ANNAMACHARYA UNIVERSITY, RAJAMPET

(ESTD UNDER AP PRIVATE UNIVERSITIES (ESTABLISHMENT AND REGULATION) ACT, 2016
RAJAMPET, Annamayya District, AP, INDIA

Course : DevOps

Course Code : **24FMCA33T**

Branch : **MCA**

Prepared by : **B. Hari Krishna**

Designation : **Assistant Professor**

Department : **MCA**

ANNAMACHARYA UNIVERSITY, RAJAMPET

(ESTD UNDER AP PRIVATE UNIVERSITIES (ESTABLISHMENT AND REGULATION) ACT, 2016
RAJAMPET, Annamayya District, AP, INDIA

Title of the Course : **DevOps**
Category : **PC**
Year : **II**
Semester : **I**
Course Code : **24FMCA33T**
Branch : **MCA**

Lecture Hours	Tutorial Hours	Practice Hours	Credits
3	0	0	3

COURSE OBJECTIVES:

- To give basic knowledge of AWS practices and Linux commands.
- To give strong knowledge of AWS practices.
- To give strong foundation of Version control system.
- To give strong foundation of Jenkins CI/CD methodologies.
- To give foundation of the Docker containerization tool.

COURSE OUTCOMES:
The Student will be able to

1. Summarize cloud computing components and AWS cloud.
2. Apply AWS services.
3. Analyze version control system.
4. Analyze continuous integration frame work.
5. Apply Docker containerization.

UNIT I **13**

AWS INTRODUCTION: Introduction to cloud computing, Service Models in cloud computing, Deployment models in Cloud Computing, Introduction to AWS, AWS account creation and free tier limitations and overview. Ec2 Service, Launching an EC2 instance, Elastic Load Balancer and types of ELB, introduction to Linux, Linux architecture, File types, Directories, File permissions

UNIT II **12**

AWS SERVICES: S3 Bucket- bucket creation, storage classes, S3 versioning, Bucket policies, Replication Rules in S3, IAM service-Groups, Users, Policies & Roles, AWS-CLI Commands, RDS, VPC, Route53, Monitoring tools

UNIT III **10**

Git&GitHub: introduction to Version control system, types of VCS, GIT life cycle, basic commands, git branches, create a remote repository

UNIT IV **11**

JENKINS: Why continuous integration, Introduction to Continuous Integration, Release and relation with Devops, Jenkins Introduction and setup, Jenkins projects/jobs, Master/Slave concept, Jenkins poll scm, build periodically, web hooks, Jenkins pipeline, Jenkins plugins, Jenkins CLI and Jenkins administration

UNIT V **10**

DOCKER: Introduction to docker, Virtualization an Docker cli and Containerization differences, Docker Installation, Docker cli, Docker File, Docker Volumes, Docker-compose, Docker Network.

PRESCRIBED TEXTBOOKS:

1. Mark Wilkins, Learning Amazon Web Servicse(AWS) Pearson Publications, 1st Edition, 2019.
2. Sandeep Rawat, CI/ CD Pipeline with Docker and Jenkins, BPB Publications, 3rd Edition, 2023

REFERENCE BOOK:

1.Len Bass, Ingo Weber, LimingZhu- DevOps: A Software Architect's Perspective, 1st Edition, 2015.

CO-PO MAPPING:

Course Outcomes	Foundation Knowledge	Problem Analysis	Development of Solutions	Modern Tool Usage	Individual and Teamwork	Project Management and Finance	Ethics	Life-long Learning
24FMCA33T.1	2	2	1	-	-	-	-	-
24FMCA33T.2	3	2	1	-	-	-	-	-
24FMCA33T.3	3	3	2	-	-	-	-	-
24FMCA33T.4	3	3	2	-	-	-	-	-
24FMCA33T.5	3	2	1	-	-	-	-	-

UNIT-I

AWS INTRODUCTION: Introduction to cloud computing

The term Cloud refers to a Network or Internet. In other words, we can say that Cloud is something, which is Present at remote location. Cloud can provide services over network, i.e., on public networks or on private Networks, i.e., WAN, LAN or VPN.

Cloud Computing

Cloud computing means storing and accessing the data and programs on remote servers that are hosted on internet instead of computer's hard drive or local server. Cloud computing is also referred as Internet based computing.

Cloud Computing is the delivery of computing services such as servers, storage, databases, networking, software, analytics, intelligence, and more, over the Cloud (Internet).

Cloud Computing provides an alternative to the on-premises datacenter. With an on-premises datacenter, we have to manage everything, such as purchasing and installing hardware, virtualization, installing the operating system, and any other required applications, setting up the network, configuring the firewall, and setting up storage for data. After doing all the setup, we become responsible for maintaining it through its entire lifecycle.

But if we choose Cloud Computing, a cloud vendor is responsible for the hardware purchase and maintenance. They also provide a wide variety of software and platform as a service. We can take any required services on rent. The cloud computing services will be charged based on usage.

The cloud environment provides an easily accessible online portal that makes handy for the user to manage the compute, storage, network, and application resources.

ADVANTAGES OF CLOUD COMPUTING

- **Cost:** It reduces the huge capital costs of buying hardware and software.
- **Speed**: Resources can be accessed in minutes, typically within a few clicks.

- **Scalability**: We can increase or decrease the requirement of resources according to the business requirements.
- **Productivity:** While using cloud computing, we put less operational effort. We do not need to apply patching, as well as no need to maintain hardware and software. So, in this way, the IT team can be more productive and focus on achieving business goals.
- **Reliability**: Backup and recovery of data are less expensive and very fast for business continuity.
- **Security:** Many cloud vendors offer a broad set of policies, technologies, and controls that strengthen our data security.

Disadvantages

Although Cloud Computing provides a wonderful set of advantages, it has some drawbacks as well that often raise questions about its efficiency.

Security issues

Security is the major issue in cloud computing. The cloud service providers implement the best security standards and industry certifications; however, storing data and important files on external service providers always bears a risk.

AWS cloud infrastructure is designed to be the most flexible and secured cloud network. It provides scalable and highly reliable platform that enables customers to deploy applications and data quickly and securely.

Technical issues

As cloud service providers offer services to number of clients each day, sometimes the system can have some serious issues leading to business processes temporarily being suspended. Additionally, if the internet connection is offline then we will not be able to access any of the applications, server, or data from the cloud.

Not easy to switch service providers

Cloud service providers promise vendors that the cloud will be flexible to use and integrate; however switching cloud services is not easy. Most organizations may find it difficult to host and integrate current cloud applications on another platform. Interoperability and support issues may arise such as applications developed on Linux platform may not work properly on Microsoft Development Framework

Service Models In Cloud Computing

Cloud Computing can be defined as the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. Companies offering these computing services are called cloud providers and typically charge for cloud computing services based on usage.

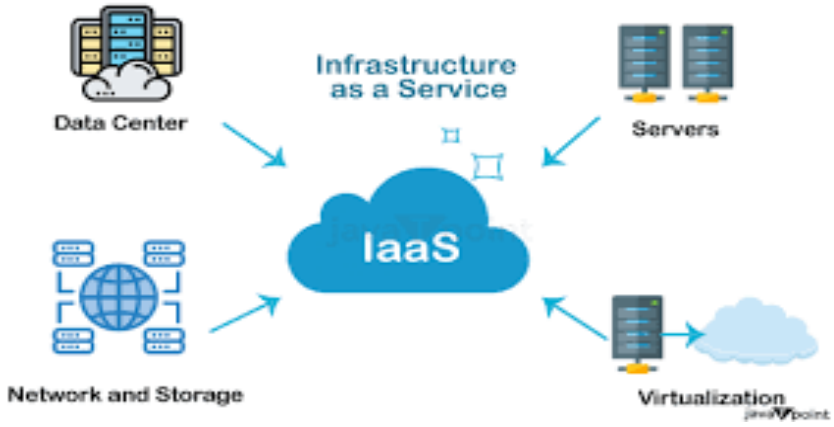
TYPES OF CLOUD SERVICES

Most cloud computing services fall into three broad categories:

1. Software as a service (Saas)
2. Platform as a service (PaaS)
3. Infrastructure as a service (IaaS)

These are sometimes called the cloud computing stack, because they are build on top of one another. Knowing what they are and how they are different, makes it easier to accomplish your goals.

IAAS:



IaaS is a type of cloud-computing service that offers the rental computing infrastructures. The cloud provider provides various infrastructure services such as servers, virtual machines, network storage. The services can be scaled up and down as per the client requirements.

Infrastructure as a service (IaaS) is a service model that delivers computer infrastructure on an outsourced basis to support various operations. Typically IaaS is a service where infrastructure is provided as an outsource to enterprises such as networking equipments, devices, database and web servers.

Infrastructure as a service (IaaS) is also known as Hardware as a service (HaaS).IaaS customers pay on a per-use basis, typically by the hour, week or month. Some providers also charge customers based on the amount of virtual machine space they use.

It simply provides the underlying operating systems, security, networking, and servers for developing such applications, services, and for deploying development tools, databases, etc.

IaaS is offered in three models: public, private, and hybrid cloud. The private cloud implies that the infrastructure resides at the customer-premise. In the case of public cloud, it is located at the cloud computing platform vendor's data center, and the hybrid cloud is a combination of the two in which the customer selects the best of both public cloud and private cloud.

Examples: Digital Ocean, Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE),

Advantages:

- **Cost Effective**: Eliminates capital expense and reduces ongoing cost and IaaS customers pay on a per use basis, typically by the hour, week or month.
- **Website hosting**: Running websites using IaaS can be less expensive than traditional web hosting.
- **Security**: The IaaS Cloud Provider may provide better security than your existing software.
- Maintenance: There is no need to manage the underlying data center or the introduction of new releases of the development or underlying software. This is all handled by the IaaS Cloud Provider.

Disadvantages of IaaS

- **Security:** Security is one of the biggest issues in IaaS. Most of the IaaS providers are not able to provide 100% security.
- **Maintenance & Upgrade:** Although IaaS service providers maintain the software, but they do not upgrade the software for some organizations.
- **Interoperability issues:** It is difficult to migrate VM from one IaaS provider to the other, so the customers might face problem related to vendor lock-in.

PaaS:

PaaS is a category of cloud computing that provides a platform and environment to allow developers to build applications and services over the internet. PaaS services are hosted in the cloud and accessed by users simply via their web browser.

- A PaaS provider hosts the hardware and software on its own infrastructure. As a result, PaaS frees users from having to install in-house hardware and software to develop or run a new application. Thus, the development and deployment of the application takes place independent of the hardware.
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.
- PaaS providers provide the Programming languages, Application frameworks, Databases, and Other tools:

Example: AWS Elastic Beanstalk, Windows Azure, Force.com, Google App Engine

Advantages:

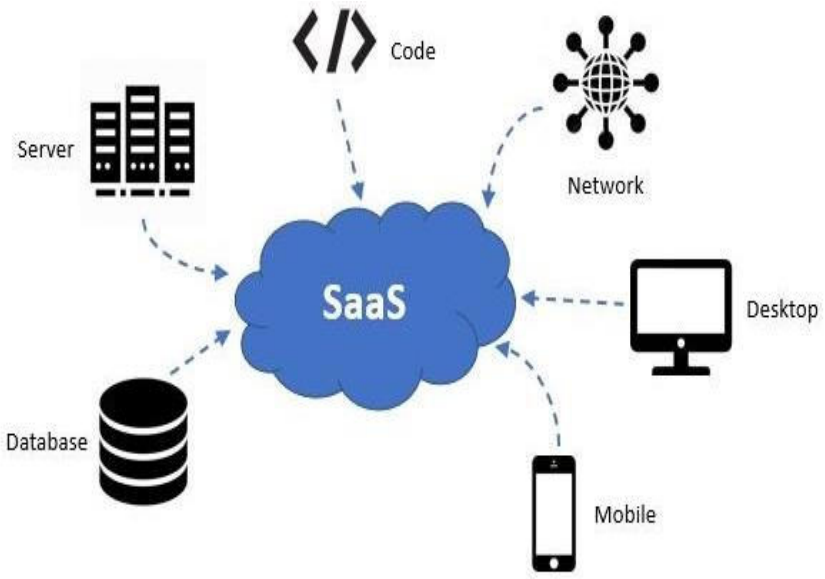
- **Simple and convenient for users**: It provides much of the infrastructure and other IT services, which users can access anywhere via a web browser.
- **Cost Effective:** It charges for the services provided on a peruse basis thus eliminating the expenses one may have for on premises hardware and software.
- **Efficiently managing the lifecycle**: It is designed to support the complete web application lifecycle: building, testing, deploying, managing and updating.
- **Efficiency:** It allows for higher-level programming with reduced complexity thus, the overall development of the application can be more effective

Disadvantages

- **Vendor lock-in:** One has to write the applications according to the platform provided by the PaaS vendor, so the migration of an application to another PaaS vendor would be a problem.
- **Data Privacy:** Corporate data, whether it can be critical or not, will be private, so if it is not located within the walls of the company, there can be a risk in terms of privacy of data.
- **Integration with the rest of the systems applications:** It may happen that some applications are local, and some are in the cloud. So there will be chances of increased complexity when we want to use data which in the cloud with the local data.

SaaS:

Software-as-a-Service (SaaS) is a way of delivering services and applications over the Internet. Instead of installing and maintaining software, we simply access it via the Internet, freeing ourselves from the complex software and hardware management. It removes the need to install and run applications



On our own computers or in the data centers eliminating the expenses of hardware as well as software maintenance. There are the following services provided by SaaS providers

Business Services – SaaS Provider provides various business services to start-up the business. The SaaS business services include ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), billing, and sales.

Document Management – SaaS document management is a software application offered by a third party (SaaS providers) to create, manage, and track electronic documents.

SaaS provides a complete software solution which you purchase on a pay-as-you-go basis from a cloud service provider. Most SaaS applications can be run directly from a web browser without any downloads or installations required. The SaaS applications are sometimes called Web-based software, on-demand software, or hosted software.

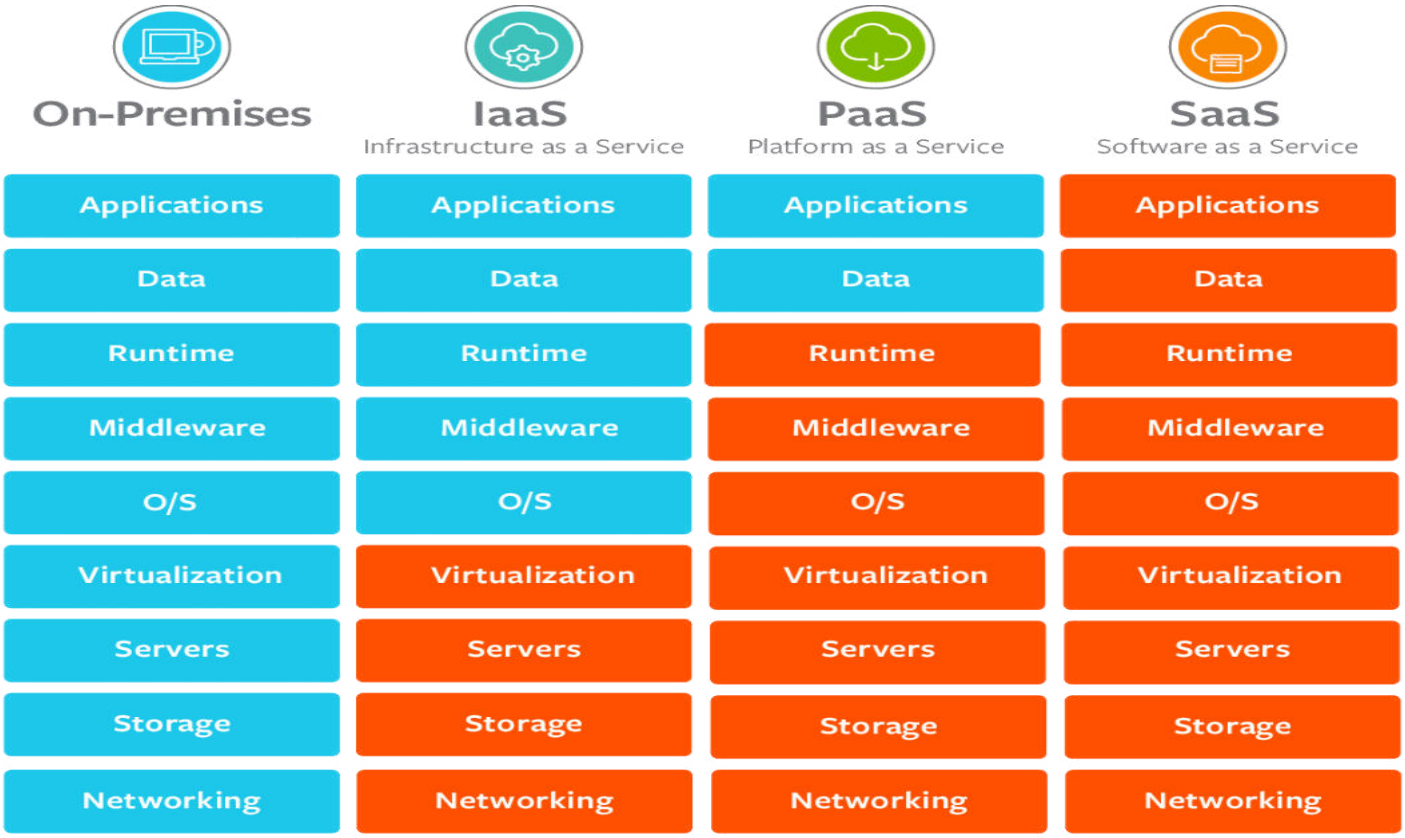
Example: Big Commerce, Google Apps, Salesforce, Dropbox, Cisco WebEx, ZenDesk and GoTo Meeting.

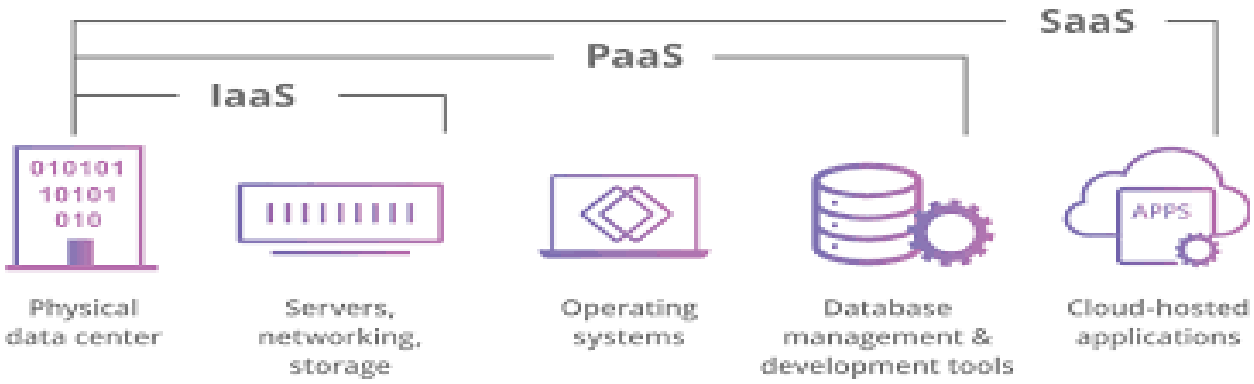
Advantages of SaaS

- **Cost Effective:** Pay only for what you use
- **Reduced time:** Users can run most SaaS apps directly from their web browser without needing to download and install any software. This reduces the time spent in installation and configuration, and can reduce the issues that can get in the way of the software deployment.
- **Accessibility:** We can Access app data from anywhere.
- **Automatic updates:** Rather than purchasing new software, customers rely on a SaaS provider to automatically perform the updates.
- **Scalability:** It allows the users to access the services and features on demand.

Disadvantages of SaaS

- **Security:** Actually, data is stored in the cloud, so security may be an issue for some users. However, cloud computing is not more secure than in-house deployment.
- **Latency issue:** Since data and applications are stored in the cloud at a variable distance from the end-user, there is a possibility that there may be greater latency when interacting with the application compared to local deployment. Therefore, the SaaS model is not suitable for applications whose demand response time is in milliseconds.
- **Total Dependency on Internet:** Without an internet connection, most SaaS applications are not usable.
- **Switching between SaaS vendors is difficult:** Switching SaaS vendors involves the difficult and slow task of transferring the very large data files over the internet and then converting and importing them into another SaaS also.





Deployment Models In Cloud Computing

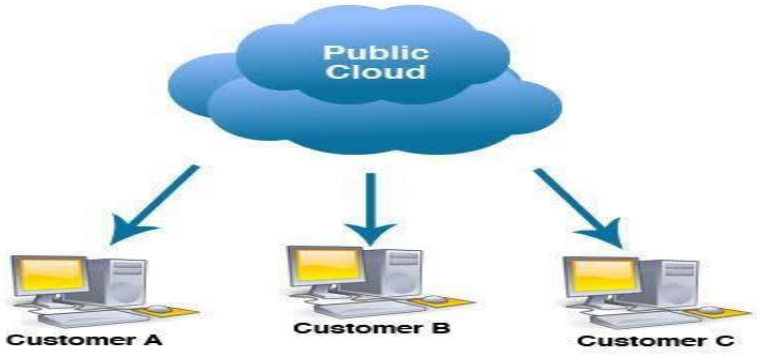
Organizations have many exciting opportunities to reimaging, repurpose and reinvent their businesses with the cloud. The last decade has seen even more businesses rely on it for quicker time to market, better efficiency, and scalability. It helps them achieve lo ng-term digital goals as part of their digital strategy.

Types of Cloud Computing Deployment Models

Most cloud hubs have tens of thousands of servers and storage devices to enable fast loading. It is often possible to choose a geographic area to put the data "closer" to users. Thus, deployment models for cloud computing are categorized based on their location. To know which model would best fit the requirements of your organization.

- Public cloud
- Private cloud
- Hybrid cloud
- Community cloud

1. Public Cloud



The public cloud makes it possible for anybody to access systems and services. The public cloud may be less secure as it is open for everyone. The public cloud is one in which cloud infrastructure services are provided over the internet to the general people or major industry groups. The infrastructure in this cloud model is owned by the entity that delivers the cloud services, not by the consumer. It is a type of cloud hosting that allows customers and users to easily access systems and services. This form of cloud computing is an excellent example of cloud hosting, in which service providers supply services to a

variety of customers. In this arrangement, storage backup and retrieval services are given for free, as a subscription, or on a per-use basis. Example: Google App Engine.

Advantages of the public cloud model:

- **Minimal Investment:** Because it is a pay-per-use service, there is no substantial upfront fee, making it excellent for enterprises that require immediate access to resources.
- **No setup cost:** The entire infrastructure is fully subsidized by the cloud service providers, thus there is no need to set up any hardware.
- **Infrastructure Management is not required:** Using the public cloud does not necessitate infrastructure management.
- **No maintenance:** The maintenance work is done by the service provider (Not users).
- **Dynamic Scalability:** To fulfill your company's needs, on demand resources are accessible.

Disadvantages of the public cloud model:

- **Data Security and Privacy Concerns:** Because it is open to the public, it does not provide complete protection against cyber-attacks and may expose weaknesses.
- **Issues with Reliability:** Because the same server network is accessible to a wide range of users, it is susceptible to failure and outages.
- **Limitation on Service/License**: While there are numerous resources that you may share with renters, there is a limit on how much you can use.

2. Private Cloud

The private cloud deployment model is the exact opposite of the public cloud deployment model. It's a one-on-one environment for a single user (customer). There is no need to share your hardware with anyone else. The distinction between private and public cloud is in how you handle all of the hardware. It is also called the "internal cloud" & it refers to the ability to access systems and services within a given border or organization. The cloud platform is implemented in a cloud-based secure environment that is protected by powerful firewalls and under the supervision of an organization's IT department. The private cloud gives the greater flexibility of control over cloud resources.

Advantages of the private cloud model:

- **Better Control:** You are the sole owner of the property. You gain complete command over service integration, IT operations, policies, and user behavior.

- **Data Security and Privacy:** It's suitable for storing corporate information to which only authorized staff has access. By segmenting resources within the same infrastructure, improved access and security can be achieved.

- **Supports Legacy Systems:** This approach is designed to work with legacy systems that are unable to access the public cloud.
- **Customization**: Unlike a public cloud deployment, a private cloud allows a company to tailor its solution to meet its specific needs.

The disadvantages of the private cloud model are:

- **Restricted Scalability:** Private clouds have restricted scalability because they are scaled within the confines of internal hosted resources. The choice of underlying hardware has an impact on scalability.
- **Higher Cost:** Due to the benefits you would receive, your investment will be higher than the public cloud (pay for software, hardware, and staffing etc).

Hybrid compute model cloud the main aim to combine these cloud (Public and Private) is to create a unified, automated, and well-managed computing environment. In the Hybrid cloud, non-critical activities are performed by the public cloud and critical activities are performed by the private cloud. Mainly, a hybrid cloud is used in finance, Healthcare and Universities. The best hybrid cloud provider companies are Amazon, Microsoft, **Google, Cisco, and NetApp.**

By bridging the public and private worlds with a layer of proprietary software, hybrid cloud computing gives the best of both worlds. With a hybrid solution, you may host the app in a safe environment while taking advantage of the public cloud's cost savings. Organizations can move data and applications between different clouds using a combination of two or more cloud deployment methods, depending on their needs.

Advantages of the hybrid cloud model:

- **Flexibility and control:** Businesses with more flexibility can design personalized solutions that meet their particular needs.
- **Cost**: Because public clouds provide for scalability, you'll only be responsible for paying for the extra capacity if you require it.

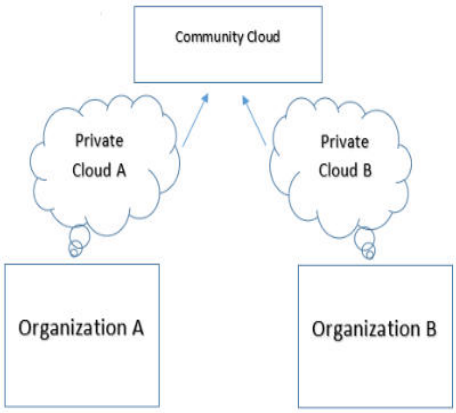
- **Security**: Because data is properly separated, the chances of data theft by attackers are considerably reduced.

The disadvantages of the hybrid cloud model are:

- **Maintenance:** A hybrid cloud computing strategy may necessitate additional maintenance, resulting in a greater operational expense for your company.
- **Difficult Integration:** When constructing a hybrid cloud, data and application integration might be difficult. It's also true that combining two or more infrastructures will offset a significant upfront cost.

Community cloud

It allows systems and services to be accessible by a group of organizations. It is a distributed system that is created by integrating the services of different clouds to address the specific needs of a community, industry, or business. The infrastructure of the community could be shared between the organization which has shared concerns or tasks. It is generally managed by a third



Party or by the combination of one or more organizations in the community.

Advantages of the community cloud model:

- **Cost Effective:** It is cost-effective because the cloud is shared by multiple organizations or communities.
- **Security**: Community cloud provides better security.
- **Shared resources:** It allows you to share resources, infrastructure, etc. with multiple organizations. Collaboration and data sharing: It is suitable for both collaboration and data sharing.

Introduction to AWS

AWS stands for Amazon Web Services. Amazon Web Services (AWS) is widely used secure cloud services platform, offering computing power, content delivery, database storage, and other functionality to help businesses scale and grow.

The AWS service is provided by the 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.

Amazon Web Services (AWS), a subsidiary of Amazon.com, has invested billions of dollars in IT resources distributed across the globe. These resources are shared among all the AWS account holders across the globe. These account themselves are entirely isolated from each other. AWS provides on-demand IT resources to its account holders on a pay-as-you-go pricing model with no upfront cost. Enterprises use AWS to reduce capital expenditure of building their own private IT infrastructure (which can be expensive depending upon the enterprise's size and nature). All the maintenance cost is also barred by the AWS that saves a fortune for the enterprises.

History of AWS:

- 2003: In 2003, Chris Pinkham and Benjamin Black presented a paper on how Amazon's own internal infrastructure should look like. They suggested to sell it as a service and prepared a business case on it. They prepared a six-page document and had a look over it to proceed with it or not. They decided to proceed with the documentation.
- 2004: A SQS stand for "Simple Queue Service" was officially launched in 2004. A team launched this service in Cape Town, South Africa.
- 2006: AWS (Amazon Web Services) was officially launched.
- 2007: In 2007, over 180,000 developers had signed up for the AWS.
- 2010: In 2010, amazon.com retail web services were moved to the AWS, i.e., amazon.com is now running on AWS.
- 2011: AWS suffered from some major problems. Some parts of volume of EBS (Elastic Block Store) was stuck and were unable to read and write requests. It took two days for the problem to get resolved.
- 2012: AWS hosted a first customer event known as re:Invent conference. First re:invent conference occurred in which new products were launched. In AWS, another major problem occurred that affects many popular sites such as Pinterest, Reddit, and Foursquare.
- 2013: In 2013, certifications were launched. AWS started a certifications program for software engineers who had expertise in cloud computing.
- 2014: AWS committed to achieve 100% renewable energy usage for its global footprint.

Definition

AWS stands for Amazon Web Services. Amazon Web Services (AWS) is widely used secure cloud services platform, offering computing power, content delivery, database storage, and other functionality to help businesses scale and grow.

The AWS service is provided by the 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.

Features of AWS:

Flexibility:

- The flexibility of AWS allows us to choose which programming models, languages, and operating systems are better suited for their project, so we do not have to learn new skills to adopt new technologies
- Flexibility means that migrating legacy applications to the cloud is easy, and cost-effective. Instead of re-writing the applications to adopt new technologies, you just need to move the applications to the cloud and tap into advanced computing capabilities.

Cost-effective:

- Cost is one of the most important factors that need to be considered in delivering IT solutions.
- For example, developing and deploying an application can incur a low cost, but after successful deployment, there is a need for hardware and bandwidth. Owing our own infrastructure can incur considerable costs, such as power, cooling, real estate, and staff

Scalable and elastic:

- Scalability in aws has the ability to scale the computing resources up or down when demand increases or decreases respectively
- Elasticity in aws is defined as the distribution of incoming application traffic across multiple targets such as Amazon EC2 instances, containers, IP addresses, and Lambda functions.

Secure:

- AWS provides a scalable cloud-computing platform that provides customers with end-to-end security and end-to-end privacy.
- AWS maintains confidentiality, integrity, and availability of your data which is the utmost importance of the aws.

Experienced:

- The AWS cloud provides levels of scale, security, reliability, and privacy.
- Nowadays, Amazon has become a global web platform that serves millions of customers, and AWS has been evolved since 2006, serving hundreds of thousands of customers worldwide.

Advantages 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 prototype.
- A media company can use the AWS to provide different types of content such as box or audio files to the worldwide files.

Global Architecture

When it comes to choosing a service we need to check the service availability in the desired location. AWS is a global leader when we talk about cloud service providers. It is well known for its availability zones. AWS cloud spans 44 availability zones with 18 geographical regions and one local region around the world. Each of the AWS regions has multiple availability zones and data centers. Many more availability zones are a work in progress and you can see that in the future.

Consistency & Reliability

AWS is a useful platform to keep the backup of your data. Reliability is also the best selling point of AWS which can be achieved through multiple backups of servers at multiple physical locations. According to a report in 2015 AWS had the minimum downtime. AWS has been "far better at keeping its public cloud service running than either Microsoft or Google." To write a reliable and stable application in AWS you need to learn about the AWS consistency models on AWS.

Scheduling

Scheduling means you can start and stop AWS services at predetermined times which means you can schedule your services to run at a particular time or on a particular day or even when a certain event takes place

PaaS Offerings

It duplicates alike methods in accomplished services for data warehousing, backup, database, transcoding, caching, application management, storage, and infrastructure management. This is very helpful in decreasing the total time and energy consumed in setting-up & managing the infrastructure. This helps decrease the overall cost of the market

Customization

AWS platform allows it's user high level of customization to meet the requirement of individual businesses. Customization option such as customer-defined tagging allows users to effortlessly monitor and handle specific resources. AWS tags can be utilized for cost tracking, security and automation as well.

Pay-As-You-Go

Based on the concept of Pay-As-You-Go, AWS provides the services to the customers.

AWS provides services to customers when required without any prior commitment or upfront investment. Pay-As-You-Go enables the customers to procure services from AWS.

- **Computing**
- **Programming models**
- **Database storage**
- **Networking**

Disadvantages of AWS

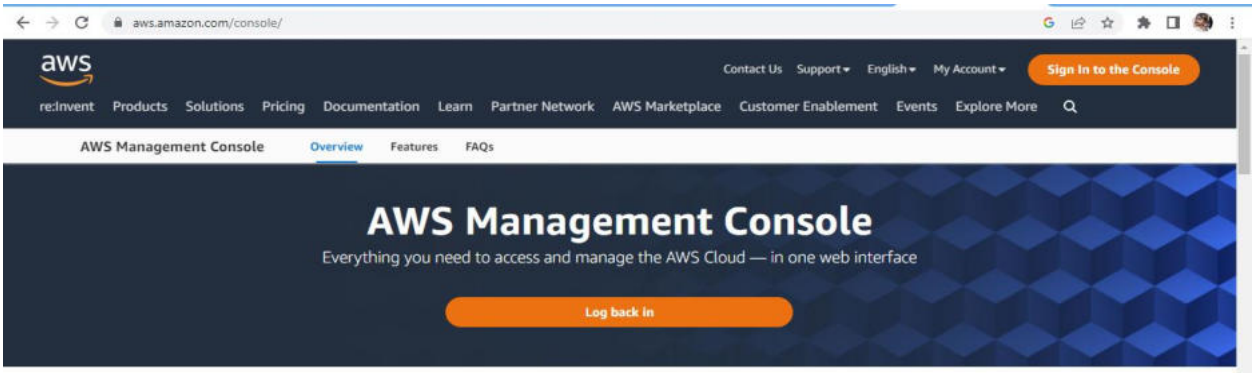
- If you need more immediate or intensive assistance, you'll have to opt for paid support packages.
- Amazon Web Services may have some common cloud computing issues when you move to a cloud. For example, downtime, limited control, and backup protection.
- AWS sets default limits on resources which differ from region to region. These resources consist of images, volumes, and snapshots.
- Hardware-level changes happen to your application which may not offer the best performance and usage of your applications.

AWS account creation and free tier limitations and overview

Following are the steps to access AWS services –

- Sign up using your email address
- Open the Amazon Web Services (AWS) home page.
- Choose Create an AWS Account.
- Enter your email address, password, AWS account name, and then choose Continue. Be sure that you enter your account information correctly, especially your email address.

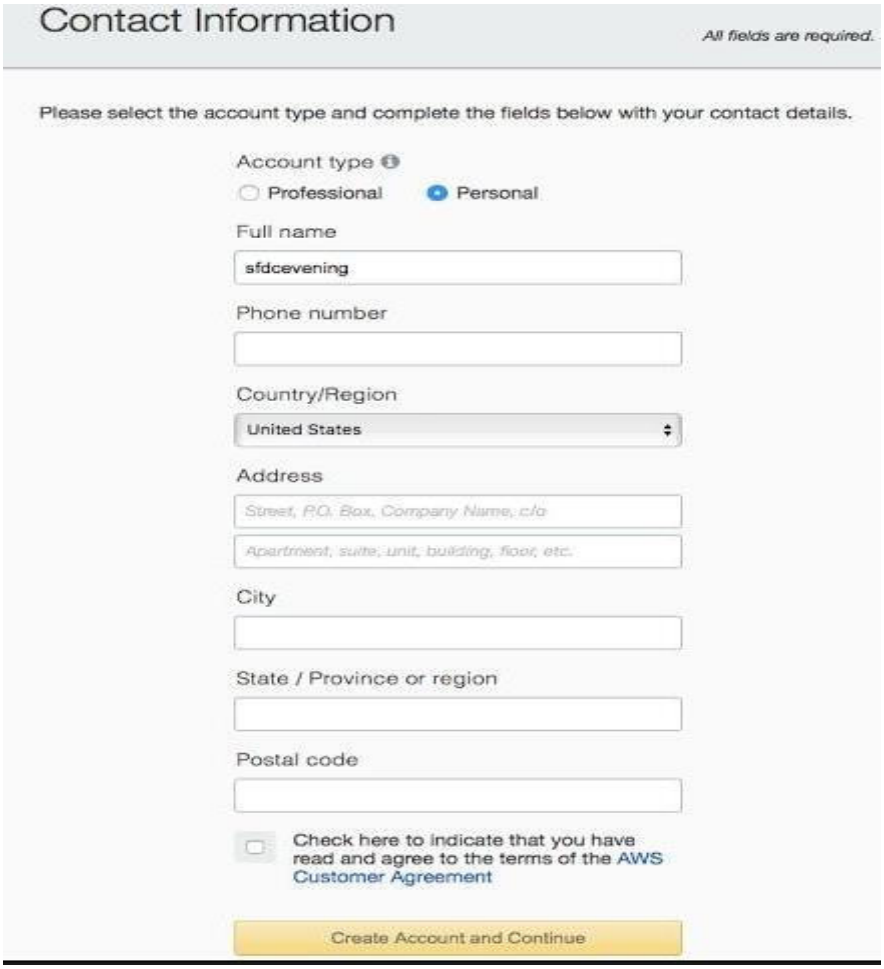
The following screen appears after opening the website, then click on the Complete Sign Up to create an account and fill the required details.

