# GIT

* **VCS -----Build/package------ST----PT----UAT(pre-prod)---Live**
* **Git is a common server to store a code for multiple developers**
* **Git is a Distributed version control system**
* **Version should be maintained for every change**
* **Multiple developers can work on the same the file/code**
* **Git work flow in local Repo:**
  + **Workingtree(untrackedfile) --{Add} ----- StagingArea(trackedfile) ----{commit}-- Localrepo**

* **Untracked file is a new file which has no history in local repo. Modified file is existing file which has history in local repo.**
* **Now add the untracked file to staging area and commit the changes to local repo**
* **Git has a way to ignore certain files/foders. For that create a file called as .gitignore in the repository root.**

#### Moving Changes from Staging Area to Working Tree

**Working tree add staging area COMMIT localrepo**

**reset**

* **Every Change moved from staging Area to Local Repo is called as COMMIT**
* **Every commit has a commit ID and HEAD is a pointer in Git**
* **MERGE:**
  + **Combine a changes from one branch to another branch is called merge**
  + **Merge can be two types**
  + **1.Fast-forward(no extra commits created)when there is no change in parent branch after creating a child branch.**
  + **2.Merge(Extra commit created) when there is a change in parent (mater)branch after creating a child branch.**

* **Rebase:**
  + **Combine a changes from one branch to another branch with clear history of individual branch is called merge**
* **Git Uses hashing (SHA-1) to calculate commit ids. Commit id is hash of many things.**
* **GIT Remote repository:**

**Working tree add staging area COMMIT localrepo Push Pull**

**Remoterepository**

* **GIT Platforms (where the git remote repository present)**
  + **Git hub**
  + **Bit bucket**
* **Whenever a Remote Repo is added, new branches gets created in Local Repository. <Remote-name>/<branch-name> will be the name for the remote branch**
* **Interactive rebase:**
  + **Using the interactive rebase we can change the history of commits Like merging two commits and removing commits**
* **Cherrypick:**
  + **To pick individual/sequence of commit(s) from one branch to other use cherry-pick**
* **Bare Repositories:**
  + **Git repositiries with only .git folders**
* **GIT FLOW or BRANCHING STRATEGY:**
  + **Git Flow is the successful branching suggested by**[**Vincent Driessen**](https://nvie.com/posts/a-successful-git-branching-model/)
  + **Flow-chart**
  + **Master -------- sprint-1(developer)----------- Feature ( Developer woking branch) Internal devilary) Releasebranch------ Hotflix branch(after delivare customer find a bugs we need to resolve using this branch) Master(devlivar to customer)**
  + **We need to create a branches from the master that is sprint-1 and feature branch.**
  + **Developer can work only on feature branch and after completing their work they will merge a changes to sprint-1 branch**
  + **Sprint-1 branch changes are merged to release branch this is internal delivary**
  + **When ever we have a changes in release branch we have to create a environment and unit tesing after completing this we can merge into masterbranch.**
  + **If the customer is facing a problem with the application some of the experts can go to hotflix branch and they can resolve.**
* **Git stash:**
  + **Stash helps in preserving the work done in git working tree for future purposes.**
  + **To stash your changes**
  + **1. git stash**
  + **2.git stash list**
  + **To apply the changes from stash to working tree(git stash apply)**
  + **To apply the changes from stash and remove the changes from stash list(git stash pop)**
* **Git is a 5 areas:**
  + **W.T-----S.A-----LP----RR**

**Stash**

* **Git communication protocol :**
  + **Local (provide folder path)**
  + **SSH (using the SSh link from remote remote-repositary)**
  + **Http(s) (using the Http link from remote remote-repositary)**
  + **Git (using the git link)**
* **GIT Tag:**
  + **Git tag is a special recognization of commit**
  + **Generally we apply a tag for every release**
  + **Two types of tags**
  + **1.lightweighttag(name and creation)**
  + **2.Annotated tag(it is having name and metadata)**
  + **Creation of tag ------ git tag –a <name of the tag>**
  + **Pushing tag to remote------git push origin <tagname>**
* **Git servers**
  + **Hosted servers (Git hub,Git lab—like remote repository)**
  + **Self hosted servers(we are going to create manually)**
* **Git Hooks**
  + **Hook is a way of adding additional functionality to git as a response to some git actions.**
  + **Two types**
  + **1.client side hook**
  + **2.serverside hook (On Hosted Git Servers they are also called as WebHooks.)**
* **Git commands:**
  + **Git init -------------(make it as a repository)**
  + **Git status----------(show all tracked and untracked files)**
  + **Git add -------------(add a changes to staging area)**
  + **Git reset ----------(remove a specific file from staging area&history is discarded)**
  + **Git reset –hard----(remove all changes from staging area)**
  + **Git reset –soft ----(**
  + **Git checkout <filename> ---(discard the changes in working Dir.)**
  + **Git revert -----(modify a commit and will create a new revert commit)**
  + **Git config –global user.name<>--(configure the git in local)**
  + **Git config –global user.email<>**
  + **Git commit –m <msg>**
  + **Git add –u -----(add only modified files)**
  + **Git log ------(show all the commits)**
  + **.gitignore----(ignore a files specified in .gitignore)**
  + **Git clean –fd----(remove untracked files from working tree)**
  + **Git rm --------(used to remove files from remote repository)**
  + **Git merge <specific branch name to merge>---(merge a commits to another branch)**
  + **Git rebase< name of branch> (merge with clear history)**
  + **Git rebase –i HEAD~1----(change a commits history)**
  + **Git diff <two branch names>-----(show the difference)**
  + **git remote add <name-of-remote> <url>----(add remote repository)**
  + **git push <name-of-the-remote> <branch-name>(push changes to remote)**
  + **git pull <name-of-the-remote> <branch-name> (pull changes from remote )**
  + **git pull-----(fetch+merge)**
  + **git clone <url> -----(clone a repository from remote)**
  + **git cherry-pick <commitid>---(pick a specific commit and merge into another branch)**
  + **git clone --bare <url>-----(clone only .git folders)**
  + **Git reflog----(permenant record of commits)**
  + **Git stash save <msg>----(save a files from staging to stasharea)**
  + **Git stash apply <stash1)---- (copy a files from stash to branch)**
  + **Git stash pop<stash0> ------(cut a files from stash to branch)**
  + **Git commit –amend ----(fix any broken commit or modify the most recent commit)**
  + **Git status------(difference betwn working tree and staging area)**