Git Workshop

- *Made a change to code, realised it was a mistake and wanted to revert back?*
- *Lost code or had a backup that was too old?*
- *Had to maintain multiple versions of a product?*
- *Wanted to see the difference between two (or more) versions of your code?*
- *Wanted to prove that a particular change broke or fixed a piece of code?*
- *Wanted to review the history of some code?*
- *Wanted to submit a change to someone else's code?*

- *Wanted to share your code, or let other people work on your code?*
- *Wanted to see how much work is being done, and where, when and by whom?*
- *Wanted to experiment with a new feature without interfering with working code?*

Agenda

- What is Version Control System?
- Different Types of VCS
- Centralized Version Control System (CVCS)
- Distributed Version Control System (DVCS)
- Introduction to Git
- Installation of Git
- Git Commands
- GitHub- Remote Repository
- Eclipse IDE and Git-GitHub

Version Control System

A tool that manages and tracks different versions of software or other content is referred to generically as a version control system (VCS).

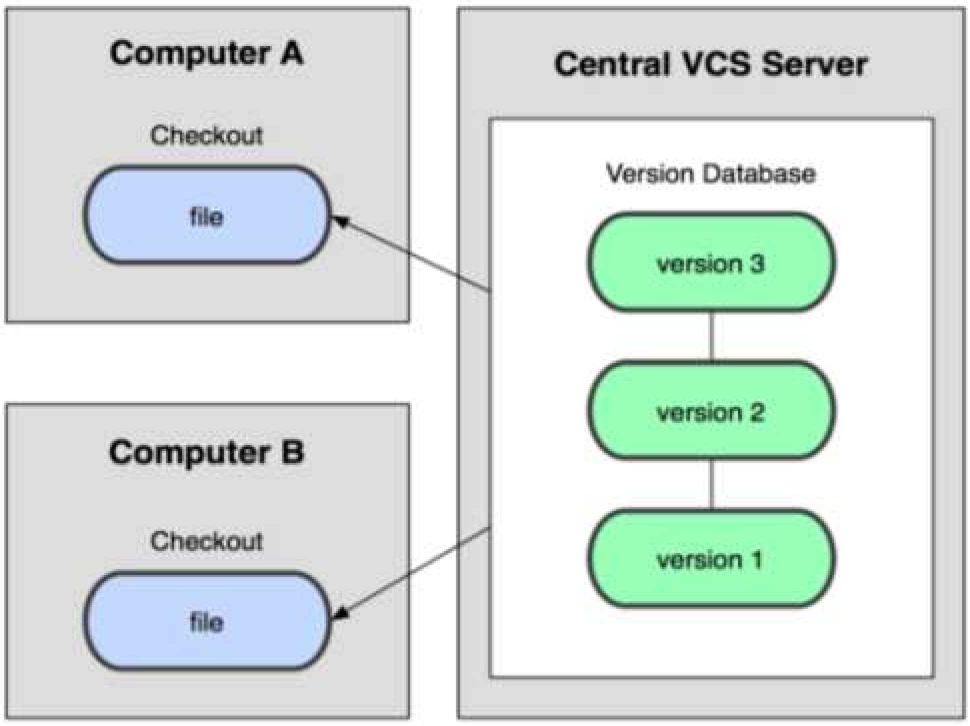
- A version control system (VCS) allows to track the history of a collection of files.
- These versions are stored in a specific place, typically called a repository.
- It is is a software that helps software developers to work together and maintain a complete history of their work.
- Allows developers to work simultaneously
- Does not allow overwriting each other’s changes.
- Maintains a history of every version

Types of Version Control System

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

Centralized Version Control System

- A centralized version control system provides a server software component which stores and manages the different versions of the files.
- Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration.
- "Committing" a change simply means recording the change in the central system. Other programmers can then see this change.