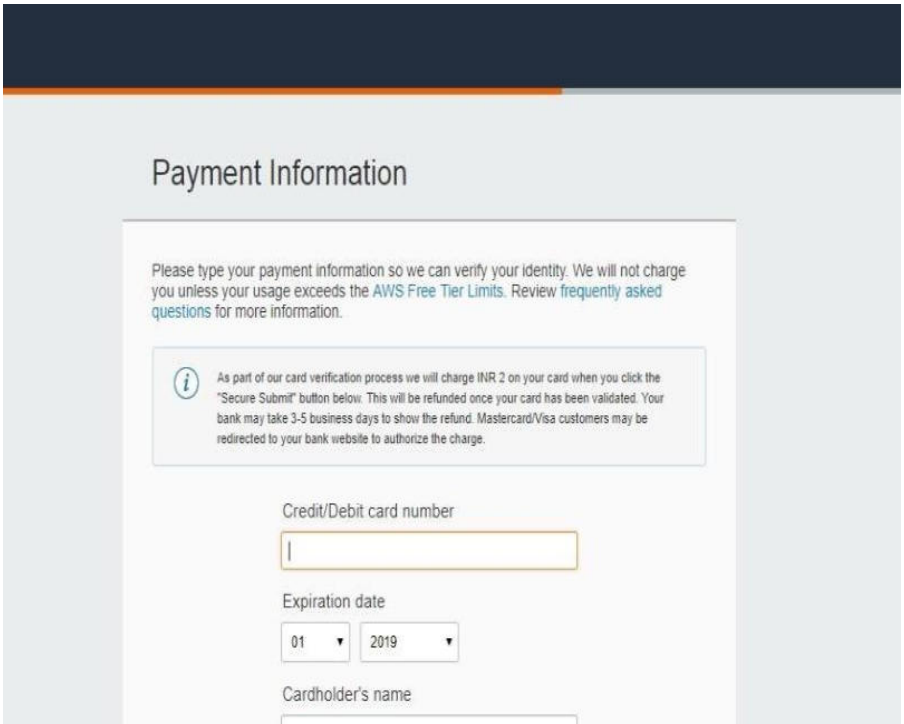


The following screen appears after clicking on the “sign in to the console” button. If you are an already existing user of an AWS account, then enter the email address of your AWS account otherwise “create an aws account”.

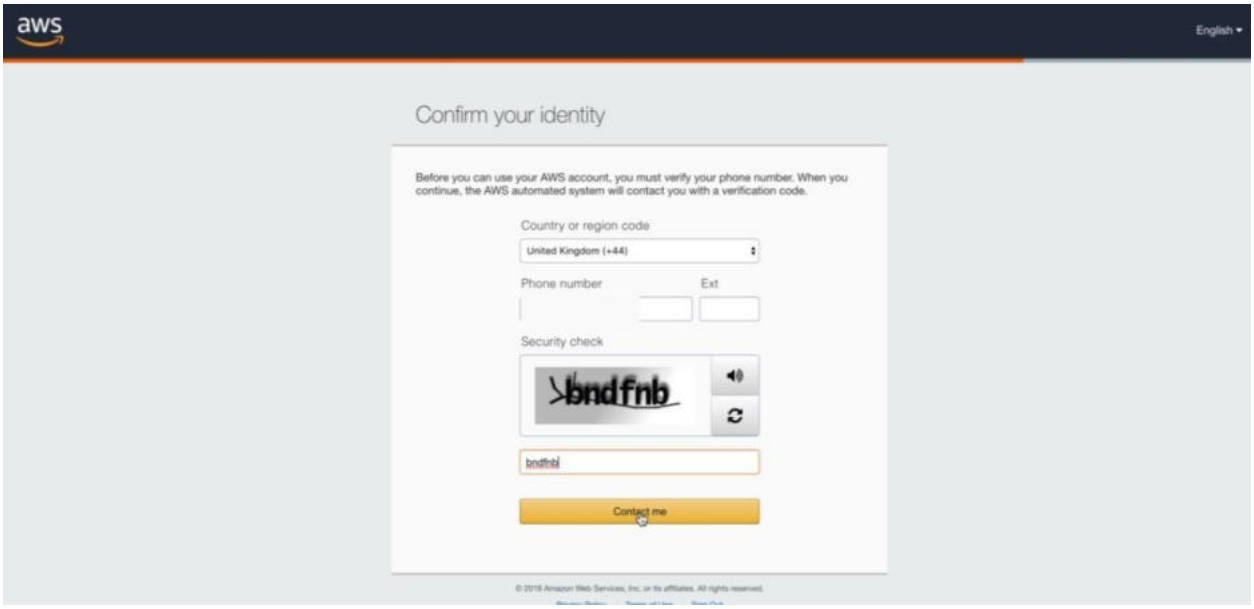
On clicking on the “create an aws account” button, the following screen appears that requires some fields to be filled by the user.

After providing the information, fill your contact information.





After providing your payment information, confirm your identity by entering your phone number and security check code, and then click on the “Contact me” button.



- AWS will contact you to verify whether the provided contact number is correct or not.
- When number is verified, then the following message appears on the screen.
- The final step is the confirmation step. Click on the link to log in again; it redirects you to the "Management Console".

AWS Free Tier limitations

The AWS Free Tier provides customers the ability to explore and try out AWS services free of charge up to specified limits for each service. The Free Tier is comprised of three different types of offerings, a 6-month Free Tier, an Always Free offer, and short term trials. Services with a 6-month Free Tier allow customers to use the product for free up to specified limits for one year from the date the account was activated. Services with an Always Free offer allow customers to use the product for free up to specified limits as long as they are an AWS customer. Services with a short term trial are free to use for a specified period of time or up to a one-time limit depending on the service selected. Details on the limits and services provided for free are detailed in each card on the Free Tier page. If your application use exceeds the free tier limits, you simply pay standard, pay-as-you-go service rates

All services that offer AWS Free Tier have limits on what you can use without being charged. Many services have multiple types of limits. For example, Amazon EC2 has limits on both the type of instance you can use and how many hours you can use in one month. Amazon S3 has a limit on how much storage you can use and on how often you can call certain operations each month. For example, the AWS Free Tier covers the first 20,000 times you retrieve a file from Amazon S3, but you're charged for additional file retrievals. Each service has limits that are unique to that service.

Some of the most common limits are by time, such as hourly or by the minute, or by requests, which are the requests you send to the service, also known as API operations.

Hourly usage in the AWS Free Tier

Some services, such as Amazon EC2, Amazon RDS, and Elastic
Load Balancing, charge for usage on an hourly basis. The AWS Free Tier for these services provides you with a monthly allotment of hours for the first 12 months. For example, the AWS Free Tier for Amazon EC2 provides you with 750 hours usage of Linux (any combination of t1.micro, t2.micro, and t3.micro instances), plus 750 hours usage of Windows (any combination of t1.micro, t2.micro, and t3.micro instances). How you divide this allotment is up to you. In this example, you can run 750 hours of a Linux t2.micro, or t1.micro instance with 750 hours of a Windows t2.micro, or t1.micro instance each month for the first 12 months. In Regions where t2.micro is not available, the t3.micro equivalent is supported under AWS Free Tier. For example, you can use one Linux instance continuously for a month or 10 Linux instances for 75 hours a month.

- 750 hours of an Elastic Load Balancer plus 15 GB data processing
- 750 hours of Amazon RDS Single-AZ Micro DB Instances, running MySQL, MariaDB, PostgreSQL
- 750 hours of Amazon ElastiCache Micro Cache Node usage – enough hours to run continuously each month.
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with EBS Magnetic) and 1 GB of snapshot storage
- 5 GB of Amazon S3 standard storage, 20,000 Get Requests, and 2,000 Put Requests
- 25 GB of Storage, 25 Units of Read Capacity and 25 Units of Write Capacity, enough to handle up to 200M requests per month with Amazon DynamoDB
- 25 Amazon SimpleDB Machine Hours and 1 GB of Storage
- 100,000 Requests of Amazon Simple Queue Service
- 100,000 Requests, 100,000 HTTP notifications and 1,000 email notifications for Amazon Simple Notification Service
- 10 Amazon Cloud watch metrics, 10 alarms, and 1,000,000 API requests

Ec2 Service, Launching an EC2 instance

EC2: Amazon EC2 (Elastic Compute Cloud) is a web service interface that provides resizable compute capacity in the AWS cloud. It is designed for developers to have complete control over web-scaling and computing resources. EC2 instances can be resized and the number of instances scaled up or down as per our requirement. These instances can be launched in one or more geographical locations or regions, and Availability Zones (AZs).

METHOD-I

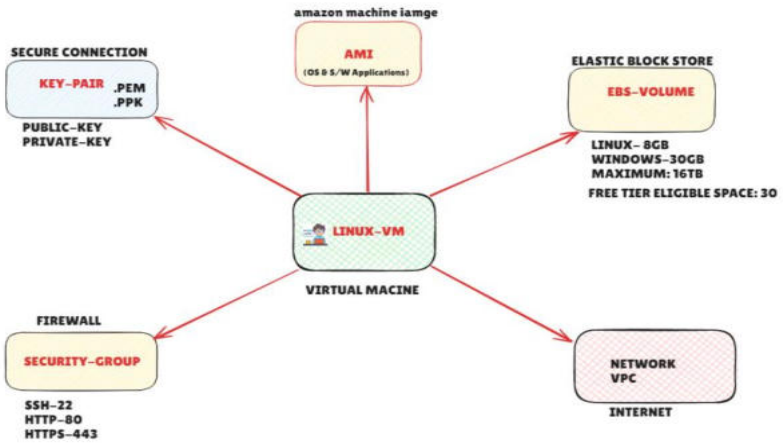
You can launch an EC2 instance using the AWS Management Console as described in the following procedure

- Open the Amazon EC2 console at **https://console.aws.amazon.com/ec2**
- In the navigation bar at the top of the screen, we display the current AWS Region
- From the EC2 console dashboard, in the Launch instance pane, choose **Launch instance**.
- Under **Name and tags**, for Name, enter a descriptive name for your instance.
- Under Application and OS Images (Amazon Machine Image)

1. Choose Quick Start, and then choose the operating system (OS) for your instance
2. From **Amazon Machine Image (AMI),** select an AMI that is marked **Free Tier eligible**.

- Under Instance type, for Instance type, choose **t2.micro**, which is eligible for the Free Tier
- Under Key pair (login), for Key pair name, choose an existing key pair or choose Create new key pair to create your first key pair
- Under Network settings, notice that we selected your default VPC, selected the option to use the default subnet in an Availability Zone that we choose for you, and configured a security group with a rule that allows connections to your instance from anywhere
- Under **Configure storage**, notice that we configured a root volume but no data volumes.
- **Review a summary** of your instance configuration in the Summary panel, and when you're ready, choose Launch instance.

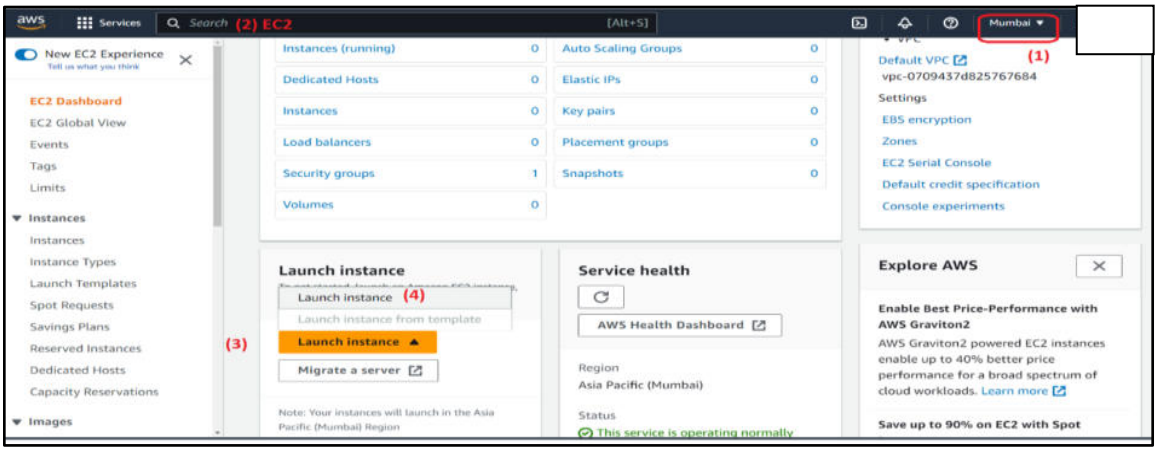
1.



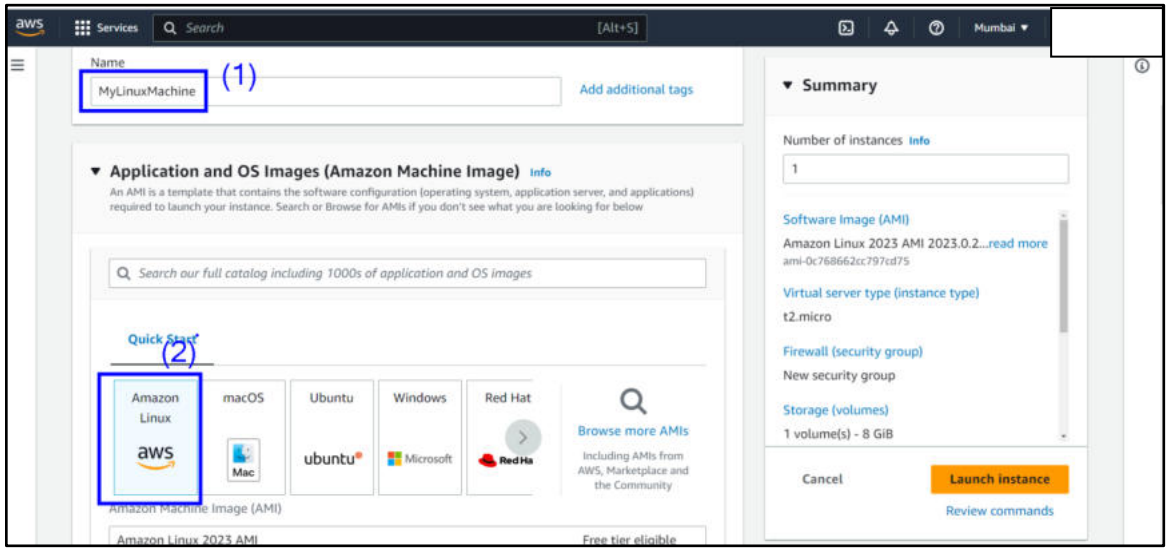
Goto:https://aws.amazon.com/console/ => click on login

2. Choose Region based on your location. Ex: For India: (Mumbai) ap-south-1

3. Click on launch Instance and choose Launch Instance

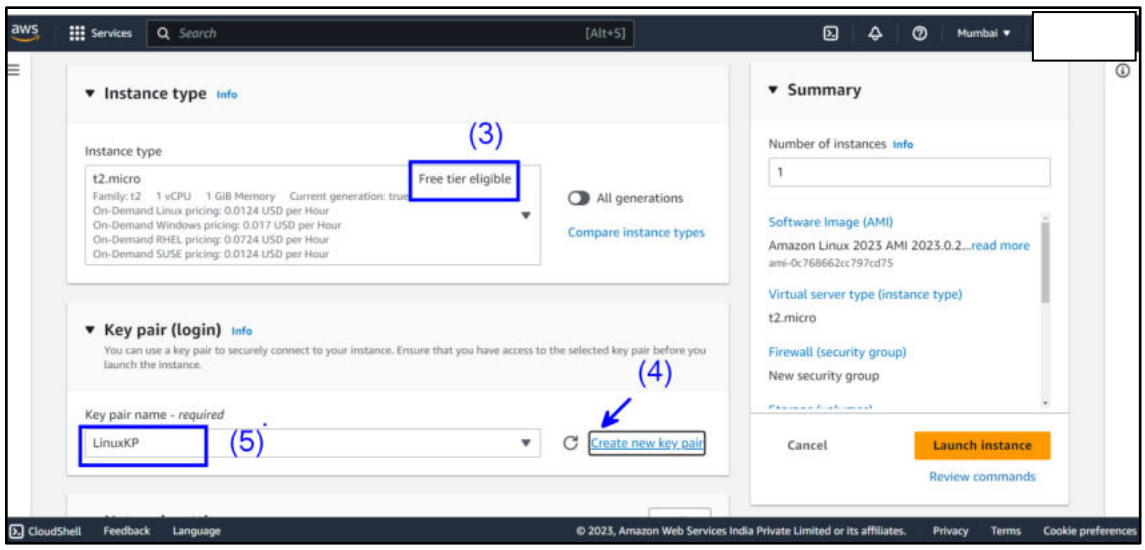


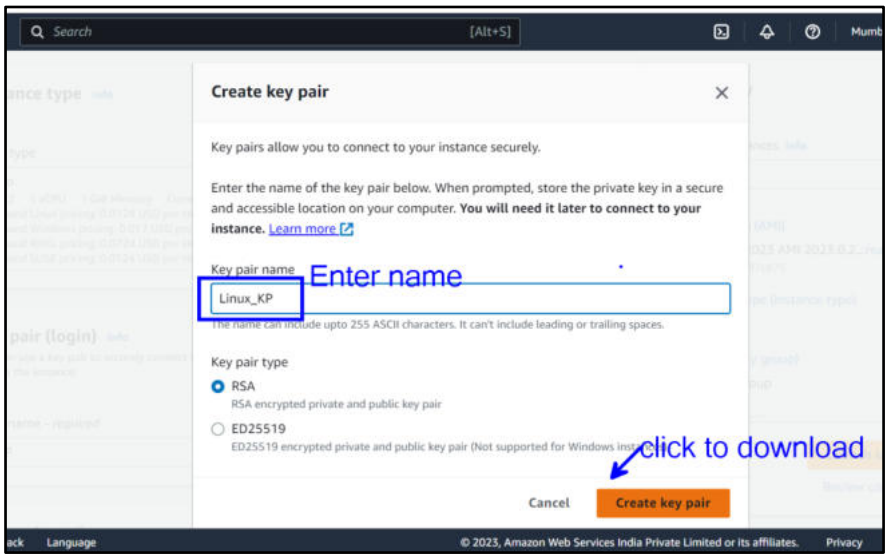
4. Enter any name for easy identity and choose Amazon Linux (AMI)



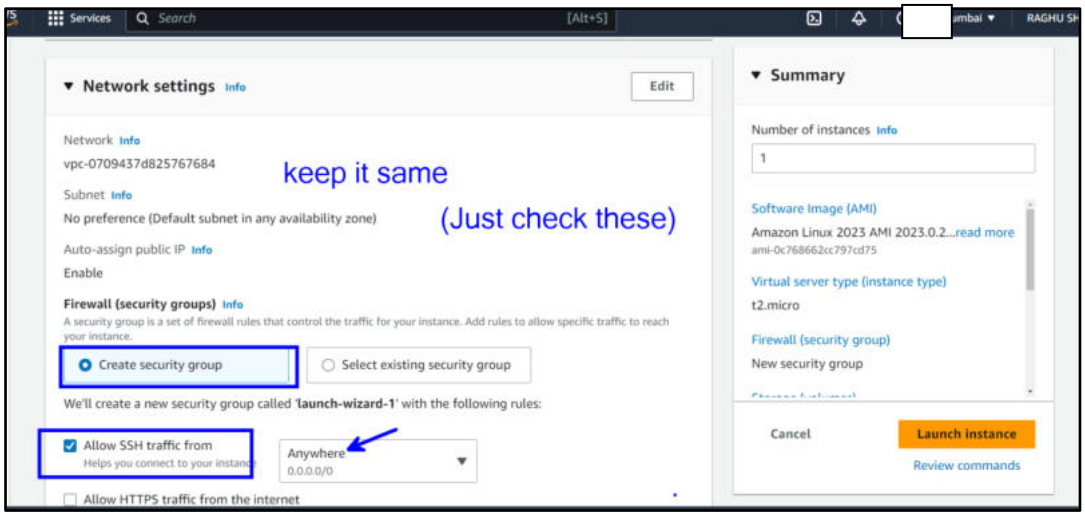
5. Please select “Free tier eligible” to avoid billing issue.

6. Click on create new Key new pair for creating a .PEM file (Privacy Enhanced Mail / Secret Key)

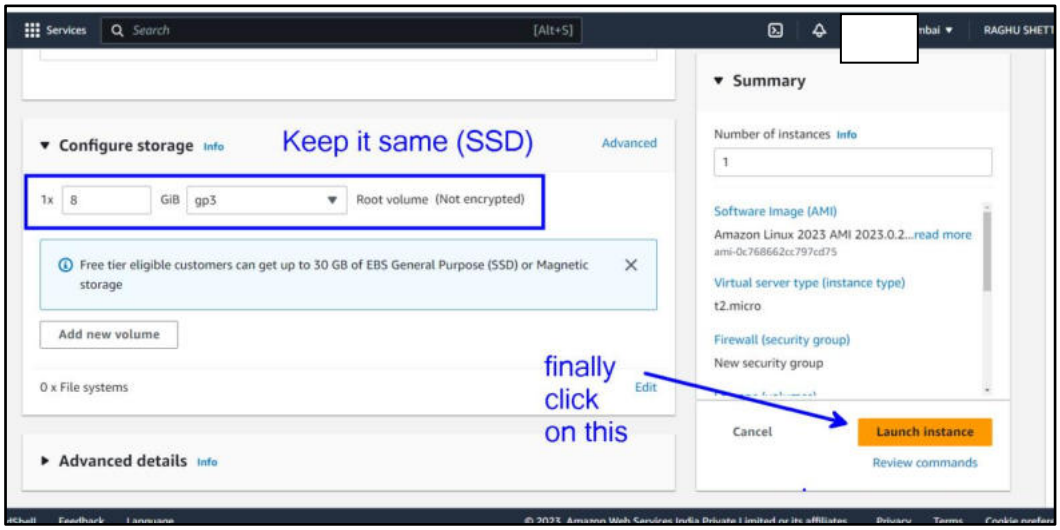




7. Keep networking details as defaults/same as selected options

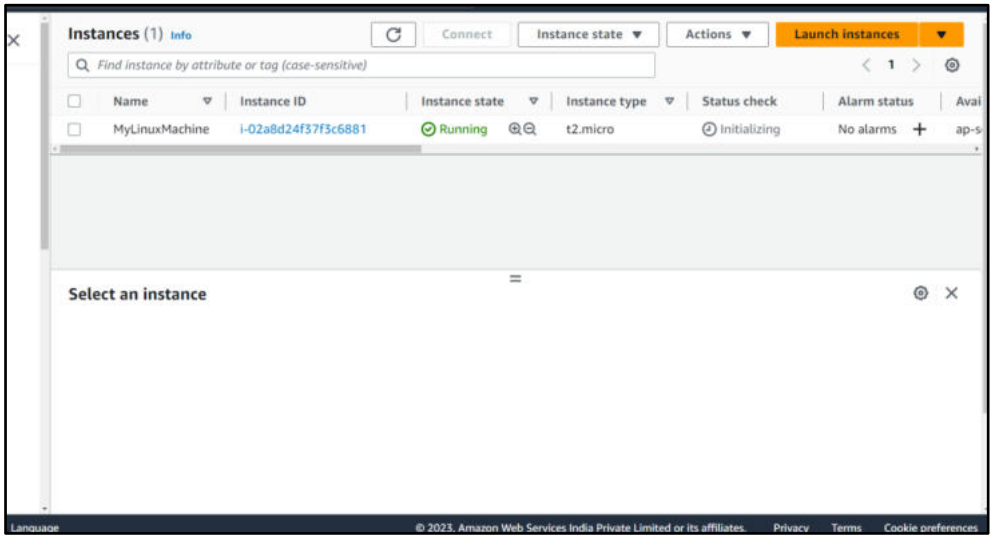


8. Keep Configure storage details as defaults

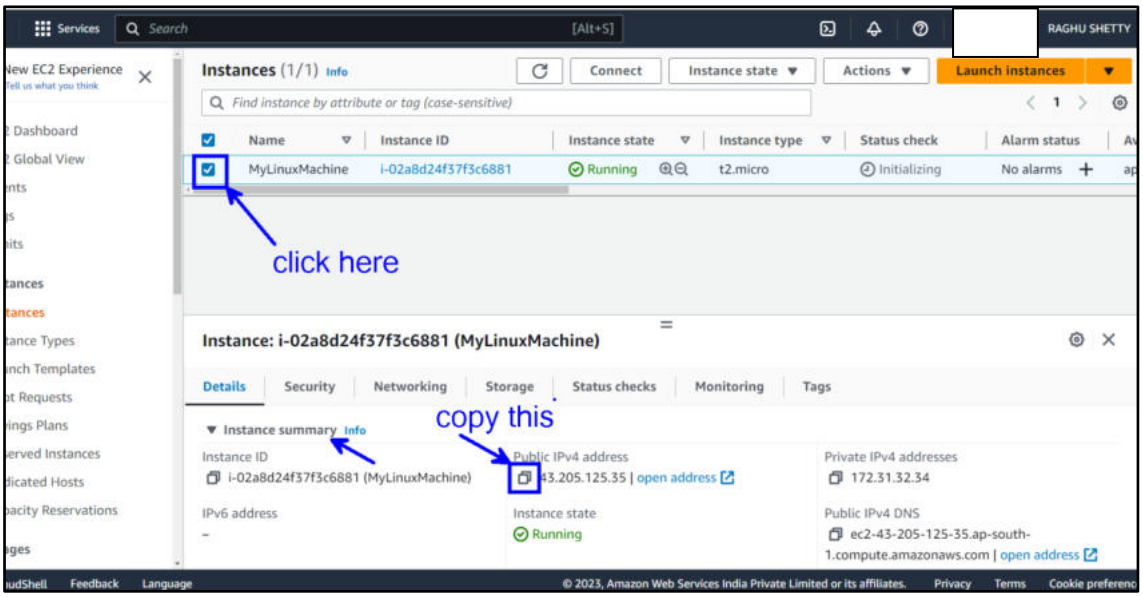


9. Wait for few seconds to get EC2 Instance created then click on ID (Showing on success page)

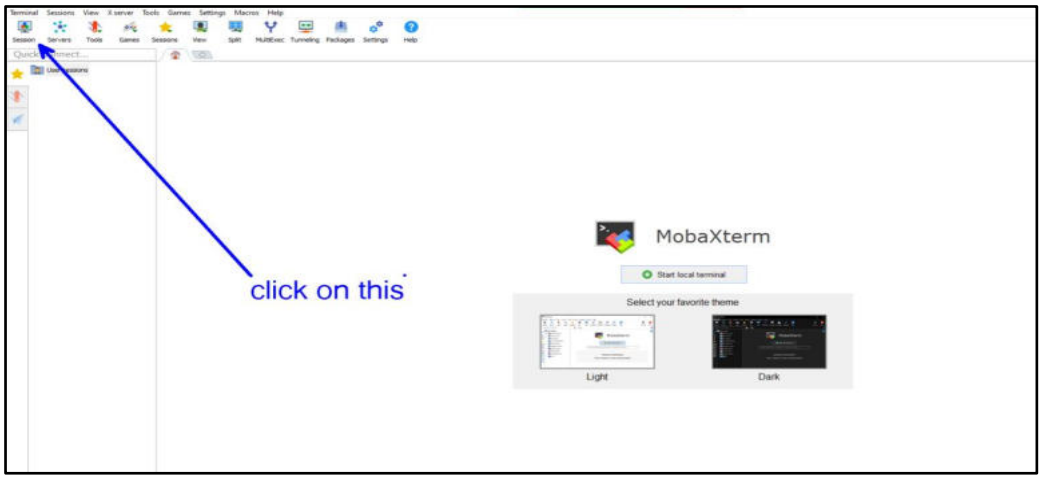
10. Refresh to see Instance state updated (Should be in Running)



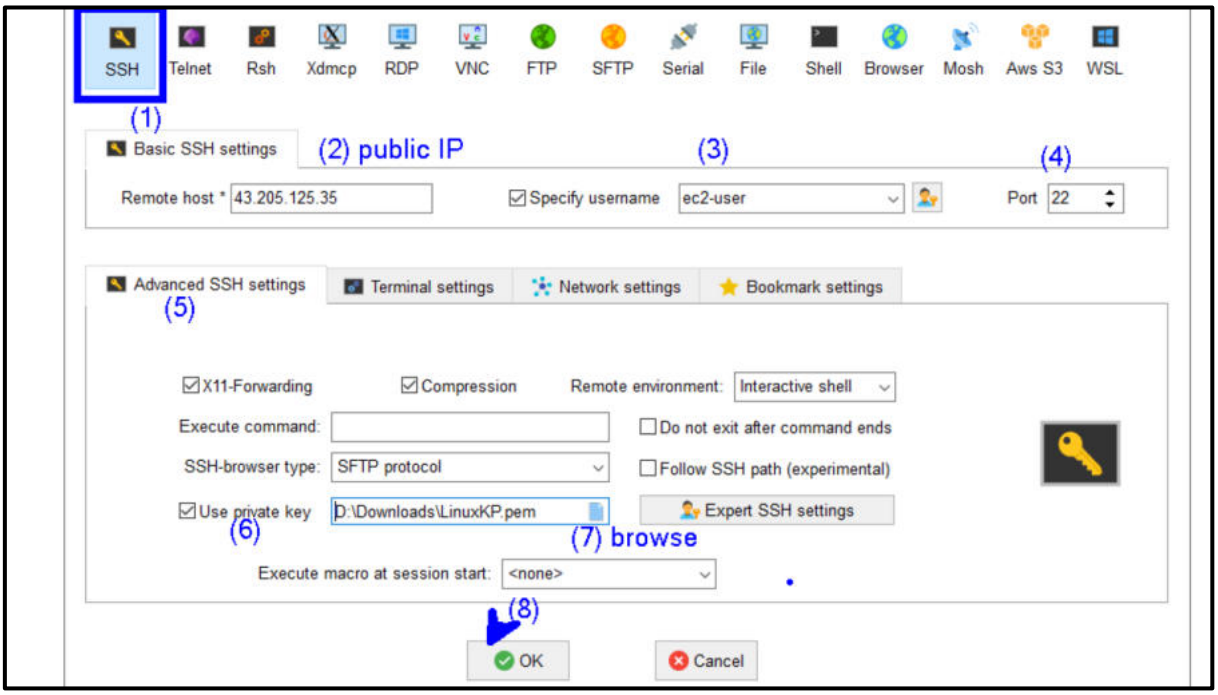
11. Choose checkbox of our instance and copy Public IP



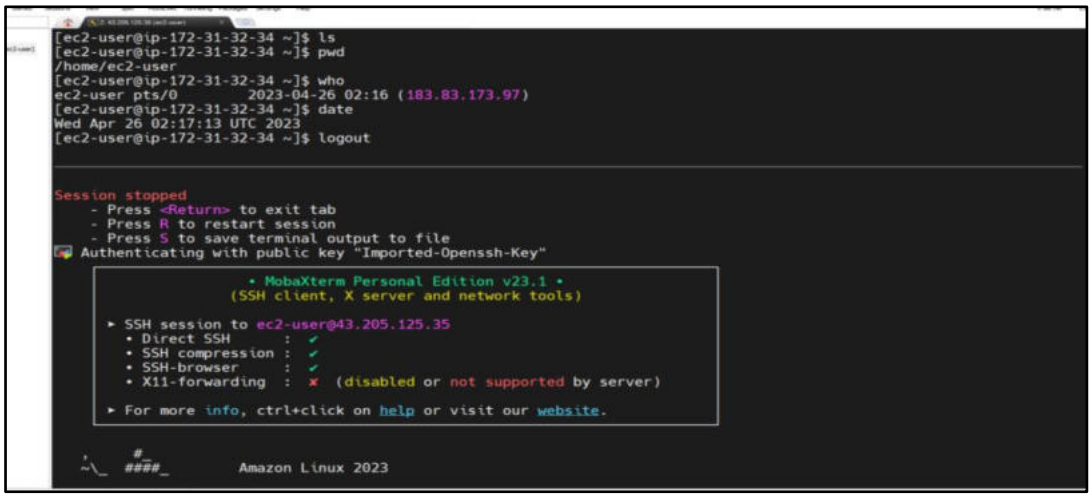
12. Open MobaXTerm and click on Session option



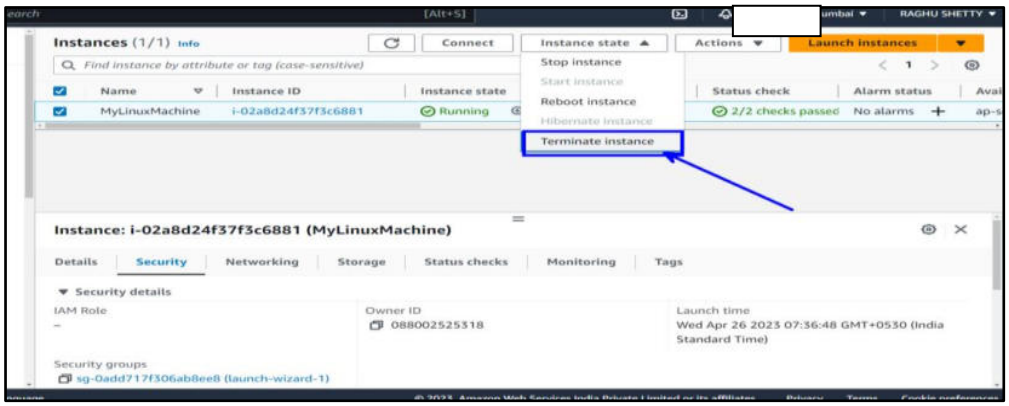
13. Select SSH and fill details also select .PEM file



14. Type commands on terminal (ls, clear, pwd, who, date, logout)



15. **Delete EC2 Instance using Instance State ->Terminate**



METHOD-II

Bootstrapping allows us to write and put a startup script while launching an EC2 Instance so that it execute automatically as soon as the instance launch.

- Create **Linux EC2** Machine
- Select **AMI** (Amazon Linux)
- Choose an **Instance Type**
- Configure **Instance Type**
- Create security Group
- In Advanced Details Add Code

```
#!/bin/bash
sudo su
yum update -y
yum install httpd –y
cd /var/www/html
echo "ANNAMACHARYA UNIVERSITY RAJAMPET" > index.html
service httpd start
chkconfig httpd on
```

- Add Tags
- Configure Security Group Click on Add Rule

- Change type as HTTP & Source Anywhere
- Review Instance Launch
- Click on launch
- Copy the Public IP and Paste in browser

Types of EC2 INSTANCES:

- General Purpose Instances
- Compute Optimized Instances
- Memory-Optimized Instances
- Storage Optimized Instances
- Accelerated Computing Instances

1. General-Purpose Instances

The computation, memory, and networking resources in general-purpose instances are balanced. Scenarios, where you can use General Purpose Instances, are gaming servers, small databases, personal projects, etc. Assume you have an application with a kind of equal computing, memory, and networking resource requirements. Because the program does not require optimization in any particular resource area, you can use a general-purpose instance to execute it.

2. Compute-Optimized Instances

Compute-optimized instances are appropriate for applications that require a lot of computation and help from high-performance CPUs. You may employ compute-optimized instances for workloads including web, application, and gaming servers just like general-purpose instances. This instance type is best suited for high-performance applications like web servers, Gaming servers.

3. Memory-Optimized Instances

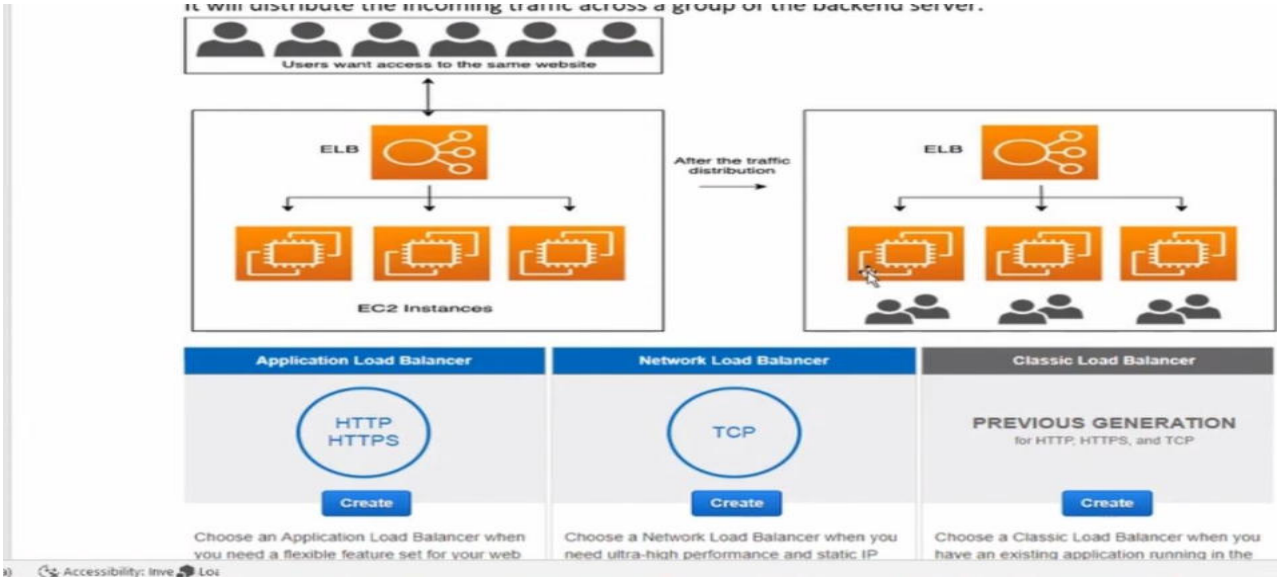
Memory-optimized instances are geared for workloads that need huge datasets to be processed in memory. Memory here defines RAM which allows us to do multiple tasks at a time. Data stored is used to perform the central processing unit (CPU) tasks it loads from storage to memory to run. This process of preloading gives the CPU direct access to the computer program

4. Storage Optimized Instances

Storage-optimized instances are made for workloads that demand fast, sequential read and write access to huge datasets. Distributed file systems, data warehousing applications, and high-frequency online transaction processing (OLTP) systems are examples of workloads that are suited for storage-optimized instances. Storage-optimized instances are built to provide applications with the lowest latency while accessing the data.

