
COSC
Git
Workshop

What is Version Control?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Collaboration: VCS allows multiple team members to work on the same project simultaneously, ensuring seamless collaboration.

History Tracking: It maintains a detailed history of changes, making it easy to track who made what modifications and when.

Rollback Capability: VCS enables quick and safe rollbacks to previous versions, preventing the loss of valuable work in case of errors.

Branching and Merging: Developers can create separate branches for feature development or bug fixes and merge them back into the main codebase when ready.

Conflict Resolution: VCS provides tools for managing and resolving conflicts that may arise when multiple team members edit the same file.

But what is Git?

Git, the leading modern version control system used globally, was initially crafted by Linus Torvalds, renowned as the creator of the Linux operating system kernel, in 2005. Today, Git remains a mature and actively maintained open-source project, serving as the backbone for an impressive array of software projects, encompassing both commercial and open-source endeavors.

GitHub

Empowering Developers, Together.

- GitHub is a web-based platform for version **control** and **collaboration** among developers.
- It offers free access to **public** and **private** **repositories**.
- Repositories store project files and their revision history.
- Repositories can be public or private, with **multiple collaborators** allowed.

Git or GitHub?

VS

Installation of Git

WINDOWS

The official installer for windows is through this link:

https://gitscm.com/download/win

LINUX

To install git type the command in the terminal:

```
sudo apt install git
```

MAC OS

For installing Git in macos follow the steps in the link:

https://gitscm.com/download/mac

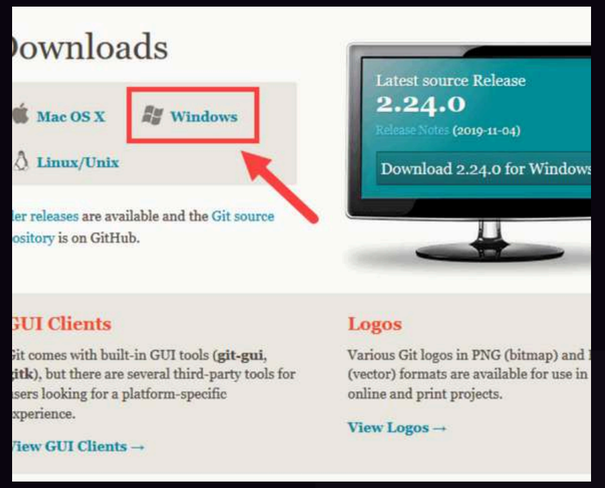
Starting from Scratch

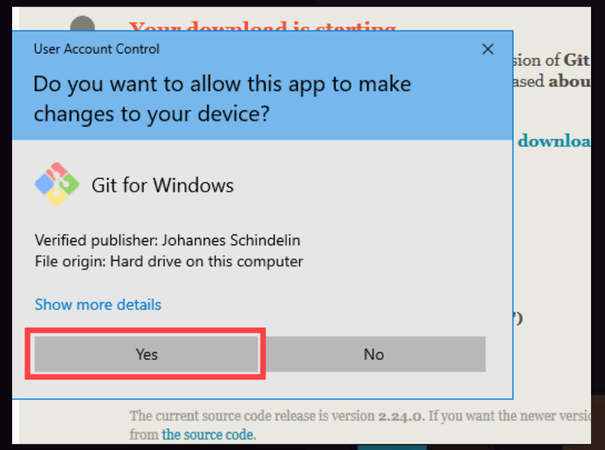
Visit The Website: Go to downloads page in official website https://git-scm.com/downloads

Download Software: Click on Windows download link and download 64-bit/32-bit installer.

Launch Installer: Go to your download location and double-click the file to launch the installer

User Account Control: Click "Yes" to allow app changes on User Account Control dialog.





Starting from Scratch

Choose Installation Location:

Use default location unless needed, then click Next.

