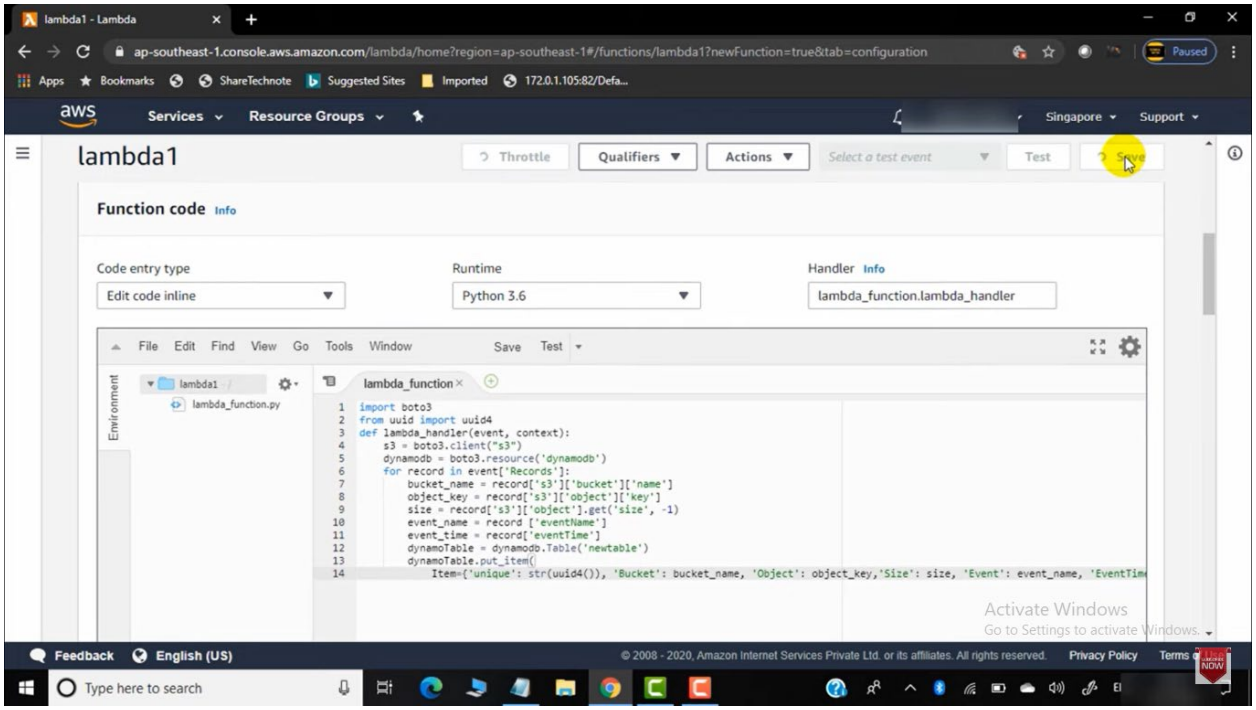





Lambda
ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/create/function
Apps
Bookmarks
ShareTechnote
Suggested Sites
Imported
172.0.1.105:82/Defa...
Paused
aws
Services
Resource Groups
Singapore
Support
Other supported
.NET Core 2.1 (C#/PowerShell)
Java 8
Node.js 10.x
Python 2.7
Python 3.6
Python 3.7
Ruby 2.5
Node.js 12.x
Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.
Choose or create an execution role
Cancel
Create function
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms
Type here to search


Lambda
ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/create/function
Apps
Bookmarks
ShareTechnote
Suggested Sites
Imported
172.0.1.105:82/Defa...
Paused
aws
Services
Resource Groups
Singapore
Support
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function.
Python 3.6
Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.
Choose or create an execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
lambda-for-dynamodb
View the lambda-for-dynamodb role on the IAM console.
Cancel
Create function
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms
Type here to search



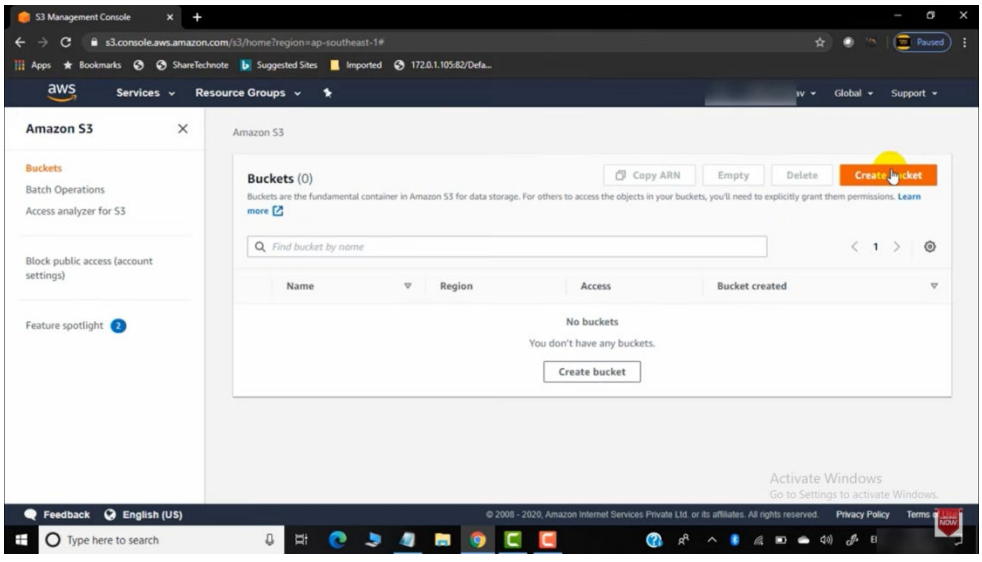


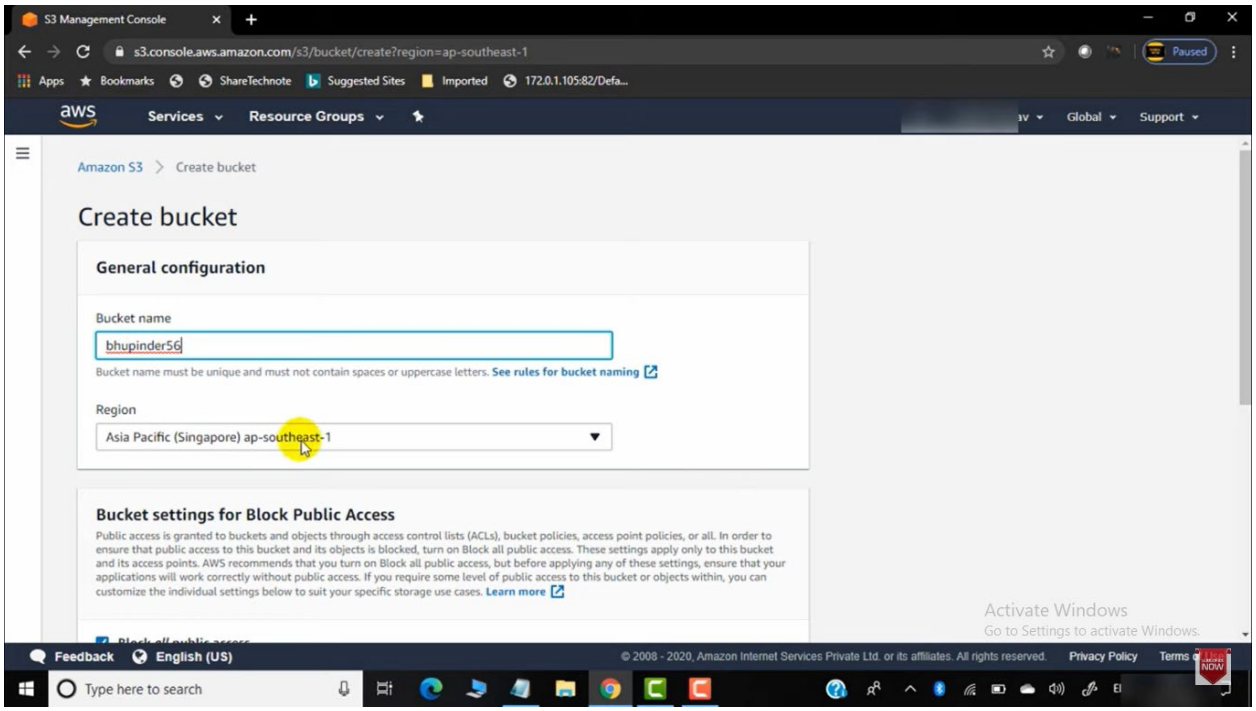
Add this code in lambda function

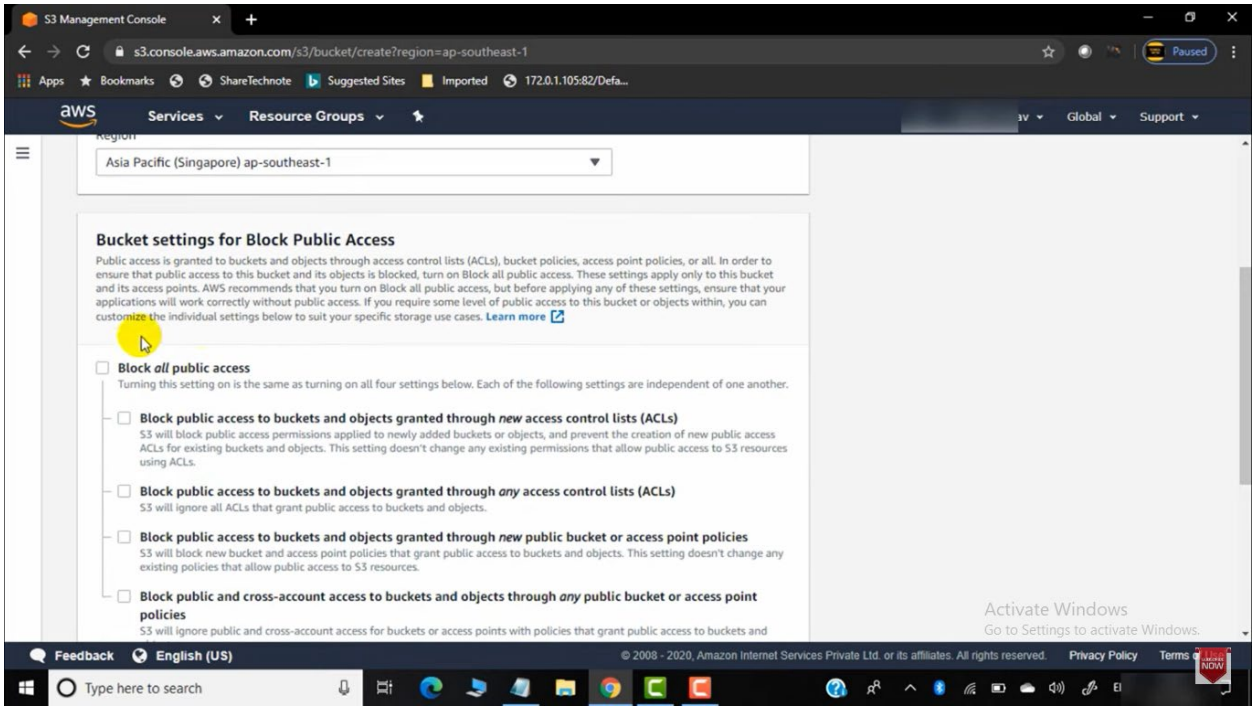
```
import boto3

from uuid import uuid4
```

```
def lambda_handler(event, context):
    s3 = boto3.client("s3")
    dynamodb = boto3.resource('dynamodb')
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        object_key = record['s3']['object']['key']
        size = record['s3']['object'].get('size', -1)
        event_name = record ['eventName']
        event_time = record['eventTime']
        dynamoTable = dynamodb.Table('newtable')
        dynamoTable.put_item(
            Item={'unique': str(uuid4()), 'Bucket': bucket_name, 'Object': object_key,'Size': size, 'Event': event_name, 'EventTime': event_time})
```

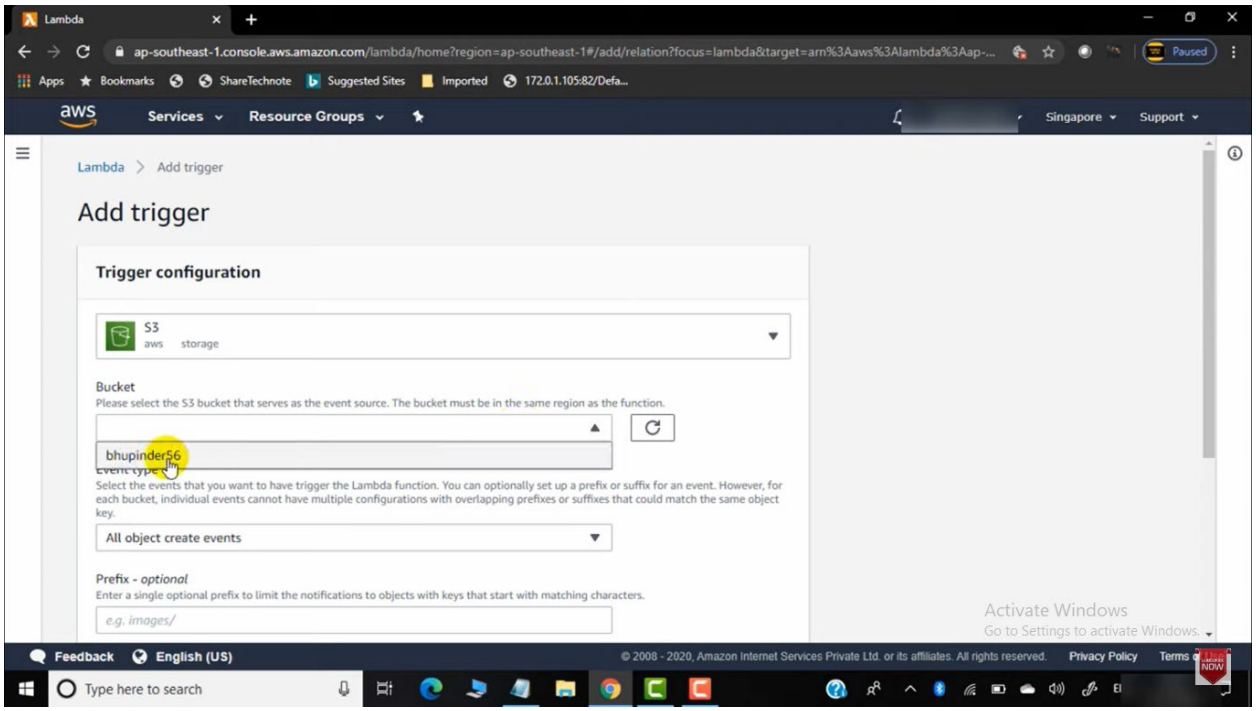
Now, In S3 create a bucket

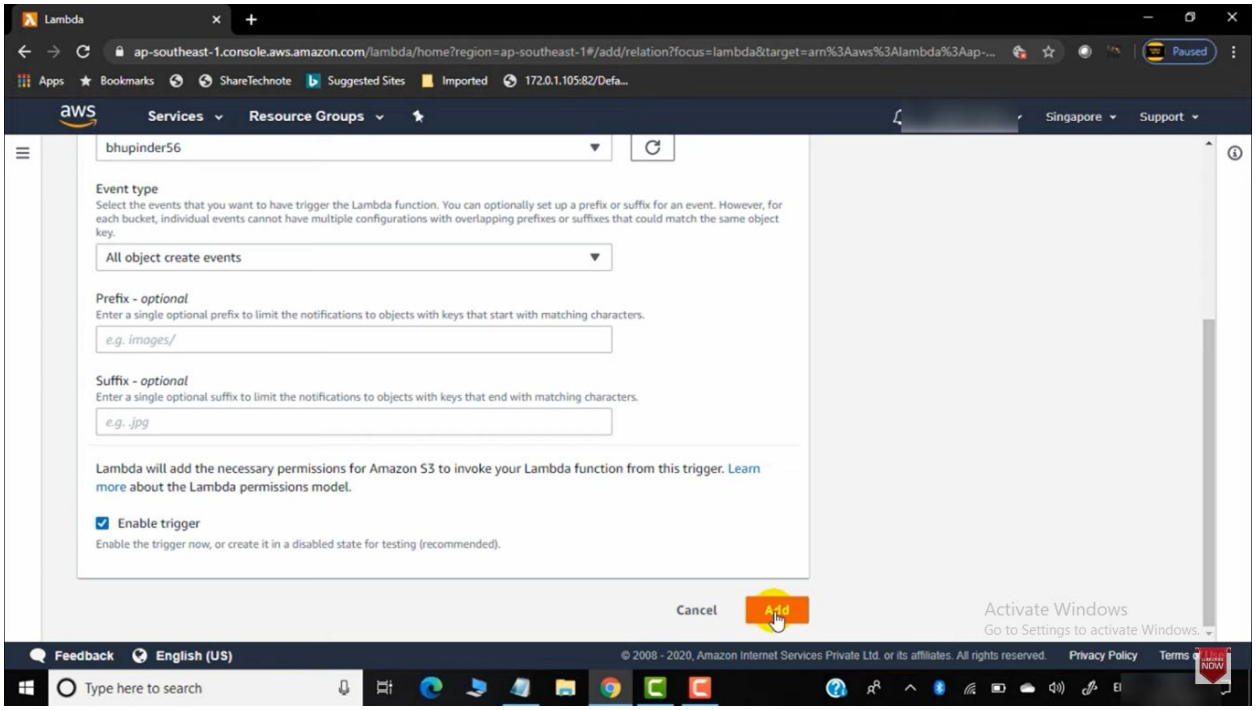




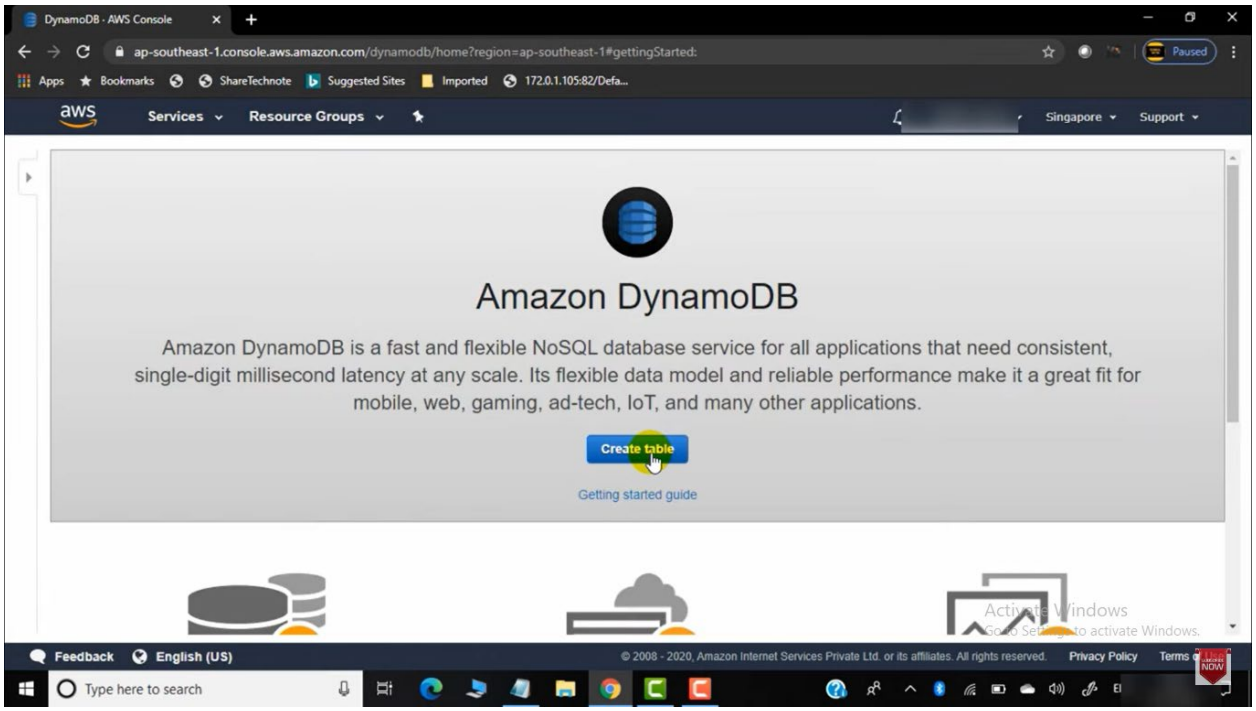


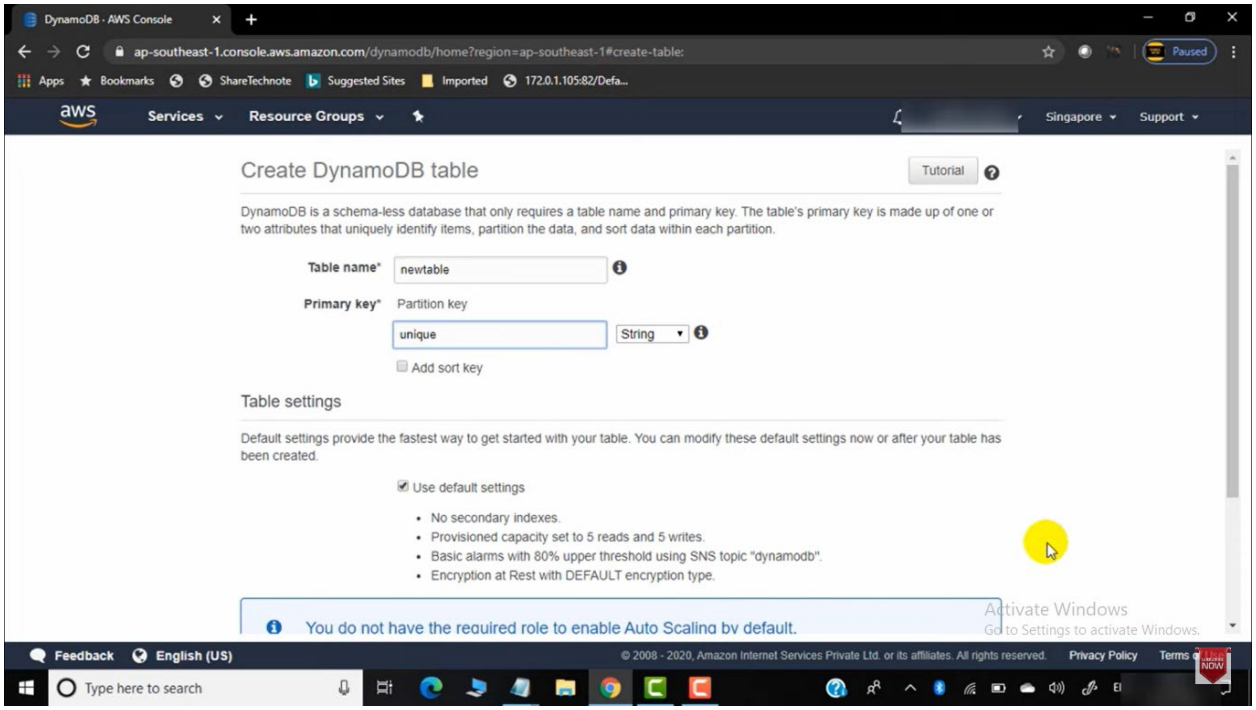
In Lambda add trigger

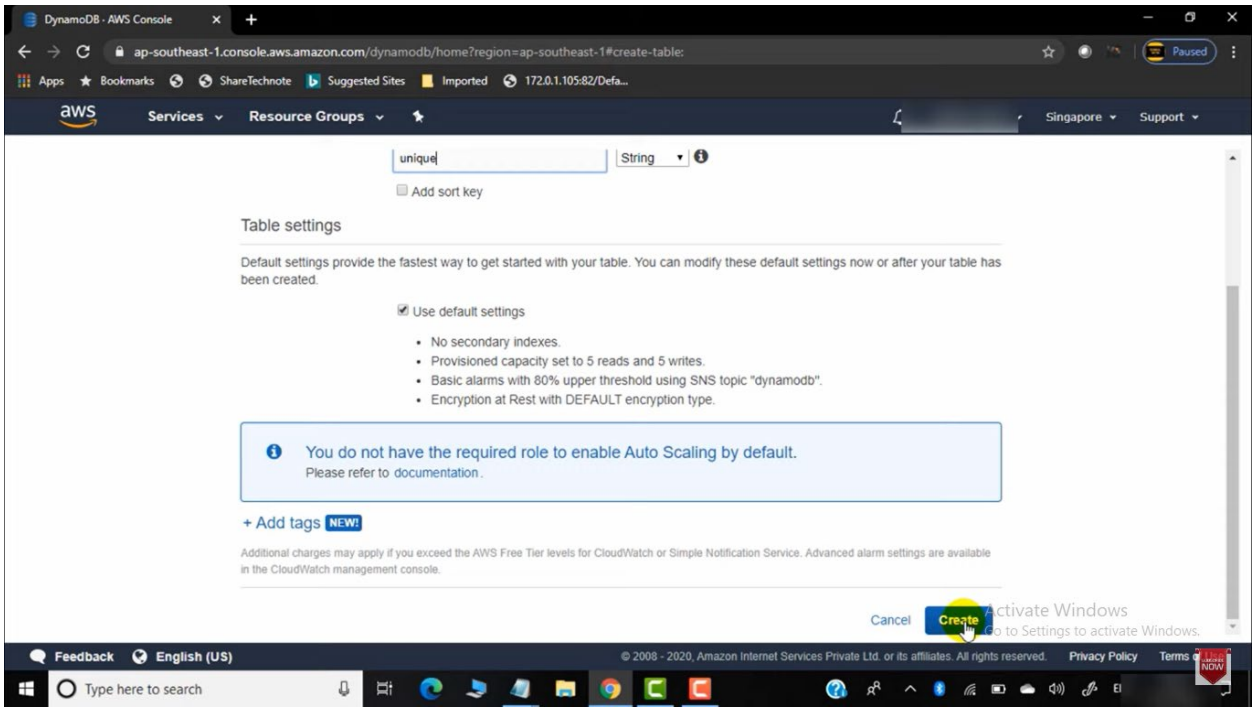




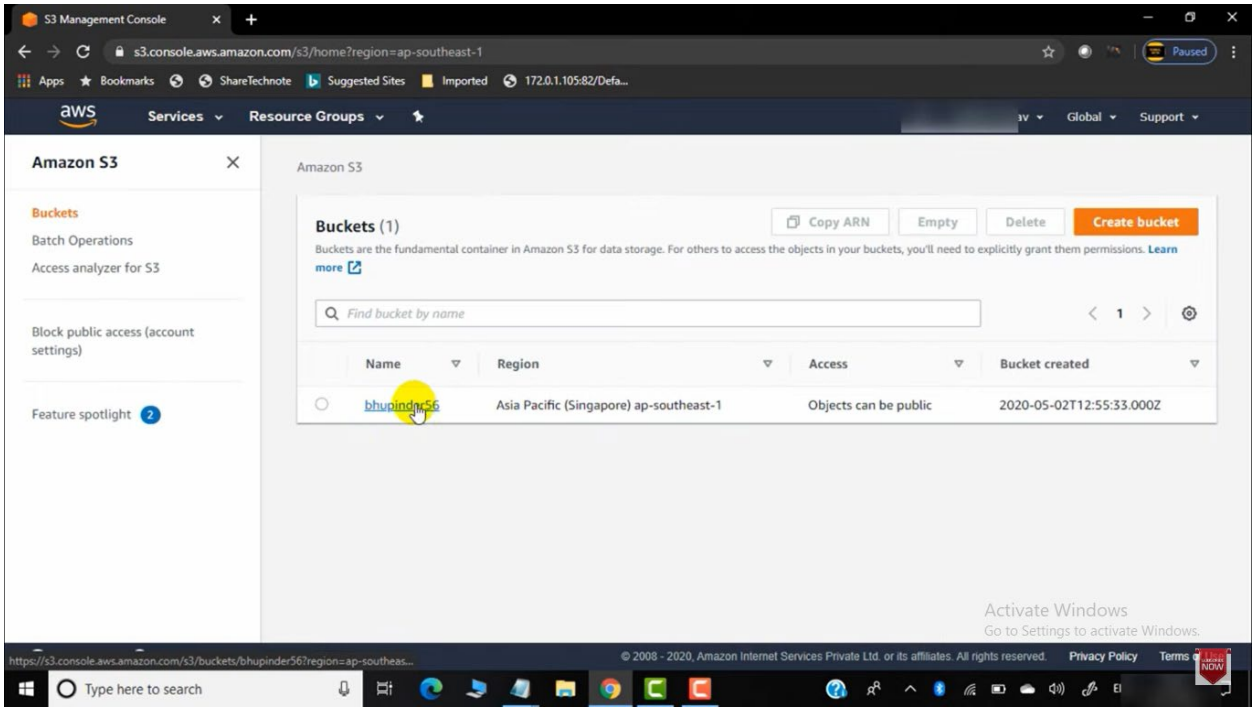
In Amazon DynamoDB create a table

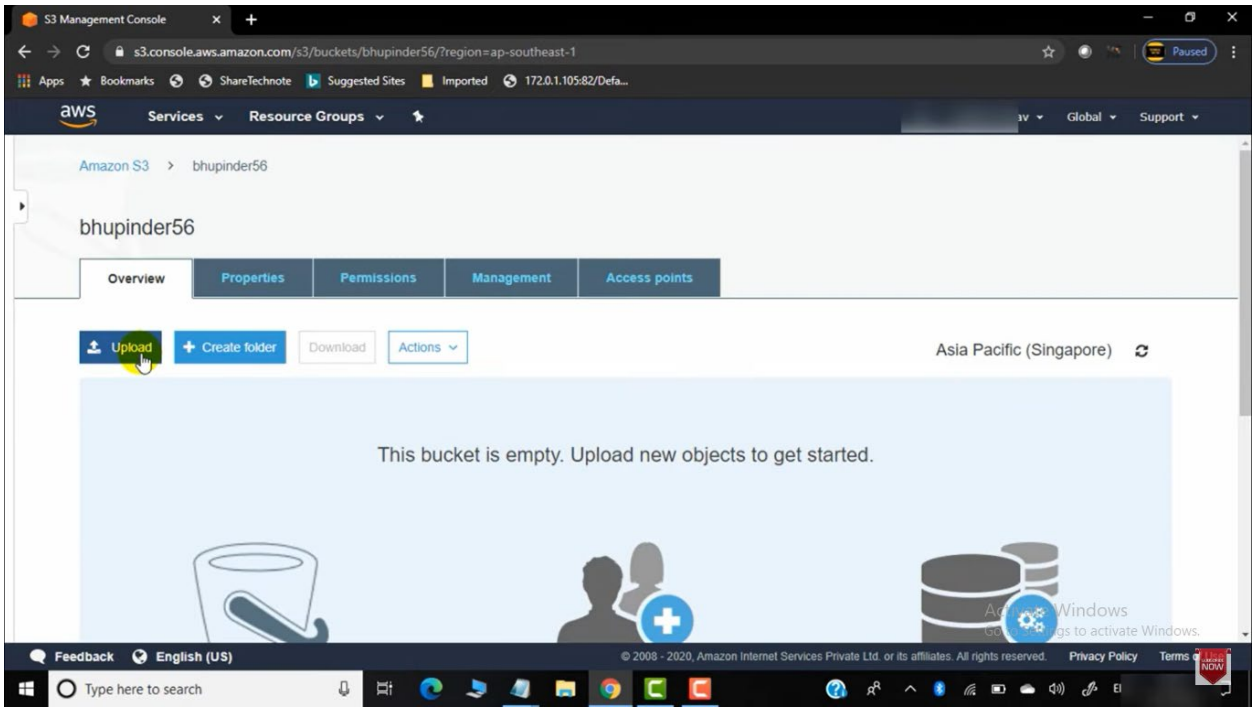

DynamoDB - AWS Console
ap-southeast-1.console.aws.amazon.com/dynamodb/home?region=ap-southeast-1#gettingStarted:
Services
Resource Groups
Singapore
Support
Amazon DynamoDB
Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.
Create table
Getting started guide
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms
Activate Windows
Type here to search

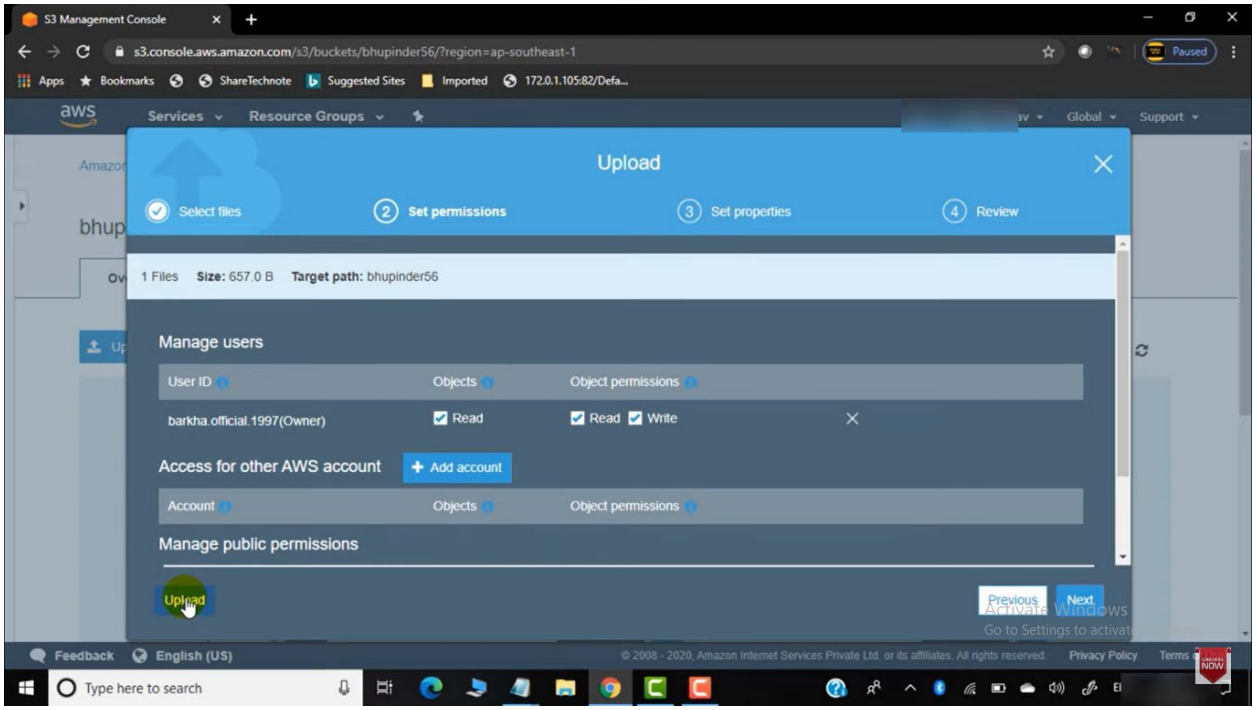

DynamoDB - AWS Console
ap-southeast-1.console.aws.amazon.com/dynamodb/home?region=ap-southeast-1#create-table:
Services
Resource Groups
Singapore
Support
Create DynamoDB table
Tutorial
DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.
Table name*
newtable
Primary key*
Partition key
unique
String
Add sort key
Table settings
Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.
Use default settings
No secondary indexes.
Provisioned capacity set to 5 reads and 5 writes.
Basic alarms with 80% upper threshold using SNS topic "dynamodb".
Encryption at Rest with DEFAULT encryption type.
You do not have the required role to enable Auto Scaling by default.
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms
Activate Windows
Type here to search

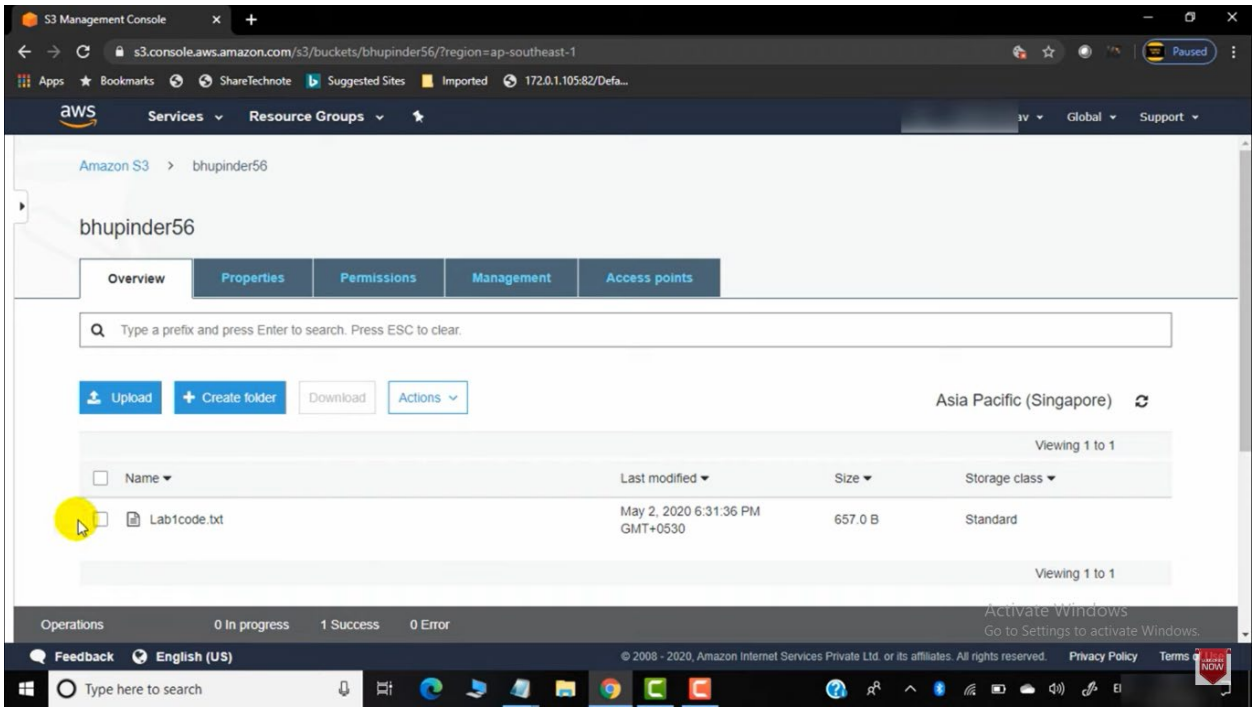


Now in S3 upload some data

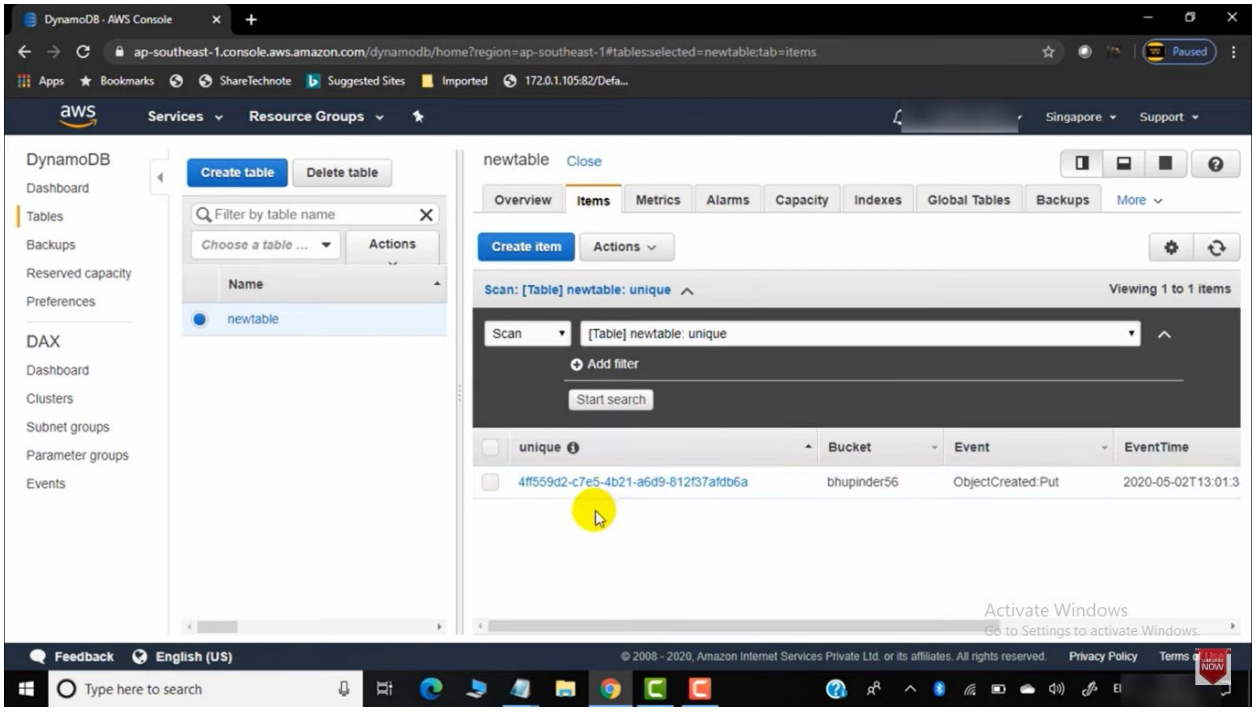



S3 Management Console
s3.console.aws.amazon.com/s3/buckets/bhupinder56/?region=ap-southeast-1
Services
Resource Groups
Global
Support
Amazon S3 > bhupinder56
bhupinder56
Overview
Properties
Permissions
Management
Access points
Upload
Create folder
Download
Actions
Asia Pacific (Singapore)
This bucket is empty. Upload new objects to get started.
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms
Type here to search


S3 Management Console
s3.console.aws.amazon.com/s3/buckets/bhupinder56/?region=ap-southeast-1
Upload
Select files
Set permissions
Set properties
Review
1 Files Size: 657.0 B Target path: bhupinder56
Manage users
User ID
Objects
Object permissions
barkha.official.1997(Owner)
Read
Read Write
Access for other AWS account
Add account
Account
Objects
Object permissions
Manage public permissions
Upload
Previous
Next
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms
Type here to search



Now go to DynamoDB, here we can see all the details of file uploaded to S3 bucket



What is Amazon S3?

