Git

Git is a version control system for tracking changes in the computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linux Torvalds in 2005 for development of Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano.

As with most other distributed version control systems, and unlike most client-server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.

Git is free and open source software distributed under the terms of the GNU General Public License version 2.

## Characteristics:

* Strong support for non-linear development
* Distributed development
* Compatibility with existent systems and protocols
* Efficient handling of large projects
* Cryptographic authentication of history
* Toolkit-based design
* Pluggable merge strategies
* Garbage accumulates until collected
* Periodic explicit object packing

## Installation:

Installing on Linux

If you want the basic git tools on linux via a binary installer, you can generally do so through the package management tool that comes with your distribution. If you’re on Fedora(or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use dnf:

$ sudo dnf install git-all

If you’re on a Debian-based distribution, such as Ubuntu, try apt:

$ sudo apt install git-all

Installing on Mac

There are several ways to install Git on Mac. The easiest is probably to install the Xcode Command Line Tools. On Mavericks(10.9) or above you can do this simply by trying to run git from the Terminal the very first time.

$ git —version

If you don’t have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is maintained and available for download at the Git website, at URL: <http://git-scm.com/download/mac>.

For GUI installation on Mac you can go to <http://mac.github.com>.

Installing on Windows

There are also a few ways to install git on windows. The most official build is available for download on the Git website. Just go to <http://git-scm.com/download/win> and the download will start automatically. Note that this is a project called Git for windows, which is separate from Git itself; for more information on it, go to <https://git-for-windows.github.io/>.

Process for initialising 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:

### Configuring Tooling

Configure user information for all local repositories

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

Sets the name you want attached to your commit transactions

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

Sets the email you want attached to your commit transactions

* git config --global color.ui auto

Enables helpful colorisation of command line output

### Create Repositories

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

### Make Changes

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

### Group Changes

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

### Refactor File names

* git mv old\_filename new\_filename

Rename the file, specifying the old file name and the new name you’d like to give this file. This will stage your change for commit

* git status

To check the old and new file names

### Review History

* git reset HEAD^

It will move to the last commit from current commit. Each ‘^’ makes it move parent of the current commit

* git reset HEAD~[num]

Here you can give number of times you want to move in the past commits

### Synchronise Changes

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

Create New Git Repository:

### 1) Initialise the repository

Command:

$ git init

### 2) Add all the files to the repository

Command:

$ git add . (to all all files)

$ git add <file-name> (to all a specific 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