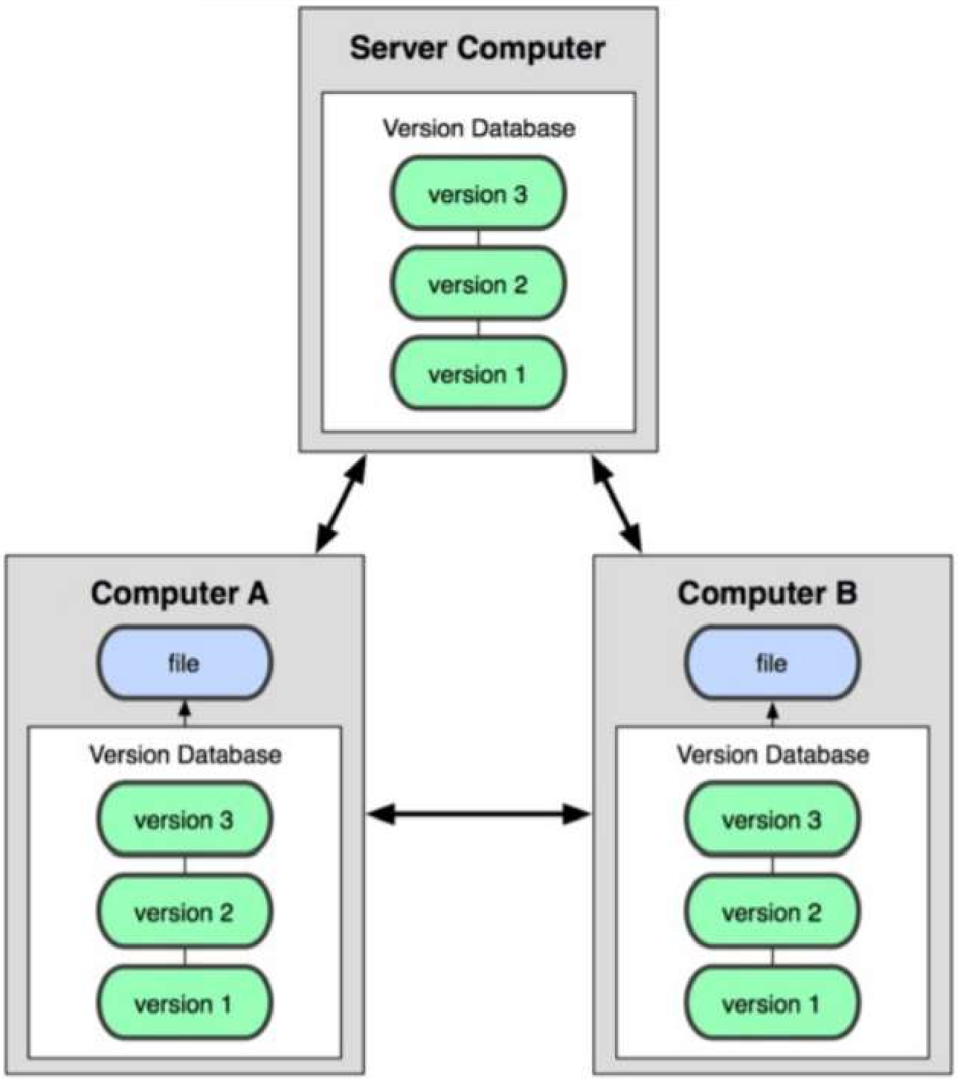


Ref - https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

- Centralized version control systems are based on the idea that there is a single "central" copy of the project somewhere (probably on a server), and programmers will "commit" their changes to this central copy.
- But the major drawback of CVCS is its single point of failure, i.e., failure of the central server.

- Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all.
- And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then will have to lose the entire history of the project.

Distributed Version Control System



Ref- https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

- It is a form of version control in which the complete codebase, including its full history, is mirrored on every developer's computer.

- This enables automatic management branching and merging, speeds up of most operations (except pushing and pulling), improves the ability to work offline, and does not rely on a single location for backups.
- In a distributed version control system, each user has a complete local copy of a repository on the individual computer.
- This copying process is typically called cloning and the resulting repository can be referred to as a clone.
- Every clone contains the full history of the collection of files and a cloned repository has the same functionality as the original repository.
- Git falls under distributed version control system.

Git

- Git is a distributed version-control system for tracking changes in source code during software development.
- It is designed for coordinating work among programmers, but it can be used to track changes in any set of files.
- The Linux kernel is an open source software project of large scope. For most of the lifetime of the Linux kernel maintenance (1991–2002), changes to the software were passed around as patches and archived files.
- In 2002, the Linux kernel project began using a proprietary DVCS called BitKeeper.
- In 2005, the relationship between the community that developed the Linux kernel and the commercial company that developed BitKeeper broke down, and the tool's free-of-charge status was revoked.
- This prompted the Linux development community (and Linus Torvalds, the creator of Linux) to develop their own tool based on some of the lessons

they learned while using BitKeeper. Some of the goals of the new system were as follows:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)

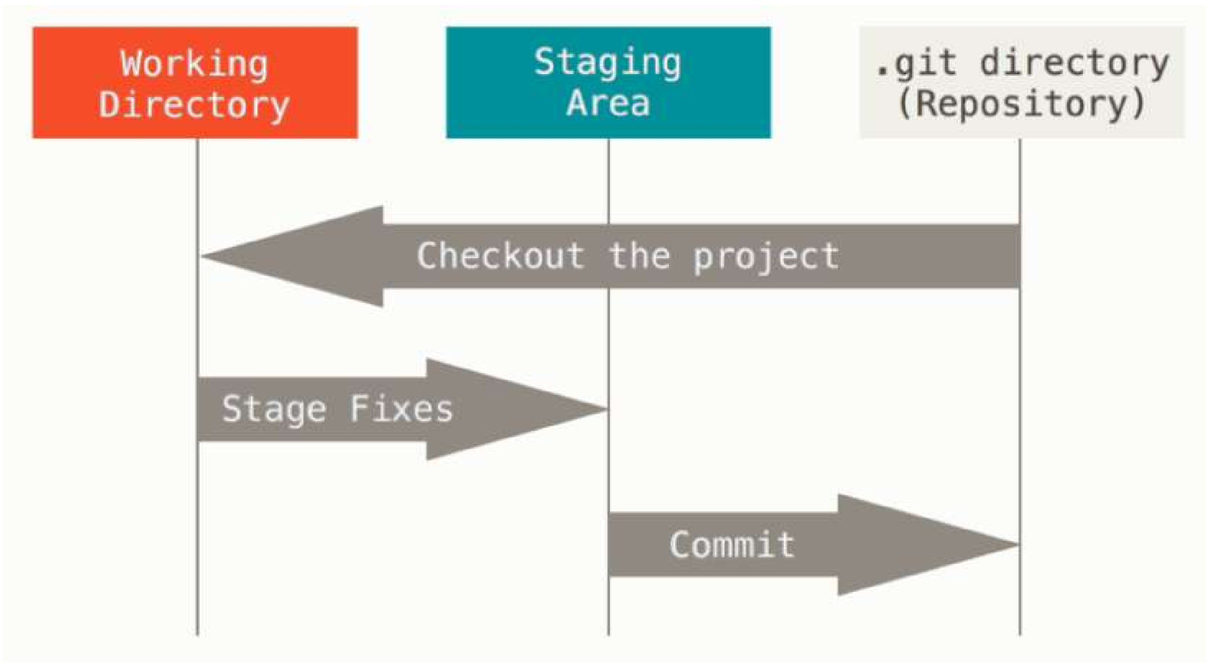
Ref- https://git-scm.com/downloads/logos

- Fully distributed

The Three States

Git has three main states that files can reside in: modified, staged, and committed:

- Modified means that user have changed the file but have not committed it to the database yet.
- Staged means that user have marked a modified file in its current version to go into next commit snapshot.
- Committed means that the data is safely stored in the local database.