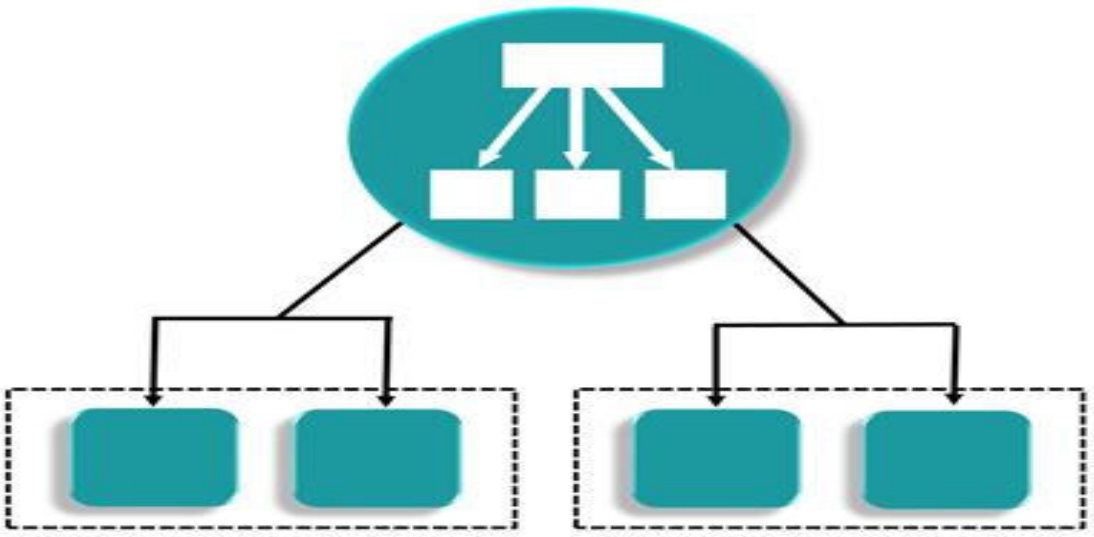
Elastic Load Balancer and types of ELB

Load Balancer is a virtual machine or appliance that balances your web application load that could be Http or Https traffic that you are getting in. It balances a load of multiple web servers so that no web server gets overwhelmed.



Application Load Balancer:

- An Amazon Web Services (AWS) launched a new load balancer known as an Application load balancer (ALB) on August 11, 2016.
- It is used to direct user traffic to the public AWS cloud.
- It identifies the incoming traffic and forwards it to the right resources.
- It is operated at Layer 7 of the OSI Model.
- It is best suited for load balancing of HTTP and HTTPs traffic.
- Application load balancers are intelligent, sending specific requests to specific web servers.



Need For Load Balancing

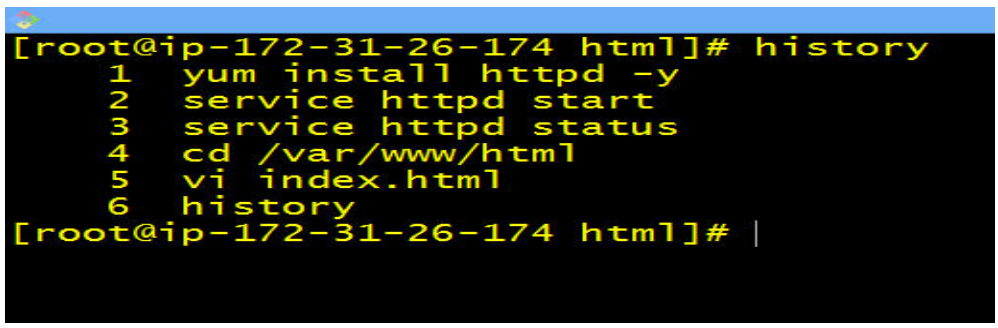
The main use for an Elastic Load Balancer is to make your resources highly available. There are four types of forwarding rules that a user can assign to a load balancer.

- **HTTP**
  It is the standard balancing request based on HTTP mechanism. It directs the traffic to the backend information for the original request.
- **HTTPS**
  It is the same as the HTTP forwarding rule with an additional feature of encryption.
- **UDP**
  A UDP Load Balancer uses User Datagram protocol which provides low latency
- **TCP**
  TCP is used for Load balancing all the applications that do not use protocols like HTTP and HTTPS

PROCEDURE

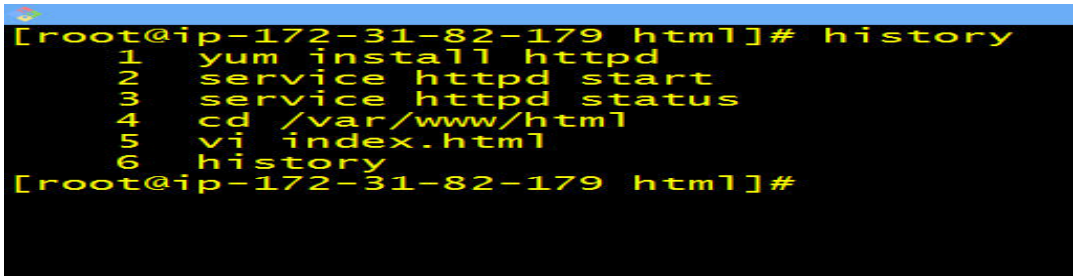
I will create 2 EC2 instances and will make them work like servers. I will then host a Web page on each of them. By using a load balancer, I will handle the request to my web page.

- Go to compute service in your AWS Console. Select on Launch Instance.
- Select the type of Instance you want
- Select the type of storage you want for your Instance.
- Select the VPC and subnet where you want your instance.
- Select the capacity of your Instance.
- You can add tags and names to identify your Instance.
- Configure your security group to allow access to the range of audience.
- Take a last review for the instance settings and Click on Launch.
- It will take 2-5 minutes and your instance will be ready.
- Open the putty.

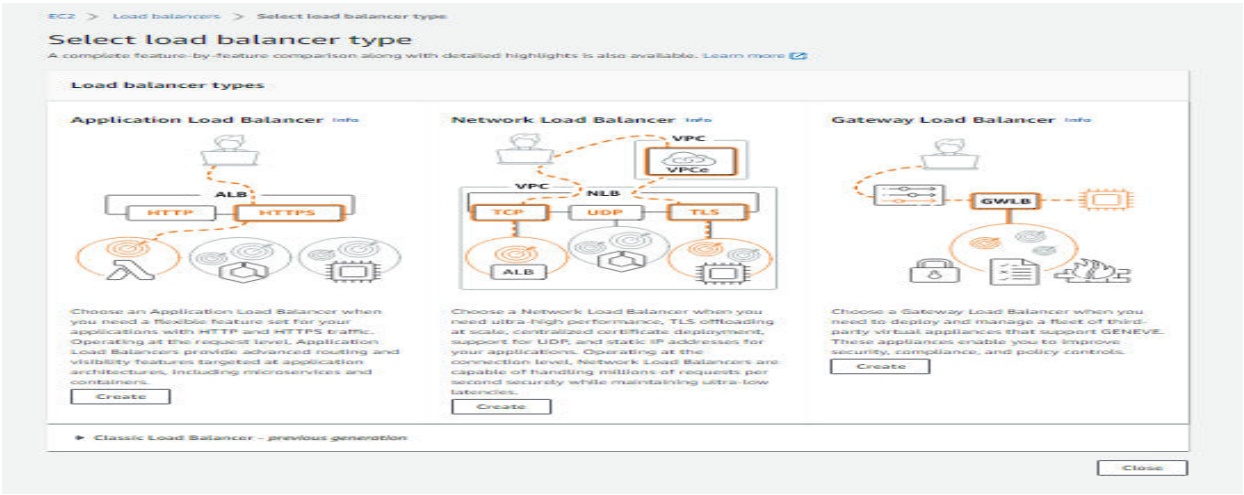


- Run the command **sudo su** to provide the privileges to the root device.
- Run the command **yum update -y** to update the EC2 instance.
- Install the Apache server by using the command **yum install httpd -y.**
- Start the server by using the command service httpd start.
- Move to the html directory by running the command **cd /var/www/html.**
- Run the command **vi index.html** to create the editor

- Same as creating second instance also



- Click on the Create Load Balancer. On clicking, Four types of Load Balancers are shown:



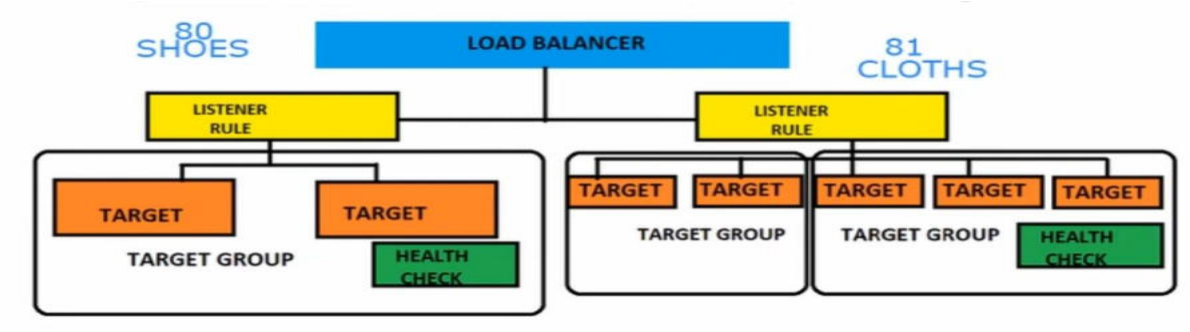
- We create a **Application Load Balancer**.
- On clicking on the **create** button
- **Load Balancer name:** It is the name of the Load balancer that the user provides. Suppose I have given a MyALB as a load balancer name.

Linear Configuration: It describes from which protocol and port, it is listening, and to which port it is passing.

- Click on the **Next** button.
- Configure Health check.
- **Ping Protocol:** It defines the type of protocol.

- **Ping port:** It defines the port number.
- **Ping Path:** It defines the path of the web page that we created, i.e., **healthcheck.html**
- **Response Timeout:** It defines how long it will take and waits for the response.
- **Interval:** It is the amount of time between health checks.
- Now, just add your instance, i.e. "***Register Targets***".

Creating Target Group:

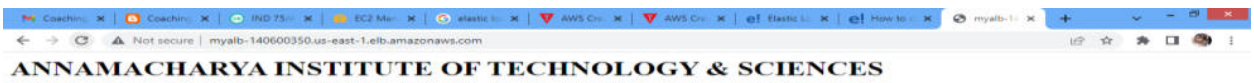


- In the navigation pane, under Load Balancing, choose Target Groups.
- Choose Create target group.
- Under Basic configuration, keep the Target type as instance.
- For Target group name, enter a name for the new target group.
- Keep the default protocol (HTTP) and port (80).
- Select the VPC containing your instances. Keep the protocol version as HTTP1.
- For Health checks, keep the default settings.
- Choose Next.
- On the Register targets page, complete the following steps
- For Available instances, select atleast two instances.
- Keep the default port 80, and choose Include as pending below.
- Choose Create target group.
- The final step "***Review***" all the settings

Test your load balancer

- In the navigation pane, under Load Balancing, choose Load Balancers.
- Select the newly created load balancer.
- Choose Description and copy the DNS name of the load balancer
- Paste the DNS name into the address field of an internet-connected web browser. If everything is working, the browser displays the default page of your server.

SAMPLE OUTPUT



Introduction to Linux

- It is an operating system, by using that, users/applications can communicate with hardware components.
- It was developed/created in 1960s.
- With lot of extensions and improvements to base version, several flavors introduced by organization/companies (flavors like Red hat Linux, Ubuntu, CentOS etc.)

Features of Linux:

- It is FOSS (Freeware and Open Source Software)
- UNIX can be used by multiple users simultaneously and hence it is Multi User operating System.
- Several tasks can be executed simultaneously and hence it is multi-tasking operating system.
- It is user friendly and provides both CUI and GUI Support.
- When compared with windows, UNIX is more secured.

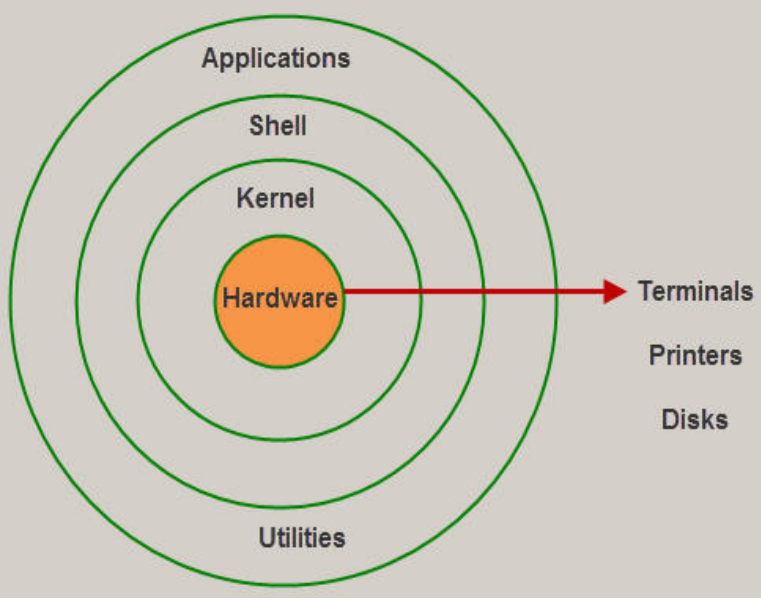
Flavors of Linux:

As UNIX is open source, multiple flavors are available with lot of extensions and improvements.

- Ubuntu
- RedHat
- Centos
- Fedora
- Slackware

All these flavors have lot of similarity. Hence if we are perfect with one flavor, we can work on any other flavor very easily.

Linux architecture



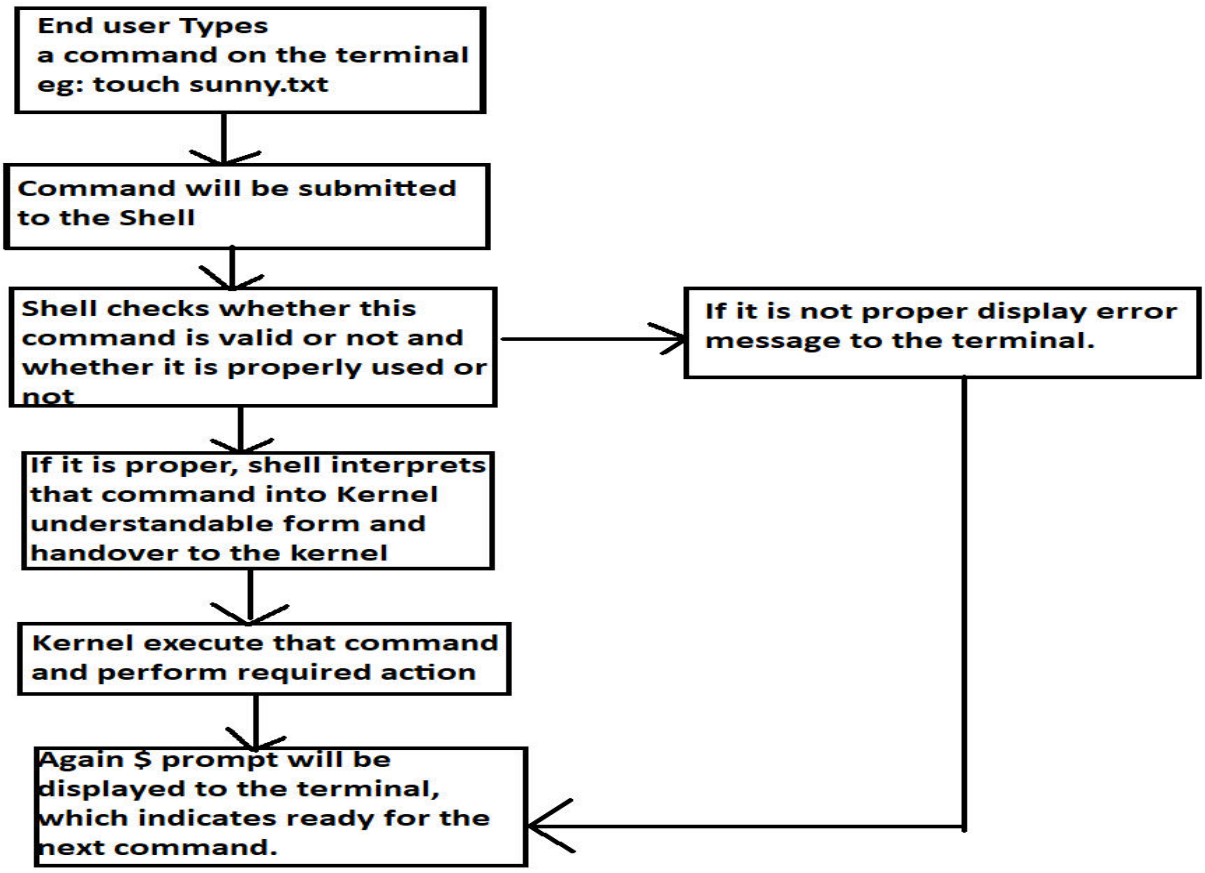
Shell:

- It is the outer layer of UNIX operating System.
- It reads our command, verify syntax and check whether the corresponding command related application is available or not.
- If everything is proper, then shell interprets our command into kernel understandable form and handover to the kernel.
- Shell acts as interface between user and kernel.

Kernel:

- It is the core component of LINUX operating system.
- It is responsible to execute our commands.
- It is responsible to interact with hardware components.
- Memory allocation and processor allocation will takes care by kernel

Command Execution Flow:



- User types the command in the terminal.

  touch python.txt

- Shell reads that command. It will check whether that command is valid or not and whether it is used properly or not. If everything is proper, then shell interprets/translates that command into kernel understandable form.

- Shell handovers that interpreted command to the kernel.
- Kernel executes that command and perform required activity.
- Once command execution completed, then shell returns unix prompt ($ OR # OR %).
- $ OR # OR % represents it is ready for the next command.

File types:

Regular Files (-)

These are the most common files — containing data, text, programs, images, etc.

- Text file: notes.txt
- Binary file: /bin/ls

Directory Files (d)

- A directory is a special file that holds other files or directories (like folders in Windows).
- Example: /home, /etc, /var/log

Symbolic Link Files (l)

- A symbolic link (or symlink) is like a shortcut or reference to another file or directory.
- Example: /etc/rc3.d/S20network → ../init.d/network

Character Device Files (c)

- Used to represent devices that handle data character by character (like keyboards, serial ports, terminals).
- Example: /dev/tty, /dev/random, /dev/null

Block Device Files (b)

Used to represent devices that handle data in blocks (like hard drives, USBs, etc.).

- Example: /dev/sda, /dev/sdb1

Socket Files (s)

Used for network communication between processes (e.g., client-server communication).

- Example: /var/run/docker.sock, /tmp/.X11-unix/X0

Directories

- We can create directories by using mkdir command.
- mkdir dir1

To create a directory

- mkdir dir1 dir2 dir3

To create multiple directories

- mkdir dir1/dir2/dir3
- mkdir -p dir1/dir2/dir3

How to remove Directories:

We can remove directories by using rmdir command.

- rmdir dir1

To remove empty directory dir1

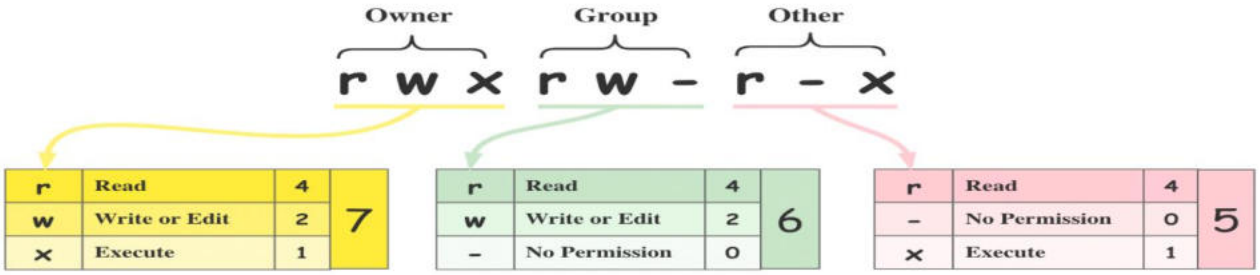
- $ rmdir dir1 dir2 dir3

File Permissions

File Permissions describe the allowed operations by various users.With respect to file permissions, all users are categorized into the following 4 types.

Permission Types: For files and directories, there are 4 types of permissions.

Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwx	Read + Write + Execute



Eg : $ chmod u+w,g+rw,o+r demo.txt
adding write permission to the user

adding read and write permissions to the group adding read permission to the others

Eg : $chmod u+x,g-w,o+w demo.txt adding execute permission to the user removing write permission from the group adding write permission to the others

Eg : $ chmod a=- demo.txt Now user permissions: --- group permission: --- others permission: ---

Eg : $ chmod a=rwx demo.txt Now user permissions: rwx group permission: rwx

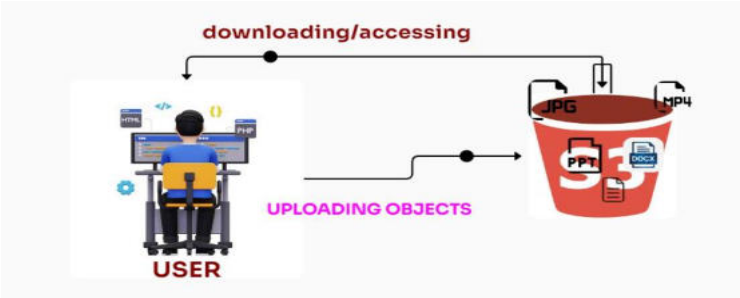
others permission: rwx

UNIT-II

UNIT II AWS SERVICES:, S3 Bucket- bucket creation, storage classes, S3 versioning, Bucket policies, Replication Rules in S3, IAM service-Groups, Users, Policies & Roles, AWS-CLI Commands, RDS, VPC, Route53, Monitoring tools

AWS SERVICES: S3 Bucket- bucket creation

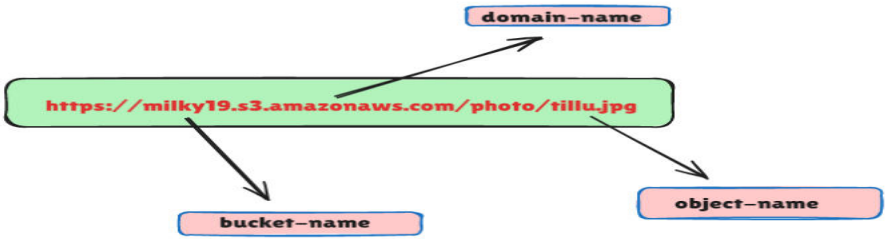
Amazon S3 (Simple Storage Service) is a scalable, high-speed, low-cost web-based service designed for online backup and archiving of data and application programs. It allows uploading, store, and downloading any type of files up to 5 TB in size. This service allows the subscribers to access the same systems that Amazon uses to run its own web sites. The subscriber has control over the accessibility of data, i.e. privately/publicly accessible.



Buckets

A bucket is a container for objects stored in Amazon S3. You can store any number of objects in a bucket and can have up to 100 buckets in your account. Every object is contained in a bucket.

For example, if the object named **photos/tillu.jpg** is stored in the amzn-s3-demo-bucket bucket in the N.Virgina Region, then it is addressable by using the URL **https://milky19.s3.amazonaws.com /tillu.jpg.**



When you create a bucket, you enter a bucket name and choose the AWS Region where the bucket will reside. After you create a bucket, you cannot change the name of the bucket or its Region. Bucket names must follow **the bucket naming rules**

- Bucket names must be between **3 (min) and 63 (max)** characters long.
- Bucket names can consist only of **lowercase letters, numbers, dots (.), and hyphens (-).**
- Bucket names must begin and end with a **letter or number.**
- Bucket names must not contain **two adjacent** periods.
- Bucket names must not be formatted as an **IP address** (for example, 192.168.5.4).

Objects

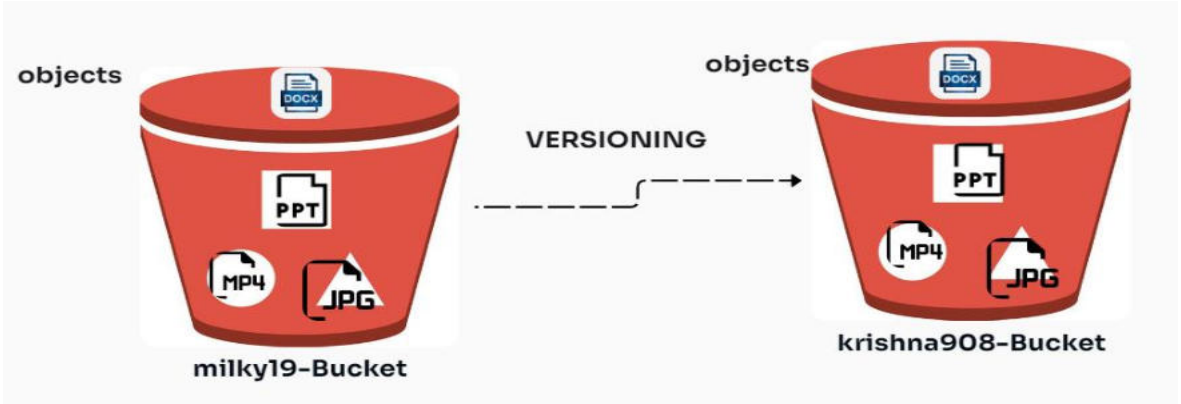
Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object. These pairs include some default metadata, such as the date last modified, and standard HTTP metadata, such as Content-Type. You can also specify custom metadata at the time that the object is stored.

Keys

An object key (or key name) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, object key, and optionally, version ID (if S3 Versioning is enabled for the bucket) uniquely identify each object. So you can think of Amazon S3 as a basic data map between "bucket + key + version" and the object itself.

S3 Versioning

You can use S3 Versioning to keep multiple variants of an object in the same bucket. With S3 Versioning, you can preserve, retrieve, and restore every version of every object stored in your buckets.



S3 BUCKET CREATION STEPS

- In the left navigation pane, choose Buckets.
- Choose **Create bucket**.

- The **Create bucket** page opens.
- Under General configuration, view the AWS Region where your bucket will be created.
- Under Bucket type, choose **General purpose**.
- For Bucket name, enter a name for your bucket.
- Under Object Ownership, to disable or **enable ACLs** and control ownership of objects uploaded in your bucket
- Under Block Public Access settings for this bucket, choose the **Block Public Access** settings that you want to apply to the bucket.
- (Optional) Under Bucket Versioning, you can choose if you wish to keep variants of objects in your bucket.
- Choose **Create bucket**.

UPLOADING OBJECTS INTO S3 BUCKETS:

- In the Buckets list, choose the name of the bucket that you want to upload your object to.
- On the Objects tab for your bucket, choose **Upload**.
- Under Files and folders, choose **Add files**.
- Choose a file to **upload**, and then **choose Open**.
- Choose **Upload.**

Deleting an objects:

- If you want to choose which objects you delete without emptying all the objects from your bucket, you can delete an object.
- In the Buckets list, choose the **name of the bucket** that you want to delete an **object** from.
- Select the **object** that you want to delete.
- Choose **Delete** from the options in the upper right.
- On the Delete objects page, type delete to **confirm deletion** of your objects.
- Choose **Delete objects**.

Deleting your bucket:

- After you empty **your bucket or delete all the objects** from your bucket, you can delete your bucket.
- To delete a bucket, in the Buckets list, **select the bucket.**
- Choose **Delete**.
- To **confirm deletion**, in Delete bucket, type the **name of the bucket**.

Storage classes

The different storage classes provided are:

- S3 Standard
- S3 Standard-IA
- S3 Intelligent-Tiering
- S3 One Zone-IA

- S3 Glacier
- S3 Glacier Deep Archive
- S3 Outposts

S3 Standard

S3 Standard is the default storage class if none of the storage class is specified during upload. It is ideal for frequently accessed data because it provides low latency and high availability. It has a wide range of use cases from cloud applications and web services, websites hosting, big data analytics, mobile gaming, and content distribution. It is the most expensive storage class among all others.

- High Availability and low latency
- Data is stored in multiple locations.
- The durability of 99.999999999% and availability of 99.99% availability
- Most expensive storage class among all others.

S3 Standard-IA

S3 Standard-Infrequent Access is optimized for long-lived and less frequently accessed data but requires rapid access whenever required. Similar to S3 Standard, it also offers high durability, low latency, and high throughput but has a low per GB storage price and per GB retrieval fee. The S3 Standard-IA is ideal for backups, long-term storage, and as a data store for disaster recovery

- High Availability and Low Latency
- The durability of 99.999999999% and availability of 99.99% availability

S3 Intelligent-Tiering

S3 Intelligent Tiering optimizes costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead. It moves objects that have not been accessed for 30 consecutive days to the infrequent access tier. If the object is accessed then it is automatically moved back to the frequent access tier. No retrieval fees or additional tiering fees are using the S3 Intelligent-Tiering storage class.

- Low latency and high throughput performance
- Automatically moves the data between two access tiers.
- The durability of 99.999999999% and availability of 99.99% availability
- Small monthly monitoring and auto-tiering fee

S3 One Zone-IA

S3 One Zone- Infrequent Access is for the data that is accessed less frequently but available for millisecond access. Since the other S3 storage class store data in a minimum of 3 Availability Zones (AZ), S3 One Zone-IA stores data in only one AZ which makes the costs 20% lesser than the S3 Standard-IA. It offers the same high durability, high throughput, and low latency. It can be considered a good choice for storing secondary backup copies or easily re-creatable data if an AZ fails.

- Low Latency and High throughput performance
- The durability of 99.999999999% and availability of 99.5% availability
- Data will be lost if the Availability Zone where the data is stored is destroyed.
- Suitable for larger objects greater than 128 KB kept for at least 30 days

S3 Glacier

S3 Glacier is a low-cost storage class for data archiving where data access is infrequent. It provides a configurable retrieval time for the data from minutes to hours. This storage class uses a very low-cost Glacier storage service but the objects are still managed through S3.

- Low-cost design for long-term archiving
- Data will be available in case of entire Availability Zone destruction
- The durability of 99.999999999% and availability of 99.9% availability
- It has a minimum storage duration period of 90 days.

S3 Glacier Deep Archive

The S3 Glacier Deep Archive provides the lowest-cost storage class and supports long-term retention and digital preservation for data that may be accessed only once or twice in a year.It is ideal for those industries which store data for 5-10 years or longer like healthcare, finance, etc. It can also be used for backup and disaster recovery.

- Lowest cost storage option in S3
- The durability of 99.999999999% and availability of 99.9% availability
- Retrieval costs can be reduced by using bulk retrieval
- It has a minimum storage duration period of 180 days

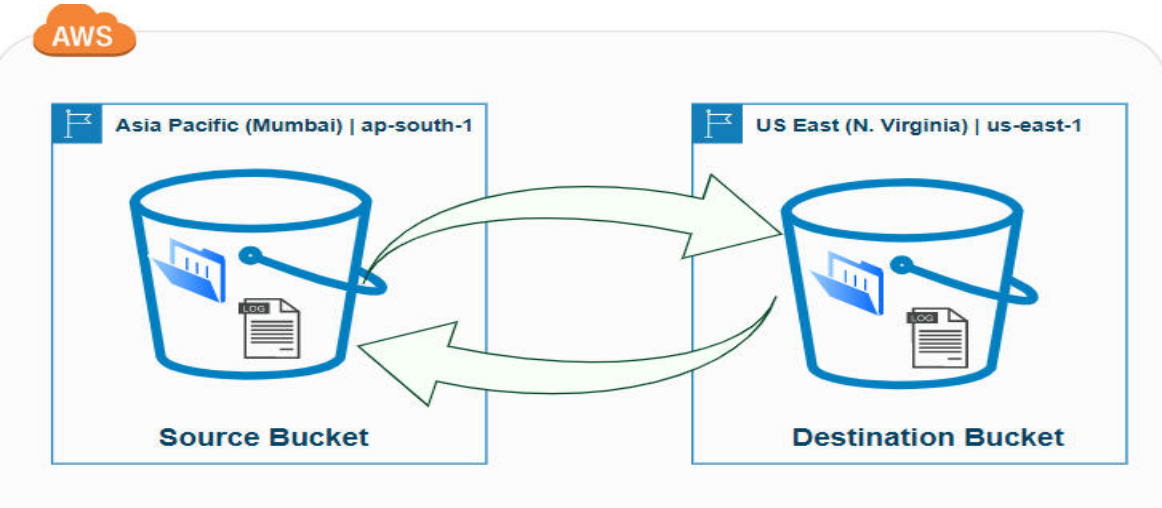
S3 Outposts

S3 on Outposts provides object storage to our on-premises AWS outposts environment. S3 on Outposts makes it easy to store, retrieve, secure, control access, tag, and report on the data. It is ideal for workloads with local data residency requirements, and to satisfy demanding performance needs by keeping data close to on-premises.

- S3 Object compatibility and bucket management is through S3 SDK
- For durable and redundant storage of data on Outposts
- S3 on Outposts will give users 48TB or 96TB of S3 storage capacity, with up 100 buckets on each **Outpost.**

S3 versioning, Bucket policies, Replication Rules in S3

Amazon S3 Cross-Region Replication (CRR) is a feature that allows you to automatically replicate objects across different AWS regions. With CRR, you can create a replication rule that specifies the source bucket and the destination bucket, and S3 will automatically replicate new objects and object updates to the destination bucket.



CRR can be useful for a number of use cases such as:

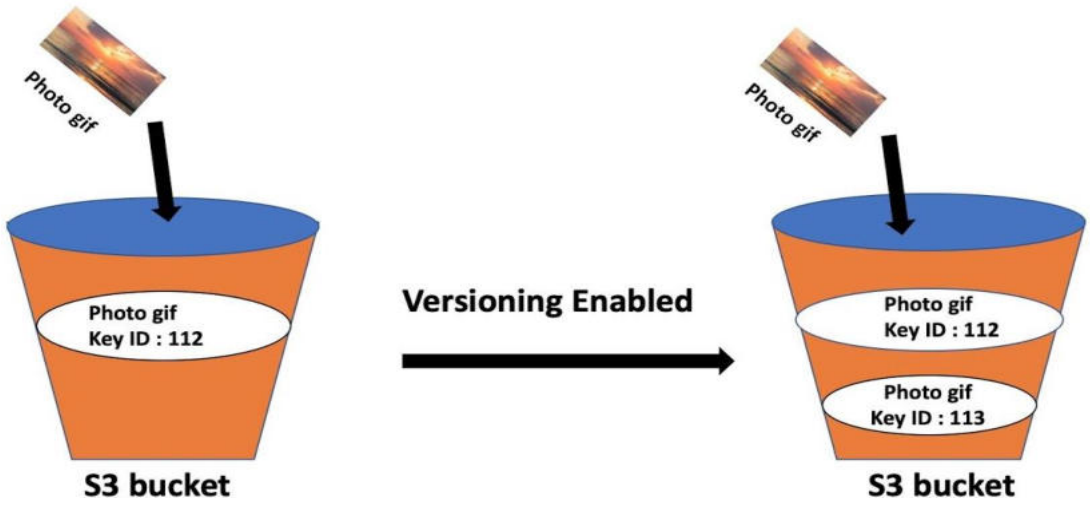
- Providing disaster recovery by replicating data to a different region in case of a regional outage.
- Providing read-only access to objects from a different region with lower latency
- Complying with data sovereignty and compliance requirements by storing data in specific regions
- Reducing the cost of data transfer by replicating data to a region closer to your users

AWS-S3 FEATURES

- **Low cost and Easy to Use** − Using Amazon S3, the user can store a large amount of data at very low charges.
- **Secure** − Amazon S3 supports data transfer over SSL and the data gets encrypted automatically once it is uploaded. The user has complete control over their data by configuring bucket policies using AWS IAM.
- **Scalable** − Using Amazon S3, there need not be any worry about storage concerns. We can store as much data as we have and access it anytime.
- **Higher performance** − Amazon S3 is integrated with Amazon CloudFront, that distributes content to the end users with low latency and provides high data transfer speeds without any minimum usage commitments.
- **Integrated with AWS services** − Amazon S3 integrated with AWS services include Amazon CloudFront, Amazon CLoudWatch, Amazon Kinesis, Amazon RDS, Amazon Route 53, Amazon VPC, AWS Lambda, Amazon EBS, Amazon Dynamo DB, etc.

S3 VERSIONING

Amazon S3 Versioning allows the retention of multiple versions of an object within a single S3 bucket, including images. This functionality provides a mechanism for recovering from accidental deletions or overwrites.



Enabling Versioning:

Versioning is enabled at the bucket level. Once activated, S3 automatically assigns a unique version ID to each new or updated object, including images, stored in that bucket.

Storing Multiple Versions:

When a new version of an image is uploaded with the same key (filename) as an existing image in a versioning-enabled bucket, S3 does not overwrite the old version. Instead, it stores the new image as a distinct version alongside the previous one, each with its own unique version ID.

Delete Markers:

If an image is deleted from a versioning-enabled bucket, S3 does not permanently remove it. Instead, it inserts a "delete marker" as the current object version. This marker indicates that the object is considered deleted, but previous versions remain accessible.