S3 stands for simple storage service; it is used for storing data in the form of objects in the AWS Cloud.

- Amazon Simple Storage Service (S3) is storage for the internet.
- It is designed for large-capacity, low-cost storage provision across multiple geographical regions.
- Amazon S3 provides developers and IT teams with Secure, Durable and Highly Scalable object storage.
- S3 is a safe place to store the files.
- It is Object-based storage, i.e., you can store the images, word files, pdf files, etc.
- The files which are stored in S3 can be from 0 Bytes to 5 TB.
- It has unlimited storage which means that you can store the data as much as you want.
- Files are stored in Bucket. A bucket is like a folder available in S3 that stores the files.
- S3 is a universal namespace, i.e., the names must be unique globally. Bucket contains a DNS address.
- Therefore, the bucket must contain a unique name to generate a unique DNS address.

Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements. Features of Amazon S3 Storage classes Amazon S3 offers a range of storage classes designed for different use cases. For example, you can store mission-critical production data in S3 Standard for frequent access, save costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA, and archive data at the lowest costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive. You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers optimized for frequent and infrequent access, and two opt-in archive access tiers designed for asynchronous access for rarely accessed data. For more information, see Using Amazon S3 storage classes (p. 706). For more information about S3 Glacier Flexible Retrieval, see the Amazon S3 Glacier Developer Guide. Storage management Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.

• S3 Lifecycle – Configure a lifecycle policy to manage your objects and store them cost effectively throughout their lifecycle. You can transition objects to other S3 storage classes or expire objects that reach the end of their lifetimes.

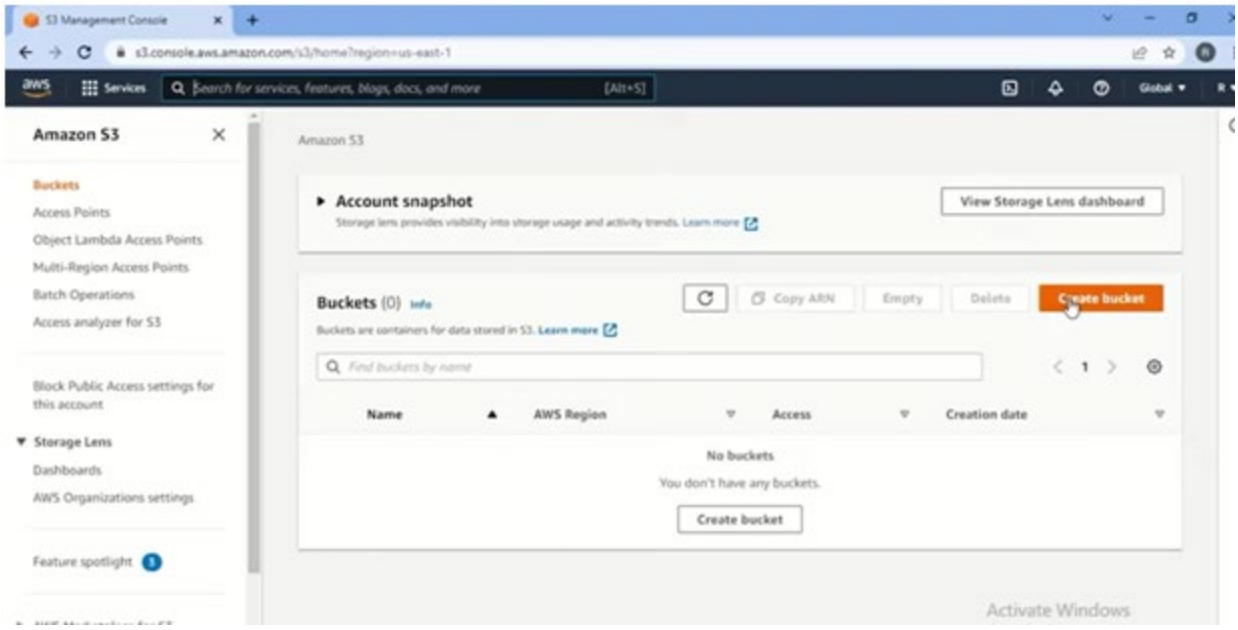
• S3 Object Lock – Prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use Object Lock to help meet regulatory requirements that require write once-read-many (WORM) storage or to simply add another layer of protection against object changes and deletions.

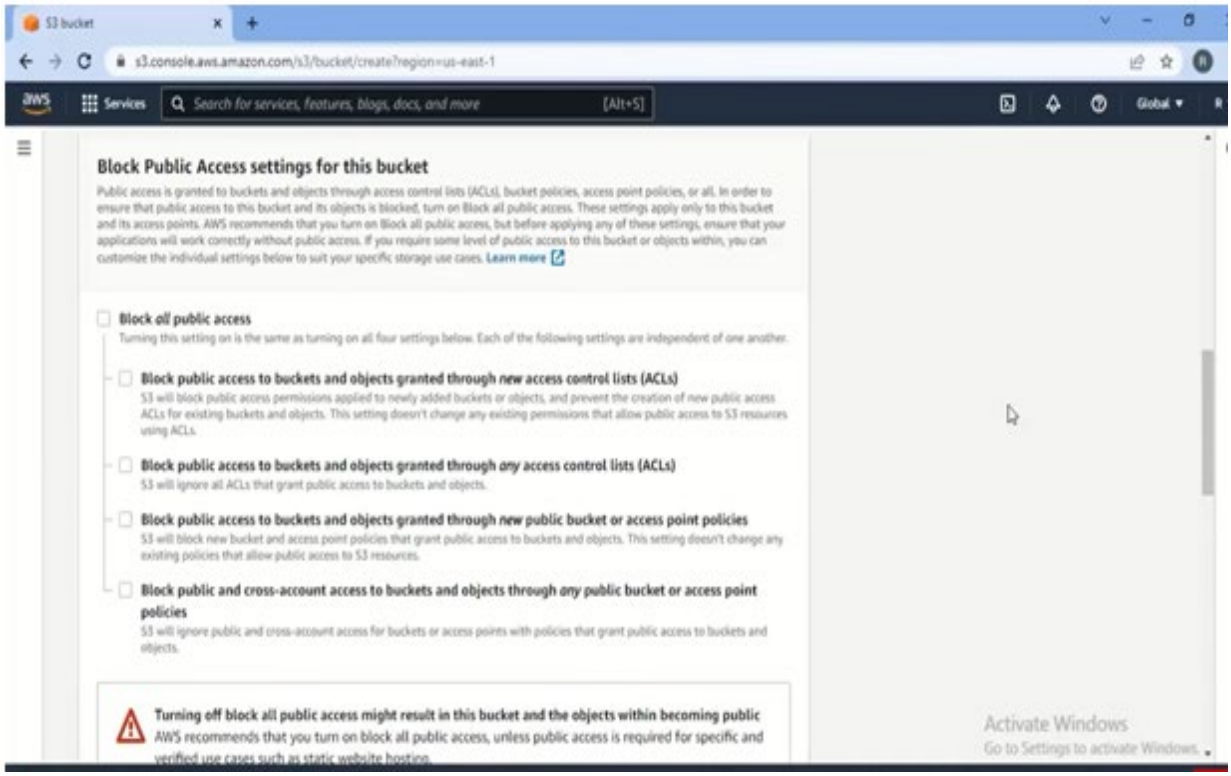
• S3 Replication – Replicate objects and their respective metadata and object tags to one or more destination buckets in the same or different AWS Regions for reduced latency, compliance, security, and other use cases.

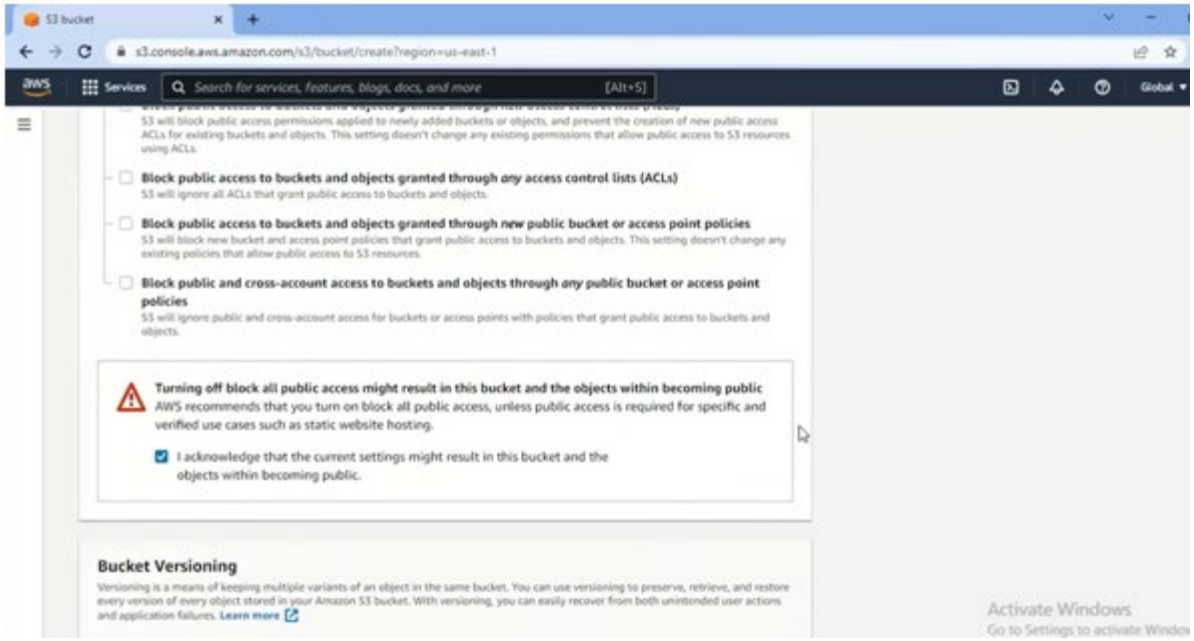
• S3 Batch Operations – Manage billions of objects at scale with a single S3 API request or a few clicks in the Amazon S3 console. You can use Batch Operations to perform operations such as Copy, Invoke AWS Lambda function, and Restore on millions or billions of objects.

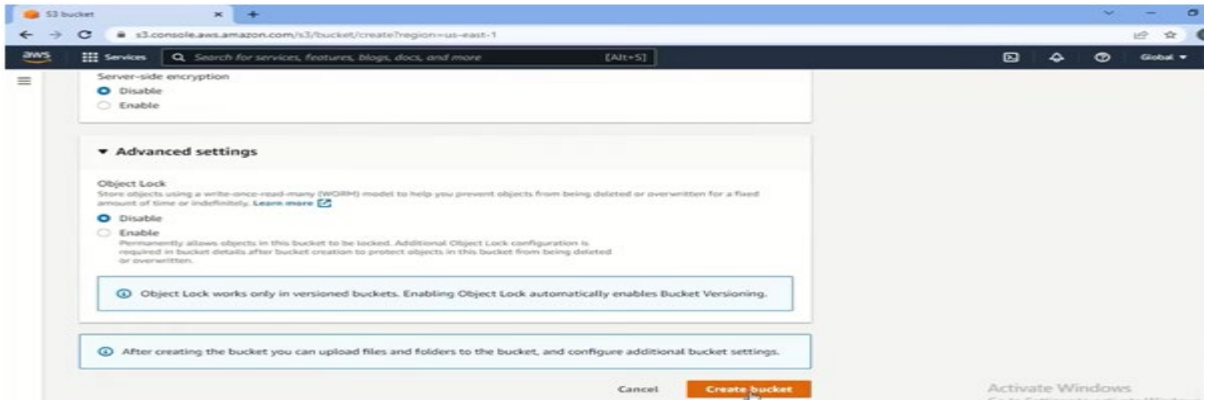
Hosting static website on S3

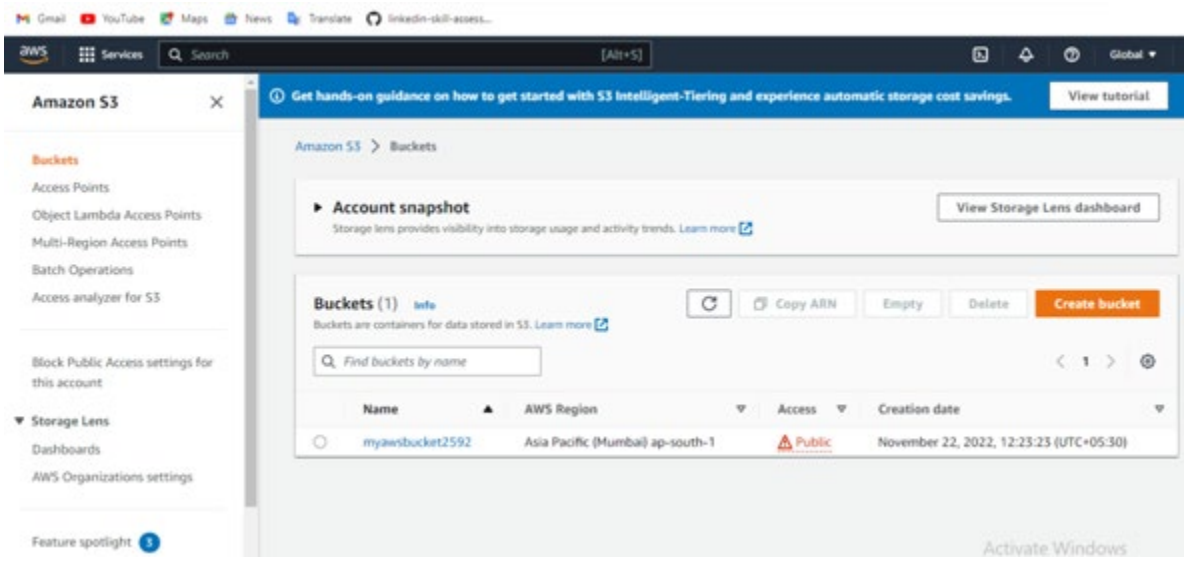
In AWS S3 create a bucket named myawsbucket2592

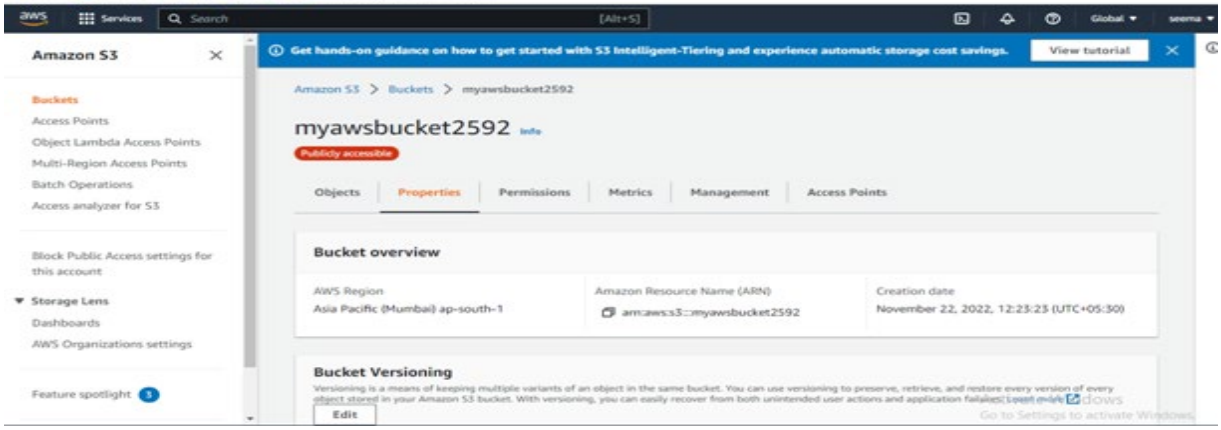


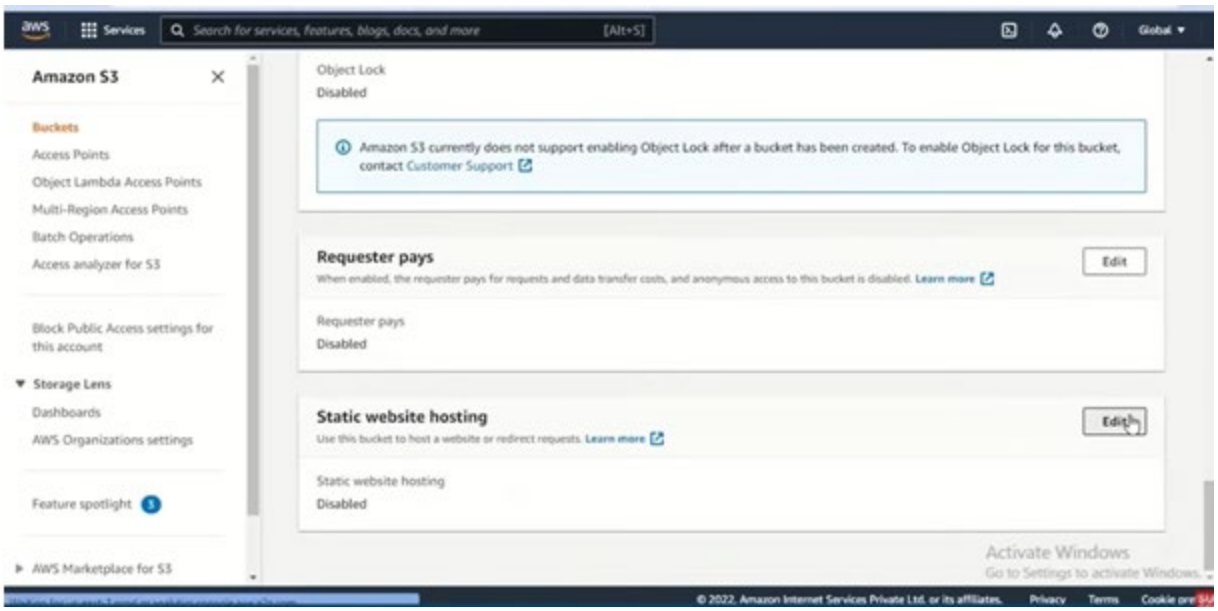

S3 bucket
s3.console.aws.amazon.com/s3/bucket/create?region=us-east-1
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.
Activate Windows
Go to Settings to activate Windows.

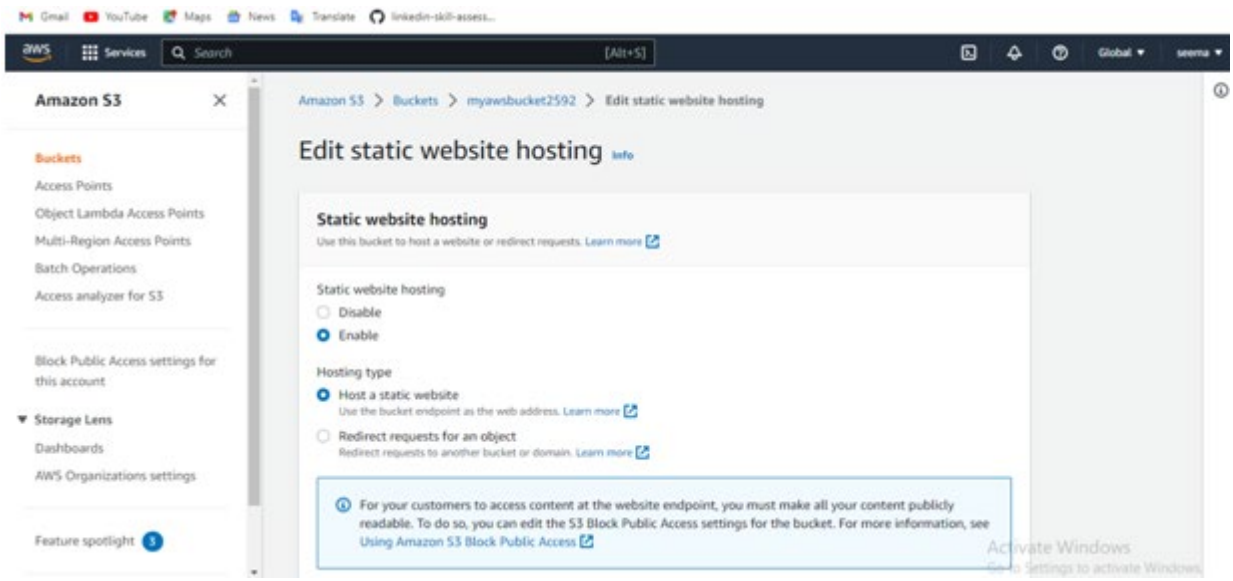

S3 bucket
s3.console.aws.amazon.com/s3/bucket/create?region=us-east-1
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.
I acknowledge that the current settings might result in this bucket and the objects within becoming public.
Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more
Activate Windows

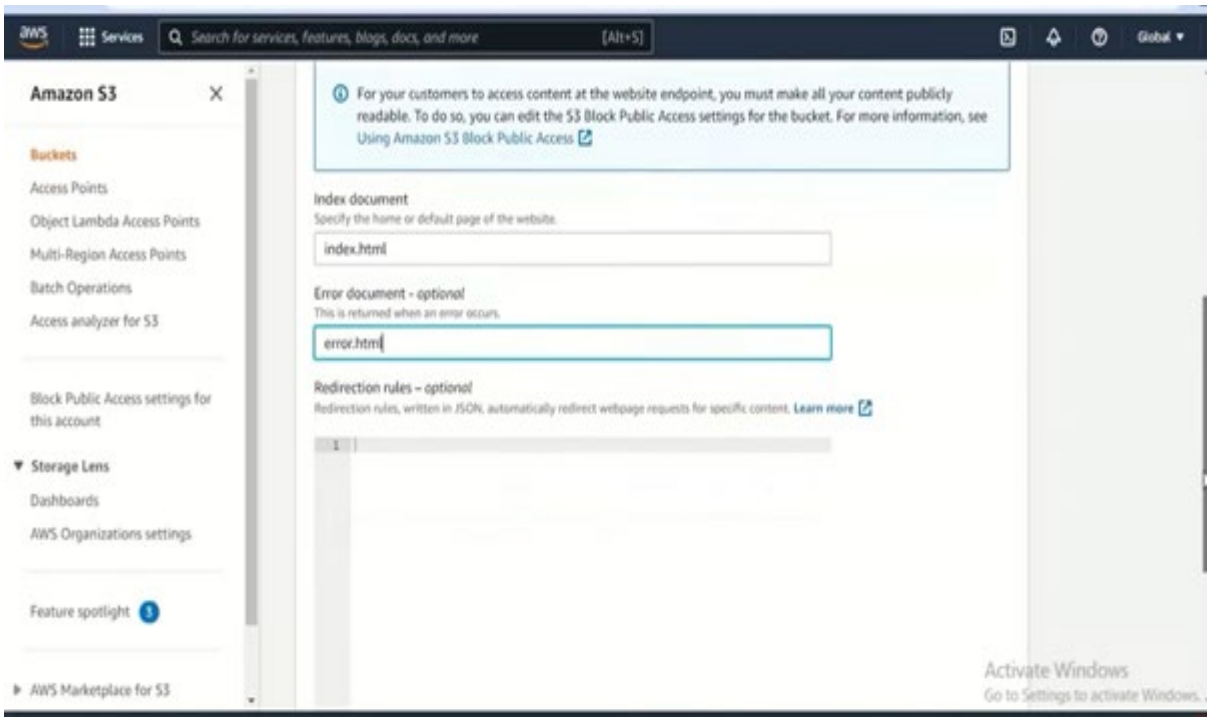

Server-side encryption
Disable
Enable
Advanced settings
Object Lock
Disable
Enable
Object Lock works only in versioned buckets. Enabling Object Lock automatically enables Bucket Versioning.
After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket

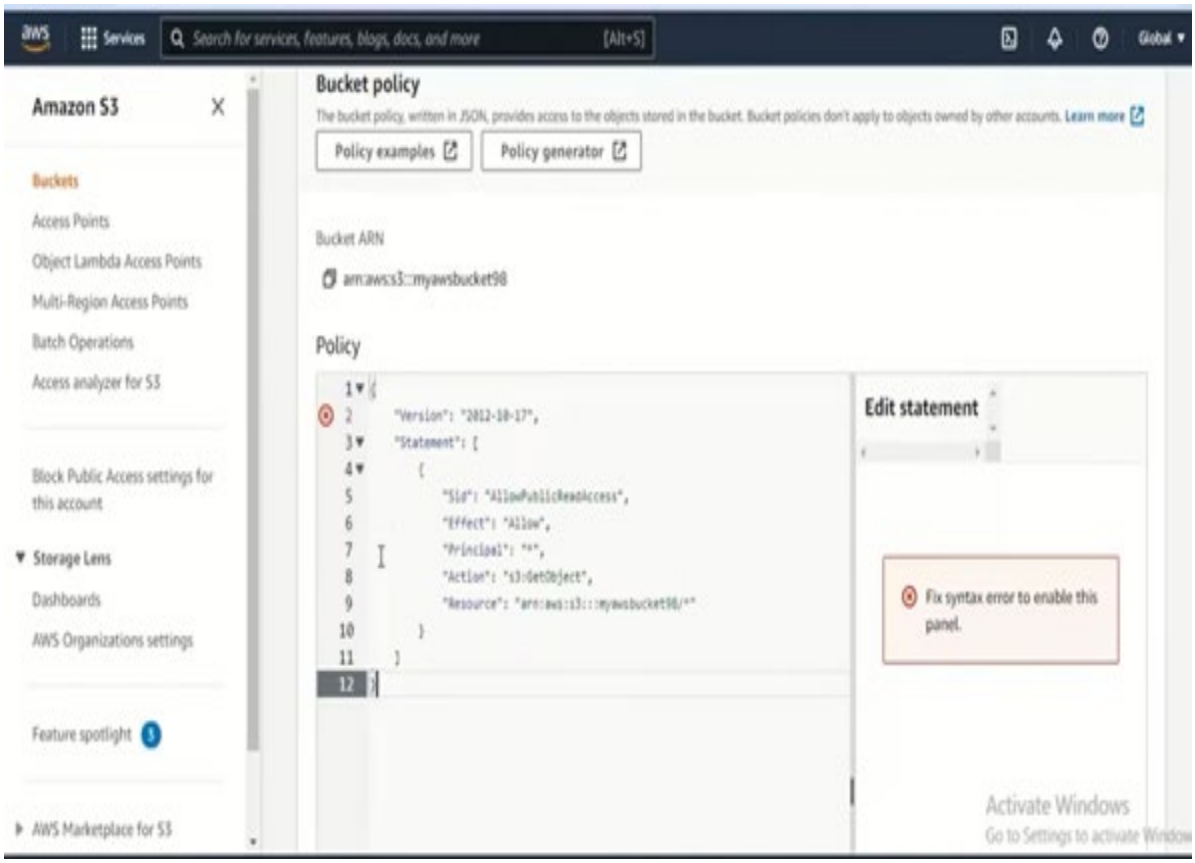

Amazon S3
Get hands-on guidance on how to get started with S3 Intelligent-Tiering and experience automatic storage cost savings.
View tutorial
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
Amazon S3 > Buckets
Account snapshot
View Storage Lens dashboard
Buckets (1)
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
Access
Creation date
myawsbucket2592
Asia Pacific (Mumbai) ap-south-1
Public
November 22, 2022, 12:23:23 (UTC+05:30)

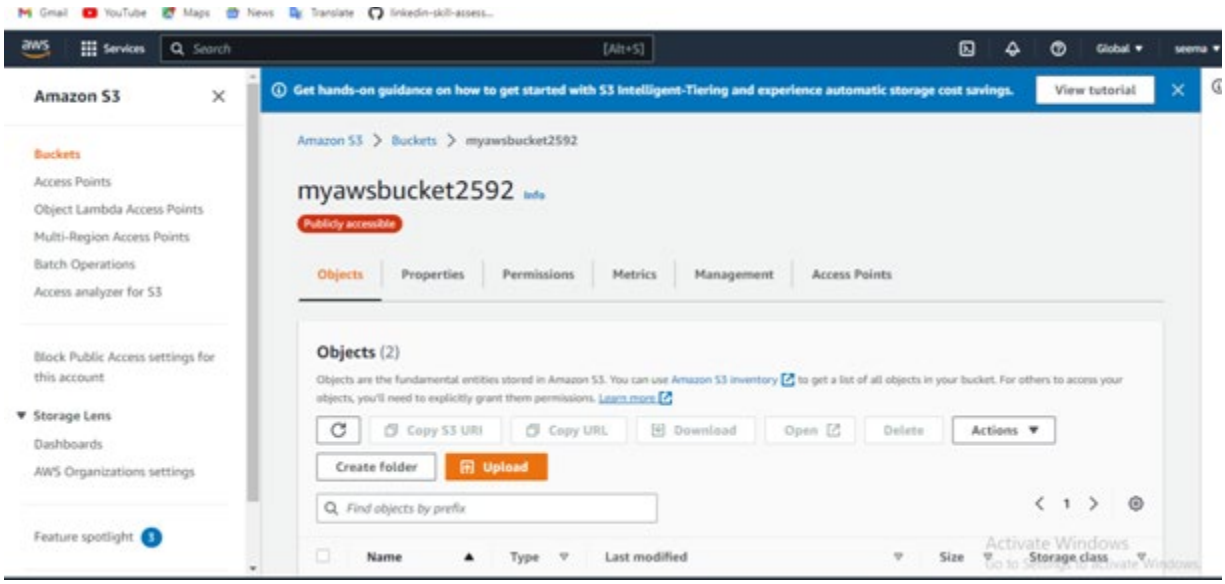

Amazon S3
Get hands-on guidance on how to get started with S3 Intelligent-Tiering and experience automatic storage cost savings.
View tutorial
Amazon S3 > Buckets > myawsbucket2592
myawsbucket2592
Publicly accessible
Objects
Properties
Permissions
Metrics
Management
Access Points
Bucket overview
AWS Region
Asia Pacific (Mumbai) ap-south-1
Amazon Resource Name (ARN)
arn:aws:s3:::myawsbucket2592
Creation date
November 22, 2022, 12:23:23 (UTC+05:30)
Bucket Versioning
Edit

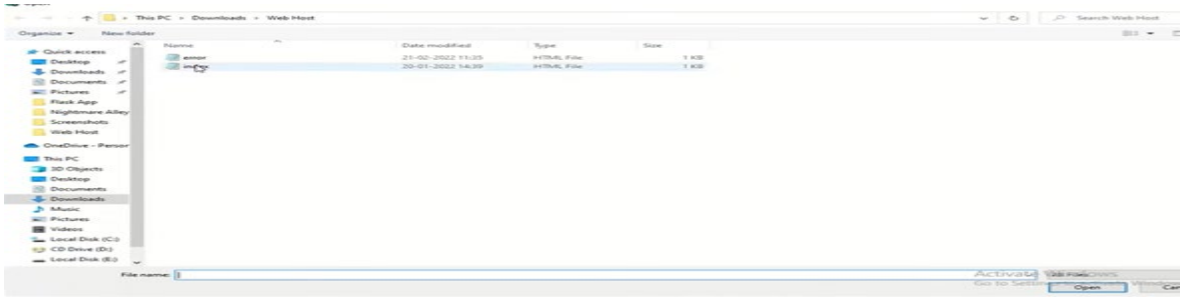

Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Object Lock
Disabled
Amazon S3 currently does not support enabling Object Lock after a bucket has been created. To enable Object Lock for this bucket, contact Customer Support
Requester pays
When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. Learn more
Edit
Requester pays
Disabled
Static website hosting
Use this bucket to host a website or redirect requests. Learn more
Edit
Static website hosting
Disabled
Activate Windows
Go to Settings to activate Windows.

