**Version Control System**

**Version Control System** is software that helps software developers to work together and maintain a complete history of their work.

* Allows developers to work simultaneously.
* Does not allow overwriting each other’s changes.
* Maintains a history of every version.

**Types of Version Control Systems**

* Centralized version control system (CVCS).
* Distributed/Decentralized version control system (DVCS).

**Centralized version control system (CVCS)**

Central server to store all files and enables team collaboration. But the major drawback of CVCS is its single point of failure, i.e., failure of the central server.

**Distributed Version Control System**

DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the sever goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository.

Git does not rely on the central server and that is why you can perform many operations when you are offline.

**Advantages of Git**

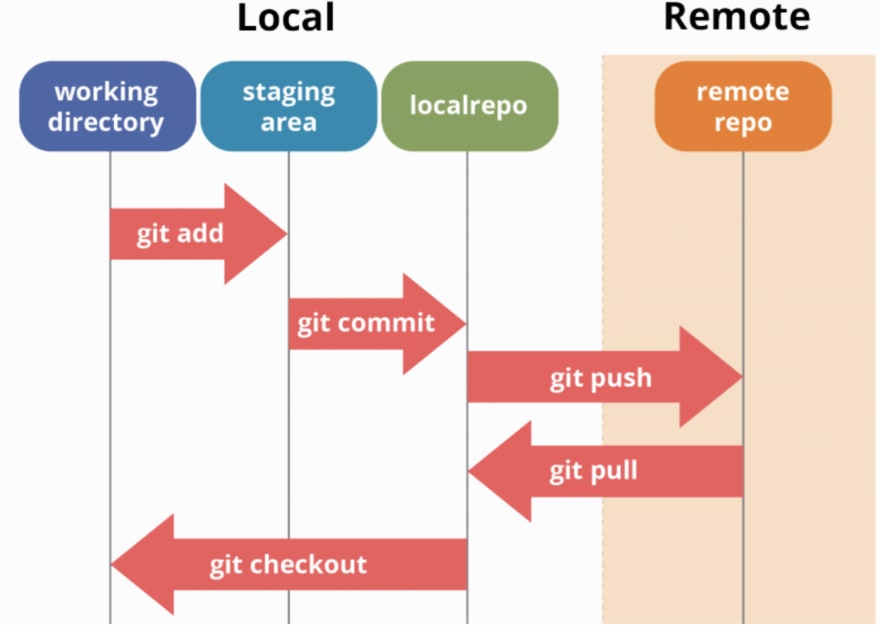
* Free and Open source.
* Fast and small
* Implicit backup
* Security
* No need of powerful hardware
* Easier branching

**Local Repository**

Every VCS tool provides a private workplace as a working copy. Developers make changes in their private workplace and after commit, these changes become a part of the repository. Git takes it one step further by providing them a private copy of the whole repository. Users can perform many operations with this repository such as add file, remove file, rename file, move file, commit changes, and many more.

**Work Flow of Git**

1. You modify a file from the working directory.
2. You add these files to the staging area.
3. You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.



**Git Installation**

Please refer below link for installing git

<https://phoenixnap.com/kb/how-to-install-git-windows#:~:text=Steps%20For%20Installing%20Git%20for%20Windows%201%20Extract,which%20extra%20option%20you%20would%20like%20to%20enable>.

**Setting up a repository**

**git init / git clone / git config**

**git init**

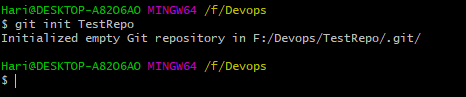
The git init command initializes a new, empty repository.



Executing git init creates a .git subdirectory in the current working directory, which contains all of the necessary Git metadata for the new repository. This metadata includes subdirectories for objects, refs, and template files. A HEAD file is also created which points to the currently checked out commit.

Compared to SVN, the git init command is an incredibly easy way to create new version-controlled projects. Git doesn’t require you to create a repository, import files, and check out a working copy. Additionally, Git does not require any pre-existing server or admin privileges. All you have to do is cd into your project subdirectory and run git init, and you'll have a fully functional Git repository.

git init <directory>



**Git Files Directory**

How the directory structure is organized and where the components of the Git tree fall in to it.

* The .**git** directory holds the meat of your local repo. It holds subdirectories for objects, refs, config, index, HEAD, and more.
* The **objects** folder is where the three objects: commit, tree and blob are housed. Each object gets its own sub folder.
  + **Blob**

Blobs, on the other hand, are just contents — binary streams of data, every blob in git are identified by its SHA-1 hash. SHA-1 hashes consist of 20 bytes

* + **Tree**

A tree is a simple object that has a bunch of pointers to blobs and other trees - it generally represents the contents of a directory or subdirectory.

* + **Commit**

Commit set a message about the changes you were done. The commit also saves a revision of the code and you can revert the code to any version anytime.

* The **refs** folder holds references to all of the objects in the directory. Let's say you have some orphaned objects on branches that no longer exist. You can go into this folder and prune them off of your branch.
* The **config** folder contains all of your configurations including the credentials you use to connect to GitHub; your email and name that you used to configure your global config.
* The **HEAD** file is what contains the pointer or reference to your working directory or branch and its latest commit.
* The **index** file is for when you use git add to stage your files for a commit. They leave the index file after the commit.

**git add**

The **git add** command adds a file to the Git staging area. This area contains a list of all the files you have recently changed.



**git commit**

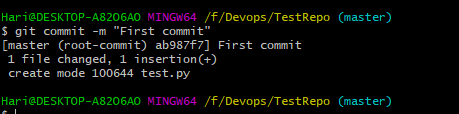
The **git commit** the staged snapshot.

**git commit –a**

Commita snapshot of all changes in the working directory.

**git commit –m**

A shortcut command that immediately creates a commit with a passed commit message.



**git commit -am "commit message"**

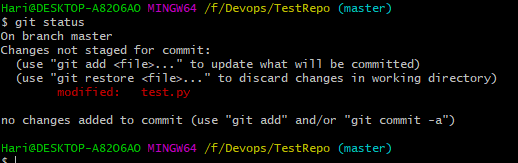
A power user shortcut command that combines the -a and -m options. This combination immediately creates a commit of all the staged changes and takes an inline commit message.

**git commit –amend**

This option adds another level of functionality to the commit command. Passing this option will modify the last commit. Instead of creating a new commit, staged changes will be added to the previous commit.

**git status**

The git status command displays the state of the working directory and the staging area.



