**Course: Software engineering Principles & Practices Code: 20CS44P**

**Week 9: Development**

**Session No. 5**

**Learn version control and configuration management with Git.**

What is Git?

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

* Tracking code changes
* Tracking who made changes
* Coding collaboration

What does Git do?

* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project

Working with Git

* Initialize Git on a folder, making it a **Repository**
* Git now creates a hidden folder to keep track of changes in that folder
* When a file is changed, added or deleted, it is considered **modified**
* You select the modified files you want to **Stage**
* The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files
* Git allows you to see the full history of every commit.
* You can revert back to any previous commit.
* Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

Why Git?

* Over 70% of developers use Git!
* Developers can work together from anywhere in the world.
* Developers can see the full history of the project.
* Developers can revert to earlier versions of a project.

What is GitHub?

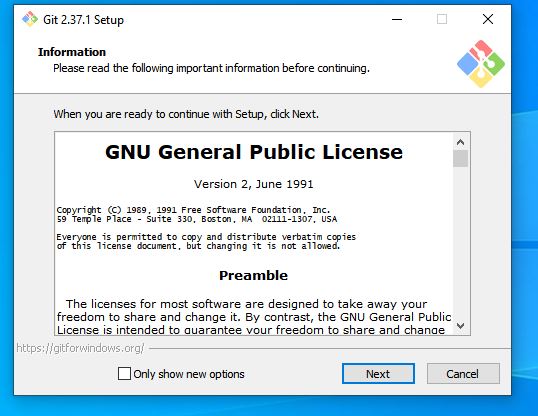
* Git is not the same as GitHub.
* GitHub makes tools that use Git.
* GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
* In this tutorial, we will focus on using Git with GitHub.

**How to install git?**

Let us now look at the various steps in the Git installation on Windows.

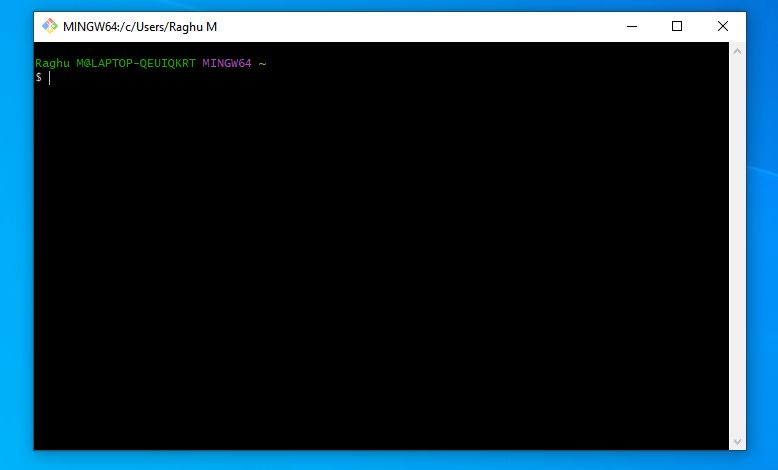
Step 1: Download the [latest version of Git](https://git-scm.com/downloads) and choose the 64/32 bit version. After the file is downloaded, install it in the system.

Link for download git: [**https://git-scm.com/downloads**](https://git-scm.com/downloads)

****

Step 2: To start using Git, we are first going to open up our Command shell.

For Windows, you can use Git bash, which comes included in Git for Windows. For Mac and Linux you can use the built-in terminal.



Step 4: check the installed version of git.

Raghu M@LAPTOP-QEUIQKRT MINGW64 ~

$ git --version

git version 2.37.1.windows.1

Step 5: Configure Git

Now let Git know who you are. This is important for version control systems, as each Git commit uses this information:

Raghu M@LAPTOP-QEUIQKRT MINGW64 ~

$ git config --global user.name "raghum"

Raghu M@LAPTOP-QEUIQKRT MINGW64 ~

$ git config --global user.email "raghavendram.gptcse@gmail.com"

Raghu M@LAPTOP-QEUIQKRT MINGW64 ~

$

Step 6: change the working directory to your destination drive (optional step)

Raghu M@LAPTOP-QEUIQKRT MINGW64 ~

$ cd D:

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d

$

Step 7: Create folder of your files or source code. And change the path to working folder

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d

$ mkdir myapp

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d

$ cd myapp

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp

$

Step 8:Initialize Git

Once you have navigated to the correct folder, you can initialize Git on that folder:

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp

$ git init

Initialized empty Git repository in D:/myapp/.git/

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

## Step 9:Git Adding New Files

You just created your first local Git repo. But it is empty.

So let's add some files, or create a new file using your favourite text editor. Then save or move it to the folder you just created.

Or create files using touch command just like bellow

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ touch home.html

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

Let's go back to the terminal and list the files in our current working directory:

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ ls

home.html

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

ls will **list** the files in the directory. We can see that home.html is there.

Step 10:Git status Then we check the Git status and see if it is a part of our repo:

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

home.html

nothing added to commit but untracked files present (use "git add" to track)

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

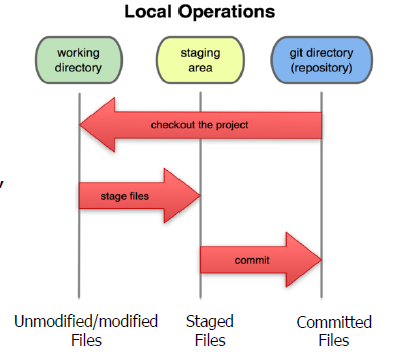
Now Git is **aware** of the file, but has not **added** it to our repository!

Files in your Git repository folder can be in one of 2 states:

* Tracked - files that Git knows about and are added to the repository
* Untracked - files that are in your working directory, but not added to the repository

 When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment.

Git Staging Environment



**Staged** files are files that are ready to be **committed** to the repository you are working on.For now, we are done working with home.html. So we can add it to the Staging Environment:

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git add home.html

The file should be **Staged**. Let's check the status::

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: home.html

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

Now the file has been added to the Staging Environment.

You can add all files in the current directory to the Staging Environment:

By using add all or add . Command.

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git add --all

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git add .

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

Git Commit

Since we have finished our work, we are ready move from stage to commit for our repo.

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

When we commit, we should **always** include a **message**.

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

**$ git commit -m "First release of app"**

[master (root-commit) 66700ff] First release of app

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 home.html

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$

Let's check the status::

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git status

On branch master

nothing to commit, working tree clean

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

Git Commit Log

To view the history of commits for a repository, you can use the log command:

Raghu M@LAPTOP-QEUIQKRT MINGW64 /d/myapp (master)

$ git log

commit 66700ff9d9ce69fbbd614adeea8f147643b80127 (**HEAD -> master**)

Author: raghum <raghavendram.gptcse@gmail.com>

Date: Mon Jul 18 20:41:08 2022 +0530

First release of app

Git Help

If you are having trouble remembering commands or options for commands, you can use Git help.

There are a couple of different ways you can use the help command in command line:

* git *command* -help -  See all the available options for the specific command
* git help --all -  See all possible commands