Sr. Software Engineer (SSE)

ABSTRACT

This is one of the subject from my personal notes series named “Coding-With-Arqam” that I am developing from the start of my professional development career.

Subject

Git

Git

--> Commands:

-> git config user.name

-> git config user.email

--> Git Push:

-> Local branch: git push <repo name> <branch name>

-> Remote branch: git push <remote> --all

-> git push <remote> --tags

-> git push <remote> --delete // delete some specific remote branch

-> git add <filename> //

-> git commit –m //

-> git push origin master //

-> git push --force <remote-name><Branch-name>

--> Git Fetch vs Git Pull:

-> GitHub is a platform that provides to host the code under software development version control.

-> It provides access control and various features such as bug tracking, task management, etc for every project.

-> Git: Fetches the required information only to the local repository => Fetches the required information not only to the local repository but also to the workspace that you are currently working in.

-> Github: the content of the specified branch is only downloaded. => the content of the specified branch is downloaded and also the changes are committed to the local repository.

-> Its main function is to fetch the content => Combination of fetch and merges the content.

-> It has only command-line syntax => It has command-line syntax as well as a pull request to post the changes.

-> Command used: git fetch <remote> <branch> => Command used: git pull <branch>

-> How to Make a Git Pull Request?

-> Code in local repo -> push code -> creates pull req -> CR, feedback -> update code on negetive feedback -> project maintainer then merges the code into the original repository and closes the pull request.

-> It's also possible to create a pull request even if the implementation of the feature is incomplete.

For example, if the user/developer is having difficulty in implementing the requirement,

then they can create a pull request containing the status as work in progress.

-> Less better => More

-> Safer => Less safer

--> Git ReBase vs Merge:

-> Commits:

->

--> GitHub:

-> Git repository hosting service.

-> GitHub also facilitates with many of its features, such as access control and collaboration.

--> Git vs GitHub:

-> Distributed version control tool that can manage a programmer's source code history. => Cloud-based tool developed around the Git tool.

-> Installs Git tool locally -> Online service

-> Git focused on version control and code sharing => GitHub focused on centralized source code hosting.

--> GitHub VS SVN:

-> Distributed version control platform => Centralized

-> Uses multiple repositories for accessing and maintaining of code => Does not have centralized repo for code maintenance.

-> Available in offline means can push code while offline => Online connection required.

-> Faster commit because you work on local repo => Slower because you have to push the code on centralized repo.

-> Single repo called branch where you keep the original/modified code => Additional repo called trunk along with branch where the final developed code is stored.

-> Content is stored as metadata => Storees files of content.

-> Cloning feature is available => Not available.

-> Supports branching and merging => Doen not supports merging.

-> SVN:

-> Ceantralized repo split into three key areas:

-> Trunk:

-> The developer keeps the original code.

-> No one should ever commit any broken code into the trunk as this is the central area where everyone has access to view the original code.

-> When you want to modify or change the features and functionalities you should branch your code from the trunk when you have finished coding then you should merge all changes back from the branch into the trunk.

-> Branches:

-> A branch is used when you want to add/modify a new feature.

-> So you branch the code from the trunk of that project.

-> When you finished coding you can merge your changes back into the trunk.

-> Tags:

-> It is similar to branching your code except that it will be never used.

-> They will do is by taking a copy of the trunk code and placing it inside a new folder along with the tag directory.

-> The difference between tag and branches is that it is not used for developing the code instead it is used for reverting your code back.

-> Tags are used when you are deploying the code from trunk and you will make a tag on the trunk and mark it as a new feature.

-> When you realize that the new tag has a broken code you can easily revert it back and fix the problem.

--> Git vs Mercurial:

-> Git is a little bit of complex than Mercurial.

-> It holds Linux heritage. => It is python based.

-> Git is slightly slower than Mercurial.

-> Git supports the unlimited number of parents. => Mercurial allows only two parents.

--> Git tools:

-> GitBash

-> Git GUI:

-> Git GUI is a powerful alternative to Git BASH.

-> It offers a graphical version of the Git command line function, as well as comprehensive visual diff tools.

-> We can access it by simply right click on a folder or location in windows explorer.

-> Also, we can access it through the command line by typing "git gui".

-> Gitk:

-> Graphical history viewer tool.

-> It's a robust GUI shell over git log and git grep.

-> Example:

-> gitk [git log options]

-> Git Third-Party Tools:

-> SourceTree, GitHub Desktop, etc.

--> Git Terminology:

-> Branch:

-> A branch is a version of the repository that diverges from the main working project.

-> It is an essential feature available in most modern version control systems.

-> A Git project can have more than one branch.

-> We can perform many operations on Git branch-like rename, list, delete, etc.

-> Checkout:

-> Act of switching between different versions of a target entity.

-> The git checkout command is used to switch between branches in a repository.

-> Cherry-Picking:

-> Apply some commit from one branch into another branch.

-> In case you made a mistake and committed a change into the wrong branch, but do not want to merge the whole branch. You can revert the commit and cherry-pick it on another branch.

-> Clone:

-> It is used to make a copy of the target repository or clone it.

-> If I want a local copy of my repository from GitHub, this tool allows creating a local copy of that repository on your local directory from the repository URL.

-> Fetch:

-> It is used to fetch branches and tags from one or more other repositories, along with the objects necessary to complete their histories.

-> HEAD:

-> HEAD is the representation of the last commit in the current checkout branch.

-> We can think of the head like a current branch.

-> When you switch branches with git checkout, the HEAD revision changes, and points the new branch.

-> Index:

-> The Git index is a staging area between the working directory and repository.

-> Master:

-> Master is a naming convention for Git branch.

-> It's a default branch of Git.

-> After cloning a project from a remote server, the resulting local repository contains only a single local branch. This branch is called a "master" branch.