**git log**

The git log command displays committed snapshots.

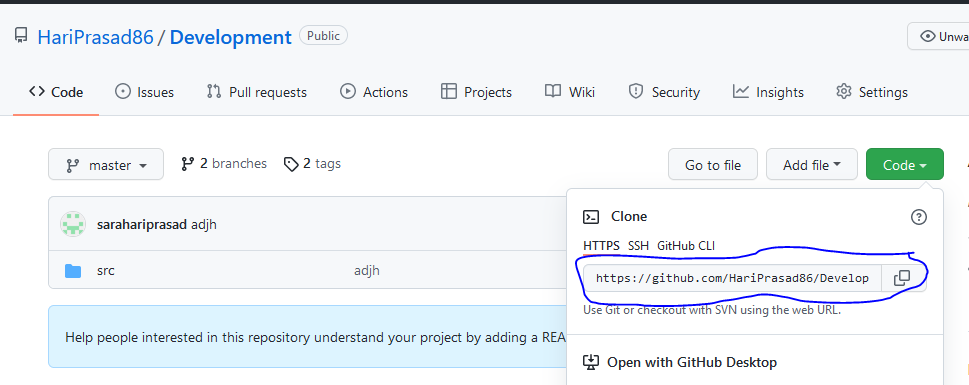


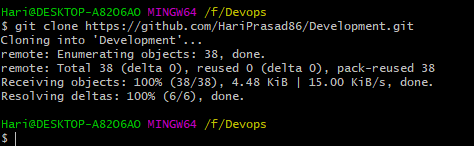
**Git clone**

The **git clone** is a command-line utility which is used to make a local copy of a remote repository.

git clone <repository URL>

Copy the git remote repository url like below screen

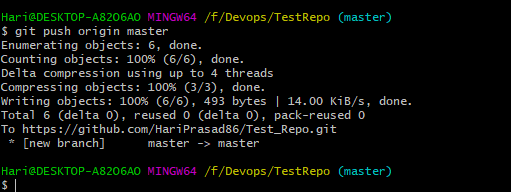




**Git push**

The git push command is used to upload local repository content to a remote repository.





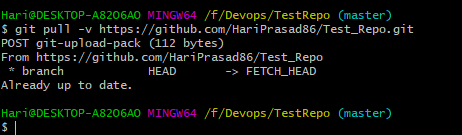
**Git pull**

The pull command is used to access the changes (commits)from a remote repository to the local repository.

$ **git pull <option> [<repository URL><refspec>...]**

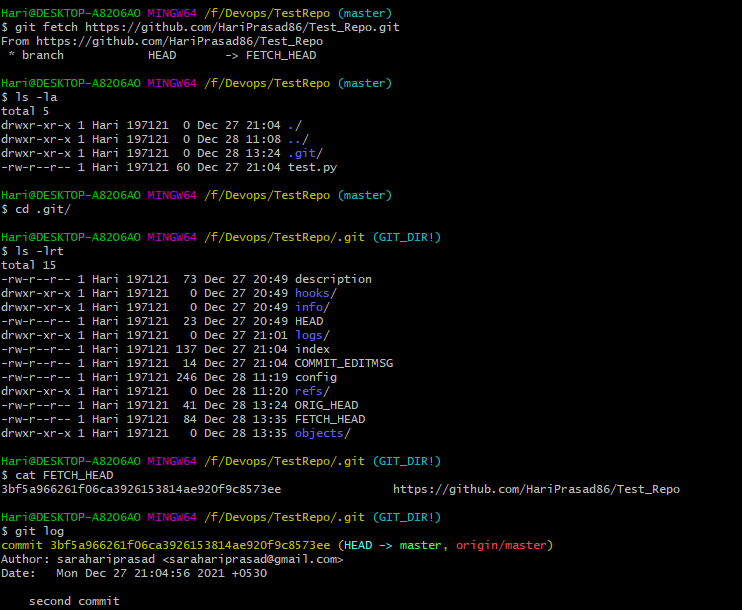
**<option>:** Options are the commands; these commands are used as an additional option in a particular command. Options can be -q (quiet), -v (verbose), -e(edit) and more.

**<repository URL>:** Repository URL is your remote repository's URL where you have stored your original repositories like GitHub or any other git service



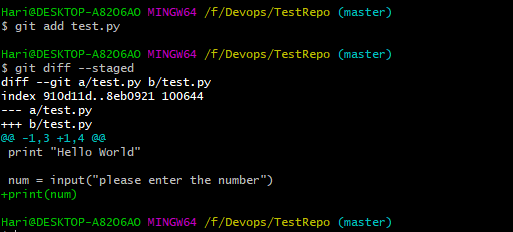
**Git fetch**

The git fetch command is used to download commits, files and references from a remote repository into the local repository. It is used to see what other members of the team have been working on.



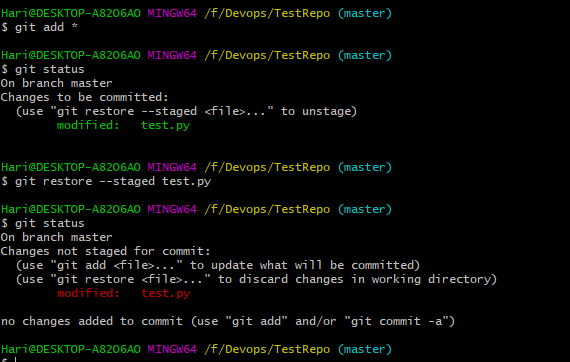
**Git diff**

The above command will display the changes of already staged files. Consider the below output:

****

**git restore**

The "**restore**" command helps to unstage or even discard uncommitted local changes.

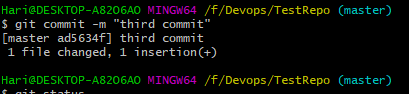


**Git revert**

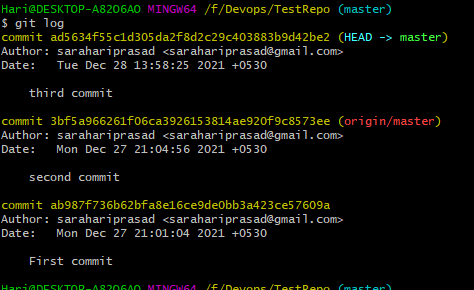
The git revert command can be considered an 'undo' type command, however, it is not a traditional undo operation. Instead of removing the commit from the project history.

Please find the below output screens.

