Git command line

* There are many different ways to use Git. 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.
* Basic Git Commands
  + Git Config command
  + Git init command
  + Git clone command
  + Git add command
  + Git commit command
  + Git status command
  + Git push Command
  + Git pull command
  + Git Branch Command
  + Git Merge Command
  + Git log command
  + Git remote command

Git commands- Config

* Git config command
  + Git config
    - Get and set configuration variables that control all facets of how Git looks and operates.
  + You can view all of your settings and where they are coming from using:
    - $ git config --list --show-origin
  + Set the name:
    - $ git config --global user.name "User name"
    - $ git config --global user.name "Monica"
  + Set the email:
    - $ git config --global user.email "gupta.monicag@gmail.com"
  + Set the default editor: (here visual studio code, can have notepad.exe)
    - $ git config --global core.editor Vim
    - The below tells default editor VS Code to wait till the next command of git is given.
      * $ git config --global core.editor "code --wait"
  + Set the default branch:
    - $ git config --global init.defaultbranch main
  + Check the setting:
    - $ git config --list
* Git config command
  + Git alias
    - Set up an alias for each command:
    - $ git config --global alias.co checkout
    - $ git config --global alias.br branch
    - $ git config --global alias.ci commit
    - $ git config --global alias.st status

Git commands

* Starting a project
  + Git init
    - Create a local repository:
    - $ git init
  + Git clone
    - Make a local copy of the server repository.
    - $ git clone
* Local changes
  + Git add
    - Add a file to staging (Index) area:
      * $ git add Filename
    - Add all files of a repo to staging (Index) area:
      * $ git add\*
    - Git commit
      * Record or snapshots the file permanently in the version history with a message.
      * $ git commit -m " Commit Message"
* Track changes
  + Git diff
    - Track the changes that have not been staged:
      * $ git diff
    - Track the changes that have staged but not committed:
      * $ git diff --staged
    - Track the changes after committing a file:
      * $ git diff HEAD
    - Track the changes between two commits:
      * $ git diff Git Diff Branches:
      * $ git diff < branch 2>
  + Git status
    - Display the state of the working directory and the staging area.
      * $ git status
    - Git show Shows objects:
    - $ git show
* Commit History
  + Git log
    - Display the most recent commits and the status of the head:
      * $ git log
    - Display the output as one commit per line:
      * $ git log -oneline
    - Displays the files that have been modified:
      * $ git log -stat
    - Display the modified files with location:
      * $ git log -p
  + Git blame
    - Display the modification on each line of a file:
      * $ git blame <file name>
* Ignoring files
  + .gitignore
  + Specify intentionally untracked files that Git should ignore. Create .gitignore:
  + $ touch .gitignore List the ignored files:
    - $ git ls-files -i --exclude-standard
* Branching
  + Git branch
    - Create branch:
      * $ git branch
    - List Branch:
    - $ git branch --list
    - Delete a Branch:
      * $ git branch -d
    - Delete a remote Branch:
      * $ git push origin -delete
    - Rename Branch:
      * $ git branch -m
  + Git checkout
    - Switch between branches in a repository.
    - Switch to a particular branch:
      * $ git checkout
    - Create a new branch and switch to it:
      * $ git checkout -b
    - Checkout a Remote branch:
      * $ git checkout
* Branching
* Git stash
  + Switch branches without committing the current branch.
  + Stash current work:
    - $ git stash
  + Saving stashes with a message:
    - $ git stash save ""
  + Check the stored stashes:
    - $ git stash list
  + Re-apply the changes that you just stashed:
    - $ git stash apply
  + Track the stashes and their changes:
    - $ git stash show
  + Re-apply the previous commits:
    - $ git stash pop
  + Delete a most recent stash from the queue:
    - $ git stash drop
  + Delete all the available stashes at once:
    - $ git stash clear
  + Stash work on a separate branch:
    - $ git stash branch
* Git cherry pic
  + Apply the changes introduced by some existing commit:
    - $ git cherry-pick
* Merging
  + Git merge
    - Merge the branches/ Merge the specified commit to currently active branch:
      * $ git merge
  + Git rebase
    - Apply a sequence of commits from distinct branches into a final commit.
      * $ git rebase
    - Continue the rebasing process:
      * $ git rebase -continue
    - Abort the rebasing process:
      * $ git rebase --skip
  + Git interactive rebase
    - Allow various operations like edit, rewrite, reorder, and more on existing commits.
      * $ git rebase -i
* Remote
  + Git remote
    - Check the configuration of the remote server:
      * $ git remote -v
    - Add a remote for the repository:
      * $ git remote add
    - Fetch the data from the remote server:
      * $ git fetch
    - Remove a remote connection from the repository:
      * $ git remote rm
    - Rename remote server:
      * $ git remote rename
    - Show additional information about a particular remote:
      * $ git remote show
    - Change remote:
      * $ git remote set-url
  + Git origin master
    - Push data to the remote server:
      * $ git push origin master
    - Pull data from remote server:
      * $ git pull origin master
* Pushing Updates
  + Git push
    - Transfer the commits from your local repository to a remote server. Push data to the remote server:
      * $ git push origin master
    - Force push data:
      * $ git push -f
    - Delete a remote branch by push command:
      * $ git push origin -delete edited
* Pulling updates
  + Git pull
    - Pull the data from the server:
      * $ git pull origin master
    - Pull a remote branch:
      * $ git pull
  + Git fetch
    - Download branches and tags from one or more repositories. Fetch the remote repository:
      * $ git fetch< repository Url>
    - Fetch a specific branch:
      * $ git fetch
    - Fetch all the branches simultaneously:
      * $ git fetch -all
    - Synchronize the local repository:
      * $ git fetch origin
* Undo changes
  + Git revert
    - Undo the changes/ Revert a particular commit:
      * $ git revert
  + Git reset
    - Reset the changes:
      * $ git reset -hard
      * $ git reset -soft
      * $ git reset --mixed
* Removing files
  + Git rm
    - Remove the files from the working tree and from the index:
      * $ git rm <file Name>
    - Remove files from the Git But keep the files in your local repository:
      * $ git rm --cached

Git commands- Config, Init, clone, remote

* 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. Git config is also used in other scenarios.
  + $ git config --global user.name "username@mail.com"
* Git Init command
  + This command is used to create a local repository.
  + $ git init Demo
* Git clone command
  + This command is used to make a copy of a repository from an existing URL. If I want a local copy of my repository from GitHub, this command allows creating a local copy of that repository on your local directory from the repository URL.
  + $ git clone URL
* 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 add command
  + This command is used to add one or more files to staging (Index) area.
  + To add one file
    - $ git add Filename
  + To add more than one file
    - $ git add\*
* Git commit command
  + Commit command is used in two scenarios. They are as follows.
  + Git commit -m
    - This command changes the head. It records or snapshots the file permanently in the version history with a message.
    - $ 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.
    - $ git commit -a
* 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. It also lists the files that you've changed and those you still need to add or commit.
  + $ git status
* 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 command can be used as follows.
    - Git push origin master
    - This command sends the changes made on the master branch, to your remote repository.
    - $ git push [variable name] master
    - Git push -all
    - This command pushes all the branches to the server repository.
    - $ 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.
  + $ git pull URL
* Git Branch Command
  + This command lists all the branches available in the repository.
  + $ git branch
* Git Merge Command
  + This command is used to merge the specified branch?s history into the current branch.
  + $ git merge BranchName
* Git log Command
  + This command is used to check the commit history.
  + $ 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 -3