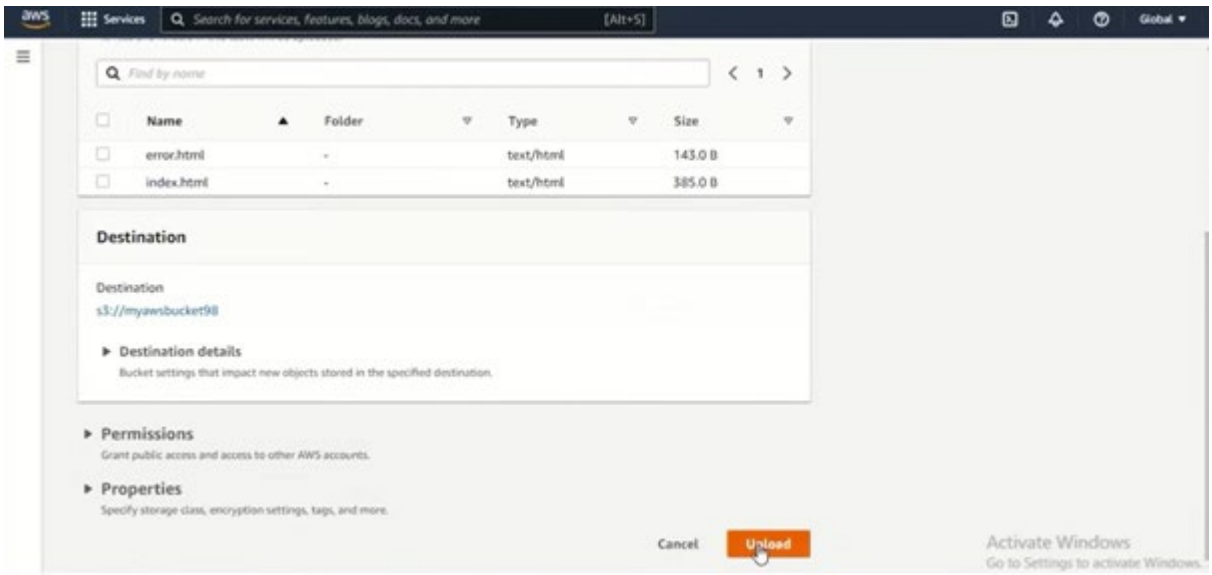

Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
Amazon S3 > Buckets > myawsbucket2592 > Edit static website hosting
Edit static website hosting Info
Static website hosting
Use this bucket to host a website or redirect requests. Learn more
Static website hosting
Disable
Enable
Hosting type
Host a static website
Use the bucket endpoint as the web address. Learn more
Redirect requests for an object
Redirect requests to another bucket or domain. Learn more
For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access
Activate Windows

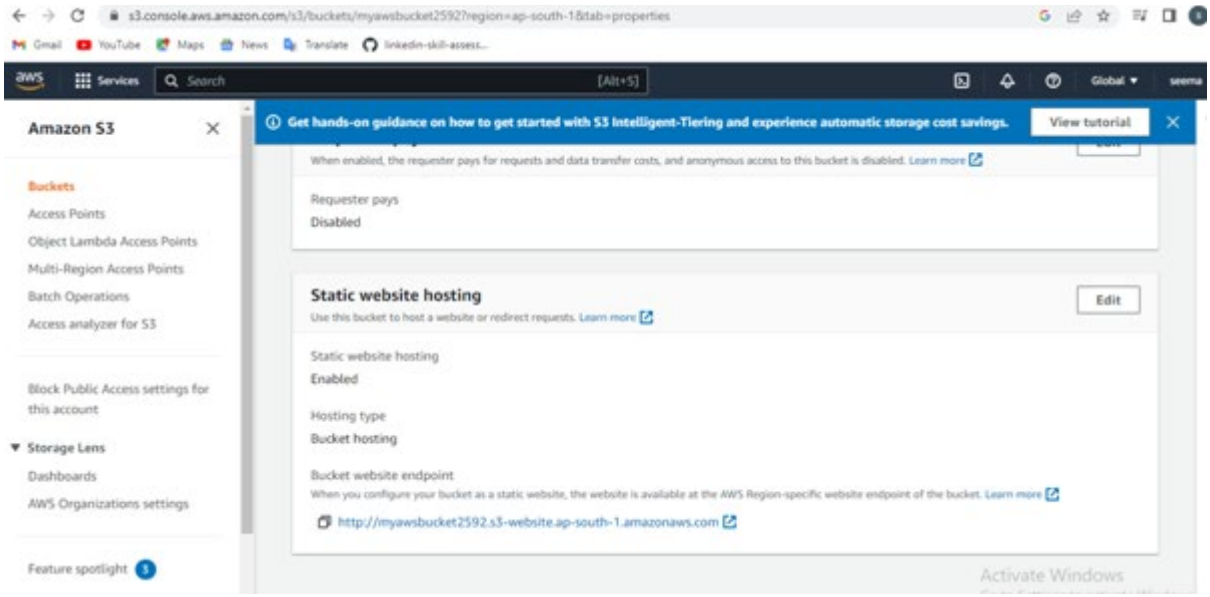

Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access
Index document
Specify the home or default page of the website.
index.html
Error document - optional
This is returned when an error occurs.
error.html
Redirection rules - optional
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. Learn more
Activate Windows
Go to Settings to activate Windows.

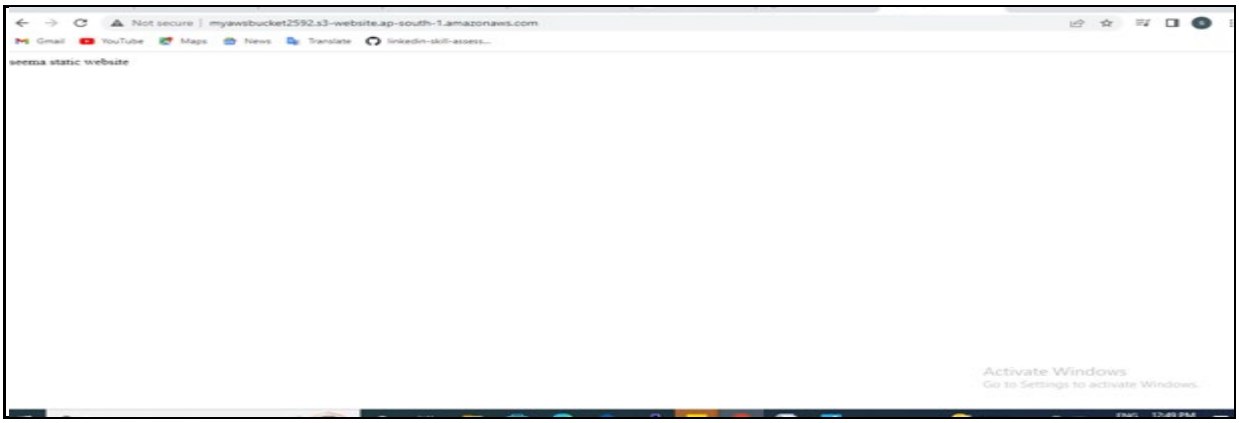

Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Bucket policy
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more
Policy examples
Policy generator
Bucket ARN
arn:aws:s3:::myawsbucket98
Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "AllowPublicReadAccess",
"Effect": "Allow",
"Principal": "*",
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::myawsbucket98/*"
}
]
}
Edit statement
Fix syntax error to enable this panel.
Activate Windows
Go to Settings to activate Windows


Amazon S3
Get hands-on guidance on how to get started with S3 Intelligent-Tiering and experience automatic storage cost savings.
View tutorial
Amazon S3 > Buckets > myawsbucket2592
myawsbucket2592
Publicly accessible
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (2)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class



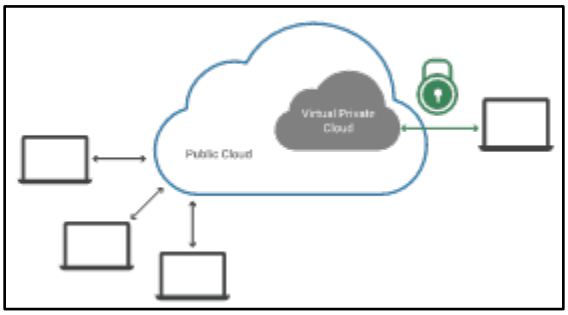

Find by name
Name
Folder
Type
Size
error.html
text/html
143.0 B
index.html
text/html
385.0 B
Destination
Destination
s3://myawsbucket98
Destination details
Bucket settings that impact new objects stored in the specified destination.
Permissions
Grant public access and access to other AWS accounts.
Properties
Specify storage class, encryption settings, tags, and more.
Cancel
Upload


s3.console.aws.amazon.com/s3/buckets/myawsbucket2592?region=ap-south-1&tab=properties
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
Get hands-on guidance on how to get started with S3 Intelligent-Tiering and experience automatic storage cost savings.
View tutorial
Requester pays
Disabled
Static website hosting
Use this bucket to host a website or redirect requests. Learn more
Edit
Static website hosting
Enabled
Hosting type
Bucket hosting
Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://myawsbucket2592.s3-website.ap-south-1.amazonaws.com
Activate Windows


Not secure | myawsbucket2592.s3-website.ap-south-1.amazonaws.com
seema static website
Activate Windows


Not secure | myawsbucket2592.s3-website.ap-south-1.amazonaws.com/xyz
error
Activate Windows

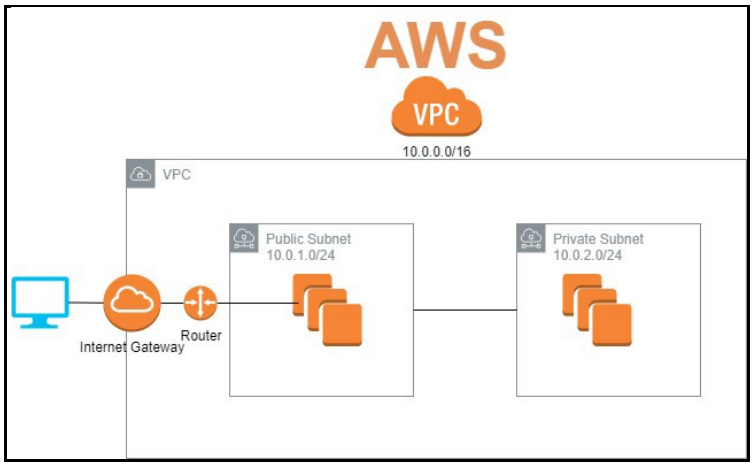
Virtual private cloud (VPC)



A virtual private cloud (VPC) is a secure, isolated private cloud hosted within a public cloud. VPC customers can run code, store data, host websites, and do anything else they could do in an ordinary private cloud, but the private cloud is hosted remotely by a public cloud provider. (Not all private clouds are hosted in this fashion.) VPCs combine the scalability and convenience of public cloud computing with the data isolation of private cloud computing.

Imagine a public cloud as a crowded restaurant and a virtual private cloud as a reserved table in that crowded restaurant. Even though the restaurant is full of people, a table with a "Reserved" sign on it can only be accessed by the party who made the reservation. Similarly, a public cloud is crowded with various cloud customers accessing computing resources – but a VPC reserves some of those resources for use by only one customer.

- It is logically isolated with other Virtual Network in your AWS cloud
- Maximum 5 VPC can be create & 200 subnets in one VPC
- We can attach max 5 elastic IP
- Once we created VPC, DHCP, NACL, security group will be created automatically
- A VPC is configured o an AWS region & does not extend between regions
- Once VPC is create you cannot change its CIDR, block range
- If you need a different CIDR size, create a new VPC
- Different subnets within a VPC cannot overlap
- You can however expand your VPC CIDR by adding new/extra IP address ranges (except GovCloud & AWS China)



How is a VPC isolated within a public cloud?

A VPC isolates computing resources from the other computing resources available in the public cloud. The key technologies for isolating a VPC from the rest of the public cloud are:

Subnets: A subnet is a range of IP addresses within a network that are reserved so that they're not available to everyone within the network, essentially dividing part of the network for private use. In a VPC these are private IP addresses that are not accessible via the public Internet, unlike typical IP addresses, which are publicly visible.

VLAN: A LAN is a local area network, or a group of computing devices that are all connected to each other without the use of the Internet. A VLAN is a virtual LAN. Like a subnet, a VLAN is a way of partitioning a network, but the partitioning takes place at a different layer within the OSI model (layer 2 instead of layer 3).

VPN: A virtual private network (VPN) uses encryption to create a private network over the top of a public network. VPN traffic passes through publicly shared Internet infrastructure – routers, switches, etc. – but the traffic is scrambled and not visible to anyone.

A VPC will have a dedicated subnet and VLAN that are only accessible by the VPC customer. This prevents anyone else within the public cloud from accessing computing resources within the VPC – effectively placing the "Reserved" sign on the table. The VPC customer connects via VPN to their VPC, so that data passing into and out of the VPC is not visible to other public cloud users.

Some VPC providers offer additional customization with:

- Network Address Translation (NAT): This feature matches private IP addresses to a public IP address for connections with the public Internet. With NAT, a public-facing website or application could run in a VPC.

- BGP route configuration: Some providers allow customers to customize BGP routing tables for connecting their VPC with their other infrastructure. (Learn how BGP works.)

Types of VPC

1. **Default VPC**
    a. Created in each AWS region when AWS account is created
    b. It has default CIDR, security group, NACL & route table settings
    c. It has internet gateway by default
2. **Custom VPC**
    a. It is VPC created by AWS account owner
    b. AWS user creating the custom VPC can decide CIDR
    c. Has its default security group, network ACL & route table
    d. Does not have internet gateway by default, needs to be created if needed

- **Public Subnet**
    - If a subnet traffic is routed to internet gateway then it is called public subnet
    - If you want your instance in a public subnet to communicate with the internet over IPv4, it must have public IPv4 address or an Elastic IP address
- **Private subnet**
    - If a subnet does not routed to internet gateway then it is called private subnet

- When you create a VPC, you must specify IPv4 CIDR block, for VPC, the allowed block size is /16 to /28 netmask
- First four & last IP address of subnet cannot be assigned
- For e.g. Our network is 10.0.0/16
    - 10.0.0.0 → network address
    - 10.0.0.1 → Reserved by AWS for VPC router
    - 10.0.0.2 → Reserved by AWS. IP address of DNS server

- 10.0.0.3 → Reserved for future use
- 10.0.0.255 → broadcast address

Implied Router/ Logical Router

- It is the central routing function
- It connects different AZ together & connects the VPC to the internet gateway
- You can have up to 200 route tables per VPC
- You can have up to 50 route entries per route table
- Each subnet must be associated with only route table at any given time
- If you do not specify a subnet to route table association, the subnet will be associated with the default VPC route table
- You can also edit the main route table if you need, but you cannot delete main route table
- However you can make custom route table manually become the main route table then you can delete the former main as it is no longer a main route table
- You can associate multiple subnets with the same route table

Internet Gateway:

- An internet gateway is a virtual router that connects a VPC to the internet.
- Default VPC is already attached with an internet gateway.
- If you create a new VPC then you must attach the internet gateway in order to access the internet.
- Ensure that your subnet's route table points to the internet gateway.
- It performs NAT between your private and public IPV4 address.
- It supports both IPV4 and IPV6.

NAT Gateway:

- You can use a network address translation gateway to enable instances in a private subnet to connect to the internet or other AWS services but prevent the internet from initiating a connection with those instances.

- You are charged for creating and using a NAT gateway in your account. NAT gateway hourly usage and data processing rates apply. Amazon EC2 charges for data transfer also apply.
- To create a NAT gateway, you must specify the public subnet in which the NAT gateway should reside.
- You must also specify an elastic IP address to associate with NAT gateway when you create it.
- No need to assign public IP address to your private instance.
- After you have created a NAT gateway you must update the route table associated with One or more of your private subnets to point internet bound traffic to the NAT gateway. This enables instances in your private subnet to communicate with the internet.
- Deleting a NAT gateway, disassociates its elastic IP address, but does not release the address from your account.

Security Group:

- It is a virtual firewall works at ENI level.
- Up to 5 security groups per EC2 instance interface can be applied.
- Can only have permit rules, cannot have deny rules.
- Stateful, return traffic of allowed inbound traffic is allowed even if there are no rules to allow it.

NACL:

- It is a function performed on the implied router.
- NACL is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets.
- Your VPC automatically comes with a modifiable default NACL. By default, it allows all inbound and outbound IPV4 traffic and if applicable IPV6 traffic.
- You can create a custom NACL and associate it with a subnet. By default each NACL denies all inbound and outbound traffic until you add rules.
- Each subnet in your VPC must be associated with a NACL. If you don't explicitly associate a subnet with a NACL, the Subnet is automatically associated with the default NACL.

- You can associate a NACL with multiple subnets; however a subnet can be associated with only one NACL at a time. When you associate a NACL with a subnet the previous association is removed.
- A NACL contains a numbered list of rules that we evaluate in order starting with the lowest numbered rule.
- The highest number that you can use for a rule is 32766. Recommended that you start by creating rules with rule numbers that a multiple of 100, so that you can insert new rules where you need later.
- It functions at the subnet level.
- NACL are stateless, outbound traffic for an allowed inbound traffic must be explicitly allowed too.
- You can have permit and deny rules in a NACL.

VPC Peering:

- A VPC peering connection is a network connection between two VPC that enables you to route traffic between them using private IPV4 addresses or IPV6 addresses.
- Instances in either VPC can communicate with each other as if they are within the same network.
- You can create a VPC peering connection between your own VPC or with a VPC in another AWS account. The VPC can be in different region.

VPC Endpoint

- A VPC endpoint enables you to privately connect your VPC to supported AWS services, instances in your VPC do not require public IP address to communicate with resources in the service. Endpoints are virtual devices.

Difference between security group and NACL.

Security Group	**NACL**
Operate at instance level.	Operates at the subnet level.
Supports allow rules only.	It permits allow as well as deny rules.
Stateful, return traffic is	Stateless, return traffic must be

automatically allowed	explicitly allowed by rules
Applies to an instance only.	Applies to all instances in the subnet.

What are the advantages of using a VPC instead of a private cloud?

Scalability: Because a VPC is hosted by a public cloud provider, customers can add more computing resources on demand.

Easy hybrid cloud deployment: It's relatively simple to connect a VPC to a public cloud or to on-premises infrastructure via the VPN. (Learn about hybrid clouds and their advantages.)

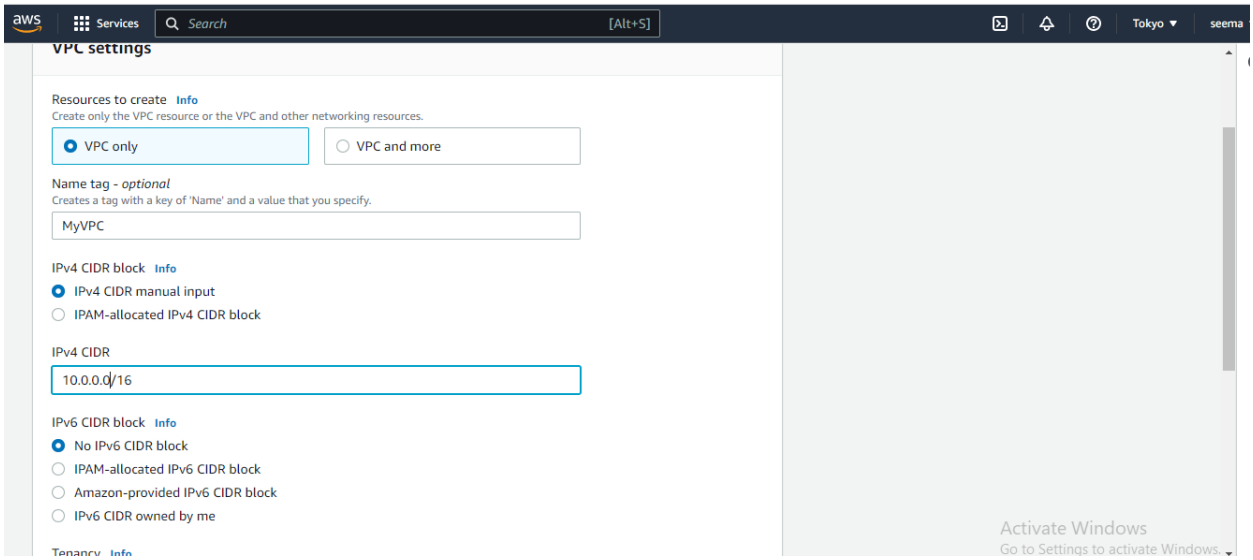
Better performance: Cloud-hosted websites and applications typically perform better than those hosted on local on-premises servers.

Better security: The public cloud providers that offer VPCs often have more resources for updating and maintaining the infrastructure, especially for small and mid-market businesses. For large enterprises or any companies that face extremely tight data security regulations, this is less of an advantage.

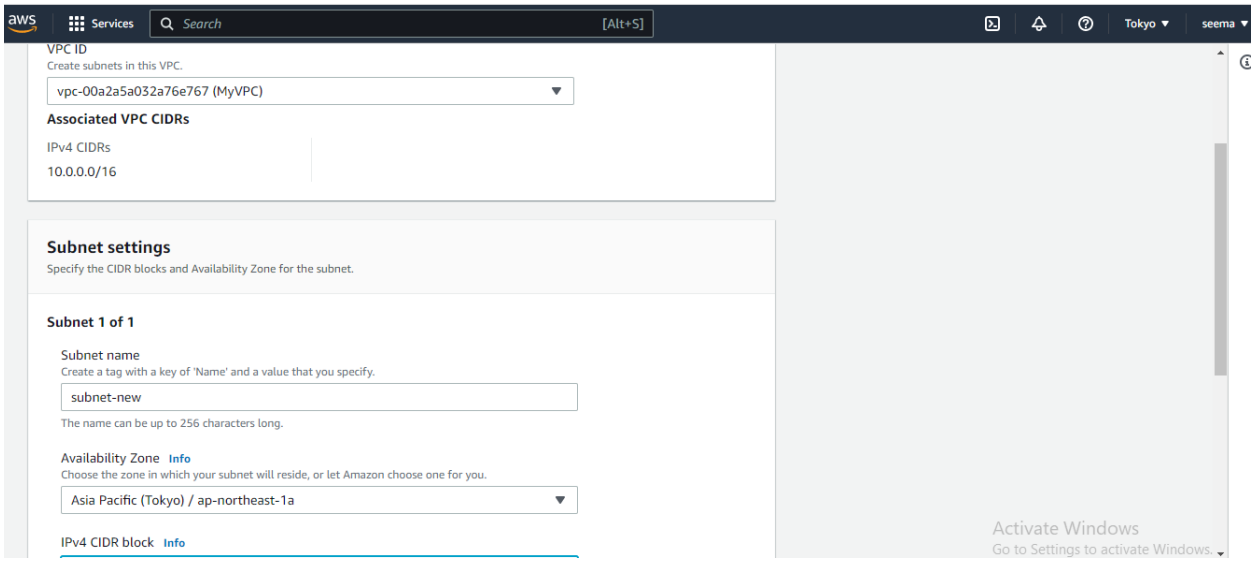
Steps to create VPC

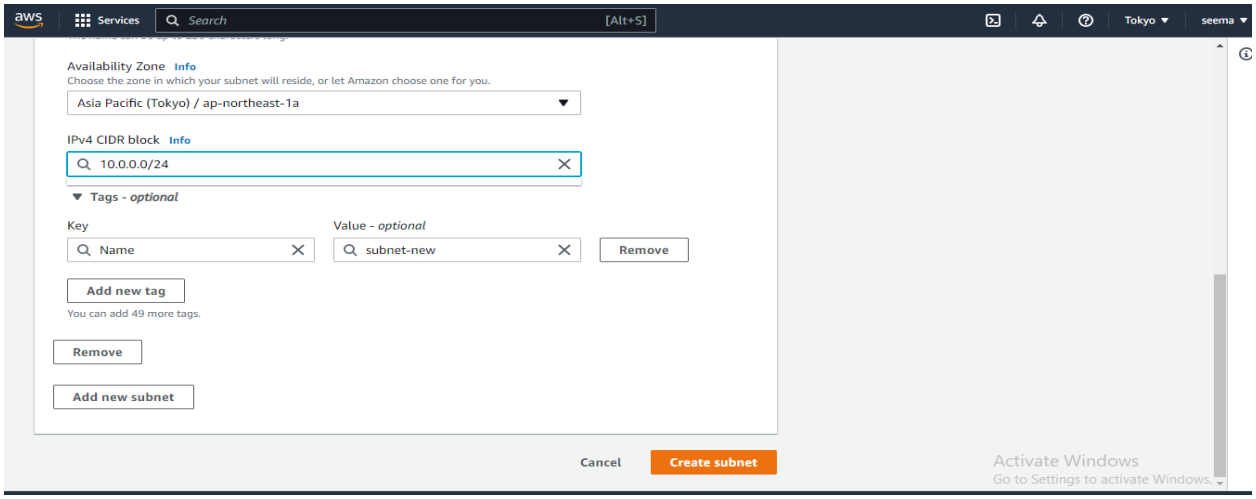
Create VPC → subnet → internet gateway → route table

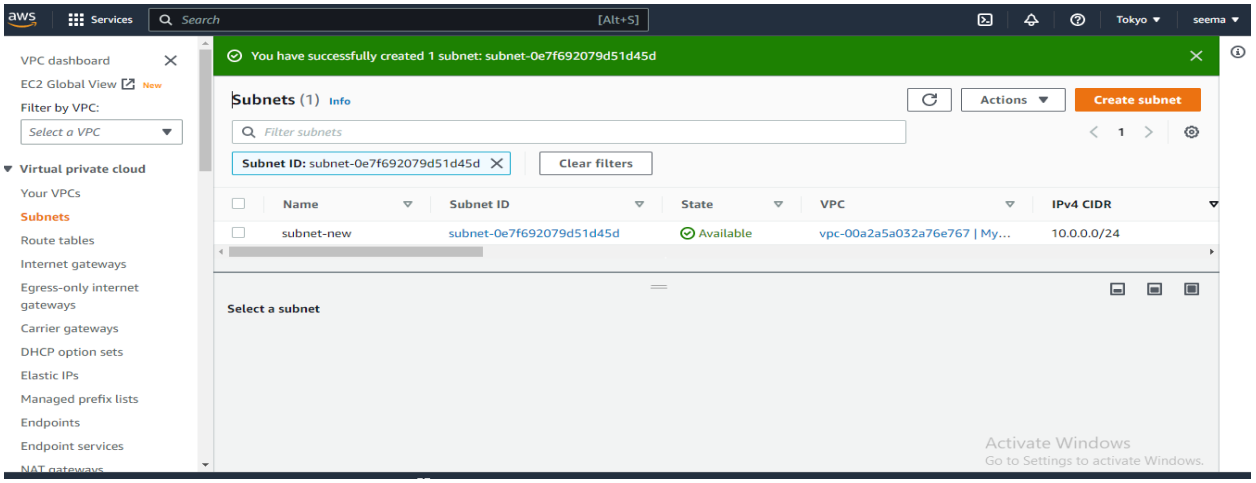
Create a VPC with name “MyVPC” & IPv4-CIDR As 10.0.0.0/16



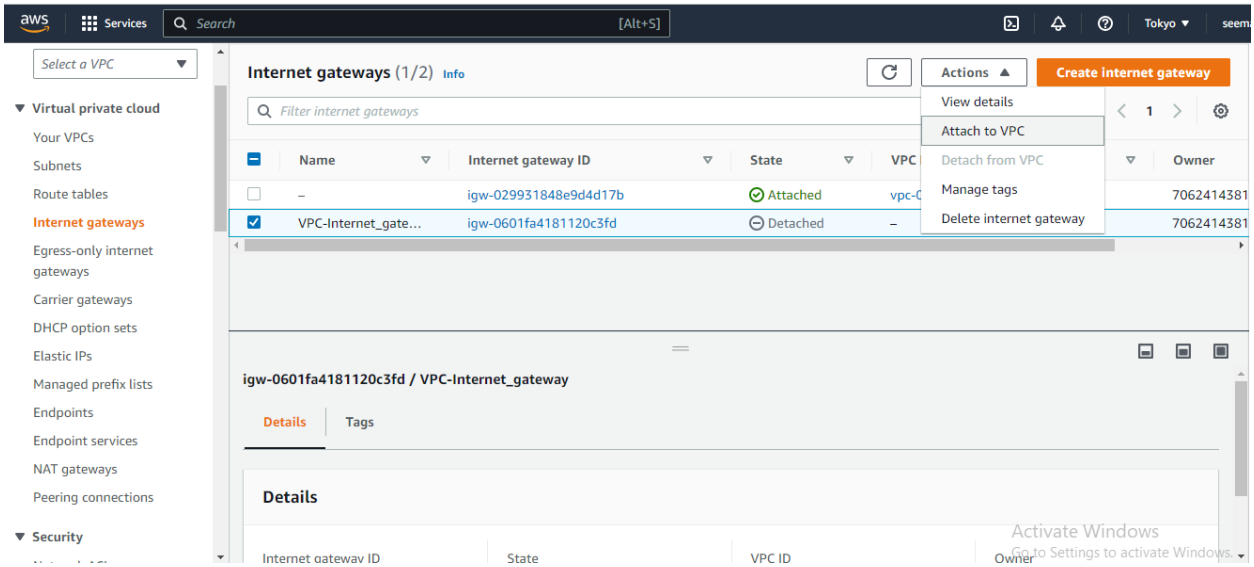
Create a subnet with name “subnet-new’’



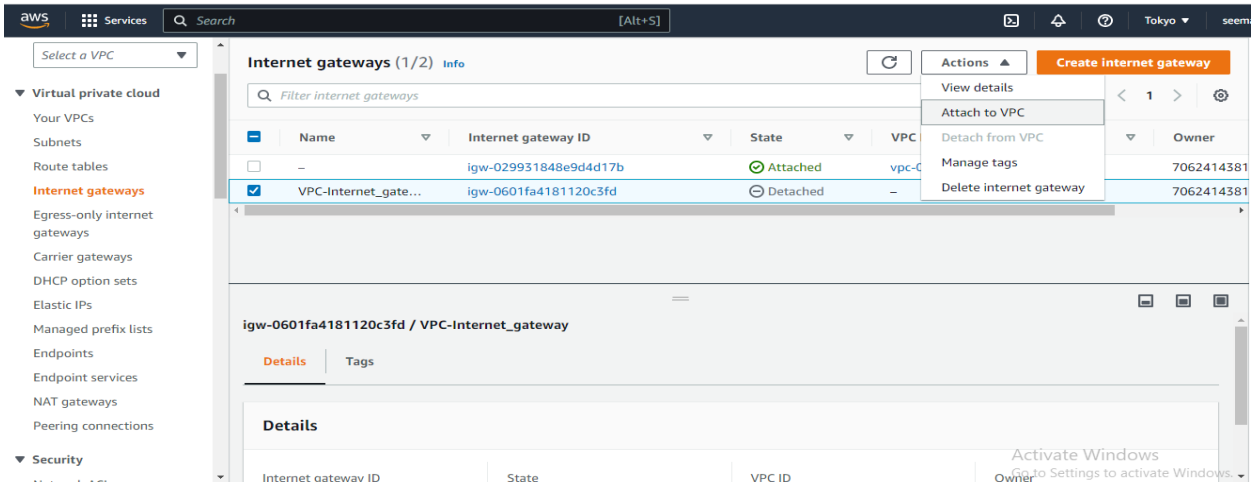


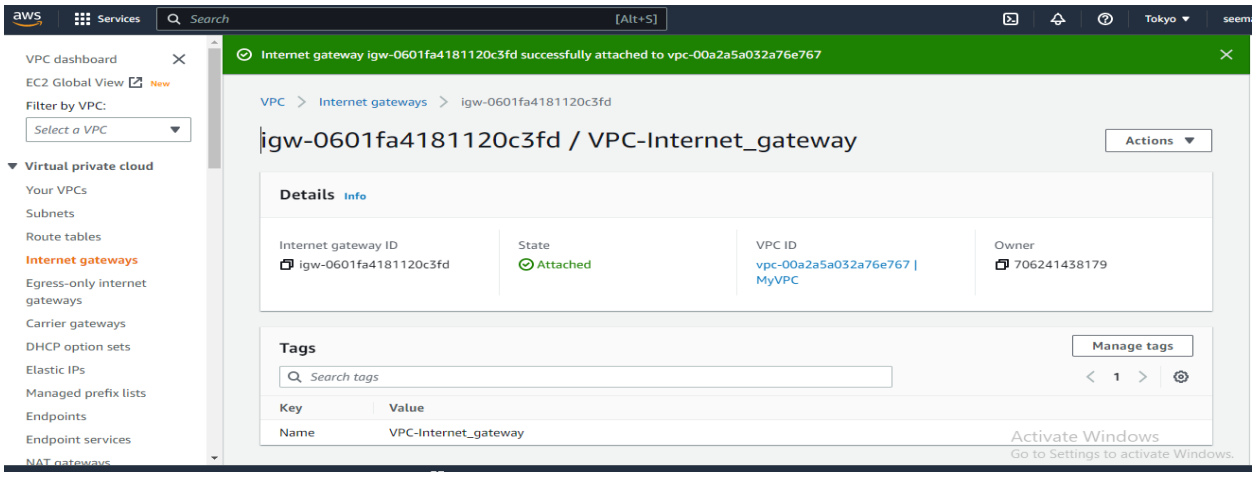


Create a internet gateway with name “VPC-internet_gateway”

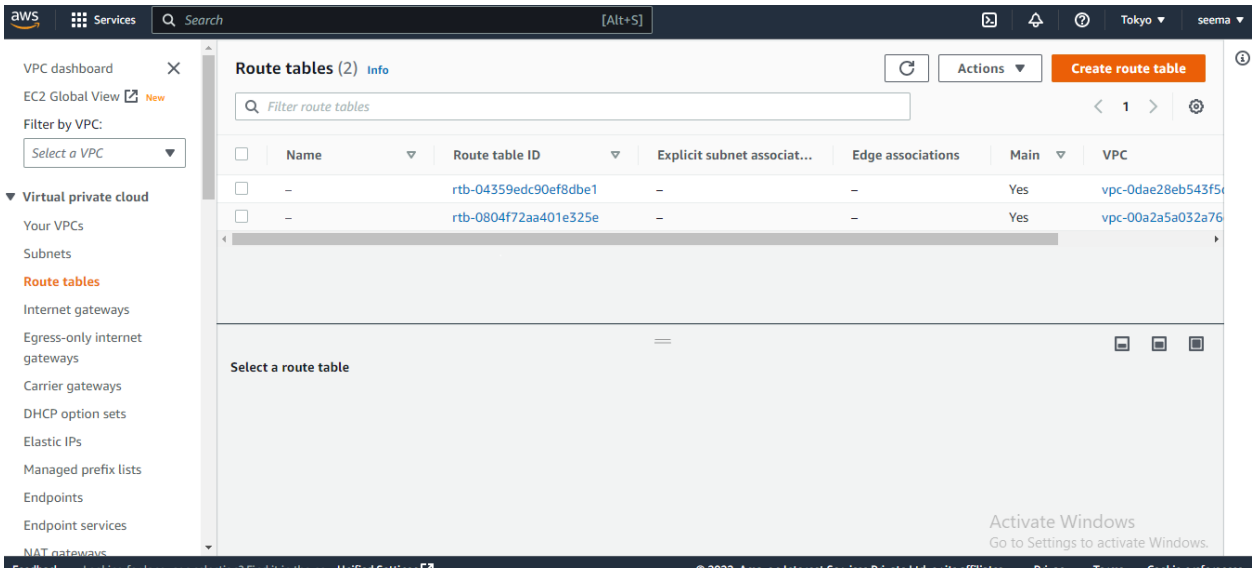


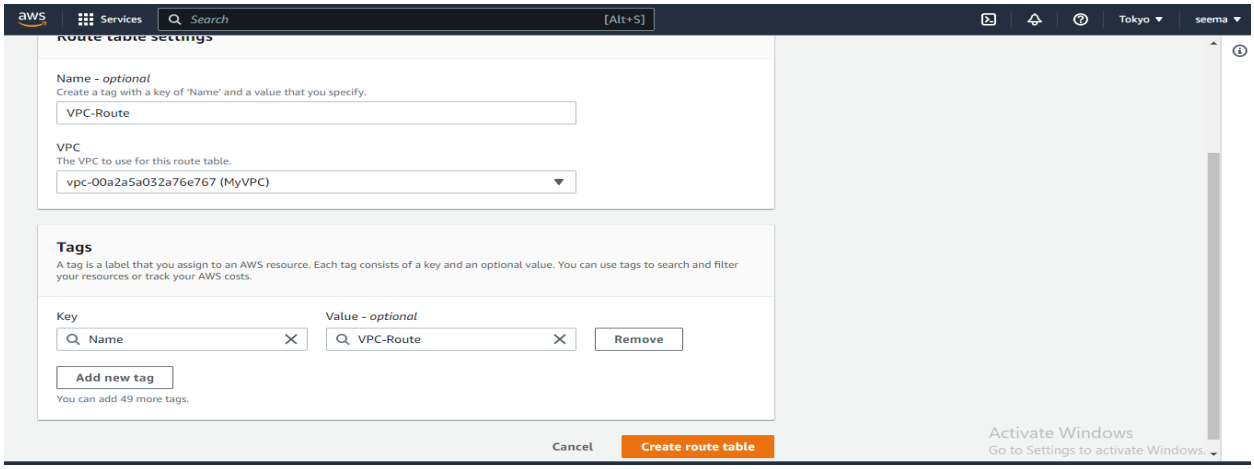
Now attach this internet gateway to our VPC