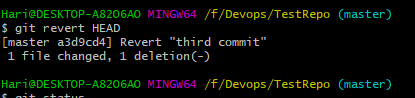
Step1. Commited new changes to local repository.



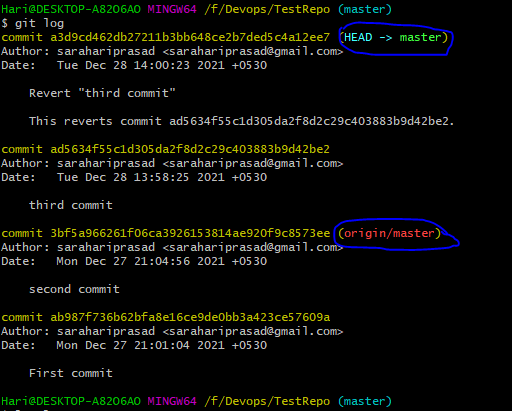
Step2. Check git log and find the latest commit.



Step 3. Reverting commit changes.



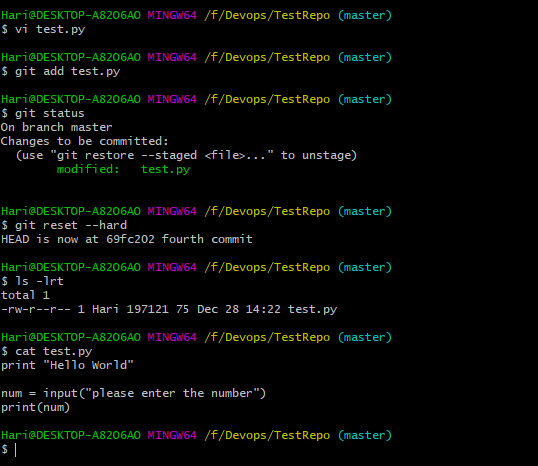
Step4. Check the git log again.



Finally we are at version c8573ee

**Git reset**

The **git reset** command is a complex and versatile tool for undoing changes. It has three primary forms of invocation. These forms correspond to command line arguments --soft, --mixed, --hard. The three arguments each correspond to Git's three internal state management mechanism's, The Commit Tree (HEAD), The Staging Index, and The Working Directory.

****

**BRANCHING STRATEGIES**

Git branching strategies allow a code base to evolve organically in a coherent way. A branching strategy is a convention, or a set of rules, that describes when branches are created, naming guidelines for branches, what use branches should have, and so on

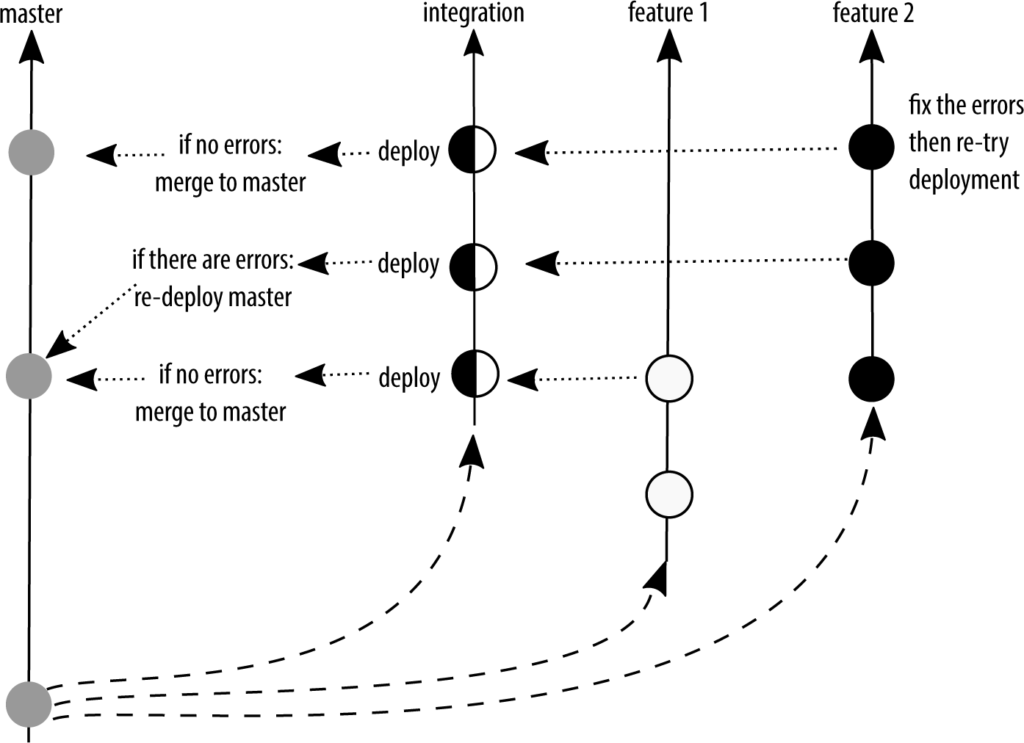
**Master branch** – The branch from which software is released to production. Only production ready code can be committed to this branch. All commits are tagged, since they represent releases.

**Develop/Dev branch** – The main branch that developers work on.

**Feature branches** – Feature branches are used to develop new features. Feature branches will be eventually merged back into the development branch. Each feature branch is as small as possible.

**Release branches** – Once a set of features & their associated bug fixes have been implemented and merged into the develop branch a release branch is created. The branch is assigned a name composed of the release prefix, followed by the numeric form of choice for your release.

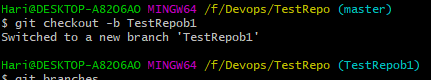
**Hotfix/Patch branches** – Hotfixes are production issues that need an immediate fix before a planned release. The development team creates a hotfix branch from master and applies the appropriate fixes.



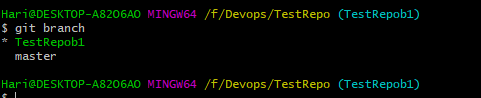
**Create a branch:**

Branch we can create local or remote please follow the below.

