

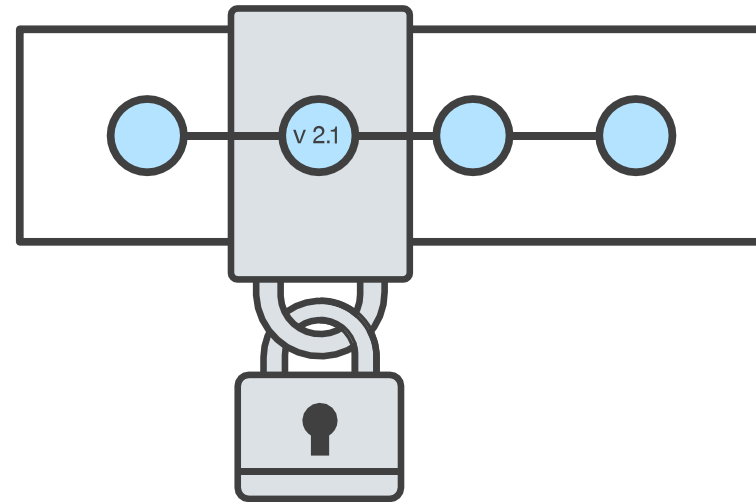
**WELCOME  
SOFTWARE  
ENGINEERS**



**GitHub**

# VERSION CONTROL

- Records changes to a file or set of files over time so that you can recall specific versions later.



# GIT

- Version-control system.
- For tracking changes in computer files.
- Coordinating work on those files.
- Among multiple people.



# GITHUB

- Web-based hosting service for version control using Git.
- Offers all of the distributed version control.
- Offers source code management functionality of Git as well as adding its own features.



**GitHub**

# Creating a GitHub Account

---

Username

Email

Password

Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter.  
Read our documentation on [safer password practices](#).

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.



# Creating a GitHub Account

---

Username

 ✓

Email

 ✓

Password

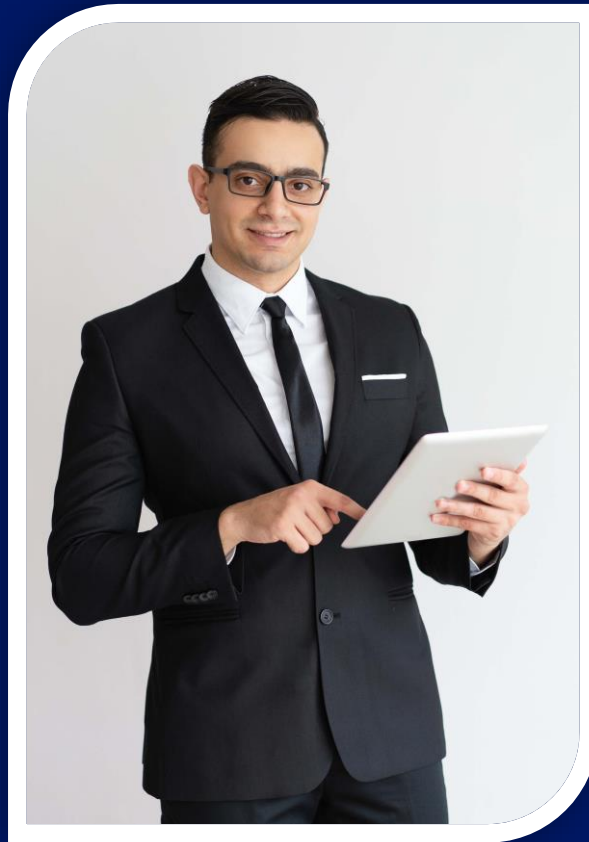
Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter.  
Read our documentation on [safer password practices](#).

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.



# Verify Email



# Git Installation for Linux

```
sudo apt update
```

```
sudo apt install git
```

 Linux is first choice of programmers.





# Git Installation for MacOS

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

 Rich kids, proceed.



# Git Installation for MacOS

```
brew install git
```

 Rich kids, proceed.