Ref- https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

The basic Git workflow goes something like this:

1. Modify files in the working tree.
2. Selectively stage just those changes that need to be the part of next commit, which adds only those changes to the staging area.

3. Can do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to Git directory.

Advantages of Git

Free and open source-

- Git is released under GPL's open source license.
- It is available freely over the internet. Anyone can use Git to manage propriety projects without paying a single penny.
- As it is an open source, can download its source code and also perform changes according to the requirements.

Fast and small-

- As most of the operations are performed locally, it gives a huge benefit in terms of speed.
- Git does not rely on the central server; that is why, there is no need to interact with the remote server for every operation performed.
- Though Git mirrors entire repository, the size of the data on the client side is small. This illustrates the efficiency of Git at compressing and storing data on the client side.

Implicit backup-

- The chances of losing data are very rare when there are multiple copies of it. Data present on any client side mirrors the repository, hence it can be used in the event of a crash or disk corruption.

Security-

- Git uses a common cryptographic hash function called secure hash function (SHA1), to name and identify objects within its database.
- Every file and commit is check-summed and retrieved by its checksum at the time of checkout. It implies that it is impossible to change file, date,

and commit message and any other data from the Git database without knowing Git.

No need of powerful hardware-

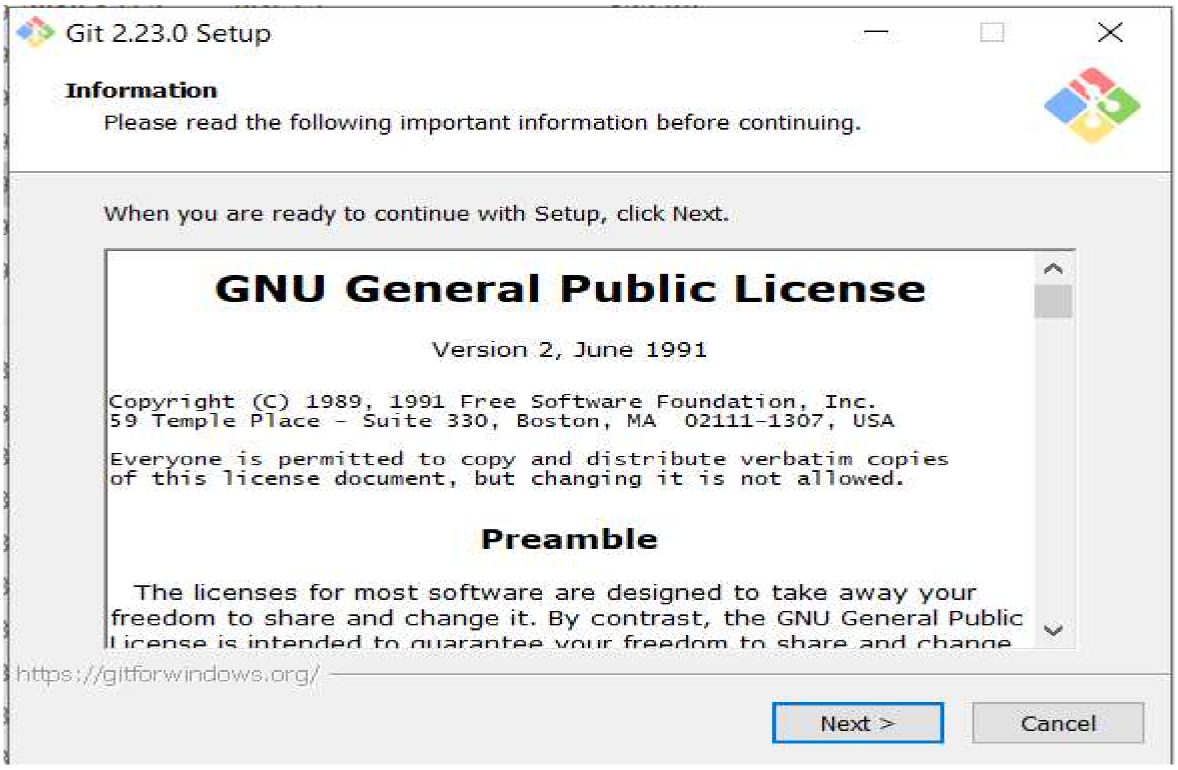
- In case of CVCS, the central server needs to be powerful enough to serve requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware limitations of the server can be a performance bottleneck.
- In case of DVCS, developers don't interact with the server 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.

Easier branching-

- Git is very simple. It takes only a few seconds to create, delete, and merge branches.

