**STEVE A. TOMEY**

**ID#: 7080**

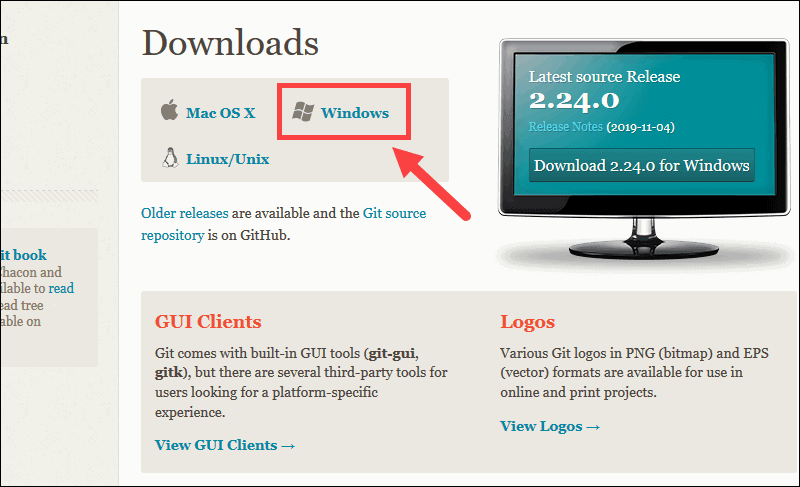
1. *Explain in your words the terms Git and Github.*

Ans: Git is a version control system that lets you manage and keep track of your source code history while GitHub is a cloud-based hosting service that lets you manage Git repositories.

1. *Explain the steps to download, install and configure Git on your PC.*

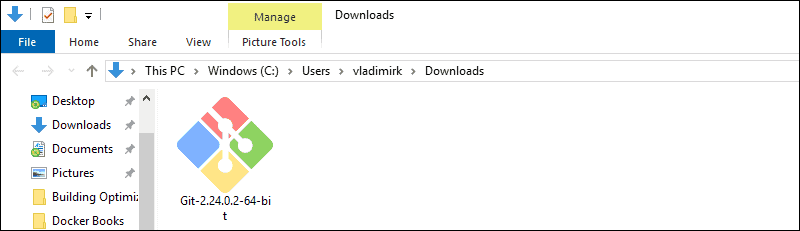
**ANS:**

* Browse to the official Git website: <https://git-scm.com/downloads>
* Click the download link for Windows and allow the download to complete.

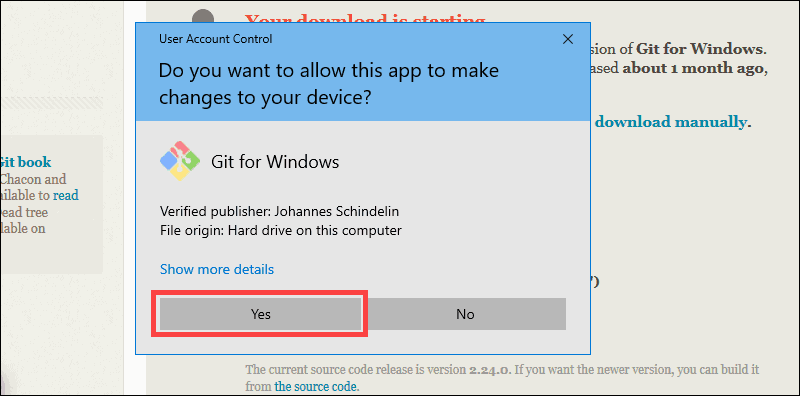


**Extract and Launch Git Installer**

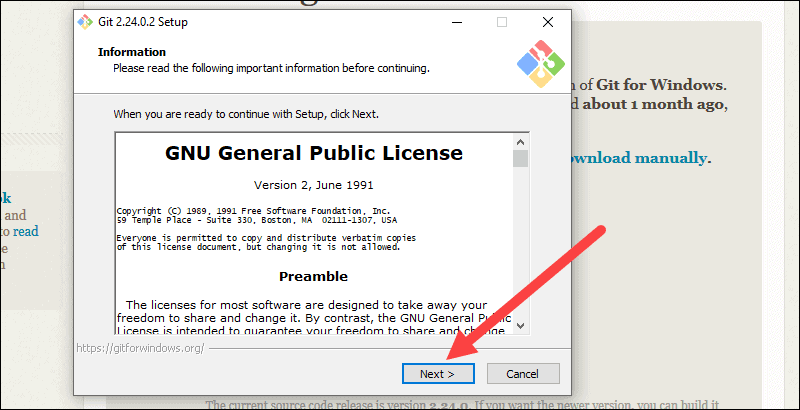
* Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.