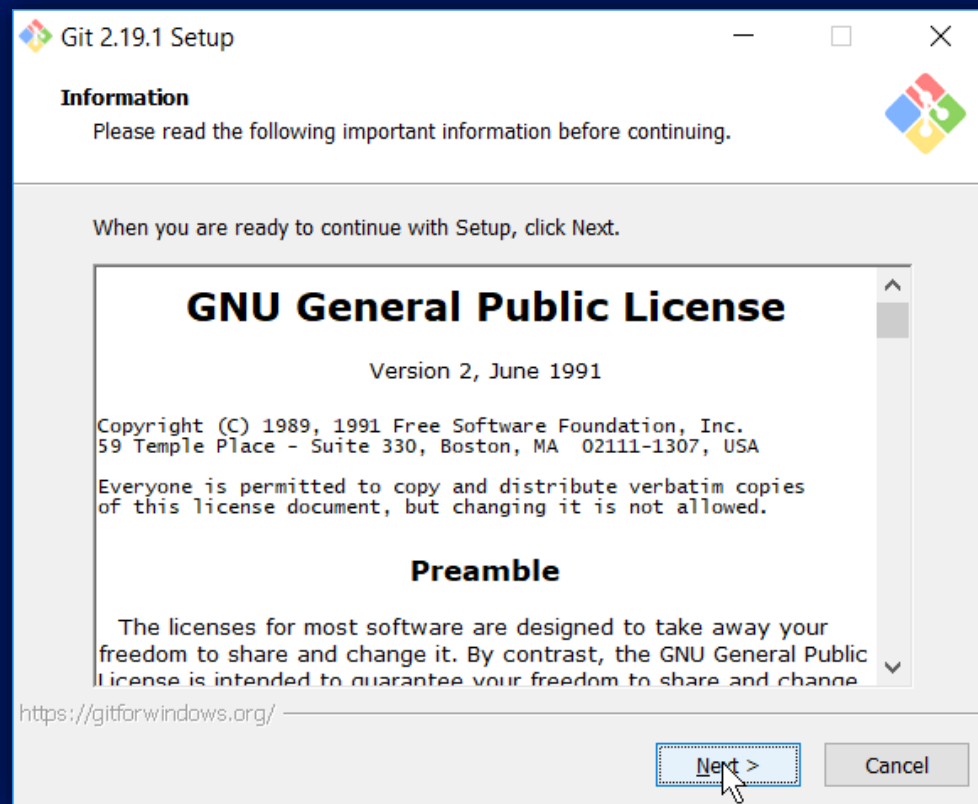


# Git Installation for Windows

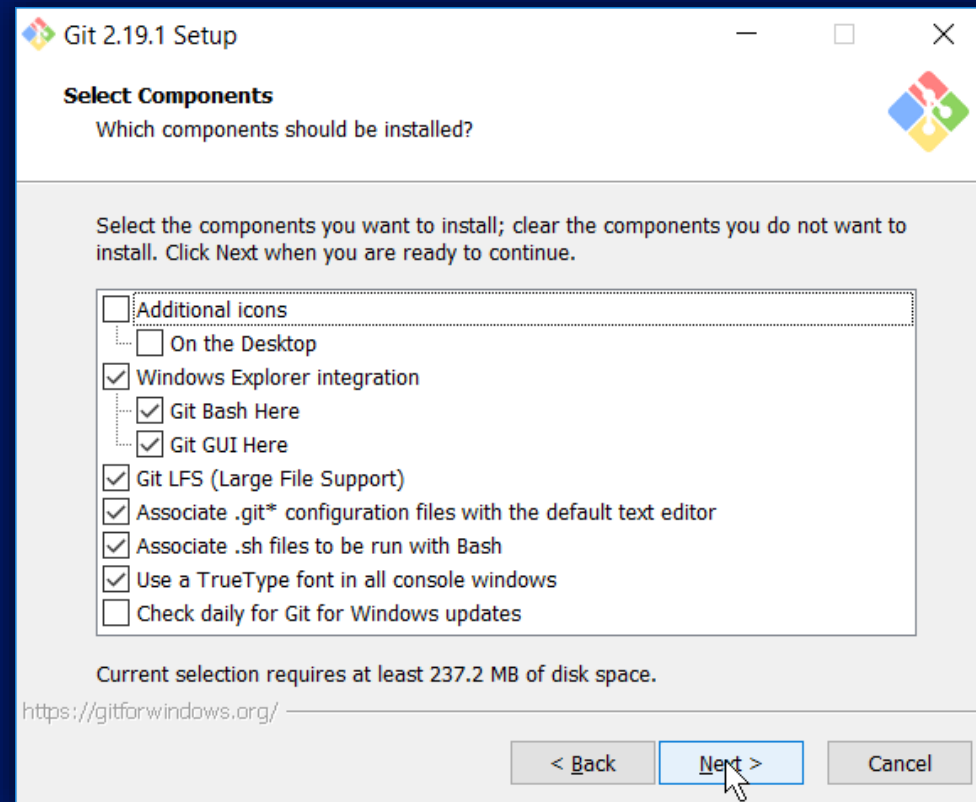
[bit.ly/gcesgit](https://bit.ly/gcesgit)