Git Installation

Link - https://git-scm.com/downloads



Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

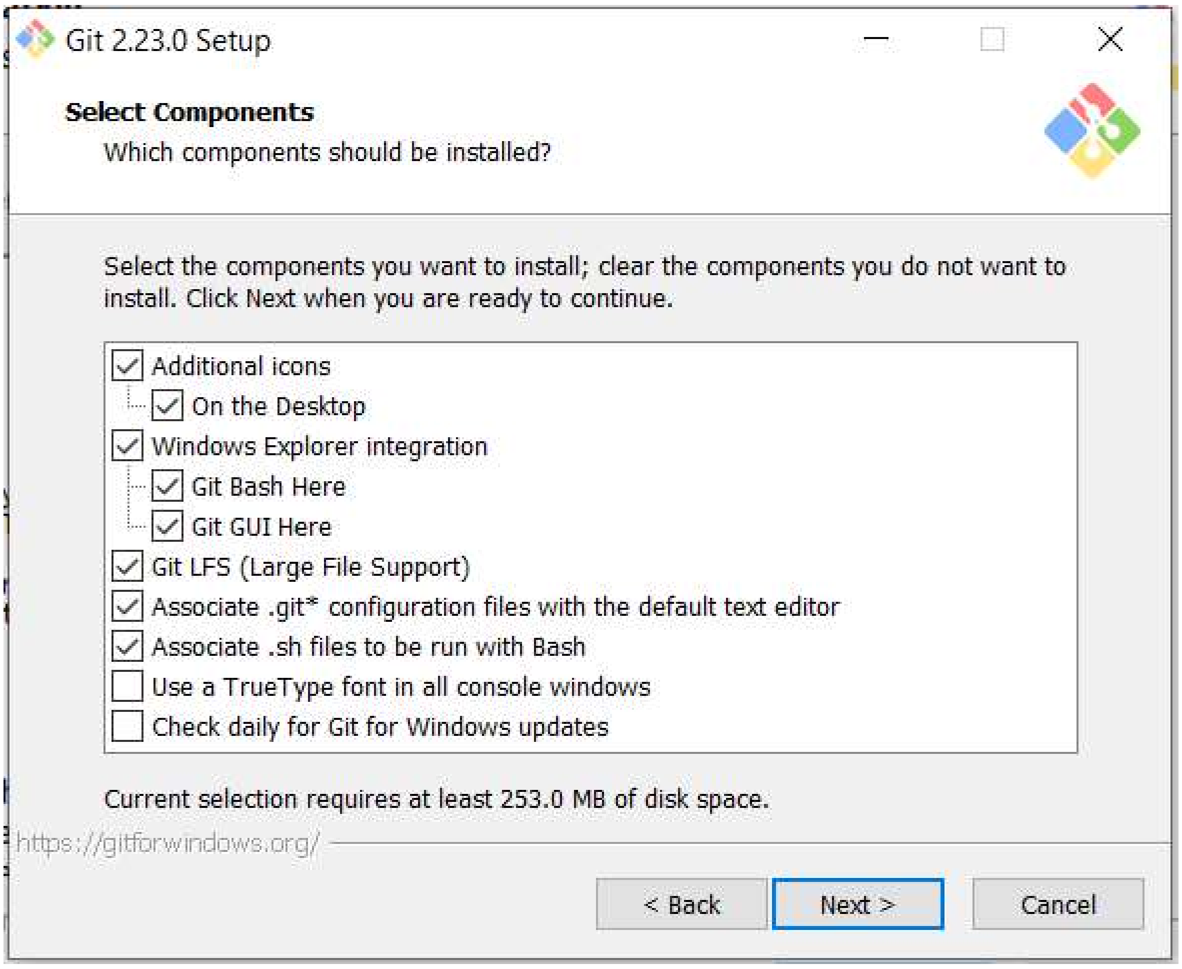
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://gitforwindows.org/


Git 2.23.0 Setup

Select Components

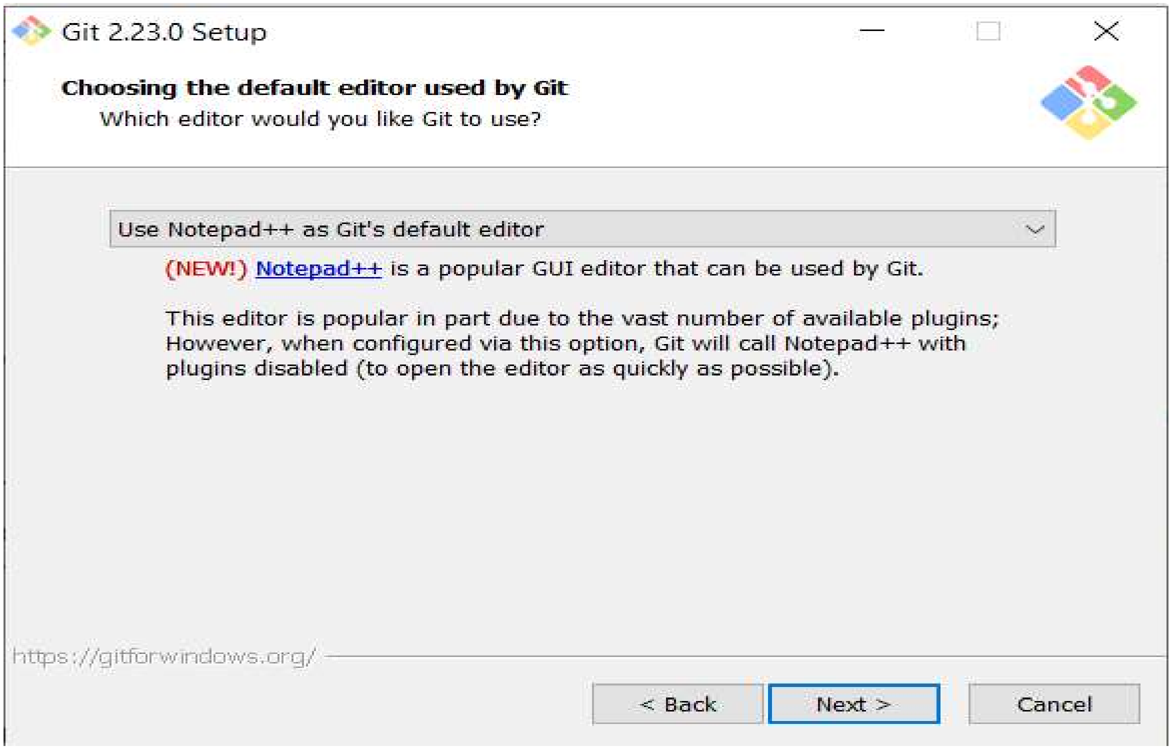
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Additional icons
  - [✓] On the Desktop
- [✓] Windows Explorer integration
  - [✓] Git Bash Here
  - [✓] Git GUI Here
- [✓] Git LFS (Large File Support)
- [✓] Associate .git* configuration files with the default text editor
- [✓] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows
- [ ] Check daily for Git for Windows updates

Current selection requires at least 253.0 MB of disk space.