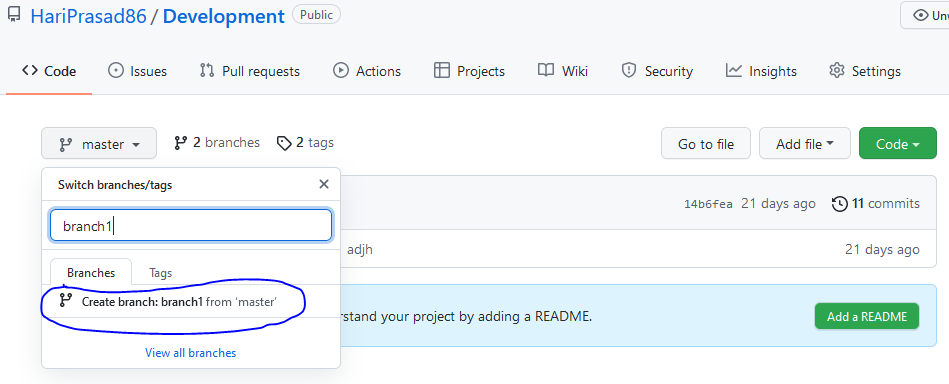
Below is the output for new branch in local repo.

****

**Command to check available branches**

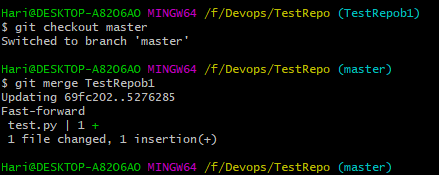
****

**Other way to create branch in github please follow below screen.**

****

**Git merge**

In Git, the merging is a procedure to connect the forked history. It joins two or more development history together.

****

**Build Tools**

**What is a Build Tool?**

A build tool is a tool that automates everything related to building the software project. Building a software project typically includes one or more of these activities:

Generating source code (if auto-generated code is used in the project).

Generating documentation from the source code.

Compiling source code.

Packaging compiled code into JAR files or ZIP files.

Installing the packaged code on a server, in a repository or somewhere else.

**Installing Maven**

Download Maven latest Maven software from [Download latest version of Maven](https://maven.apache.org/download.cgi)

For example: **apache-maven-3.1.1-bin.zip**

Extract package

Add Maven Path in environment variable

**Maven Overview**

Maven is centered around the concept of POM files (Project Object Model). A POM file is an XML representation of project resources like source code, test code, dependencies (external JARs used) etc. The POM contains references to all of these resources. The POM file should be located in the root directory of the project it belongs to.

**Maven Repository**

A maven repository is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

Local Repository

Central Repository

Remote Repository

**Maven Local Repository**

Maven local repository is located in your local system. It is created by the maven when you run any maven command.

By default, maven local repository is %USER\_HOME%/.m2 directory. For example: C:\Users\Hari\.m2.

**Maven Central Repository**

Maven central repository is located on the web. It has been created by the apache maven community itself.

The path of central repository is: <http://repo1.maven.org/maven2/>

**Maven Remote Repository**

Maven remote repository is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

**Maven pom.xml file**

POM is an acronym for Project Object Model. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

Maven reads the pom.xml file, then executes the goal.

**Elements of maven pom.xml file**

|  |
| --- |
|  |
| **project** | It is the root element of pom.xml file. |
| **modelVersion** | It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0. |
| **groupId** | It is the sub element of project. It specifies the id for the project group. |
| **artifactId** | It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs. |
| **version** | It is the sub element of project. It specifies the version of the artifact under given group. |

**Example of pom.xml file below**

<project xmlns="http://maven.apache.org/POM/4.0.0"   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.javatpoint.application1</groupId>

  <artifactId>my-app</artifactId>

  <version>1</version>

 </project>

**Below command to generate Maven project**

**syntext**

**mvn archetype:generate -DgroupId={groupID} -DartifactId={artifactID} -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

**example below**

**mvn archetype:generate -DgroupId=devops -DartifactId=devopsProject -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

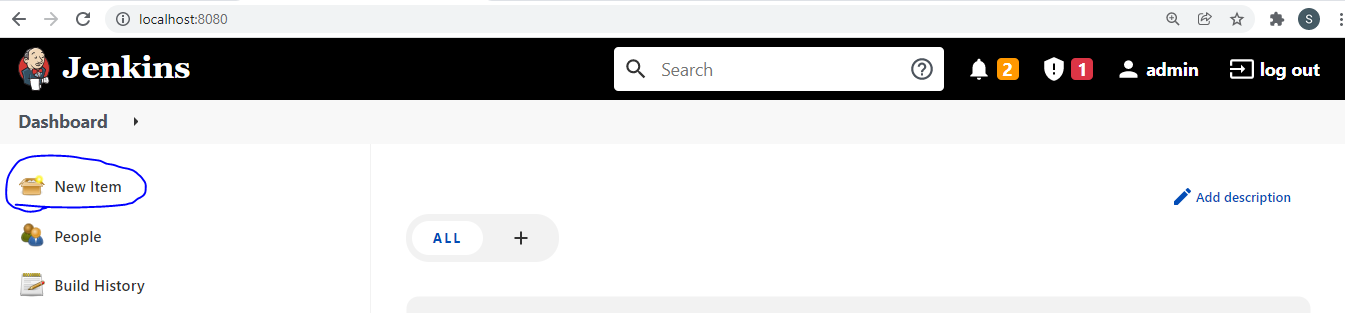
**command for web application and creating war package**

**mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWeb -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false**

