GIT

Git is an example of a distributed version control system (DVCS) commonly used for open source and commercial software development. DVCSs allow full access to every file, branch, and iteration of a project, and allows every user access to a full and self-contained history of all changes. Unlike once popular centralized version control systems, DVCSs like Git don’t need a constant connection to a central repository. Developers can work anywhere and collaborate asynchronously from any time zone.

**How to install GIT?**

Installing on Linux

If you want to install 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 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/>.

**What’s a repository?**

A repository, or [Git project](https://git-scm.com/), encompasses the entire collection of files and folders associated with a project, along with each file’s revision history. The file history appears as snapshots in time called commits, and the commits exist as a linked-list relationship, and can be organized into multiple lines of development called branches. Because Git is a DVCS, repositories are self-contained units and anyone who owns a copy of the repository can access the entire codebase and its history. Using the command line or other ease-of-use interfaces, a git repository also allows for: interaction with the history, cloning, creating branches, committing, merging, comparing changes across versions of code, and more

**Basic GIT Commands**

* git init :initializes a brand new Git repository and begins tracking an existing directory. It adds a hidden subfolder within the existing directory that houses the internal data structure required for version control.
* git clone:creates a local copy of a project that already exists remotely. The clone includes all the project’s files, history, and branches.
* git add:stages a change. Git tracks changes to a developer’s codebase, but it’s necessary to stage and take a snapshot of the changes to include them in the project’s history. This command performs staging, the first part of that two-step process.
* Git commit:saves the snapshot to the project history and completes the change-tracking process. In short, a commit functions like taking a photo. Anything that’s been staged with git add will become a part of the snapshot with git commit.
* Git status: shows the status of changes as untracked, modified, or staged.
* Git branch: shows the branches being worked on locally.
* Git merge: merges lines of development together. This command is typically used to combine changes made on two distinct branches. For example, a developer would merge when they want to combine changes from a feature branch into the master branch for deployment.
* Git pull:updates the local line of development with updates from its remote counterpart. Developers use this command if a teammate has made commits to a branch on a remote, and they would like to reflect those changes in their local environment.
* Git push: updates the remote repository with any commits made locally to a branch.

**Git Checkout:**

In Git terms, a "checkout" is the act of switching between different versions of a target entity. The git checkout command operates upon three distinct entities: files, commits, and branches. In addition to the definition of "checkout" the phrase "checking out" is commonly used to imply the act of executing the git checkout command.

# Git Branching

Nearly every VCS has some form of branching support. Branching means you diverge from the main line of development and continue to do work without messing with that main line. In many VCS tools, this is a somewhat expensive process, often requiring you to create a new copy of your source code directory, which can take a long time for large projects.