https://gitforwindows.org/


Git 2.23.0 Setup

Choosing the default editor used by Git

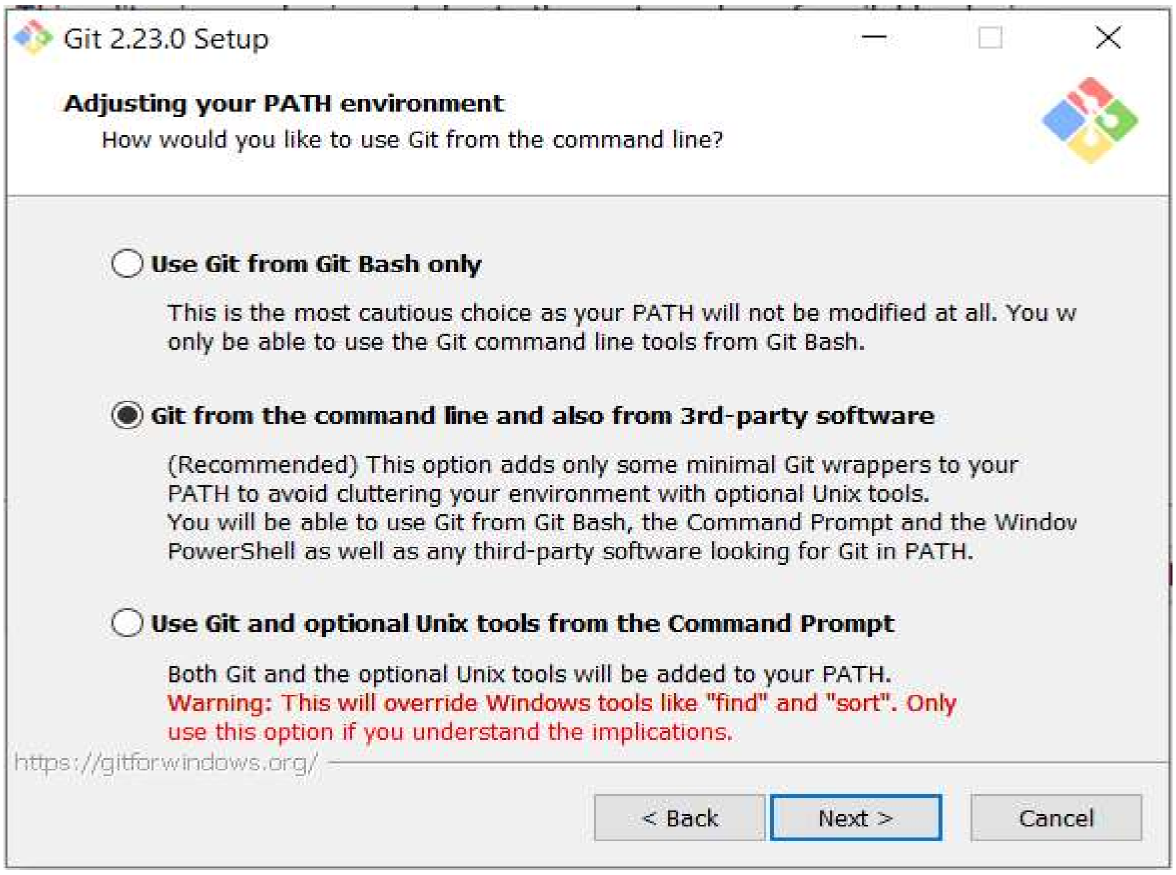
Which editor would you like Git to use?

Use Notepad++ as Git's default editor

(NEW!) Notepad++ is a popular GUI editor that can be used by Git.

This editor is popular in part due to the vast number of available plugins; However, when configured via this option, Git will call Notepad++ with plugins disabled (to open the editor as quickly as possible).

https://gitforwindows.org/


Git 2.23.0 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the most cautious choice as your PATH will not be modified at all. You w
only be able to use the Git command line tools from Git Bash.

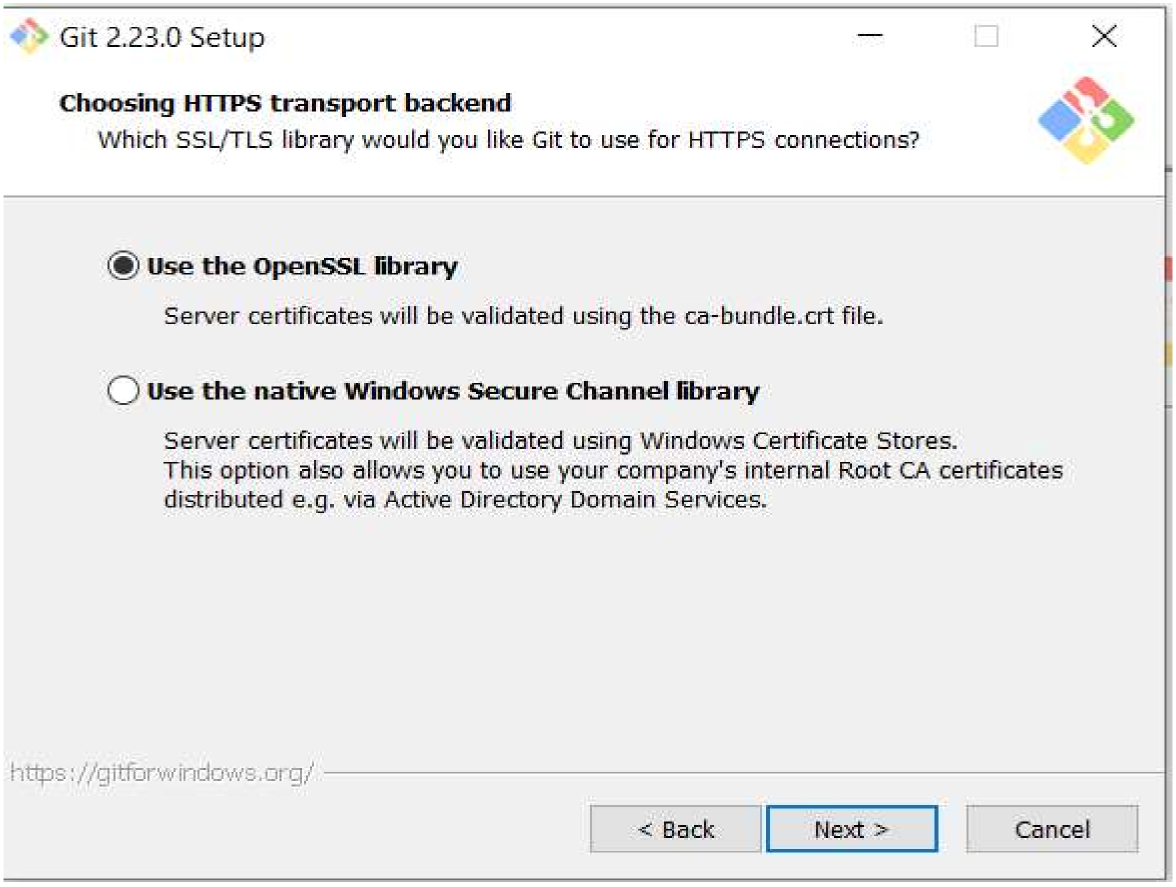
● **Git from the command line and also from 3rd-party software**