Restoring Images:

To restore a previous version of an image, the delete marker can be removed, or a specific older version can be promoted as the current version.

This allows recovery of images that were accidentally deleted or overwritten with an undesired version.

Viewing Versions:

Users can view all versions of an image within a bucket by enabling the "Show versions" option in the S3 console, which displays each version with its unique ID.

Cost Considerations:

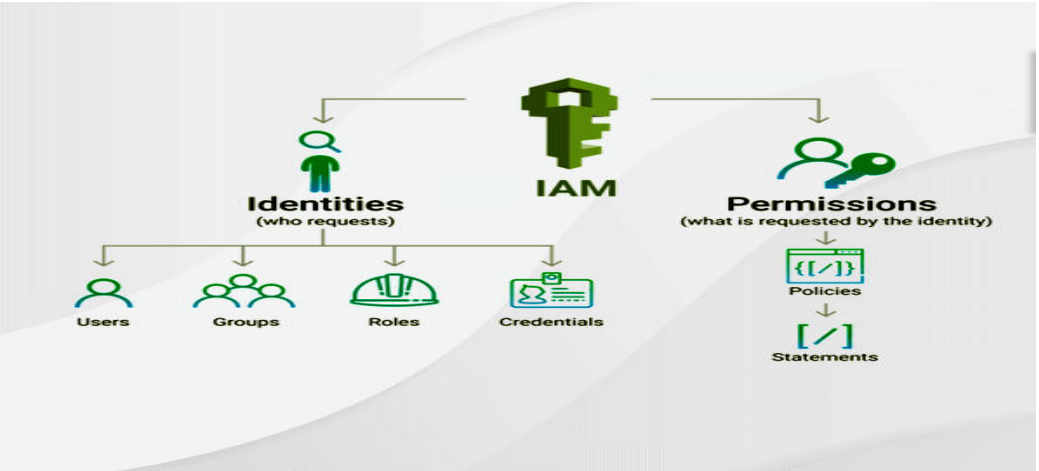
While versioning provides data protection, it also means that multiple copies of an image are stored, potentially increasing storage costs. Delete markers also incur a nominal storage charge.

In essence, S3 Versioning provides a robust mechanism for managing and recovering different iterations of images stored in S3, ensuring data integrity and enabling rollbacks to previous states.

IAM service-Groups, Users, Policies & Roles

IAM (Identity and Access Management) is a service provided by Amazon Web Services (AWS) that helps manage access to AWS resources. With IAM, you can:

- Create and manage AWS users and groups
- Grant and deny permissions to AWS resources
- Set up and enforce password policies
- Generate access keys and credentials for programmatic access to AWS resources
- Manage multi-factor authentication (MFA) for enhanced security



User:

In the AWS Identity and Access Management (IAM) service, a user is a person or service that interacts with AWS resources. An IAM user is identified by a unique user name and can be assigned security credentials, such as an AWS access key and secret access key for programmatic access, or a password for AWS Management Console access.An IAM user can be granted permissions to perform specific actions on AWS resources, or denied access to those resources, through the use of IAM policies. This enables you to control who has access to your AWS resources, what actions they can perform, and when they can perform those actions.

Group

In the AWS Identity and Access Management (IAM) service, a group is a collection of IAM users. An IAM group is used to grant the same permissions to multiple IAM users, making it easier to manage permissions for a set of users.

Roles

A role is an AWS identity that has specific permissions to access AWS resources. Unlike an IAM user, which is associated with a specific person or service, an IAM role does not have any permanent credentials and is intended to be assumed by another AWS entity, such as an EC2 instance, an AWS Lambda function, or an AWS service.

For example, you could create an IAM role for an EC2 instance that needs to access your Amazon S3 bucket to read and write data. The EC2 instance can then assume the role, using its temporary security credentials, to perform the actions allowed by the role's policies.

Policies

A policy is a document that defines one or more permissions. An IAM policy is attached to an IAM user, group, or role, and grants or denies the permissions defined in the policy to the entity.An IAM policy is written in JSON and is composed of a set of statements, each of which specifies an action, a resource, and a condition. The action is the operation that you are allowing or denying, such as "**s3: PutObject**" to allow writing an object to an Amazon S3 bucket

Policies are written in **JSON** format and consist of a few key elements.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "s3:*",
         "Resource": "*"
      }
   ]}
```

Creating IAM users using AWS console

- Sign in to the AWS Management Console and open the IAM console
- In the navigation pane, choose **Users** and then choose **Add users**.
- Type the user name for the **new user**. This is the sign-in name for AWS.
- Select the type of access this set of users will have. You can select **programmatic access,** access to the AWS Management Console, or both.
- For Console password, choose one of the following
  - Auto generated password. Each user gets a randomly generated password that meets the account password policy
  - Custom password. Each user is assigned the password that you type in the box.
  - Choose **Next: Permissions**
- On the **Set permissions** page, specify how you want to assign permissions to this set of new users.
- Choose **Next: Tags**.
- Choose **Next: Review** and we can choose create option
- To view the users' **access keys** (access key IDs and secret access keys), choose Show next to each password and access key that you want to see. To save the access keys, choose Download **.csv** and then **save** the file to a safe location.

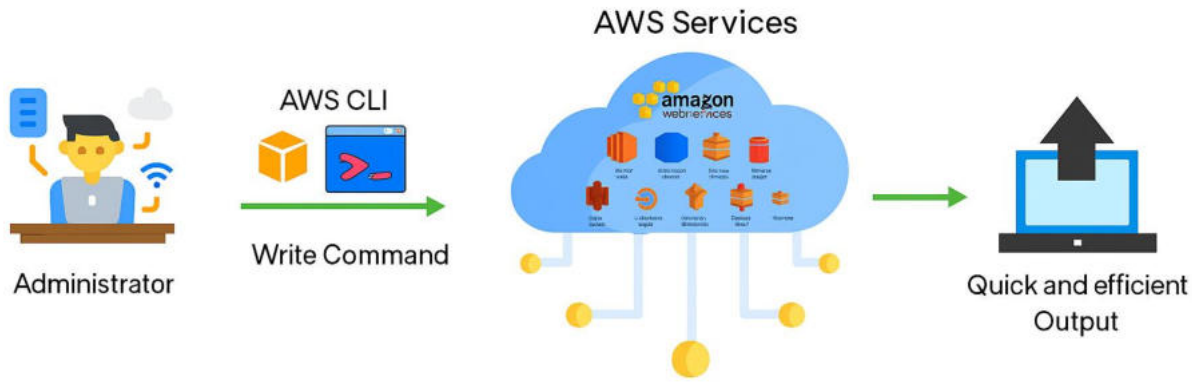
Features of IAM

- **Centralised control of your AWS account**: You can control creation, rotation, and cancellation of each user's security credentials. You can also control what data in the aws system users can access and how they can access.
- **Shared Access to your AWS account**: Users can share the resources for the collaborative projects.
- **Granular permissions**: It is used to set a permission that user can use a particular service but not other services.
- **Identity Federation**: An Identity Federation means that we can use Facebook, Active Directory, LinkedIn, etc with IAM. Users can log in to the AWS Console with same username and password as we log in with the Active Directory, Facebook, etc.
- **Multifactor Authentication**: An AWS provides multifactor authentication as we need to enter the username, password, and security check code to log in to the AWS Management Console.
- **Permissions based on Organizational groups**: Users can be restricted to the AWS access based on their job duties, for example, admin, developer, etc.
- **Free to use:** AWS IAM is a feature of AWS account which is offered at no additional charge. You will be charged only when you access other AWS services by using IAM user.

AWS-CLI

AWS CLI (Command Line Interface) provides a powerful way to interact with AWS services via command line commands. This is includes managing S3 buckets and files such as creating buckets, uploading and downloading files, syncing data between buckets, and deleting buckets with the --force option.

- It also involves IAM (Identity and Access Management) operations like creating and deleting users and groups, which help control access to AWS resources.
- Additionally, the commands demonstrate managing EC2 instances by retrieving instance details, stopping, starting, and terminating instances using their instance IDs.
- These operations showcase essential AWS resource management skills using CLI for automation and efficient cloud infrastructure handling.



Step 1: Configure AWS CLI with your credentials.

- aws configure

Step 2: List all existing S3 buckets.

- aws s3 ls

Step 3: List all objects inside the bucket krishna908.

- aws s3 ls s3://krishna908

Step 4: Create a new S3 bucket named tillu909.

- aws s3 mb s3://tillu909

Step 5: Create or edit a file named index.html on your local system.

- cat > index.html

Step 6: List files in the current directory to confirm file creation.

- ll

Step 7: Upload index.html file to the S3 bucket tillu909.

- aws s3 cp index.html s3://tillu909

Step 8: Download the file liitle.jpg from the bucket krishna908 to your local system.

- aws s3 cp s3://krishna908/liitle.jpg .

Step 9: List files in the current directory to confirm download.

- ll

Step 10: List all existing S3 buckets again.

- aws s3 ls

Step 11: Try to delete the bucket tillu909.

- aws s3 rb s3://tillu909

Step 12: Force delete the bucket tillu909 along with all its contents.

- aws s3 rb s3://tillu909 --force

Step 13: Create another new S3 bucket named milky890.

- aws s3 mb s3://milky890

Step 14: List files in the current directory.

- ll

Step 15: List all S3 buckets again.

- aws s3 ls

Step 16: Sync all files from bucket krishna908 to bucket milky890.

- aws s3 sync s3://krishna908 s3://milky890

Step 17: Force delete the bucket milky890 with all its contents.

- aws s3 rb s3://krishna908 --force.

Step 18: Create an IAM user named pavan.

- aws iam create-user --user-name pavan

Step 19: Create another IAM user named kiran

- aws iam create-user --user-name kiran

Step 20: Delete the IAM user kiran

- aws iam delete-user --user-name kiran

Step 21: Create an IAM group named dev-team.

- aws iam create-group --group-name dev-team

Step 22: Delete the IAM user pavan

- aws iam delete-user --user-name pavan

Step 23: Describe details of the EC2 instance

- aws ec2 describe-instances --instance-ids i-04aa69fc61ea31e35

Step 24: Stop the EC2 instance.

- aws ec2 stop-instances --instance-ids i-04aa69fc61ea31e35

AWS-RDS

The Amazon Relational Database Service (RDS AWS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, re-sizable capacity in an industry- Standard relational database and manages common database administration tasks.RDS is not a database; it's a service that manages databases. RDS provides the following database services.

MySQL Server

- It is an open source relational database.
- It is free to download and use.
- It is very popular in the developer community.
- It is easy to set up, operate, and scale MySQL deployments in aws.

PostgreSQL

- It is an open source Relational database for enterprise developers and start-ups.
- It is easy to set up, operate, and scale PostgreSQL deployments in the cloud.
- With Amazon RDS, you can scale PostreSQL deployments in aws cloud in minutes with cost-effective and resizable hardware capacity.
- It manages time-consuming administrative tasks such as PostgreSQL software installation, storage management, replication for high availability, and backups for disaster recovery.

Aurora

- It is a relational database, and closed source database engine.
- It is compatible with MySQL and delivers five times throughput of MySQL on the same hardware.
- It is also compatible with PostgreSQL and delivers three times throughput of PostgreSQL on the same hardware.
- Amazon RDS with Aurora manages the time-consuming administrative tasks such as software installation, patching, and backups.
- The main features of Aurora are fault-tolerant, distributed, a self-healing storage system that auto-scales upto 64 TB per database instance.
- It provides high-performance, availability, point-in-time recovery, continuous backed up to S3, and replication across three availability zones.

MariaDB

- MariaDB is an open source relational database developed by the developers of MySQL.
- It is easy to set up, operate, and scale MariaDB deployments in the aws cloud.
- With Amazon RDS, you can deploy MariaDB databases in minutes with cost- effective and resizable hardware capacity.
- It frees you from managing the time-consuming administrative tasks such as software installation, patching, monitoring, scaling, and backups.
- It is object-relational database management system which was developed by Oracle Inc.

Creation of RDS:

- First select the RDS service from the AWS Management Console.
- Since we will be launching a MySQL instance, select the MySQL instance from the list of Dbs
- Choose **Create database** and make sure that Easy create is chosen.
- In Configuration, choose **MySQL**.
- For **DB instance size**, choose **Freetier.**
- For DB instance identifier, enter a name for the **DB instance**
- To enter your master password, clear the Auto generates a password box, and then enter the same password in Master password and **Confirm password.**
- Choose **Createdatabase.**
- In the Databases list, choose the name of the new **MySQL DB** instance.

Deleting a DB instance

- After you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.
- Sign in to the AWS Management Console and open the **Amazon RDS** console
- In the navigation pane, choose **Databases.**
- Choose the **DB instance** that you want to delete.
- For **Actions**, choose **Delete.**
- For **Create final snapshot**... **Choose No**, and select the acknowledgment.
- Choose **Delete.**

Cost of Amazon RDS

- When using Amazon RDS, pay only for only the usage without any minimum and setup charges. Billing is based on the following criteria − •Instance class − Pricing is based on the class of the DB instance consumed.
- **Running time:** Price is calculated by the instance-hour, which is equivalent to a single instance running per hour.
- **Storage:** Bill is calculated as per the storage capacity plan chosen in terms of per GB.
- I/O requests per month − Billing structure also includes total number of storage I/O requests made in a billing cycle.
- **Backup storage :** There is no additional charges for backup storage up to 100% of database. This service is free only for active DB instances.

Free Tier

- AWS has an amazing free tier usage for most of its services, so that the customer can first use the service and then do the needful.
- 750 hours of Amazon RDS usage in single-AZ **for db.t2.micro** instance, every month for one year from signup.
- 20 GB of Database Storage: any combination of General Purpose (SSD) or Magnetic storage. 10 million IOs
- 20GB of backup storage

Features of Amazon RDS

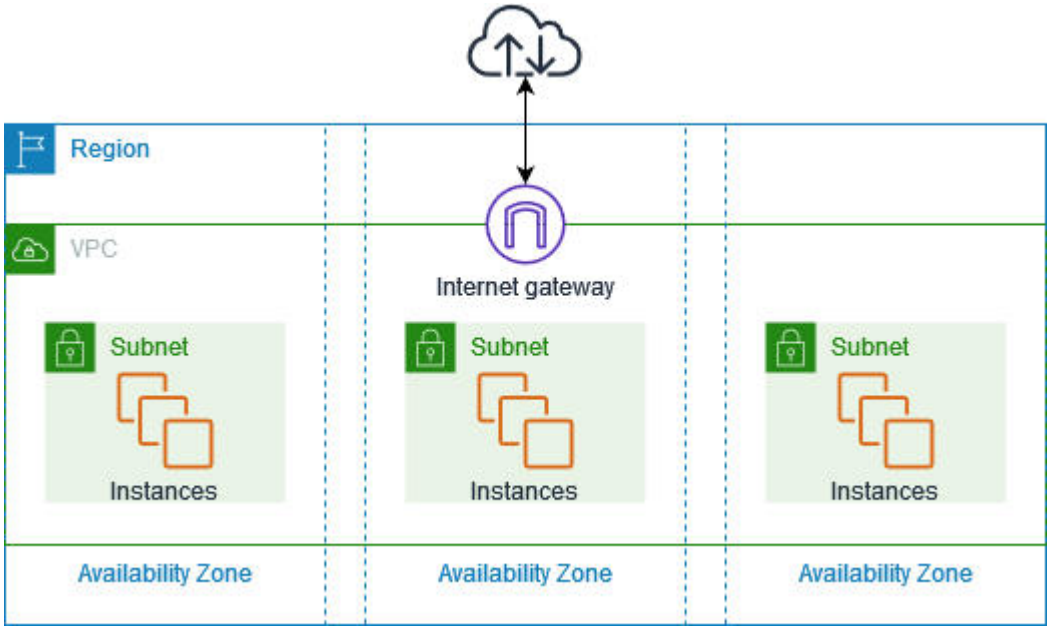
Amazon RDS has the following features :

- **Scalable :** Amazon RDS allows to scale the relational database by using AWS Management Console or RDS-specific API. We can increase or decrease your RDS requirements within minutes.
- **Host replacement** : Sometimes these situations occur when the hardware of Amazon RDS fails. There is no need to worry; it will be automatically replaced by Amazon.
- **Inexpensive :** Using Amazon RDS, we pay only for the resources we consume. There is no up-front and long-term commitment.
- **Secure :** Amazon RDS provides complete control over the network to access their database and their associated services.
- **Automatic backups** : Amazon RDS backs up everything in the database including transaction logs up to last five minutes and also manages automatic backup timings.
- **Software patching** : automatically gets all the latest patches for the database software. We can also specify when the software should be patched using DB Engine Version Management.

VPC

With Amazon Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following diagram shows an example VPC. The VPC has one subnet in each of the Availability Zones in the Region, EC2 instances in each subnet, and an internet gateway to allow communication between the resources in your VPC and the internet.



VPC

You can launch AWS resources into a defined virtual network using Amazon Virtual Private Cloud (Amazon VPC). With the advantages of utilizing the scalable infrastructure of AWS, this virtual network closely mimics a conventional network that you would operate in your own data center. /16 user-defined address space maximum (65,536 addresses)

2. Subnets

- A subnet divides the large network into smaller, logically separated networks
- You can create up to 200 subnets.
- Some subnets can be public (accessible from the internet), and some can be private (only internal communication).
- This helps you control traffic and increase security within your VPC.

3. Route Tables

- Route Tables are mainly used to Define the protocol for traffic routing between the subnets. They tell data where to go inside your VPC.
- Each subnet must be associated with a route table.
- Routes define the direction of traffic based on IP addresses.

4. Network Access Control Lists

Network Access Control Lists (NACL) for VPC serve as a firewall by managing both inbound and outbound rules. There will be a default NACL for each VPC that cannot be deleted.

5. Internet Gateway(IGW)

The Internet Gateway (IGW) will make it possible to link the resources in the VPC to the Internet allowing resources like web servers to be accessed publicly. Without an IGW, your VPC cannot communicate with the internet.

6. Network Address Translation (NAT)

A NAT Gateway allows instances in private subnets to initiate outbound internet connections, without allowing inbound access from the internet. It allows instances in a private subnet to access the internet outbound (like downloading updates) while keeping them hidden from outside users.

VPC-CREATION

- In the AWS Console, search for “VPC” in the search bar.
- Click on "VPC" under Networking & Content Delivery.
- In the VPC Dashboard, click “Create VPC”.

Choose one of the following options:

- VPC only (for custom setup)
- Configure VPC Settings
- Click "Create VPC" to proceed.

VPCs need subnets for resource allocation.

- Go to Subnets → Create subnet.
- Choose your VPC, then configure:
- Name
- Availability Zone
- CIDR block (e.g., 10.0.1.0/24, 10.0.2.0/24)

Create and Attach an Internet Gateway (for public access)

- Go to Internet Gateways → Create Internet Gateway.
- Name it (e.g., MyIGW) and click Create.
- Attach it to your VPC:
- Select the IGW → Actions → Attach to VPC → select your VPC.

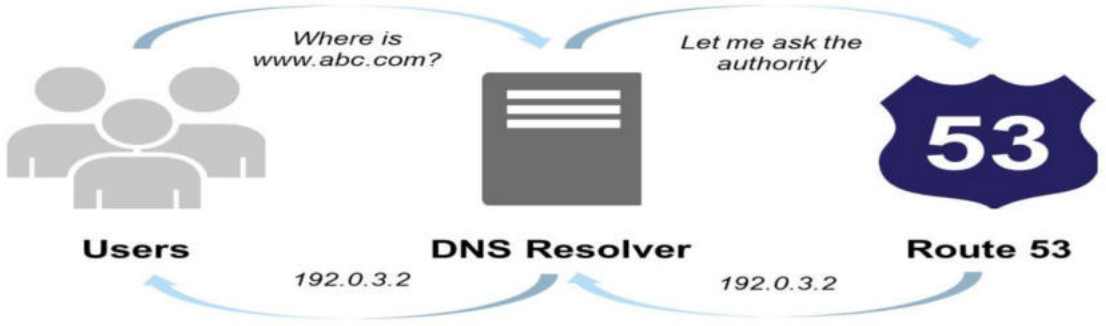
Configure Route Tables

- Go to Route Tables → Create route table.
- Associate it with your VPC.
- Add routes:
- For public subnets: Add 0.0.0.0/0 pointing to the Internet Gateway.
- Associate the route table with the public subnets.

Amazon Route53

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. It is basically designed for developers and corporate to route the end users to Internet applications by translating human-readable names like www.geeksforgeeks.org into the numeric IP addresses like 192.0.1.1 that computers use to connect to

each other. You cannot use Amazon Route 53 to connect your on- premises network with AWS Cloud



DNS Service: Route 53 primarily acts as a DNS service. It allows you to register domain names, manage the DNS records associated with those domains, and resolve DNS queries for your applications.

Domain Registration: You can use Route 53 to register new domain names or transfer existing domain names from other registrars. It provides a simple and integrated way to manage your domain names and their associated DNS records.

DNS Record Types: Route 53 supports various DNS record types, including A records (IPv4 addresses), AAAA records (IPv6 addresses), CNAME records (canonical names), MX records (mail exchange), TXT records (text records), and more. You can configure these records to point to your resources or external services.

Traffic Routing and Health Checks: Route 53 allows you to route traffic based on various policies like weighted routing, geolocation routing, latency-based routing, and failover routing. You can also set up health checks to monitor the health of your resources and route traffic based on the health of those resources.

Global DNS Service: Route 53 is a global service with a distributed architecture. It ensures low-latency DNS resolution by automatically routing DNS queries to the nearest available data center.

Alias Records: Route 53 supports alias records, which are a special type of DNS record that can be used to route traffic to AWS resources directly. For example, you can create an alias record that points to an Elastic Load Balancer (ELB), an Amazon S3 bucket, or an AWS CloudFront distribution.

Integration with AWS Services: Route 53 seamlessly integrates with other AWS services like ELB, S3, CloudFront, and API Gateway, making it easier to manage the DNS for these services.

Benefits of Route53

Highly Reliable: Route53 is built using AWS's highly available and reliable infrastructure. The distributed nature of the AWS DNS servers helps ensure a consistent ability to route the end-users to the web application.

Scalable: It automatically scales the resources during large traffic and also handles large queries without the user's intervention.

Easy to use: Very user-friendly and easy to configure DNS settings. It can start to answer your DNS queries within minutes. Can be mapped easily to any resource.

Health Check: Route 53 monitors the health of the application. If any failure is detected, it automatically redirects the user to a healthy resource before the customer can identify the problem.

Flexible: You can decide which policy you want to use at given time. Simple: Using routing types, Route53 helps to manage traffic globally. Cost-effective: Payment is done only according to the services used.

Secure: By integrating it with IAM, the access to Amazon Route53 is secured by giving its permissions to only the authorized users.

Mapped with various AWS services: It can be used to map domain names to Amazon EC2 instances, S3 buckets, and other AWS resources.

Methodologies related to Route53

Records: Records are created to route internet traffic to the resources. They are the objects present in the hosted zone which determines how the internet traffic has to be routed for a domain name so that it finally reaches the resources. The name of each record in a hosted zone must end with the name of the hosted zone.

Hosted zone: When the domain name is registered, Route53 creates a public hosted zone that has the same name as the domain name. It is a collection of records that contains information about how to route traffic of its domains and all of its sub domains.

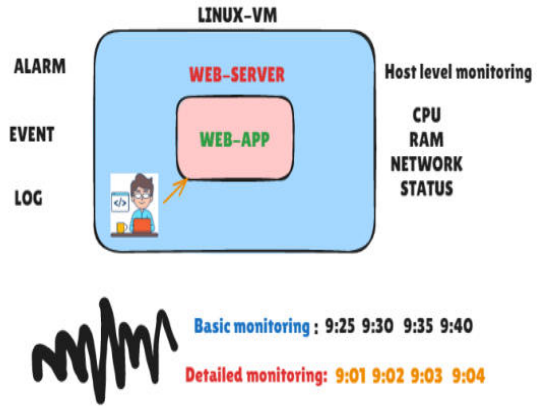
DNS query: It is a request for information sent from DNS client to the DNS server. Alias records: Alias records helps in routing internet traffic to AWS resources like S3 bucket, Amazon Cloud Front, etc. It is created at the top node of the DNS namespace. Name servers: They are the servers in the DNS that translates the domain name into IP address so that internet traffic can be routed to the resources.

DNS failover: A method for routing the traffic from unhealthy resources to healthy resources, whenever a failure is detected.

Routing policy: Routing policy determines how Amazon Route53 responds to queries.

Amazon Cloudwatch Or Monitoring Tools

Amazon Cloud Watch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use Cloud Watch to collect and track metrics, which are variables you can measure for your resources and applications.



The Cloud Watch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications, and display custom collections of metrics that you choose.

We can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money

- **Basic Monitoring for Amazon EC2 instances**: Ten pre-selected metrics at fiveminute frequency, free of charge
- **Detailed Monitoring for Amazon EC2 instances**: Seven pre-selected metrics at one minute frequency, for an additional charge
- **Amazon EBS volumes**: Ten pre-selected metrics at five-minute frequency, free of charge
- **Elastic Load Balancers:** Ten pre-selected metrics at one-minute frequency, free of charge
- **Auto Scaling groups:** Seven pre-selected metrics at one-minute frequency, optional and charged at standard pricing
- **Amazon RDS DB instances:** Thirteen pre-selected metrics at one-minute frequency,free of charge.

CloudWatch Metrics

CloudWatch Provides metrics for every services in AWS

- **Metrics:** Metrics is a variable to monitor(CPUUtilization, NetworkIn, etc)
- **Alarms:** An Alarm basically watches over a particular metric for a stipulated period of time and performs some actions based on its trigger. These actions can be anything from sending a notification to the user using SNS.

- **Namespaces:** A container for CloudWatch metrics. It is a grouping to know what this metric belongs to. For example: AWS/EC2, AWS/AutoScaling/ AWS/ELB
- **Dimensions:** Dimension is a name/value pair that you uniquely identify a metric.
- For example: AutoScalingGroupName, ImageId, InstanceID, InstanceType, VolumeID.
- **Timestamps:** To know what timestamp it had captured.
- **Units:** Unit represents the statistic's unit of measure. For example: EC2 NetworkIn metric in bytes.

CloudWatch Events

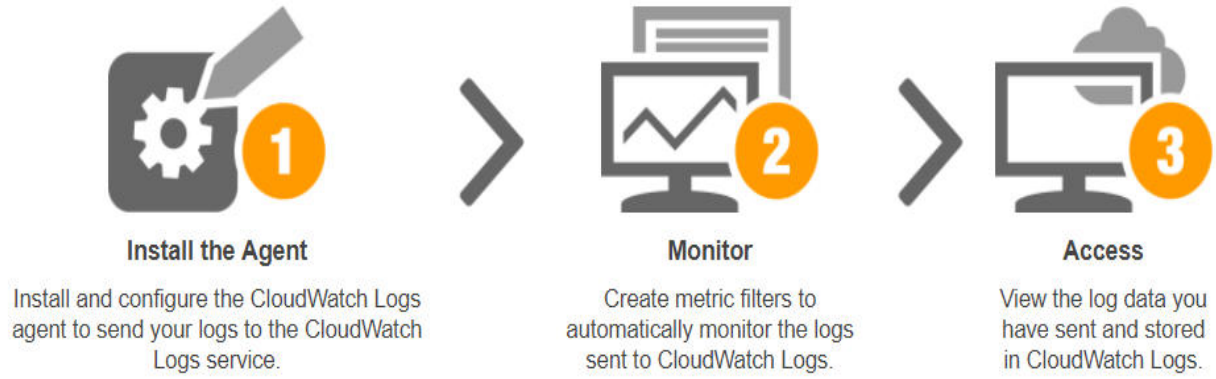
Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in Amazon Web Services (AWS) resources.

- Determine events of interest
- Create rules to match events
- Route to targets

You can Analyze your logs with CloudWatch Logs Insights. No Infrastructure or setup needed.

CloudWatch Logs

- CloudWatch Logs helps you to aggregate, monitor, and store logs.
- For example, you can: Monitor HTTP response codes in Apache logs Receive alarms for errors in kernel logs Count exceptions in application logs



Advantages

- A large amount of data is produced by web applications nowadays so Amazon cloud watch acts as a dashboard that contains the organized collection of whole data.
- It improves the total cost of ownership by providing alarms and also takes automated actions when there is an error in limits provided.
- Applications and resources can be optimized by examining the logs and metric data.
- Detailed Insights from the application are provided through data like CPU utilization, capacity utilization, memory utilization, etc

- It provides a great platform to compare and contrast the data produced by various AWS services.

CREATING CLOUDWATCH

- Open the **CloudWatch** console.
- From the navigation pane, click Alarms -→**Create Alarm.**
- Select your metric and then perform either of the following actions:
- Select the service **namespace** containing the metrics that you want to include.
- Search for a metric in the search box and then **press Enter.**
- From the "**Graphed metrics**" tab, perform the following actions:
- From Statistics, choose any of the available **statistics** or **percentiles**.
- From Period, specify the evaluation period of the alarm.
- Click "**Select metric**" to display the "Specify metric and conditions" page that shows all information of the metric and statistics that you have specified.
- From the "**Conditions**" tab, perform the following actions:
- Specify the name and description of the alarm.
- Use the "Whenever <metric> is" field to specify the threshold value of the metric.
- Click "**Additional configuration**."
- In "**Datapoints to alarm**," specify the number of evaluation periods (or data points) required to trigger the alarm.
- From "**Missing data treatment**," configure the alarm settings in the event of any missing data points.
- Click Next.
- From the "**Notification**" tab, select the Amazon SNS topic that should be notified when the alarm is triggered.
- Select the respective options if you want the alarm to perform auto-scaling or any other EC2 actions.
- In the "**Preview and create**" tab, preview the alarm settings and then click "**Create alarm**."

UNIT – III

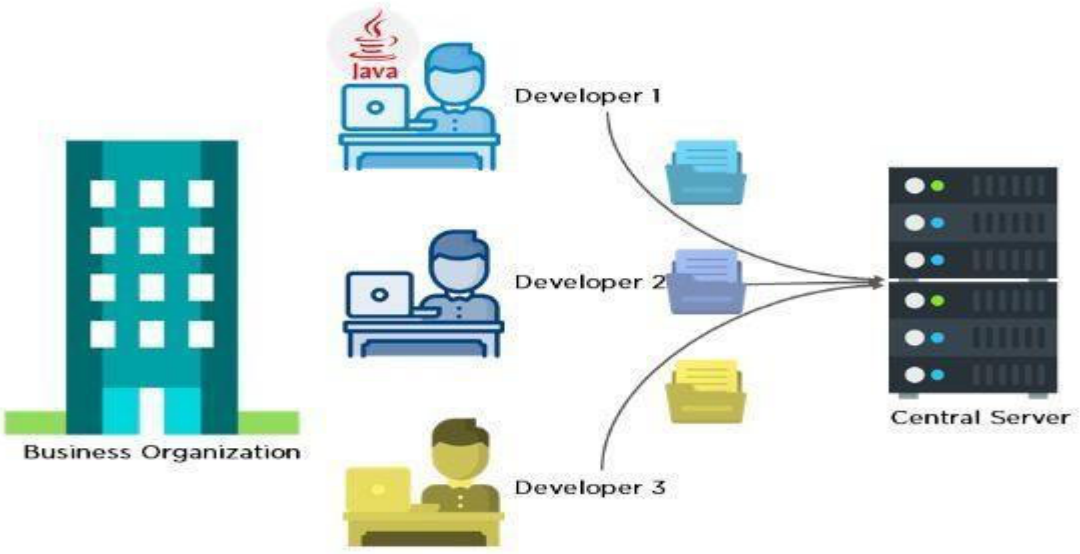
Git&GitHub: introduction to Version control system, types of VCS, GIT life cycle, basic commands, git branches, create a remote repository

Introduction to Version control system

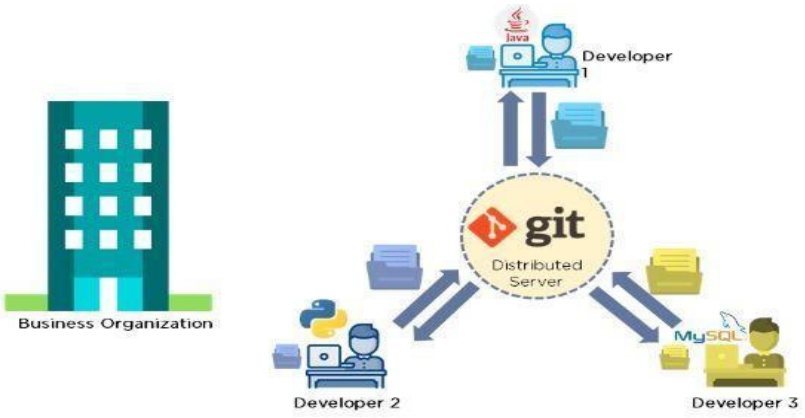
Git is a tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. Linus Torvalds created Git in 2005 for the development of the Linux kernel.

The scenario before Git

- Developers used to submit their codes to the central server without having copies of their own
- Any changes made to the source code were unknown to the other developers
- There was no communication between any of the developers



The Scenario after Git:



Git Version Control System

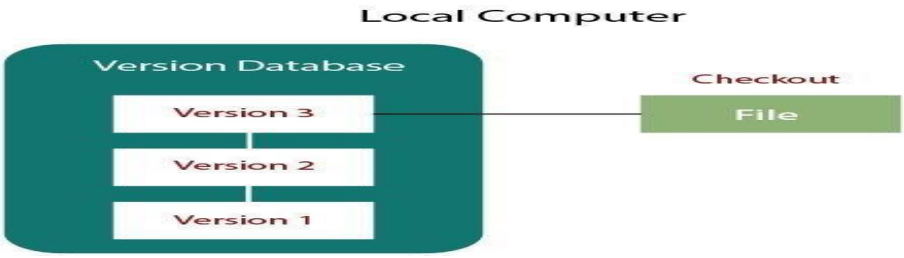
A version control system is software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers. The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code.

Developers can compare earlier versions of the code with an older version to fix the mistakes.

Types of Version Control System

1. Localized version Control System
2. Centralized version control systems
3. Distributed version control systems

Local Version Control Systems: It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.



To deal with this issue, programmers developed local VCSs that had a simple database. Such databases kept all the changes to files under revision control. A local version control system keeps local copies of the files. The major drawback of Local VCS is that it has a single point of failure.

Centralized VCS

Centralized Version Control System:

The name itself indicates that, this type contains only one central repository and every developer should be connected to that repository. The total project code will be stored in the central repository. If 4 developers are there, still we have only one repository. This type of VCS is very easy to setup and use.

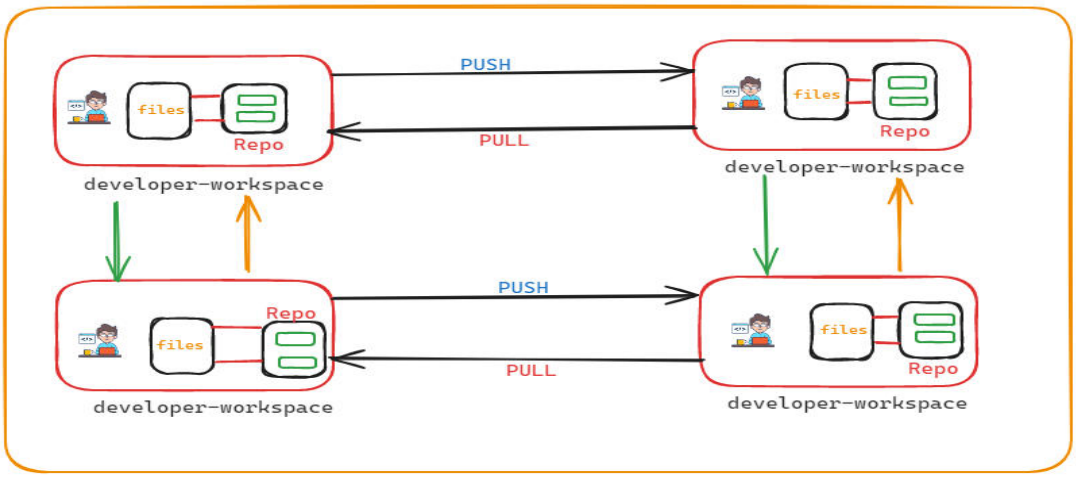
Eg: CVS, SVN, Perforce, TFS, Clear case etc

Problems with Centralized VCSs:

Central Repository is the only place where everything is stored, which causes single point of failure. If something goes wrong to the central repository then recovery is very difficult.

- All commit and checkout operations should be performed by connecting to the central repository via network. If network outage, then no version control to the developer. i.e in this type, developer work space and remote repository server should be connected always.
- All commit and checkout operations should be performed by connecting to the central repository via network and hence these operations will become slow, which causes performance issues. No local operations and every version control operation should be remote operation.
- Organization of central repository is very complex if number of developers and files increases.
- Every programmer can extract or update their workstations with the data present in the repository or can make changes to the data or commit in the repository. Every operation is performed directly on the repository.

DVCS: Distributed Version Control System or De-centralized Version Control System. The name itself indicates the repository is distributed and every developer's workspace contains a local copy of the repository. There is no question of central repository.



If 4 developers are there 4 repositories will be there.

- The checkout and commit operations will be performed locally. Hence performance is more.

- To perform checkout and commit operations network is not required. Hence if there is any network outage, still version control is applicable.
- If something goes wrong to any repository there is a chance to recover. There is no question of single point of failure.
- To perform push and pull operations network must be required, but these operations are not most common operations and we are performing very rarely.

Advantages:

- All operations (except push & pull) are very fast because the tool only needs to access the hard drive, not a remote server. Hence, you do not always need an internet connection.
- Committing new change-sets can be done locally without manipulating the data on the main repository. Once you have a group of change-sets ready, you can push them all at once.
- Since every contributor has a full copy of the project repository, they can share changes with one another if they want to get some feedback before affecting changes in the main repository.
- If the central server gets crashed at any point of time, the lost data can be easily recovered from any one of the contributor's local repositories.

Benefits of the version control system:

- Enhances the project development speed by providing efficient collaboration,

Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,

- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small changes
- Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,
- For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is Git, Helix core, Microsoft TFS,
- Helps in recovery in case of any disaster or contingent situation,
- Informs us about Who, What, When, Why changes have been made.

Use of Version Control System:

A repository: It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.

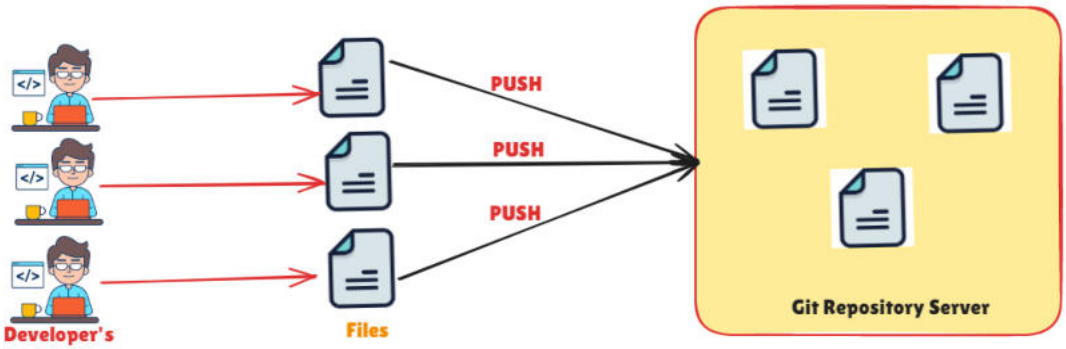
Copy of Work (sometimes called as checkout): It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.

Difference between Centralized Version Control System and Distributed Version Control System

Sr. No.	Key	Centralized Version Control	Distributed Version Control
1	Working	In CVS, a client need to get local copy of source from server, do the changes and commit those changes to central source on server.	In DVS, each client can have a local branch as well and have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository.
2	Learning Curve	CVS systems are easy to learn and set up.	DVS systems are difficult for beginners. Multiple commands needs to be remembered.
3	Branches	Working on branches in difficult in CVS. Developer often faces merge conflicts.	Working on branches in easier in DVS. Developer faces lesser conflicts.
4	Offline Access	CVS systems do not provide offline access.	DVD systems are workable offline as a client copies the entire repository on their local machine.
5	Speed	CVS is slower as every command need to communicate with server.	DVS is faster as mostly user deals with local copy without hitting server everytime.
6	Backup	If CVS Server is down, developers cannot work.	If DVS server is down, developer can work using their local copies.

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

- Git is foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.
- Git was created by Linus Torvalds in 2005 to develop Linux Kernel. It is also used as an important distributed version-control tool for the DevOps.
- Git is easy to learn, and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and ClearCase.
- Git makes it possible for several people involved in the project to work together and track each other's progress over time. In software development, the tool helps in Source Code Management. Git favors not only programmers but also non-technical users by keeping track of their project files.
- Git is used to tracking changes in the source code
- The distributed version control tool is used for source code management
- It allows multiple developers to work together
- It supports non-linear development through its thousands of parallel branches



FEATURES OF GIT

Distributed Version Control system provides the following list of key features.

- Free and open source
- Speed
- Scalable
- Reliable
- Secure
- Economical
- Supports non-linear development
- Easy branching
- Distributed development
- Compatibility with existing systems or protocol

Free and open source

Git is released under GPL's (General Public License) open source license. You don't need to purchase Git. It is absolutely free. And since it is open source, you can modify the source code as per your requirements.

Speed

Since you do not have to connect to any network for performing all operations, it completes all the tasks really fast.

Scalable

Git is very scalable. So, if in future, the number of collaborators increase Git can easily handle this change. Though Git represents an entire repository, the data stored on the client's side is very small as Git compresses all the huge data through a lossless compression technique.

Reliable

Since every contributor has its own local repository, on the events of a system crash, the lost data can be recovered from any of the local repositories. You will always have a backup of all your files.

Secure

Git uses the SHA1 (Secure Hash Function) to name and identify objects within its repository. Every file and commit is check-summed and retrieved by its checksum at the time of checkout. The Git history is stored in such a way that the ID of a particular version (a commit in Git terms) depends upon the complete development history leading up to that commit.

Economical

In case of CVCS, the central server needs to be powerful enough to serve requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware limitations of the server can be a performance bottleneck.

Supports non-linear development:

Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history.

Easy branching

Branch management with Git is very simple. It takes only few seconds to create, delete, and merge branches. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the master branch always contains production quality code.

Distributed development:

Git gives each developer a local copy of the entire development history, and changes are copied from one such repository to another. These changes are imported as additional development branches, and can be merged in the same way as a locally developed branch.

Compatibility with existing systems or protocol

Repositories can be published via http, ftp or a Git protocol over either a plain socket, or ssh. Git also has a Concurrent Version Systems (CVS) server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories. Apache Subversion (SVN) and SVK repositories can be used directly with Git-SVN.

Advantages of Git:

A version control application allows us to keep track of all the changes that we make in the files of our project. Every time we make changes in files of an existing project, we can push those changes to a repository. Other developers are allowed to pull your changes from the repository and continue to work with the updates.

Some of the significant **advantage**s of using Git are as follows.

Saves Time

- Git is lightning fast technology. Each command takes only a few seconds to execute so we can save a lot of time as compared to login to a GitHub account and find out its features.

Offline Working

- One of the most important benefits of Git is that it supports offline working. If we are facing internet connectivity issues, it will not affect our work. In Git, we can do almost everything locally. Comparatively, other CVS like SVN is limited and prefer the connection with the central repository.

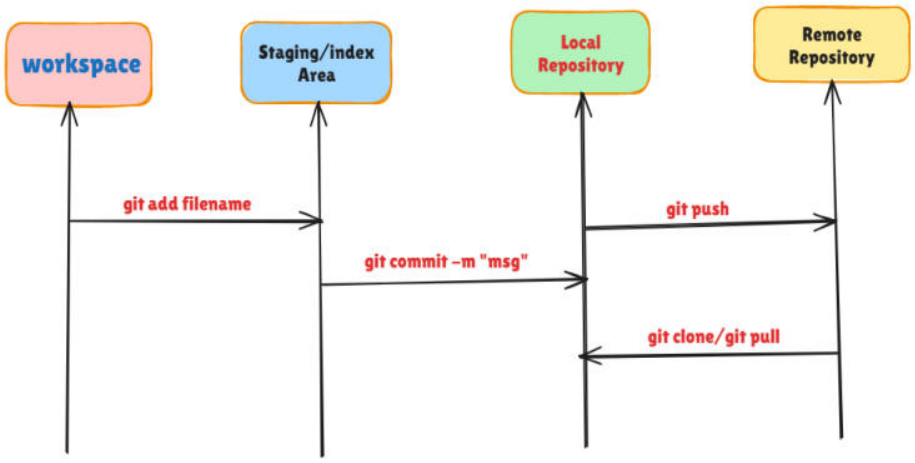
Undo Mistakes

- One additional benefit of Git is we can Undo mistakes. Sometimes the undo can be a savior option for us. Git provides the undo option for almost .

Track the Changes

- Git facilitates with some exciting features such as Diff, Log, and Status, which allows us to track changes so we can check the status, compare our files or branches.

GIT life cycle



Working Directory: Where developers are required to create/modify files. Here version control is not applicable. Here we won't use the work like version-1, version-2 etc

- **Repository:** Where we have to store files and metadata.
- Every file in GIT is in one of the following states:

1) Untracked:

- The files which are newly created in working directory and git does not aware of these files are said to be in untracked state.

2) Staged:

- The files which are added to staging area are said to be in staged state.
- these files are ready for commit.

3) In Repository/ Committed:

- Any file which is committed is said to be In Repository/Committed State.

4) Modified:

- Any file which is already tracked by git, but it is modified in working directory is said to be in Modified State.
- Here version control is applicable.
- Here we can talk about versions like version-1, version-2 etc

Commit: The process of sending files from working directory to the repository.
Checkout: The process of sending files from repository to working directory.

GIT INSTALLATION ON LINUX

- sudo yum install git -y
- git –version

Basic Git Commands

Here is a list of most essential Git commands that are used daily.

- **git config** command
- **git init** command
- **git add** command
- **git status** command
- **git commit** command
- **git log** command
- **git push** command
- **git pull** command
- **git clone** command
- **git branch** command
- **git merge** command
- **git diff** command
- **git rm** command
- **git reset** command
- **.gitignore** command
- **git cherrypick** command
- **git checkout** command
- **git fork** command

- **git aliase** name
- **git revert** command
- **git stash command**

- **git tag** command
- **git reflog** command

1. **git config command**

- This command configures the user. The **git config** command is the first and necessary command used on the Git command line.
- This command sets the **author name** and **email address** to be used with your commits.

Syntax :

- $ gitconfig –global user.name “milky”
- $ gitconfig –global user. krishna04.b@gmail.com

2. **Git init :**

- The command git init is used to create an empty Git repository.
- After the git init command is used, a **.git folder** is created in the directory with some subdirectories.
- Once the repository is initialized, the process of creating other files begins.

Syntax : $ git init