Component Selection:

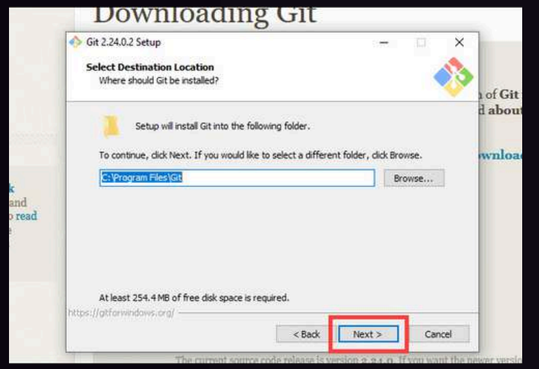
Use default components, if applicable, click Next.

Start Menu Folder:

Click Next to create a start menu folder.

Select Text Editor:

Choose Notepad++ or preferred editor from dropdown, click Next.



Starting from Scratch

Initial Branch Name:

Keep 'master' (unless required), click Next.

Adjusting your PATH environment:

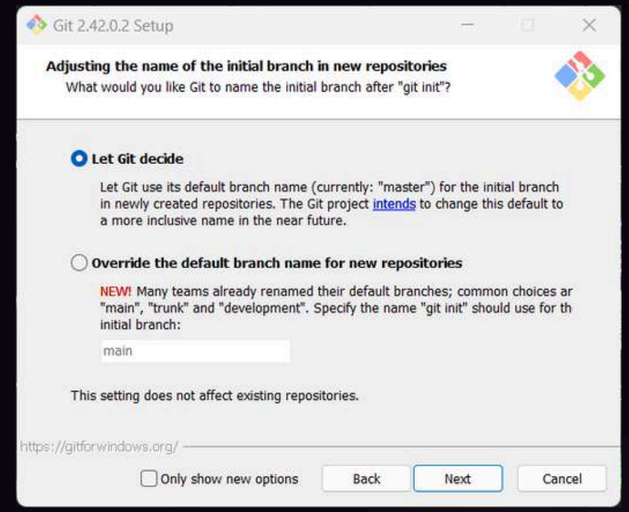
Select recommended option, click Next.

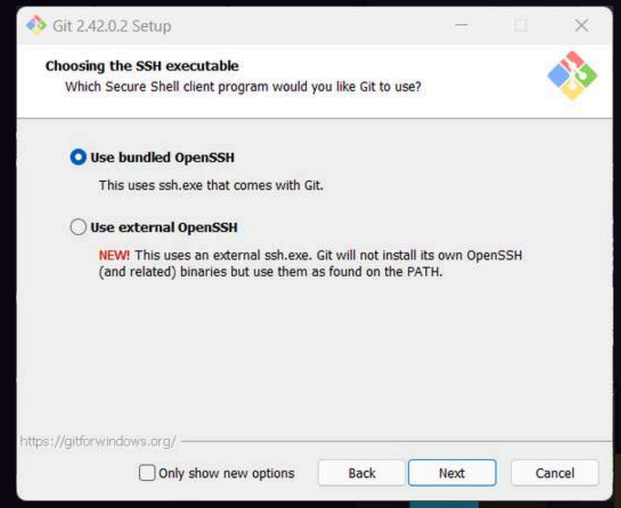
Choosing the SSH executable:

Select the default option (bundled OpenSSH)

Choosing HTTPS transport backend:

Select the default option (OpenSSL library)