(Recommended) This option adds only some minimal Git wrappers to your
PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov
PowerShell as well as any third-party software looking for Git in PATH.

○ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only
use this option if you understand the implications.

https://gitforwindows.org/


Git 2.23.0 Setup

Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

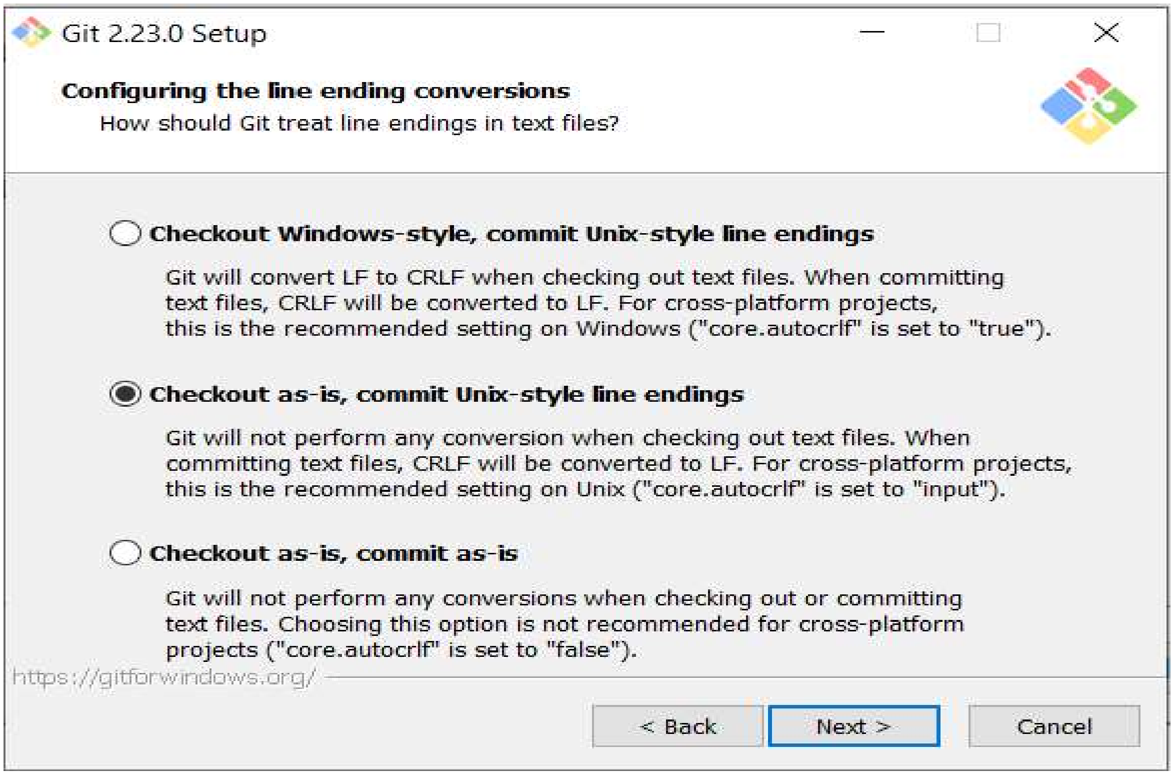
[✓ ● **Use the OpenSSL library**]

Server certificates will be validated using the ca-bundle.crt file.

○ **Use the native Windows Secure Channel library**

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.

https://gitforwindows.org/


Git 2.23.0 Setup

Configuring the line ending conversions

How should Git treat line endings in text files?

○ **Checkout Windows-style, commit Unix-style line endings**

Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").