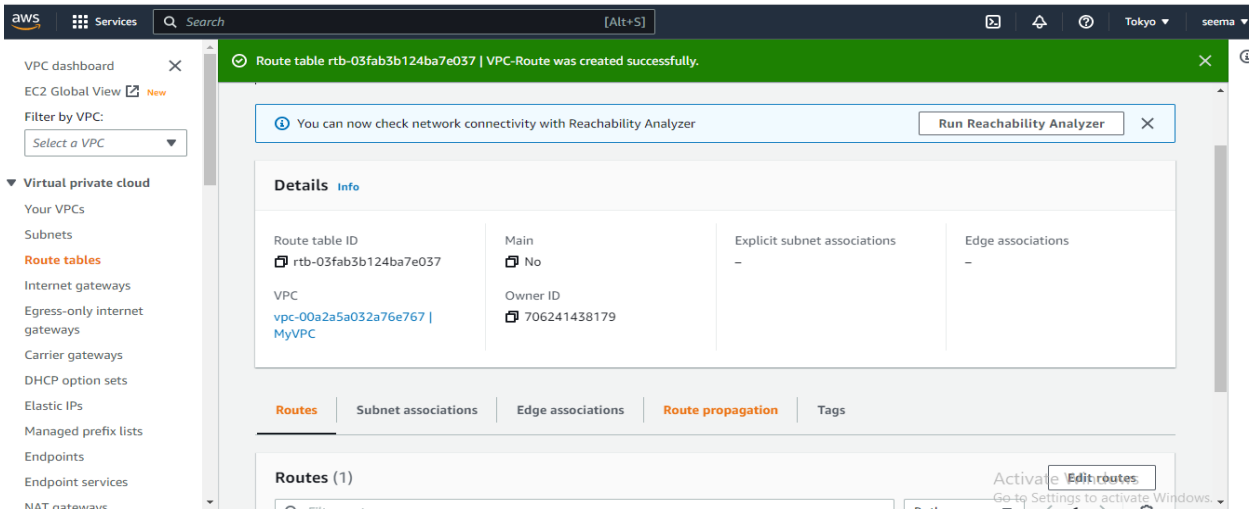


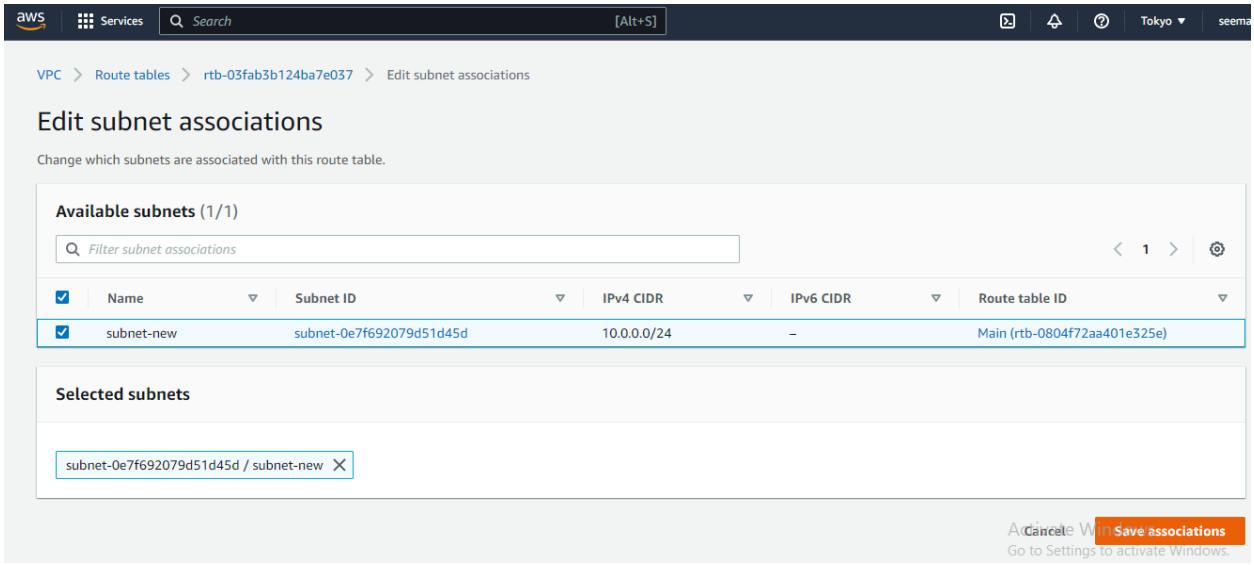


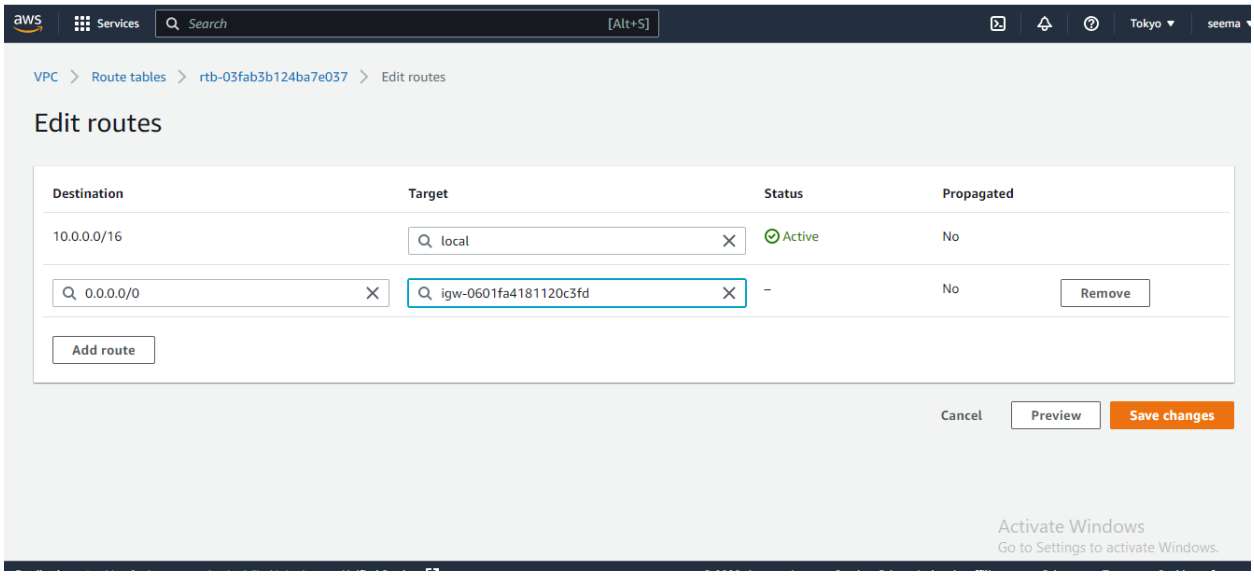
Now create a route table with name “VPC-Route”



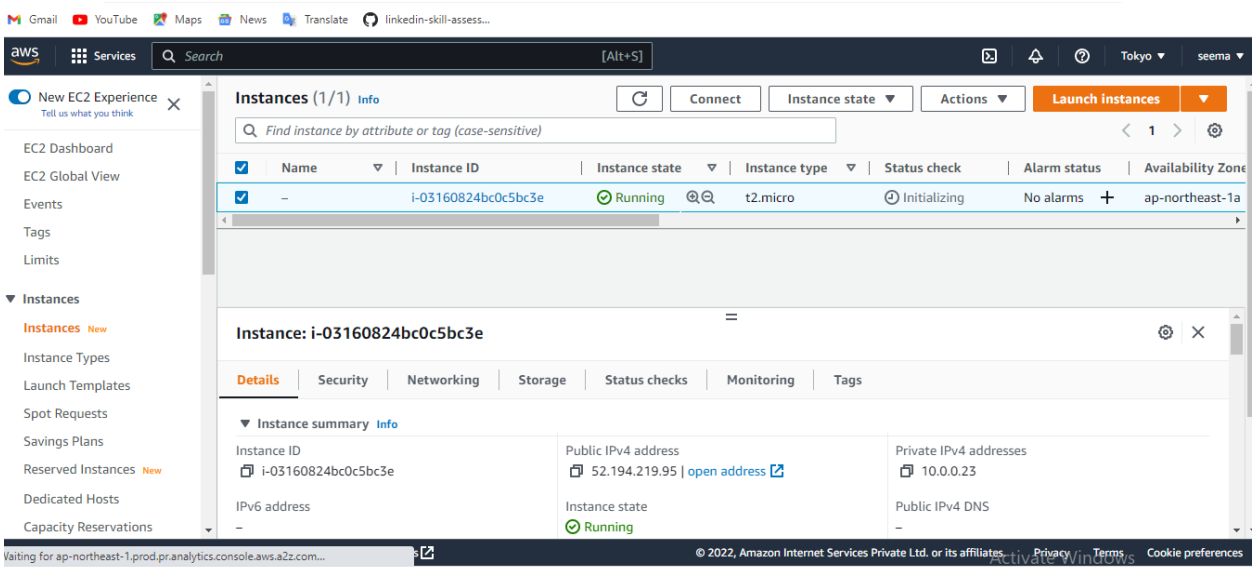




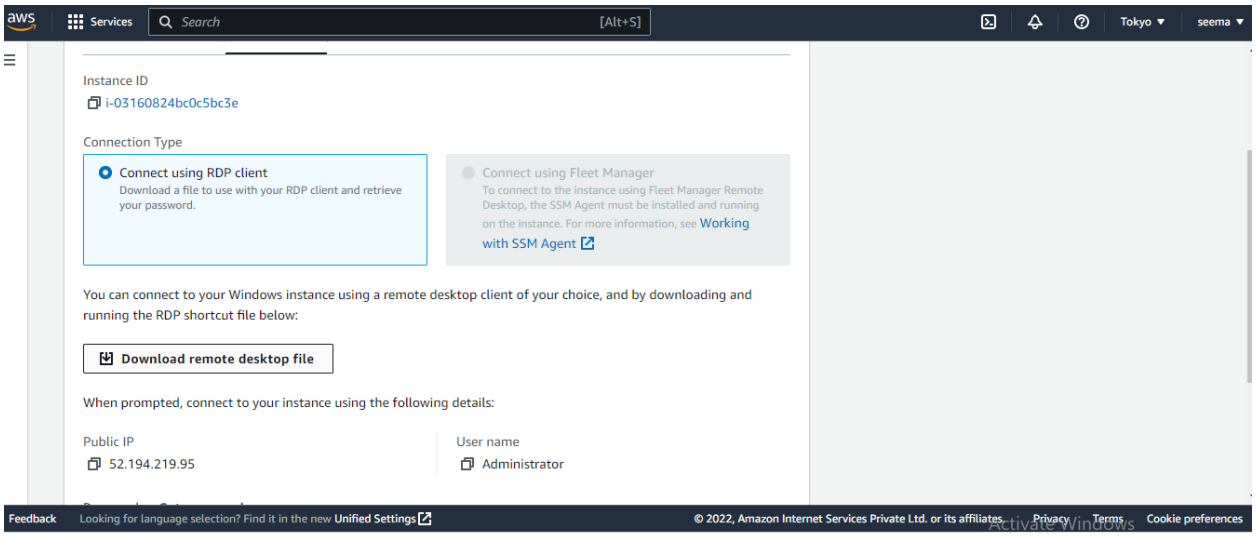


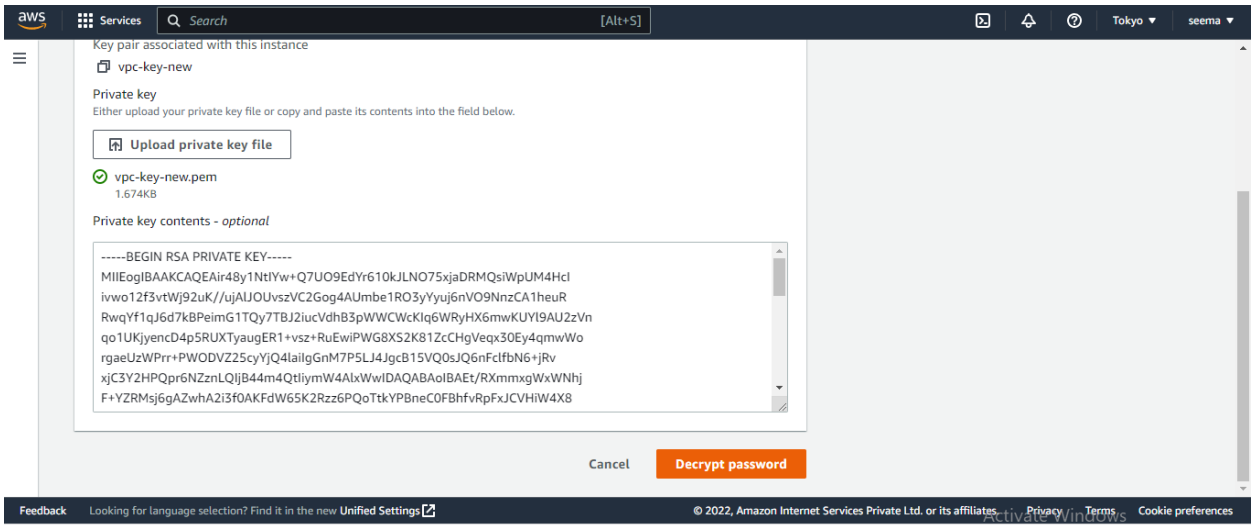


Now create a EC2 instance & select windows server 2012 R2 operating system. In network setting select a VPC that we have created

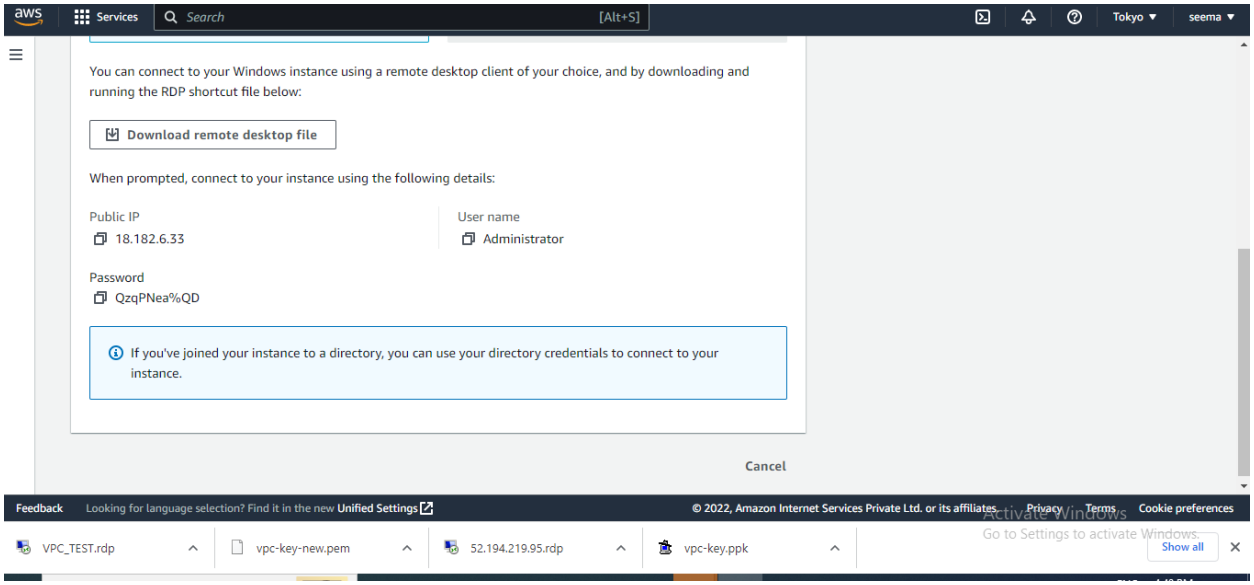


Now click on connect & download remote desktop file & click on get password & upload private key file that we have created while creating EC2 instance.

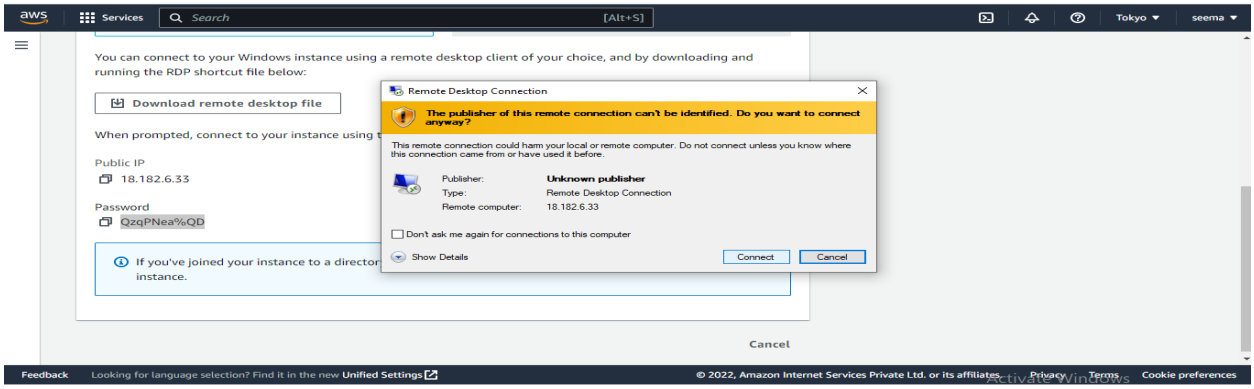


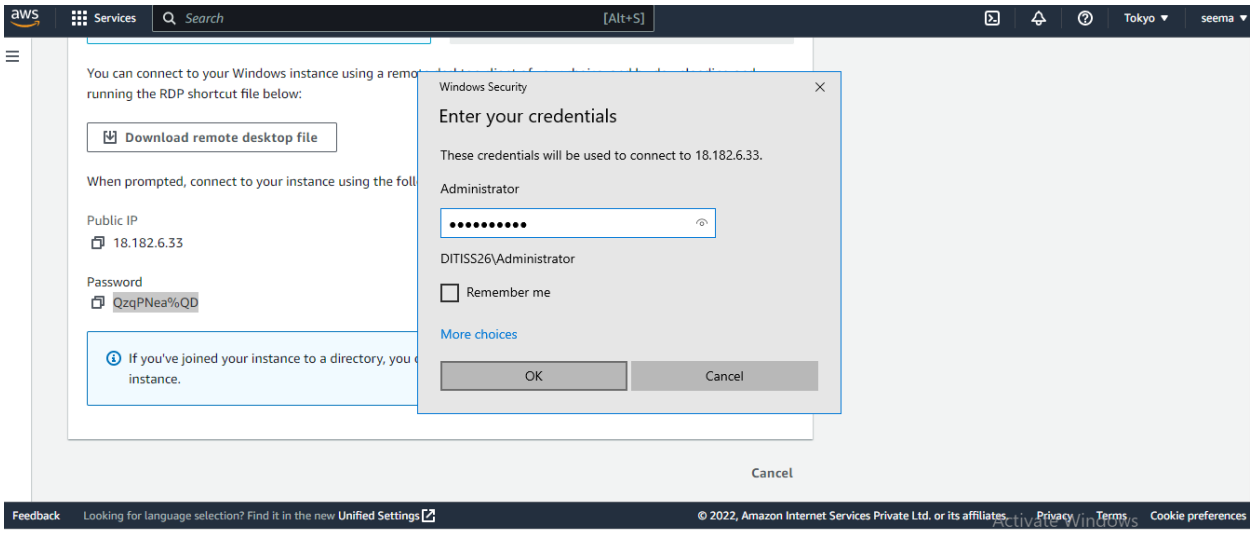


Click on decrypt password & save that password

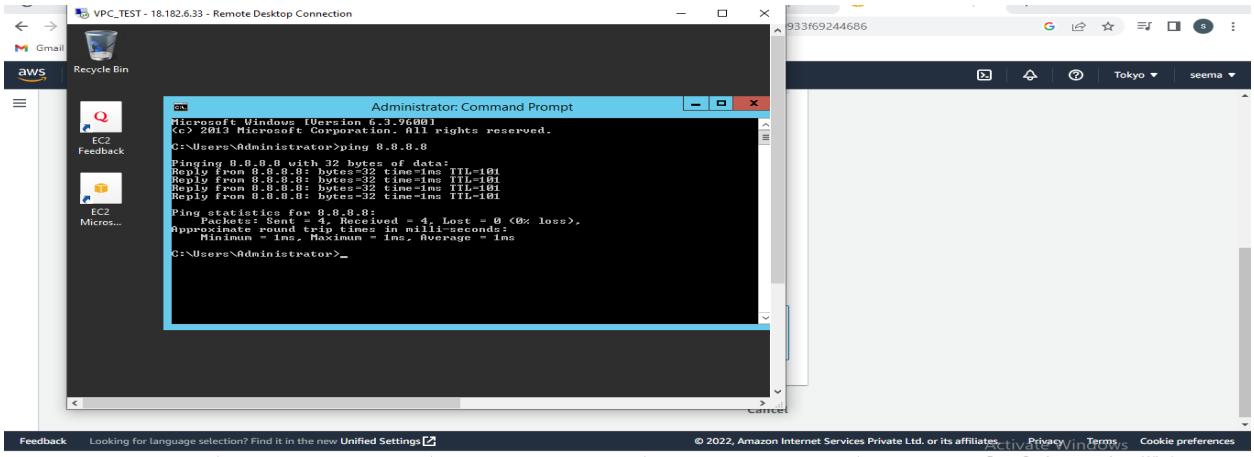


Now click on RDP file that we downloaded & connect with the password we saved





Once connected try to ping 8.8.8.8 to check if our VPC is configured correctly & it is connecting to internet



IDENTITY AND ACCESS MANAGEMENT (IAM)

IAM refers to a framework or policy and technologies for ensuring that the proper people in an Organizations have the appropriate access to technology resources.

OR

AWS Identity and Access Management is a web service that you security control access to AWS resources. We use IAM to control who is authenticated (signed-in) and authorized (has Permission) to use resources.

- When you first create AWS account, you begin in a single sign-in identity that has completely access to all AWS services and resources in the account.

- This identity is called the AWS account “Root-User” and is accessed by sighed-in with the email address and password that you used to create the account.
- AWS strongly recommends that you do not use the root user for you everyday tasks, even the administrative ones.
- Use other IAM user account to manage the administrative task of your account and securely lock away the root user credentials and use them to perform only a few account and service management task.
- IAM user limit is 5000 per AWS account. You can add up to 10 users at one time.
- You are also limit to 300 groups per AWS account.
- Default limits of managed policies attached to an IAM role and IAM user is 10.
- IAM user can be a member of maximum 10 groups.
- We can assign maximum two access keys to an IAM user.

IAM Features:

1. Shared access to your AWS account:

You can grant other people permission to administer and use resources in your AWS account without having to share your access credentials.

2. Granular permission:

- You can grant different permission to different people for different resources.
- For instance, you can allow some users complete access to EC2, S3, Dynamo DB, Redshift while for others, you can allow read only access to just some S3 buckets, or permission to administer just some EC2 instances or to access your billing information but nothing else.

3. Secure access to AWS resources for application that run on Amazon EC2:

You can use IAM features to securely give application that run on EC2 instances the credentials that they need in order to access other AWS resources. For example, include S3 buckets and RDS or Dynamo DB databases.

4. Multifactor Authentication (MFA):

You can add two factor authentications to your account and to individual users for extra security. You can use physical hardware or virtual MFA (for e.g: Google Authenticator)

5. Identity federation:
You can allow users who already have passwords elsewhere. For e.g: in your corporate network of with an internet identity provider to get temporary access to your AWS account.

6. Identity information for assurance:
If you use AWS Cloud Trail, you receive log records that include information about those who made request for resources in your account. That information is based on IAM Identities.

7. PCI-DSS compliance:
IAM supports the processing, storage and transmission of credit cards by a merchant of service provider, and has been validated as being complaint with payment card industries (PCI) data security standards (DSS).

8. Eventually consistent:

- If a request to change some data is successful, the change is committed and safely stored. However the change must be replicated across IAM which can take some time.
- IAM achieves high availability by replicating data across multiple servers within AWS data centre around the world.

Fee to Use: AWS IAM is a feature of AWS account offered at no additional charge. You will be charged only for use of other AWS products by your IAM users.

IAM Terms:
Following are the major terms which are used in an IAM account.
1. Principal
2. Request
3. Authentication
4. Authorization

5. Action/Operation

6. Resources

1. Principal:

- A principal is a person or application that can make a result for an action or operation on an AWS resources.
- Your administrative IAM user is your first principal.
- You can allow users and services to assume a role.
- IAM users, roles, federated users and application are all AWS principals.
- You can support federated users of programmatic access to allow an application to access your AWS account.

2. Request:

When a principal tries to use the AWS management console, the AWS API of the AWS CLI that principal sends a request to AWS. The request includes the following information:

- Actions: That the principal wants to perform.
- Resources: upon which the actions are performed.
- Principal information: it's including the environment from which the request was made.

Request context: before AWS can evaluate and authorize a request, AWS gathers the request information. Principal (the requester) which is determined based on the authorization data. This includes the aggregate permissions that the associated with that principal.

- Environment data: such as IP address, user agent, SSL enabled status, or the time of the day.
- Resource data: it is related to the resource that is being requested.

3. Authentication:

- A principal sending a request must be authenticated (sighed into AWS) to send a request to AWS.
- Some AWS services, like AWS S3 allow request from anonymous users, they are exception to the role.
- To authenticate from the console as a root user, you must sign-in with your user name and password.

- To authenticate from the API to CLI, you must provide your access key and secrete key.
- You might also be required to provide additional security information like MFA (e.g: Google Authentication)

4. Authorization:

- To authorize request, IAM uses value from the request context to check, for matching policies and determine whether to allow or deny the request.
- IAM policies are stored in IAM as JSON documents and specify the permission that are allowed or denied.
- User (identity) Based Policy specifies permission allowed/denied for principals.
- Note: by default the AWS root user access to all the resources in that account.

Resource Based Policies:

- It specifies permission allowed/denied for resources. Popular for granting cross account permission.
- IAM checks each policy that matches the context of your request.
- If a single policy includes a denied actions, IAM denies the entire request and stop evaluating. This is called explicit deny.

The evaluation logic follows these Rules:

- By default, all request are denied.
- An explicit allow overrides this default.
- An explicit deny overrides any allows.
- You can create a new IAM policy in the AWS management console using one of the following ways:
- JSON: you can create your own JSON syntax.
- Visual Editor: you can construct a new policy from scratch in the visual editor. If you can use the visual editor you do not have to understand JSON syntax.
- Import: you can import a managed policy with in your account and then edit the policy to customize it to your specific requirement.

5. Actions:

- Actions are defined by a service, and more the things that you can do to a resource such as viewing, creating, editing, and deleting that resource.
- IAM supports approx. 40 actions for a user resource including create user, delete user etc.
- Any actions or resources that are not explicitly allowed are denied by default.
- After your request has been authenticated and authorized, AWS approves the actions in your request.

6. Resource:

- A resource is an entity that exists within a service.
- Examples are EC2 instances, S3 bucket, IAM users, and Dynamo DB table.
- Each AWS service defines a set of actions that can be performed on each resource.
- After AWS approves the actions in your request those actions can be performed on the related resources within your account.
- If you create a request to perform an unrelated action on a resource that request is denied.
- When you provide permissions using an identity based policy in IAM then you provide permissions to access resources only within the same account.

Identity Federation:

- If your account users already have a way to be authenticated such as authentication through your corporate network, you can federated those user identities into AWS.
- A user who has already logged to the corporate using their corporate identity, the corporate can replace their existing identity with a temporary identity in your AWS account.
- The user can work in the AWS management console.
- Similarly, an application that the user is working with can make programmatic request using permission that you make.

Federation is particularly useful in those cases:-

1. If your corporate directory is compatible with Security Assertion Markup Language (2.0):

- You can configure your corporate directory to provide Single Sign-On (SSO) access to the AWS management console for your users.

- If your corporate directory is not compatible with SAML 2.0, you can create identity broker application to provide single sign-on access to the AWS management console for your users.
- If your corporate directory is Microsoft active directory, you can use AWS directory service to establish trust between your corporate directory and your AWS account.

2. Your users already have Internet Identities:

- if you are creating a mobile app or web-based app that can let users identity themselves through an internet identity provider like login with amazon,facebook, google or any open ID connect (OIDC) compatible identity provider, the app can use web federation to access AWS.
- AWS recommends to use AWS Cognito for identity federation.

IAM Users and SSO:

- IAM users in your account have access only to the AWS resources that you specify in the policy that is attached to the user of to an IAM group that use the user belongs to.
- To work in the console user must have permissions to perform the actions that the console performs such as listing and creating AWS resources.

IAM Identities:

- IAM identities is what you create under your AWS account to provide authentication for people, application and process in your AWS account.
- Identities represents the user and can be authenticated and then authorized to perform actions in AWS.
- Each of these can be associated with one or more policies to determine what actions a user, role or member of the group can do with which resources and under what conditions
- IAM group is a collection of IAM user.
- IAM role is very limit IAM user.

A. IAM Users:

- An IAM user is an entity that you create in AWS. It represents the person or service who uses the IAM user to interact with AWS.
- You can create 5 users at time.
- An IAM user can represent an actual person or an application that requires AWS access to perform actions on AWS resources.
- A primary use of IAM users is to give people the ability to sign-in to the AWS management console for interactive task and to make programmatic request to AWS services using the API or CLI.

For any user you can assign them:

- A username and password to access the AWS console.
- An access key ID and secrete key that can use for programmatic access.
- The newly created IAM user have no password and no access key. You need to create the user password.
- Each IAM user is associated with one and only one AWS account.
- Users are defined within your account, so user do not have to do payment. Bill would be pay by the parent account.

B. IAM Groups:

- An IAM group is a collection of IAM users.
- It is a way to assign permission/policies to multiple users at once.
- Use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users.
- For E.g: you could have a group called HR and give that group the types of permissions that HR department typically needs.
- Any user in that group automatically has the permission that are assigned to the group.
- If a new user joins your organization and should have administrator privileges, you can assign the appropriate permissions by adding the user to that group.
- If a person changes job in your organization, instead of editing that user's permission, you can remove him or her from the old groups and add him or her to the appropriate new groups.

IAM Group Limitations:

- A group is not truly an identity in IAM because it cannot be identified as a principal in a permission policy.
- Group cannot be nested.
- One has a limit of 300 groups in an AWS account.
- A user can be a member of up to 10 IAM groups.

C. IAM Roles:

- An IAM role is very similar to a user, in that it is an identity with permission policies that determine what the identity can and cannot do in AWS.
- An IAM role does not have any credentials (password or access key) associated with it.
- Instead of being associated with one person, a role is intended to be assumable by anyone who needs it.
- An IAM user can assume a role to temporarily take on different permissions for a specific task.
- An IAM role can be assigned to a federated user who sign-in by using an external identity provider instead of IAM.

IAM Temporary Credentials:

- Temporary credential are primary used with IAM roles but there are also other uses.
- You can request temporary credentials that have a more restricted set of permissions than your standard IAM users.
- This prevent you from accidentally performing task that are permitted by the more restricted credentials.
- A benefit of temporary credentials in that they expire automatically after a set period of time.

Permissions and Policies:

- The access management portion of AWS Identity and Access Management (IAM) helps you to define what a user or other entity is allowed to do in an account, often referred to as authorization.

- Permissions are granted through policies that are created then attached to user, groups of roles.

Policies and User:

- By default, IAM users can't access anything in your account.
- You grant permissions to a user by creating a policy, which is a document that defines the effect, actions, resource and optional conditions.
- Any actions or resources that are not explicitly allowed are denied by default.

IAM Multiple Policies:

- Users of groups can have multiple policies attached to them that grant different permission.
- In the case of multiple policies attached to a user or group, the user's permission are calculated based on the combination of policies.

Federated Users and Roles:

- Federated user don't have permanent identities in your account the way that IAM users do.
- To assign permissions to federated users you can create an entity referred to as a role and define permission for the role.
- When a federated user sign-in to AWS the users is associated with the role and is granted the permission that are defined in the role.

Resource based Policy:

- In some cases like S3 bucket, you can attach a policy to a resource in addition to attaching it to a group or user. This is called a resource based policy.
- A resource based policy contains slightly different information than user-based policy.
- In resource based policy you specify what actions are permitted and what resource is affected.
- You also explicitly list who is allowed access to the resource (a principal).

- Resource based policies include a principal element that specifies who is granted the permissions.

IAM User-The Root User:

- When you first create an AWS account, you create an account (or root user) identity, which you use to sign-in to AWS.
- The account root user credentials are the e-mail address to create the account and a password which can be used to sign-in to the AWS Management console as the root user.
- When you sign-in as root user, you have complete unrestricted access to all resources in your account including access to your billing information and the ability to change your password.
- The level of access is necessary when you initially set up the account.
- It is not possible to restrict the permission that are granted to the AWS account.

AWS Recommends That:

- AWS recommends that you don't use root user credentials for everyday access.
- Also AWS recommends that you do not share your root user credentials with anyone because doing so gives them unrestricted access to your account.
- Create an IAM user for yourself and then assign yourself administrative permission for your account.
- You can then sign-in as that user to add more users as needed.
- An IAM user with administrator permissions is not the same things as the AWS account root users.

IAM Users:

- An IAM user is an entity that you create in AWS. It represents the person or service who uses the IAM user to interact with AWS.
- An IAM can represent an actual person or an application that requires AWS access to perform action on AWS resources.

- IAM users are global entities, like an AWS account is today. No region is required to be specified when you define user permissions. Users can use AWS services in any geographic region.

For Any User You can assign them:

- A user name and password to access the AWS console.
- An access key (access key and secrete key) that they can use for programmatic access (issuing request) to your AWS account.
- You assign either or both based on the user activities and needs.
- You can view and download your secret access key only when you create the access key.
- You cannot view or recover a secret access key later.
- If you lose your secrete access key, you can create a new access key.
- Each IAM user is associated with one AWS account.

By Default a new IAM User:

- A new IAM user has no permission to do anything.
- Has no password and no access key (neither an access key ID nor a secret access key). It means no credentials of any kind.
- You must create the type of credentials for an IAM users based on what the user will be doing.
- You can grant user permissions by attaching IAM policies to them directly or making them members of IAM group where they inherit the group policies/permissions.
- You can have up to 5000 user per account.

IAM Roles:

- An IAM role is a set of permissions that grant access to the actions and resources in AWS.
- These permissions area attached to the role, not to an IAM user or group. Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

- A role does not have standard long-term credentials (password or access key) associated with it.
- If a user assumes a role, temporary security credentials are created dynamically and provided to the user.

☐Following entities can used role:

- An IAM user in the same AWS account.
- An IAM user in a different AWS account.
- A webserver offered by AWS such as Amazon EC2.

There are Two ways to use a Role:

1. Internally in the IAM Console:

- IAM users in your account using the IAM console can switch to a role to temporarily use the permissions of the role in the console.
- The user give up their original permission and take on the permission assigned to the role.
- When the user exists the role, their original permissions are restored.

2. Programmatically with the AWS CLI, tools for windows powershell or API:

- An application or a service offered by AWS (like Amazon EC2) can assume a role by requesting temporary security credentials for a role with which to make programmatic request to AWS.
- You use a role this way so that you don't have to share or maintain long-term security credentials for each entity that requires access to a resource.

Difference between IAM Role and Resource Based Policy:

- Unlike a user-based policy, a resource based policy specifies who can access that resource.
- Cross account access with a resource based policy has an advantage over a role, with a resource that is accessed through a resource-based policy, the user still works in the trusted account and does not have to give up this or her user permissions in place of the role permissions.