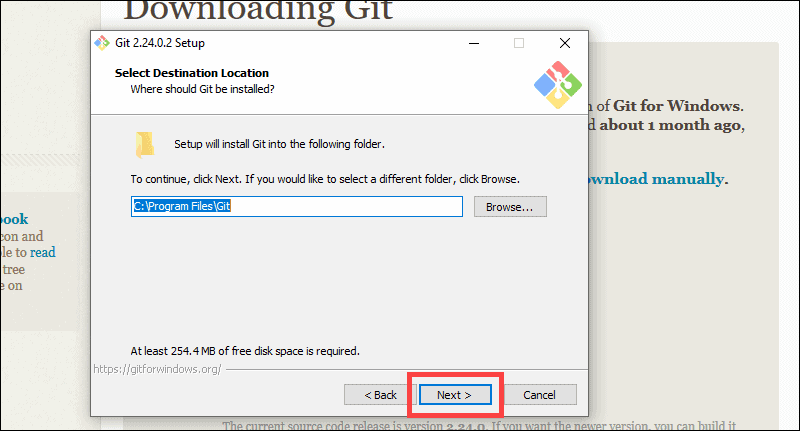
* Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.



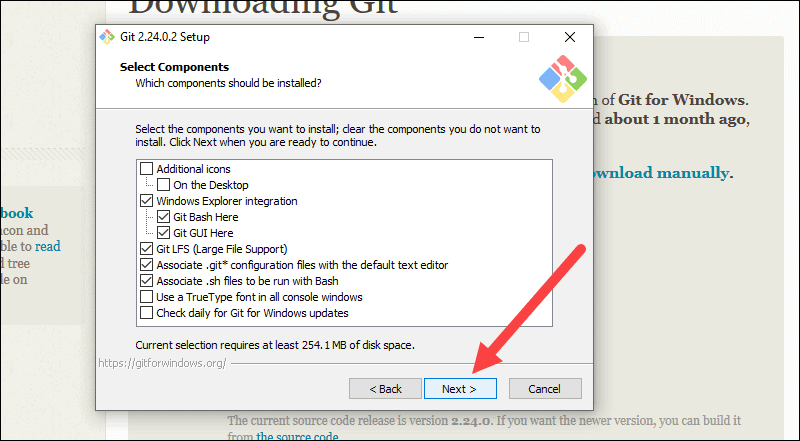
* Review the GNU General Public License, and when you’re ready to install, click **Next**.



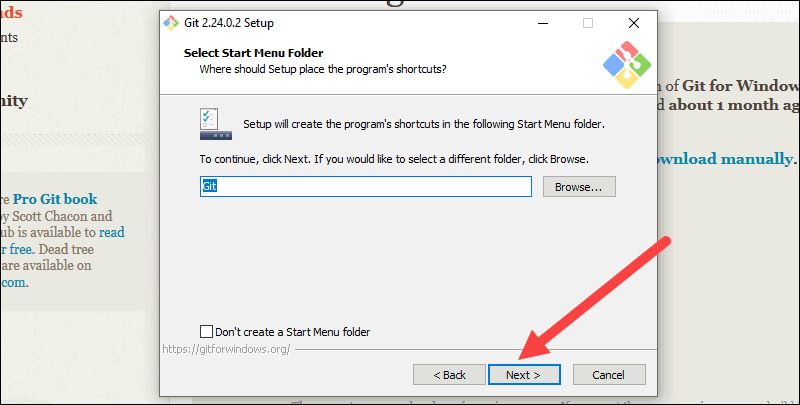
* The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.