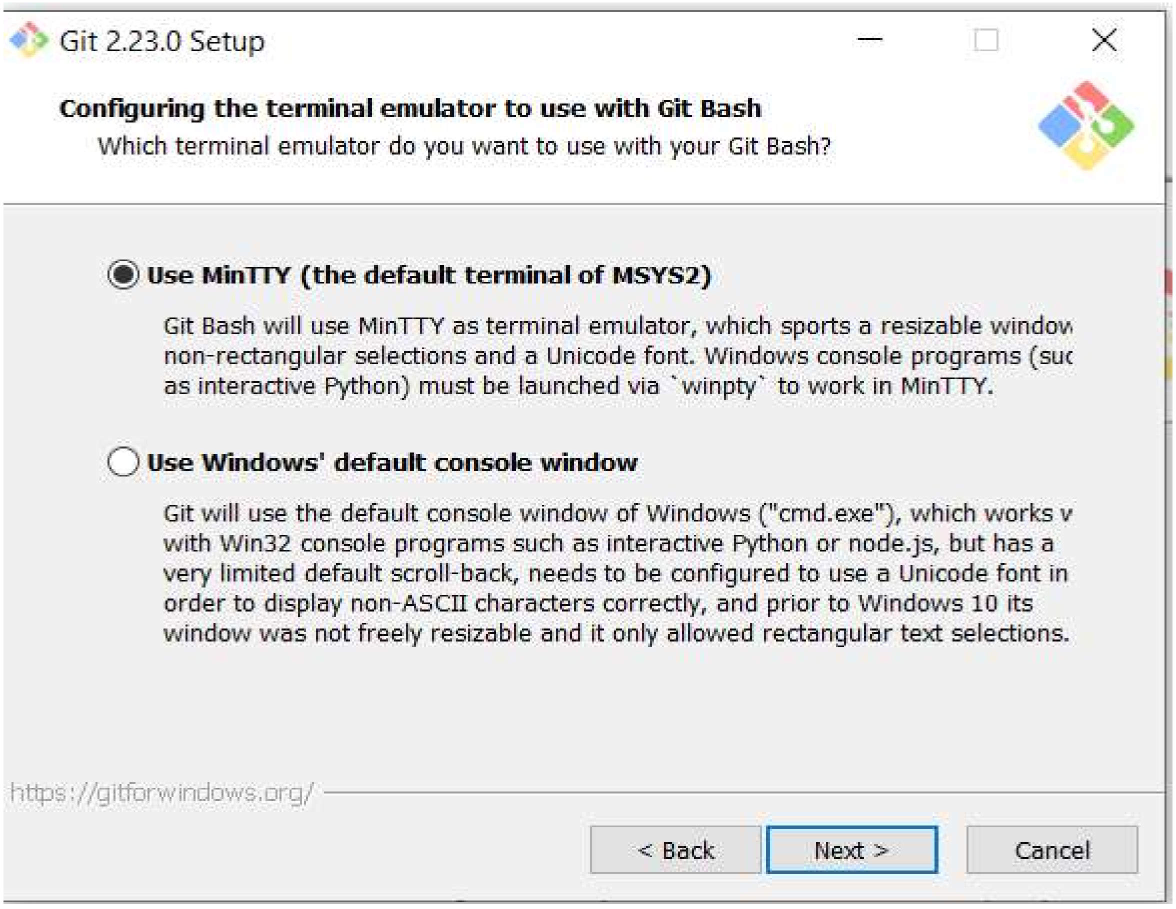
[✓ ● **Checkout as-is, commit Unix-style line endings**]

Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Unix ("core.autocrlf" is set to "input").

○ **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").

https://gitforwindows.org/



Git 2.23.0 Setup

Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

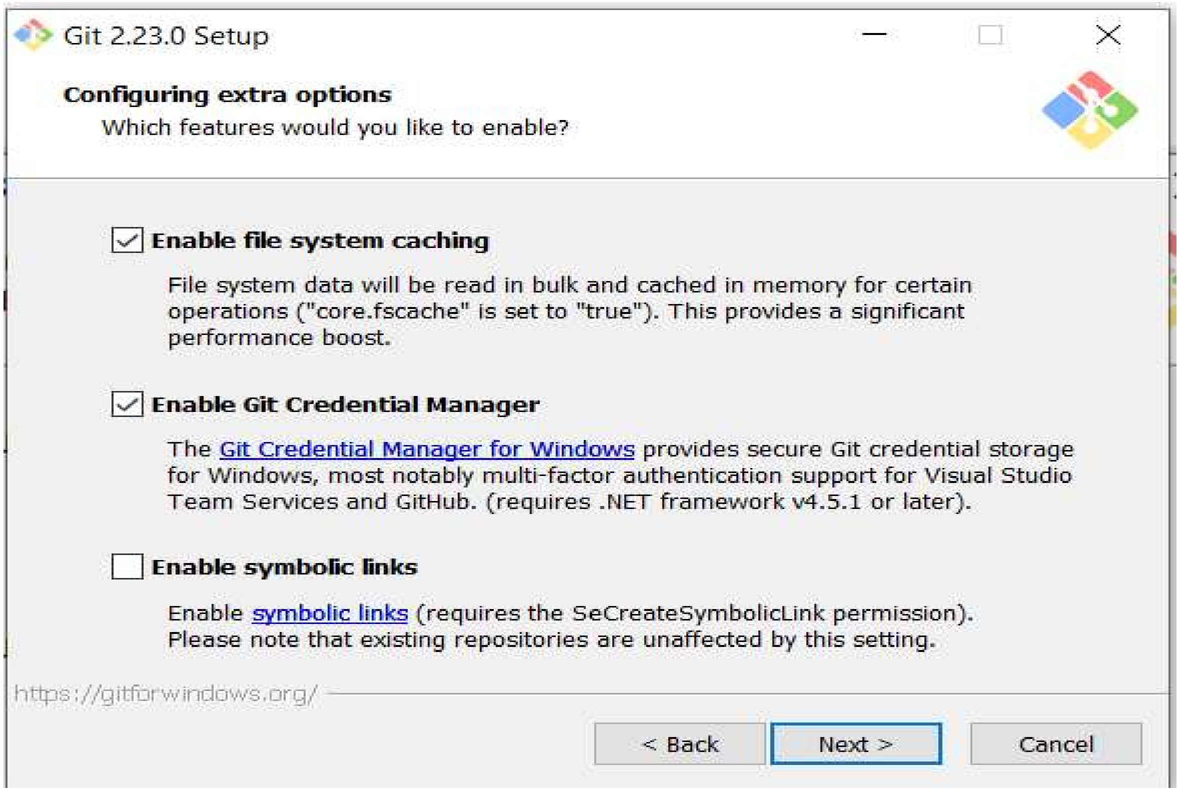
[✓ **Use MinTTY (the default terminal of MSYS2)**]

Git Bash will use MinTTY as terminal emulator, which sports a resizable window
non-rectangular selections and a Unicode font. Windows console programs (suc
as interactive Python) must be launched via `winpty` to work in MinTTY.