- In other words, the user continuous to have access to resources in the trusted account at the same time as he or she has access to the resource in the trusting account.
- This is useful for task such as copying information to or form the shared resource in the other account.
- Note that not all services support resource-based policy.

IAM Role Delegation:

- Delegation is the granting of permission to someone to allow access to resource that you control.
- Delegation involves setting up a trust between the account that owns the resource (the trusting account) and the account that contains the users that need to access the resource (the trusted account).

The trusted and trusting accounts can be of the following:

i. The same account

ii. Two accounts that are both under your organization's control.

iii. Two account owned by different organizations.

To delegate permission to access a resource you create an IAM role that has two policies attached.

i. The Trust Policy

ii. The Permission Policy

- The trusted entity is included in the policy as the principal element in the document.
- When you create a trust policy, you cannot specify a wildcard (*) as a principal.

Cross Account Permissions:

- You might need to allow user from another AWS account to access resources in your AWS account. If so, don't share security credentials, such as access keys between accounts. Instead use IAM roles.
- You can define a role in the trusted account that specifies what permissions the IAM users in the other account are allowed.

- You can also designate which AWS accounts have the IAM users that are allowed to assume the role. We do not define users here rather AWS account.

Role for Cross-Account Access:

- Granting access to resources in one account to a trusted principal in a different account.
- Roles are the primary way to grant cross-account access.
- However with some of the web services offered by AWS, you can attach a policy directly to a resource. These are called resource-based policy. You can use them to grant principals in another AWS account access to the resource.

The following services support resource-based policy:

- Amazon S3.
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Glacier Vault

Introduction to Openstack

It is a free open standard cloud computing platform that first came into existence on July 21′ 2010. It was a joint project of Rackspace Hosting and NASA to make cloud computing more ubiquitous in nature. It is deployed as Infrastructure-as-a-service (IaaS) in both public and private clouds where virtual resources are made available to the users. The software platform contains interrelated components that control multi-vendor hardware pools of processing, storage, networking resources through a data center.

In OpenStack, the tools which are used to build this platform are referred to as “projects”. These projects handle a large number of services including computing, networking, and storage services. Unlike virtualization, in which resources such as RAM, CPU, etc are abstracted from the hardware using hypervisors; OpenStack uses a number of APIs to abstract those resources so that users and the administrators are able to directly interact with the cloud services.

OpenStack components

Apart from various projects which constitute the OpenStack platform, there are nine major services namely Nova, Neutron, Swift, Cinder, Keystone, Horizon, Ceilometer, and Heat. Here is the basic definition of all the components which will give us a basic idea about these components.

1. **Nova (compute service):** It manages the compute resources like creating, deleting, and handling the scheduling. It can be seen as a program dedicated to the automation of resources that are responsible for the virtualization of services and high-performance computing.
2. **Neutron (networking service):** It is responsible for connecting all the networks across OpenStack. It is an API driven service that manages all networks and IP addresses.
3. **Swift (object storage):** It is an object storage service with high fault tolerance capabilities and it is used to retrieve unstructured data objects with the help of Restful API. Being a distributed platform, it is also used to provide redundant storage within servers that are clustered together. It is able to successfully manage petabytes of data.
4. **Cinder (block storage):** It is responsible for providing persistent block storage that is made accessible using an API (self- service). Consequently, it allows users to define and manage the amount of cloud storage required.
5. **Keystone (identity service provider):** It is responsible for all types of authentications and authorizations in the OpenStack services. It is a directory-based service that uses a central repository to map the correct services with the correct user.
6. **Glance (image service provider):** It is responsible for registering, storing, and retrieving virtual disk images from the complete network. These images are stored in a wide range of back-end systems.
7. **Horizon (dashboard):** It is responsible for providing a web-based interface for OpenStack services. It is used to manage, provision, and monitor cloud resources.
8. **Ceilometer (telemetry):** It is responsible for metering and billing of services used. Also, it is used to generate alarms when a certain threshold is exceeded.
9. **Heat (orchestration):** It is used for on-demand service provisioning with auto-scaling of cloud resources. It works in coordination with the ceilometer.

These are the services around which this platform revolves around. These services individually handle storage, compute, networking, identity, etc. These services are the base on which the rest of the projects rely on and are able to orchestrate services, allow bare-metal provisioning, handle dashboards, etc.

Advantages of using OpenStack

- It boosts rapid provisioning of resources due to which orchestration and scaling up and down of resources becomes easy.

- Deployment of applications using OpenStack does not consume a large amount of time.
- Since resources are scalable therefore they are used more wisely and efficiently.
- The regulatory compliances associated with its usage are manageable.

Disadvantages of using OpenStack

- OpenStack is not very robust when orchestration is considered.
- Even today, the APIs provided and supported by OpenStack are not compatible with many of the hybrid cloud providers, thus integrating solutions becomes difficult.
- Like all cloud service providers OpenStack services also come with the risk of security breaches.

Hyper Converged Infrastructure (HCI)

Hyper convergence is an IT framework that combines storage, computing and networking into a single system in an effort to reduce data center complexity and increase scalability.

Hyper converged platforms include a hypervisor for virtualized computing, software-defined storage, and virtualized networking. They typically run on standard, off-the-shelf servers and multiple nodes can be clustered to create pools of shared compute and storage resources, designed for convenient consumption.

The use of commodity hardware, supported by a single vendor, yields an infrastructure that's designed to be more flexible and simpler to manage than traditional enterprise storage infrastructure. For IT leaders who are embarking on data center modernization projects, hyper convergence can provide the agility of public cloud infrastructure without relinquishing control of hardware on their own premises.

Hyper convergence vs. cloud

You may be wondering what the distinction is between an HCI platform and the cloud. After all, both use software-defined management systems to pool hardware resources which can then be virtualized.

Both public and private clouds seek to make things easier for the user by putting a layer of abstraction between them and the compute resources they're using. A wide variety of hardware and software could be supplying those resources behind the scenes.

HCI, by contrast, is aimed at making it easier for IT to provide services to users by standardizing and unifying the platform. IT should be able to easily deploy, maintain, and scale HCI

infrastructure to meet an organization's needs, but HCI doesn't seamlessly scale up and down in response to user requests in real time the way the cloud can.

What is Agile?

Agile Methodology involves continuous iteration of development and testing in the SDLC process. This software development method emphasizes iterative, incremental, and evolutionary development.

Agile development process breaks the product into smaller pieces and integrates them for final testing. It can be implemented in many ways, including scrum, kanban, XP, etc.

Scrum

If you are just getting started, think of Scrum as a way to get work done as a team in small pieces at a time, with continuous experimentation and feedback loops along the way to learn and improve as you go. Scrum helps people and teams deliver value incrementally in a collaborative way. As an agile framework, Scrum provides just enough structure for people and teams to integrate into how they work, while adding the right practices to optimize for their specific needs. It starts with understanding the Scrum framework which is defined in The Scrum Guide and was first introduced to the world in 1995 as a better way of team collaboration for solving complex problems. The Scrum framework is fairly simple being made up of a Scrum Team consisting of a Product Owner, a Scrum Master and Developers, each of which have specific accountabilities. The Scrum Team takes part in five events and produces three artifacts. Scrum co-creators Ken Schwaber and Jeff Sutherland wrote and maintain The Scrum Guide, which explains Scrum clearly and succinctly. The guide contains the definition of Scrum, describing the Scrum accountabilities, events, artifacts and the guidance that binds them together.

So, why is it called Scrum? People often ask, “Is Scrum an acronym for something?” and the answer is no. It is actually inspired by a scrum in the sport of rugby. In rugby, the team comes together in what they call a scrum to work together to move the ball forward. In this context, Scrum is where the team comes together to move the product forward.

Scrum is an empirical process, where decisions are based on observation, experience and experimentation. Scrum has three pillars: transparency, inspection and adaptation. This supports the concept of working iteratively. Think of Empiricism as working through small experiments, learning from that work and adapting both what you are doing and how you are doing it as needed.

One critical Scrum Team characteristic that binds all of the elements together is Trust. If Trust is not present on a Scrum Team, there will likely be tension and bottlenecks in the way of getting work done. The Scrum Values are also critical for Scrum Teams to adhere to as they help to guide how you work and drive trust. The Scrum Values of Courage, Focus, Commitment, Respect, and Openness, are all important elements that Scrum Team members must consider when working together. The Scrum Values are particularly important in environments where experimentation is core to making progress.

In a nutshell, Scrum requires an environment where:

- Increments of valuable work are delivered in short cycles of one month or less, which are called Sprints. Ongoing feedback occurs during the Sprint, allowing for inspection and adaptation of the process and what will be delivered.
- The Scrum Team has a Scrum Master, a Product Owner and Developers, who are accountable for turning the selection of the work into an Increment of value during a Sprint.
- The Scrum Team and other members of their organization, business, users or customer-base known as stakeholders, inspect the results of the Sprint and adjust for the next one.

Kanban

Kanban is a popular Agile Software Development Methodology. It is basically a signaling device that instructs the moving of parts in a 'pull' production system, developed as part of the TPS (Toyota Production System). Kanban is about envisioning the existing workflow in terms of steps. These steps can be created on the whiteboard.

The main aim of Kanban is to reduce WIP (Work-In-Progress), or inventory, between processes by ensuring the upstream process creates parts as long as its downstream process needs it. The goal of the Kanban execution is to ensure work items move to the next steps quickly to realize business value faster.

The Kanban method is an approach to evolutionary and incremental systems and process change for organizations. A work-in-progress limited pull system is used as the central mechanism to uncover system operation (or process) complications and encourage collaboration to improve the system continuously.

Electronic Kanban boards are also available in ALM tools like Rally (CA Agile), Jira, SwiftKanban, LeanKit Kanban, etc. stages could be configured in these tools, and movement of tickets between stages could be viewed in these tools.

When Would The Kanban Approach Be Needed?

Kanban is best suited in the below scenarios:

- Dynamic/ frequent changing requirements which need to be delivered faster.
- In case of changing priorities, the prioritized work can be pulled by the team as soon as the WIP limit drops.
- Frequent releases are there (Periodically).
- When incoming work is continuous.
- Where task priority needs to be decided dynamically based upon task nature and type.
- The best suit is for Ticket or Production support projects.
- Kanban could be used by any function of an organization as well, for instance in Marketing, Sales, and HR.

However, Kanban might not be the right fit for projects where:

- Tasks could remain in the 'wait' state for long.
- Mainly research-oriented takes are there.
- For enhancements where requirements are evolving/ unclear.
- No prior scope is not defined and tasks keep on evolving.
- Too much dependency is there between tasks.
- If all the items across work stages need to be collated, then only deployed.

Kanban Board/Card

It is critical to understand the visualization of workflow stages in the task execution pipeline. Kanban board provides a simple way to understand the process. It can be explained as follows:

1. Every request received is put on the Kanban board.
2. A column on the board represents a stage (these stages are termed as the Work stage) during the lifecycle of bugs/ tickets. For instance, the Kanban board can have 4 stages- Received / Acknowledged, In-progresses, UAT & Done.
3. The received stage could be called a "Backlog" also.

4. The team could decide the names for the phases based on the terminology used by their respective teams.
5. Kanban board could be a simple whiteboard on which sticky notes could be used with ticket details or an electronic Kanban board could be used.
6. ALM tools like Rally/ Jira could be configured to use the Kanban board.
7. The board can give a signal in case the bugs/ tickets are stuck in one stage for a long.
8. For electronic boards, one can configure the Kanban board in a way that tickets/ user stories along with the time stamp are visible.
9. For whiteboards that are maintained manually, the team can enter the date/ time.

Principles of Kanban

Kanban is based on four key principles which are mentioned below:

1. **Start with the existing process:** It is a change management method that starts with the existing process. Changes are done in the system in incremental and evolutionary ways. Unlike Scrum, there's no specific process or roles defined in Kanban.
2. **Agree to continue evolutionary and incremental changes:** After starting with the existing process, the team must agree on continuous, incremental, and evolutionary changes. The changes should be small and incremental. Rapid and substantial changes may be effective but they will be subjected to larger resistance as well by the Team.
3. **Admire current roles, processes, responsibilities & titles:** Though Kanban suggests continuous incremental changes in the process, it respects current roles, responsibilities, and job titles. This helps the team to gain confidence as they get started with Kanban.
4. **Leadership at all levels:** Kanban does not expect leadership from a specific set, rather the actions of leadership at all levels in the organization are very much encouraged.

Kanban Practices

The following are the six core Kanban practices:

1. **Limit WIP:** Limiting Work-In-Process (WIP) implies that a pull system is executed on either parts or the whole workflow. It (PULL system) will act as one of the key stimuli for incremental, continuous, and evolutionary changes to the system. Limit WIP assigns explicit limits to the number of items that may be in progress at each workflow state.

2. **Visualize:** Visualizing the workflow and making it visible is important so as to know how work proceeds. Without understanding the flow of work, incorporating the right changes is difficult. Usually, a card wall with columns and cards is used to visualize the flow of work. Different states or steps within the workflow are represented by the columns on the card wall.

3. **Manage flow:** Flow of work through every state within the workflow should be observed, measured, and informed. By managing the flow vigorously, the incremental, continuous, and evolutionary modifications to the system can be assessed to have negative or positive effects on the system.

4. **Improve Collaboratively, Evolve Experimentally:** Kanban encourages small incremental, continuous, and evolutionary changes. Whenever teams have a common understanding of concepts about work, process, workflow, and risk, they are more likely to be able to form a shared understanding of a problem and suggest enhancement actions that could achieve a consensus.

5. **Implement Feedback Loops:** Early feedback from clients and the pull system are important in Kanban. If we get feedback from different stakeholders and processes, it will help to eliminate risk and optimize the delivery process.

6. **Make Policies Explicit:** Until the mechanism of a process is not made clear, it is difficult to hold a debate and discuss ways to improve it. Without a clear understanding of how work is truly done and how things actually work, any conversation of complications tends to be anecdotal, emotional, and subjective. With a clear understanding, it is possible to hold a more rational, empirical, objective discussion of issues. It is more likely to facilitate consensus around improvement suggestions.

	Scrum	**Kanban**
Origin	Software development	Lean manufacturing
Ideology	Learn through experiences, self-organize and prioritize, and reflect on wins and losses to continuously improve.	Use visuals to improve work-in-progress

Cadence	Regular, fixed-length sprints (i.e. two weeks)	Continuous flow
Practices	Sprint planning, sprint, daily scrum, sprint review, sprint retrospective	Visualize the flow of work, limit work-in-progress, manage flow, incorporate feedback loops
Roles	Product owner, scrum master, development team	No required roles

What is Lean Methodology?

Lean methodology originally sprouted in Japan at Toyota Production System. Now it has made its way into businesses, workplaces, and other knowledge-driven settings around the world. As Jim Benson of Modus Cooperandi said, “Lean is both a philosophy and a discipline which, at its core, increases access to information to ensure responsible decision making in the service of creating customer value.”

Fundamentals of Lean Methodology

Though search inquiries on lean methodology will immediately bring “eliminating waste” to the front, this is not the complete definition. Fundamentally, the method emphasizes the idea of “continuous improvement.” Lean thinkers who brought the methodology from Japan to the West (specifically James Womack and Daniel Jones) specified five core principles:

- Value: Understand what customers value in a product or service
- Value Stream: What goes into maximizing value and eliminating waste throughout the entire process from design to production
- Flow: All product processes flow and synchronizes seamlessly with each other

- Pull: Flow is made possible by “pull,” or the idea that nothing is made before it is needed, thereby creating shorter delivery cycles
- Perfection: Relentlessly pursue perfection by constantly engaging the problem-solving process

The idea is to refine internal processes as much as possible to give consumers the highest value possible in a product or service. Anything that doesn’t contribute to the product’s value to the customer is considered inefficient.

Another key to lean is its definition of waste, of which there are eight types:

- Motion: Unnecessary movement of people or processes (equipment and manufacturing machinery, for example). Repetitive movements that do not add value translates to wasted time and resources.
- Over-processing: Doing unnecessary processes or steps than what is required to create a valuable product.
- Extra-processing: Products require more work or quality than necessary to deliver value to the customer.
- Defects: Manufacturing processes create defective products — which becomes wasted materials.
- Transport: Like motion, but over greater distances to include the transport of tools, inventory, people, or products further than necessary.
- Human Potential: Underused skills and talent due to poor employee management and team structure lead to a lack of morale and productivity.
- Waiting: Idle equipment and waiting on materials or equipment can slow down processes and efficiency.
- Inventory: Excessive products and inventory take up space, reveal overproduction, and create back work.

It’s easy to see how continuous improvement is always possible and includes every level of the business, from talent management, manufacturing, IT, marketing, and more.

Kaizen

Over 30 years ago, Masaaki Imai sat down to pen the groundbreaking book ‘Kaizen: The Key to Japan’s Competitive Success’ (McGraw Hill). Through this book, the term KAIZEN™ was introduced in the western world. Today KAIZEN™ is recognized worldwide as an important pillar of an organization’s long-term competitive strategy. Since introducing this term as a systematic approach for business improvement, companies that implement KAIZEN™ have continually yielded superior results.

"KAIZEN™ means improvement. Moreover, it means continuing improvement in personal life, home life, social life, and working life. When applied to the workplace KAIZEN™ means continuing improvement involving everyone – managers and workers alike." Masaaki Imai, Founder of Kaizen Institute

Introduction to DevOps

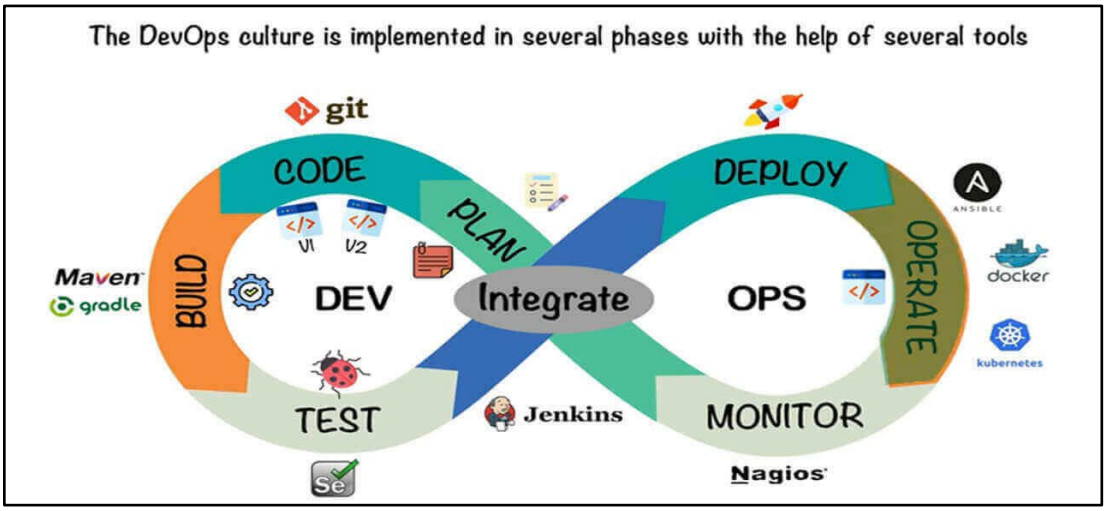
DevOps is a software development method which focuses on communication, integration, and collaboration among IT professionals to enable rapid deployment of products.

DevOps is a culture that promotes collaboration between Development and Operations Teams. This allows deploying code to production faster and in an automated way. It helps to increase an organization’s speed to deliver applications and services. It can be defined as an alignment of development and IT operation.

Key Difference Between Agile and DevOps

- DevOps is a practice of bringing development and operations teams together whereas Agile is an iterative approach that focuses on collaboration, customer feedback and small rapid releases.
- DevOps focuses on constant testing and delivery while the Agile process focuses on constant changes.
- DevOps requires a relatively large team while Agile requires a small team.
- DevOps leverages both shifts left and right principles, on the other hand, Agile leverages shift-left principle.
- The target area of Agile is Software development whereas the Target area of DevOps is to give end-to-end business solutions and fast delivery.

- DevOps focuses more on operational and business readiness whereas Agile focuses on functional and non-function readiness.



Difference between Agile and DevOps

Parameter	**Agile**	**DevOps**
What is it?	Agile refers to an iterative approach which focuses on collaboration, customer feedback, and small, rapid releases.	DevOps is considered a practice of bringing development and operations teams together.
Purpose	Agile helps to manage complex projects.	DevOps central concept is to manage end-to-end engineering processes.
Task	The Agile process focuses on constant changes.	DevOps focuses on constant testing and delivery.

Implementation	Agile methods can be implemented within a range of tactical frameworks like a sprint, safe and scrum.	The primary goal of DevOps is to focus on collaboration, so it doesn't have any commonly accepted framework.
Team skill set	Agile development emphasizes training all team members to have a wide variety of similar and equal skills.	DevOps divides and spreads the skill set between the development and operation teams.
Team size	The Small Team is at the core of Agile. As the smaller the team, the fewer people on it, the faster they can move.	Relatively larger team size as it involves all the stakeholders.
Duration	Agile development is managed in units of "sprints." This time is much less than a month for each sprint.	DevOps strives for deadlines and benchmarks with major releases. The ideal goal is to deliver code to production DAILY or every few hours.
Feedback	Feedback is given by the customer.	Feedback comes from the internal team.
Target Areas	Software Development	End-to-end business solution and fast delivery.
Shift-Left Principles	Leverage shift-left	Leverage both shifts left and right.

Emphasis	Agile emphasizes software development methodology for developing software. When the software is developed and released, the agile team will not care what happens to it.	DevOps is all about taking software which is ready for release and deploying it in a reliable and secure manner.
Cross-functional	Any team member should be able to do what's required for the progress of the project. Also, when each team member can perform every job, it increases understanding and bonding between them.	In DevOps, development teams and operational teams are separate. So, communication is quite complex.
Communication	Scrum is the most common method of implementing Agile software development. Daily scrum meetings are carried out.	DevOps communications involve specs and design documents. It's essential for the operational team to fully understand the software release and its hardware/network implications for adequately running the deployment process.
Documentation	The Agile method is to give priority to the working system over complete documentation. It is ideal when you're flexible and responsive. However, it can hurt when you're trying to turn things over to another team for deployment.	In DevOps, process documentation is foremost because it will send the software to the operational team for deployment. Automation minimizes the impact of insufficient documentation. However, in the development of complex software, it's difficult to transfer all the knowledge required.
Automation	Agile doesn't emphasize automation. Though it helps.	Automation is the primary goal of DevOps. It works on the principle to maximize efficiency when deploying software.

Goal	It addresses the gap between customer needs and development & testing teams.	It addresses the gap between development + testing and Ops.
Focus	It focuses on functional and non-function readiness.	It focuses more on operational and business readiness.
Importance	Developing software is inherent to Agile.	Developing, testing and implementation all are equally important.
Speed vs. Risk	Teams using Agile support rapid change, and a robust application structure.	In the DevOps method, the teams must make sure that the changes which are made to the architecture never develop a risk to the entire project.
Quality	Agile produces better application suites with the desired requirements. It can easily adapt according to the changes made on time, during the project life.	DevOps, along with automation and early bug removal, contributes to creating better quality. Developers need to follow Coding and Architectural best practices to maintain quality standards.
Tools used	JIRA, Bugzilla, and Kanboard are some popular Agile tools.	Puppet, Chef, TeamCity OpenStack, AWS are popular DevOps tools.
Challenges	The agile method needs teams to be more productive which is difficult to match every time.	The DevOps process needs development, testing and production environments to streamline work.
Advantage	Agile offers a shorter development cycle and improved defect detection.	DevOps supports Agile's release cycle.

What is version control?

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software team's work faster and smarter. They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

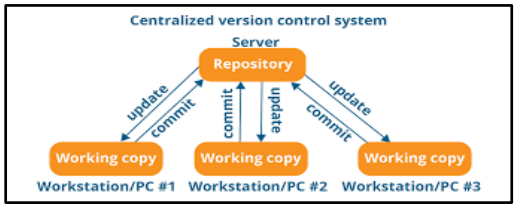
Version Control System is divided into two types

1. **Centralized version control system (CVCS)**

In centralized source control, there is a server and a client. The server is the master repository that contains all of the versions of the code. To work on any project, firstly the user or client needs to get the code from the master repository or server. So the client communicates with the server and pulls all the code or current version of the code from the server to their local machine. In other terms we can say, you need to take an update from the master repository and then you get the local copy of the code in your system. So once you get the latest version of the code, you start making your own changes in the code and after that, you simply need to commit those changes straight forward into the master repository. Committing a change simply means merging your own code into the master repository or making a new version of the source code. So everything is centralized in this model.

There will be just one repository and that will contain all the history or version of the code and different branches of the code. So the basic workflow involved in the centralized source control is getting the latest version of the code from a central repository that will contain other people's code as well, making your own changes in the code, and then committing or merging those changes into the central repository.

Eg. SVN, CVS, Perforce



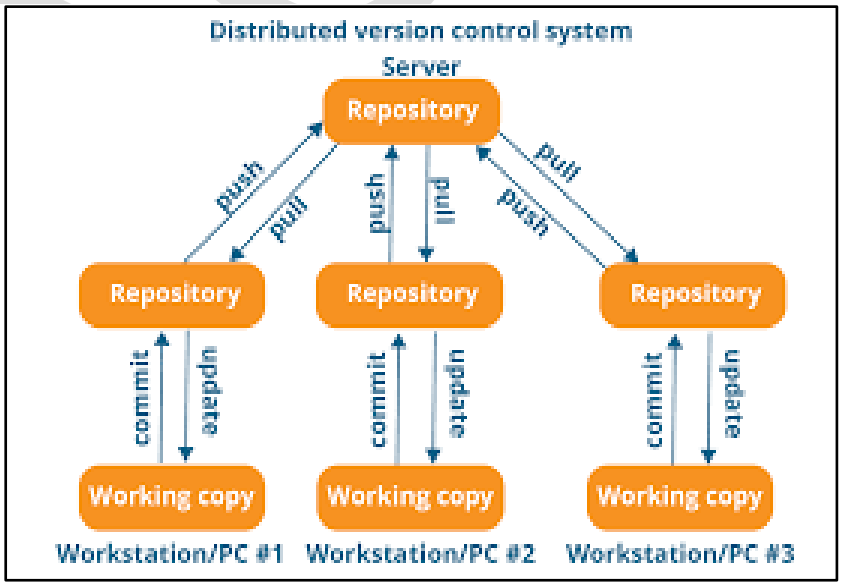
2. Distributed version control system (DVCS)

In distributed version control most of the mechanism or model applies the same as centralized. The only major difference you will find here is, instead of one single repository which is the server, here every single developer or client has their own server and they will have a copy of the entire history or version of the code and all of its branches in their local server or machine. Basically, every client or user can work locally and disconnected which is more convenient than centralized source control and that's why it is called distributed.

You don't need to rely on the central server, you can clone the entire history or copy of the code to your hard drive. So when you start working on a project, you clone the code from the master repository in your own hard drive, then you get the code from your own repository to make changes and after doing changes, you commit your changes to your local repository and at this point, your local repository will have '*change sets*' but it is still disconnected with the master repository (master repository will have different '*sets of changes*' from each and every individual developer's repository), so to communicate with it, you issue a request to the master repository and push your local repository code to the master repository. Getting the new change from a repository is called "pulling" and merging your local repository's 'set of changes' is called "pushing".

It doesn't follow the way of communicating or merging the code straight forward to the master repository after making changes. Firstly you commit all the changes in your own server or repository and then the 'set of changes' will merge to the master repository.

E.g. Git, Mercurial



CVCS	DVCS
Client need to get local copy of source from server, do the changes & commit those changes to central source on server	Each client can have a local branch as well & have a complete history on it. Client need to push the changes to branch which will then by pushed to server repository
Easy to learn & setup	Difficult for beginners multiple commands need to be remembered
Working on branches ios difficult in CVCS developer often faces merge conflict	Working on branches is easier in DVCS developers faces less conflict
It is slower as every command need to communicate with server	It is faster as mostly user deals with local copy without hitting server every time
If CVCS server is down, developer can't work	If DVCS server is down, developer can work using their local copies
Continuous internet required	Internet not required continuously

GIT

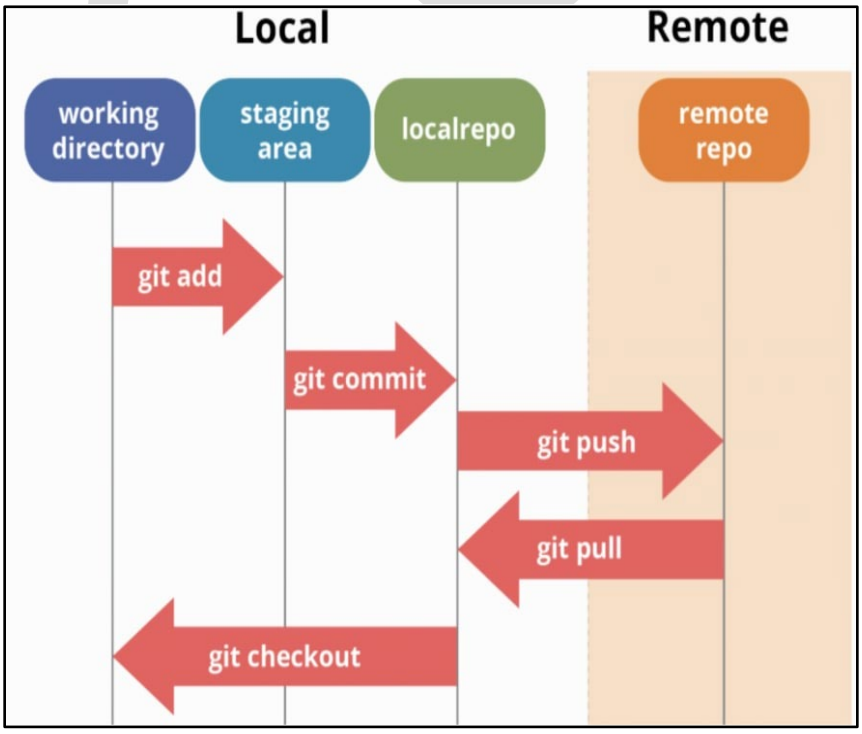
Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano. As with most other distributed version- control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server. Git works on local machine while Github works at remote server. Git is software while Github is service.

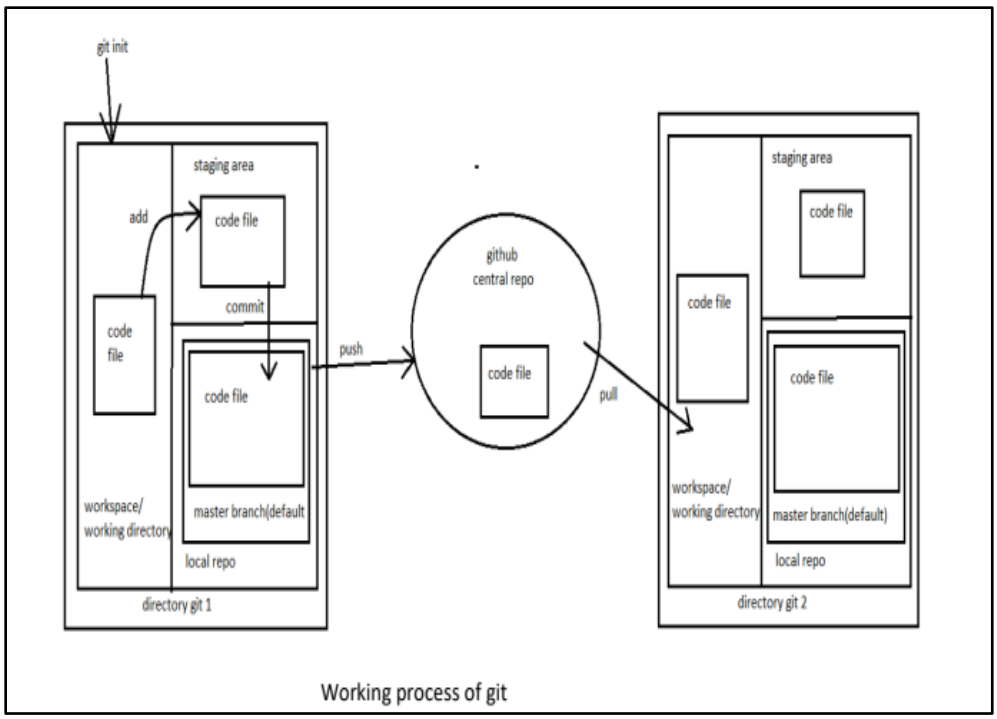
Stages of Git

1. Working directory
2. Staging area
3. Local Repo

1. **Working directory**
   It's a space where you write your code.

2. **Staging Area**
   - When you send your code from workspace to staging area that process is called "Add'
   - You can't change code once you add it to staging area
   - It is like rough draft space
   - Helps in reviewing changes
   - Sending code from staging area to local repo is "commit'
   - Used for code review by other team members





Working process of git

Repository

- It is a place where you have all your codes or kind of folder on server
- It is kind of folder related to one product
- Changes are personal to that particular repository

Server

- It stores all repositories
- It stores metadata also

Working directory

- Where you see files physically & do modification
- At a time you can work on particular branch

In other CVCS, developers generally make modifications & commit their changes directly to the repository. But git uses a different strategy git does not track each & every modified file

Whenever you commit an operation, git looks for the files present in the staging area only those files present in the staging area considered for commit & not all the modified files

Commit:-

- It means sending code files from the staging area to a local repository.
- It stores changes in the local repository and you will get a commit-ID.
- It has 40 alpha-numeric characters.
- It uses the SHA-1 checksum concept.
- Even if you change one dot (.) commit-ID will get changed.

SHA-1 Checksum Concept:-

➢ If you have a code of 1000 words in a file, so SHA-1 checksum will evaluate a binary value(like 234123411) & file will be sent to a person. Other side there will be SHA-1 checksum implemented & the output value for SHA-1 of that file is also same as sender value then it means there is no changes are made in that file, the receiver gets that file as it is sent by the sender.

➢ If any small changes is done like dot(.) then SHA-1 checksum value will be changed.

➢ It is used to know the status of code is changed or not in between sending & receiving.

Commit ID/ version ID/ version

- Reference to identify each change
- To identify who changed the file

Tags

- Tag assign a meaningful name with specific version in the repository
- Once a tag is created for a particular save, even if you create a new commit, it will not be updated

Snapshots (Incremental Backup):-

- Represents some data of a particular time.
- It is always incremental i.e. it stores the changed (appended data) only, not the entire copy.
- It'll save only updated stuff; previous code will not be saved with a new code line as a new file, but when you retrieve all code lines it will automatically merge old & new code line Y to show all code lines in a single file.

Commit

- Store changes in local repository. You will get one commit ID
- It is 40 alphanumeric characters
- It uses SHA-1 checksum concept
- Even if you change one dot, commit ID will get change
- It actually helps you to track the changes
- Commit is also known as SHA1 hash

PUSH

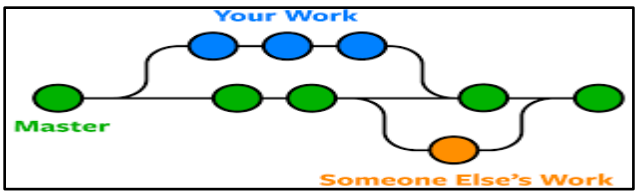
- Push operation copies changes from a local repository instances to a remote or central repo
- Used to store the changes permanently into git repo

PULL

- It copies changes from remote repo to local machine
- Pull operation is used for synchronization between two repo

Branch

- Product is same, so one repository but different task
- Each task has one separate branch
- Finally merges (code) of all branches
- Useful when you want to work parallel
- Can create one branch on the basis of another branch
- Changes are particular to that particular branch
- Default branch is ‘master’
- File created in the workspace will be visible in any of the branch workspace until you commit. Once you commit then that file belongs to that particular branch



Advantages of git

- Free & open source.
- Fast & Small:- As most of the operations are performed locally, therefore it's fast. Small in size too.
- Security:- Git uses a common cryptographic hash function called secure hash function(SHA-1) to name & identify objects within its database.
- No need for powerful hardware.
- Easier Branching:- If we create a new branch, it'll copy all the codes to the new branch.

Types of repositories

Bare repository (central repo)

- Store & share only
- All central repositories are bare repo

Non bare repo (local repo)

- Where you can modify the files
- All local repositories are non bare repo

How to use git & create a github account?

Configuring git:-

➢ sudo su

➢ yum update –y

➢ yum install git

➢ git –version

➢ git config –global user.name "abdul"

➢ git config –global user.email abdul@gmail.com

➢ git config –list (to see created user in git)

➢ open www.github.com

➢ create an account by sign up.

➢ Click join free plan.