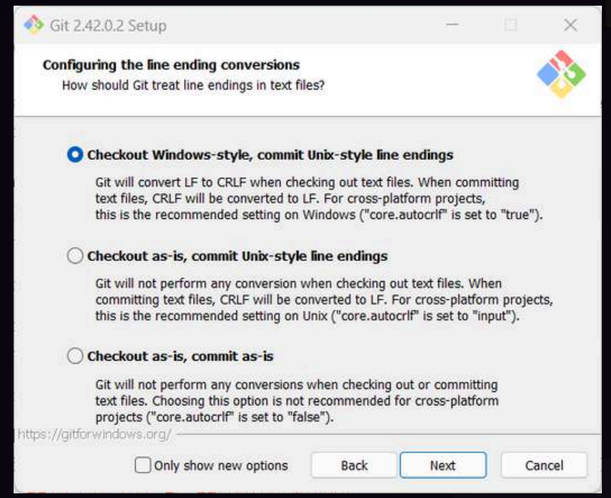
Starting from Scratch

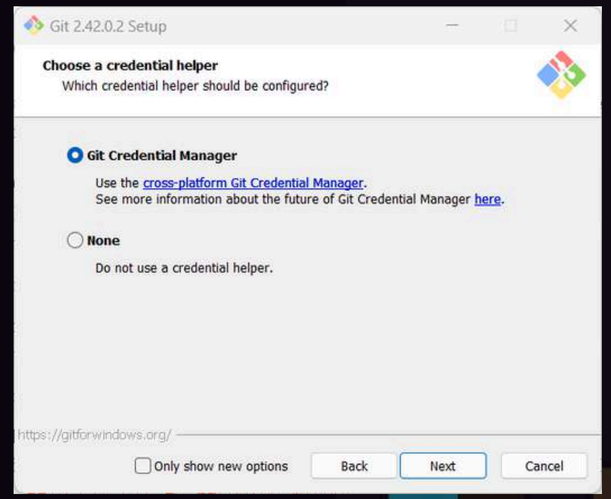
Configuring the line ending conversions:
Select the default option (Change based on your preference)

Configuring the terminal emulator:
Select the default option (Use MinTTY).

Choosing the default behavior of git pull:
Select Default (fast-forward or merge).

Choose a Credential Manager:
Select the default option (Git Credential Manager).





Starting from Scratch

Configuring extra options:

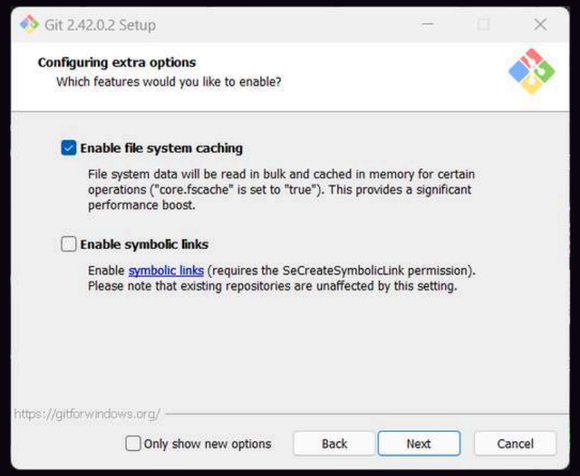
You can leave it like that or select more features if you want.

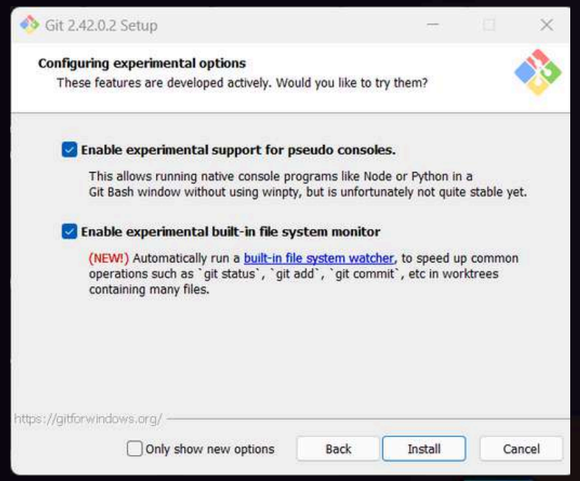
Configuring experimental options:

You can enable the options you like

Finish Installation:

You Finally click on Install to complete the installation process





GitHub Accounts

Getting Started

Setting your username:

```
$ git config --global user.name "username"
```

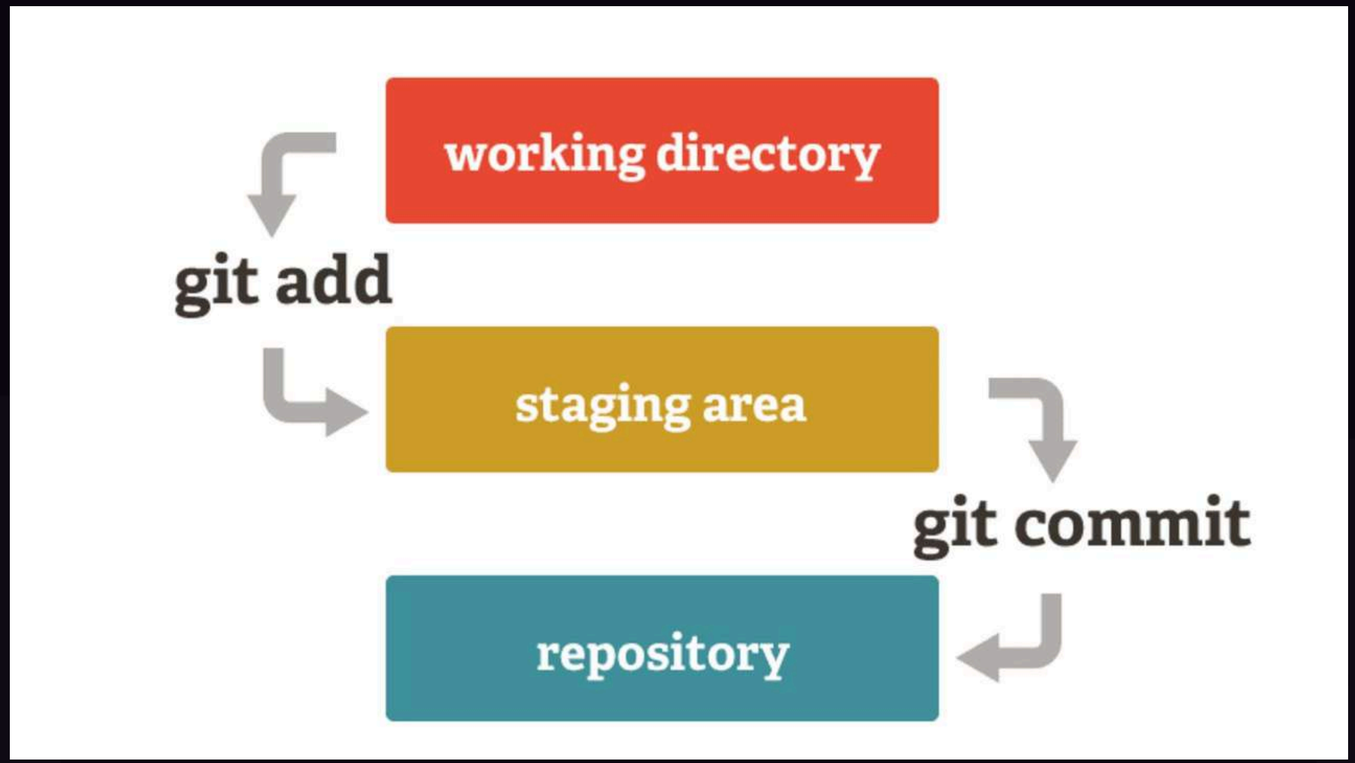
Setting your email-id:

```
$ git config --global user.email "email"
```