```
MICLAB@AITS MINGW64 /e/B.HARI KRISHNA/project
$ git init
Initialized empty Git repository in E:/B.HARI KRISHNA/project/.git/
```

3 git clone

The git clone command is used to create a local working copy of an existing remote repository.

The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

Syntax

$ git clone URL

4. git add

- Add command is used after checking the status of the files, to add those files to the staging area.
- Before running the commit command, "git add" is used to add any new or modified files.

Syntax: $ **git add Filename**

```
MICLAB@AITS MINGW64 /e/B.HARI KRISHNA/project (master)
$ touch sample.html

MICLAB@AITS MINGW64 /e/B.HARI KRISHNA/project (master)
$ git add sample.html

MICLAB@AITS MINGW64 /e/B.HARI KRISHNA/project (master)
$ git status
On branch master

No commits yet

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

- To add more than one file

$ git add *

Git status command

The status command is used to display the state of the working directory and the staging area. It allows you to see which changes have been staged, which haven't, and which files aren?t being tracked by Git. It does not show you any information about the committed project history. For this, you need to use the git log.

```
$

MICLAB@AITS MINGW64 /e/B.HARI KRISHNA/project (master)
$ git status
On branch master

No commits yet

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

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


MICLAB@AITS MINGW64 /e/B.HARI KRISHNA/project (master)
$
```

Git commit command

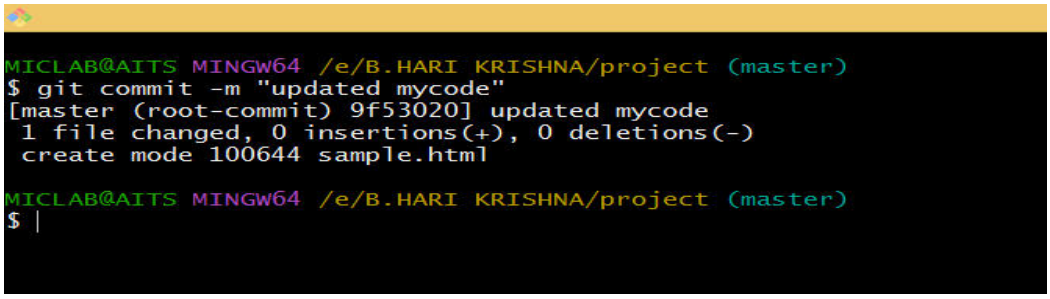
It refers to recording snapshots of the repository at a given time. Committed snapshots will never change unless done explicitly.

Git commit -m

This command changes the head. It records or snapshots the file permanently in the version history with a message.

Syntax

$ git commit -m " Commit Message"



Git commit -a

This command commits any files added in the repository with git add and also commits any files you've changed since then.

Syntax

$ git commit -a

Git push Command

It is used to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repo. It's the complement to git fetch, but whereas fetching imports commits to local branches on comparatively pushing exports commits to remote branches. Remote branches are configured by using the git remote command. Pushing is capable of overwriting changes, and caution should be taken when pushing.

git push -u origin master

This command sends the changes made on the master branch, to your remote repository.

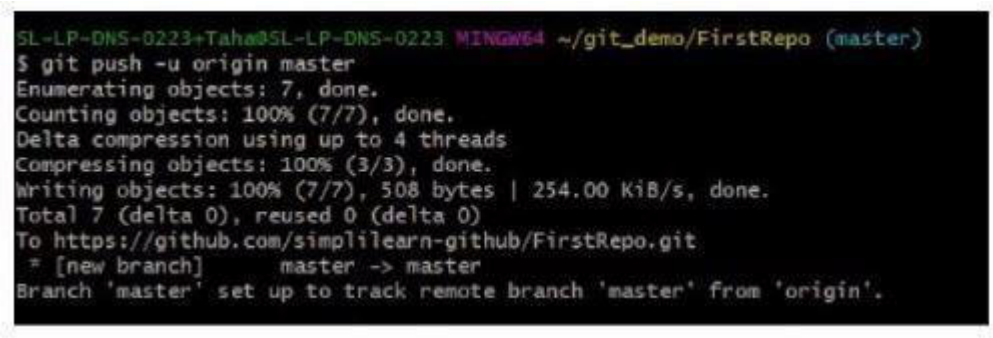
Syntax

$ git push [variable name] master

Git push -all

This command pushes all the branches to the server repository.

Syntax $ git push –all



Git pull command

Pull command is used to receive data from GitHub. It fetches and merges changes on the remote server to your working directory.

Syntax $ git pull URL

Git log Command : This command is used to check the commit history.

Syntax: $ git log

By default, if no argument passed, Git log shows the most recent commits first. We can limit the number of log entries displayed by passing a number as an option, such as -3 to show only the last three entries.

- $ git log
- $ git log --graph
- $ git log --graph --all
- $ git log --oneline –graph --all

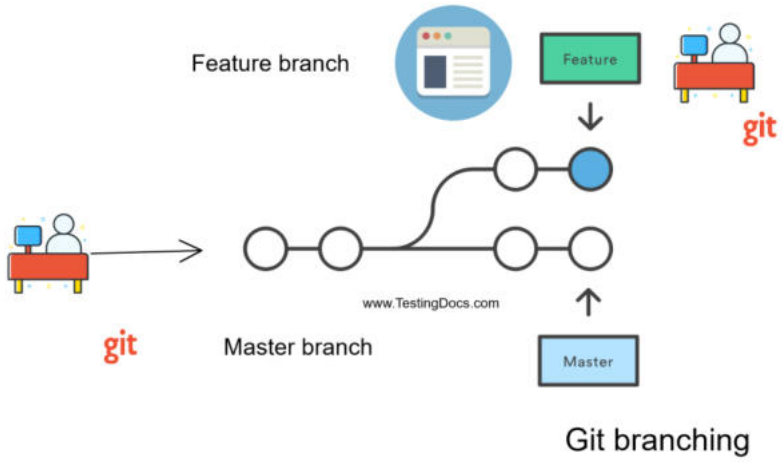
```
[root@ip-172-31-87-33 ~]# git log
commit 382ad7cc174688f8a914e41b21266da33fcdb18e (HEAD -> master)
Author: milky <krishna04@gmail.com>
Date:   Sat Nov 26 09:04:46 2022 +0000

    my new added into repo
[root@ip-172-31-87-33 ~]# git log --graph
* commit 382ad7cc174688f8a914e41b21266da33fcdb18e (HEAD -> master)
  Author: milky <krishna04@gmail.com>
  Date:   Sat Nov 26 09:04:46 2022 +0000

      my new added into repo
[root@ip-172-31-87-33 ~]# git log --graph --all
* commit 382ad7cc174688f8a914e41b21266da33fcdb18e (HEAD -> master)
  Author: milky <krishna04@gmail.com>
  Date:   Sat Nov 26 09:04:46 2022 +0000

      my new added into repo
[root@ip-172-31-87-33 ~]# git log --oneline --graph --all
* 382ad7c (HEAD -> master) my new added into repo
[root@ip-172-31-87-33 ~]#
```

Git remote Command

Git Remote command is used to connect your local repository to the remote server. This command allows you to create, view, and delete connections to other repositories. These connections are more like bookmarks rather than direct links into other repositories. This command doesn't provide real-time access to repositories.

GIT BRANCH

Branch operation allows creating another line of development. We can use this operation to fork off the development process into two different directions. For example, we released a product for 6.0 version and we might want to create a branch so that the development of 7.0 features can be kept separate from 6.0 bug fixes.



Git branching

Git Master Branch

The master branch is a default branch in Git. It is instantiated when first commit made on the project. When you make the first commit, you're given a master branch to the starting commit point. When you start making a commit, then master branch pointer automatically moves forward. A repository can have only one master branch.

Master branch is the branch in which all the changes eventually get merged back. It can be called as an official working version of your project.

Operations on Branches

We can perform various operations on Git branches. The git branch command allows you to create, list, rename and delete branches. Many operations on branches are applied by git checkout and git merge command. So, the git branch is tightly integrated with the git checkout and git merge commands.

Create Branch

You can create a new branch with the help of the git branch command. This command will be used as:

Syntax:

$ git branch <branch name>

```
[root@ip-172-31-87-33 ~]# git branch feature
[root@ip-172-31-87-33 ~]# git branch devops
[root@ip-172-31-87-33 ~]# git branch aws
[root@ip-172-31-87-33 ~]# git branch
  aws
  devops
  feature
* master
```

List Branch

You can List all of the available branches in your repository by using the following command.
Either we can use git branch – list or git branch command to list the available branches in the repository.

Syntax:

$ git branch –list

Or $ git branch

Or $ git branch -a

```
[root@ip-172-31-87-33 ~]# git branch feature
[root@ip-172-31-87-33 ~]# git branch devops
[root@ip-172-31-87-33 ~]# git branch aws
[root@ip-172-31-87-33 ~]# git branch
  aws
  devops
  feature
* master
[root@ip-172-31-87-33 ~]# git branch --list
  aws
  devops
  feature
* master
[root@ip-172-31-87-33 ~]# git branch -a
  aws
  devops
  feature
* master
[root@ip-172-31-87-33 ~]#
```

Here, both commands are listing the available branches in the repository. The symbol * is representing currently active branch.

Delete Branch

You can delete the specified branch. It is a safe operation. In this command, Git prevents you from deleting the branch if it has unmerged changes. Below is the command to do this.

Syntax:

$ git branch -d<branch name>

```
[root@ip-172-31-87-33 ~]# git branch -a
  aws
  devops
  feature
* master
[root@ip-172-31-87-33 ~]# git branch -d aws
Deleted branch aws (was 382ad7c).
[root@ip-172-31-87-33 ~]#
```

This command will delete the existing branch aws from the repository.

The git branch d command can be used in two formats. Another format of this command is git branch D. The ‘git branch D’ command is used to delete the specified branch.

$ git branch -D <branch name>

Delete a Remote Branch

You can delete a remote branch from Git desktop application. Below command is used to delete a remote branch:

Syntax:

$ git push origin -delete <branch name>

Switch Branch

Git allows you to switch between the branches without making a commit. You can switch between two branches with the git checkout command. To switch between the branches, below command is used:
$ git checkout<branch name>

Switch from master Branch

You can switch from master to any other branch available on your repository without making any commit.
Syntax:
$ git checkout <branch name>

Rename Branch

We can rename the branch with the help of the git branch command. To rename a branch, use the below command:

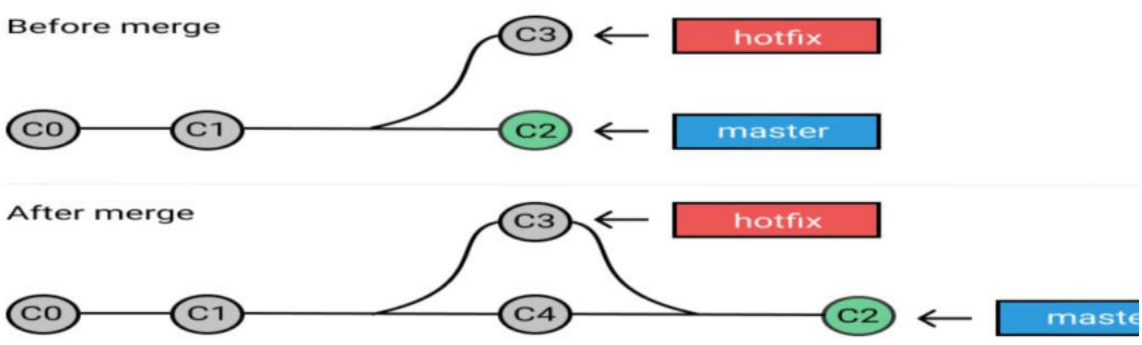
Syntax:

$ git branch -m <old branch name><new branch name>

Merge Branch

Git allows you to merge the other branch with the currently active branch. You can merge two branches with the help of git merge command. Below command is used to merge the branches:

Merging is the way to combine the work of different branches together. This will allow us to branch off, develop a new feature, and then combine it back in.



The diagram above shows us two different branches->new Branch and master. Now, when we merge the work of new Branch into master, it creates a new commit which contains all the work of master and new Branch.

Syntax: $ git merge <branch name>

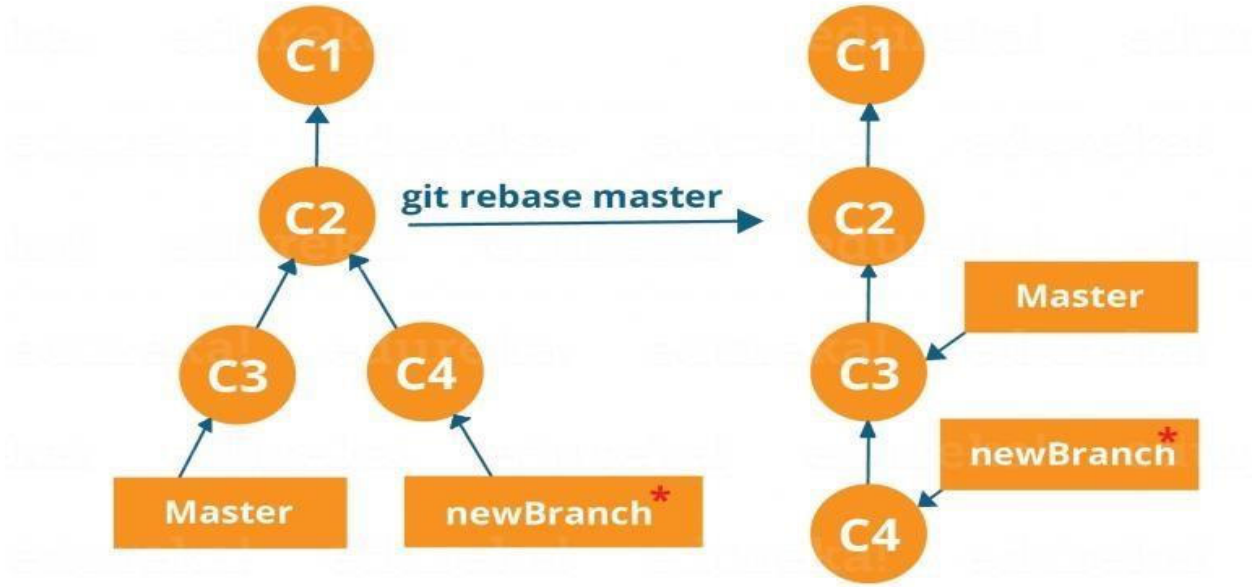
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git merge renamedB1
Already up to date.

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

From the above output, you can see that the master branch merged with renamedB1. Since I have made no-commit before merging, so the output is showing as already up to date.

Rebasing

This is also a way of combining the work between different branches. Rebasing takes a set of commits, copies them and stores them outside your repository.

The advantage of rebasing is that it can be used to make linear sequence of commits. The commit log or history of the repository stays clean if rebasing is done.



Syntax: git rebase master

CREATING A REMOTE REPOSITORY

In Git, the term remote is concerned with the remote repository. It is a shared repository that all team members use to exchange their changes. A remote repository is stored on a code hosting service like an internal server, GitHub, Subversion, and more. In the case of a local repository, a remote typically does not provide a file tree of the project's current state; as an alternative, it only consists of the .git versioning data.

The developers can perform many operations with the remote server. These operations can be a clone, fetch, push, pull, and more.

Checking remote repository

To check the configuration of the remote server, run the git remote command. The git remote command allows accessing the connection between remote and local. If you want to see the original existence of your cloned repository, use the git remote command

Syntax: git remote

The given command is providing the remote name as the origin. Origin is the default name for the remote server, which is given by Git.

Git remote -v:

Git remote supports a specific option -v to show the URLs that Git has stored as a short name. These short names are used during the reading and write operation. Here, -v stands for verbose. We can use –verbose in place of -v.

Syntax: git remote -v

The above output is providing available remote connections. If a repository contains more than one remote connection, this command will list them all

Git Remote Add

When we fetch a repository implicitly, git adds a remote for the repository. Also, we can explicitly add a remote for a repository. We can add a remote as a shot nickname or short name. To add remote as a short name, follow the below command:

Syntax:

$ git remote add <short name><remote URL>

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$ git remote add hd https://github.com/ImDwivedi1/hello-world

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$ git remote -v
hd      https://github.com/ImDwivedi1/hello-world (fetch)
hd      https://github.com/ImDwivedi1/hello-world (push)

