**GitLab Command:** Version Control system (**VCS**)

* **git version**
  + get current version of GIT which install in your system
* **git config user.name, git config user.email, git config --list**
  + For get the current user details
* **git config --global user.email "[test@gmail.com](mailto:test@gmail.com)" || git config --global user.name “Test”**
  + For change / set the user name
* **git init**
  + Initialize a local directory for Git version control
  + Create an empty Git repository or reinitialize an existing one
* **git clone <https://gitlab.com/gitlab-org/gitlab.git>**
  + get copy or clone from any public repository URL
  + Clone a repository into a new directory
* **git remote -v** 
  + Get git remote repository URL or link for push & pull or clone
* **git checkout master**
  + Get Checkout/Update code from **‘master’** repository or any repository
* **git checkout master**
  + Switch to a Different Branch by changing master to the branch name
* **git checkout -b {your\_local\_branch\_name}**
  + If you want to create a branch from the current branch
* **git checkout -b {your\_local\_branch\_name} origin/<remote\_branch\_name>**
  + you want a branch from a remote branch, you can try
* **git switch branchName**
  + Switch to different branch from current branch
* **git status**
  + Get file status
  + Changes not staged for commit:
* **git branch TestDev**
  + Create a new branch in the save git repository
* **git checkout -b bug\_fixes origin/master | git checkout -b bug\_fixes**
  + If you want to create a local branch bug\_fixes from a remote branch, say master or if you are in master branch
* **git push -u origin bug\_fixes**
  + You can push to the branch to remote repo by
* **git branch**
  + get current branch
* **git branch -m oldBranchName newBranchName**
  + rename a branch in local repository
* **git remote set-url origin https://<token>@github.com/<username>/<repo>**
  + **git remote set-url origin <https://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@github.com/mappsdeveloper1991/ReactNative>**
  + If any issue related to add/commit/and push request
* **git add -A**
  + Add anything in or all change in in single command
* **git add \* | git add . | git add fileName | git add folderName**
  + for add file/folder/all in untracked to tracked in staging area
  + Add new Untracked files or folder
  + Adds the file to your local repository and stages it for commit. To unstage a file, use 'git reset HEAD YOUR-FILE'.
* **git commit -m"Search, Superigniter, BrandShop and lot of bug fixed in this version with notification”** 
  + Commits the tracked changes and prepares them to be pushed to a remote repository.
  + commit the all add & changes file on staging area,
  + Make sure coming message should be proper & more then 25 character
* **git push | git push origin master | git push --set-upstream origin palmleaf\_android\_abhishek | git push origin branchName**
  + for push code on GitLab remote repository after add and commit
  + Push the changes in your local repository to GitHub.com
  + Pushes the changes in your local repository up to the remote repository you specified as the origin
* **git pull**
  + To get updates code from the remote repository directory/branch to the current local branch
* **git pull origin master**
  + Anytime you want to update your branch from any other branch say master,
* **git log** 
  + Get git log history
* **git diff** 
  + check diff b/w current local working copy and committed working copy diff
* **git merge master**
  + Merge a branch with the master branch
* **git push --set-upstream origin ChildBarnch**
  + Marge ChildBarnch into current/working branch
  + Marge remote branch and run the commit & push command
* **git restore <file>**..."
  + to discard changes in the working directory or file
  + Restore only from local directory not from staging & remote area
* **git restore RootPath or file name** "
  + to discard changes all in file in the rootPath directory or file
  + Restore only from local directory not from staging & remote area
* **git restore --staged <file>...**
  + Restore only from staging area directory not from local & remote directory
* **git reset 9a7655a6c2fc6fd394bc75ef10bbc1ae60f99c03**
  + reset all committed code after that commit version
* **git mv oldFileName newFileName**
  + Move or rename a file, a directory, or a symlink
* **git rm <file>**
  + Remove files from the working tree and from the indexx

**GIT BASICS**

**git config --global user.name <name>:** Define the author name to be used for all commits by the current user.

**git config --global user.name :** get the author name to be used for all commits by the current user.

**git config --global user.email <email>:** Define the author email to be used for all commits by the current user.

**git config --global user.email :** get the author email to be used for all commits by the current user.

**git init <directory>:** Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

**git clone <repo> :** Clone repo located at <repo> onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH

**git add <directory>:** Stage all changes in for the next commit. Replace <directory> with a <file> to change a specific file.

**git commit -m “<message>” :** Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message.

**git status :** List which files are staged, unstaged, and untracked.

**git log:** Display the entire commit history using the default format. For customization see additional options

**git diff:**  Show unstaged changes between your index and working directory

**GIT BRANCHES**

**git branch:**  List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.

**git checkout -b <branch>:** Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

**git merge <branch>:** Merge <branch> into the current branch.

**GIT** **REMOTE REPOSITORIES**

**git remote add <name> <url>:**  Create a new connection to a remote repo. After adding a remote, you can use <name> as a shortcut for <url> in other commands.

**git fetch <remote> <branch>:** Fetches a specific <branch> branch, from the repo. Leave off <branch> to fetch all remote refs.

**git pull <remote>:**  Fetch the specified remote’s copy of current branch and immediately merge it into the local copy.

**git push <remote> <branch>:** Push the branch to <remote>, along with necessary commits and objects. Creates named <branch> branch in the remote repo if it doesn’t exist

**GIT PULL/PUSH**

**git pull --rebase <remote>:** Fetch the remote’s copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches.

**git push <remote> --force**: Forces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you’re absolutely sure you know what you’re doing

**git push <remote> --all:** Push all of your local branches to the specified remote.

**git push <remote> --tags:** Tags aren’t automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local tags to the remote repo.

**GIT RESET/DIFF/REBASE/REVERT**

**git revert <commit>:** Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.

**git reset <file>:** Remove <file> from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes

**git reset :** Reset staging area to match most recent commit, but leave the working directory unchanged.

**git reset --hard:** Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory.

**git reset <commit>:** Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working directory alone.

**git reset --hard <commit>:** Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after <commit>**.**

**git clean -n:** Shows which files would be removed from working directory. Use the **-f** flag in place of the **-n** flag to execute the clean.

**git rebase <base>:** Rebase the current branch onto .<base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.

**GIT REWRITING HISTORY**

**git commit --amend :** Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit’s message.

**git rebase <base>:** Rebase the current branch onto .<base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.

**git reflog:**  Show a log of changes to the local repository’s HEAD. Add --relative-date flag to show date info or --all to show all refs

**GIT LOG**

**git log -<limit>:** Limit number of commits by <limit>. E.g. ”git log -5” will limit to 5 commits.

**git log —oneline:** Condense each commit to a single line

**git log -p:** Display the full diff of each commit.

**git log --<file>:** Only display commits that have the specified file.

**git log --graph --decorate:** --graph flag draws a text based graph of commits on left side of commit msgs. --decorate adds names of branches or tags of commits shown.