# Git Installation for Windows

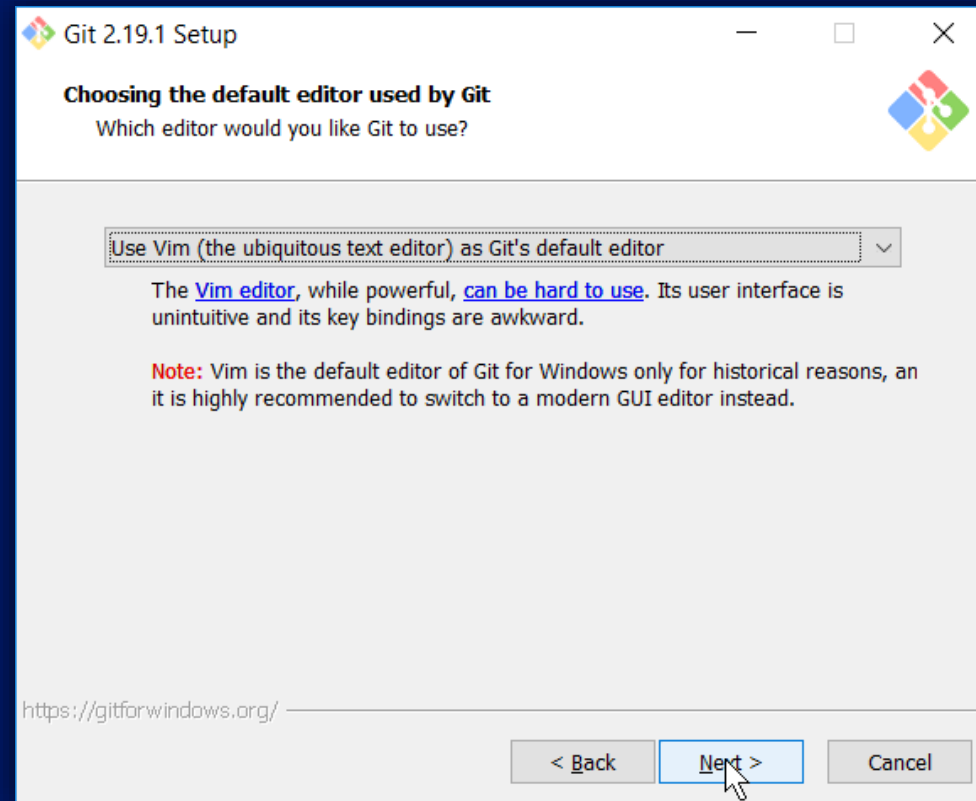


# Git Installation for Windows

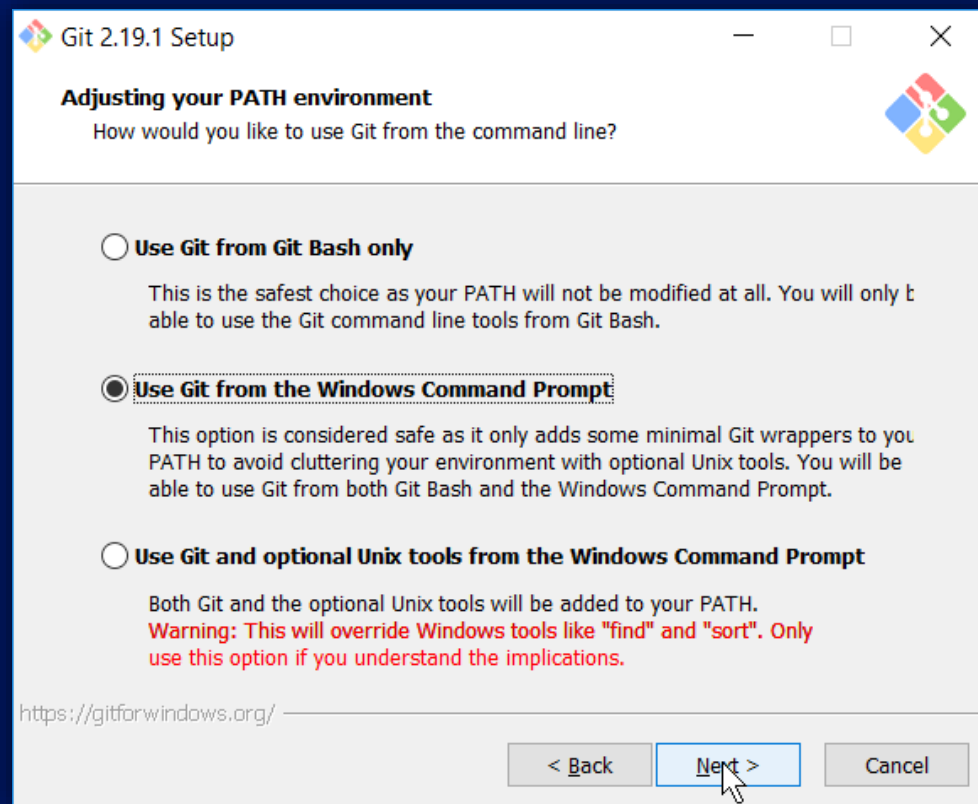


# Git Installation for Windows

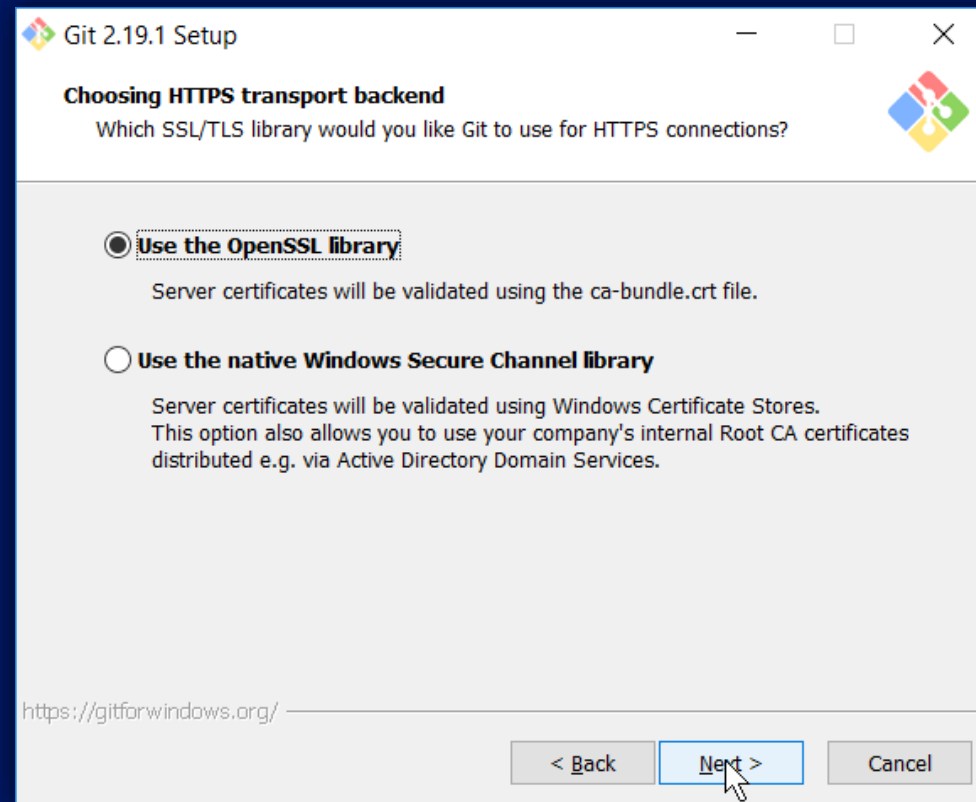
---



# Git Installation for Windows

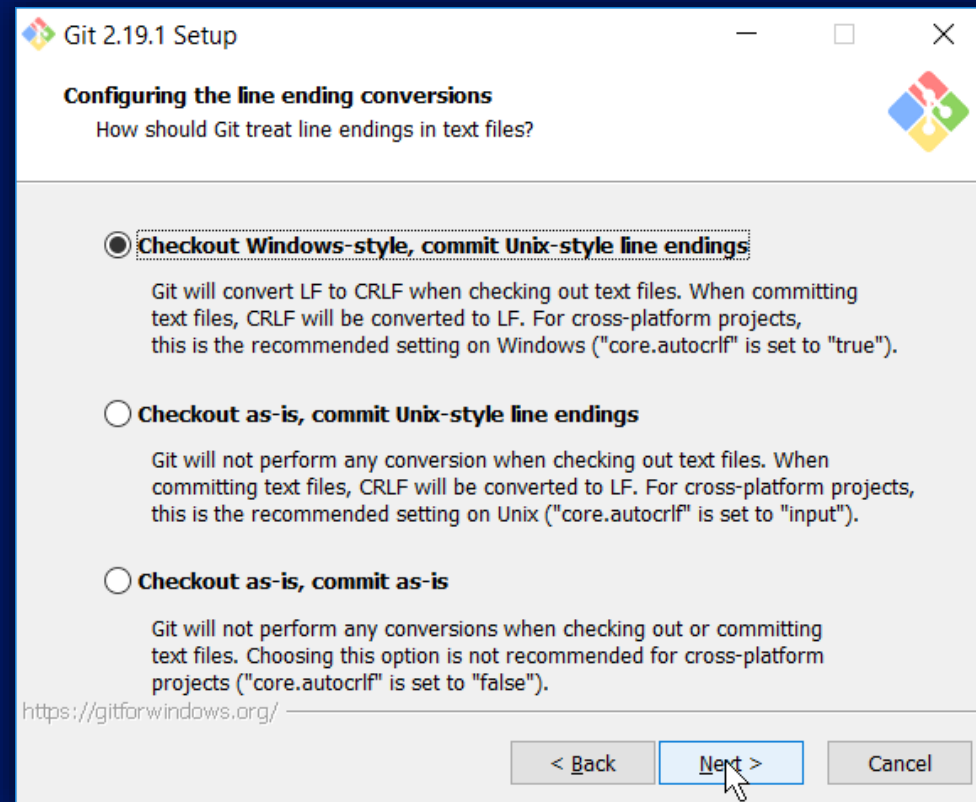


# Git Installation for Windows

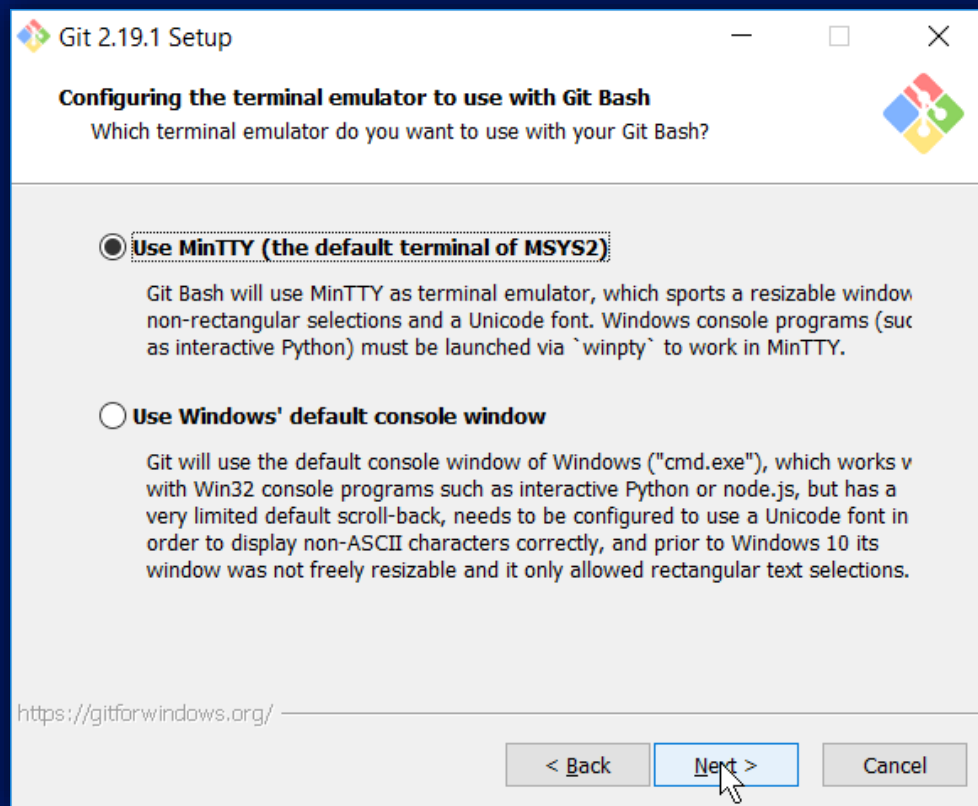




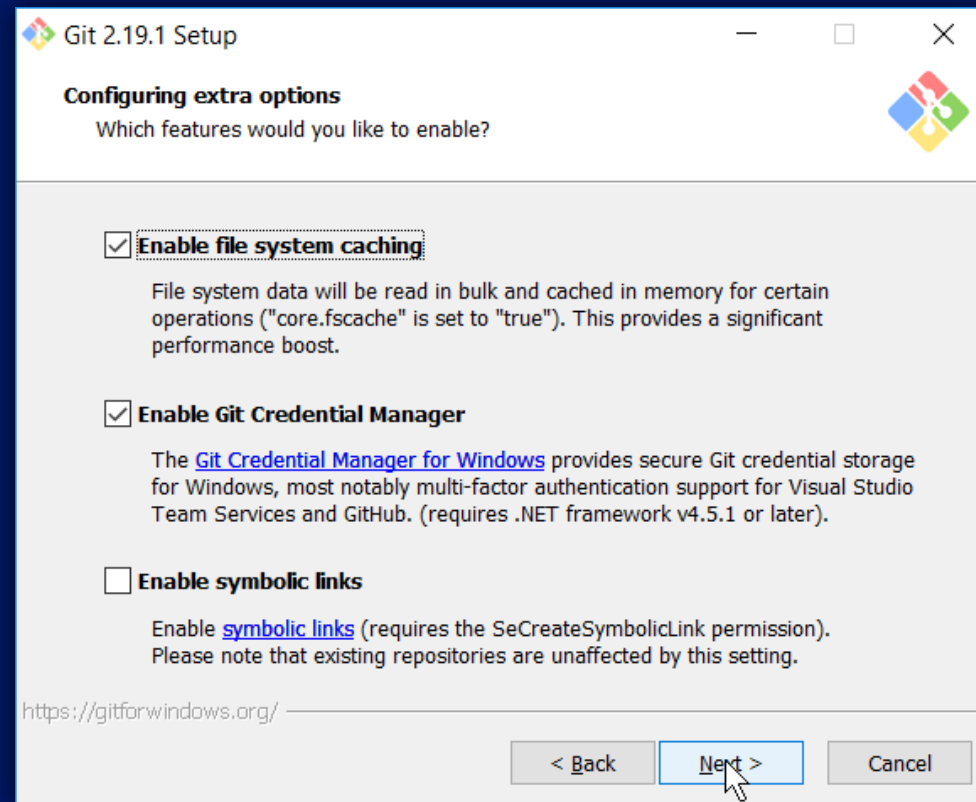
# Git Installation for Windows



# Git Installation for Windows

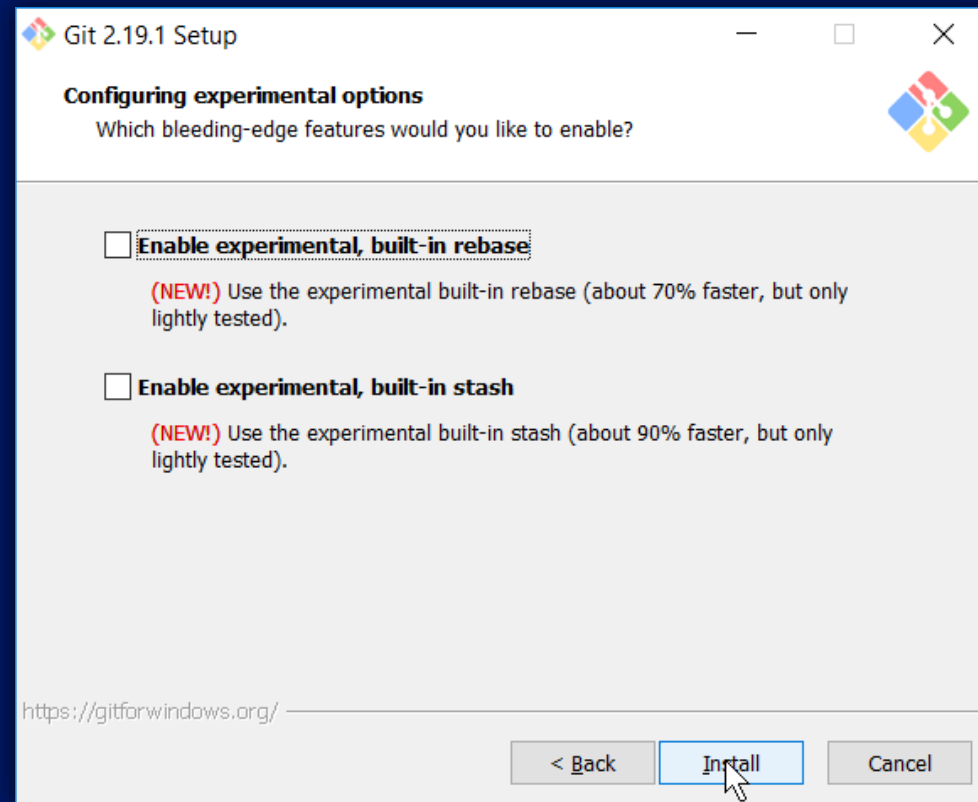