Listing git settings:

```
$ git config --list
```

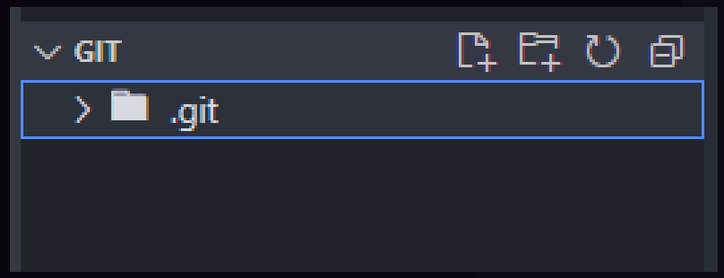
STATES IN GIT



GIT INIT

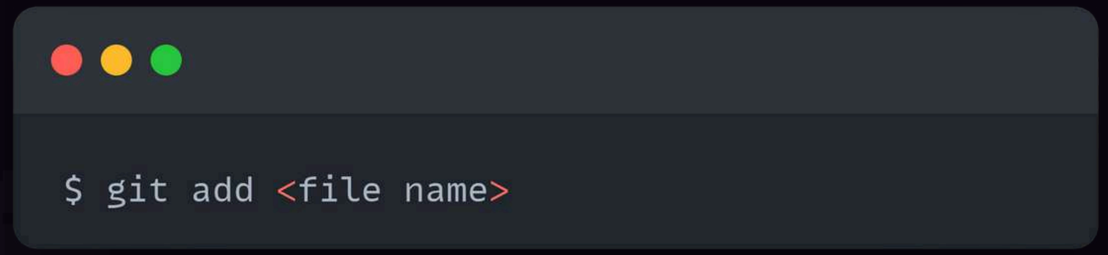
This creates a new subdirectory named .git that contains all of your necessary repository files—a Git repository skeleton.

```
$ git init
```



GIT ADD

The **git add** command is used to stage changes for the next commit.



Note: Staging means you are telling Git to include the changes in the next snapshot of your project, which will be recorded when you make a commit.

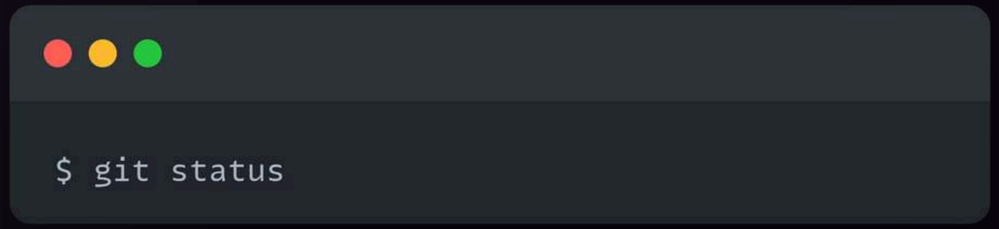
GIT COMMIT

The git commit command is used to create a new commit in your Git repository. A commit represents a snapshot of the project at a specific point in time and includes the changes you have staged using the git add command.

```
$ git commit -m "commit message"
```

GIT STATUS

The git status command is used to display information about the current state of your working directory and the staging area. When you run git status, Git provides you with information about which files have been modified, which files are staged for the next commit, and any untracked files.



GIT REMOTE

The git remote command is used to manage remote repositories. Remote repositories are copies of a Git repository that are hosted on other servers, often on platforms like GitHub.

```
$ git remote add origin <url>
```

GIT PUSH

The git push command is used to send your local commits to a remote repository.

```
$ git push <remote_name> <branch_name>
```

GIT PULl

The git pull command is used to fetch and merge changes from a remote repository into your current branch.

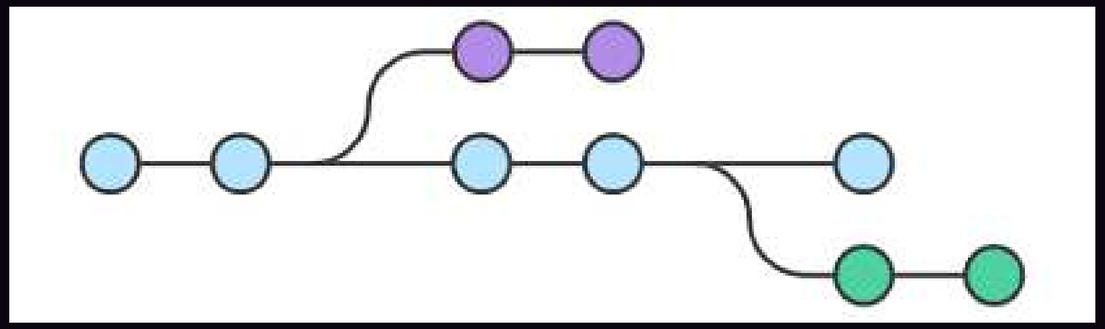
```
$ git pull <remote_name> <branch_name>
```

GIT IGNORE FILE

The .gitignore file is a configuration file used in Git to specify which files or directories should be ignored by Git when tracking changes in a repository.

