**GIT**

*(Git is one of the most popular types of Source Code Management and Distributed Version Control System)*

* Git was created in 2005 by the incredible software engineer Linus Torvalds, who also create the Linux kernel
* It is free and open source
* Allow multiple collaborators to work on a project at the same time
* It is used track changes to your code
* It maintain an organized code history

Installing Git on Your Machine

* You can download Git for free from the following website: [https://www.git-scm.com/](https://git-scm.com/)

GIT vs GITHUB Difference

* Git is a version control system that allows developers to track changes in their code. GitHub is a web-based hosting service for git repositories. In simple terms, you can use git without Github, but you cannot use GitHub without Git

Setting up a Repository

Once you have installed git, you first want to set up some basic information. We noted that git stores the author of every change, and this means you have to provide your identity. If you try the steps below before doing this, git will insist you do. Usually, it is enough to set a single identity globally, for all your git use. You do this using

1 .git config −−global user . name ” John Doe”

2 .git config −−global user . email johndoe@example . com

However, you can use several different email addresses, for example for work and for personal projects. In this case, after “git init” but before anything else, you should

1 .git config user . name ” John Doe”

2 .git config user . email johndoe@example . com

without the global flag.

Now before you can do anything else, you have to set up a git repository. You can do this in an empty directory or one already containing files. Be careful if this directory isn’t at the bottom of your directory tree as any subdirectories will also be included. Simply type

* git init

Now you can add files to the repo. You usually do this in two steps. First you add, or stage the change, that is get things ready, and then you commit. You can add multiple files, or parts of files, before carrying on.

1.git add src/

2.git commit

The second line results in a text editor opening to allow you to specify the “commit message” to explain what and why you are adding.

You now save and exit the editor, and git gives a short summary of what was committed. If you quit without saving the commit is aborted. The state of the files we committed has now been saved. Now we can make some changes to the files, and commit those. If we just try

* git commit

We can also see what changes have been made at any point using

* git status

There are two useful shortcuts: for a few files that have been previously added so are known to git, we can explicitly commit them, without an add step like

* git commit file1 .txt file2 . t x t

or we can commit everything which is changed using

* git commit −a

We can see all of the commits we have made using the log.

* git log

Branching

If you are working on several things at once, you may find branches useful. These are versions of code that git keeps separate for you, so that changes to one branch do not affect another. Whenever you create a repository, a default “master” branch is created. Adds and commits are always on the current branch. The command

* git branch

will show the name of the branch you are on. You can create a new branch using

* git branch {name}

The branch is based on the last commit. The branch command doesn’t move you to the new branch. You do this using

* git checkout {name}

You will get a message, usually Switched to branch ’name’, or an error message. To create a branch and change to it in a single step, use

* git checkout −b {new\_branch\_name } { existing\_branch\_name }

where the existing branch name is optional. This is very useful when working with a branch from a remote server, for example. Checkout also lets you go back to some previous version of the code, and create a branch from there using

* git checkout −b {new\_branch\_name } {commit ID}

Pull and Push

When the copy of the code on the remote is updated, you will need to pull in those changes, with

* git pull

To upload your changes to the remote, you can push them, using

* git push

some Git commands

1. git –version

For Git version checking

2.git clone<repository URL>

Copies a repository from remote server to your local machine.

3.git add <file>

Adds changes in a specific file to staging area, preparing them to be commit.

4. git add

Adds changes in the current directory to staging area.

5.git status

Displays the status of changes as untracked, modified or staged.

6.git merge<branch>

Merges changes from the specific branch into the currently active branch.

7.git remote –v

Lists all remote operations associated with the local respository along with their URLs.

8.git remote add<name><URL>

Adds a new remote repository.

9.git remote rm<name>

Removes a remote repository.

10.git fetch

Fetches changes from a remote repository without merging them into the current branch.

11.git reset<file>

Unstages changes for a specific file.

12.git reset –hard

Discards all changes and resets the repository to the state of the last commit.

13.git diff

Shows the difference between working directory, staging area and last commit.

14.git tag

Lists, creates or deletes tags (labels) to mark specific points in history.