|  |
| --- |
| What is Git? |
|  |

|  |
| --- |
| git is an open source vcs that allows the user to keep track |
|  |

|  |
| --- |
| of all the changes that have been made to the source code of the software. |
|  |

|  |
| --- |
| (also called vcs- version control system) Global information tracker (Git) |
|  |

|  |
| --- |
| introduced in 2005 |
|  |

|  |
| --- |
| vcs examples :- code commit, git, mercurial |
|  |

|  |
| --- |
| Git is a DevOps 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. |
|  |

|  |
| --- |
| ============================================================================= |
|  |

|  |
| --- |
| why we need git ? |
|  |

|  |
| --- |
| One of the biggest advantages of Git is its branching capabilities. |
|  |

|  |
| --- |
| Unlike centralized version control systems, Git branches are cheap |
|  |

|  |
| --- |
| and easy to merge. This facilitates the feature branch workflow popular |
|  |

|  |
| --- |
| with many Git users. Feature branches provide an isolated environment |
|  |

|  |
| --- |
| for every change to your codebase. |
|  |

|  |
| --- |
| features of git |
|  |

|  |
| --- |
| 1 system compatibility 2. collaboration 3.speed 4.distributed system 5.relibility 6. security |
|  |

|  |
| --- |
| git lifecycle assign working directory, initilization, staging, commit in local system, push to github. |
|  |

|  |
| --- |
| ============================================================================= |
|  |

|  |
| --- |
| Installing Git |
|  |

|  |
| --- |
| Debian/Ubuntu |
|  |

|  |
| --- |
| For the latest stable version for your release of Debian/Ubuntu |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # apt-get install git |
|  |

|  |
| --- |
| For Ubuntu, this PPA provides the latest stable upstream Git version |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # add-apt-repository ppa:git-core/ppa # apt update; apt install git |
|  |

|  |
| --- |
| =========================================================================== |
|  |

|  |
| --- |
| diff between working dir and local repo |
|  |

|  |
| --- |
| The contents of your project folder (the folders and files you find within it) are represented by the working directory. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| The working directory is sort of like a workbench, it's where you work on your files (you edit them, you add new files, you delete files etc.). |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| On the other hand, the .git folder (which is a hidden folder) represents the repository. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Within the .git folder there are two "places" that should be mentioned, the staging area |
|  |

|  |
| --- |
| (represented by the index file) and the commit history (represented by the objects folder). |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| The staging area is sort of like a rough draft space. Whenever you are done working on a file |
|  |

|  |
| --- |
| (or files) in your working directory, you want to copy them to the staging area (using the git add command). |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Once you have all the files that you want to update in the next version of your project in the |
|  |

|  |
| --- |
| staging area, you are ready to save them in the next version of your project which is called a commit. You do this using the git commit command. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| A commit is basically a version of your project and each commit has a 40 character hash |
|  |

|  |
| --- |
| (40 letters and numbers) and this hash acts like a name for the commit, it's a way to refer to it. |
|  |

|  |
| --- |
| ============================================================================================== |
|  |

|  |
| --- |
| git pull tries to automatically merge after fetching commits. It is context sensitive, |
|  |

|  |
| --- |
| so all pulled commits will be merged into your currently active branch. git pull automatically |
|  |

|  |
| --- |
| merges the commits without letting you review them first. If you don’t carefully |
|  |

|  |
| --- |
| manage your branches, you may run into frequent conflicts. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| git fetch gathers any commits from the target branch that do not exist in the current branch |
|  |

|  |
| --- |
| and stores them in your local repository. However, it does not merge them with your current branch. |
|  |

|  |
| --- |
| This is particularly useful if you need to keep your repository up to date, but are working on |
|  |

|  |
| --- |
| something that might break if you update your files. To integrate the commits into your current branch, you must use git merge afterwards. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ""GIT FETCH TELLS LOCAL REPOSITORY ABOUT CHENGES MADE IN REMOTE REPOSITORY IF WE WANT TO SAVE THE CHENGES THEN |
|  |

|  |
| --- |
| WE NEED TO DO <git merge origin/master> it will update whatever changes made in remote to local repo."" |
|  |