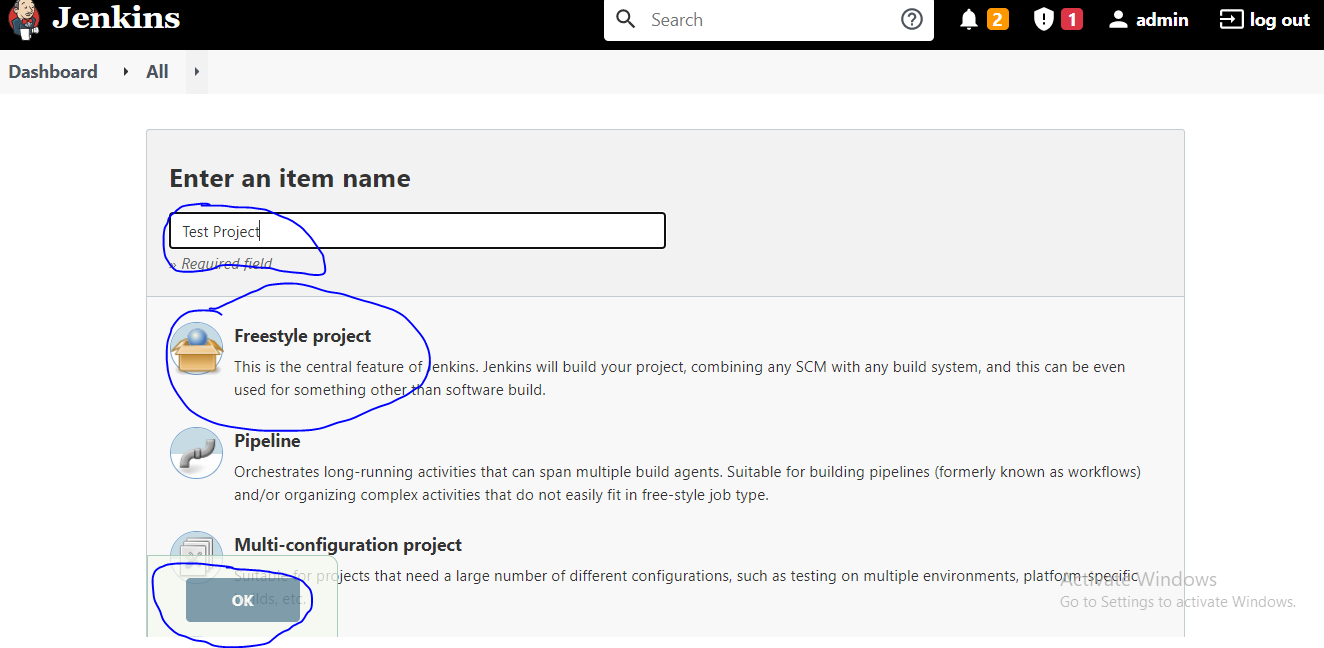
**Jenkins**

**Jenkins Freestyle configuration**

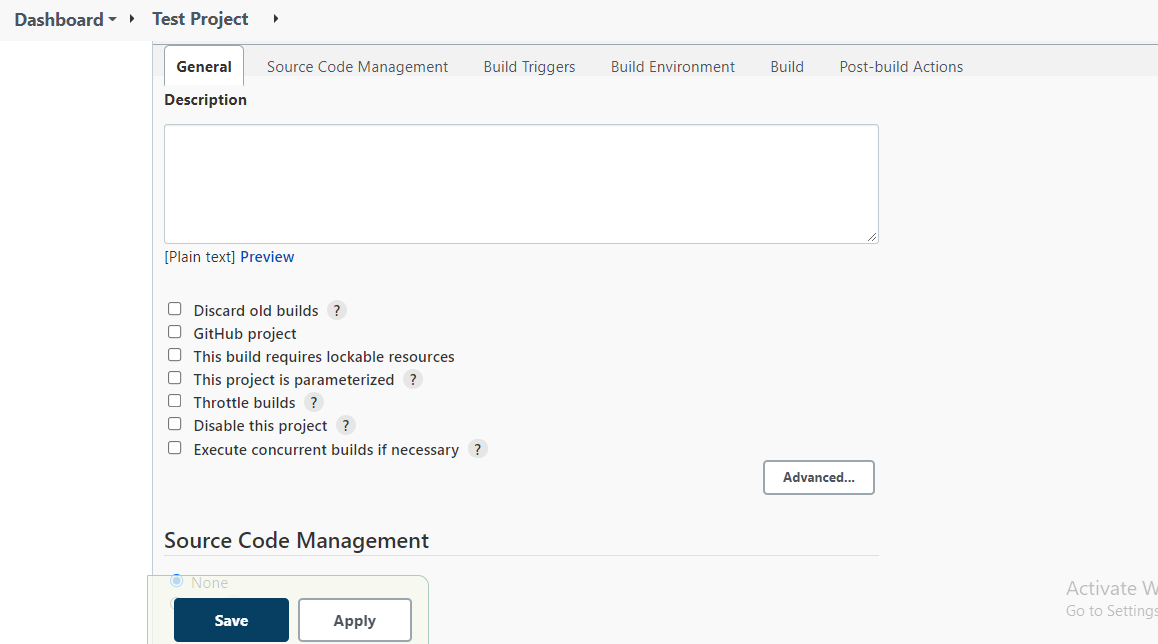
1. Click on New Item option like below screen

****

1. Enter the name of the project and select freestyle option the click on **ok** button like below screen.

****

1. **You can find below screen after you click ok button.**

****

1. Now it is on General tab, find in above screen. And here everything is optional whatever you required you can use.
2. **Description**:- Write your project description/project information.
3. **Discard old builds**:- This determines when, if ever, build records for this project should be discarded. Build records include the console output, archived artifacts, and any other metadata related to a particular build. Keeping fewer builds means less disk space will be used in the Build Record Root Directory, which is specified on the Configure System screen.

**Build age**: discard builds when they reach a certain age; for example, seven days old.

**Build count**: discard the oldest build when a certain number of builds already exist.

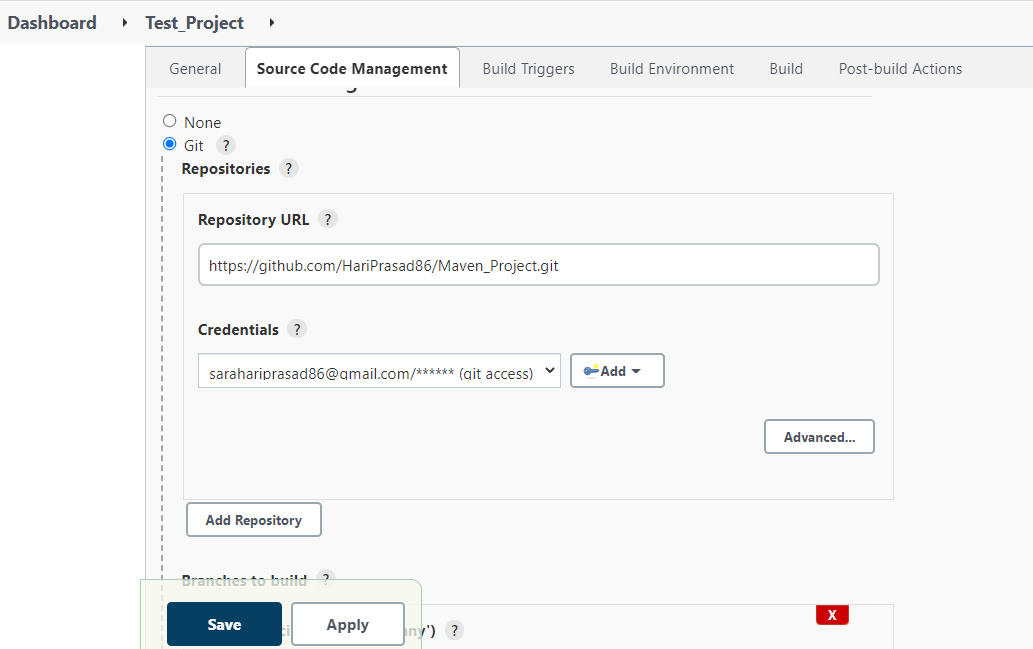
1. **GitHub project**
2. **This build requires lockable resources**
3. **This project is parameterized**:- Parameters allow you to prompt users for one or more inputs that will be passed into a build. For example, you might have a project that runs tests on demand by allowing users to upload a zip file with binaries to be tested. This could be done by adding a File Parameter here.
4. **Throttle builds**: Enforces a minimum time between builds based on the desired maximum rate.

