**What is Git?**

* Git is the most popular, open-source, widely used, and an example of distributed version control system (DVCS) used for handling the development of small and large projects in a more efficient and neat manner.
* It is most suitable when there are multiple people working on projects as a team and is used for tracking the project changes and efficiently supports the collaboration of the development process.
* With the help of the versioning system, the developer can identify who has made what changes and then run tests and fix bugs if any and then do necessary feature implementation. In case of any unforeseen circumstances, the code can be reverted to any of the previously working versions thereby saving huge efforts.

**What is a version control system?**

A VCS keeps track of the contributions of the developers working as a team on the projects. They maintain the history of code changes done and with project evolution, it gives an upper hand to the developers to introduce new code, fixes bugs, and run tests with confidence that their previously working copy could be restored at any moment in case things go wrong.

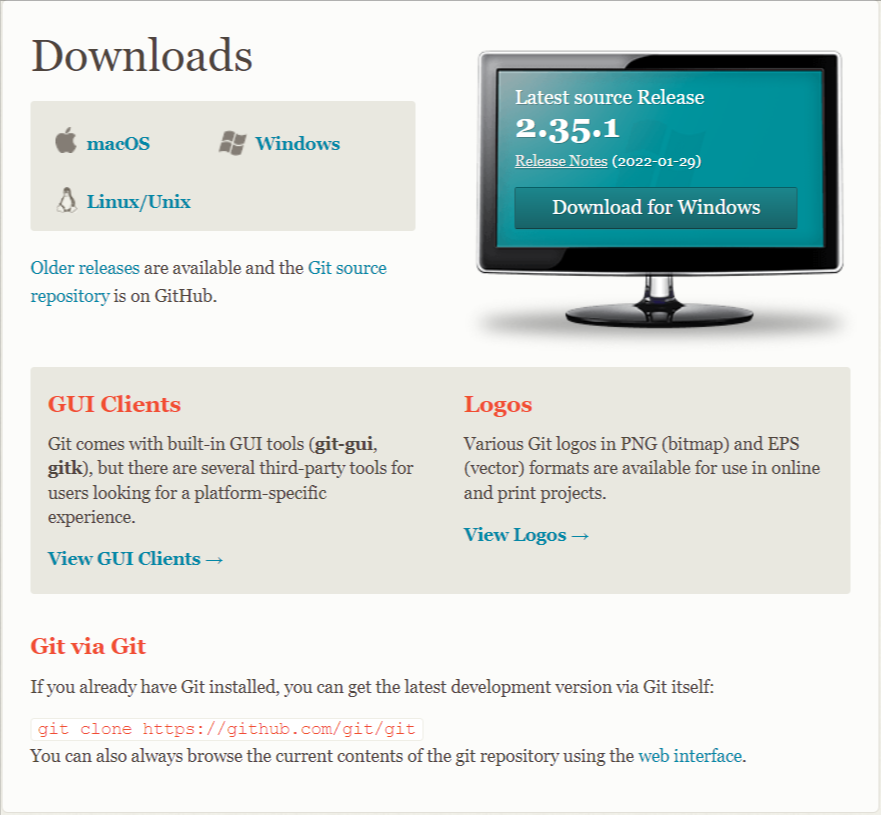
**What is a Git Repository?**

A repository is a file structure where git stores all the project-based files. Git can either stores the files on the local or the remote repository.

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Experiment No. 01** | |  |

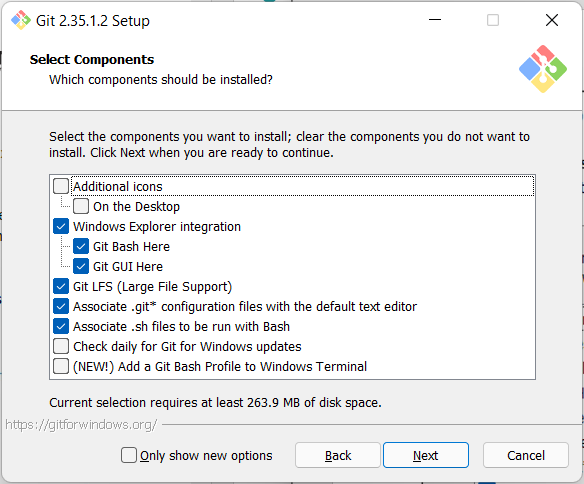
**Aim:** Setting up the git client.

Git Installation: Download the Git installation program (Windows, Mac, or Linux) from [Git - Downloads (git-scm.com).](http://git-scm.com/downloads)



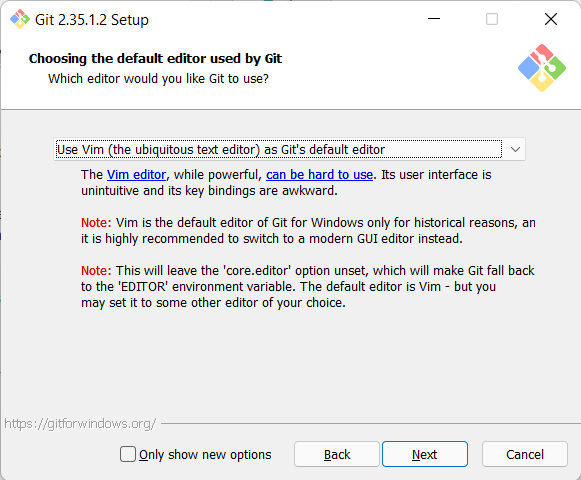
When running the installer, various screens appear (Windows screens shown). Generally, you can accept the default selections, except in the screens below where you do not want the default selections:

In the Select Components screen, make sure Windows Explorer Integration is selected as shown:



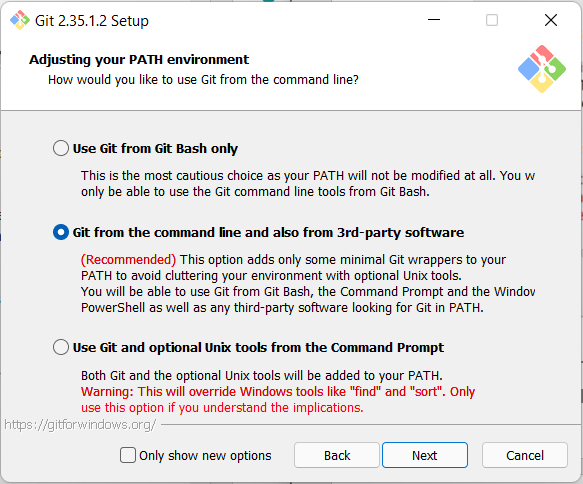
In choosing the default editor used by Git dialog, it is recommended that you

select default VIM editor- although there are better modern editors available. If you want you can also choose notepad++ or visual studio it depends on user’s prferences.

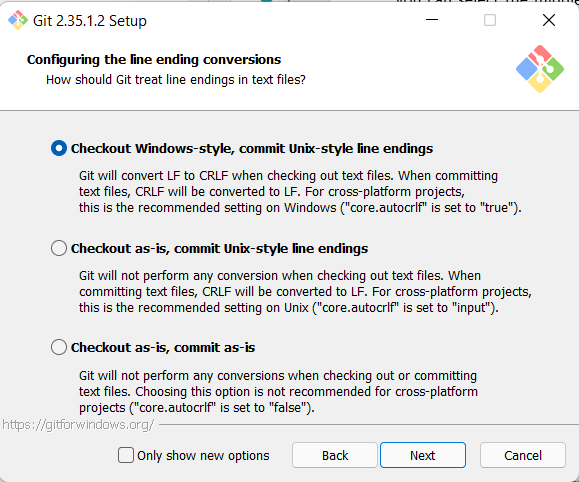


In the Adjusting your PATH screen, all three options are acceptable:

1. Use Git from Git Bash only: no integration, and no extra command in your command path.
2. Use Git from the windows Command Prompt: add flexibility – you can simply run git from a windows command prompt, and is often the setting for people in industry – but this does add some extra commands.
3. Use Git and optional Unix tools from the Windows Command Prompt: this is also a robust choice and useful if you like to use Unix like commands like grep.



In the Configuring the line ending screen, by default first option will be selected but if you want you can select the middle option (Checkout-as-is, commit Unix-style line endings). This helps migrate files towards the Unix-style (LF) terminators that most modern IDE’s and editors support. The Windows convention (CR-LF line termination) is only important for Notepad.



After this just click on next until install option comes and then press install.

**Configuring Git to ignore certain files:**

**This part is extra important and required so that your repository does not get cluttered with garbage files.**

By default, Git tracks allfiles in a project. Typically, this is notwhat you want; rather, you want Git to ignore certain files such as .bakfiles created by an editor or .classfiles created by the Java compiler. To have Git automatically ignore particular files, create a file named .gitignore( note that the filename begins with a dot) in the C:\users\namefolder (where name is your MSOE login name).

**NOTE:** The .gitignore file must NOT have any file extension (e.g. .txt). Windows normally tries to place a file extension (.txt) on a file you create from File Explorer - and then it (by default) HIDES the file extension. To avoid this, create the file from within a useful editor (e.g. Notepad++ or UltraEdit) and save the file without a file extension).

Edit this file and add the lines below (just copy/paste them from this screen); these are patterns for files to be ignored (taken from examples provided at https://github.com/github/gitignore.)

|  |
| --- |
| **#Lines (like this one) that begin with # are comments; all other lines are rules**    **# common build products to be ignored at MSOE**  **\*.o**  **\*.obj**  **\*.class \*.exe**    **# common IDE-generated files and folders to ignore workspace.xml bin / out**  **/**  **.classpath**  **# uncomment following for courses in which Eclipse .project files are not checked in # .project**  **#ignore automatically generated files created by some common applications, operating systems**  **\*.bak**  **\*.log**  **\*.ldb**  **~\***  **.DS\_Store\***  **.\_\* Thumbs.d b**    **# Any files you do not want to ignore must be specified starting with ! # For example, if you didn't want to ignore .classpath, you'd uncomment the following rule: # !.classpath** |

Note: You can always edit this file and add additional patterns for other types of files you might want to ignore. Note that you can also have a

.gitignore files in any folder naming additional files to ignore. This is useful for projectspecific build products.

Once Git is installed, there is some remaining custom configuration you must do. Follow the steps below:

1. From within File Explorer, right-click on any folder. A context menu appears containing the commands " Git Bash here" and "Git GUI here". These commands permit you to launch either Git client. For now, select Git Bash here.

1. Enter the command git config --global user.Email "name@msoe.edu"

This links your Git activity to your email address. Without this, your commits will often show up as "unknown login". Replace name with your own MSOE email name.

c. Enter the command git config --global user.name "Your Name"

Git uses this to log your activity. Replace "Your Name" by your actual first and last name.

d. Enter the command git config --global push.default simple

This ensures that all pushes go back to the branch from which they were pulled. Otherwise pushes will go to the master branch, forcing a merge.