Branching

A branch in Git is simply a lightweight, movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you’re given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.



GIT BRANCH

The git branch command is used to manage and view branches in a Git repository. It allows you to create, list, rename, delete, and switch between branches.

```
$ git branch "new-branch"
```

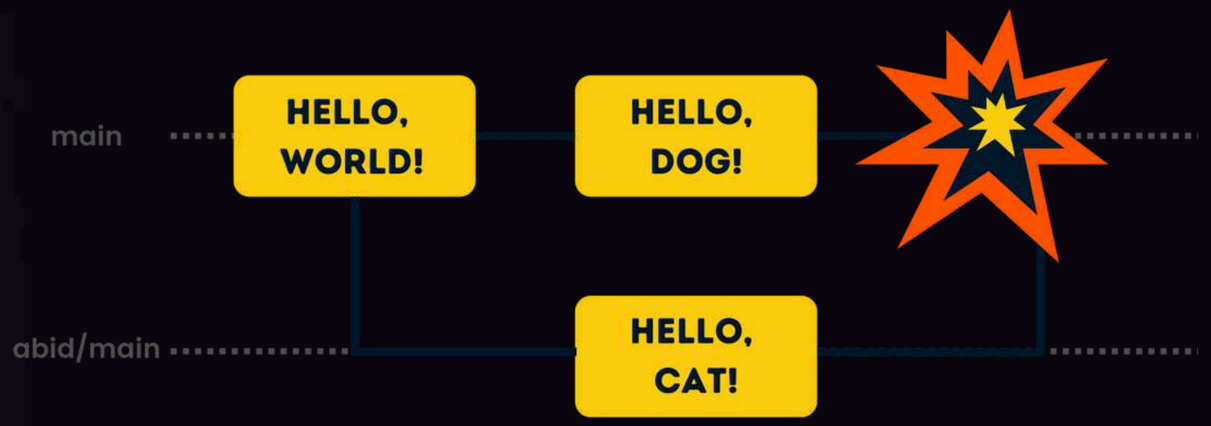
GIT CHECKOUT

The git checkout <branch_name> command is used to switch from your current branch to another branch.

```
$ git checkout <branch_name>
```

Merge conflict

A git merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.



Forking

A fork is a new repository that shares code and visibility settings with the original “upstream” repository. Forks are often used to iterate on ideas or changes before they are proposed back to the upstream repository, such as in open source projects or when a user does not have write access to the upstream repository.

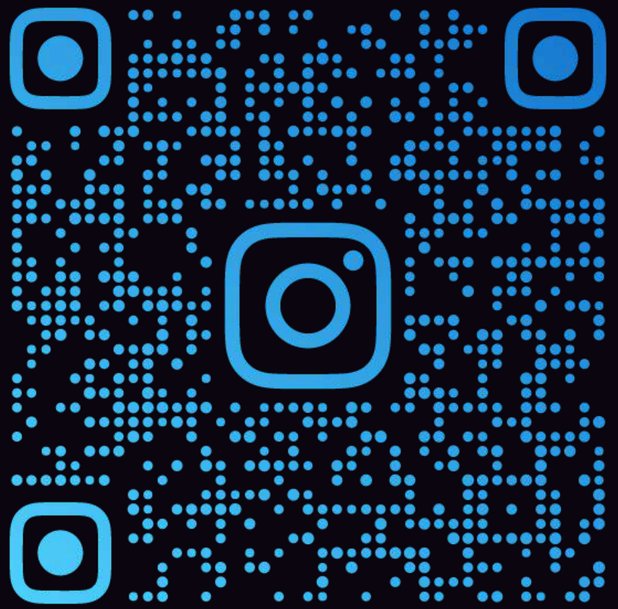
GIT CLONE

The **git clone** command is used to create a copy of a Git repository, including all of its files, commit history, and branches, on your local machine.

```
$ git clone <repository_url> [<destination_directory>]
```

Please feel free to ask any questions

Follow Us On Instagram



@CBITOSC