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$
```

In the above output, I have added a remote repository with an existing repository as a short name "hd". Now, you can use "hd" on the command line in place of the whole URL. For example, you want to pull the repository

Fetching and Pulling Remote Branch

You can fetch and pull data from the remote repository. The fetch and pull command goes out to that remote server, and fetch all the data from that remote project that you don't have yet. These commands let us fetch the references to all the branches from that remote.

To fetch the data from your remote projects,

$ git fetch <remote>

To clone the remote repository from your remote projects, run the below command:

$ **git clone<remote>**

When we clone a repository, the remote repository is added by a default name "origin." So, mostly, the command is used as git fetch origin.

The git fetch origin fetches the updates that have been made to the remote server since you cloned it. The git fetch command only downloads the data to the local repository; it doesn't merge or modify the data until you don't operate. You have to merge it manually into your repository when you want.

$ **git pull <remote>**

The git pull command automatically fetches and then merges the remote data into your current branch. Pulling is an easier and comfortable workflow than fetching. Because the git clone command sets up your local master branch to track the remote master branch on the server you cloned

Pushing to Remote Branch

If you want to share your project, you have to push it upstream. The git push command is used to share a project or send updates to the remote server. It is used as:

Syntax $ **git push <remote><branch>**

To update the main branch of the project, use the below command:

$ git push origin master

It is a special command-line utility that specifies the remote branch and directory. When you have multiple branches on a remote server, then this command assists you to specify your main branch and repository.

Generally, the term origin stands for the remote repository, and master is considered as the main branch. So, the entire statement "git push origin master" pushed the local content on the master branch of the remote location.

Git Remove Remote

You can remove a remote connection from a repository. To remove a connection, perform the git remote command with remove or rm option. It can be done as:

Syntax:

$ **git remote rm<destination>**

GitHub

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both distributed version control and source code management (SCM) functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.

Features of GitHub

GitHub is a place where programmers and designers work together. They collaborate, contribute, and fix bugs together. It hosts plenty of open source projects and codes of various programming languages.

Some of its significant features are as follows.

- Collaboration
- Integrated issue and bug tracking
- Graphical representation of branches
- Git repositories hosting
- Project management
- Team management
- Code hosting
- Track and assign tasks
- Conversations

Benefits of GitHub

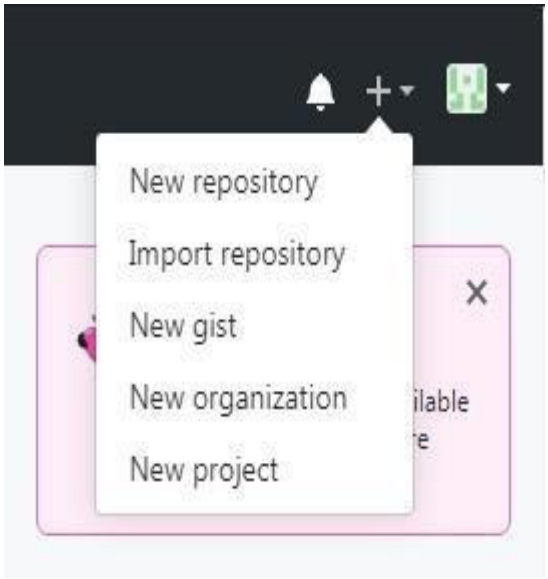
GitHub can be separated as the Git and the Hub. GitHub service includes access controls as well as collaboration features like task management, repository hosting, and team management.

The key benefits of GitHub are as follows.

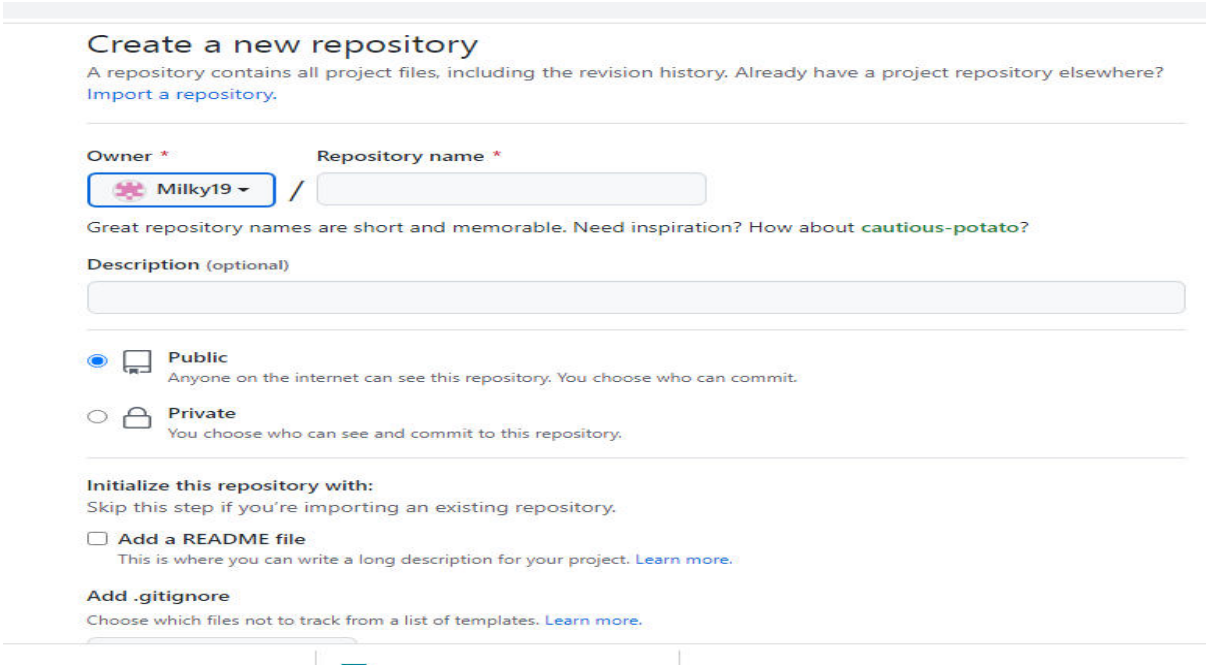
- It is easy to contribute to open source projects via GitHub.
- It helps to create an excellent document.
- You can attract recruiter by showing off your work. If you have a profile on GitHub, you will have a higher chance of being recruited.
- It allows your work to get out there in front of the public.
- You can track changes in your code across

Create a repository

We can create an unlimited public repository and unlimited private repository (For the pro user) on GitHub. To create a repository on GitHub, click on the '+' symbol on the upper right corner on the login screen.



There are some other options available like import repository, gist, organization, and new project. To create a repository, choose new repository option from the given list. When you first log in to your account.



1. Give your repository a name. The name must be unique across all of GitHub.

2. Choose whether the repository should be public or private. Public repositories are visible to everyone, while private repositories are only accessible to you and the people you choose to share them with.
3. Optionally, you can add a description for your repository.
4. Choose whether to initialize the repository with a README file.
5. Select the .gitignore and license templates that apply to your repository, if desired.
6. Finally, click on the "Create repository" button.

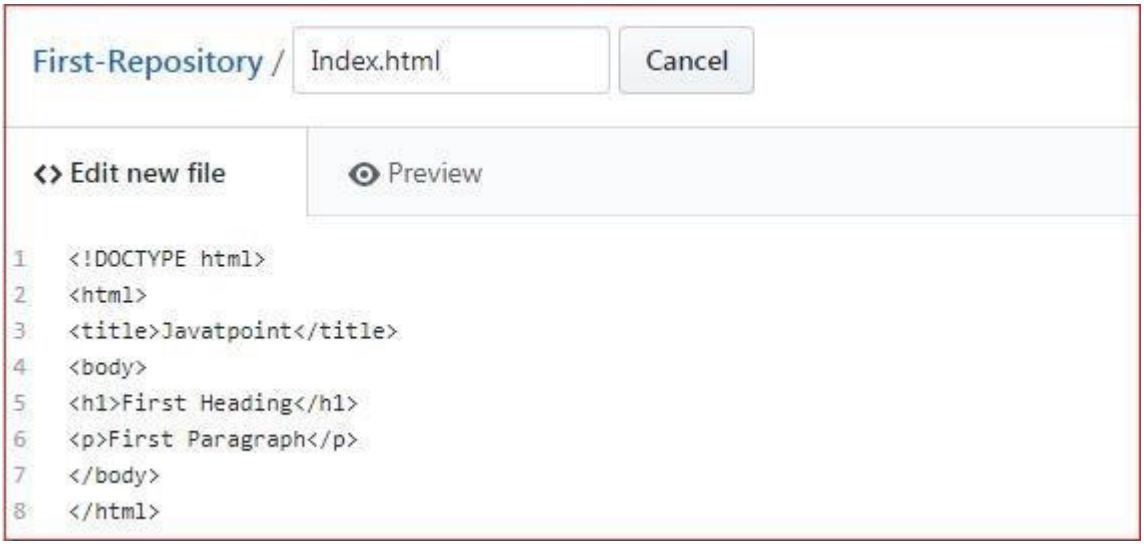
GitHub asks you to learn Git and GitHub without any code. It will ask you to read the hello world guide for the first uses. Also, you can create a repository (Project) from here.

Click on the new repository option and then fill the required details like repository name, description, and select the access of this repository. You can also initialize the repository with a README file. After filling all the details, click on the Create Repository option. It will create a repository created .

Create a file

In GitHub, creating a file is a straight forward process. Let's create a file in our newly created repository. Consider the below snap of our repository.

There are distinct options available to add files to the repository. GitHub allows us to design and upload files. To create a file, click on the 'Create new file' option. It will open a file structure.



Enter the file name on the box and type the code on the editor area.

At the bottom of the page, the commit options are available.

Click on the 'commit new file' option. We have successfully added and committed a new file to our repository.

We can edit and delete this file from our project. There are many options available, like edit, delete, Raw, Blame, and history.

UNIT – V

JENKINS: Why continuous integration, Introduction to Continuous Integration, Release and relation with Devops, Jenkins Introduction and setup, Jenkins projects/jobs, Master/Slave concept, Jenkins poll scm, build periodically, web hooks, Jenkins pipeline, Jenkins plugins, Jenkins CLI and Jenkins administration

Why continuous integration, Introduction to Continuous Integration

Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example Git, Maven 2 project, Amazon EC2, HTML publisher etc.

History of Jenkins

- Kohsuke Kawaguchi, who is a Java developer, working at SUN Microsystems, was tired of building the code and fixing errors repetitively.
- In 2004, he created an automation server called **Hudson** that automates build and test task.
- In 2011, Oracle who owned Sun Microsystems had a dispute with Hudson open source community, so they forked Hudson and renamed it as **Jenkins**.
- Both Hudson and Jenkins continued to operate independently. But in short span of time, Jenkins acquired a lot of contributors and projects while Hudson remained with only 32 projects. Then with time, Jenkins became more popular, and Hudson is not maintained anymore.

CONTINUOUS DEPLOYMENT

Continuous Deployment (CD) is a software development practice that builds upon the principles of Continuous Integration (CI) and Continuous Delivery (CD). It is an approach that aims to automate the entire software release process, from code changes to production deployment, with minimal human intervention. The primary goal of Continuous Deployment is to enable organizations to deliver new features, updates, and bug fixes to end-users quickly, reliably, and with reduced risk.

The core components of a Continuous Deployment pipeline typically include:

Version Control System: A centralized repository, such as Git, where developers commit their code changes.

Continuous Integration (CI): A practice where every code change is automatically built, compiled, and subjected to a suite of automated tests to ensure its quality and functionality.

Automated Testing: A comprehensive set of tests, including unit tests, integration tests, functional tests, and potentially other types of tests (e.g., load testing, security testing), are executed automatically to validate the code changes.

Continuous Delivery (CD): Once the code changes pass all the tests, they are automatically packaged and made available for deployment to various environments (e.g., staging, production).

Automated Deployment: In Continuous Deployment, if the code changes successfully pass through the previous stages, they are automatically deployed to the production environment without manual intervention.

Monitoring and Feedback: After deployment, the application's performance, behavior, and user feedback are closely monitored to detect any issues or regressions promptly. This feedback loop enables quick identification and resolution of problems, which can then be addressed in the next iteration of the development cycle.

Continuous Deployment often works best with a trunk-based development model, where all code changes are merged into a single trunk (mainline) instead of using long-lived feature branches. This approach helps to avoid merge conflicts and ensures that the codebase is always in a deployable state.

To support Continuous Deployment, organizations typically adopt several practices and techniques, such as:

Microservices Architecture: Breaking down monolithic applications into smaller, independently deployable microservices can facilitate Continuous Deployment by allowing teams to update and deploy individual services without impacting the entire application.

Feature Flags or Feature Toggles: These mechanisms enable organizations to release new features to production while keeping them disabled or enabled for specific user segments, allowing for controlled rollouts and rollbacks if needed.

Canary Releases or Blue/Green Deployments: These deployment strategies involve releasing new versions of the application to a subset of users or servers first, monitoring for issues, and then gradually rolling out the changes to the entire user base or infrastructure.

Automated Rollbacks: Mechanisms to automatically revert to a previous stable version of the application in case of critical issues or failures during deployment.

Immutable Infrastructure: Treating infrastructure as immutable and creating new environments for each deployment, rather than modifying existing ones, can simplify the deployment process and reduce the risk of configuration drift.

Continuous Deployment is particularly suitable for organizations that have a strong culture of automation, extensive test coverage, and a willingness to embrace frequent releases. It is commonly adopted in cloud-native architectures, web applications, and software-as-a-service (SaaS) products, where the ability to deliver new features and updates rapidly can provide a competitive advantage.

However, it's important to note that Continuous Deployment may not be appropriate for all types of applications or organizations, especially those with stringent regulatory requirements, complex

deployment processes, or where the cost of potential issues in production is extremely high. In such cases, organizations may opt for a more controlled approach, such as Continuous Delivery, where deployments to production are still triggered manually after passing through the automated stages.

CONTINUOUS MONITORING

Continuous Monitoring is an essential practice in modern software development and operations, particularly when adopting approaches like Continuous Deployment. It involves continuously monitoring the health, performance, and behavior of an application or system in production to detect and respond to issues promptly.

The primary reasons for implementing Continuous Monitoring are:

Early Issue Detection: Continuous Monitoring enables early detection of issues, such as performance degradation, errors, or unexpected behavior, as soon as they occur in the production environment. This early detection allows for swift remediation, minimizing the impact on end-users and preventing further cascading issues.

Validation of Deployments: When practicing Continuous Deployment, new code changes are automatically deployed to production frequently. Continuous Monitoring helps validate that these deployments are successful and that the application is functioning as expected after each deployment.

Proactive Identification of Regressions: By monitoring application metrics and user behavior, Continuous Monitoring can proactively identify regressions or unintended consequences introduced by recent code changes or deployments. This allows for prompt investigation and resolution before the issues escalate.

Application Health and Performance Insights: Continuous Monitoring provides insights into the overall health and performance of the application, enabling teams to identify bottlenecks, optimize resource utilization, and make informed decisions about scaling or architectural changes.

Compliance and Audit Trail: In regulated industries or environments with strict compliance requirements, Continuous Monitoring can help maintain an audit trail of the application's behavior, performance metrics, and any issues or incidents that occurred, aiding in compliance and auditing processes.

To implement Continuous Monitoring effectively, organizations typically employ a combination of tools and techniques, such as:

Application Performance Monitoring (APM) tools: These tools monitor various aspects of an application's performance, such as response times, error rates, resource utilization, and user experience metrics.

Log Aggregation and Analysis: Centralized logging and log analysis tools collect and analyze logs from various components of the application and infrastructure, enabling teams to identify and investigate issues based on log patterns or error messages.

Distributed Tracing: In complex, distributed systems, distributed tracing tools provide end-to-end visibility into request flows, helping to identify bottlenecks or performance issues across different components or services.

Synthetic Monitoring: This technique involves simulating user interactions or API calls to proactively monitor the application's availability, functionality, and performance from various locations or scenarios.

Alerting and Notification Systems: Continuous Monitoring tools integrate with alerting and notification systems to promptly notify the responsible teams or individuals when issues or anomalies are detected, enabling rapid response and resolution.

Dashboards and Reporting: Visualization tools and dashboards provide real-time insights into the application's performance, health metrics, and any deviations from expected behavior, allowing teams to monitor the system's state at a glance.

Continuous Monitoring is a critical practice in modern software development and operations, especially when adopting Continuous Deployment. It enables organizations to maintain high levels of application reliability, performance, and availability, while also facilitating rapid detection and resolution of issues, minimizing the impact on end-users and ensuring a seamless user experience.

Advantages and Disadvantages of using Jenkins

Advantages of Jenkins

- It is an open source tool. o It is free of cost. o It does not require additional installations or components. Means it is easy to install. o Easily configurable.
- It supports 1000 or more plugins to ease your work. If a plugin does not exist, you can write the script for it and share with community.
- It is built in java and hence it is portable.
- It is platform independent. It is available for all platforms and different operating systems. Like OS X, Windows or Linux.
- Easy support, since it open source and widely used. o Jenkins also supports cloud based architecture so that we can deploy Jenkins in cloud based platforms.

Disadvantages of Jenkins

- Its interface is out dated and not user friendly compared to current user interface trends.
- Not easy to maintain it because it runs on a server and requires some skills as server administrator to monitor its activity.
- CI regularly breaks due to some small setting changes. CI will be paused and therefore requires some developer's team attention.

JENKINS RELEASE AND RELATION WITH DEVOPS

- Release and deployment are integral parts of the software development process and are closely related to DevOps practices.
- DevOps is a culture, set of practices and tools that aims to automate and streamline the process of software development, testing, and deployment. It emphasizes collaboration between development and operations teams, and the use of automation and monitoring tools to improve the speed and reliability of software delivery.

- The release process refers to the process of making a new version of a software product available to users. This can include tasks such as building and testing the software, creating and updating documentation, and packaging and distributing the software.
- Deployment, on the other hand, refers to the process of making a release available to users by installing it on one or more servers or devices. This can include tasks such as configuring the environment, provisioning resources, and setting up monitoring and logging.
- In a DevOps environment, the release and deployment process is automated and integrated into the overall software development process. This allows development and operations teams to work together more closely, and to deliver new features and updates to users more quickly and reliably.
- It is important to note that DevOps practices are not only about automation, but also about collaboration, communication, and culture. The release and deployment process is one of the key areas where DevOps practices come into play. By automating and streamlining the release and deployment process, teams can reduce the time and effort required to deliver software

DevOps Release Management

- In DevOps, release management is also about planning, scheduling and controlling the software development and delivery process. But, in DevOps, both developers and IT operations collaborate from the beginning of the process to the end – allowing for fewer, shorter feedback loops and faster releases.
- DevOps teams share accountability for the services they deliver, own their code and take on-call responsibilities. With software developers and IT professionals involved in the entire delivery lifecycle and on-call, incidents are detected and resolved faster – both during the release process and after.

Streamlined CI/CD and QA

- The shift-left idea is common in DevOps. By moving QA, automation and testing earlier in the development lifecycle, the DevOps team can identify potential issues faster. This reduces the amount of time spent in feedback loops and allows the delivery pipeline to continue moving forward. The more you can integrate testing with development workflows, the easier it will be to maintain a consistent CI/CD pipeline.

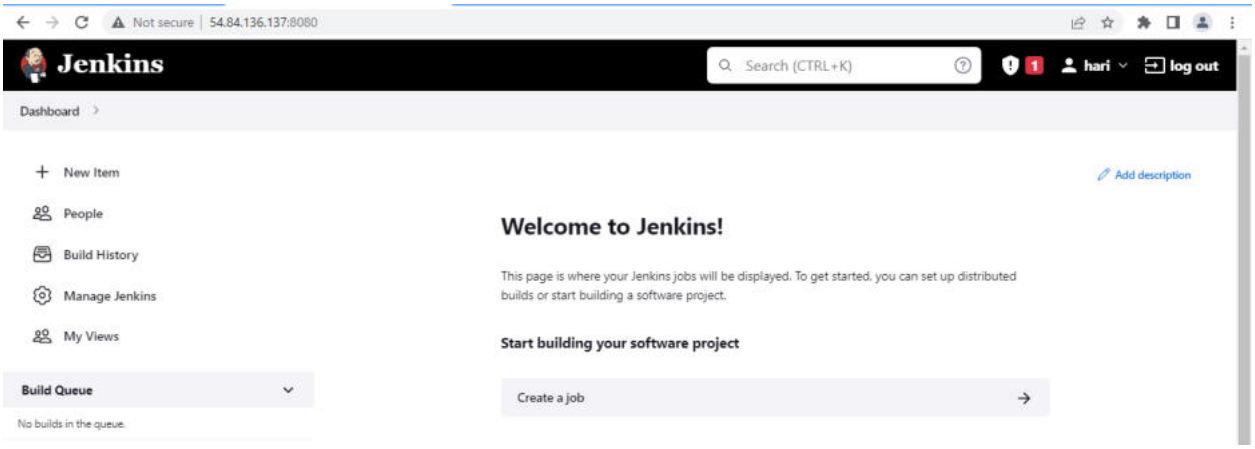
Use Automation to Your Advantage

- Rule number one in DevOps – automate anything that can improve the efficiency of your people, processes and technology. Whether it's on the software development, QA, or IT

- Operations side of the fence, automation should be used to reduce human error and make day-to-day operations easier for your people. Allowing your team to spend more time on strategic thinking and less time on day-to-day tasks, you'll be able to consistently deliver reliable services to your customers.

Jenkins Introduction and setup

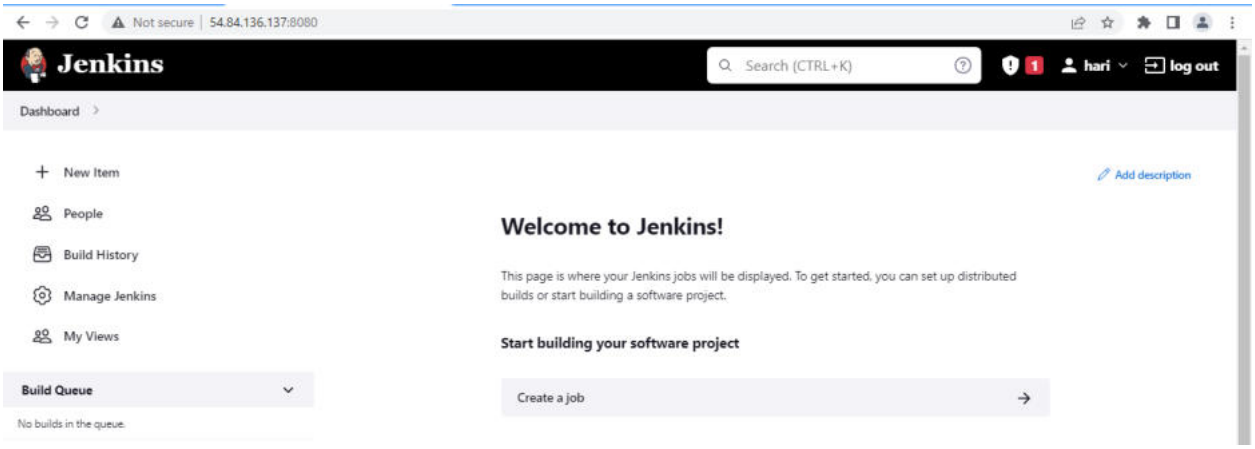
- Log in to your AWS Management Console.
- Go to EC2 → Instances → Launch Instance.
- Choose Amazon Linux 2 AMI (x86_64).
- Select an instance type — e.g., t2.micro (Free tier).
- Configure details, add storage (default is fine).
- Add security group rules:
- Port 22 → SSH (for login)
- Port 8080 → Jenkins web interface
- Launch instance and download your key pair (.pem).
- amazon-linux-extras install epel -y
- amazon-linux-extras install java11 -y
- amazon-linux-extras install java-openjdk11 -y
- java --version
- yum install jenkins -y
- sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
- sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
- yum install jenkins -y
- systemctl restart jenkins
- systemctl status jenkins
- cat /var/lib/jenkins/secrets/initialAdminPassword



Jenkins projects/jobs

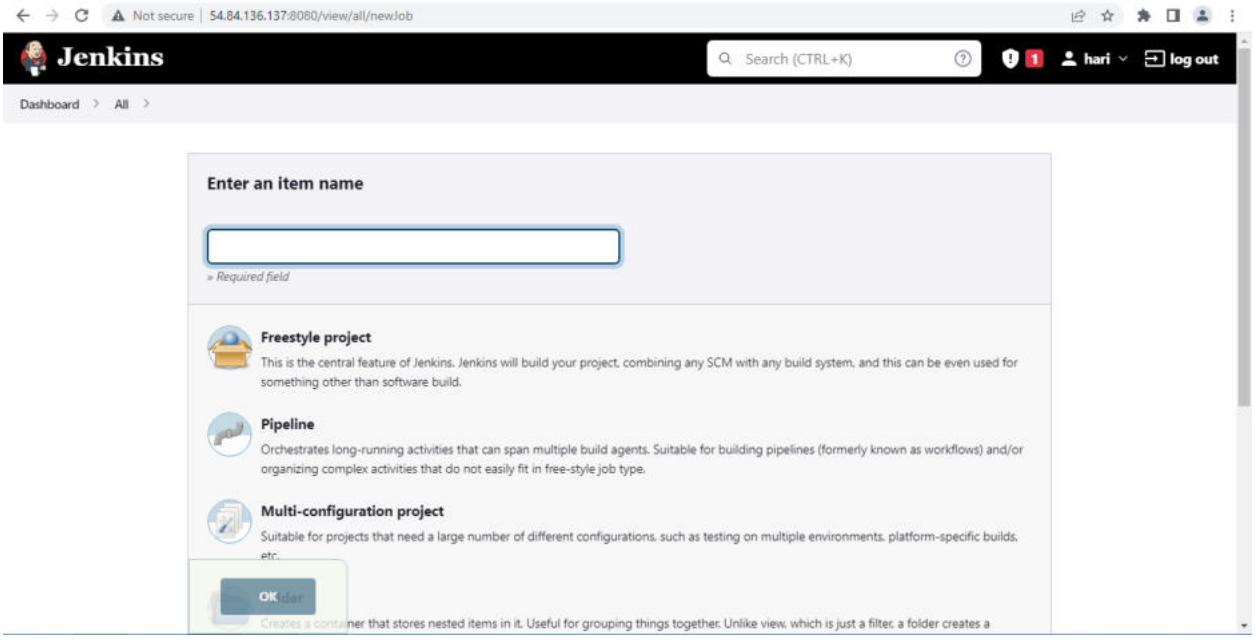
Jenkins Freestyle Project is a repeatable build job, script, or pipeline that contains steps and post-build actions. It is an improved job or task that can span multiple operations. It allows you to configure build triggers and offers project-based security for your Jenkins project. It also offers plugins to help you build steps and post-build actions.

The types of actions you can perform in a Jenkins build step or post-build action are quite limited. There are many standard plug-in available within a Jenkins Freestyle Project.



Create New Item

Click on "**New Item**" at the top left-hand side of your dashboard.

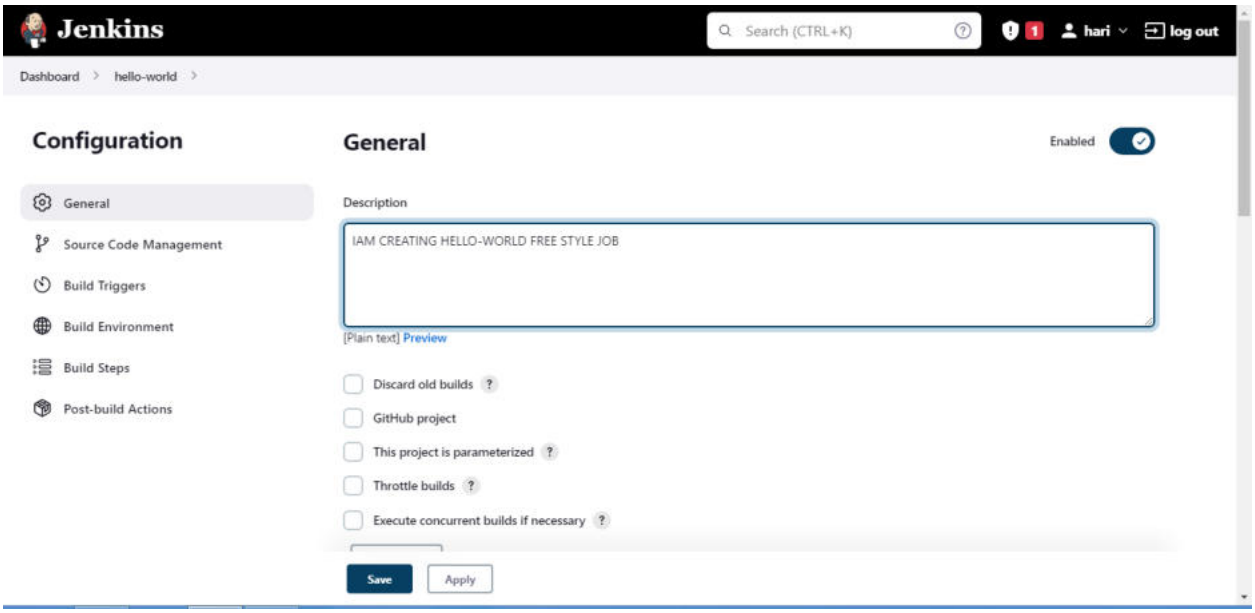


Enter Item details

- Enter the name of the item you want to create. We shall use the "Hello world".
- Select **Freestyle project**
- Click **Okay**

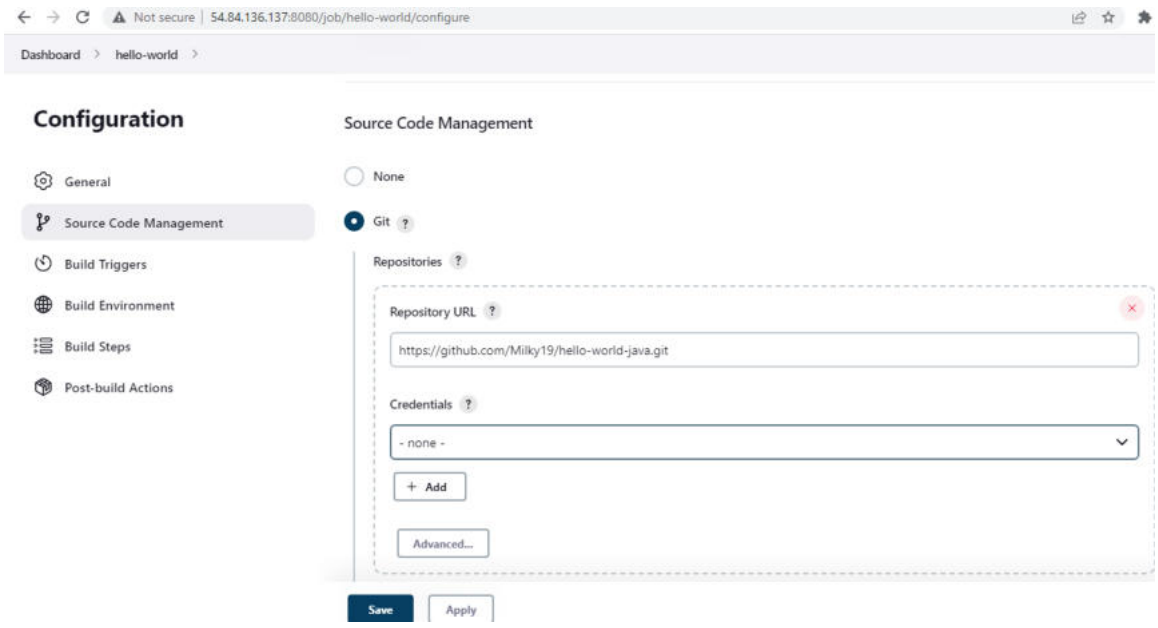
Enter Project details

- Enter the details of the project you want to test.



Enter repository URL

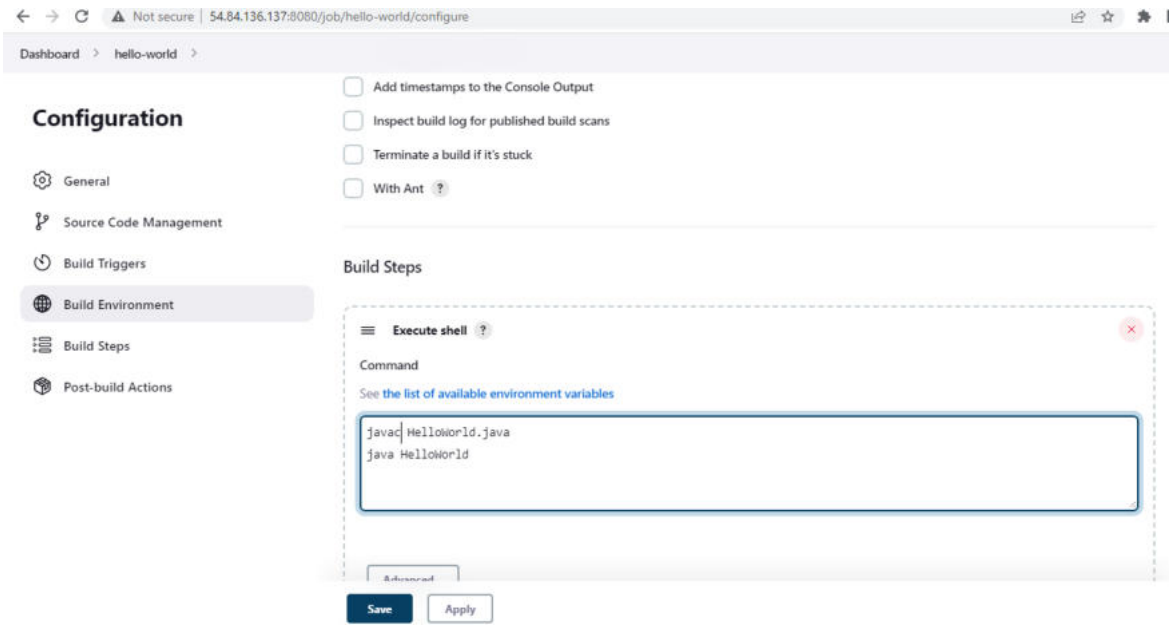
- Under Source Code Management, Enter your repository URL. We have a test repository located at Github



- It is also possible for you to use a local repository.
- If your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then pull the source code from your GitHub repository.

Tweak the settings

- Now that you have provided all the details, it's time to build the code. Tweak the settings under the **build** section to build the code at the time you want. You can even schedule the build to happen periodically, at set times.
- Click on "**Add build step**"
- Click on "**Execute shell command**" and add the commands you want to execute during the build process.



- I have added the following linux commands:
- javac HelloWorld.java
- java HelloWorld

Save the project

When you have entered all the data,

- Click **Apply**
- **Save** the project.
- **Build Source code**
- Now, in the main screen, click the **Build Now** button on the left-hand side to build the source code.

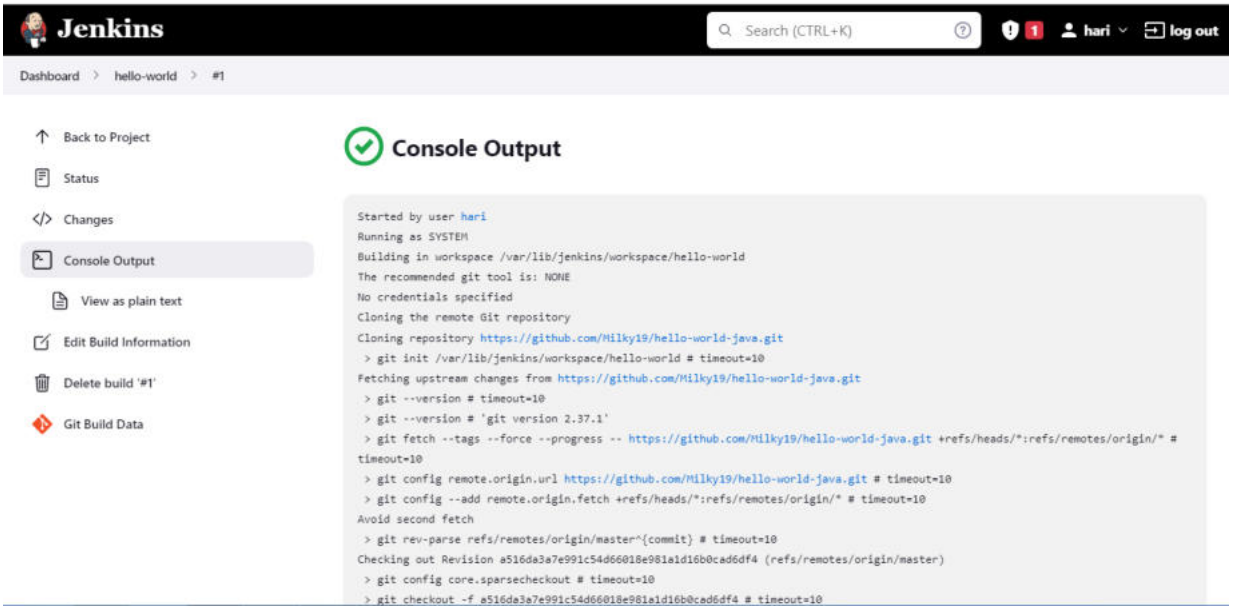
Check the status

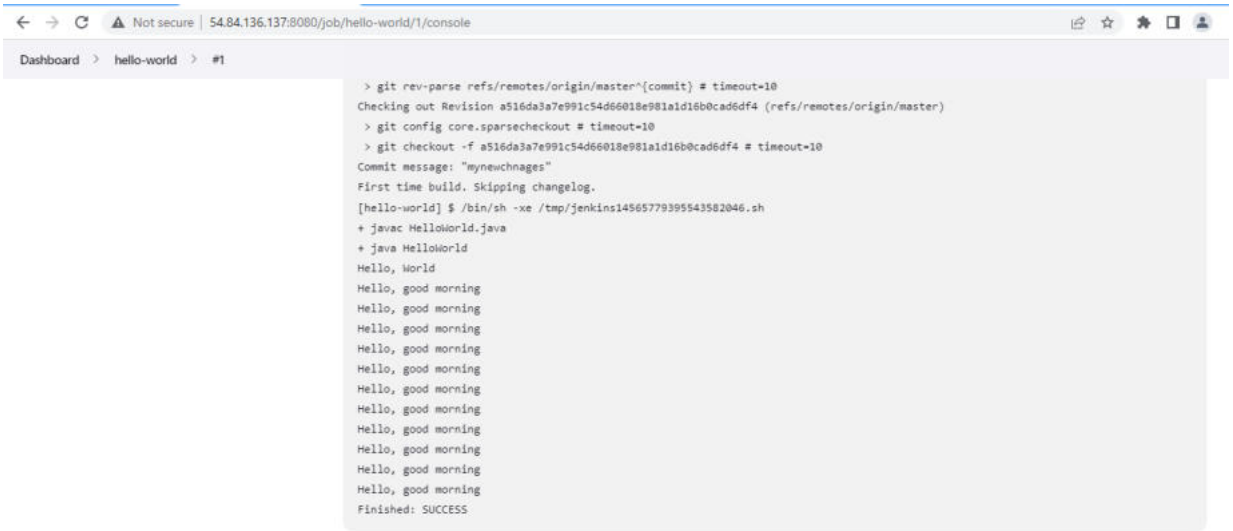
- After clicking on **Build now,** you can see the status of the build you run under **Build History**.

See the console output

- Click on the **build number** and then Click on **console output** to see the status of the build you run. It should show you a success message

Sample Output:





Master/Slave concept

Jenkins Architecture

Jenkins follows Master-Slave architecture to manage distributed builds. In this architecture, slave and master communicate through TCP/IP protocol.

Jenkins architecture has two components:

- Jenkins Master/Server o
- Jenkins Slave/Node/Build

Jenkins Master

The main server of Jenkins is the Jenkins Master. It is a web dashboard which is nothing but powered from a war file. By default it runs on 8080 port. With the help of Dashboard, we can configure the jobs/projects but the build takes place in Nodes/Slave. By default one node (slave) is configured and running in Jenkins server. We can add more nodes using IP address, user name and password using the ssh, jnlp or webstart methods.

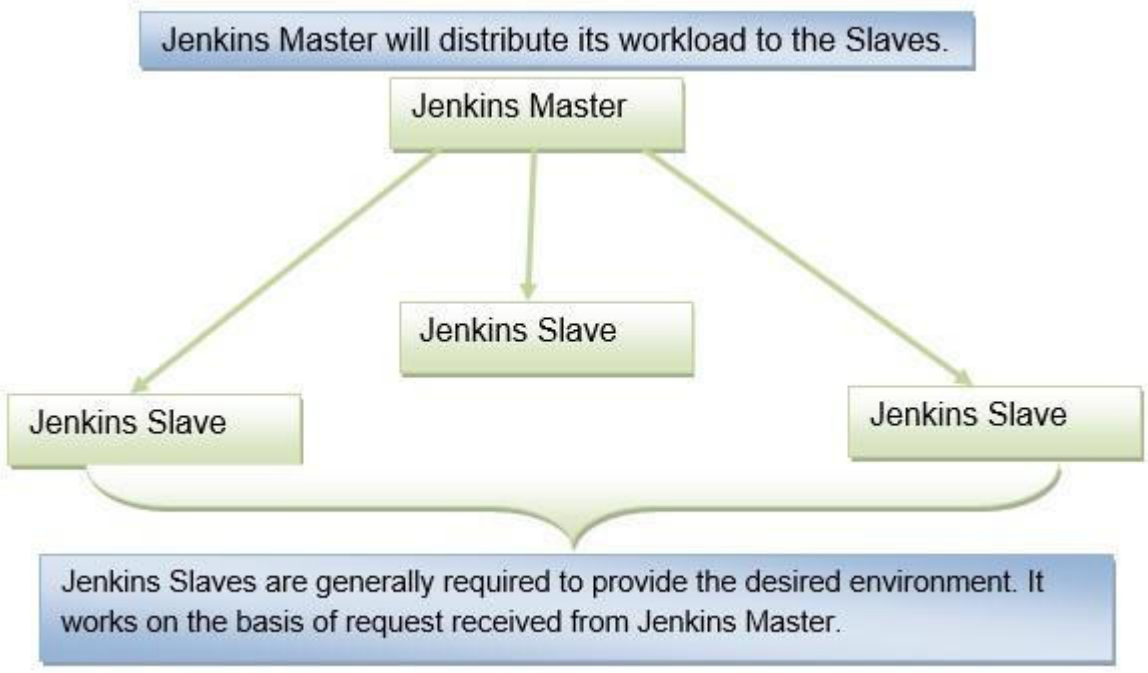
The server's job or master's job is to handle:

- Scheduling build jobs.
- Dispatching builds to the nodes/slaves for the actual execution.
- Monitor the nodes/slaves (possibly taking them online and offline as required).
- Recording and presenting the build results.
- A Master/Server instance of Jenkins can also execute build jobs directly.

Jenkins Slave

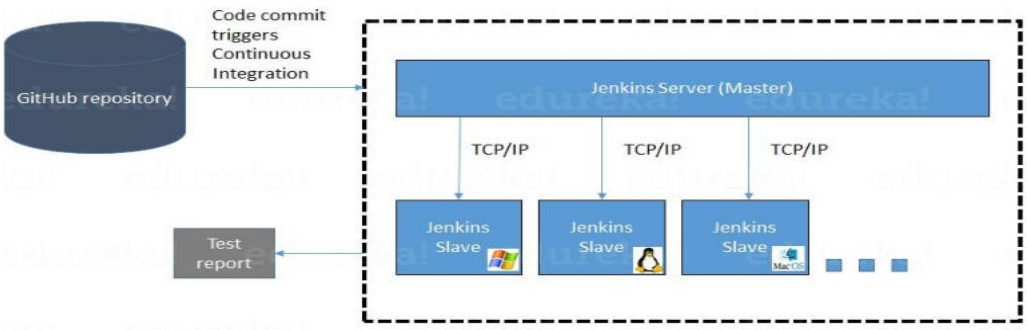
Jenkins slave is used to execute the build jobs dispatched by the master. We can configure a project to always run on a particular slave machine, or particular type of slave machine, or simple let the Jenkins to pick the next available slave/node.

As we know Jenkins is developed using Java is platform independent thus Jenkins Master/Servers and Slave/nodes can be configured in any servers including Linux, Windows, and Mac.



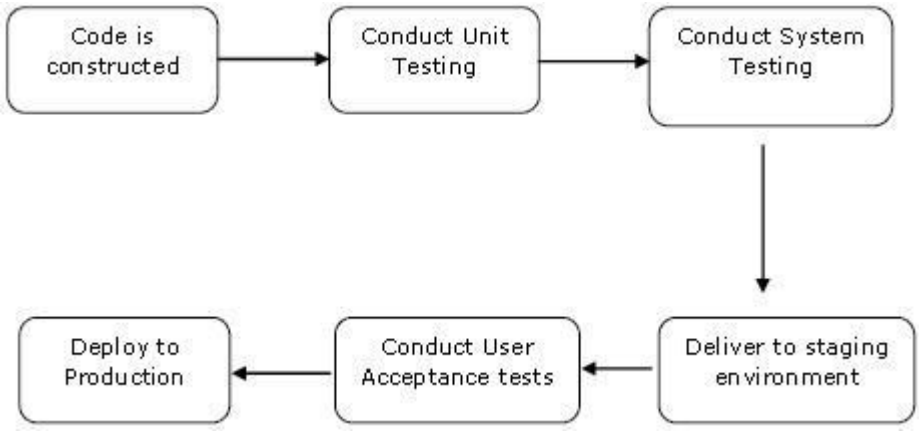
The above diagram is self explanatory. It consists of a Jenkins Master which is managing three Jenkins Slaves.

The diagram below represents the same:



Jenkins - Continuous Deployment

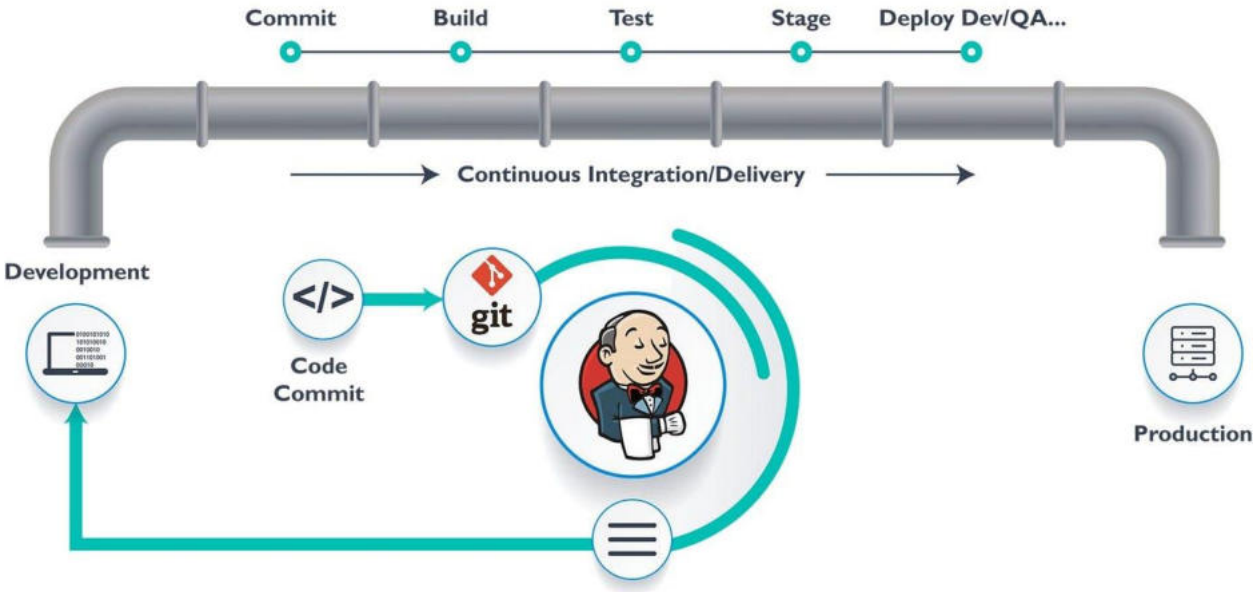
Jenkins is used in providing good support for continuous deployment and delivery. The flow of a software development till the deployment



Jenkins poll scm, build periodically, web hooks

Jenkins pipeline

A Jenkins Pipeline is a suite of plugins that enables the implementation and integration of continuous delivery pipelines within Jenkins. It provides an extensible set of tools for modeling simple to complex delivery pipelines "as code." This "Pipeline as Code" approach involves defining the entire continuous delivery process, from version control to deployment, in a text file called a Jenkinsfile. This file is typically stored and versioned in a project's source control repository.



Benefits of Jenkins Pipelines:

Automation: Automates the entire CI/CD process, reducing manual effort and potential for human error.

Version Control: The Jenkinsfile is versioned with the project code, ensuring consistency and traceability of the pipeline definition.

Visibility: Provides a clear, visual representation of the delivery process and its current status.

Flexibility: Supports a wide range of project structures, technologies, and deployment environments.

Reusability: Pipeline definitions can be reused across different projects or branches.

Resilience: Pipelines can be designed to be robust, with features like automatic retries and error handling.

In Jenkins, pipelines are used to automate the build, test, and deployment process. There are mainly two types of pipelines

Declarative Pipeline

- ✓ Introduced to make pipelines easier to write and understand.
- ✓ Uses a structured and predefined syntax (Groovy-based).
- ✓ Best suited for most CI/CD use cases.
- ✓ Defined inside a pipeline { } block.

Example:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo "Building the project..."
            }
        }
        stage('Test') {
            steps {
                echo "Running tests..."
            }
        }
        stage('Deploy') {
            steps {
                echo "Deploying the project..."
            }
        }
    }
}
```

1. pipeline { }

- ✓ The root block in a Declarative Pipeline.
- ✓ Everything goes inside this block

2. agent any

- ✓ Defines where the pipeline (or a stage) should run.
- ✓ Agent any means Jenkins can run it on any available agent/node

3. stages { }

- ✓ A container for multiple stages.
- ✓ Every stage inside it represents a phase of the pipeline (Build, Test, Deploy, etc.).