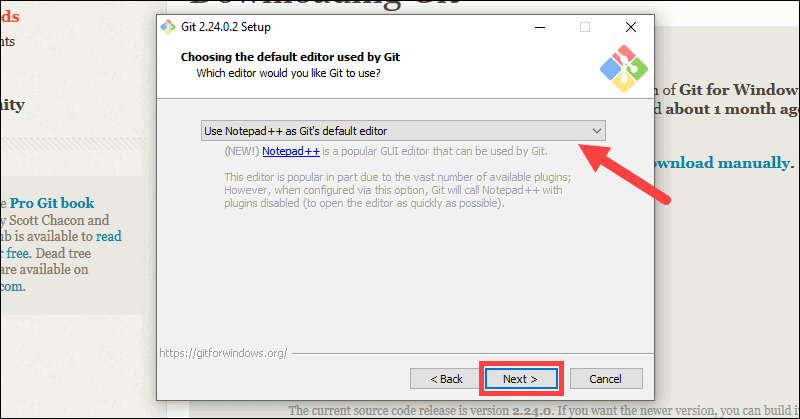
* A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.



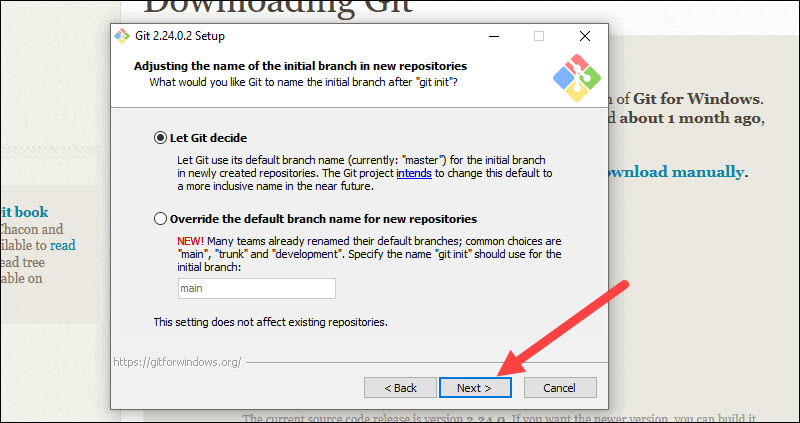
* The installer will offer to create a start menu folder. Simply click **Next**.



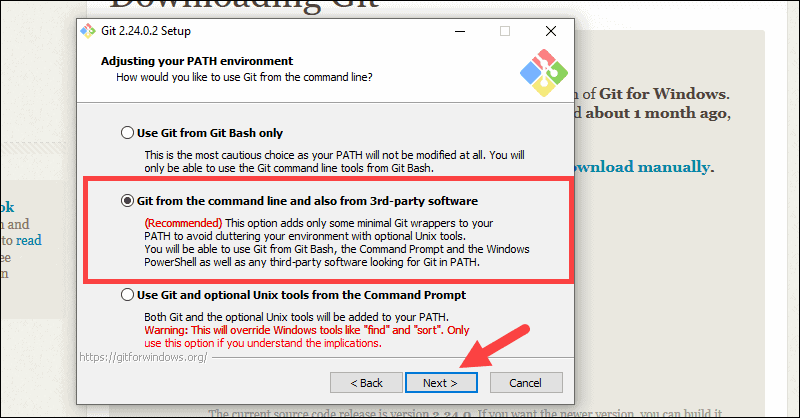
* Select a text editor you’d like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.