➢ Now click on complete setup.

➢ Now verify your email from gmail.

➢ Sign out then sign in again.

How to Commit, Push & Pull from Github?

➢ Login into linux machine.

➢ Create a directory & go inside it.

```
[root@localhost abdlqdr]# mkdir abdulgit
[root@localhost abdlqdr]# ls
abdulgit
[root@localhost abdlqdr]# cd abdulgit
[root@localhost abdulgit]#
```

➢Run (git init) command inside directory to make it local repo.

```
[root@localhost abdulgit]# git init
Initialized empty Git repository in /home/abdlqdr/abdulgit/.git/
[root@localhost abdulgit]# ls -a
.  ..  .git
[root@localhost abdulgit]#
```

➢ Create a file-> touch myfile(put some data).

```
[root@localhost abdulgit]# touch codefile
[root@localhost abdulgit]# ls
codefile
[root@localhost abdulgit]#
```

➢ git status (to check what data is in local repo).

```
[root@localhost abdulgit]# git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        codefile

nothing added to commit but untracked files present (use "git add" to track)
[root@localhost abdulgit]#
```

➢ git add . (to add code files from the working area to the staging area).

```
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   codefile

[root@localhost abdulgit]#
```

➢ git commit –m “first commit from centos” (to commit & give a message).

```
[root@localhost abdulgit]# git commit -m "first commit from centos"
[master (root-commit) c4bc63c] first commit from centos
 1 file changed, 1 insertion(+)
 create mode 100644 codefile
[root@localhost abdulgit]#
```

➢ git status (to check status of a code file).

```
[root@localhost abdulgit]# git status
On branch master
nothing to commit, working tree clean
[root@localhost abdulgit]#
```

➢ git log (it tells which commit is done by which user).

```
[root@localhost abdulgit]# git log
commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472 (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos
[root@localhost abdulgit]#
```

➢ git show <commit-id> (to see code of a particular commit).

```
[root@localhost abdulgit]# git show  c4bc63c3fda2ad0c
commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472 (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos

diff --git a/codefile b/codefile
new file mode 100644
index 0000000..d4c9c76
--- /dev/null
+++ b/codefile
@@ -0,0 +1 @@
+shaher me jin ki sakhawat ka bada charcha hai,
[root@localhost abdulgit]#
```

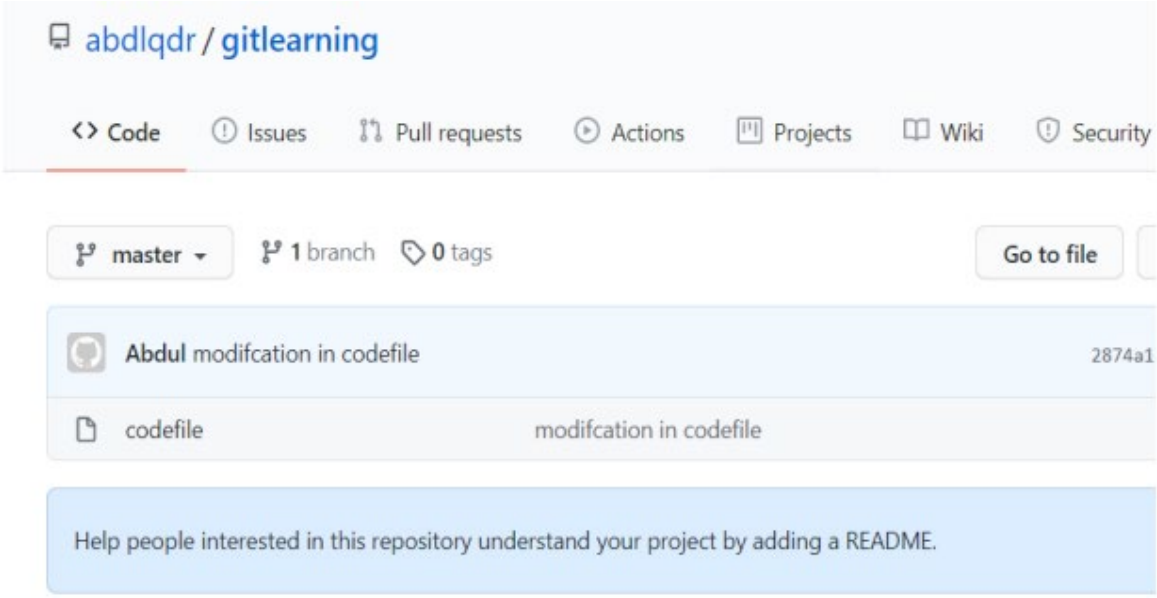
➢ git remote add origin <url> (to add github(central repo) with our local repo).

```
[root@localhost abdulgit]# git remote add origin https://github.com/abdlqdr/gitlearning.git
[root@localhost abdulgit]#
```

➢ git push –u origin master (to push code file from local repo to github in master branch).

```
[root@localhost abdulgit]# git push -u origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 256 bytes | 64.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/abdlqdr/gitlearning.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
[root@localhost abdulgit]#
```

You can see your file in github (central repo).



Now updating content in the previous file & checking status,

```
[root@localhost abdulgit]# cat >>codefile
mere tute hue kashkol se dar jate hai.
[root@localhost abdulgit]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   codefile

no changes added to commit (use "git add" and/or "git commit -a")
[root@localhost abdulgit]#
```

Now adding file to staging & checking status,

```
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   codefile

[root@localhost abdulgit]#
```

Now committing file to local repo & checking status.

```
[root@localhost abdulgit]# git commit -m "modifcation in codefile"
[master 2874a1c] modifcation in codefile
 1 file changed, 1 insertion(+)
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[root@localhost abdulgit]#
```

Now looking total git commits.

```
[root@localhost abdulgit]# git log
commit 2874a1ce855e66deb66fc799d919b395d3bac201 (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 12:15:54 2020 +0530

    modifcation in codefile

commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472 (origin/master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos
[root@localhost abdulgit]#
```

Now seeing the updated content in file.

```
[root@localhost abdulgit]# git show  2874a1ce855e66de
commit 2874a1ce855e66deb66fc799d919b395d3bac201 (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 12:15:54 2020 +0530

    modifcation in codefile

diff --git a/codefile b/codefile
index d4c9c76..08aace8 100644
--- a/codefile
+++ b/codefile
@@ -1 +1,2 @@
 shaher me jin ki sakhawat ka bada charcha hai,
+mere tute hue kashkol se dar jate hai.
[root@localhost abdulgit]#
```

Now pushing modification code file to github.

```
[root@localhost abdulgit]# git push -u origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Branch 'master' set up to track remote branch 'master' from 'origin'.
Everything up-to-date
[root@localhost abdulgit]#
```

Pulling Mechanism

Login into Linux machine.

Create a directory, go inside it & run ‘git init’ command to create local repo.

```
root@ubuntuserver:/home/abdlqdr# mkdir quadirgit
root@ubuntuserver:/home/abdlqdr# ls
quadirgit
```

Git remote add origin <github url> (to connect github with local repo).

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git remote add origin https://github.com/abdlqdr/gitlea
rning.git
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

git pull –u origin master/git pull origin master –allow

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git pull origin master --allow
From https://github.com/abdlqdr/gitlearning
 * branch            master     -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 2 ++
 codefile  | 2 ++
 2 files changed, 4 insertions(+)
 create mode 100644 README.md
 create mode 100644 codefile
```

git log (to show all commit)

add some code to codefile

```
root@ubuntuserver:/home/abdlqdr/quadirgit# cat >>codefile
khalike qauno makan daure fugha kam kar de
```

git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   codefile

no changes added to commit (use "git add" and/or "git commit -a")
```

git add . then git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   codefile
```

git commit –m “codefile modified” /then git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "codefile modified"
[master 2e899f1] codefile modified
 1 file changed, 1 insertion(+)
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
```

git push origin master

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git push origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 1.14 KiB | 61.00 KiB/s, done.
Total 11 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/abdlqdr/gitlearning.git
   5c4a4b2..2e899f1  master -> master
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

git ignore Command

➢ To ignore some files while committing.

➢ Create one hidden file “.gitignore” & enter file format which you want to

ignore.

```
[root@localhost abdulgit]# vi .gitignore
[root@localhost abdulgit]# ls
codefile  file1  README.md  ubuntufile
[root@localhost abdulgit]# ls -a
.  ..  codefile  file1  .git  .gitignore  README.md  ubuntufile
```

➢ Run git add .ignore & create some file using touch command.

```
[root@localhost abdulgit]# git add .gitignore
[root@localhost abdulgit]# touch file1.java file2.css file3.txt file4.py
[root@localhost abdulgit]# ls
codefile  file1  file1.java  file2.css  file3.txt  file4.py  README.md  ubuntufile
```

➢ Run git status, you will see it is showing only 2 files other 2 are ignored.

```
[root@localhost abdulgit]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file1.java
        file3.txt
```

➢ Now, you can commit these files as you need.

```
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   file1.java
        new file:   file3.txt

[root@localhost abdulgit]# git commit -m " java & text files"
[master 383e6f7]  java & text files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.java
 create mode 100644 file3.txt
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

➢ git log -1 (to show last & recent commit).

```
[root@localhost abdulgit]# git log -1
commit 383e6f7def8620a65942e44578ba5e2f2e8bdd4c (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Sat Sep 5 21:49:44 2020 +0530

    java & text files
```

➢ Git log -2 (to show last 2 commit)

```
[root@localhost abdulgit]# git log -2
commit 383e6f7def8620a65942e44578ba5e2f2e8bdd4c (HEAD -> master
Author: Abdul <abdul@gmail.com>
Date:   Sat Sep 5 21:49:44 2020 +0530

    java & text files

commit 6fb73bad27528cbd2cdd0a35425a603d191a1c4a
Author: Abdul <abdul@gmail.com>
Date:   Sat Sep 5 21:48:40 2020 +0530

    java & text files
```

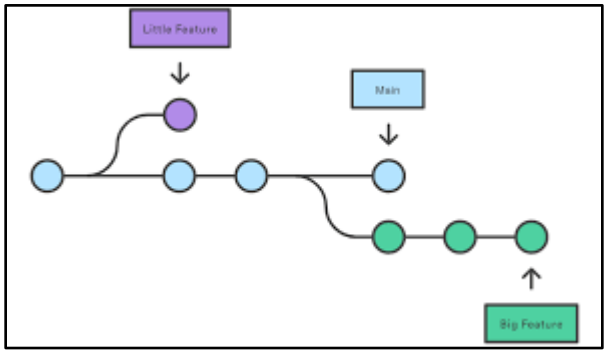
➢ git log –oneline (to display all commits in one line).

```
[root@localhost abdulgit]# git log --oneline
383e6f7 (HEAD -> master)  java & text files
6fb73ba  java & text files
c4a1e2e (origin/master) update by 5/sep
2e899f1 codefile modified
11f6f58 erge branch 'master' of https://github.com/abdlqdr/gitlearning
5c4a4b2 added read me| added string 'Hello'
8a95ee7 first commit from ubuntu
6fda145 first commit from ubuntu
2874a1c modifcation in codefile
c4bc63c first commit from centos
```

➢ git log –grep “commit” (it’s used to search a commit by a word).

```
[root@localhost abdulgit]# git log --grep "commit"
commit 8a95ee794356db9be55c6d91b8b41cddc137bbaf
Author: Quadir <quadir@gmail.com>
Date:   Sat Sep 5 13:55:49 2020 +0000

    first commit from ubuntu

commit 6fda1457bb669221b17519a84fac530e81642de8
Author: Quadir <quadir@gmail.com>
Date:   Fri Sep 4 19:06:53 2020 +0000

    first commit from ubuntu

commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos
```

Branch



The diagram above visualizes a repository with two isolated lines of development, one for a little feature, and one for a longer-running feature.

By developing them in branches, it's not only possible to work on both of them in parallel, but it also keeps the main branch frees from error. Each task has one separate branch. After done with code, merge other branches with master. This concept is useful for parallel development. You can create any no. of branches. Changes are personal to that particular branch. Default branch is 'master'. Files created in the workspace will be visible in any of the branch workspace till you commit. Once you commit, then that file belongs to that particular branch. When created new branch, data of existing branch is copied to new branch

Properties of Branches:-

➢ Each task has one separate branch.

➢ After done with coding, merge other branches with master.

➢ This concept is useful for parallel development.

➢ You can create any number of branches.

➢ Changes are personal to that particular branch .

➢ Default branch is 'master'.

➢ File created in workspace will be visible in any of the branch workspace until you commit once you commit, then that file belongs to that particular branch.

```
[root@localhost abdulgit]# git branch
  branch1
* master
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# cat >secondfile
hi
[root@localhost abdulgit]# ls
branchfile1  file1       file2.css  file4.py   secondfile
codefile     file1.java  file3.txt  README.md  ubuntufile
[root@localhost abdulgit]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
[root@localhost abdulgit]# ls
codefile  file1  file1.java  file2.css  file3.txt  file4.py  README.md  secondfile  ubuntufile
[root@localhost abdulgit]# git branch
  branch1
* master
```

```
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "2nd commit from branch1"
[branch1 81e4b59] 2nd commit from branch1
 1 file changed, 1 insertion(+)
 create mode 100644 secondfile
[root@localhost abdulgit]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
[root@localhost abdulgit]# ls
codefile  file1  file1.java  file2.css  file3.txt  file4.py  README.md  ubuntufile
```

➢ When created new branch, data of existing branch is copied to new branch.

```
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# ls
branchfile1  file1       file2.css  file4.py   secondfile
codefile     file1.java  file3.txt  README.md  ubuntufile
[root@localhost abdulgit]# git branch branchA
[root@localhost abdulgit]# git branch
* branch1
  branchA
  master
[root@localhost abdulgit]# git checkout branchA
Switched to branch 'branchA'
[root@localhost abdulgit]# ls
branchfile1  file1       file2.css  file4.py   secondfile
codefile     file1.java  file3.txt  README.md  ubuntufile
[root@localhost abdulgit]# git branch
  branch1
* branchA
  master
```

Commands of Branch:-

➢ To see list of available branches (git branch).

```
[root@localhost abdulgit]# git branch
  branch1
* master
```

➢ To create new branch (git branch <branch name>).

```
[root@localhost abdulgit]# git branch branch1
[root@localhost abdulgit]# git branch
  branch1
* master
```

➢ To switch branch. (git checkout <name of switching branch>). (*) shows

your current branch.

```
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# git branch
* branch1
  master
```

➢ To delete branch (git branch –d/-D <branch name>). [-D is used to delete

forcefully].

```
[root@localhost abdulgit]# git branch
  branch1
* master
[root@localhost abdulgit]# git branch -d branch1
Deleted branch branch1 (was 383e6f7).
[root@localhost abdulgit]# git branch
* master
```

Branch merge:-

You can’t merge branches of different repositories.

We use pulling mechanism to merge branches.

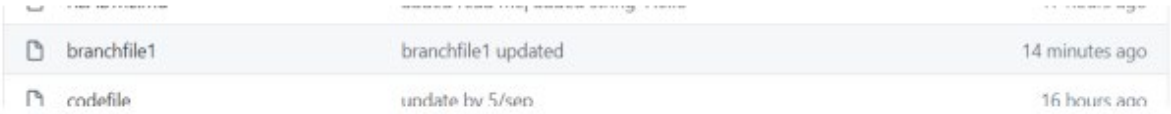
To merge branches (git merge <branch name>). Then run git log –oneline.

```
[root@localhost abdulgit]# git merge branch1
Updating 383e6f7..c2e2ba9
Fast-forward
 branchfile1 | 3 +++
 secondfile  | 1 +
 2 files changed, 4 insertions(+)
 create mode 100644 branchfile1
 create mode 100644 secondfile
[root@localhost abdulgit]# git log --oneline
c2e2ba9 (HEAD -> master, branch1) branchfile1 updated
81e4b59 2nd commit from branch1
10dcfae branch1 commit
383e6f7  java & text files
6fb73ba  java & text files
c4a1e2e (origin/master) update by 5/sep
2e899f1 codefile modified
11f6f58 erge branch 'master' of https://github.com/abdlqdr/gitlearning
5c4a4b2 added read me| added string 'Hello'
8a95ee7 first commit from ubuntu
6fda145 first commit from ubuntu
2874a1c modifcation in codefile
c4bc63c first commit from centos
```

Push the commits to github(central repo). (git push origin master)

```
[root@localhost abdulgit]# git push origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 2 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 1.24 KiB | 105.00 KiB/s, done.
Total 15 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/abdlqdr/gitlearning.git
   c4a1e2e..c2e2ba9  master -> master
```

Check github to see the changes.



git conflict:-

➢ When same file having different content in different branch if you do

branch merge, conflict occurs (resolve conflict then add & commit).

➢ Conflict occurs when you merge branch.

Ex 1:

Branch:Master branch:Branch1

File name:abdl File name:abdl

Content:My name is abdul. Content: My name is abdul.

I live in azamgarh.

➢ When you merge above branch then there is no conflict because there is same line available in both files. It'll know the second line to merge.

Ex 2:

Branch:Master branch:Branch1

File name:abdl File name:abdl

Content:My name is abdul. Content: I live in azamgarh.

➢ In above example, when we merge branches git will confuse which data to add above or below. When data is completely different then conflict occurs.

```
[root@localhost abdulgit]# cat >abdl
hello abdl
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "first commit conflict
[master 8f4d34e] first commit conflict
 1 file changed, 1 insertion(+)
 create mode 100644 abdl
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# cat >abdl
hi abdl
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "conflict branch1"
[branch1 82cba1c] conflict branch1
 1 file changed, 1 insertion(+)
 create mode 100644 abdl
[root@localhost abdulgit]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

➢ To correct conflict, In master branch open file in vi editor(vi abdl) & edit as you want & save this file, git understand that & merge file.

```
[root@localhost abdulgit]# git merge branch1
Auto-merging abdl
CONFLICT (add/add): Merge conflict in abdl
Automatic merge failed; fix conflicts and then commit the result.
[root@localhost abdulgit]# vi abdl
```

➢ After that you have to add & commit that merged file in master branch.

```
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both added:      abdl

no changes added to commit (use "git add" and/or "git commit -a")
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "conflict resolved"
[master 471fd6a] conflict resolved
[root@localhost abdulgit]# git log --oneline
471fd6a (HEAD -> master) conflict resolved
82cba1c (branch1) conflict branch1
8f4d34e first commit conflict
c2e2ba9 (origin/master) branchfile1 updated
```

git Stashing:-

➢ Suppose, you are implementing a new feature for your product. Your code is in progress & suddenly a customer escalation comes because of this, you have to keep aside your new feature work for a few hours. You can't commit your partial code & also can't throw away your changes. So, you need some temporary storage, when you can store your partial changes & later on commit it.

➢ To stash an item (only applies to modified files not new files).

git stash

```
root@ubuntuserver:/home/abdlqdr/quadirgit# touch demofile
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "demofile"
[master d14a9b3] demofile
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demofile
root@ubuntuserver:/home/abdlqdr/quadirgit# vi demofile
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash
Saved working directory and index state WIP on master: d14a9b3 demofile
```

➢ To see stashed item list

git stash list

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash list
stash@{0}: WIP on master: d14a9b3 demofile
stash@{1}: WIP on master: 24f8dda practice conflict resolved
```

➢ To apply stashed items(working file will move to workspace).

git stash apply stash stash@{1}

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash apply stash@{1}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demofile

no changes added to commit (use "git add" and/or "git commit -a")
```

➢ Then you can add & commit

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "second code done"
[master 2d3e81c] second code done
 1 file changed, 2 insertions(+), 1 deletion(-)
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
root@ubuntuserver:/home/abdlqdr/quadirgit# git log --oneline
2d3e81c (HEAD -> master) second code done
293ec7f first code done
d14a9b3 demofile
24f8dda (origin/master) practice conflict resolved
```

➢ To clear the stash items

git stash clear

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash clear
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash list
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

➢ When you get back your data from stash to workspace. Then that

data will be also there in stashing area.

git reset:-

git reset is a powerful command undo local changes to the state of a git

repository.

To reset staging area.

git reset <file name>

```
root@ubuntuserver:/home/abdlqdr/quadirgit# ls
abdl          codefile  file1       file3.txt  secondfile  ubuntufile
branchfile1   demofile  file1.java  README.md  testfile    xyz
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   testfile

root@ubuntuserver:/home/abdlqdr/quadirgit# git reset testfile
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        testfile

nothing added to commit but untracked files present (use "git add" to track)
```

git reset .

```
root@ubuntuserver:/home/abdlqdr/quadirgit# cat >testfile
my name is khan.
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   testfile

root@ubuntuserver:/home/abdlqdr/quadirgit# git reset .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        testfile

nothing added to commit but untracked files present (use "git add" to track)
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

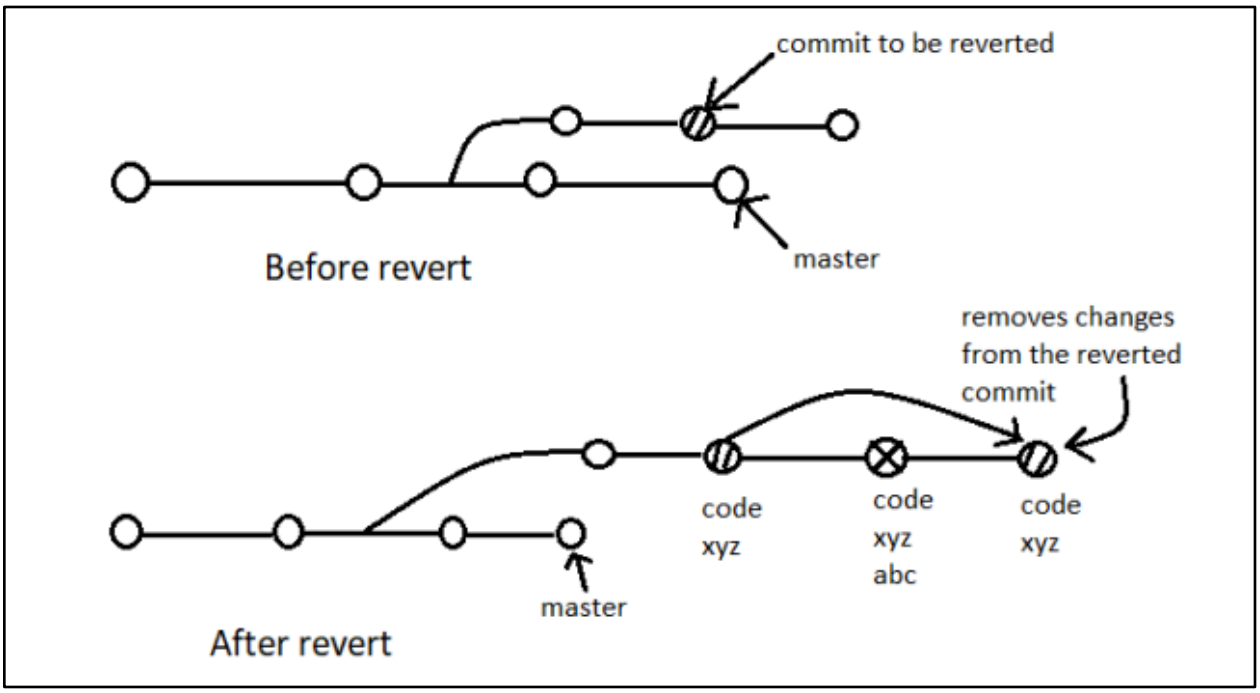
To reset the changes from both staging area & working directory at a time.

git reset –hard

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   testfile

root@ubuntuserver:/home/abdlqdr/quadirgit# git reset --hard
HEAD is now at 2d3e81c second code done
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

git revert:-

➢ Revert command helps you undo an existing commit. ➢ It doesn't delete any data in the process instead rather git creates new commit with the included files reverted to their previous state. So, your version control history moves forward while the state of your file moves backward



➢ When you revert a commit, a commit id is assigned to reverted commit.

Command for git revert:-

git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
```

cat >newfile

```
root@ubuntuserver:/home/abdlqdr/quadirgit# cat >newfile
guftgu tune sikhai hai ke mai gunga tha
```

git add .

git commit –m “code”

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "checking revert"
[master ff72eb1] checking revert
 1 file changed, 1 insertion(+)
 create mode 100644 newfile
```

git log –oneline

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git log --oneline
ff72eb1 (HEAD -> master) checking revert
f9b75e4 (origin/master) code file complete
f939e20 code file stashing test
2d3e81c second code done
293ec7f first code done
```

git revert <commit id>

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git revert  c720124c89cf496a
[master 2ed04fc] Revert "checking revert update" plese consider this code.
 1 file changed, 1 deletion(-)
```

How to remove untracked files?

git clean –n (dry run)

git clean –f (forcefully)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# touch file1 file2 file3 file4
root@ubuntuserver:/home/abdlqdr/quadirgit# ls
abdl         codefile  file1       file2  file3.txt  newfile    secondfile  xyz
branchfile1  demofile  file1.java  file3  file4      README.md  ubuntufile
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2
        file3
        file4

nothing added to commit but untracked files present (use "git add" to track)
root@ubuntuserver:/home/abdlqdr/quadirgit# git clean -n
Would remove file2
Would remove file3
Would remove file4
root@ubuntuserver:/home/abdlqdr/quadirgit# git clean -f
Removing file2
Removing file3
Removing file4
root@ubuntuserver:/home/abdlqdr/quadirgit# ls
abdl         codefile  file1       file3.txt  README.md   ubuntufile
branchfile1  demofile  file1.java  newfile    secondfile  xyz
```

Tag in git:-

Tag operation allows you giving meaningful names to a specific version in the repository.

To apply tag.

git tag –a <tag name> -m <message> <commit id>

git tag (to see tags)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag -a important -m "imp commit" c720124
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag
important
```

git show <tag name> (to see particular commit content by using tag)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git show important
tag important
Tagger: Quadir <quadir@gmail.com>
Date:   Mon Sep 7 07:25:23 2020 +0000

imp commit

commit c720124c89cf496aa94b4a48fdce0886c363bba3 (tag: important)
Author: Quadir <quadir@gmail.com>
Date:   Mon Sep 7 06:46:01 2020 +0000

    checking revert update

diff --git a/newfile b/newfile
index c84dba5..0f3dcc0 100644
--- a/newfile
+++ b/newfile
@@ -1 +1,2 @@
 guftgu tune sikhai hai ke mai gunga tha
+jab mai bolu to baton me asar bhi dena.
```

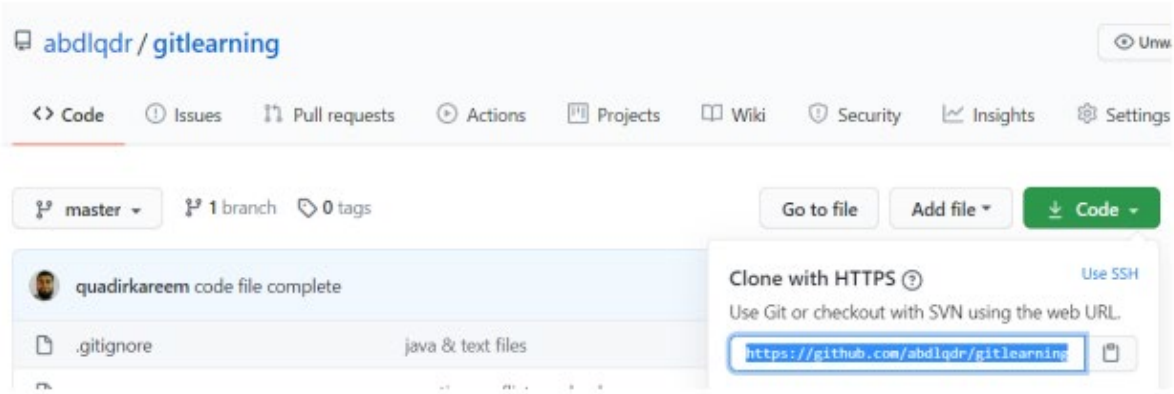
git tag –d <tag name> (to delete tag)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag -d important
Deleted tag 'important' (was 747fa38)
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag
```

Github Clone:-

➢ Open github website.

➢ Login & choose existing repository, copy github url.



➢ Now, go to your linux machine, & run command

git clone <url of github repo> (as you can see git learning repo folder)

```
[root@localhost abdlqdr]# git clone https://github.com/abdlqdr/gitlearning.git
Cloning into 'gitlearning'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 83 (delta 27), reused 76 (delta 23), pack-reused 0
Unpacking objects: 100% (83/83), done.
[root@localhost abdlqdr]# ls
abdulgit  gitlearning
[root@localhost abdlqdr]#
```

➢ It creates a local repo automatically in linux machine with the same name

as in github account. As you can see above.

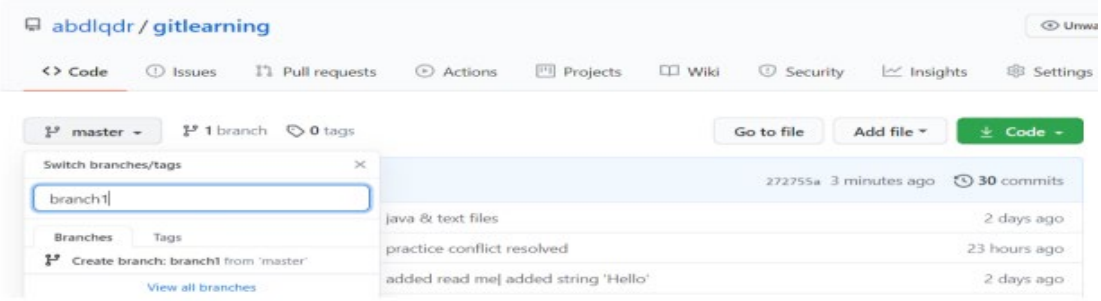
```
[root@localhost abdlqdr]# cd gitlearning
[root@localhost gitlearning]# ls
abdl         codefile  file1       file3.txt  README.md   ubuntufile
branchfile1  demofile  file1.java  newfile    secondfile  xyz
[root@localhost gitlearning]#
```

Github Features:-

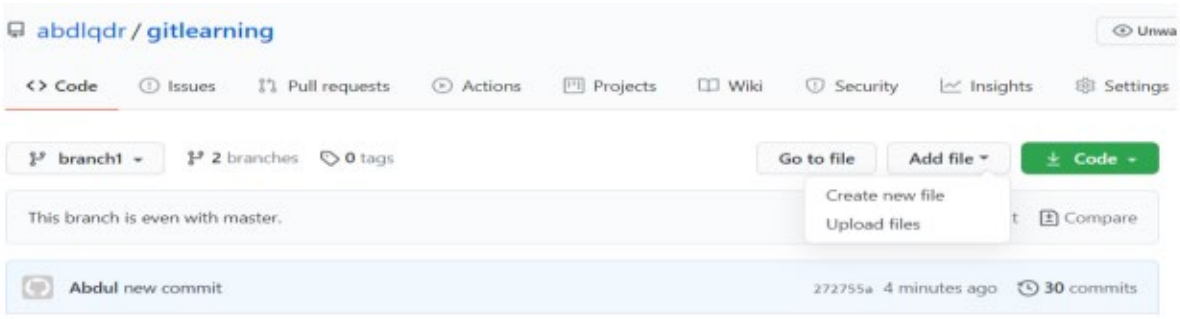
You can make branches in github exactly as you make in git. It's nature & process

is also the same as git, like master branch, creation of branches from master branch.

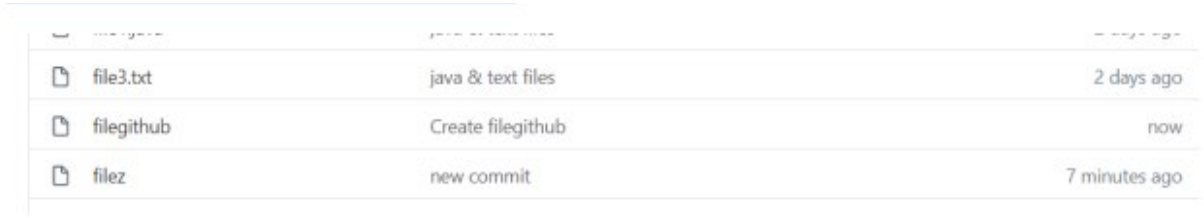
Pulling request is used to see conflict of files when you merge a branch.



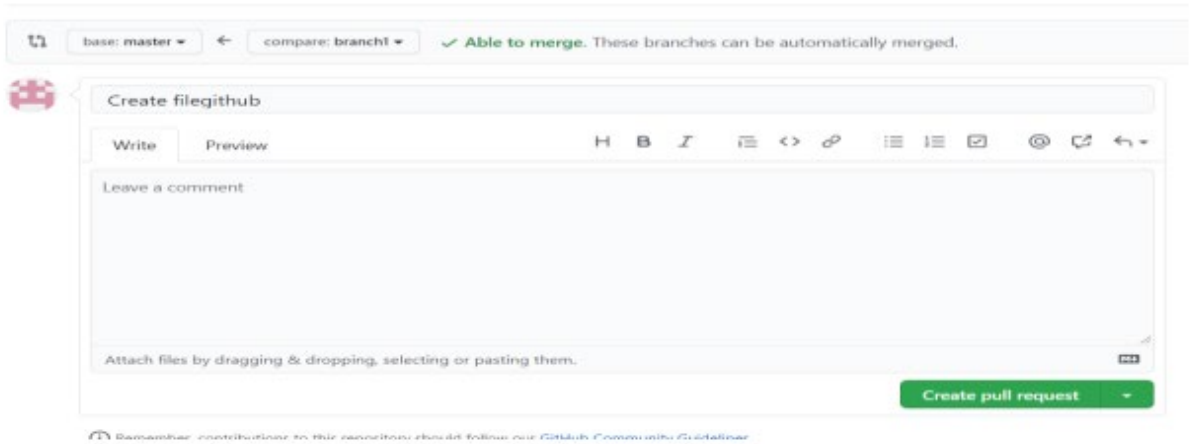
To create file in branch1



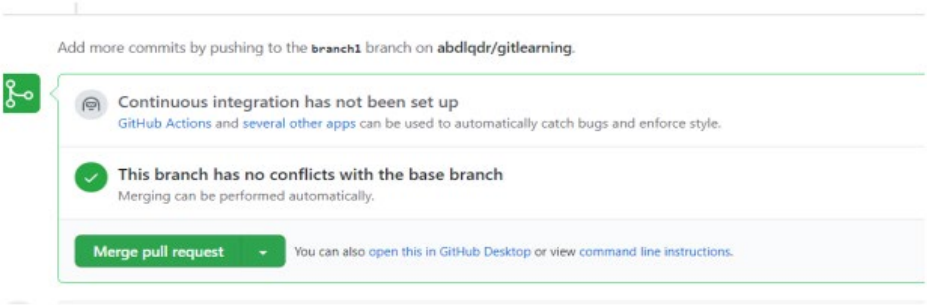
Committed file by github to github.



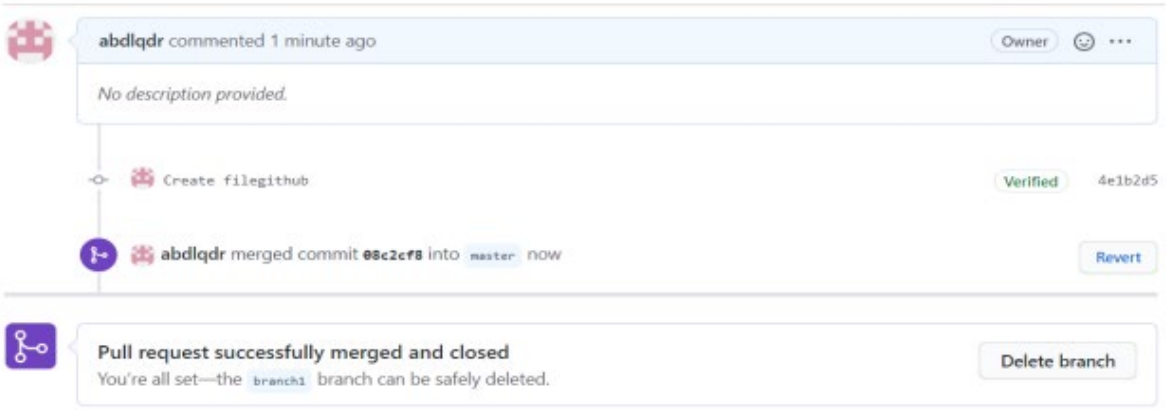
Merging branch1



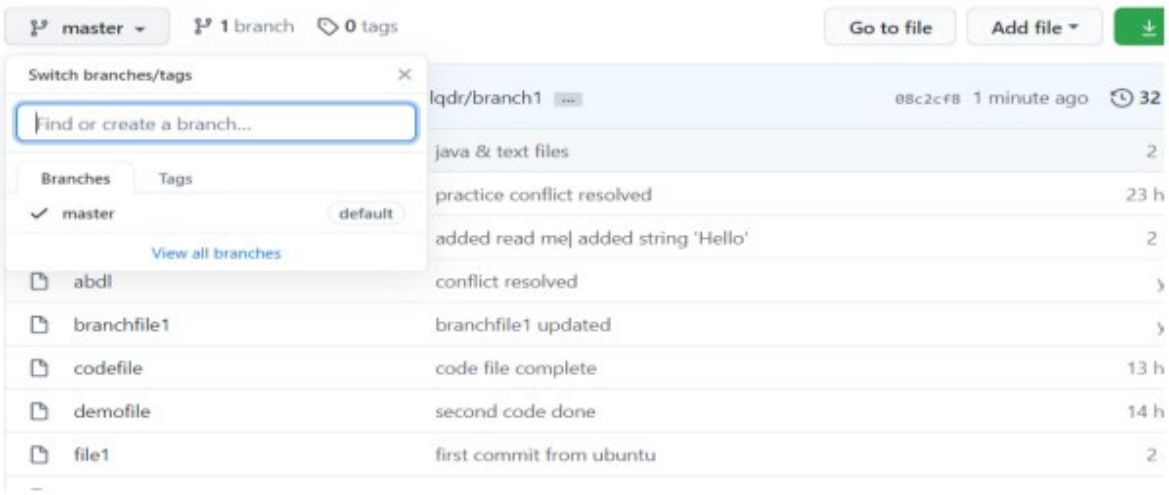
To merge branches.



