Git:

Git is a free, open source distributed version control system tool designed to handle everything from small to very large projects with speed and efficiency.

**There are two types of VCS:**

* Centralized Version Control System (CVCS)
* Distributed Version Control System (DVCS)

**Centralized VCS**

Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. It works on a single repository to which users can directly access a central server.

Please refer to the diagram below to get a better idea of CVCS:



The repository in the above diagram indicates a central server that could be local or remote which is directly connected to each of the programmer’s workstation.

Every programmer can extract or **update** their workstations with the data present in the repository or can make changes to the data or **commit** in the repository. Every operation is performed directly on the repository.

Even though it seems pretty convenient to maintain a single repository, it has some major drawbacks. Some of them are:

* It is not locally available; meaning you always need to be connected to a network to perform any action.
* Since everything is centralized, in any case of the central server getting crashed or corrupted will result in losing the entire data of the project.

**Distributed VCS**

These systems do not necessarily rely on a central server to store all the versions of a project file.

In Distributed VCS, every contributor has a local copy or “clone” of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.

You will understand it better by referring to the diagram below:



As you can see in the above diagram, every programmer maintains a local repository on its own, which is actually the copy or clone of the central repository on their hard drive. They can commit and update their local repository without any interference.

They can update their local repositories with new data from the central server by an operation called “**pull**” and affect changes to the main repository by an operation called “**push**” from their local repository.

The act of cloning an entire repository into your workstation to get a local repository gives you the following advantages:

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

**[DevOps Certification Training Course](https://www.edureka.co/devops-certification-training" \t "_blank)**

[Explore Curriculum](https://www.edureka.co/devops-certification-training" \t "_blank)

* All operations (except push & pull) are very fast because the tool only needs to access the hard drive, not a remote server. Hence, you do not always need an internet connection.
* Committing new change-sets can be done locally without manipulating the data on the main repository. Once you have a group of change-sets ready, you can push them all at once.
* Since every contributor has a full copy of the project repository, they can share changes with one another if they want to get some feedback before affecting changes in the main repository.
* If the central server gets crashed at any point of time, the lost data can be easily recovered from any one of the contributor’s local repositories.

## ****What Is Git?****

Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software.

Git provides with all the Distributed VCS facilities to the user that was mentioned earlier. Git repositories are very easy to find and access. You will know how flexible and compatible Git is with your system when you go through the features mentioned below:

## ****What is Git – Features Of Git****

  
  
**Free and open source:**  
Git is released under GPL’s (General Public License) open source license. You don’t need to purchase Git. It is absolutely free. And since it is open source, you can modify the source code as per your requirement.

  
**Speed:**  
Since you do not have to connect to any network for performing all operations, it completes all the tasks really fast. Performance tests done by Mozilla showed it was an order of magnitude faster than other version control systems. Fetching version history from a locally stored repository can be one hundred times faster than fetching it from the remote server. The core part of Git is written in C, which avoids runtime overheads associated with other high level languages.



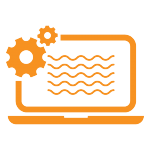
**Scalable:**  
Git is very scalable. So, if in future , the number of collaborators increase Git can easily handle this change. Though Git represents an entire repository, the data stored on the client’s side is very small as Git compresses all the huge data through a lossless compression technique.

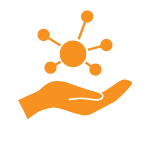


**Reliable:**  
Since every contributor has its own local repository, on the events of a system crash, the lost data can be recovered from any of the local repositories. You will always have a backup of all your files.

**Secure:**  
Git uses the ***SHA1*** (Secure Hash Function) to name and identify objects within its repository. Every file and commit is check-summed and retrieved by its checksum at the time of checkout. The Git history is stored in such a way that the ID of a particular version (a commit in Git terms) depends upon the complete development history leading up to that commit. Once it is published, it is not possible to change the old versions without it being noticed.

**Economical:**  
In case of CVCS, the central server needs to be powerful enough to serve requests of the  entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware  limitations of the server can be a performance bottleneck. In case of DVCS, developers don’t  interact with the server unless they need to push or pull changes. All the heavy lifting  happens on the client side, so the server hardware can be very simple indeed.

**Supports non-linear development:**  
Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history. A core assumption in Git is that a change will be merged more often than it is written, as it is passed around various reviewers. Branches in Git are very lightweight. A branch in Git is only a reference to a single commit. With its parental commits, the full branch structure can be constructed.

 **Easy Branching:**  
Branch management with Git is very simple. It takes only few seconds to create, delete, and merge branches. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the master branch always contains production-quality code.