* The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click **Next.**

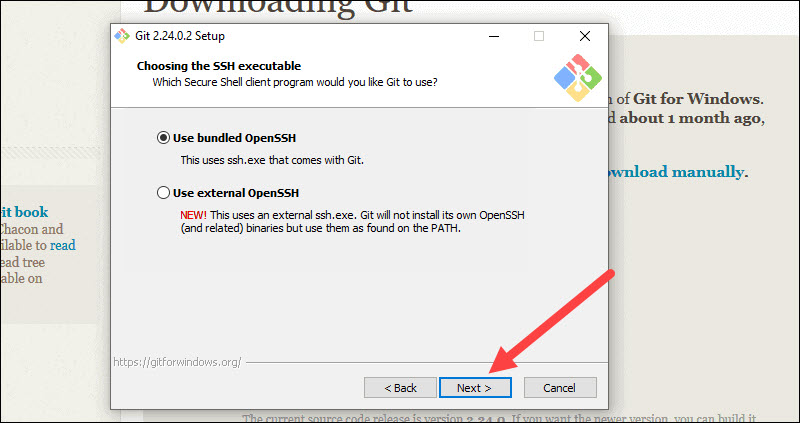


* This installation step allows you to change the **PATH environment**. The **PATH**is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.

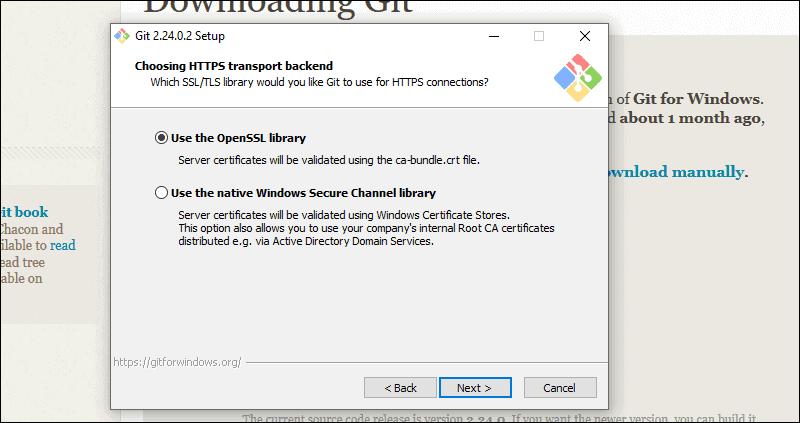


**Server Certificates, Line Endings and Terminal Emulators**

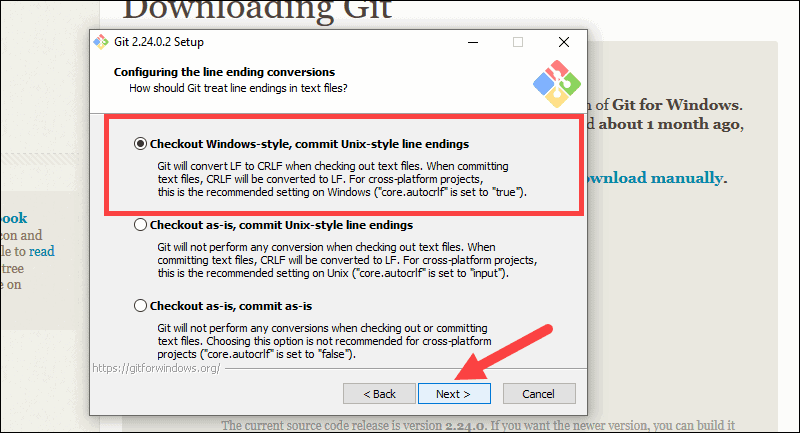
* The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**



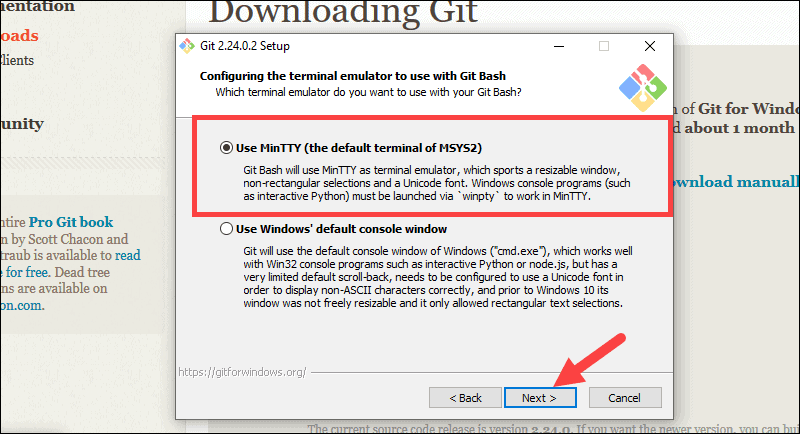
* The next option relates to server certificates. Most users should use the default. If you’re working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.



