**GIT**

Git is a [version control system](https://en.wikipedia.org/wiki/Version_control_system) for tracking changes in [computer files](https://en.wikipedia.org/wiki/Computer_file) and coordinating work on those files among multiple people. It is primarily used for [source code management](https://en.wikipedia.org/wiki/Source_code_management) in [software development](https://en.wikipedia.org/wiki/Software_development), but it can be used to keep track of changes in any set of files. As a [distributed revision control](https://en.wikipedia.org/wiki/Distributed_revision_control) system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

**How to install**

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows htps://windows.github.com

GitHub for Mac htps://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms htp://git-scm.com

**Process for initializing GIT**

To use Git on the command line, you'll need to download, install, and configure Git on your computer.

If you want to work with Git locally, but don't want to use the command line, you can instead download and install the [GitHub Desktop](https://desktop.github.com/) client. For more information, see "[Getting Started with GitHub Desktop](https://help.github.com/desktop/guides/getting-started-with-github-desktop/)."

Setting up Git

1. [Download and install the latest version of Git](https://git-scm.com/downloads).
2. [Set your username in Git](https://help.github.com/articles/setting-your-username-in-git).
3. [Set your commit email address in Git](https://help.github.com/articles/setting-your-commit-email-address-in-git).

### Next steps: Authenticating with GitHub from Git

When you connect to a GitHub repository from Git, you'll need to authenticate with GitHub using either HTTPS or SSH.

#### Connecting over HTTPS (recommended)

If you [clone with HTTPS](https://help.github.com/articles/which-remote-url-should-i-use/#cloning-with-https-urls-recommended), you can [cache your GitHub password in Git](https://help.github.com/articles/caching-your-github-password-in-git) using a credential helper.

#### Connecting over SSH

If you [clone with SSH](https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-ssh-urls), you must [generate SSH keys](https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent) on each computer you use to push or pull from GitHub.

**Commands of GIT**

1. CONFIGURE TOOLING

Configure user information for all local repositories

* git config --global user.name "[name]"

Sets the name you want atached to your commit transactions

* git config --global user.email "[email address]"

Sets the email you want atached to your commit transactions

* git config --global color.ui auto

Enables helpful colorization of command line output

1. CREATE REPOSOTORIES

Start a new repository or obtain one from an existing URL

* git init [project-name]

Creates a new local repository with the specified name

* git clone [url]

Downloads a project and its entire version history

1. MAKE CHANGES

* git status
* git diff
* git add [file]
* git diff –staged
* git reset [file]
* git commit -m "[descriptive message]"

1. GROUP CHANGES

* git branch
* git branch [branch-name]
* git checkout [branch-name]
* git merge [branch]
* git branch -d [branch-name]

1. REFACTOR FILENAMES
2. SUPPRESS TRACKING
3. SAVE FRAGMENTS
4. REVIEW HISTORY
5. REDO COMMITS
6. SYNCHRONIZE CHANGES

* git fetch [bookmark]
* git merge [bookmark]/[branch]
* git push [alias] [branch]
* git pull

**Creating new GIT repo**

1) Initialize the repository

Command:

**Git init**

2) Add all the files to the repository

Command:

**Git add .** (To add all the files)

**Git add filename**(To add the one file)

3) Save the changes

**Git commit**

**Git Branches:**

We can maintain many branches on its parent. By maintaining branches we can work individually and later we can merge those branches.

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branchname in Git is master . As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, the masterbranch pointer moves forward automatically.

**Commands:**

* git branch

Lists all local branches in the current repository

* git branch [branch-name]

Creates a new branch

**Git Checkout**

**Commands:**

git checkout [branch-name]

Switches to the specified branch and updates the working directory