# Git Installation for Windows



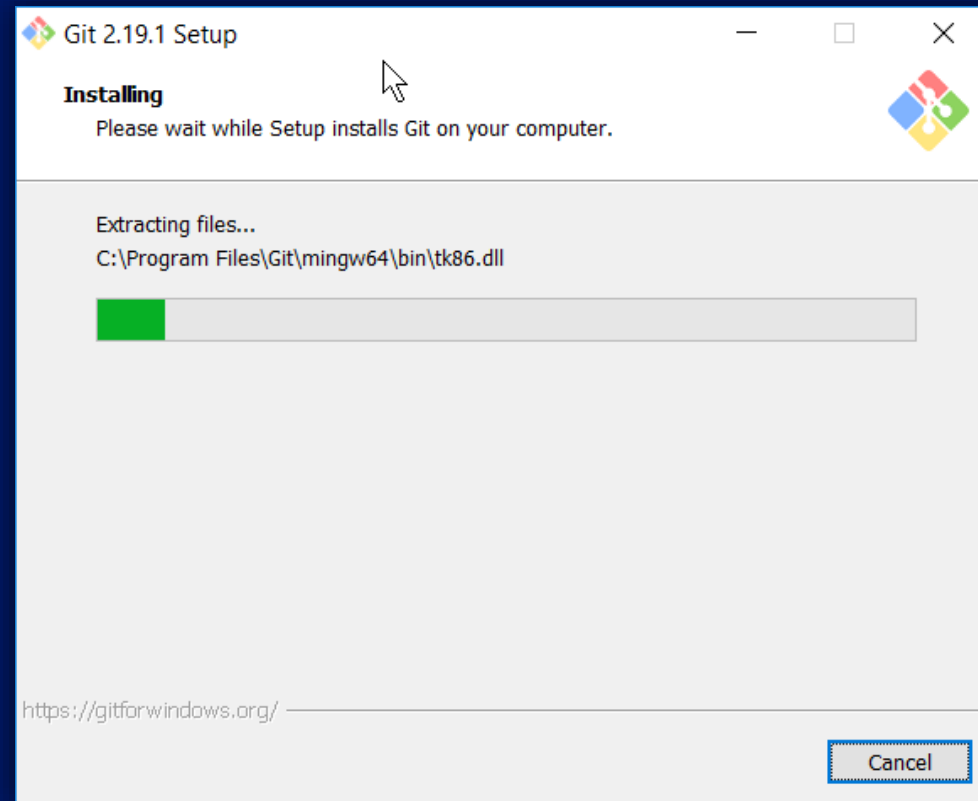
# Git Installation for Windows

---



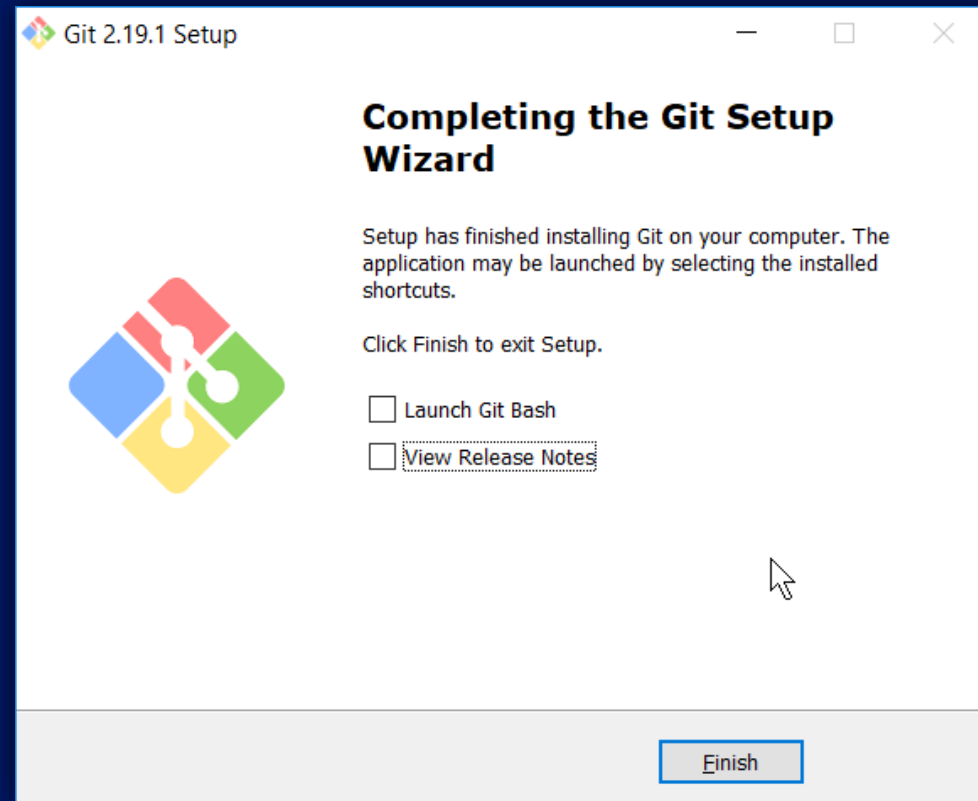
# Git Installation for Windows

---



# Git Installation for Windows

---



# Checking Git Installation

```
git --version
```

 Make sure you have newer version installed.



# Git Configuration

---

```
git config --global user.name "YourGitHubUserName"
```





# Git Configuration

---

```