-> Merge:

-> Merging is a process to put a forked history back together.

-> The git merge command facilitates you to take the data created by git branch and integrate them into a single branch.

-> Origin:

-> In Git, "origin" is a reference to the remote repository from a project was initially cloned.

-> More precisely, it is used instead of that original repository URL to make referencing much easier.

-> When we take pull from sourcetree, you may see that it is takin gpull from origin(remote branch).

-> Pull/Pull Request:

-> The term Pull is used to receive data from GitHub.

-> It fetches and merges changes on the remote server to your working directory.

- *-> The git pull command is used to make a Git pull.*
- *-> Pull requests are a process for a developer to notify team members that they have completed a feature.*
- *-> Once their feature branch is ready, the developer files a pull request via their remote server account.*
- *-> Pull request announces all the team members that they need to review the code and merge it into the master branch.*
- *-> How to Create a Pull Request:*
  - *-> To create a pull request, you need to create a file and commit it as a new branch.*

-> Push:

- *-> The push term refers to upload local repository content to a remote repository.*
- *-> Pushing is an act of transfer commits from your local repository to a remote repository.*
- *-> Pushing is capable of overwriting changes; caution should be taken when pushing.*

-> Rebase:

- *-> The process of moving or combining a sequence of commits to a new base commit.*
- *-> Rebasing is very beneficial and visualized the process in the environment of a feature branching workflow.*
- *-> From a content perception, rebasing is a technique of changing the base of your branch from one commit to another.*

-> Remote:

- *-> 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.*

-> Repository:

- *-> Repository is like a data structure used by VCS to store metadata for a set of files and directories.*
- *-> It contains the collection of the file as well as the history of changes made to those files.*
- *-> Repositories in Git is considered as your project folder.*

-> Stashing:

- *-> Sometimes you want to switch the branches, but you are working on an incomplete part of your current project.*
- *-> You don't want to make a commit of half-done work.*
- *-> The git stash command enables you to switch branch without committing the current branch.*
- *-> Stashing takes the dirty state of your working directory — that is, your modified tracked files and staged changes and saves it on a stack of unfinished changes that you can reapply at any time (even on a different branch).*

-> Tag:

- *-> Tags make a point as a specific point in Git history.*
- *-> It is used to mark a commit stage as important.*
- *-> We can tag a commit for future reference.*
- *-> Primarily, it is used to mark a projects initial point like v1.1.*
- *-> There are two types of tags:*
  - *-> Light-weighted tag*
  - *-> Annotated tag*

-> Git Revert:

- *-> Used to revert some commit.*
- *-> To revert a commit, git revert command is used. It is an undo type command.*
- *-> However, it is not a traditional undo alternative.*

-> Git Reset:

- *-> Stands for undoing changes.*
- *-> The git reset command is used to reset the changes.*

-> Git Ignore:

- *-> Used to specify intentionally untracked files that Git should ignore.*
- *-> It doesn't affect the Files that already tracked by Git.*

-> Git Squash:

- *-> Used to squash previous commits into one.*

-> Git squash is an excellent technique to group-specific changes before forwarding them to others.

-> You can merge several commits into a single commit with the powerful interactive rebase command.

-> Git Rm:

-> Remove

-> It is used to remove individual files or a collection of files.

->Git Fork:

-> A fork is a rough copy of a repository.

-> Forking a repository allows you to freely test and debug with changes without affecting the original project.

--> Git command line/Git commands:

-> Git supports many command-line tools and graphical user interfaces.

-> The Git command line is the only place where you can run all the Git commands.

-> Git config:

-> This command configures the user.

-> 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:

-> git config --global user.name "ImDwivedi1"

-> git config --global user.email "Himanshudubey481@gmail.com"

-> Git Init:

-> This command is used to create a local repository.

-> Syntax:

-> git init Demo

-> Git clone:

-> This command is used to make a copy of a repository from an existing URL.

-> Syntax:

-> git clone URL

-> Git add:

-> This command is used to add one or more files to staging (Index) area.

-> Syntax:

-> git add Filename

->git add Filename1, Filename2*

-> Git commit:

-> Commit command is used in two scenarios:

-> 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 status:

-> 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.

->Syntax:

-> git status

-> Git push:

-> 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.

-> Git push origin master:

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

-> Syntax:

-> git push [variable name] master (e.g: git push origin master)

-> Git push -all:

-> This command pushes all the branches to the server repository.

-> Syntax:

-> git push --all

-> Git pull:

-> 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 Branch:

-> This command lists all the branches available in the repository.

-> Syntax:

-> git branch

-> Git Merge:

-> This command is used to merge the specified branch's history into the current branch.

-> Syntax:

-> git merge BranchName (e.g: git merge master)

-> Git log:

-> 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 remote:

-> 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 Flow / Git Branching Model:

-> Git flow is the set of guidelines that developers can follow when using Git.

-> These are not the rules; it is a standard for an ideal project.

-> Types of branches:

-> Master

-> Develop

-> Hotfixes

-> Release branches

-> Feature branches

-> Main Branches:

-> Master:

-> The master branch is the main branch of the project that contains all the history of final changes.

-> Every developer must be used to the master branch.

-> The master branch contains the source code of HEAD that always reflects a final version of the project.

-> Develop:

-> It is parallel to the master branch.

-> It is also considered as the main branch of the project.

-> This branch contains the latest delivered development changes for the next release.

-> It has the final source code for the release.

-> It is also called as a "integration branch."

-> When the develop branch reaches a stable point and is ready to release, it should be merged with master and tagged with a release version.

-> Supportive Branches:

-> Feature branches, Release branches, Hotfix branches.

Reference Links

- *https://www.javatpoint.com/git*