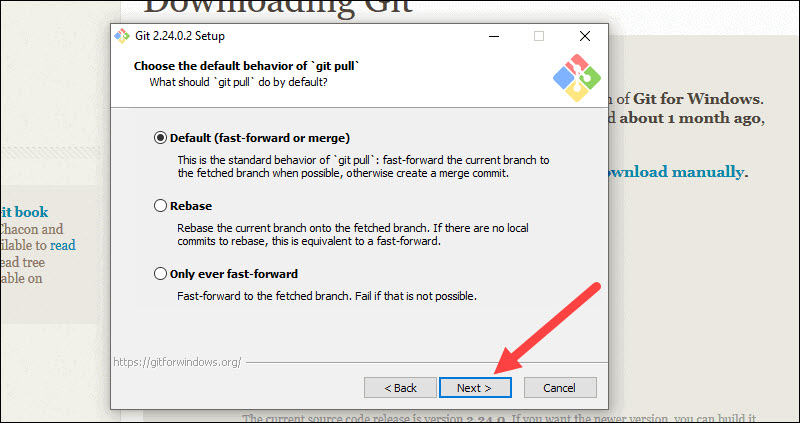
* The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.



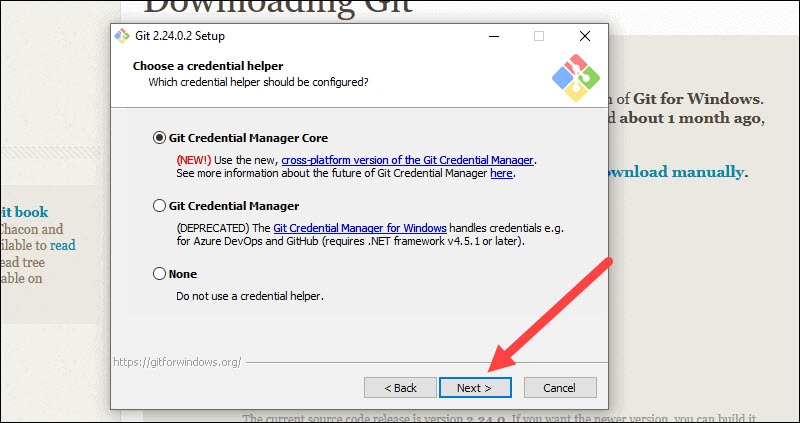
* Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.



* The installer now asks what the **git pull** command should do. The default option is recommended unless you specifically need to change its behavior. Click **Next**to continue with the installation.

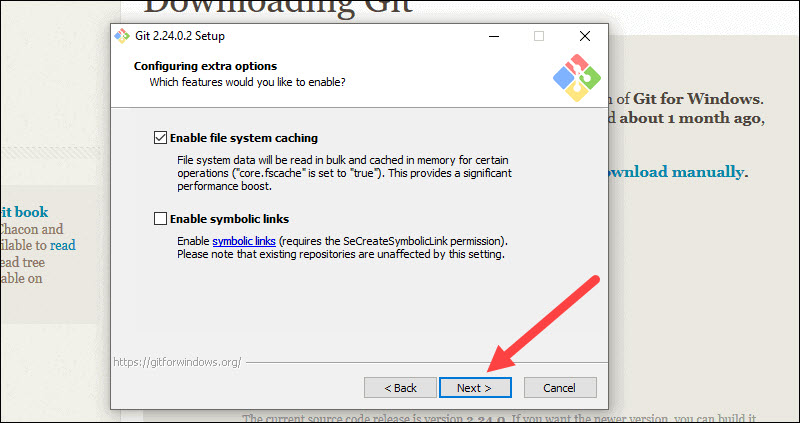


* Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.