git config --global user.name "YourGitHubEmail"
```

 Enter .edu email if you have.



# Git Configuration

---

```
git config --global core.editor "CodeEditorName"
```

 Default is VIM editor; change to other.



# Git Configuration

---

```
git config --global --edit
```



# Checking Git Configuration

---

```
git config --list
```

 It displays list of configurations.



# Checking Username

```
git config user.name
```

 Make sure to check once !



# Checking Email

```
git config user.email
```

 Make sure to check once !



If you ever get stuck!

```
git help <command name>
```

 It loads offline documentation.



# Let's play with terms

```
git help glossary
```



Let's dive !

```
init  
clone  
status
```



# Git Initialization

```
git init
```

 Now, your folder is initialized as Git Repository.



# Git Cloning

```
git clone https://github.com/thearjun/workshop.git
```



# Checking Git Status

```
git status
```

 Not Facebook Status 



# Review Repository

log  
show

# Reviewing Logs

```
git log
```

 It displays logs of commits.



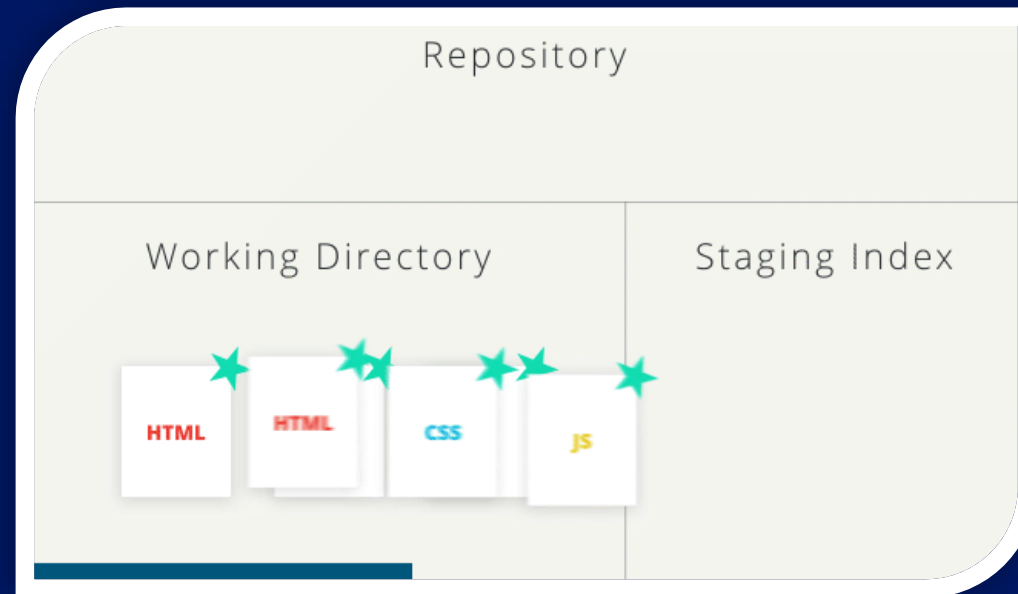
# Complete Review

```
git show
```

- ① It shows a bunch of information such as commits, diffs, etc.



# Git Repository Architecture



**i** Working Directory → Tracking / Staging Area → Git Repository





# Working With Files

```
add  
commit  
diff
```

# Adding Files to Repository

```
git add <fileName>
```

 This will track / stage selected file.