Use Windows' default console window

Git will use the default console window of Windows ("cmd.exe"), which works v
with Win32 console programs such as interactive Python or node.js, but has a
very limited default scroll-back, needs to be configured to use a Unicode font in
order to display non-ASCII characters correctly, and prior to Windows 10 its
window was not freely resizable and it only allowed rectangular text selections.

https://gitforwindows.org/


Git 2.23.0 Setup

Configuring extra options

Which features would you like to enable?

[✓] **Enable file system caching**

File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

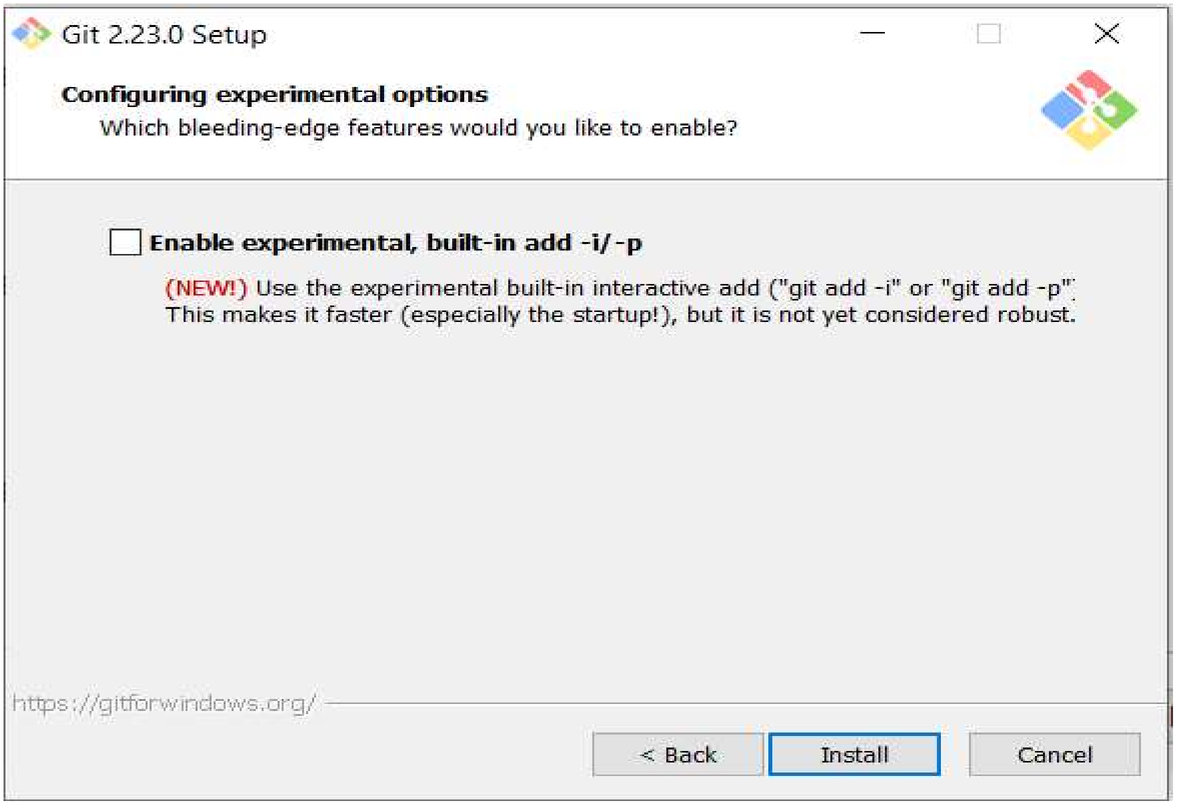
[✓] **Enable Git Credential Manager**

The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or later).

[ ] **Enable symbolic links**

Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

https://gitforwindows.org/


Git 2.23.0 Setup

Configuring experimental options

Which bleeding-edge features would you like to enable?

[ ] **Enable experimental, built-in add -i/-p**

(NEW!) Use the experimental built-in interactive add ("git add -i" or "git add -p" This makes it faster (especially the startup!), but it is not yet considered robust.

https://gitforwindows.org/

To Check the version of GIT

```
git –version
```

To get any help regarding the GIT

```
git help config
```

or

```
git config --help
```

To configure the name and email

```
git config --global user.name 'Deepti'
git config --global user.email 'deeptigutti@gmail.com'
```

- Need to pass the --global option, because then Git will always use that information for anything user do on that system.

To check the username and gmail

```
git config --list --show-origin
```

To Create the file in the Git Bash Use

```
touch filename.file-extension
```

To Intialize GIT

git init

- Initialize an existing directory as a Git repository.
- Use the git init command to create a Git repository in the current directory.
- Git does not care whether user start with an empty directory or if it contains already files. This creates a new subdirectory named .git that contains all of the necessary repository files.

Git Clone

- Retrieve an entire repository from a hosted location via URL.
- Clone operation creates the instance of the repository.
- Clone operation not only checks out the working copy, but it also mirrors the complete repository.
- The target repo can be local or remote.
- The only time networking gets involved is when the repository instances are being synchronized.

GitHub

It is a code sharing and publishing service, or that it’s a social networking site for programmers.

- GitHub is an American company that provides hosting for software development version control using Git.
- A GitHub repository can be used to store a development project.
- It can contain folders and any type of files (HTML, CSS, JavaScript, Documents, Data, Images).

- A GitHub repository can also include a licence file and a README file about the project.
- A GitHub repository can also be used to store ideas, or any resources that you want to share.

References:

[1]. https://git-scm.com/images/logos/downloads/Git-Logo-1788C.png

[2]. https://git-scm.com/downloads

[3]. https://stackoverflow.com/questions/1408450/why-should-i-use-version-control

[4]. https://git-scm.com

[5]. https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git

[6]. https://en.wikipedia.org/wiki/Distributed_version_control

[7]. https://en.wikipedia.org/wiki/Git

[8]. https://www.tutorialspoint.com/git/git_basic_concepts.htm