|  |
| --- |
| We can see that with the help of just git pull command we directly fetched and merged our remote repository with the local repository. |
|  |

|  |
| --- |
| git pull = git fetch + git merge |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Git Fetch Git Pull |
|  |

|  |
| --- |
| 1. Gives the information of a new change from a 1. Brings the copy of all the changes from a remote repository |
|  |

|  |
| --- |
| remote repository without merging into the and merges them into the current branch |
|  |

|  |
| --- |
| current branch. |
|  |

|  |
| --- |
| 2. Repository data is updated in the .git directory 2. The local repository is updated directly |
|  |

|  |
| --- |
| 3. Review of commits and changes can be done 3. Updates the changes to the local repository immediately. |
|  |

|  |
| --- |
| 4. No possibility of merge conflicts. 4. Merge conflicts are possible if the remote and the local |
|  |

|  |
| --- |
| repositories have done changes at the same place. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| git clone is used for just downloading exactly what is currently working on the remote server repository |
|  |

|  |
| --- |
| and saving it in your machine's folder where that project is placed. Mostly it is used only when we are going |
|  |

|  |
| --- |
| to upload the project for the first time. After that pull is the better option. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| git pull is a (clone(download) + merge) operation and mostly used when you are working as teamwork. |
|  |

|  |
| --- |
| In other words, when you want the recent changes in that project, you can pull. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ==================================================================================================== |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| git init >it will initiate empty git repo to that particular dir or folder |
|  |

|  |
| --- |
| .git dir will made |
|  |

|  |
| --- |
| git status > information about branch how many commits you made how many tracked or untracked files are done |
|  |

|  |
| --- |
| git status commnd should be run inside the initilised folder only |
|  |

|  |
| --- |
| git checkout -b dev it will create new branch dev |
|  |

|  |
| --- |
| git checkout -b master it will switched to master branch |
|  |

|  |
| --- |
| branch :- A branch represents an independent line of development. |
|  |

|  |
| --- |
| Branches serve as an abstraction for the edit/stage/commit process. |
|  |

|  |
| --- |
| You can think of them as a way to request a brand new working directory, |
|  |

|  |
| --- |
| staging area, and project history. |
|  |

|  |
| --- |
| git branch - it shows the currant branch where we are working |
|  |

|  |
| --- |
| git branch <branch name> it will create new branch |
|  |

|  |
| --- |
| git branch -d <branch name> it will delete particular branch |
|  |

|  |
| --- |
| rename branch |
|  |

|  |
| --- |
| step1 switch out to particular branch which we want to rename |
|  |

|  |
| --- |
| git checkout feature1 |
|  |

|  |
| --- |
| git branch -m featureA this will rename branch feature1 to featureA |
|  |

|  |
| --- |
| ======================================================================= |
|  |

|  |
| --- |
| Merging |
|  |

|  |
| --- |
| The git merge allows a user to merge different branches into one. |
|  |

|  |
| --- |
| it will create a new commit on merging. |
|  |

|  |
| --- |
| history of commits are not transferred. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for merge one branch to other we need to switch to reciveing branch |
|  |

|  |
| --- |
| where we want our merge |
|  |

|  |
| --- |
| first we need to checkout to master then |
|  |

|  |
| --- |
| git merge feature-A |
|  |

|  |
| --- |
| ======================================================================== |
|  |

|  |
| --- |
| git merge conflicts |
|  |

|  |
| --- |
| two peoples pulls same code and work or change same file and merge to main then conflicts create |
|  |

|  |
| --- |
| lets see how to resolve that |
|  |

|  |
| --- |
| git mergetool -t vimdiff this will open vim editor and let us decide to choose the correct code and save them to avoid conflicts |
|  |

|  |
| --- |
| ========================================================================== |
|  |

|  |
| --- |
| Git rebase |
|  |

|  |
| --- |
| git rebase allow us to shift set of commits and add as new base |
|  |

|  |
| --- |
| git rebase is same as git merge but it also bring history of commits to the current branch |
|  |

|  |
| --- |
| ========================================================================= |
|  |

|  |
| --- |
| git branching strategy |
|  |

master---hotfix---release---dev---features standard branching