4. stage('Name') { }

- ✓ Defines a single phase of the pipeline.
- ✓ Each stage should represent a logical step in your CI/CD workflow.

5. steps { }

- ✓ Inside each stage, steps defines what actions to perform.
- ✓ Steps can be shell commands, Jenkins steps, plugins, etc.

Scripted Pipeline

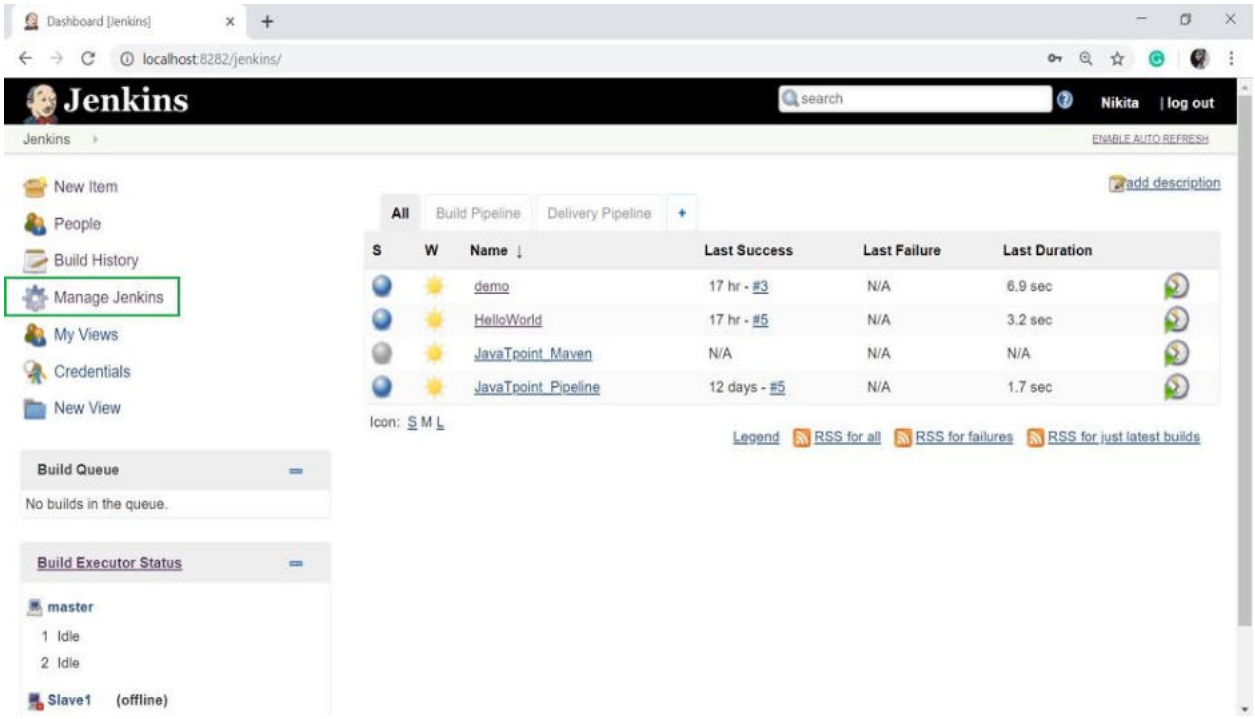
- The original pipeline style in Jenkins.
- Uses **Groovy scripting** for full flexibility and control.
- Defined inside a node { } block

```
node {
    stage('Build') {
        echo "Building the project..."
    }
    stage('Test') {
        echo "Running tests..."
    }
    stage('Deploy') {
        echo "Deploying the project..."
    }
}
```

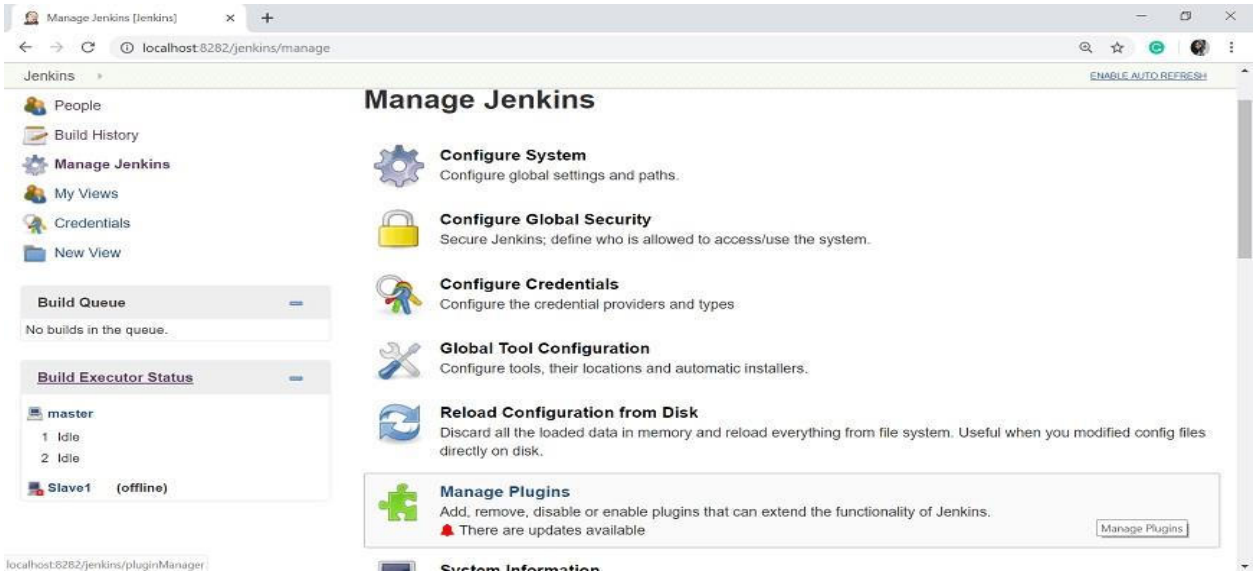
Jenkins - Managing Plug-in

Jenkins provides a variety of plugins for a different task.

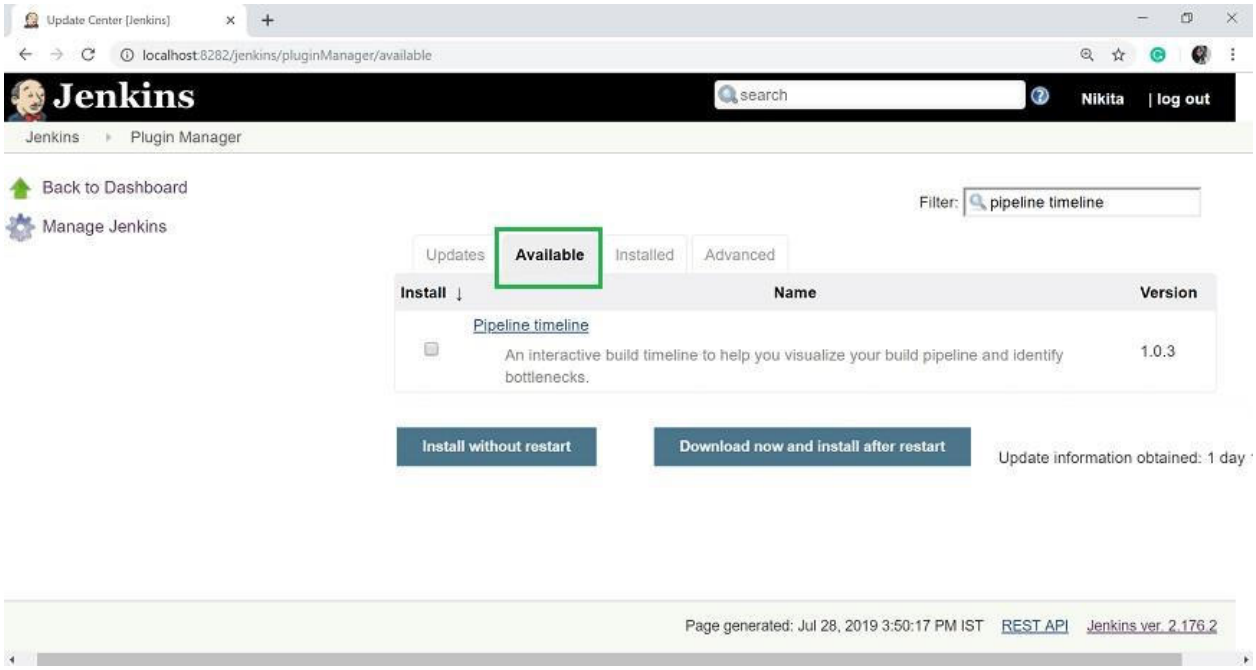
Step 1: To install a plug in, go to the Jenkins Dashboard and click on Manage Jenkins**.**



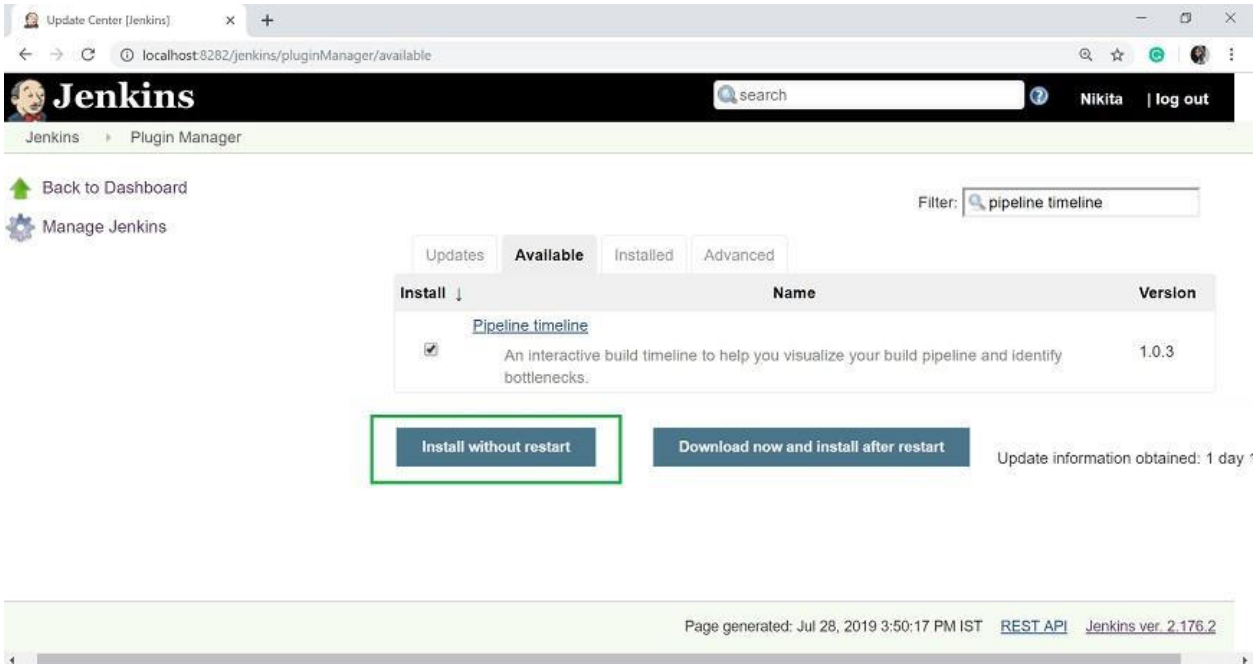
Step 2: **Scroll down and select** Manage Plugins**.**



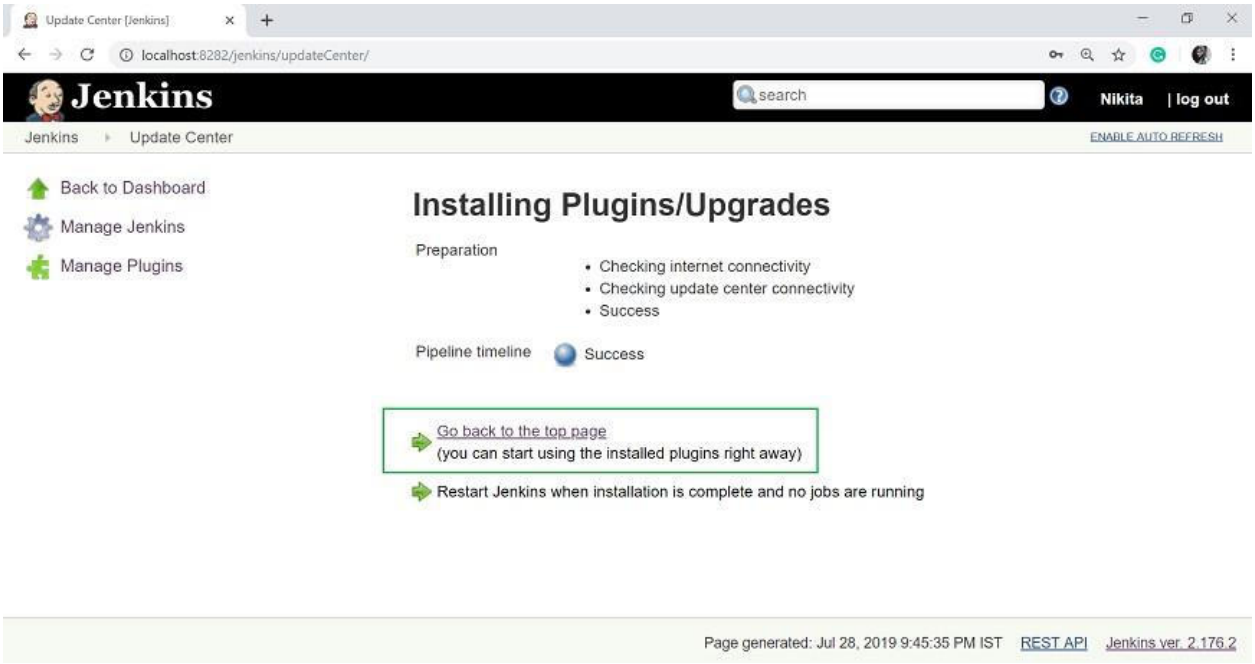
Step 3: **Go to the** Available **tab and in the filter option, search for the plugins which you want to install**



Step 4: **Select that plugins and click on** Install without restart **button. You can also** choose Download now and install after restart **button**



Once the installation has been completed successfully, click on Go back to the top page **link.**



Installing the plugin manually

Sometimes it may be necessary to install an older version of a plugin, in such a case, you install the plugin manually.

Step 1: For this, first, you have to download the plugin from the relevant plugin page on the Jenkins website. Then go to the Jenkins dashboard and select **Manage Jenkins**.

Step 2: Select **Manage Plugins**

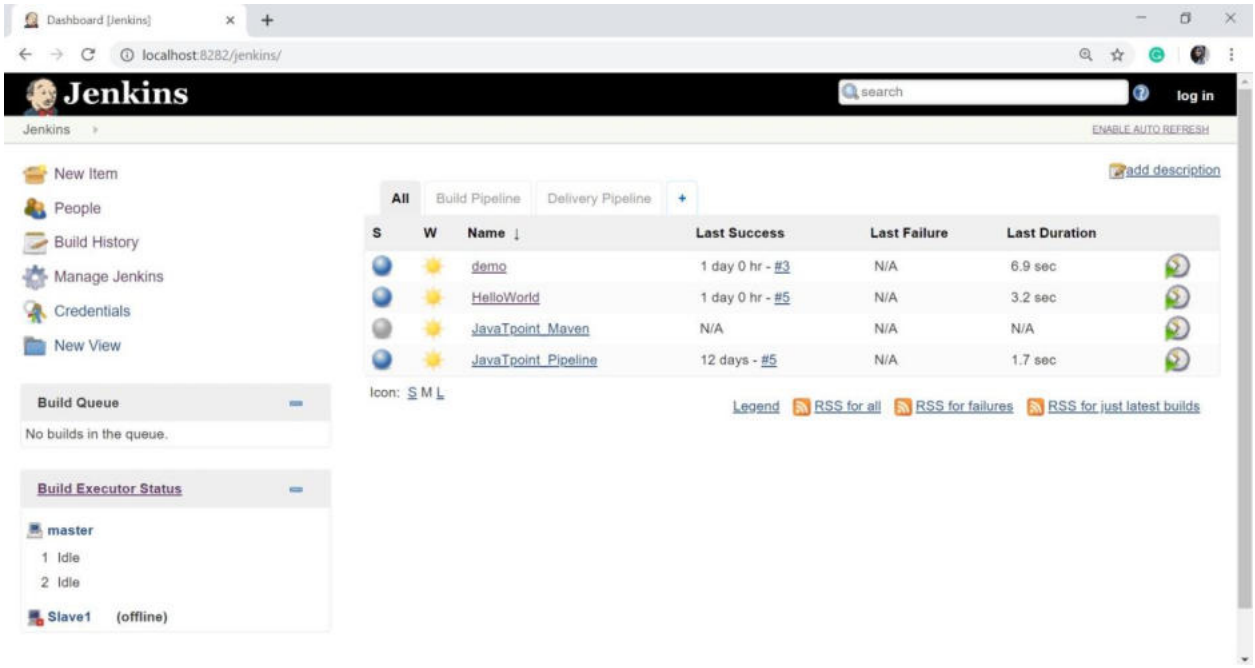
Step 3: Click on **Advanced** tab and on the upload plugin section browse the downloaded plugin's **Step 4:** Then click on the **Upload** button.

Jenkins Backup Plugins

It is important to have a Jenkins backup with its data and configurations. It includes job configs, plugins, build logs, plugin configuration, etc.

Jenkins provides a backup plugin which can be used to get backup critical configuration settings related to

Step 1: Go to the Jenkins dashboard screen and click on **Manage Jenkins**.

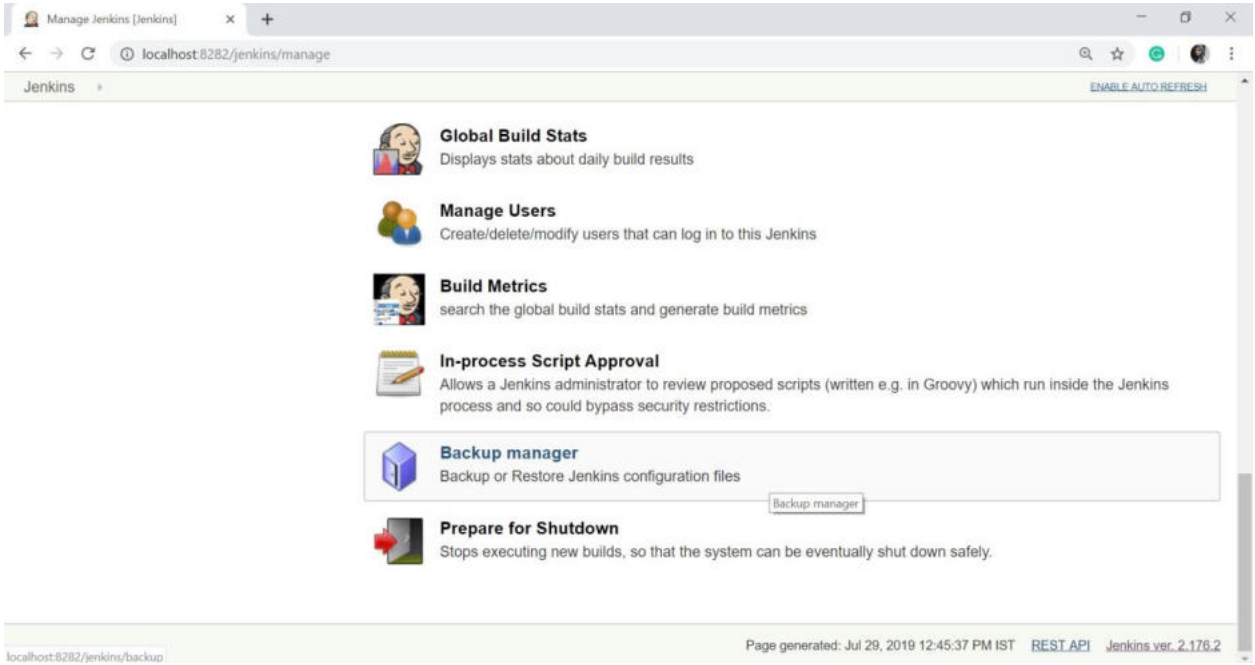


Step 2: Select **Manage Plugins**.

Step 3: Go to the Available tab and search for "**Backup**" plugin on the filter field

Step 4: After successful installation, click on **Go back to the top page** link.

Step 5: Now, when you go to the **Manage Jenkins** option, and scroll down, you will see **Backup manager** as an option. Select this option.



Step 6: Click on **setup**

Step 7: Here, enter the directory name in the field of **Backup directory**. Please make sure that it is on another drive which is different from the drive where your Jenkins instance is installed. Then, click on

t**Step 8:** Click on the **Backup Hudson configuration** link from the backup manager page to initiate the backup..

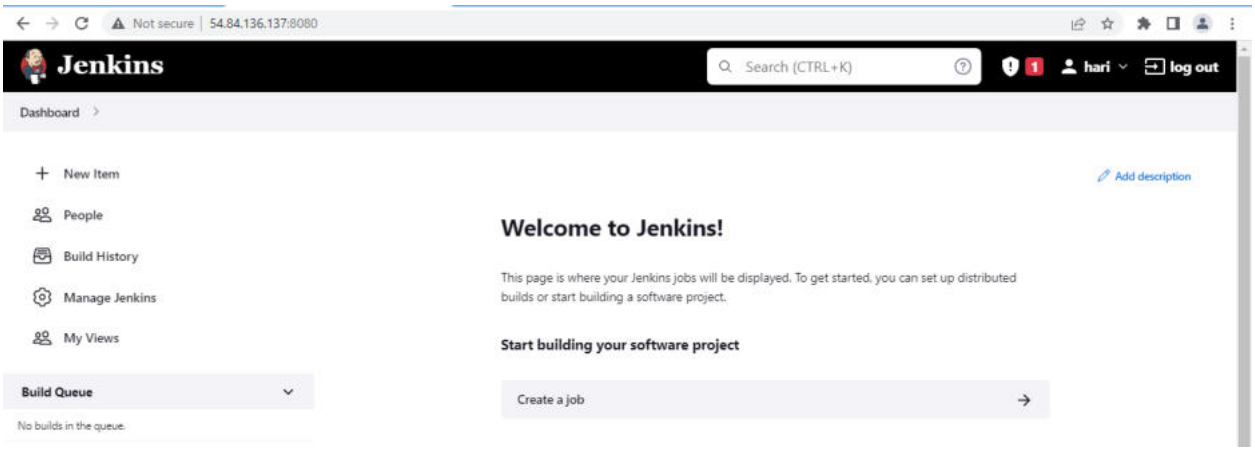
The next page will show the backup status.

To recover from a backup, go to the **Backup manager** screen, and click on **Restore Hudson configuration**.

Jenkins CLI and Jenkins administration

To manage Jenkins, click on the "Manage Jenkins" option from the left hand of the Jenkins Dashboard page.

When you click on the Manage Jenkins, you will get the various options to manage the Jenkins:



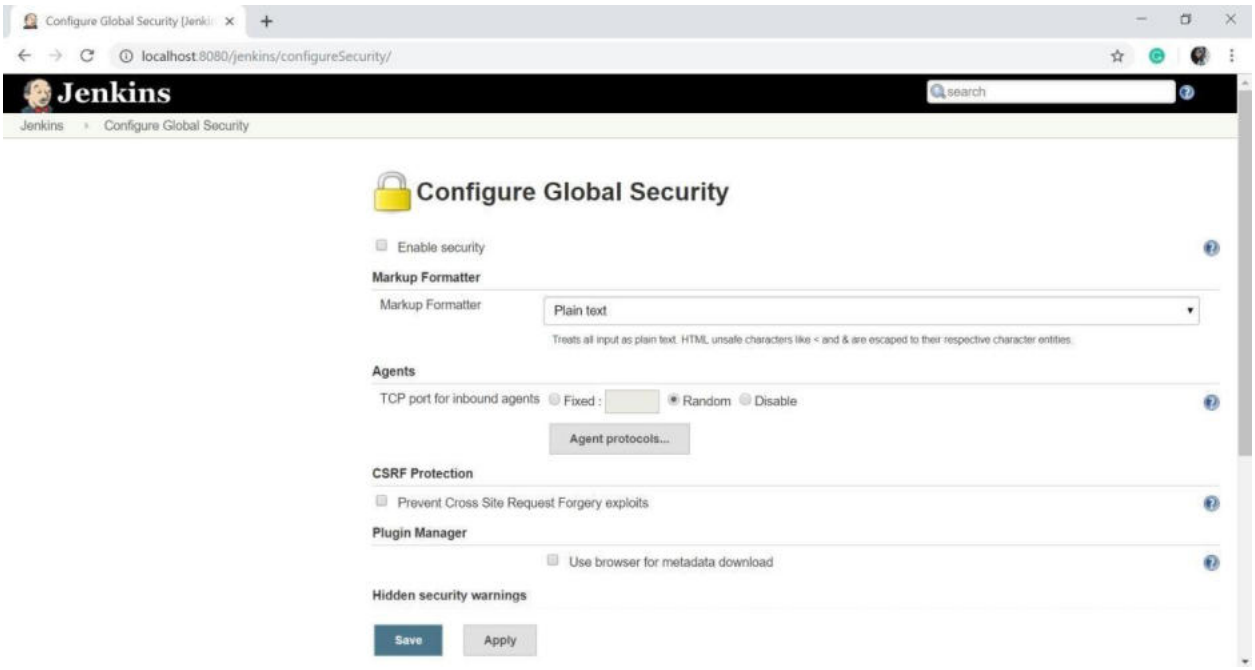
- Configure System
- Configure Global Security
- Configure Credentials
- Global Tool Configuration
- Reload Configuration from Disk
- Manage Plugins
- System Information
- System Log
- Load Statistics
- Jenkins CLI
- Script Console
- Manage Nodes
- About Jenkins
- Manage Old Data

Configure System

In this, we can manage paths to the various tools to use in builds, such as the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. Jenkins will add the required configuration fields dynamically when new plugins are installed.

Configure Global Security

Configure Global Security option provides the ability to set up users and their relevant permissions on the Jenkins instance. By default, you will not want everyone to be able to define builds or other administrative tasks in Jenkins. So Jenkins provides the ability to have a security configuration in place.



Jenkins stores all its system and job configuration details as XML files and all XML files are stored in the Jenkins home directory. Jenkins also stores all of the build histories.

If you are moving build jobs from one Jenkins instance to another or archiving old build jobs, you will have to insert or remove the corresponding build job directories to Jenkins's builds directory. You do not have to take Jenkins offline to do this?you can simply use the "Reload Configuration from Disk" option to reload the Jenkins system and build job configurations directly.

Mange plugins:

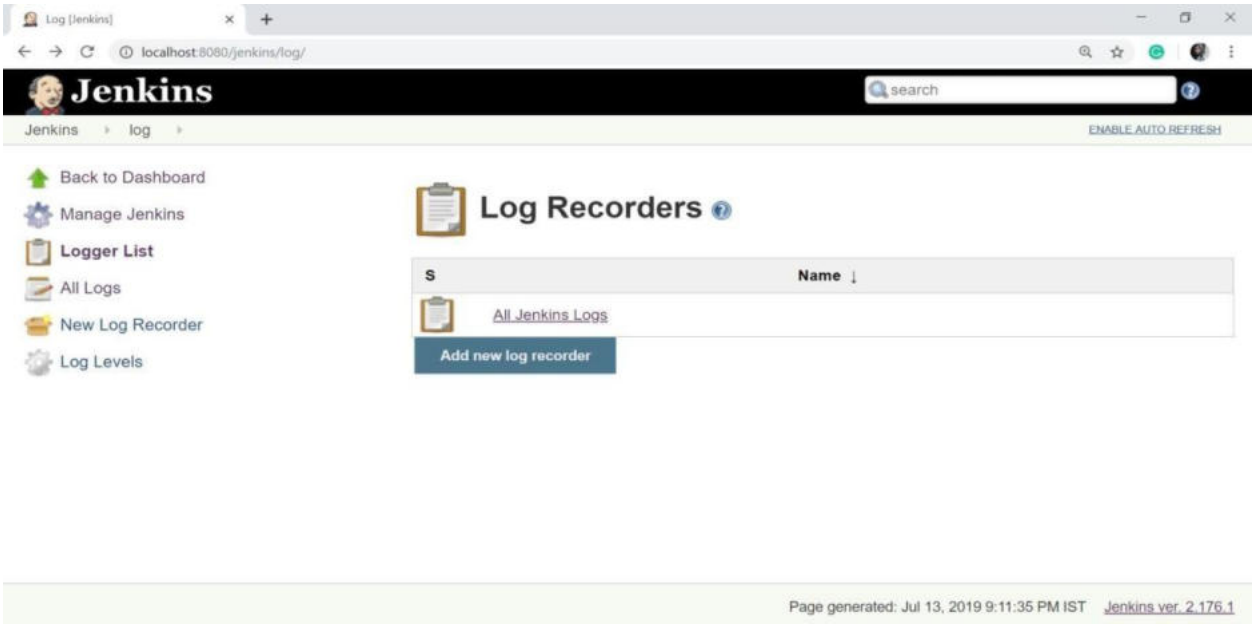
Here you can install a wide or different variety of third-party plugins from different Source code management tools such as Git, ClearCase or Mercurial, to code quality and code coverage metrics reporting. We can download, install, update, or remove the plugins from the Manage Plugins screen.

System Information

This option displays a list of all the current Java system properties and system environment variables. Here you can check what version of Java is currently running in, what user it is running under, and so forth.

System log

The System Log option is used to view the Jenkins log files in real time. As well as, the main use of this option is for troubleshooting.



Load Statistics

This option is used to see the graphical data on how busy the Jenkins instance is in terms of the number of concurrent builds and the length of the build queue which provides an idea of how long your builds need to wait before being executed. These statistics can show you a good idea of whether extra capacity or extra build nodes are required from an infrastructure perspective.

Jenkins CLI

Using this option, you can access various features in Jenkins through a command-line. To run the Jenkins through cli, first you have to download the Jenkins-cli.jar file and run it on your command prompt as follows:

Java -jar jenkins-cli.jar -s http://localhost:8080/jenkins/

This page gives the list of available commands with their description.

Script Console

This option allows you to run Groovy scripts on the server. This option is very useful for advanced troubleshooting since it requires a strong knowledge of the internal Jenkins architecture.

Manage nodes:

Jenkins can handle parallel and distributed builds. In this page, you can configure how many builds you want. Jenkins runs concurrently, and, if you are using distributed builds, set up builds nodes. A build node (slave) is another machine that Jenkins can use to execute its builds

About Jenkins

This option will show the version and license information of the Jenkins you are running. As well as it displays the list of all third party libraries.

Prepare for Shutdown

If you want to shut down Jenkins, or the server Jenkins is running on, it is best not to do so when a build is being executed. To shut down Jenkins cleanly, you can use the Prepare for Shutdown link, which prevents any new builds from being started. Eventually, when all of the current builds have completed, you will be able to shut down Jenkins cleanly.

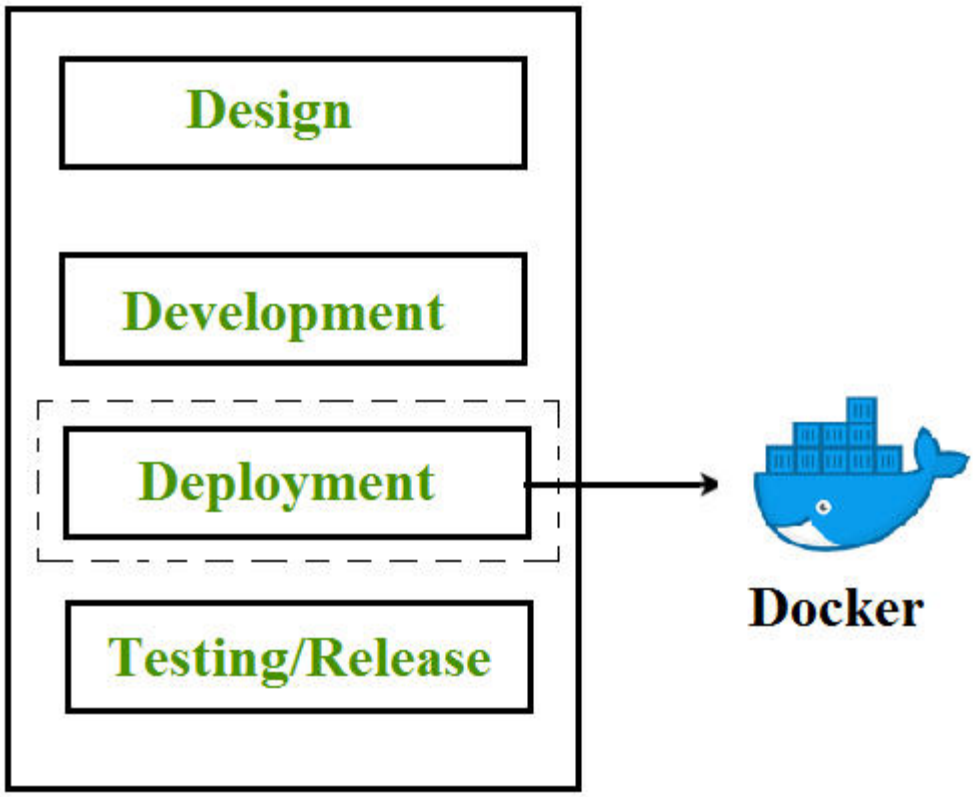
UNIT –V

DOCKER: Introduction to docker, Virtualization an Docker cli and Containerization differences, Docker Installation, Docker cli, Docker File, Docker Volumes, Docker-compose, Docker Network.

DOCKER: Introduction to docker

Docker is the containerization platform which is used to package your application and all its dependencies together in the form of containers so to make sure that your application works seamlessly in any environment which can be development or test or production. Docker is a tool designed to make it easier to create, deploy, run applications by using containers.

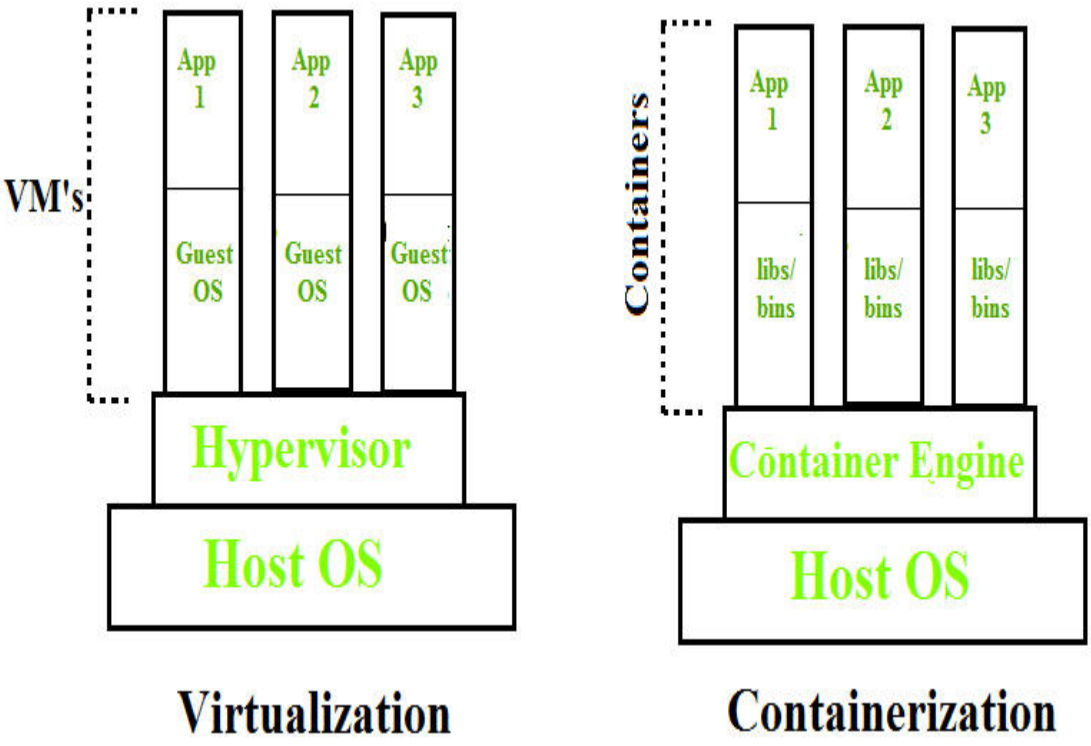
Docker is the world's leading software container platform. It was launched in 2013 by a company called Dot cloud, Inc which was later renamed as Docker, Inc. It is written in the Go language. It has been just six years since Docker was launched yet communities have already shifted to it from VM's. Docker is designed to benefit both developers and system administrators making it a part of many DevOps tool chains. Developers can write code without worrying about the testing and production environment. Sysadmins need not worry about infrastructure as Docker can easily scale up and scale down the number of systems. Docker comes into play at the deployment stage of the software development cycle.