# Adding Files to Repository


```
git add .
```

 This will track / stage all files of current directory.



# Adding Files to Repository

```
git add -u
```

 This will track / stage all files of working directory.



# Committing Changes to Repository

```
git commit -m "Commit Title" -m "Commit Description"
```

 It's all about love. COMMITMENT matters.



# Checking Difference in Files

```
git diff
```

- ① It checks difference in file between working directory and repository.



# Checking Difference in Files

```
git diff --staged
```

- ① It checks difference in file between staging area and repository.



# Renaming Files

```
git mv <old file name.extension> <new file name.extension>
```

 Extension of file is mandatory.





# Renaming Folder

```
git mv <old folder name> <new folder name>
```



# Moving Files

```
git mv <fileName> folder/ <fileName>
```

 In Git, renaming and moving are done by mv.



# Removing Files

```
git rm <file name>
```

 It needs commit too.

# Tea Break



① दुइ चोटी थपेर कसैले नखानु नि

Let's dive deeper !

checkout  
tag  
branch  
merge



# Checkout



**i** Act of switching between different versions.

# Checkout

Works on following entities :

- Files
- Commits
- Branches



# Checkout on file

```
git checkout -- <file name>
```

 Undoing the uncommitted changes





# Checkout on commit

```
git checkout <commit code> -- <file name>
```

 Getting the old versions from repository



# Checkout on Branch

```
git checkout <branch name>
```

 Changing the active branch



# Tag



 It marks the milestones and create restore points.

# Creating tags

```
git checkout master  
git tag <tag name>
```

 Checkout is necessary to create tag at master branch.



# Creating tags with annotation

```
git tag -a <tag name> -m "Message goes here"
```

 -a means annotation here.



# Displaying Tags

```
git tag  
OR  
git show <tag name>
```

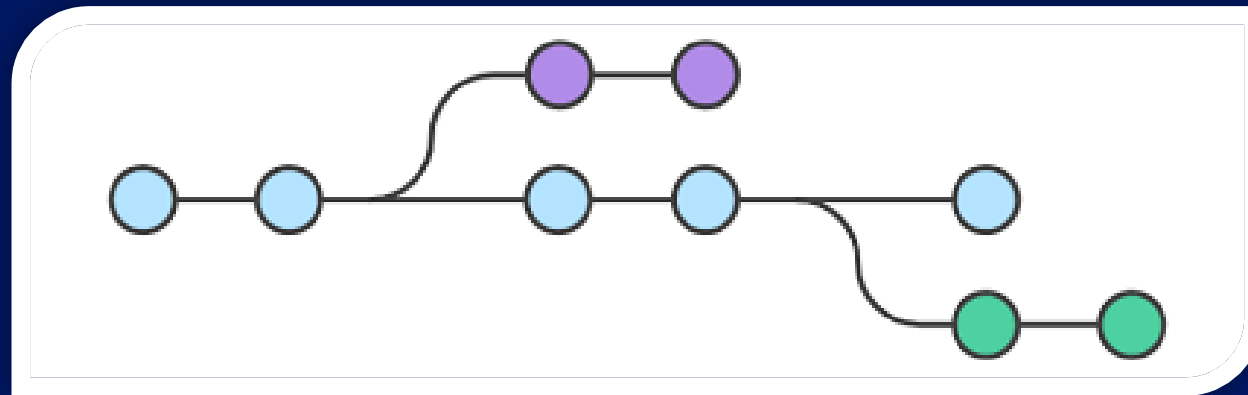


# Deleting Tags

```
git tag --delete <tag name>
```

 --delete can be replace by -d .

# Branches



① simply a lightweight movable pointer to one of these commits.



# Creating Branches

```
git branch <branch name>
```

- ① Branch is usually created when a feature is being added.



# Displaying Branches

```
git branch
```



# Switching Branches

```
git checkout <branch name>
```



# Merging Branches

```
git checkout master  
git merge <branch name>
```

- ① Master is default branch. Other branches get merged here.



# Deleting Branches

```
git branch -D <branch name>
```

 Checkout to master is mandatory before deleting other branch.



# We do a lot of mistakes

amend  
reset  
revert

- ① Even Michael Jordan missed his 9000 shots.  
Mistakes are important !

# Correcting the mistaken commit message

```
git commit --amend
```

 Change text editor to notepad; VIM is hard to learn.



# Reverting the commit

```
git revert <commit code>
```

 Revert means to return into previous state.

Mistaken commit's code is passed.





# Reset the uncommitted changes

```
git reset HEAD <file name>
```

 File need to be staged / tracked for this operation.

# Reset the committed changes

```
git reset --hard <commit code>
```

- ⓘ This operation is potentially destructive.  
Handle with care.

# Ignoring Files and Folders

Create .gitignore file at the root of the Git Repository.

```
*.* // for file  
<folder name> // for folder
```



# README.md

Markdown is a lightweight markup language with plain text formatting syntax.

```
open PDFs folder and view Markdown CheatSheet
```

 Markdown is widely used in GitHub and Stack Overflow.



# ASK ME ANYTHING



THANKS