# Install Git – Git Installation On Windows And CentOS:

## ****Install Git On Windows****

**Step 1**:

To download the latest version of Git, click on the link below:

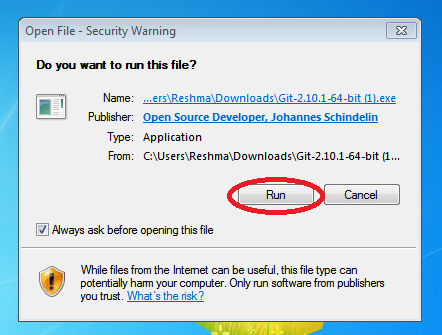
[***Download Git for Windows***](https://git-scm.com/download/win/)

[Git - Downloading Package (git-scm.com)](https://git-scm.com/download/win/)

Great! Your file is being downloaded.

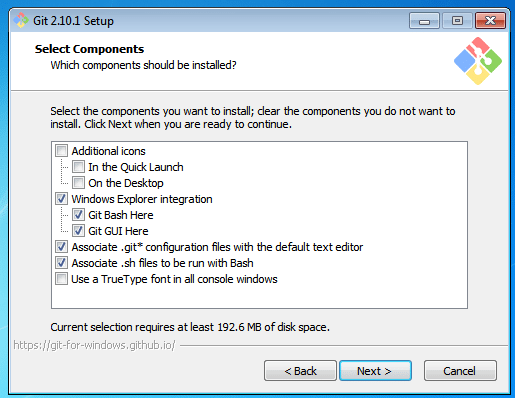
**Step 2:**

After your download is complete, **Run** the .exe file in your system.



**Step 3:**

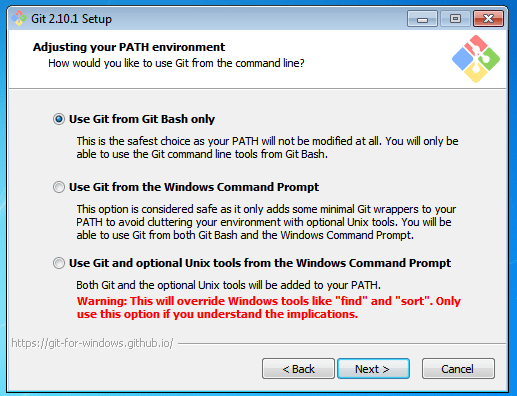
After you have pressed the **Run** button and agreed to the license, you will find a window prompt to select components to be installed.



After you have made selection of your desired components, click on **Next>**.

**Step 4:**

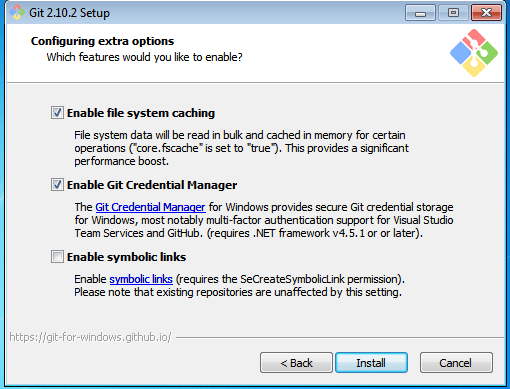
The next prompt window will let you choose the adjustment of your path environment. This is where you decide how do you want to use Git.



You can select any of the three options according to your needs. But for beginners, I recommend using **Use Git From Git Bash Only**

**Step 5:**

The next step is to choose features for your Git. You get three options and you can choose any of them, all of them or none of them as per your needs. Let me tell you what these features are:



The first is the option to **Enable file system caching**.

Caching is enabled through Cache manager, which operates continuously while Windows is running. File data in the system file cache is written to the disk at intervals determined by the operating system, and the memory previously used by that file data is freed.

The second option is to enable **Git Credential Manager**.

The **Git Credential Manager** for Windows (GCM) is a credential helper for Git. It securely stores your credentials in the Windows CM so that you only need to enter them once for each remote repository you access. All future Git commands will reuse the existing credentials.

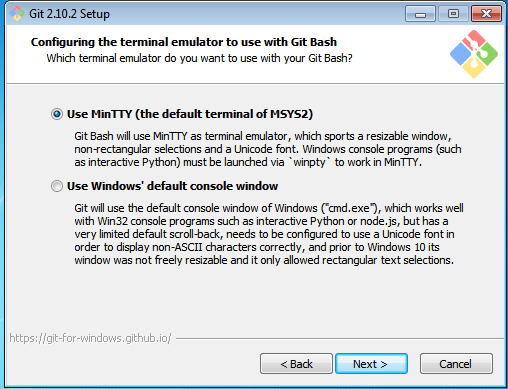
The third option is to **Enable symbolic links**.

Symbolic links or symlinks are nothing but advanced shortcuts. You can create symbolic links for each individual file or folder, and these will appear like they are stored in the folder with symbolic link.

I have selected the first two features only.

**Step 6:**

Choose your terminal.



You can choose one from the options.

The default terminal of MYSYS2 which is a collection of GNU utilities like bash, make, gawk and grep to allow building of applications and programs which depend on traditionally UNIX tools to be present.

Or you can choose the window’s default console window (cmd.exe).

**Step 7:**

Now you have got all you need. Select **Launch Git Bash** and click on **Finish**.