Containerization

Containerization is OS-based virtualization which creates multiple virtual units in the user space, known as Containers. Containers share the same host kernel but are isolated from each other through private namespaces and resource control mechanisms at the OS level. Container-based Virtualization provides a different level of abstraction in terms of virtualization and isolation when compared with hypervisors. Hypervisors use a lot of hardware which results in overhead in terms of vitalizing hardware and virtual device drivers. A full operating-system (e.g-Linux, Windows) runon top of this virtualized hardware in each virtual machine instance.
But in contrast, containers implement isolation of processes at the operating system level, thus avoiding such overhead. These containers run on top of the same shared operating system kernel of the underlying host machine and one or more processes can be run within each container. In containers you don't have to pre-allocate any RAM, it is allocated dynamically during the creation of containers while in VM's you need to first pre-allocate the memory and then create the virtual machine. Containerization has better resource utilization compared to VMs and a short boot-up process. It is the next evolution in virtualization.

Containers are able to run virtually anywhere, greatly easy development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or bare metal, on a developer's machine or in data centers on-premises; and of course, in the public cloud. Containers virtualized CPU, memory, storage, and network resources at the OS-level, providing developers with a sandboxed view of the OS logically isolated from other applications. Docker is the most popular open-source container format available and is supported on Google Cloud Platform and by Google Kubernetes Engine.



In a software driven world where omnipresence and ease of deployment with minimum overheads are the major requirements, the cloud promptly takes its place in every picture. Containers are creating their mark in this vast expanse of cloud space with the world's top technology and IT establishments relying on the concept for their infrastructural necessities. Tech giants like Face book, Google and Microsoft use containers in their streamlined processes to facilitate a secure and easy deployment into the cloud production environments. This deployment with containers offers a technique, which abstracts the application from the run-time environment; much like virtual machines and this is done, of course by virtualization.

Virtualization is a technique, which essentially creates an illusion of a resource such as a desktop, storage, network or an operating system. Devices, applications and human users possess the capability of interacting with these resources. This illusion also called virtualization expands the capabilities of traditional systems, which are limited by their own physical resources.

Now, containers enable this virtualization for applications that are deployed in them. Applications in containers run independently, isolated from any physical resource. Containers virtualized the OS, CPU, memory, storage and network resources there by providing a controlled environment that can be scaled up or down as required. A container also packages the application along with its dependencies and necessary files, which enables the application to be deployed on any environment without having to configure the server, hardware or software

A virtual machine is a form of hardware virtualization. The hardware is logically separated from the other resources. The hardware can be any system such as a desktop (with hardware and its own OS) called the host machine, on which several virtual machines or guest machines can run, each with their own separate operating systems. This is made possible by a firmware called the hypervisor.

Benefits of containers

- Applications can be deployed without any worry about the run time environment. As a result, an application can easily be moved through the software development cycle and can run anywhere, for example, on Mac OS, Linux, and Windows and even in data centers. This result in less expenditure of time on examining the environment and more time can be time on developing new functionality.
- Multiple containers with applications can be run on the same instance of physical resource sharing an operating system. These containers being lightweight are fast and efficiently utilize the computing resources available.
- Containers are isolated from one another, which gives the developer the leeway to split application services into different containers. These containers do not share any dependencies and each can be manipulated and updated by the developer at will.

Why Docker

- Docker is designed to benefit both the Developer and System Administrator. There are the following reasons to use Docker -
- Docker allows us to easily install and run software without worrying about setup or dependencies.
- Developers use Docker to eliminate machine problems, i.e. "**but code is worked on my laptop**." when working on code together with co-workers.
- Operators use Docker to run and manage apps in isolated containers for better compute density

- Enterprises use Docker to securely built agile software delivery pipelines to ship new application features faster and more securely.
- Since docker is not only used for the deployment, but it is also a great platform for development, that's why we can efficiently increase our customer's satisfaction.

Docker

Docker is an **open-source centralized platform designed** to create, deploy, and run applications. Docker uses **container** on the host's operating system to run applications. It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application works in any environment like development, test, or production.

Docker includes components such as **Docker client, Docker server, Docker machine, Docker hub, Docker composes,**

Docker Containers

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

Virtual Machine

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

Docker Features

Although Docker provides lots of features, we are listing some major features which are given below.

- Easy and Faster Configuration
- Increase productivity
- Application Isolation
- Swarm
- Routing Mesh
- Services
- Security Management

Easy and Faster Configuration

This is a key feature of docker that helps us to configure the system easily and faster.

We can deploy our code in less time and effort. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

Increase productivity

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

Application Isolation

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

Swarm

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

Routing Mesh

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

Services

Services are a list of tasks that lets us specify the state of the container inside a cluster. Each task represents one instance of a container that should be running and Swarm schedules them across nodes.

Security Management

It allows us to save secrets into the swarm itself and then choose to give services access to certain secrets.

It includes some important commands to the engine like secret inspect, secret create etc.

Advantages of Docker

Below are the prominent advantages of Docker:

1. Faster time to market
2. Developer Productivity
3. Deployment velocity
4. IT infrastructure reduction
5. IT operational efficiency
6. Faster issue to resolution
7. Portability

1. Faster Time to Market

Business is changing rapidly. We need a tool that can meet the business requirement at that pace. Here Docker container helps to deploy a new version of the software with added features into production easily and with less human interaction as we can integrate Docker with CI/CD pipeline tools such as Jenkins. We can easily rollback the deployment if the newer version has any bug or something. We can easily implement the Canary test in Docker. In the Canary test, we roll out a newer version of the software into production for few users only and if everything looks good we slowly deploy the newer version of the software into the production for all

2. Developer Productivity

It increases developer productivity exponentially because earlier developers had to test their code in a test environment and when the operations team deploy the code to the production they got errors such as missing binaries, missing.net framework, etc. because both environments have different configurations and operations team just threw the code to the developers, saying code is not working and developers had to review the whole code and test it.

3. Deployment Velocity

Docker helps to increase deployment velocity from months to weeks. We can easily deploy new code to the production by integrating it to CI/CD.

4. IT Infrastructure Reduction

As discussed above, the containerized application uses less memory as compared to virtual machines. If we take an example of ISO image of Ubuntu it is about 4 GB in size however Ubuntu Docker image is about 60 MB in size that far lesser than ISO image. It is really lightweight and uses less resources to run hence reduce IT.

5. IT Operational Efficiency

We can use docker-compose to deploy a full-stack application that includes all our services that are required to start up a working application. It is also called micro services where application components are loosely coupled to each other. We can easily scale up or down any services as per our needs. Docker uses physical servers efficiently by distributing the load equally. Docker always checks for desired state configuration and matches the current configuration and if the desired configuration does not match the current configuration, Docker will automatically scale up or scale down the containers. It makes IT operations.

6. Faster Issue Resolution

As discussed earlier, Docker has a self-healing capability. It tries to restart the container if the container is not responding and it fails to do so it will just destroy the container and create a new one for us. So if there is an issue we can resolve it quickly to meet the business requirement. There is no much troubleshooting.

7. Portability

Docker images are portable as it encapsulates everything that is required to run an application. It means if it is running on a test environment, it will run on a production environment or any public cloud or any OS, just it needs Docker to be installed on the system on which we want to run.

Docker Engine

- It is a client server application that contains the following major components.
- A server which is a type of long-running program called a daemon process.
- The REST API is used to specify interfaces that programs can use to talk to the daemon and instruct it what to do.
- A command line interface client.

Docker Installation

- To install Docker in Amazon-Linux

Step1: Update the installed packages

- sudo yum update -y

Step2: Install Docker using yum

- sudo yum install docker -y

step3: Start the Docker service

- sudo service docker start

step4: Enable Docker to start on boot

- sudo systemctl enable docker

step5: Add your user to the docker group

- sudo usermod -a -G docker ec2-user

Docker architecture:

Docker uses client-server architecture to carry out all of its operations. The three major components that become an integral part of the Docker architecture are -

- The Docker Daemon (Server)
- REST API (Docker Engine)
- Docker CLI (Client)

These components work together to allow communication between the client and the server. You will see each of this one by one.

Docker Daemon

Daemon is a background process that runs persistently and is responsible for managing all the Docker objects - Images, Containers, Volumes, and Networks. It listens to the Docker API for instructions in the form of requests, processes them, and acts accordingly.

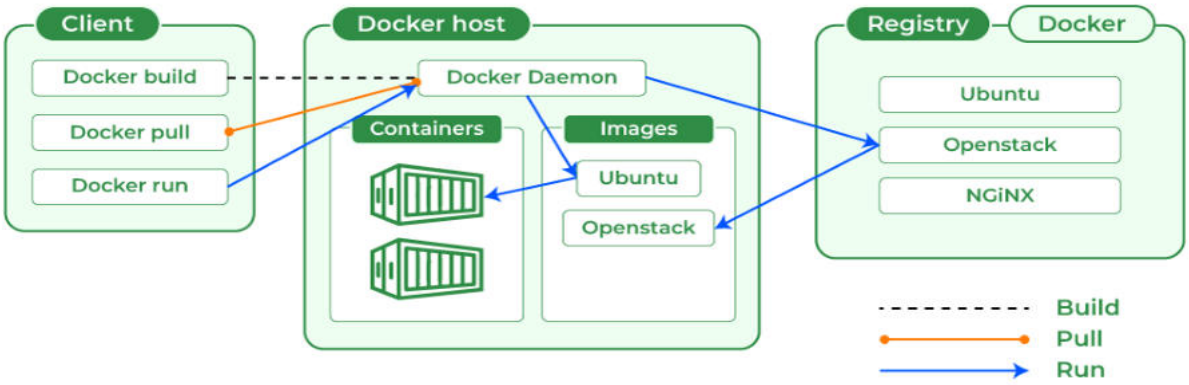
Docker REST API

The API acts as a middleman between the server and the client. The client application uses it to interact with the server (Daemon). The REST API is accessed only by the HTTP clients.**Docker Client**

Clients are used to interacting with the Docker daemon. It can be as simple as a Command Line Interface. You can directly talk to the server by executing simple commands inside the command line (client) to create and manage Docker objects.

These components work together to allow communication between the client and the server.

The Docker daemon does the heavy lifting of creating, running, and sharing Docker containers. This occurs when a Docker user executes commands in the command line. The interaction between the Docker daemon and the CLI is possible through the Docker engine called the REST API, over a network interface or UNIX sockets. Both the client and the daemon run on the same machine. However, you can also connect to a daemon on a remote machine.



The four key components that make up the entire Docker architecture are -

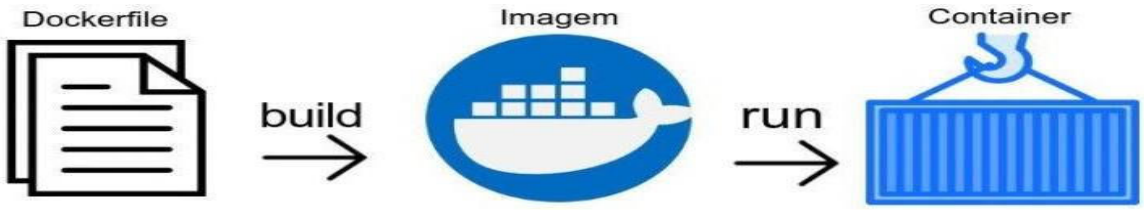
1. The Docker Daemon or the server
2. The Docker Command Line Interface or the client
3. Docker Registries
4. Docker Objects -
    1. Images
    2. Containers
    3. Network
    4. Storage

Docker Containers - Docker containers are isolated, encapsulated, packaged, and secured application environments that contain all the packages, libraries, and dependencies required to run an application. For example, if you create a container associated with the Ubuntu image, you will have access to an isolated Ubuntu environment. You can also access the bash of this Ubuntu environment and execute commands.Containers have all the access to the resources that you define while using the Dockerfile while creating an image. Such configurations include build context, network connections, storage, CPU, memory, ports, etc. For example, if you want access to a container with libraries of Java installed, you can use the Java image from the Dockerhub and run a container associated with this image using the Docker run command.

You can also create containers associated with the custom images that you create for your application using the Dockerfiles. Containers are very light and can be spun within a matter of Network- You can create a secured channel so that all the isolated containers in a cluster can communicate and share data or information. You can use several network drivers and plug-in to achieve this. Docker networks become the base of communication in any Docker network cluster.

DOCKER DOCKERFILE:

A Dockerfile is a text file that contains instructions for building a Docker image. An image is a read-only template that includes all the files, dependencies, and configuration needed to run a container. A Dockerfile provides a way to automate the process of building an image, ensuring that the resulting image is consistent and reproducible



KEYWORDS:

FROM: The FROM instruction specifies the base image to use for the Docker container. This is the starting point for building the image, and all subsequent instructions will build on top of it.

Syntax:

FROM<image-name>

Example:**FROM ubuntu:latest**

RUN: The RUN instruction executes a command in the Docker container during the build process. This can be used to install packages, update software, or perform other tasks necessary for your application.

Syntax:

```
RUN<command>
```

Example:

MAINTAINER: In a Dockerfile, the MAINTAINER keyword is used to specify the name andemail address of the person or organization that is responsible for maintaining the Dockerimage

Syntax:

```
MAINTAINER<name>[<emai
l>]
```

Example:

```
MAINTAINERkrishna<krishna04.b@gmail.com>
```

CMD: The CMD instruction specifies the command that should be executed when the Container starts running. This is typically used to start the main process in the container, such as a web server or application.

Syntax:

```
CMD["executable","param1","param2",...]
```

Example:

ENTRYPOINT: The ENTRYPOINT instruction specifies the command that should be executed when the container starts running, just like CMD. However, the difference is thatthecommandspecifiedinENTRYPOINTcannotbeoverriddenbythecommandlinearguments passed when starting the container.

Syntax:

```
ENTRYPOINT["executable","param1","param2",...]
```

Example:

```
ENTRYPOINT["python"]
```

ADD/COPY: The ADD and COPY instructions are used to add files from the host system to the Docker image. COPY is preferred overADD,as itis simpler and less powerful.

Syntax:

COPY<src><dest>

Example:

COPYapp.py/app/

EXPOSE: The EXPOSE instruction informs Docker that the container will listen on the Specified network ports at runtime. This does not actually publish the port, but simply documents the intended usage.

Syntax:

EXPOSE<port>

Example:

CREATING DOCKERFILE:

Step 1 − Create a file called **Docker File** and edit it using **vim**. Please note that the name of the file has to be "Dockerfile" with "D" as capilatal letter

Step 2 − Build your Docker File using the following instructions.

```
#This is a sample Image
FROM ubuntu
MAINTAINER “krishna”
RUN apt-get update
RUN apt-get install –y nginx
CMD [“echo”,”Image created”]
```

The following points need to be noted about the above file −

- The first line "#This is a sample Image" is a comment. You can add comments to the Docker File with the help of the **#** command
- The next line has to start with the **FROM** keyword. It tells docker, from which base image you want to base your image from. In our example, we are creating an image from the **ubuntu** image.
- The next command is the person who is going to maintain this image. Here you specify the **MAINTAINER** keyword and just mention the email ID.
- The **RUN** command is used to run instructions against the image. In our case, we first update our Ubuntu system and then install the nginx server on our **ubuntu** image.
- The last command is used to display a message to the user.

Step 3 − Save the file

The Docker File can be built with the following command −

docker build

docker build

This method allows the users to build their own Docker images.

Syntax

docker build -t ImageName

Options

- **-t** − is to mention a tag to the image
- **ImageName** − This is the name you want to give to your image.
- **TagName** − This is the tag you want to give to your image.
- **Dir** − The directory where the Docker File is present.

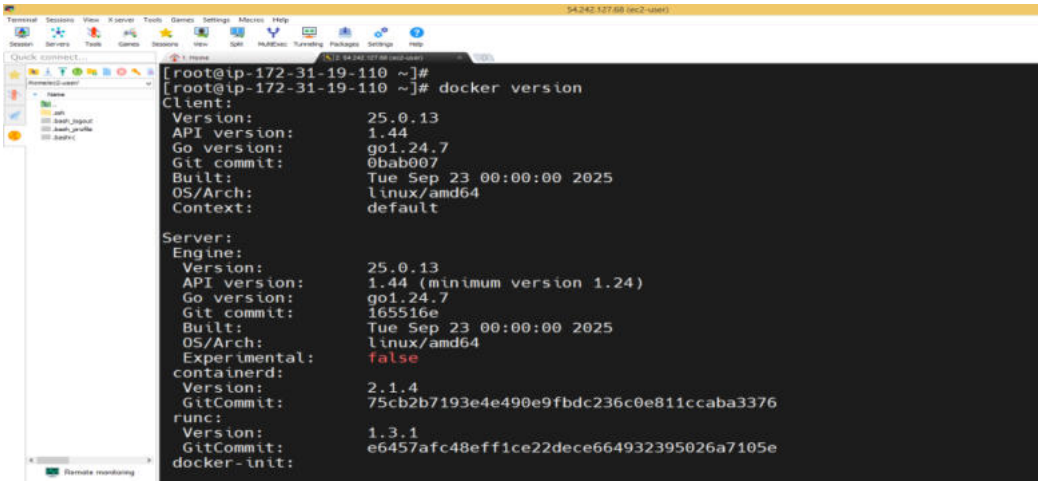
DOCKERCLI

Docker implies OS-level virtualization. Most developer prefers using docker and operating systems are tightly coupled with developers. Optimizing on the platform's functionality kicks with docker commands mastery. They are very much lightweight VM .The following are the basic commands in Docker

- **docker --version**
- **Docker version**
- **docker pull**
- **docker run**
- **docker ps**
- **docker ps -a**
- **docker exec**
- **docker stop**
- **docker kill**
- **docker commit**
- **docker login**
- **docker push**
- **docker images**
- **docker rm**
- **docker rmi**
- **docker build**

Docker –version

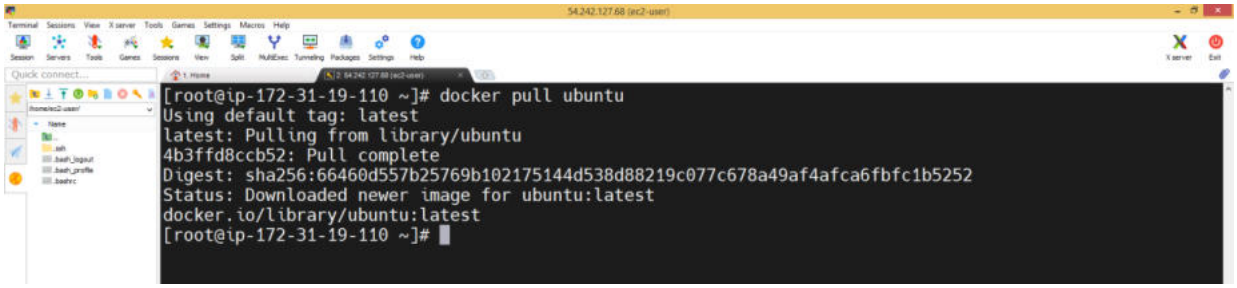
This command is used to get the currently installed version of docker



-ddocker pull

Usage: docker pull <image name>

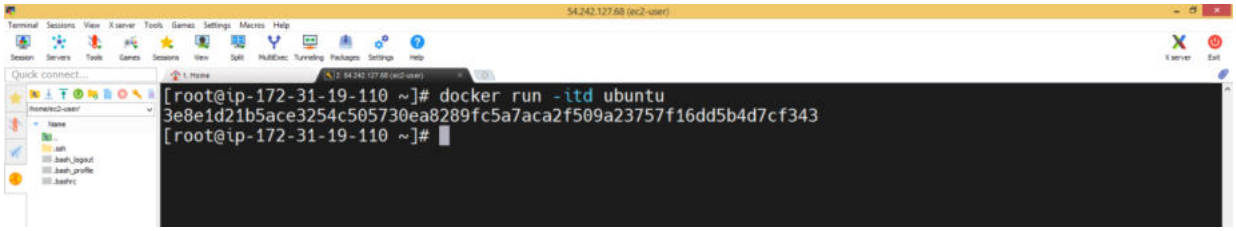
This command is used to pull images from the **docker repository**(hub.docker.com)



docker run

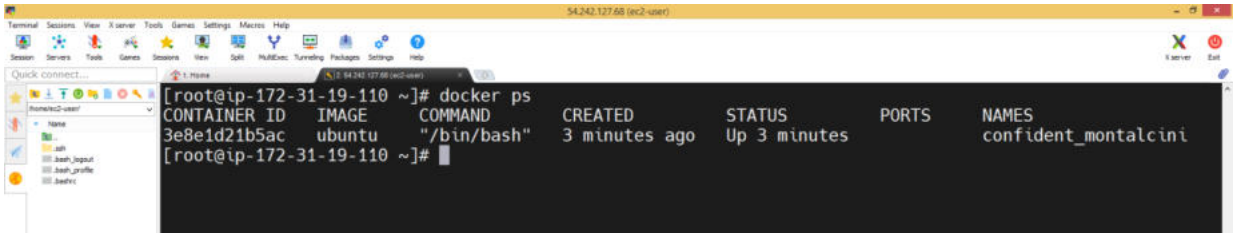
Usage: docker run -it -d <image name>

This command is used to create a container from an image



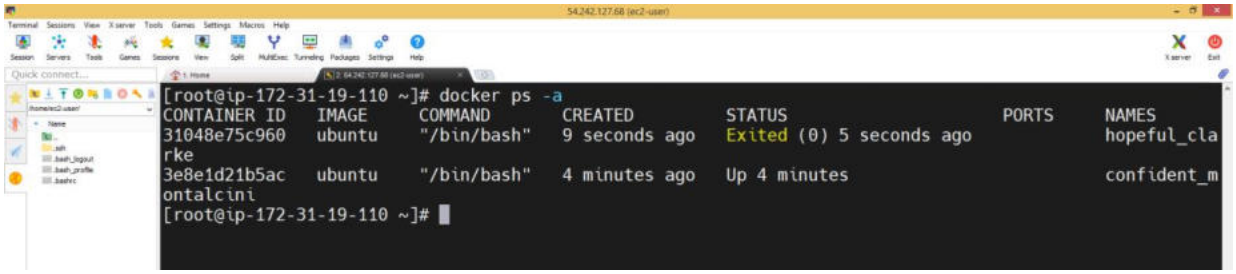
4. **dockerps**

This command is used to list the running containers



5. **dockerps -a**

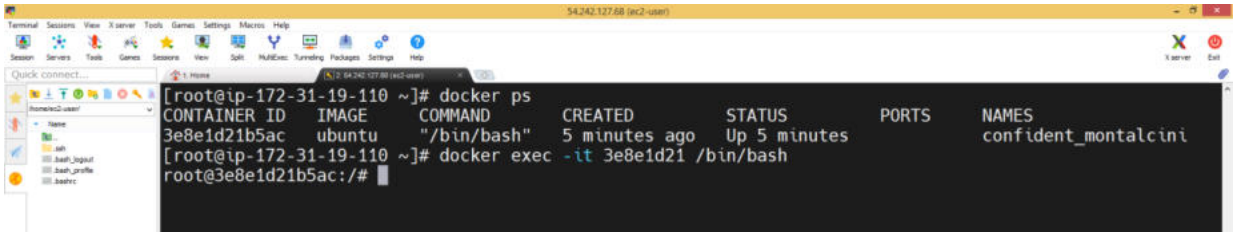
This command is used to show all the running and exited containers



6. **docker exec**

Usage: docker exec -it <container id> bash

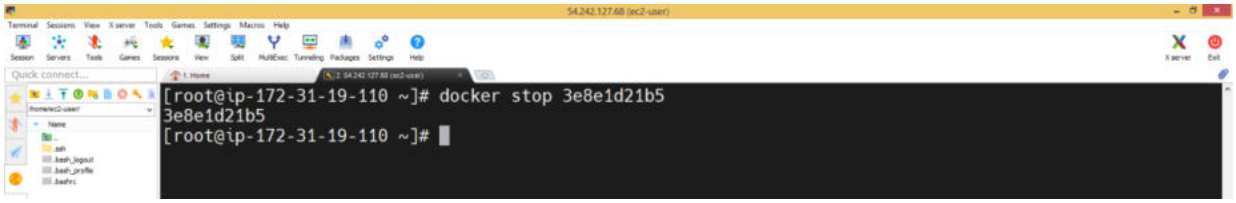
This command is used to access the running container



7. **docker stop**

Usage: docker stop <container id>

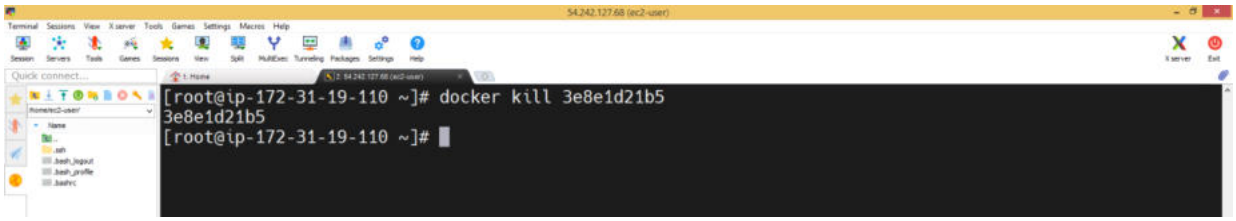
This command stops a running container



8. **docker kill**

Usage: docker kill <container id>

This command kills the container by stopping its execution immediately. The difference between 'docker kill' and 'docker stop' is that 'docker stop' gives the container time to shutdown gracefully, in situations when it is taking too much time for getting the container to stop, one can opt to kill it



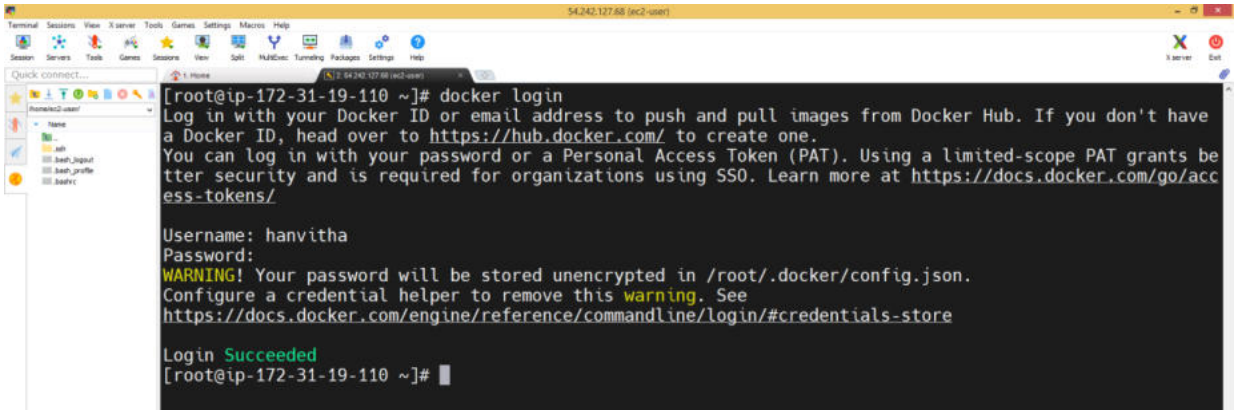
9. **docker commit**

Usage: docker commit <conatainer id><username/imagename>

This command creates a new image of an edited container on the local system

10. **docker login**

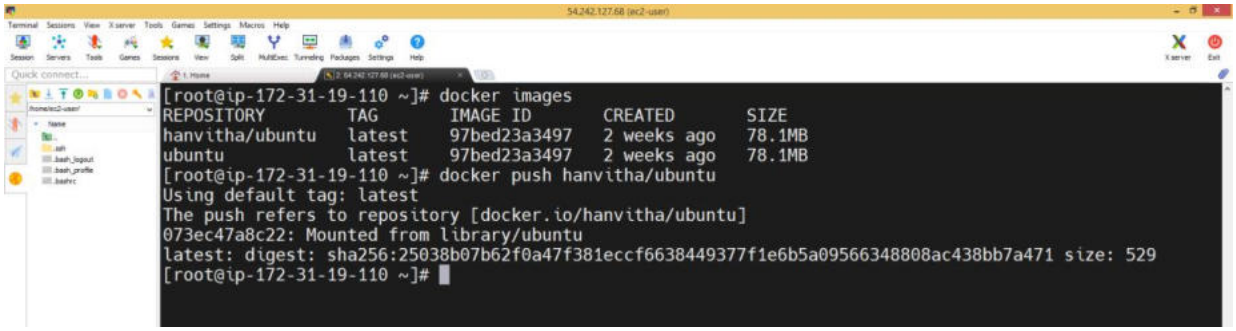
This command is used to login to the docker hub repository



11. **docker push**

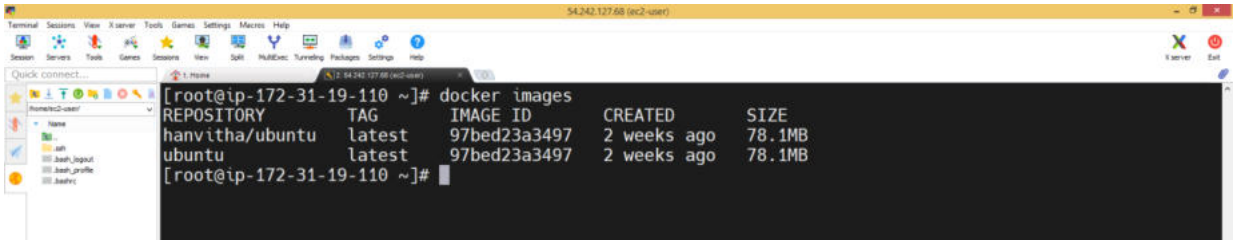
Usage: docker push <username/image name>

This command is used to push an image to the docker hub repository



12. **docker images**

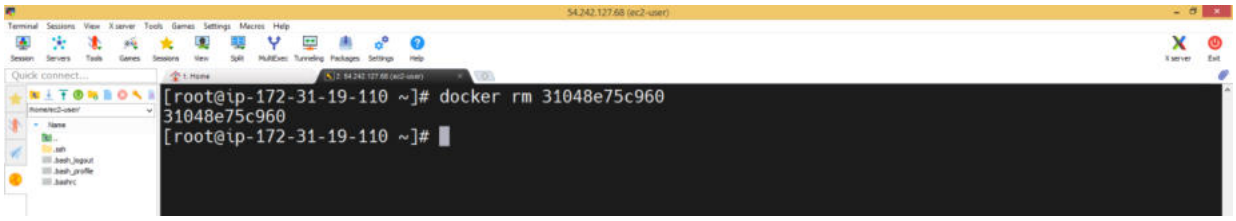
This command lists all the locally stored docker images



13. **dockerrm**

Usage: dockerrm<container id>

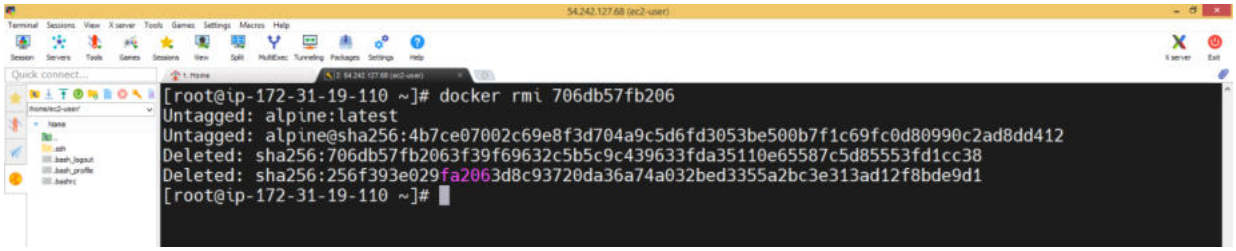
This command is used to delete a stopped container



14. **dockerrmi**

Usage: dockerrmi<image-id>

This command is used to delete an image from local storage



15. **docker build**

Usage: docker build <path to docker file>

This command is used to build an image from a specified docker file

Docker Compose

Docker Compose is a tool used for defining and running multi-container Docker applications. It allows developers to create a YAML file that defines the services, networks, and volumes that make up their application, and then use a single command to start and stop all of these containers together.

The Docker Compose file defines the various services that make up the application, including the Docker images to use, the ports to expose, and any environment variables or other configuration settings that need to be set. Developers can also define volumes and networks to share data between containers, and configure logging and other monitoring features.

Once the Docker Compose file is defined, developers can use the docker-compose up command to start all of the containers defined in the file, or docker-compose down to stop and remove them. Developers can also use other commands to manage individual containers, scale services up or down, or view logs and other diagnostic information.

Networks: A network is used to enable communication between containers. Docker Compose allows you to define custom networks for your application to use.

Volumes: A volume is a shared directory that can be used by multiple containers. Docker Compose allows you to define custom volumes for your application.

Environment variables: Docker Compose allows you to set environment variables for your application, which can be used to configure containers at runtime.

Dependencies: Docker Compose allows you to define dependencies between services, so that one service can be started only after another service has started.

docker-compose up: This command starts the application defined in the docker-compose.yml file. By default, it starts the application in the foreground and displays the logs in the console. Use the -d option to run the application in the background.

docker-compose down: This command stops and removes the containers, networks, and volumes created by the docker-compose.yml file.

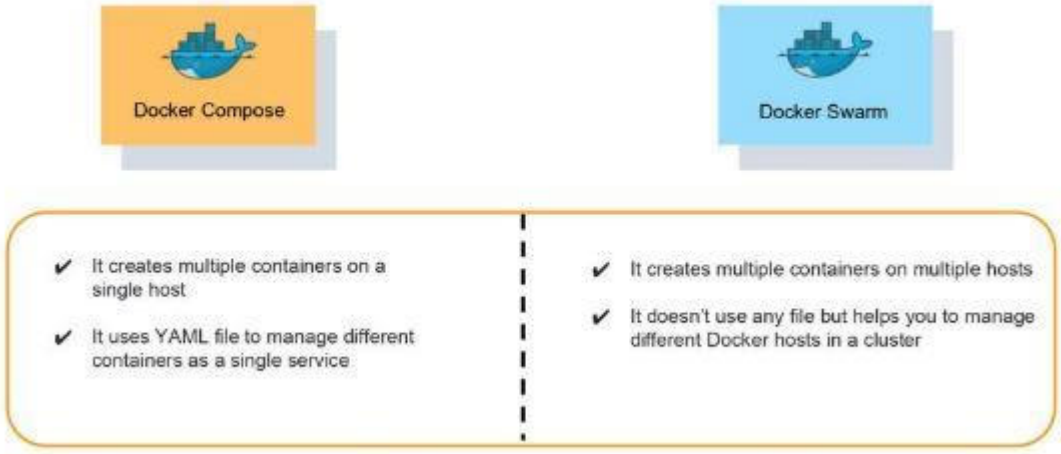
docker-compose ps: This command lists the running containers for the application.

docker-compose logs: This command displays the logs of the containers in the application. Use the -f option to tail the logs.

Benefits of docker compose :

- **Single host deployment** - This means you can run everything on a single piece of hardware
- **Quick and easy configuration** - Due to YAML scripts
- **High productivity** - Docker Compose reduces the time it takes to perform tasks
- **Security** - All the containers are isolated from each other, reducing the threat landscape

Now, you might be thinking that Docker Compose is quite similar to Docker Swarm, but that's not the case. Here are some of the differences between Docker Compose and Docker Swarm



It provides the following commands for managing the whole lifecycle of our application.

- Start all services: Docker Compose up
- Stop all services: Docker Compose down
- Install Docker Compose using pip: pip install -U Docker-compose
- Check the version of Docker Compose: Docker-compose-v
- Run Docker Compose file: Docker-compose up -d
- List the entire process: Docker ps
- Scale a service - Docker Compose up -d -scale
- Use YAML files to configure application services - Docker Compose.yml

To implement compose, it consists the following steps

1. Put Application environment variables inside the Dockerfile to access publicly.
2. Provide services name in the docker-compose.yml file so they can be run together in an isolated environment.
3. run docker-compose up and Compose will start and run your entire app.

A typical **docker-compose.yml** file has the following format and arguments.

1 .Installing Docker Compose

vim app.py

```
from flask import Flask

app = Flask(__name__)
count = 0

@app.route("/")
def hello():
    global count
    count += 1
    return f"Hello, krishna! This page has been visited {count} times."

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

vim docker-compose.yml

```
version: '3'

services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/app
```

vim Dockerfile

```
FROM python:3.9-slim-buster

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install the required packages
RUN pip install --no-cache-dir -r requirements.txt

# Set the command to run when the container starts
CMD ["python", "app.py"]
```

vim requirements.txt

```
flask
```

Now, we can see the output by following the running http url.

Docker Networking

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

Bridge - Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.

Host - It is used when we don't need for network isolation between the container and the host.

None - It disables all the networking.

Overlay - Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.

Macvlan - Macvlan is used when we want to assign MAC addresses to the containers.

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

Data Volume - Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.

Directory Mounts - It is one of the best options for docker storage. It mounts a host's directory into a container.

Storage Plugins - It provides an ability to connect to external storage platform

- **Bridge Driver** - Bridge network driver is mostly used when you have a multi-container application running in the same host machine. This is the default network driver.
- **Host Driver** - If you don’t require any type of network isolation between the Docker host machine and the containers on the network, you can use the Host driver.
- **Overlay Driver** - When you use Docker swarm mode to run containers on different hosts on the same network, you can use the overlay network driver. It allows different swarm services hosting different components of multi-container applications to communicate with each other.
- **Macvlan** - The macvlan driver assigns mac addresses to each container in the network. Due to this, each container can act as a standalone physical host.