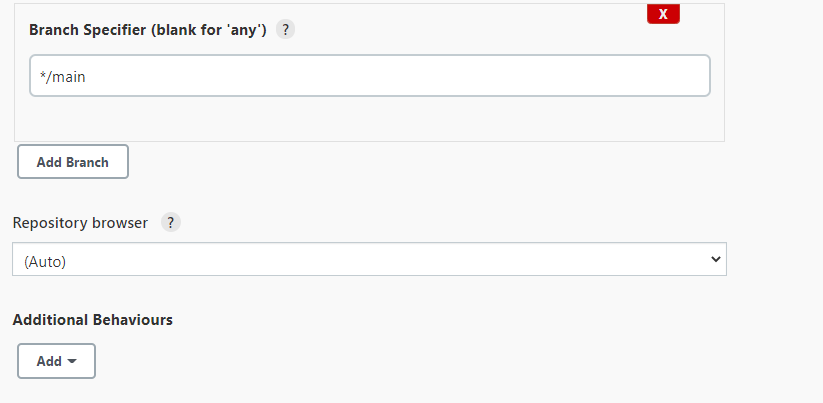
Note: this does not enforce an "average" rate, it only looks at the time since last build.

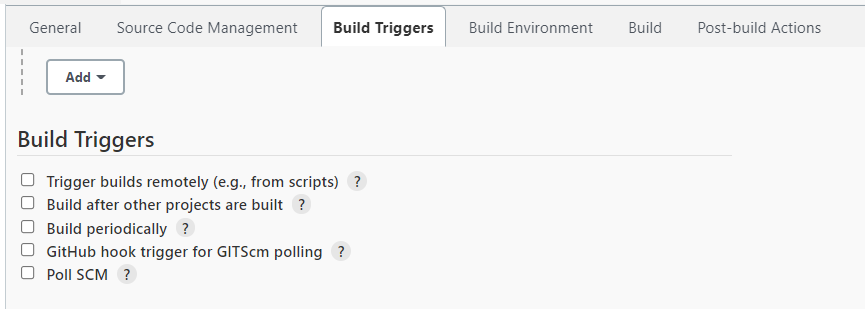
1. **Disable this project**: When this option is checked, no new builds of this project will be executed.

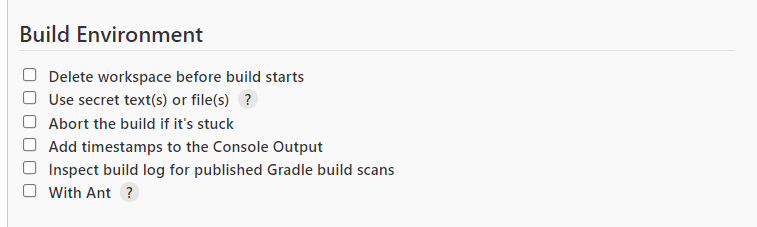
This can be helpful when you want to temporarily prevent a project from being built. For example, if your project depends on some infrastructure — e.g. a test server, or a source code repository — and you know it will be unavailable for a period of time, you could disable the project to prevent unnecessary build failures (and any corresponding notifications) during this period.

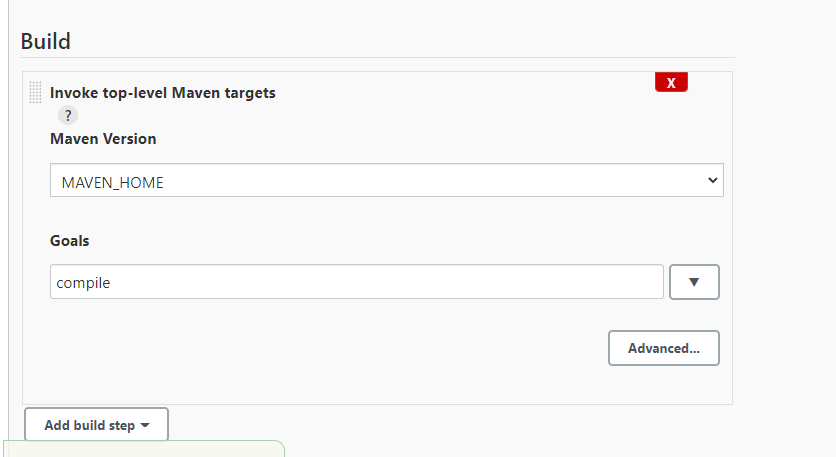
1. **Execute concurrent builds if necessary:** When this option is checked, multiple builds of this project may be executed in parallel. By default, only a single build of a project is executed at a time — any other requests to start building that project will remain in the build queue until the first build is complete.