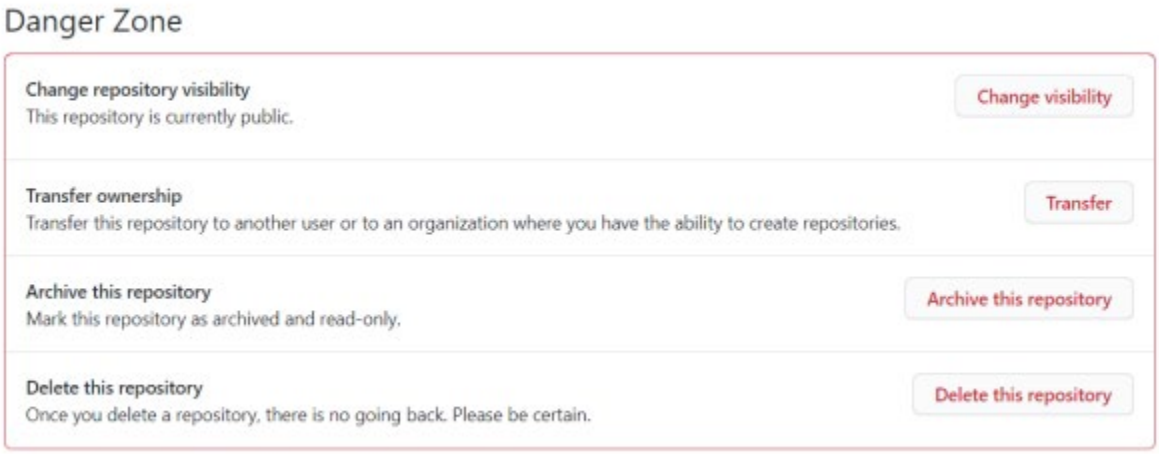
Click delete branch to delete branch



Only master branch is showing



To delete the total repository go to settings & scroll down & click delete repository.



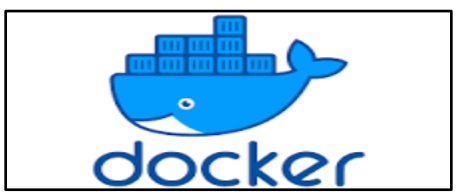
git checkout	**git reset**	**git revert**
Can be used to manipulate commits or files.	Can be used to manipulate commits or files.	Does not manipulate your commits or files.

Discards the changes in the working repository.	Unstages a file and bring our changes back to the working directory.	Removes the commits from the remote repository.
Does not make any changes to the commit history.	Alters the existing commit history,	Adds a new commit to the existing commit history.
Moves HEAD pointer to a specific commit.	Discards the uncommitted changes.	Rollbacks the changes which we have committed.
Used in the local repository.	Used in a local repository.	Used in the remote repository.

Docker

Docker is a container management service. The keywords of Docker are develop, ship and run anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.



Features of Docker

- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.

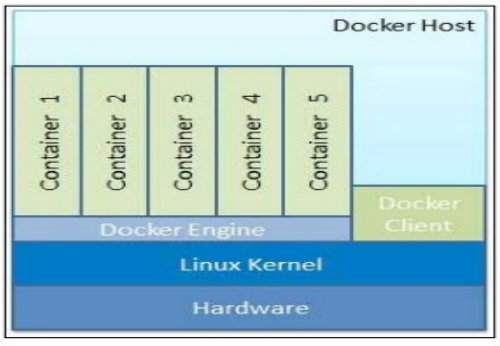
Components of Docker

Docker has the following components

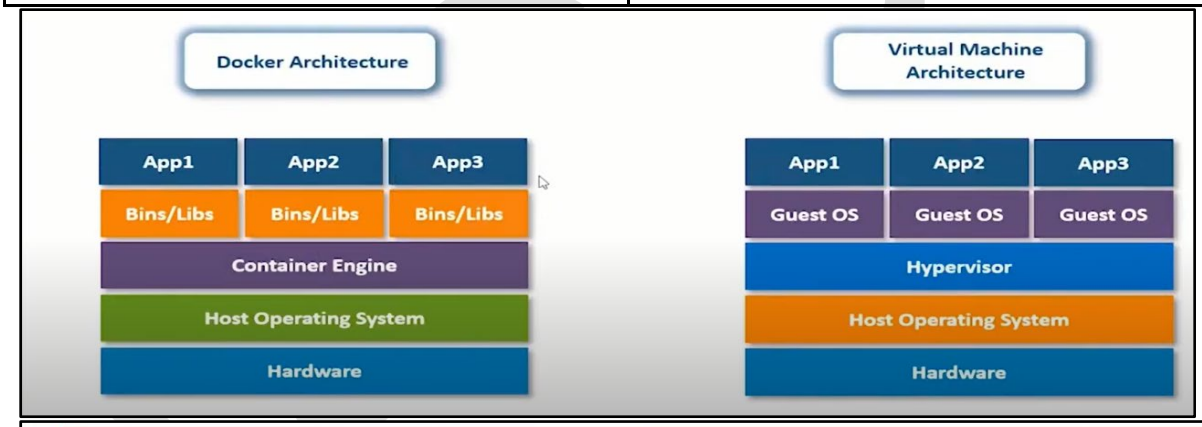
- Docker for Mac − It allows one to run Docker containers on the Mac OS.
- Docker for Linux − It allows one to run Docker containers on the Linux OS.
- Docker for Windows − It allows one to run Docker containers on the Windows OS.
- Docker Engine − It is used for building Docker images and creating Docker containers.
- Docker Hub − this is the registry which is used to host various Docker images.
- Docker Compose − this is used to define applications using multiple Docker containers.

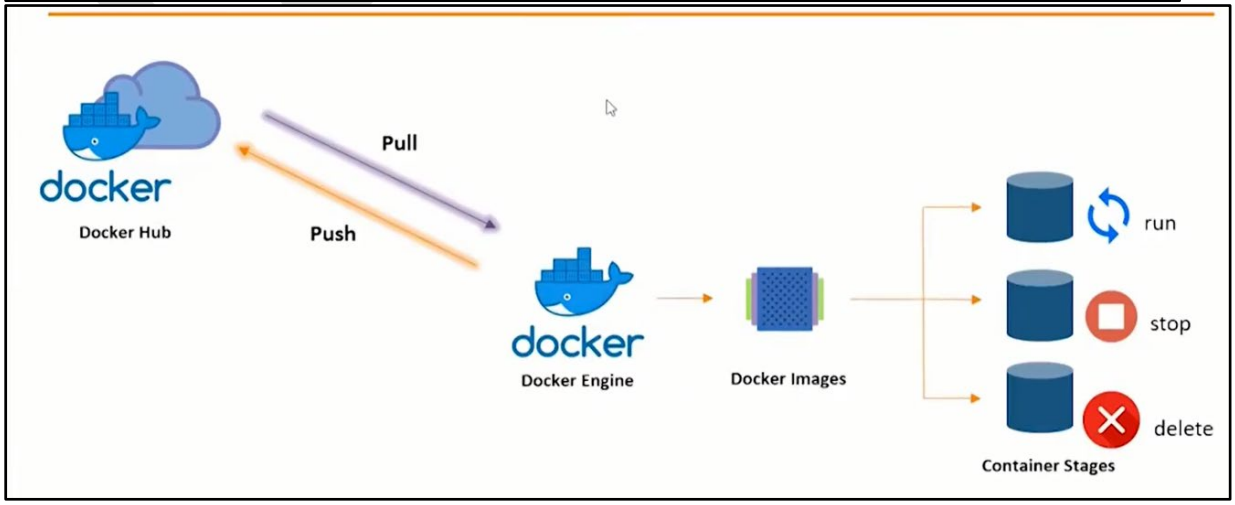
Docker on Linux

Suppose that we want to directly run the containers on a Linux machine. The Docker engine produces, monitors, and manages multiple containers as illustrated in the following diagram:



Docker	VM
Virtualize Application layer	Virtualize Application & OS layer
Uses Host OS kernel	Uses guest OS kernel
Docker image much smaller	Size is more
Docker containers start & run very fast	Comparatively slow
Light weight	Heavy weight
Native performance	Limited performance
Process level isolation hence less secure	Fully isolated hence more secure





Docker container lifecycle/ Docker ecosystem

Docker is ecosystem i.e. it is a set of s/w or packages

Docker hub

- It is registry for container which provide all dependencies
- Container does not install any OS it take from docker hub & then simply run it
- Container does not have OS but it have software

- Docker is an open source centralized platform designed to create, deploy & run applications
- Docker uses container on the host OS to run applications, it allows applications to use same Linux kernel as a system on the host computer rather than creating a whole virtual OS
- We can install on any os but docker engine runs natively on linux distribution
- Docker written in 'go' language
- Docker is a tool that performs OS level virtualization also known as containerization
- Before docker many users faces the problem that particular code is running in the developer's system but not in the user's system
- Docker was first released in march 2013, it is developed by Solomon Hukes Sebastian Pahl
- Docker is a set of platform as a service that uses os level virtualization whereas vmware uses hardware level virtualization

Advantages of docker

- No pre allocation of RAM
- Light weight
- Less cost
- It took very less time to create container
- Continuous integration efficiency → docker enables you to build a container image & use that same image across every step of deployment process
- It can run on physical h/w, virtual h/w or on cloud
- You can reuse image

Disadvantages of docker

- It is not a good solution for application that require rich GUI
- Difficult to manage large amount of containers
- Docker does not support cross platform compatibility means if an application is designed to run in a docker container on windows, then it can't run on linux or vice versa
- Docker is suitable when development OS & testing OS are same, if the os is different we should use vm
- No solution for data recovery & backup

Docker client → where we write code, write docker file

Docker engine → it makes images, converts images to containers. Execute commands

Docker hub → stores all images

Docker image → it is like template

Docker compose → runs multiple docker containers

Docker daemon

- Runs on host os
- It is responsible for running containers to manage docker services
- It can communicate with other daemons

Docker client

- Docker user can interact with docker daemon through a client (CLI)
- It uses commands & rest API to communicate with docker daemon
- When a client runs any server command on docker client terminal, client terminal sends these docker commands to docker daemon
- It is possible for docker client to communicate with more than one daemon

Docker host

- It is used to provide an environment to execute & run application
- It contains docker daemon, images, containers, n/w & storage

Docker hub registry

- Manages &v stores docker images

- Two types of registry in docker
- 1. Public registry: also called docker hub
- 2. Private registry: used to share images within the enterprise/company

Docker images

- Read only binary templates used to create docker containers
- Single file with all dependencies or & configuration requires to run a program/ container

Docker container

- It hold entire packages i.e. needed to run the application or
- We can say image is a template & container is a copy of that template
- Container is like VM
- Images cannot be modified, containers can be modified
- It id portable artifact, easily shared & moved around between development teams or development & operations teams
- Makes development & deployment efficient
- Containers lives in container repository
- Container is layers of images mostly linux base image, because small in size
- Application image on top

Before container	**After container**
Installation process different on each os environment	Own isolated environment
Many steps where something could go wrong	Packaged with all needed configuration
	One command to install app
	Run same app with two different versions

Ways to create images

1. Take image from docker hub
2. Create image from docker file
3. Create image from existing docker containers

Basic commands in docker

- To install docker

  # sudo apt-get install docker.io

- To see version of docker
  - # docker –version

```
root@ubuntu:~# docker version
Client:
 Version:           18.09.5
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        e8ff056
 Built:             Thu Apr 11 04:43:57 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.5
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.8
  Git commit:       e8ff056
  Built:            Thu Apr 11 04:10:53 2019
  OS/Arch:          linux/amd64
  Experimental:     false
```

- To see more information on the Docker running on the system
  - # docker info

The output will provide the various details of the Docker installed on the system such as

• Number of containers

• Number of images

• The storage driver used by Docker

• The root directory used by Docker

• The execution driver used by Docker

```
linuxtechi@docker:~$ sudo docker info
[sudo] password for linuxtechi:
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 1.12.5
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 0
 Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: overlay bridge null host
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Security Options: apparmor seccomp
Kernel Version: 4.4.0-21-generic
Operating System: Ubuntu 16.04 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 3.121 GiB
Name: docker
ID: QUPF:QBRQ:PRAV:ZOVK:OEPJ:PIUG:GFJO:ZZWE:VPDC:6NSE:IYGV:5QAS
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
WARNING: No swap limit support
Insecure Registries:
 127.0.0.0/8
linuxtechi@docker:~$
```

- To see all images present in your local machine

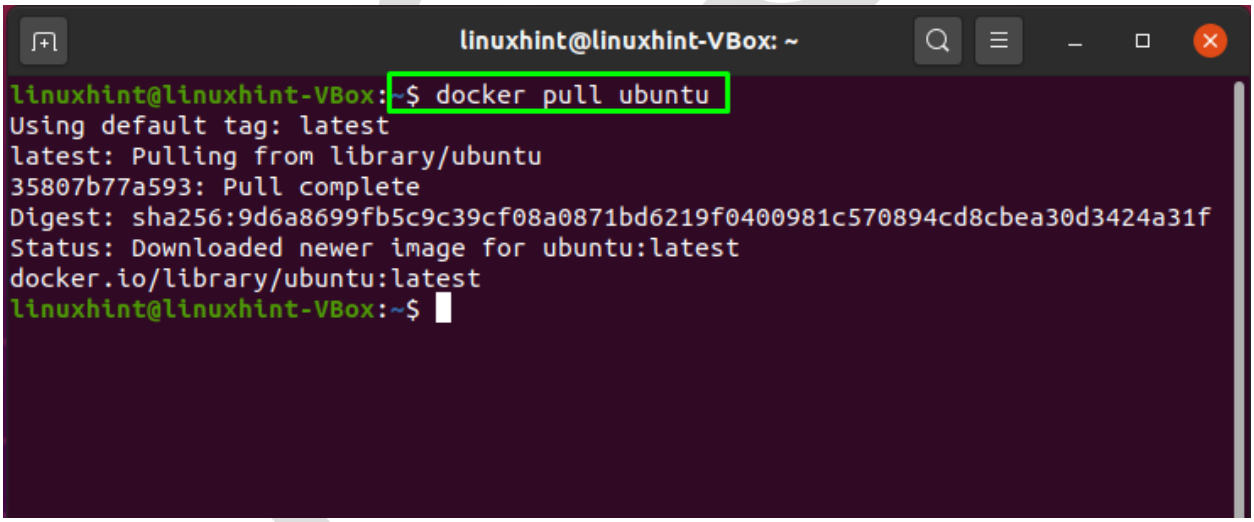
```
tim@nova:/home/../local/tmp/ % docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
 STATUS              PORTS               NAMES
946501d04a32        mysql               "/entrypoint.sh mysql"   9 hours ago
 Up 9 hours          3306/tcp            phpmyadmin-mysql
tim@nova:/home/../local/tmp/ % docker images
REPOSITORY                      TAG                 IMAGE ID            CREATED
    VIRTUAL SIZE
mysql                           latest              6762f304c834        2 weeks ago
    283.5 MB
corbinu/docker-phpmyadmin       latest              9fe36d18c039        11 weeks ago
    417.8 MB
ubuntu                          14.04               d2a0ecffe6fa        11 weeks ago
    188.4 MB
ubuntu                          latest              d2a0ecffe6fa        11 weeks ago
    188.4 MB
```

docker images

- To find out docker images in docker hub
    - # docker search <image name>
    - # docker search ubuntu
    -

```
bosko@pnap:~$ sudo docker search ubuntu
NAME                      DESCRIPTION
 STARS     OFFICIAL   AUTOMATED
ubuntu                    Ubuntu is a Debian-based Linux operating sys…
 15005     [OK]
websphere-liberty         WebSphere Liberty multi-architecture images …
 289       [OK]
ubuntu-upstart            DEPRECATED, as is Upstart (find other proces…
 112       [OK]
neurodebian               NeuroDebian provides neuroscience research s…
 93        [OK]
ubuntu/nginx              Nginx, a high-performance reverse proxy & we…
 61
open-liberty              Open Liberty multi-architecture images based…
 55        [OK]
```

- To pull images from central docker repository/ download image from docker hub to local machine
    - # docker pull <image -name>
        - docker pull ubuntu



- To see all container status

    # docker ps -a …… it will display container id,status,image etc

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND      CREATED         STATUS                     PORTS                                  NAMES
6b29d9c35dfd   ubuntu    "bash"       6 minutes ago   Up 6 minutes               0.0.0.0:80→80/tcp, :::80→80/tcp        pedantic_feynman
12e6146764b3   ubuntu    "/bin/bash"  14 minutes ago  Exited (1) 7 minutes ago                                          goofy_galileo
```

- To run container from image name
    - # docker run <image name> or <container id>

docker run -it ubuntu or <container ID>

docker run -it -d ubuntu or <container ID>

(i for interactive t for terminal d for Run container in background and print container ID, -d, --detach Run container in background and print container ID)

Run a command in a new container

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED         STATUS         PORTS     NAMES
20df81a5e610   ubuntu    "bash"    7 minutes ago   Up 7 minutes             blissful_lamport

┌──(kali㉿kali)-[~]
└─$ sudo docker run -it ubuntu
root@6c4a96caad1a:/# exit
exit

┌──(kali㉿kali)-[~]
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    27 seconds ago   Exited (0) 6 seconds ago              crazy_noether
20df81a5e610   ubuntu    "bash"    7 minutes ago    Up 7 minutes                          blissful_lamport

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    3 minutes ago    Up 2 minutes              crazy_noether
20df81a5e610   ubuntu    "bash"    10 minutes ago   Up 10 minutes             blissful_lamport

┌──(kali㉿kali)-[~]
└─$ sudo docker exec -it 20df81a5e610 bash
root@20df81a5e610:/# exit
exit

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    4 minutes ago    Up 3 minutes              crazy_noether
20df81a5e610   ubuntu    "bash"    11 minutes ago   Up 11 minutes             blissful_lamport
```

- To start/stop container

docker start <container id>

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                     PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    27 seconds ago   Exited (0) 6 seconds ago             crazy_noether
20df81a5e610   ubuntu    "bash"    7 minutes ago    Up 7 minutes                         blissful_lamport

┌──(kali㉿kali)-[~]
└─$ sudo docker start 6c4a96caad1a
6c4a96caad1a

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    47 seconds ago   Up 2 seconds              crazy_noether
20df81a5e610   ubuntu    "bash"    8 minutes ago    Up 8 minutes              blissful_lamport
```

docker stop <container id>

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
20df81a5e610   ubuntu    "bash"    2 minutes ago    Up 2 minutes              blissful_lamport
e96b45ff8a86   ubuntu    "bash"    5 minutes ago    Up 5 minutes              brave_euclid

┌──(kali㉿kali)-[~]
└─$ sudo docker stop e96b45ff8a86
e96b45ff8a86

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS     NAMES
20df81a5e610   ubuntu    "bash"    2 minutes ago    Up 2 minutes                          blissful_lamport
e96b45ff8a86   ubuntu    "bash"    6 minutes ago    Exited (137) 3 seconds ago            brave_euclid
```

- To stop all containers
    - #docker stop $(docker ps -a -q)

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
[sudo] password for kali:
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS                     PORTS                                   NAMES
f999b31d78d    new_dockerfile       "/bin/sh -c 'apachec…"   3 hours ago    Up 3 hours                 0.0.0.0:84->80/tcp, :::84->80/tcp       xenodochial_kirch
a08603c0f2e    new_dockerfile       "/bin/sh -c 'apachec…"   22 hours ago   Exited (137) 21 hours ago                                  nice_kowalevski
9fc3806107d    seemanaaz/apache     "bash"                   23 hours ago   Exited (137) 21 hours ago                                  seema

┌──(kali㉿kali)-[~]
└─$ sudo docker stop $(sudo docker ps -a -q)
f999b31d78d
a08603c0f2e
9fc3806107d

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS                      PORTS     NAMES
f999b31d78d    new_dockerfile       "/bin/sh -c 'apachec…"   3 hours ago    Exited (137) 2 seconds ago            xenodochial_kirch
a08603c0f2e    new_dockerfile       "/bin/sh -c 'apachec…"   22 hours ago   Exited (137) 21 hours ago             nice_kowalevski
9fc3806107d    seemanaaz/apache     "bash"                   23 hours ago   Exited (137) 21 hours ago             seema
```

- To go inside container

    # docker exec -it <container id> bash

Run a command in a running container

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    8 minutes ago    Up 7 minutes              crazy_noether
20df81a5e610   ubuntu    "bash"    15 minutes ago   Up 15 minutes             blissful_lamport

┌──(kali㉿kali)-[~]
└─$ sudo docker exec -it 20df81a5e610 bash
root@20df81a5e610:/# exit
exit

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
6c4a96caad1a   ubuntu    "bash"    8 minutes ago    Up 7 minutes              crazy_noether
20df81a5e610   ubuntu    "bash"    15 minutes ago   Up 15 minutes             blissful_lamport
```

- To rename a container

  #docker rename CONTAINER NEW_NAME

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                      PORTS                                  NAMES
af999b31d78d   new_dockerfile    "/bin/sh -c 'apachec…"   25 minutes ago   Up 25 minutes               0.0.0.0:84->80/tcp, :::84->80/tcp      xenodochial_kirch
ca08603c0f2e   new_dockerfile    "/bin/sh -c 'apachec…"   19 hours ago     Exited (137) 19 hours ago                                          nice_kowalevski
a9fc3806107d   seemanaaz/apache  "bash"                   20 hours ago     Exited (137) 19 hours ago                                          keen_burnell

┌──(kali㉿kali)-[~]
└─$ sudo docker rename keen_burnell seema

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                      PORTS                                  NAMES
af999b31d78d   new_dockerfile    "/bin/sh -c 'apachec…"   27 minutes ago   Up 27 minutes               0.0.0.0:84->80/tcp, :::84->80/tcp      xenodochial_kirch
ca08603c0f2e   new_dockerfile    "/bin/sh -c 'apachec…"   19 hours ago     Exited (137) 19 hours ago                                          nice_kowalevski
a9fc3806107d   seemanaaz/apache  "bash"                   20 hours ago     Exited (137) 19 hours ago                                          seema
```

- To rename a docker image

  #docker tag oldname newname

```
┌──(kali㉿kali)-[~]
└─$ sudo docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
new_dockerfile      latest    3ee83067355a   19 hours ago     224MB
seemanaaz/apache    latest    6d7fcf55a067   20 hours ago     224MB
seematest           latest    01f994d5b08f   20 hours ago     77.8MB
seema               latest    a07d22570929   20 hours ago     77.8MB
ubuntu              latest    216c552ea5ba   7 days ago       77.8MB
```

```
┌──(kali㉿kali)-[~]
└─$ sudo docker tag new_dockerfile seemanaaz/new_dockerfile

┌──(kali㉿kali)-[~]
└─$ sudo docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
new_dockerfile              latest    3ee83067355a   19 hours ago     224MB
seemanaaz/new_dockerfile    latest    3ee83067355a   19 hours ago     224MB
seemanaaz/apache            latest    6d7fcf55a067   20 hours ago     224MB
seematest                   latest    01f994d5b08f   20 hours ago     77.8MB
seema                       latest    a07d22570929   20 hours ago     77.8MB
ubuntu                      latest    216c552ea5ba   7 days ago       77.8MB
```

- To delete a container

  # docker rm <container id>

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS                          PORTS     NAMES
20df81a5e610   ubuntu    "bash"    About a minute ago   Up About a minute                         blissful_lamport
8ced14475976   ubuntu    "bash"    2 minutes ago        Exited (0) About a minute ago             great_khayyam
2b3ae7ddaed9   ubuntu    "bash"    2 minutes ago        Exited (0) 2 minutes ago                  wizardly_mendel
e96b45ff8a86   ubuntu    "bash"    5 minutes ago        Up 5 minutes                              brave_euclid
32c1c2e71955   ubuntu    "bash"    5 minutes ago        Exited (0) 5 minutes ago                  nostalgic_panini

┌──(kali㉿kali)-[~]
└─$ sudo docker rm 32c1c2e71955
32c1c2e71955

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS                          PORTS     NAMES
20df81a5e610   ubuntu    "bash"    About a minute ago   Up About a minute                         blissful_lamport
8ced14475976   ubuntu    "bash"    2 minutes ago        Exited (0) About a minute ago             great_khayyam
2b3ae7ddaed9   ubuntu    "bash"    2 minutes ago        Exited (0) 2 minutes ago                  wizardly_mendel
e96b45ff8a86   ubuntu    "bash"    5 minutes ago        Up 5 minutes                              brave_euclid
```

- To delete running container
  - #docker rm -f <container id>

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS              PORTS     NAMES
e0a60bccb049   ubuntu    "bash"    About a minute ago   Up About a minute             gifted_galois

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS                      PORTS     NAMES
e0a60bccb049   ubuntu    "bash"    About a minute ago   Up About a minute                     gifted_galois
20df81a5e610   ubuntu    "bash"    3 hours ago          Exited (137) 31 minutes ago           blissful_lamport

┌──(kali㉿kali)-[~]
└─$ sudo docker rm e0a60bccb049
Error response from daemon: You cannot remove a running container e0a60bccb0496b7bd5d4f7649ef8dde1f780efe9155250c6b34795fffc2ce8b1. Stop the container before attemptin
g removal or force remove

┌──(kali㉿kali)-[~]
└─$ sudo docker rm -f e0a60bccb049
e0a60bccb049

┌──(kali㉿kali)-[~]
└─$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS     NAMES
20df81a5e610   ubuntu    "bash"    3 hours ago      Exited (137) 32 minutes ago           blissful_lamport
```

- To delete all containers at time

- #docker rm -f $(sudo docker ps -a -q)

```
┌──(kali㉿kali)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND   CREATED          STATUS          PORTS     NAMES
1ba0142c063d   seematest   "bash"    9 minutes ago    Up 9 minutes              thirsty_goldstine
a36bcda6a500   ubuntu      "bash"    12 minutes ago   Up 12 minutes             eager_bartik

┌──(kali㉿kali)-[~]
└─$ sudo docker rm -f $(sudo docker ps -a -q)
1ba0142c063d
a36bcda6a500
9c765e923f4c
20df81a5e610

┌──(kali㉿kali)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

- To delete/remove an image

  #docker rmi <image -id>

```
┌──(kali㉿kali)-[~]
└─$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    216c552ea5ba   6 days ago   77.8MB

┌──(kali㉿kali)-[~]
└─$ sudo docker pull debian
Using default tag: latest
latest: Pulling from library/debian
f606d8928ed3: Pull complete
Digest: sha256:e538a2f0566efc44db21503277c7312a142f4d0dedc5d2886932b92626104bff
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest

┌──(kali㉿kali)-[~]
└─$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    216c552ea5ba   6 days ago   77.8MB
debian       latest    d91720f514f7   6 days ago   124MB

┌──(kali㉿kali)-[~]
└─$ sudo docker rmi debian
Untagged: debian:latest
Untagged: debian@sha256:e538a2f0566efc44db21503277c7312a142f4d0dedc5d2886932b92626104bff
Deleted: sha256:d91720f514f733190e529148212d53a33021d62900b9901138cfb3fc404dd33c
Deleted: sha256:8e079fee21864e07daa88efcf74f23ad5ade697c06417d0c04a45dfe580ab7f3

┌──(kali㉿kali)-[~]
└─$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    216c552ea5ba   6 days ago   77.8MB
```

- To delete all images
  - #docker rm $(docker ps -a -q)
- Create a new image from a container's changes

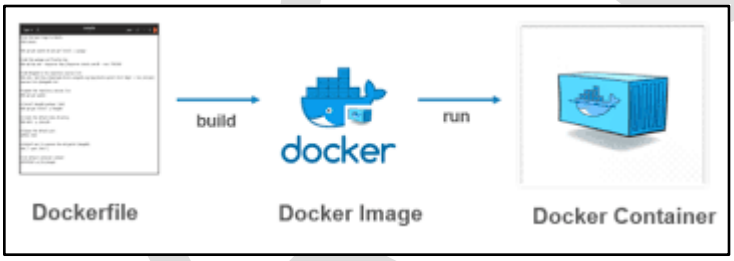
docker commit <container ID> <name for image>

```
┌──(kali㉿kali)-[~]
└─$ sudo docker commit a36bcda6a500 seematest
sha256:01f994d5b08ffeabf73ed5653271c7c89805fc38e535127fb044b9a04dc062b5
```

```
┌──(kali㉿kali)-[~]
└─$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
seematest    latest    01f994d5b08f   25 seconds ago   77.8MB
seema        latest    a07d22570929   5 minutes ago    77.8MB
ubuntu       latest    216c552ea5ba   6 days ago       77.8MB
```

Introduction to docker file

- It is a text file that contains all commands a user could call on the command line to assemble an image.
- Using docker build can create an automated build that executes several command line instructions in succession



FROM → for base image this command must be on top of dockerfile

RUN → to execute commands it will create a layer in image

MAINTAINER → author/owner/description

COPY → copy files from local system (docker VM). we need to provide source, destination (we can't download from internet & any remote repo)

ADD → similar to copy but it provides a feature to download files from internet, also we extract file at docker image side

EXPOSE → to expose ports such as port 8080 for tomcat, port 80 for nginx etc

WORKDIR → to set working directory for a container creation

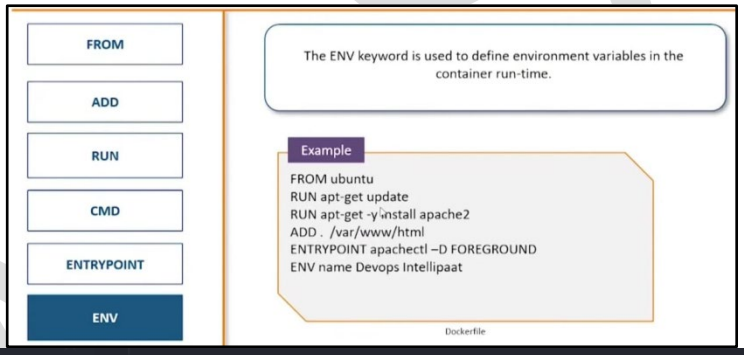
CMD → execute commands but during container creation

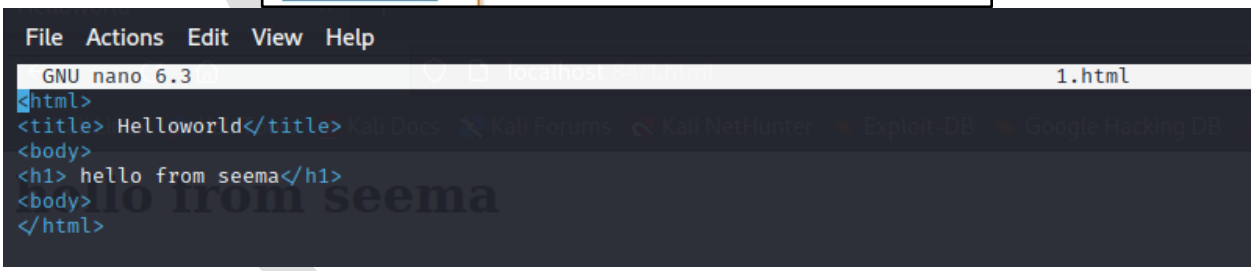
ENTRYPOINT → similar to CMD, but has higher priority over CMD, first commands will be executed by ENTRYPOINT only

ENV → environment variable

ARG → used inside dockerfile to define the name of the parameter & its default value.

Difference between ENV & ARG is that after you set an environment variable, using ARG you will not be able to access that later on when you try to run docker container





```
┌──(kali㉿kali)-[~/dockerfile]
└─$ sudo docker build . -t new_dockerfile
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM ubuntu
 ---> 216c552ea5ba
Step 2/6 : RUN apt-get update
 ---> Running in 6a2d36a58992
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [463 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [360 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [431 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [4644 B]
Get:11 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [8056 B]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [797 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [481 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [540 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [7271 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [3175 B]
Fetched 23.4 MB in 53s (445 kB/s)
Reading package lists ...
```

```
┌──(kali㉿kali)-[~/dockerfile]
└─$ sudo docker images
REPOSITORY        TAG       IMAGE ID       CREATED             SIZE
new_dockerfile    latest    3ee83067355a   7 seconds ago       224MB
seemanaaz/apache  latest    6d7fcf55a067   About an hour ago   224MB
seematest         latest    01f994d5b08f   About an hour ago   77.8MB
seema             latest    a07d22570929   2 hours ago         77.8MB
ubuntu            latest    216c552ea5ba   6 days ago          77.8MB

┌──(kali㉿kali)-[~/dockerfile]
└─$ sudo docker run -it -p 84:80 -d new_dockerfile
ca08603c0f2e5ab3fd51fe4b211ab68ae13e9cb5e2a41f821fbbc71de7d547af

┌──(kali㉿kali)-[~/dockerfile]
└─$ sudo docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED             STATUS          PORTS                                   NAMES
ca08603c0f2e   new_dockerfile    "/bin/sh -c 'apachec…"   12 seconds ago      Up 11 seconds   0.0.0.0:84->80/tcp, :::84->80/tcp       nice_kowalevski
a9fc3806107d   seemanaaz/apache  "bash"                   About an hour ago   Up 54 minutes   0.0.0.0:82->80/tcp, :::82->80/tcp       keen_burnell
```

Helloworld

localhost:84/1.html

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

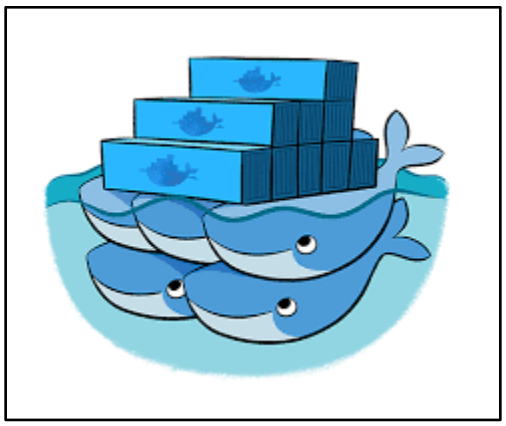
hello from seema

Container Orchestration

It monitors containers' health. Container orchestration is the automation of much of the operational effort required to run containerized workloads and services. This includes a wide range of things software teams need to manage a container's lifecycle, including provisioning, deployment, scaling (up and down), networking, load balancing and more.Because containers are lightweight and ephemeral by nature, running them in production can quickly become a massive effort. Particularly when paired with microservices—which typically each run in their own containers—a containerized application might translate into operating hundreds or thousands of containers, especially when building and operating any large-scale system.This can introduce significant complexity if managed manually. Container orchestration is what makes that operational complexity manageable for development and operations—or DevOps—because it provides a declarative way of automating much of the work. This makes it a good fit for DevOps teams and culture, which typically strive to operate with much greater speed and agility than traditional software teams.

Docker swarm is one of the container orchestration tools which comes with docker.

