**Installation and Configuration of GIT**

**Git Commands**

**git config**

Usage: git config –global user.name “[name]”

Usage: git config –global user.email “[email address]”

This command sets the author name and email address respectively to be used with your commits.

Git Config Command - Git Commands - Edureka

**git init**

Usage: git init [repository name]

This command is used to start a new repository.

GitInit Command - Git Commands - Edureka

**git clone**

Usage: git clone [url]

This command is used to obtain a repository from an existing URL.



**git add**

Usage: git add [file]

This command adds a file to the staging area.

Git Add Command - Git Commands - Edureka

Usage: git add \*

This command adds one or more to the staging area.

Git Add Command - Git Commands - Edureka

**git commit**

Usage: git commit -m “[ Type in the commit message]”

This command records or snapshots the file permanently in the version history.



Usage: git commit -a

This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

Git Commit Command - Git Commands - Edureka

**git diff**

Usage: git diff

This command shows the file differences which are not yet staged.



 Usage: git diff –staged

This command shows the differences between the files in the staging area and the latest version present.



Usage: git diff [first branch] [second branch]

This command shows the differences between the two branches mentioned.



**git reset**

Usage: git reset [file]

This command unstages the file, but it preserves the file contents.



Usage: git reset [commit]

This command undoes all the commits after the specified commit and preserves the changes locally.

Git Reset Command - Git Commands - Edureka

Usage: git reset –hard [commit]  This command discards all history and goes back to the specified commit.

Git Reset Command - Git Commands - Edureka

**git status**

Usage: git status

This command lists all the files that have to be committed.



**git rm**

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

Git Rm Command - Git Commands - Edureka

**git log**

Usage: git log

This command is used to list the version history for the current branch.



Usage: git log –follow[file]

This command lists version history for a file, including the renaming of files also.



**git show**

Usage: git show [commit]

This command shows the metadata and content changes of the specified commit.



**git tag**

Usage: git tag [commitID]

This command is used to give tags to the specified commit.



**git branch**

Usage: git branch

This command lists all the local branches in the current repository.

Git Branch Command - Git Commands - Edureka

Usage: git branch [branch name]

This command creates a new branch.

Git Branch Command - Git Commands - Edureka

Usage: git branch -d [branch name]

This command deletes the feature branch.

Git Branch Command - Git Commands - Edureka

**git checkout**

Usage: git checkout [branch name]

This command is used to switch from one branch to another.

Git Checkout Command - Git Commands - Edureka

Usage: git checkout -b [branch name]

This command creates a new branch and also switches to it.

Git Checkout Command - Git Commands - Edureka

**git merge**

Usage: git merge [branch name]

This command merges the specified branch’s history into the current branch.

Git Merge Command - Git Commands - Edureka

**git remote**

Usage: git remote add [variable name] [Remote Server Link]

This command is used to connect your local repository to the remote server.

Git Remote Command - Git Commands - Edureka

**git push**

Usage: git push [variable name] master

This command sends the committed changes of master branch to your remote repository.



Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.



Usage: git push –all [variable name]

This command pushes all branches to your remote repository.



Usage: git push [variable name] :[branch name]

This command deletes a branch on your remote repository.



**git pull**

Usage: git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.



**git stash**

Usage: git stash save

This command temporarily stores all the modified tracked files.

Git Stash Command - Git Commands - Edureka

Usage: git stash pop

This command restores the most recently stashed files.



Usage: git stash list

This command lists all stashed changesets.

Git Stash Command - Git Commands - Edureka

Usage: git stash drop

This command discards the most recently stashed changeset.

Git Stash Command - Git Commands - Edureka

Want to learn more about git commands? Here is a [Git Tutorial](https://www.edureka.co/blog/git-tutorial/) to get you started. Alternatively, you can take a top-down approach and start with this [DevOps Tutorial.](https://www.edureka.co/blog/devops-tutorial)

|  |  |  |
| --- | --- | --- |
| Git task | Notes | Git commands |
| [**Tell Git who you are**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | Configure the author name and email address to be used with your commits.  Note that Git [strips some characters](http://stackoverflow.com/questions/26159274/is-it-possible-to-have-a-trailing-period-in-user-name-in-git/26219423#26219423) (for example trailing periods) from user.name. | git config --global user.name "Sam Smith"  git config --global user.email sam@example.com |
| [**Create a new local repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-init) |  | git init |
| [**Check out a repository**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone) | Create a working copy of a local repository: | git clone /path/to/repository |
| For a remote server, use: | git clone username@host:/path/to/repository |
| [**Add files**](https://www.atlassian.com/git/tutorials/saving-changes#git-add) | Add one or more files to staging (index): | git add <filename>  git add \* |
| [**Commit**](https://www.atlassian.com/git/tutorials/saving-changes#git-commit) | Commit changes to head (but not yet to the remote repository): | git commit -m "Commit message" |
| Commit any files you've added with git add, and also commit any files you've changed since then: | git commit -a |
| [**Push**](https://www.atlassian.com/git/tutorials/syncing#git-push) | Send changes to the master branch of your remote repository: | git push origin master |
| [**Status**](https://www.atlassian.com/git/tutorials/inspecting-a-repository#git-status) | List the files you've changed and those you still need to add or commit: | git status |
| [**Connect to a remote repository**](https://www.atlassian.com/git/tutorials/syncing#git-remote) | If you haven't connected your local repository to a remote server, add the server to be able to push to it: | git remote add origin <server> |
| List all currently configured remote repositories: | git remote -v |
| [**Branches**](https://www.atlassian.com/git/tutorials/using-branches) | Create a new branch and switch to it: | git checkout -b <branchname> |
| Switch from one branch to another: | git checkout <branchname> |
| List all the branches in your repo, and also tell you what branch you're currently in: | git branch |
| Delete the feature branch: | git branch -d <branchname> |
| Push the branch to your remote repository, so others can use it: | git push origin <branchname> |
| Push all branches to your remote repository: | git push --all origin |
| Delete a branch on your remote repository: | git push origin :<branchname> |
| [**Update from the remote repository**](https://www.atlassian.com/git/tutorials/syncing) | Fetch and merge changes on the remote server to your working directory: | git pull |
| To merge a different branch into your active branch: | git merge <branchname> |
| View all the merge conflicts:  View the conflicts against the base file:  Preview changes, before merging: | git diff  git diff --base <filename>  git diff <sourcebranch> <targetbranch> |
| After you have manually resolved any conflicts, you mark the changed file: | git add <filename> |
| **Tags** | You can use tagging to mark a significant changeset, such as a release: | git tag 1.0.0 <commitID> |
| CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using: | git log |
| Push all tags to remote repository: | git push --tags origin |
| [**Undo local changes**](https://www.atlassian.com/git/tutorials/undoing-changes) | If you mess up, you can replace the changes in your working tree with the last content in head:  Changes already added to the index, as well as new files, will be kept. | git checkout -- <filename> |
| Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this: | git fetch origin  git reset --hard origin/master |
| **Search** | Search the working directory for foo(): | git grep "foo()" |