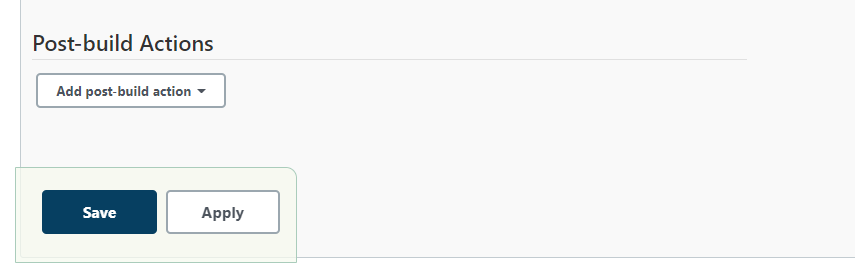
****

****

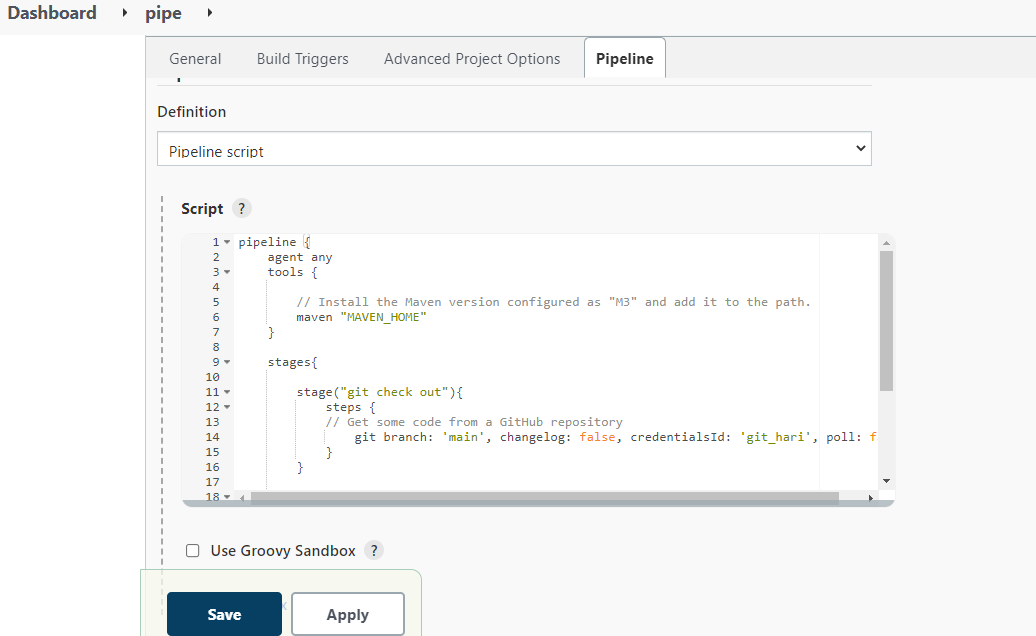
****

****

****

****

**Pipeline**

****

**Sample Script for pipeline script**

**pipeline {**

**agent any**

**tools {**

**// Install the Maven version configured as "M3" and add it to the path.**

**maven "MAVEN\_HOME"**

**}**

**stages{**

**stage("git check out"){**

**steps {**

**// Get some code from a GitHub repository**

**git branch: 'main', changelog: false, credentialsId: 'git\_hari', poll: false, url: 'https://github.com/HariPrasad86/Maven\_Project.git'**

**}**

**}**

**stage("build") {**

**steps {**

**// To run Maven on a Windows agent, use**

**bat "mvn clean package"**

**}**

**}**

**}**

**}**

**Cloud Concepts**

**What is cloud?**

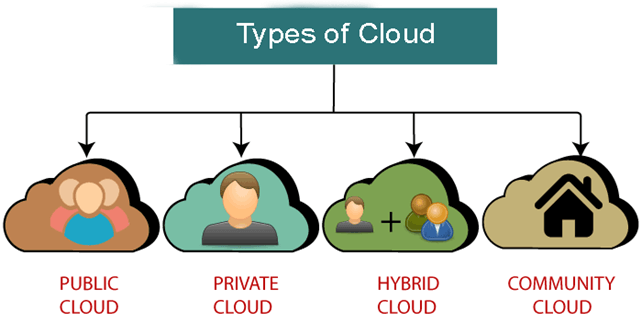
"The cloud" refers to servers that are accessed over the Internet and the software and databases that run on those servers. Cloud servers are located in data centers all over the world. By using cloud computing, users and companies do not have to manage physical servers themselves or run software applications on their own machines.

**Why cloud?**

12 business advantages of cloud computing.

1. Cost Savings
2. Security
3. Flexibility
4. Mobility
5. Insight
6. Increased Collaboration
7. Quality Control
8. Disaster Recovery
9. Loss Prevention
10. Automatic Software Updates
11. Competitive Edge
12. Sustainability

**Types of cloud**

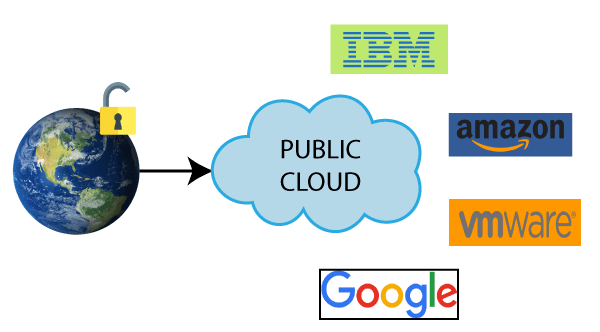


**Public Cloud:-**

**Public cloud** is open to all to store and access information via the Internet using the pay-per-usage method.

In public cloud, computing resources are managed and operated by the Cloud Service Provider (CSP).

**Example:** Amazon elastic compute cloud (EC2), IBM SmartCloud Enterprise, Microsoft, Google App Engine, Windows Azure Services Platform.



**Advantages of Public Cloud**

There are the following advantages of Public Cloud -

* Public cloud is owned at a lower cost than the private and hybrid cloud.
* Public cloud is maintained by the cloud service provider, so do not need to worry about the maintenance.
* Public cloud is easier to integrate. Hence it offers a better flexibility approach to consumers.
* Public cloud is location independent because its services are delivered through the internet.
* Public cloud is highly scalable as per the requirement of computing resources.
* It is accessible by the general public, so there is no limit to the number of users.

**Disadvantages of Public Cloud**

* Public Cloud is less secure because resources are shared publicly.
* Performance depends upon the high-speed internet network link to the cloud provider.
* The Client has no control of data.

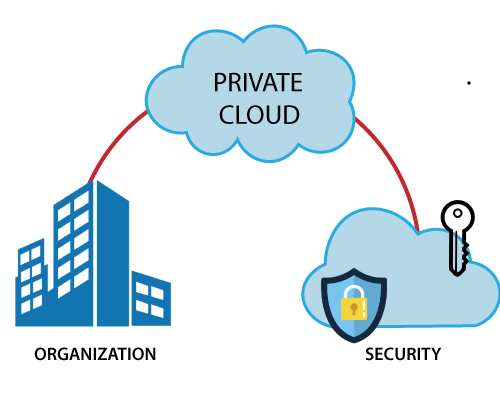
**Private Cloud**

Private cloud is also known as an internal cloud or corporate cloud. It is used by organizations to build and manage their own data centers internally or by the third party. It can be deployed using Opensource tools such as Openstack and Eucalyptus.