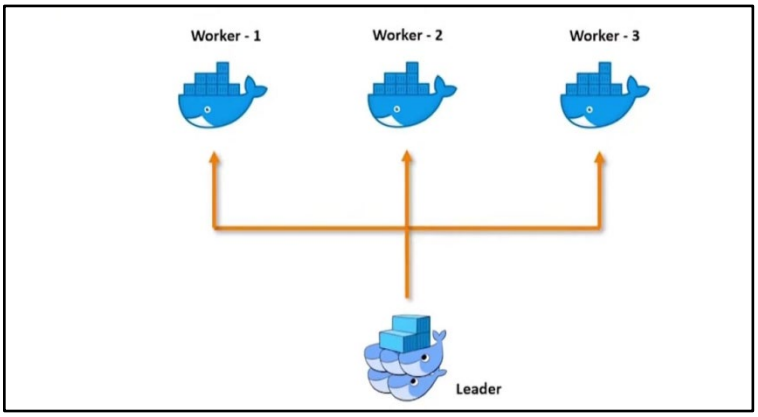
Docker Swarm



The term "swarm" refers to the group of anything e.g., nodes that form a cluster. In the Cluster, all nodes work by co-coordinating with each other, or we can say that all Nodes work as a whole. Docker Swarm is a container orchestration tool running the Docker application. It has been configured to join together in a cluster. The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.

It is basically a collection of either virtual machines or physical machines that run the Docker Application. This group of several machines is configured to make a cluster.

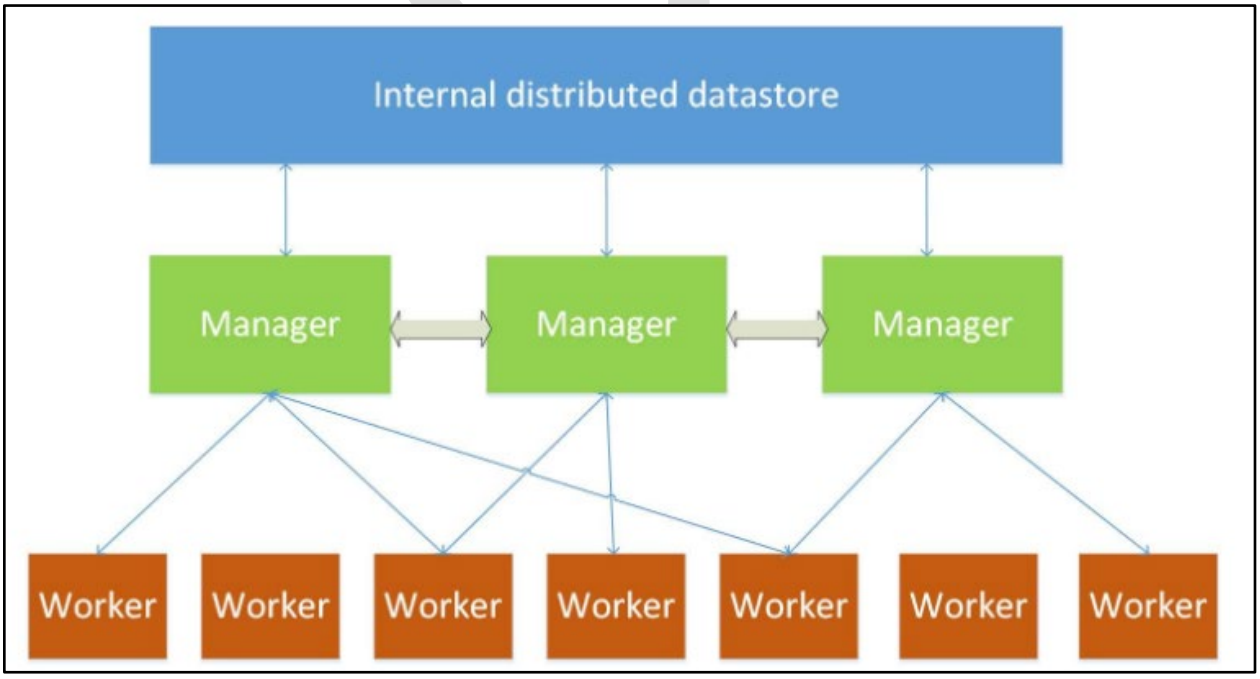
When the configuration of these machines gets complete and takes a form of cluster, we can still run the Docker commands that we're used to, but now they will be executed by the several machines present in the Cluster.



Types of Nodes

There are three types of docker swarm nodes which are given below.

1. Leader Node
2. Manager Node
3. Worker Node



1. **Leader Node:**

Once the cluster formation process gets completed, an algorithm known as "Raft consensus" is used to make a leader node among the Nodes available in the Cluster.

The leader node takes care of tasks such as task orchestration decisions for the swarm, managing swarm. If the leader node gets down or becomes unavailable due to any reason, the leadership is transferred to another Node using the same algorithm.

2. **Manager Node:**

Once the Cluster gets established successfully, an algorithm is used to choose one of them as the leader node, and that algorithm is known as the "Raft consensus".

The Node which is chosen as the leader has the responsibility to make all of the swarm management, also make the decisions for the swarm.

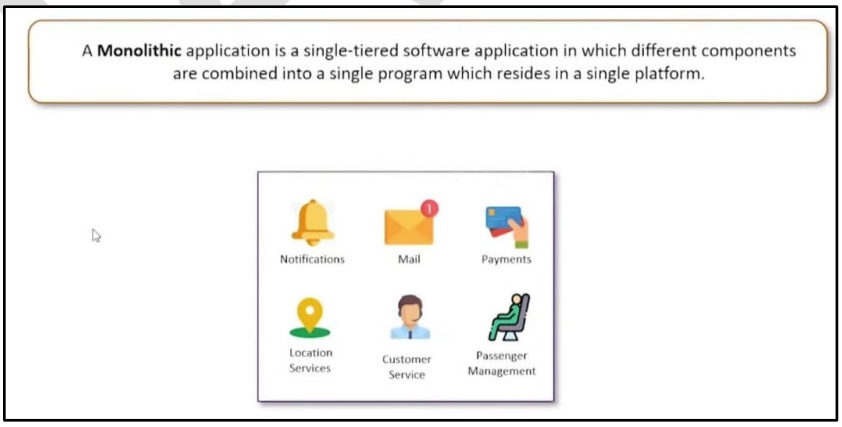
If somehow the leader node becomes unavailable due to some fatal error or hardware failure, another node is again chosen from the available nodes.

3. **Worker Node**

Inside the docker swarm that contains a vast number of hosts, every worker node performs the received tasks/operations. Also, it executes each task allocated by the leader node(or manager node).

In general, all Nodes are the worker nodes even the manager node is also a worker node and capable of performing the task/operations when required resources are available for them.

Monolithic application



If all the functionalities of a project exist in a single codebase, then that application is known as a monolithic application. We all must have designed a monolithic application in our lives in which

we were given a problem statement and were asked to design a system with various functionalities. We design our application in various layers like presentation, service, and persistence and then deploy that codebase as a single jar/war file. This is nothing but a monolithic application, where **"**mono**"** represents the single codebase containing all the required functionalities.

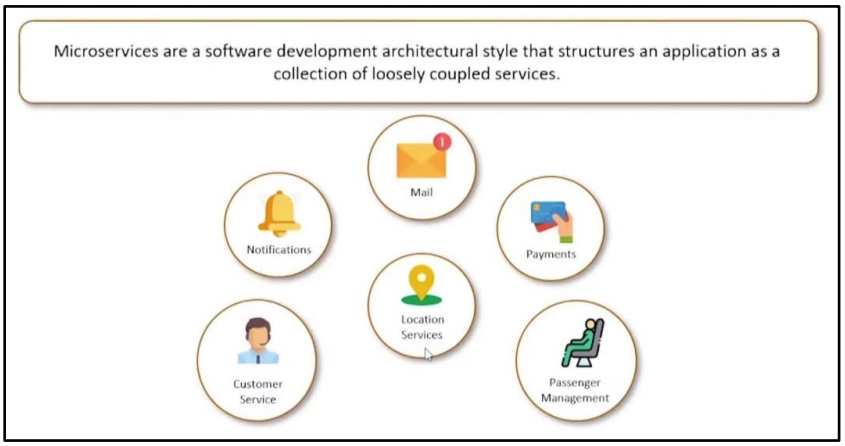
Disadvantages of Monolithic applications:

- It becomes too large with time and hence, difficult to manage.
- We need to redeploy the whole application, even for a small change.
- As the size of the application increases, its start-up and deployment time also increases.
- For any new developer joining the project, it is very difficult to understand the logic of a large Monolithic application even if his responsibility is related to a single functionality.
- Even if a single part of the application is facing a large load/traffic, we need to deploy the instances of the entire application in multiple servers. It is very inefficient and takes up more resources unnecessarily. Hence, horizontal scaling is not feasible in monolithic applications.
- It is very difficult to adopt any new technology which is well suited for a particular functionality as it affects the entire application, both in terms of time and cost.
- It is not very reliable, as a single bug in any module can bring down the entire monolithic application.

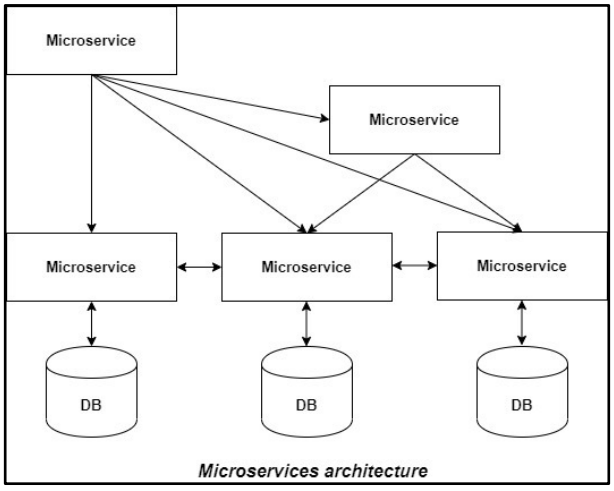
Advantages of monolithic applications:

- Simple to develop relative to microservices, where skilled developers are required in order to identify and develop the services.
- Easier to deploy as only a single jar/war file is deployed.
- Relatively easier and simpler to develop in comparison to microservices architecture.
- The problems of network latency and security are relatively less in comparison to microservices architecture.
- Developers need not learn different applications, they can keep their focus on one application.

Micro services



It is an architectural development style in which the application is made up of smaller services that handle a small portion of the functionality and data by communicating with each other directly using lightweight protocols like HTTP. According to Sam Newman, "Microservices are the small services that work together."



Advantages of micro services:

- It is easy to manage as it is relatively smaller.
    - If there's any update in one of the micro services, then we need to redeploy only that micro service.
    - Micro services are self-contained and, hence, deployed independently. Their start-up and deployment times are relatively less.

- It is very easy for a new developer to onboard the project as he needs to understand only a particular microservice providing the functionality he will be working on and not the whole system.
- If a particular microservice is facing a large load because of the users using that functionality in excess, then we need to scale out that microservice only. Hence, the microservices architecture supports horizontal scaling.
- Each microservice can use different technology based on the business requirements.
- If a particular microservice goes down due to some bug, then it doesn't affect other microservices and the whole system remains intact and continues providing other functionalities to the users.

Disadvantages of microservices:

- Being a distributed system, it is much more complex than monolithic applications. Its complexity increases with the increase in a number of microservices.
- Skilled developers are required to work with microservices architecture, which can identify the microservices and manage their inter-communications.
- Independent deployment of microservices is complicated.
- Microservices are costly in terms of network usage as they need to interact with each other and all these remote calls result in network latency.
- Microservices are less secure relative to monolithic applications due to the inter-services communication over the network.
- Debugging is difficult as the control flows over many microservices and to point out why and where exactly the error occurred is a difficult task.

Kubernetes

Kubernetes is an open-source container management tool which automates container deployment, container scaling and load balancing.

- It schedules, runs and manages isolated containers which are running on virtual/ physical/ cloud machines.
- All top cloud providers support Kubernetes.

History:

- Google developed an internal system called ‘borg’ (later named as omega) to deploy and manage thousands of google applications and services on their cluster.
- In 2014, google introduced Kubernetes as an open-source platform written in Golang, and later donated to CNCF (cloud Native Computing Foundation).

Online platform for K8s:

1. Kubernetes playground
2. Play with K8s
3. Play with Kubernetes classroom

Cloud based K8s services:

1. GKE: Google Kubernetes Services
2. AKS: Azure Kubernetes services
3. Amazon EKS: Amazon Elastic Kubernetes Services

Kubernetes installation tool:

1. Minicube
2. Kubeadm

Problems with scaling up the containers:

- Containers cannot communicate with each other.
- Autoscaling and load balancing was not possible.
- Containers had to be managed carefully.

Features of Kubernetes:

- Orchestration (clustering of any number of containers running on different networks)
- Autoscaling (supports both horizontal and vertical scaling)
- Auto-healing
- Load balancing
- Platform independent (cloud/ virtual/ physical)

- Fault tolerance (node/ POD failure)
- Rollback (going back to previous version)
- Health monitoring of containers
- Batch execution (one time, sequential, parallel)

Features	**Kubernetes**	**Docker Swarm**
Installation & cluster configuration	Complicated & time consuming	Fast & easy
Supports	K8s can work with almost all container types like Docker, Rockit, ContainerD	Work with Docker only
GUI	Available	Not available
Data Volumes	Only shared with containers in same pod	Can be shared with any other container
Updates & Rollback	Process scheduling to maintain services while updating	Progressive updates & service health monitoring through update
Autoscaling	Support vertical & horizontal autoscaling	Do not support auto scaling
Logging & monitoring	Inbuilt tool present for monitoring	Third party tools like splunk

How does Kubernetes work?

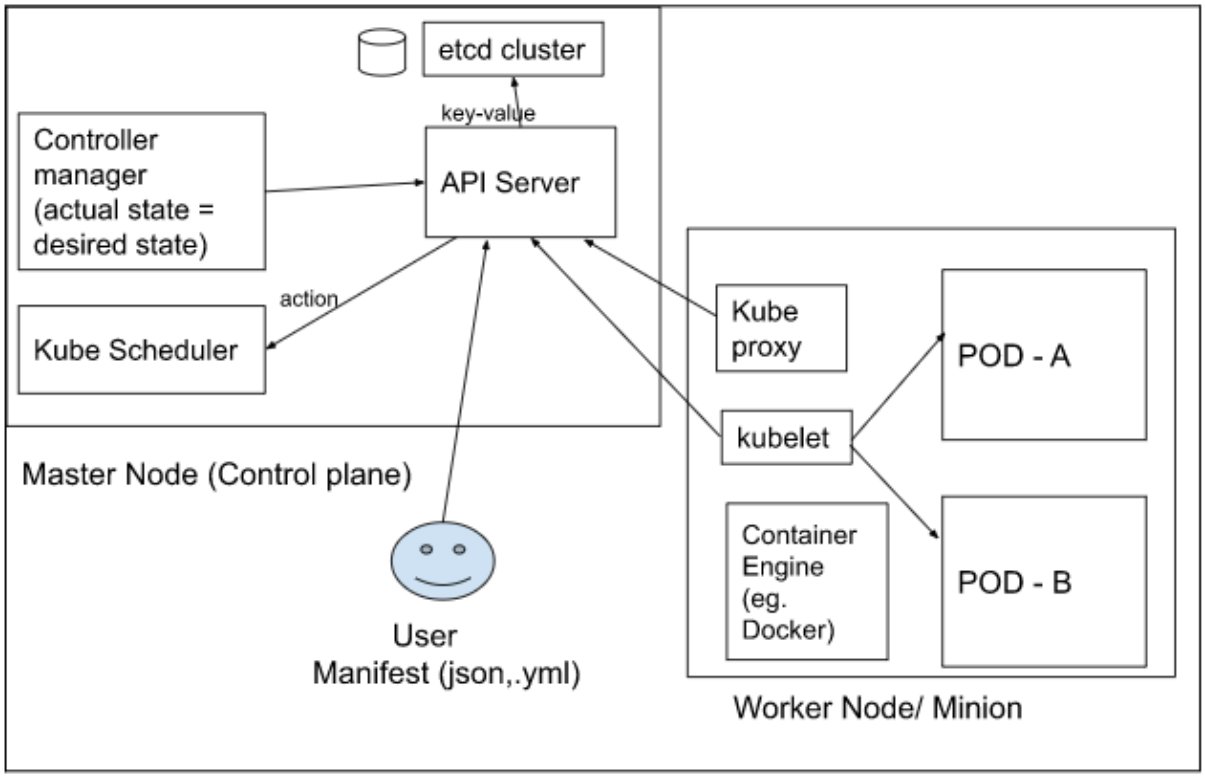
In Kubernetes, there is a master node and multiple worker nodes, each worker node can handle multiple pods. Pods are just a bunch of containers clustered together as a working unit. You can start designing your applications using pods. Once your pods are ready, you can specify pod definitions to the master node, and how many you want to deploy. From this point, Kubernetes is in control. It takes the pods and deploys them to the worker nodes. If a worker node goes down, Kubernetes starts new pods on a functioning worker node. This makes the process of managing

the containers easy and simple. It makes it easy to build and add more features and improve the application to attain higher customer satisfaction. Finally, no matter what technology you're invested in, Kubernetes can help you.

Kubernetes Architecture:

- We create manifest (.yml/json).
- Apply this to the cluster (to master) to bring the desired state.
- POD runs on a node which is controlled by the master.

Kubernetes is a master-slave type of architecture. It operated with Master node and worker node principles



Kubernetes Architecture

Role of Master Node:

- Kubernetes cluster contains containers running or bare metal/ vm instances. Cloud instances/ all mix.
- Kubernetes designates one or more of these as masters and all others as workers.
- The master is now going to run a set of k8s processes. These processes will resume smooth functioning of the cluster. These processes are called “control planes”.

- Can be multi-master for high availability.
- Master runs the control plane to run the cluster smoothly.

Components of Control Plane (master):

- Kube-API server
- Etcd
- Kube-scheduler
- Controller manager

1. Kube-API server (for all communications)

- This api-server interacts directly with the user (i.e we apply .yml or json manifest to kube-apiserver).
- This kube-apiserver is meant to scale automatically as per load.
- Kube api-server is the front-end of the control-plane.

2. Etcd:

- It stores metadata and status of the cluster.
- Etcd is a consistent and highly available store (key-value store).
- Source of touch for cluster state (info about state of cluster).
- Fully replicated: the entire state is available on every node in the cluster.
- Secure: implements automatic TLS with optional client-certificate authentication.
- Fast: benchmarked at 10,000 writes per second.

3. Kube scheduler:

- When users make requests for the creation and management of PODs, the kube scheduler is going to take action on these requests.
- Handles POD creation and management.
- Kube scheduler matches/ assigns any node to create and run PODs.
- A scheduler watches for newly created PODs that have no nodes assigned. For every POD that the scheduler discovers the scheduler becomes responsible for finding best node for that POD to run on

- Scheduler gets the information for hardware configuration from the configuration file and schedules the PODs on nodes accordingly.

4. Controller Manager:

- Make sure that the actual state of the cluster matches the desired state.
- Two possible choices for controller manager:
    - If k8s on cloud, then it will be cloud-controller-manager.
    - If k8s on non-cloud then it will be kube-controller-manager

Node/Worker Node/ Minion

Worker Node is going to run 3 important pieces of software/ process.

1. Kublet
2. Container engine
3. Kubeproxy

1. Kublet:

- Agent running on the node.
- Listening to Kubernetes master. (e.g-POD creation request)
- Use port 10255.
- Send success/fail reports to master.

2. Container Engine:

- Works with kublet.
- Pulling images.
- Start/ stop containers.
- Exposing containers on ports specified in manifest.

3. Kubeproxy:

- Assign IP to each POD.
- It is required to assign IP addresses to PODs (dynamic).

- Kube-proxy runs on each node and this makes sure that each POD will get its own unique IP address. These 3 components are collectively called NODE.

POD:

- Smallest unit in Kubernetes.
- POD is a group of one or more containers that are deployed together on the same host.
- A cluster is a group of nodes.
- A cluster has at least one worker node and master node.
- In Kubernetes the control unit is the POD, not containers.
- It consists of one or more tightly coupled containers.
- POD runs on a node which is controlled by the master.
- Kubernetes only knows about PODs (does not know about individual containers).
- Cannot start containers without a POD.
- One POD usually contains one container.

Multi container PODs:

- Share access to memory space.
- Connect to each other using local host <container port>
- Share access to the same volume.
- Containers within POD are deployed in an all-or-nothing manner.
- Entire POD is hosted on the same node (scheduler will decide about which node).

POD limitations:

- No auto healing or auto scaling.
- POD crashes.

Setup Kubernetes master and node on AWS:

Login into AWS account – launch 3 instances – ubuntu (t2.medium)

Master must have 2vcpu and 4gb RAM

Now using puttygen, create private key

Access all the 3 instances (1 master, 2 node) using putty.

Commands common for master and node:

sudo su

apt-get update

Now install https package

apt-get install apt-transport-https

This https is needed for intra cluster communication (particularly from control plane to individual pods)

Now install docker on all 3 instances

sudo apt install docker.io -y

To check whatever docker is installed or not

docker --version

systemctl start docker

systemctl enable docker

Setup open GPG key. This is required for intra cluster communication. It will be added to source key on this node i.e when k8s sends signed information to our host, it is going to accept those information because this open GPG key is present in the source key.

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | # sudo apt-key add

Edit source list file (apt-get install nano)

nano /etc/apt/source.list.d/Kubernetes.list

deb http://apt.kubernetes.io/kubernetes-xenial.main

Exit from nano: ctrl+x, caps+y – enter

apt-get update – install all packages

apt-get install -y kublet kubeadm kubectl Kubernetes-cni

Bootstrapping the master node (in master)

To initialize k8s cluster

kubeadm init

You will get one long command started from “kubeadm join 172.31.6.165:6443”

Copy this command and save on notepad

Create both .kube and its parent directories (-p)

mkdir -p $HOME/.kube

Copy configuration to kube directory (in configuration file)

sudo cp -i /etc/Kubernetes/admin.config $HOME /.kube/config

Provide user permissions to config file

chown $(id-u): $(id-g) $ HOME/.kube/config

Deploy flannel node network for its repository path. Flannel is going to place a binary in each node.

sudo kubectl apply -f

https://raw.githubusercontent.com/coreos/flannel/master/documentation/kube-flannel.yml

sudo kubectl apply -f

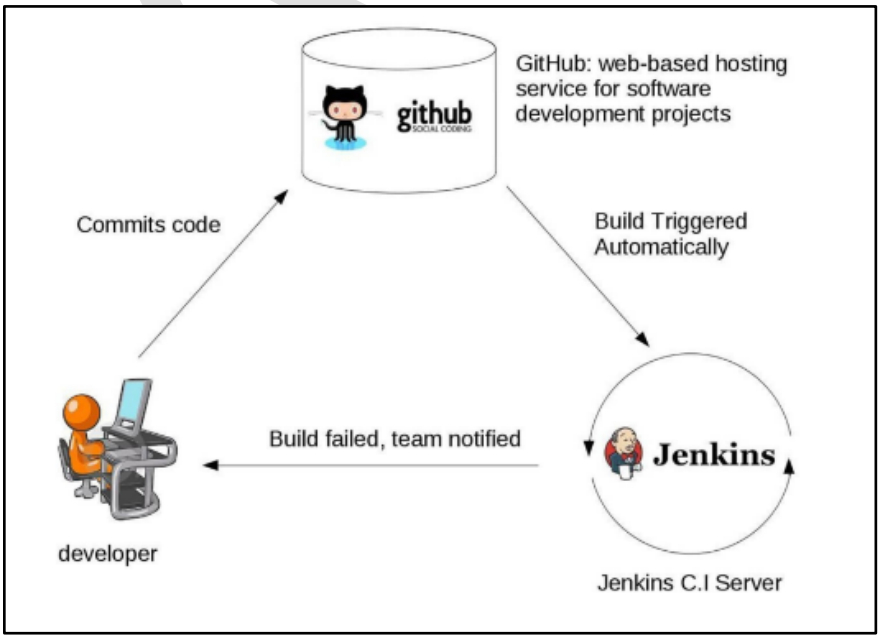
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifest/kube-flannel-rbac.yml

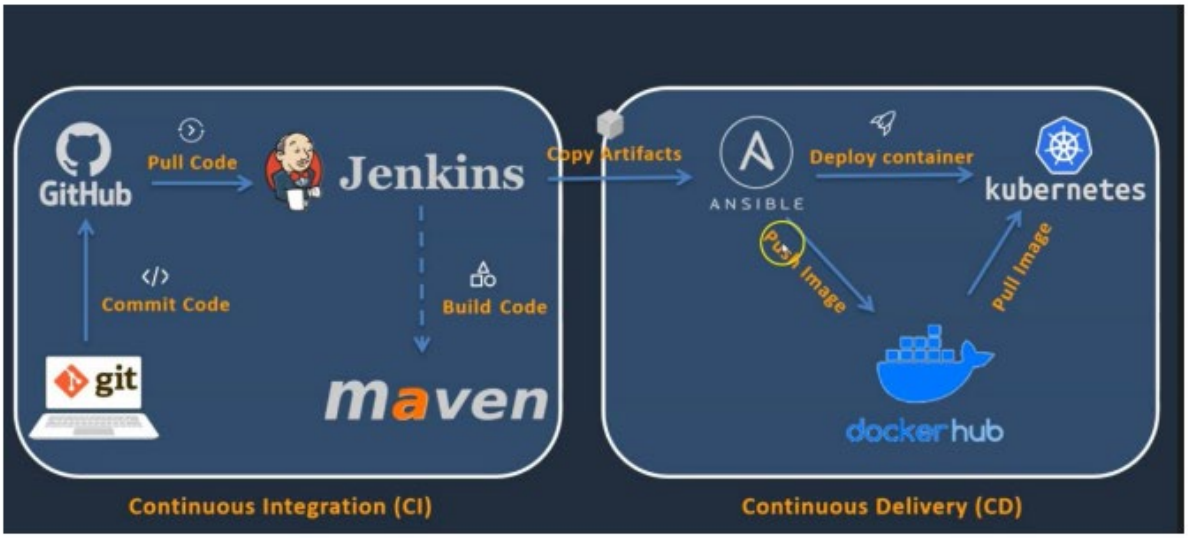
Configure worker node: paste the long commands above on both the nodes

Go to master: # kubectl get nodes

Jenkins

- Jenkins is an open-source project written in Java that runs on windows, mac, OS and other Unix like os. It is free; community supported and might be your first-choice tool for CI.
- Jenkins automates the entire s/w development life cycle.
- Jenkins was originally developed by sun microsystem in 2004 under the name hudson.
- The project was later named Jenkins when Oracle bought Microsoft.
- It can run on any major platform without any compatibility issues.
- Whenever developers write code, we integrate all that code of all developers at that point of time and we build, test and deliver/ deploy to the client. This process is called as CI/CD.
- Jenkins helps us to achieve this.
- Because of CI now bugs will be reported fast and get rectified fast. So that entire software development happens fast.
- Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
- Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.
- In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, the build is tested for deployment. If deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process and hence the name continuous integration/deployment.





Workflow of Jenkins:

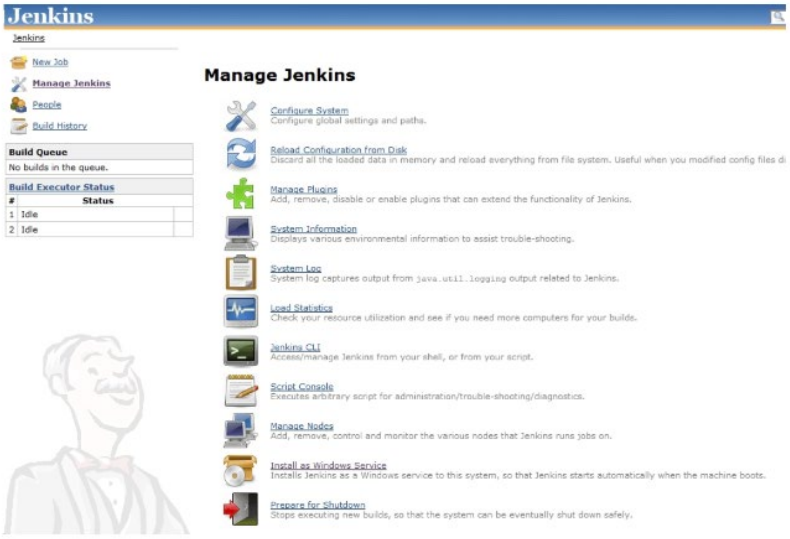
- We can attach git, maven, selenium and artifactory plugins to Jenkins.
- Once developers put codes in github Jenkins pulls that code and sends it to maven for build.
- Once build is done, then Jenkins will pull that code and send it to the artifactory as per requirement and so on.
- We can also deploy with Jenkins.

Advantages of Jenkins:

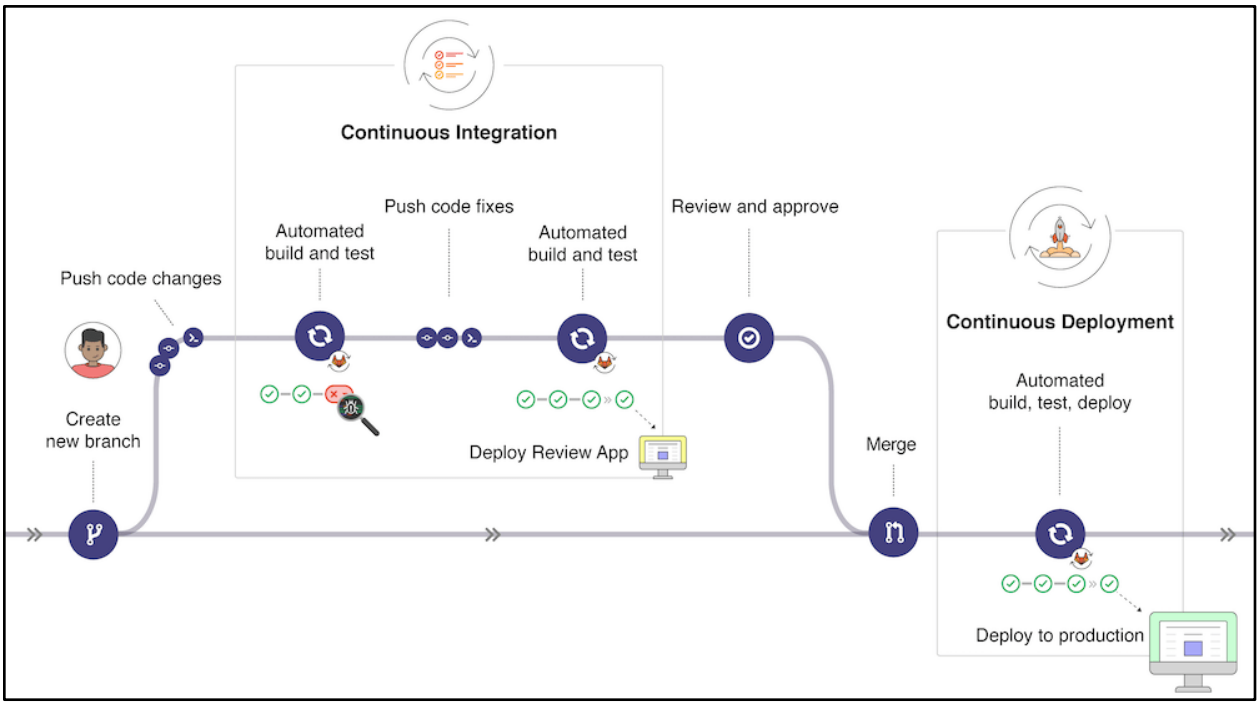
- It has lots of plugins available.
- You can write your own plugins.
- You can use community plugins.
- Jenkins is not just a tool. It is a framework i.e you can do whatever you want. All you need is plugins.
- We can attach slaves (nodes) to jenkins master. It instructs other (slaves) to do job. If slaves are not available, Jenkins itself does the job.
- Jenkins also behaves as a cron server replacement i.e can do scheduled tasks.
- It can create labels.

Jenkins Plugins:-

By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git. In fact, for integration with tools like Maven you need to install respective plugins in your Jenkins.



CI/CD Project:



Go to google chrome – search ‘git download’ – download 2.31.1 for windows – click to download.

Git download and install:

Open the download file preamble – c:\programfiles\git – select components – select start menu folder – choosing default editor vim – let git decide – git from the command line and also from

the 3rd party s/w – use the open SSL library – check as-is, commit unix-style line encodings – use minty – choose default behavior – git credentials manager core – enable file system caching – install.

Go to command prompt in laptop

- Git config –global username ‘username’
- Git config –global user.email ‘mailid’
- Git config –global –list

Download and Installation of JDK16:

Go to google chrome – search ‘java development kit download’ – download java SE development kit 16 for windows x64.

Run and follow the steps to install.

Now go to C drive – program files – java jdk16 – select path and copy it

Search ‘edit system environment variables’ in laptop – go to use variables – new

Variable name- JAVA_HOME

Variable value- paste the path here

Now go to system variable – new

Variable name – JAVA_HOME

Variable value- paste the path here

Now go inside programfiles – bin – paste the path

Again, go to ‘edit system environment variables’ – system variable – path – new – paste path

Now verify in command prompt

C:\users\home\echo% JAVA_HOME%

o/p- c:\ programfiles\java\jdk-16

Maven download and install:

Go to google chrome – search maven.apache.org – download – binary zip archive

Extract files – c:\devtools

Go to c:\ - devtools – apache-maven – copy the path

Now search ‘edit system environmental variables’ – system variable – new

Variable name – M2_HOME

Variable value - paste the path here

Now go inside apache maven folder – bin – copy path

Now again go to ‘environmental variable’ – system variable – path – new – paste path

Now open command prompt

C:\users\home > mvn -version

C:\users\home > echo % M2_HOME%

Now restart the laptop

Jenkins download and install:

Go to google chrome – Jenkins.io – download – select LTS – windows – download

Open download file – run and install

After installation it automatically open as local host: 8080

Unlock the page by using password

Now install suggested plugins

Ask for username and password

Username – admin

Password – admin123

Email address- iacsd@gmail.com

Save and continue

Start using Jenkins

Plugins: plugins are small libraries that add new abilities to Jenkins and can provide integration points to other tools.

Go to google chrome – localhost:8080 – login

Go to manage Jenkins on left side of Jenkins dashboard – manage plugins – available – select maven integration and green balls – install without restart

Go to new item – maven project

Now go to manage Jenkins – global tool configuration

Go to add JDK

Uncheck the install automatically option

NAME – JAVA

JAVA_HOME – c:\programfiles\java\jdk

Now go to maven

Name – MAVEN

MAVEN_HOME – c:\devtools\apache-maven

Maven Project (by maven):

Go to https://github.com/technicalguftgu/time-tracker

Click on time tracker repo

'fork' to copy this repo

Sign-in into your github account

Click on time-tracker repo

Clone

Go to c drive

Git clone <url of time-tracker repo>

Cd time-tracker

C:\time-tracker > maven clean package

Maven Project (by Jenkins):

Now go to Jenkins – new item – enter name -> mymavenproject

Then select maven project – ok

Source code management – git – repo url

Build option – root POM – pom.xml

Goals and options – clean package – save

Go to Jenkins home page – click on mymavenproject – build now

Scheduled Project:

Click on any project – configure – build triggers – build periodically - * * * * * - save

Can see automatic builds after every 1 min.

You can manually trigger build as well.

Source Code Polling (Poll SCM):

Now go to Jenkins home page – go to mymavenproject – configure

Now go to build trigger

Enable poll SCM

Schedule * * * * * - save

Now go to the github account – do some changes in README.md – commit changes.

You can see after 1 min, it builds automatically.

User Management:

go to Jenkins homepage – manage Jenkins – manage user

create two users – Tom & Bob

now login as Tom

{by default you have all the permissions}

Login as ‘admin’ again

Go to manage Jenkins – manage plugins – search ‘role-based authorization strategy’ – install

Without restart.

Go to Jenkins homepage – manage Jenkins – configure global security – select role-based

Strategy – save.

Login as ‘Tom’ – access denied

Now attach permission

Go to Jenkins – manage Jenkins – manage and assign role – manage roles

Role to add – employee

Go to item project – add developer and tester [pattern – dev* test*]

Then assign roles

User/group to add Tom & Bob

How to install Jenkins on Ubuntu:

➢ Login into AWS account and create one ubuntu instance.

➢ Access it through putty and login as ‘ubuntu’.

➢ Use given commands

sudo apt-get update

sudo apt-cache search openjdk

#sudo apt-get install openjdk-8-jdk

#java -version

#sudo vi /etc/apt/source.list

Paste at the bottom

deb https://pkg.jenkins.io/debian-stablebinary/

#sudo apt-get update

wget -q -o ---

sudo apt-cache search Jenkins

sudo apt-cache Madison Jenkins

sudo apt-get install Jenkins -y

sudo service Jenkins status -q

Copy public ip from AWS and paste in chrome url – public:8080

Go to instance #sudo cat /var/lib/initialpassword