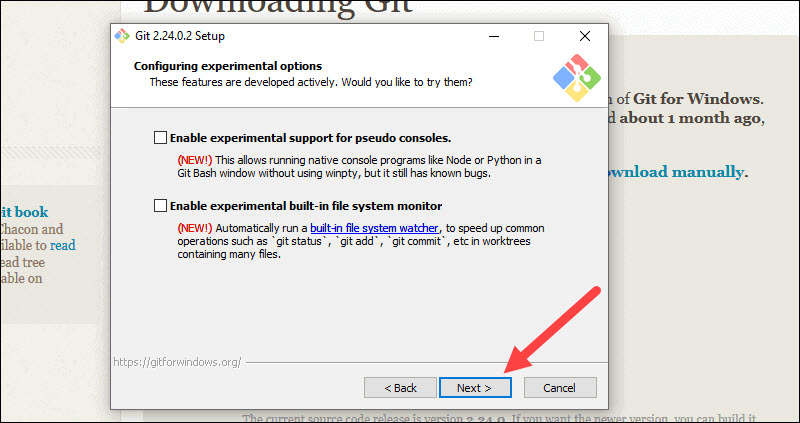


**Additional Customization Options**

* The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click **Next**.

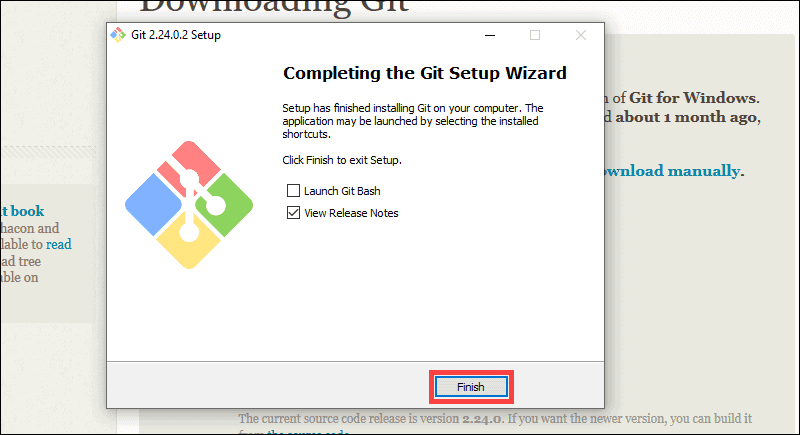


* Depending on the version of Git you’re installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click **Install**.



**Complete Git Installation Process**

* Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click **Finish**.



1. *Discuss the process to create and initialize a project in Git.*

**ANS:** Create a directory to contain the project.

* Go into the new directory.
* Type git init.
* Write some code.
* Type git add to add the files (see the [typical use page](https://kbroman.org/github_tutorial/pages/routine.html)).
* Type git commit.

The first file to create (and add and commit) is probably a ReadMe file, either as plain text or with[Markdown](https://daringfireball.net/projects/markdown/), describing the project.

Markdown allows you to add a bit of text markup, like [hyperlinks](https://en.wikipedia.org/wiki/Hyperlink), **bold**/*italics*, or to indicate code with a monospace font. Markdown is easily converted to html for viewing in a web browser, and GitHub will do this for you automatically.

1. *Explain the steps to pull and push from a local repository to a remote repository.*

**ANS:**

. git remote add origin https://github.com/myname/testsequence.git

. git branch -M master

. git push -u origin master

Let’s take this apart… After you have created a repository on a service in the above case Github, you need to add it as a target remote repository.

Line two is optional… But in this case, if I want to push my current master branch which was called main instead master, and the repository at the remote has master as the master branch, you want to do a force rename of the local *master*branch.

Lastly, push the branch to the remote… the -u option will automatically set the upstream branch for you.

1. *In your own words, explain the term push, pull, commit, stage, and branches in Git.*

**ANS:** Well, basically git commit puts your changes into your local repo, while git push sends your changes to the remote location.

The git pull is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows **while** *stage* is simply to prepare a file to its finely for a commit.

In Git, branches are a part of your everyday development process. Git branches are effectively **a pointer to a snapshot of your changes**. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.