Based on the location and management, National Institute of Standards and Technology (NIST) divide private cloud into the following two parts-

**On-premise private cloud**

**Outsourced private cloud**



**Advantages of Private Cloud**

There are the following advantages of the Private Cloud -

* Private cloud provides a high level of security and privacy to the users.
* Private cloud offers better performance with improved speed and space capacity.
* It allows the IT team to quickly allocate and deliver on-demand IT resources.
* The organization has full control over the cloud because it is managed by the organization itself. So, there is no need for the organization to depends on anybody.
* It is suitable for organizations that require a separate cloud for their personal use and data security is the first priority.

**Disadvantages of Private Cloud**

* Skilled people are required to manage and operate cloud services.
* Private cloud is accessible within the organization, so the area of operations is limited.
* Private cloud is not suitable for organizations that have a high user base, and organizations that do not have the prebuilt infrastructure, sufficient manpower to maintain and manage the cloud.

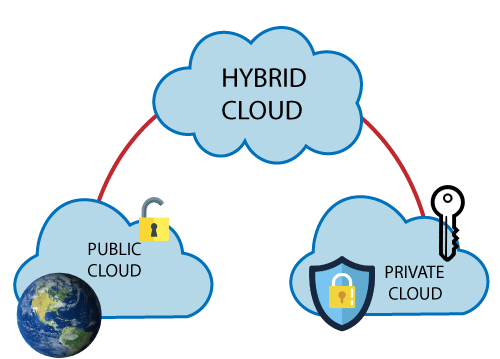
**Hybrid Cloud**

Hybrid Cloud is a combination of the public cloud and the private cloud. we can say:

**Hybrid Cloud = Public Cloud + Private Cloud**

Hybrid cloud is partially secure because the services which are running on the public cloud can be accessed by anyone, while the services which are running on a private cloud can be accessed only by the organization's users.

Example: Google Application Suite (Gmail, Google Apps, and Google Drive), Office 365 (MS Office on the Web and One Drive), Amazon Web Services.



**Advantages of Hybrid Cloud**

There are the following advantages of Hybrid Cloud -

* Hybrid cloud is suitable for organizations that require more security than the public cloud.
* Hybrid cloud helps you to deliver new products and services more quickly.
* Hybrid cloud provides an excellent way to reduce the risk.
* Hybrid cloud offers flexible resources because of the public cloud and secure resources because of the private cloud.

**Disadvantages of Hybrid Cloud**

* In Hybrid Cloud, security feature is not as good as the private cloud.
* Managing a hybrid cloud is complex because it is difficult to manage more than one type of deployment model.
* In the hybrid cloud, the reliability of the services depends on cloud service providers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Public Cloud** | **Private Cloud** | **Hybrid Cloud** | **Community Cloud** |
| **Host** | Service provider | Enterprise (Third party) | Enterprise (Third party) | Community (Third party) |
| **Users** | General public | Selected users | Selected users | Community members |
| **Access** | Internet | Internet, VPN | Internet, VPN | Internet, VPN |
| **Owner** | Service provider | Enterprise | Enterprise | Community |

**Microsoft AZUR**

Windows Azure, which was later renamed as Microsoft Azure in 2014, is a cloud computing platform, designed by Microsoft to successfully build, deploy, and manage applications and services through a global network of datacenters.

**WHAT ARE AZURE CLOUD SERVICES TYPES**

The MS Azure cloud provides a variety of services for almost every scenario you may imagine and need. These services are categorized into different cloud computing types such as IaaS, PaaS, and SaaS. That is to know the amount of responsibility and control you need against the time you have to spend to build the things that matter.

**Infrastructure as a Service (IaaS)**

Azure IaaS is the most basic cloud computing service category. It enables you to rent IT infrastructures like operating systems, networks, storage, virtual machines, and servers on a pay-as-you-go basis.

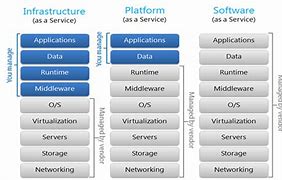
This Azure cloud service gives you a lot of control over how you run the infrastructures.

**Platform as a Service (PaaS)**

Azure PaaS is a cloud computing service that supplies on-demand software application development, testing, delivery, and management. It is designed to help developers create mobile or web apps easily and quickly, without setting up or managing the underlying infrastructure of databases, networks, storage, and servers necessary for development.

**Software as a Service (SaaS)**

Azure SaaS refers to the method for providing software on demand and over the Internet. It is on a subscription basis, where cloud computing providers host and manage the underlying infrastructure and software applications. These providers also handle maintenance, including security patching and software upgrades.



**Azure key Concepts**

**Regions**

Azure is a global cloud platform which is available across various regions around the world. When you request a service, application, or VM in Azure, you are first asked to specify a region. The selected region represents datacenter where your application runs.

**Datacenter**

In Azure, you can deploy your applications into a variety of data centers around the globe. So, it is advisable to select a region which is closer to most of your customers. It helps you to reduce latency in network requests.

**Azure portal**

The Azure portal is a web-based application which can be used to create, manage and remove Azure resource and services. It is located at <https://portal.azure.com>.

**Resources**

Azure resource is an individual computer, networking data or app hosting services which charged individually. Some common resources are virtual machines( VM), storage account, or SQL databases.

**Resource groups**

An Azure resource group is a container which holds related resource for an Azure solution. It may include every resource or just resource which you wants to manage.

**Resource Manager templates**

It is a JSON which defines one or more resource to deploy to a resource group. It also establishes dependencies between deployed resources

**Automation**

Azure allows you to automate the process of creating, managing and deleting resource by using PowerShell or the Azure command-line Interface(CLI).

**Azure PowerShell**

PowerShell is a set of modules that offer cmdlets to manage Azure. In most cases, you are allowed to use, the cmdlets command for the same tasks which you are performing in the Azure portal.

**Azure command-line interface(CLI)**

The Azure CLI is a tool that you can use to create, manage, and remove Azure resources from the command line.

**REST APIs**

Azure is built on a set of REST APIs help you perform the same operation that you do in Azure portal Ul. It allows your Azure resources and apps to be manipulated via any third party software application.

**Azure Account creation**

To create your azure account please follow the below link.

<https://k21academy.com/microsoft-azure/create-free-microsoft-azure-trial-account/#:~:text=1%20Go%20to%20the%20Azure%20Home%20Page.%202,to%20the%20Pay%20as%20you%20go%20subscription%20policy>.

After sign to the Azure account you can see portal like below screen.

