Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted By:
ANSHU KUMAR
2110990220 G08

Submitted To:
Dr.MONIT
KAPOOR

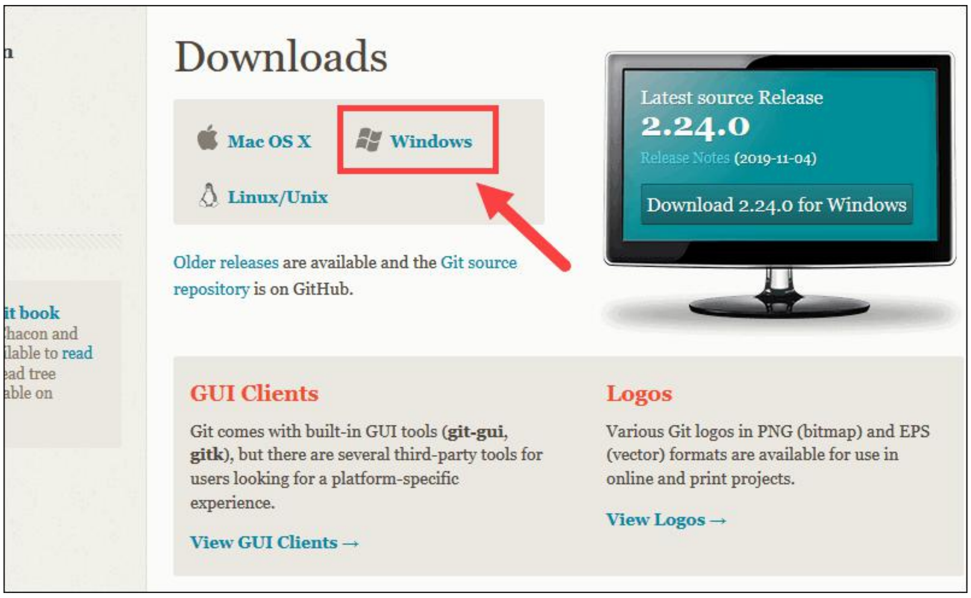
EXPERIMENT	TOPIC	PAGE No.
EXPERIMENT -1	Setting up of Git Client	03-12
EXPERIMENT-2	Setting up Git Hub Account	13-17
EXPERIMENT-3	How to Use Git Log	18-20
EXPERIMENT-4	Create and visualize Branch	21-24
EXPERIMENT-5	Life Cycle of the Git	25-30

AIM :-**Setting up of Git Client**

Steps For Installing Git for Windows

Download Git for Windows

1. Browse to the official Git Website: https://git-scm.com/downloads
2. click the download link for Windows and allow the download to complete.



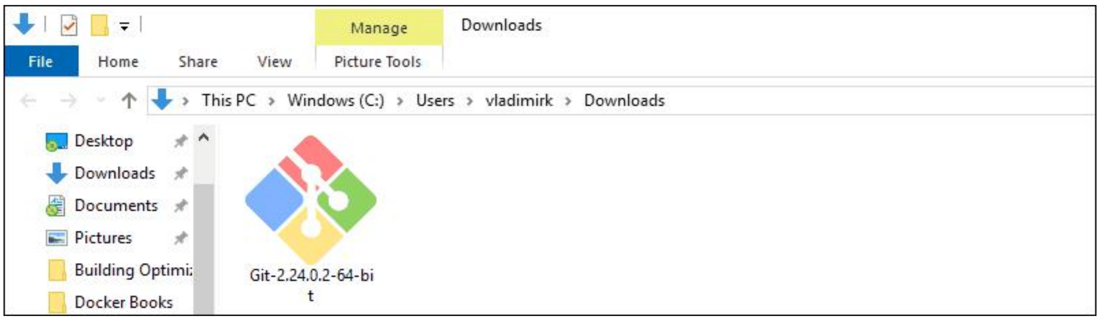
Older releases are available and the Git source repository is on GitHub.

Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a platform-specific experience.

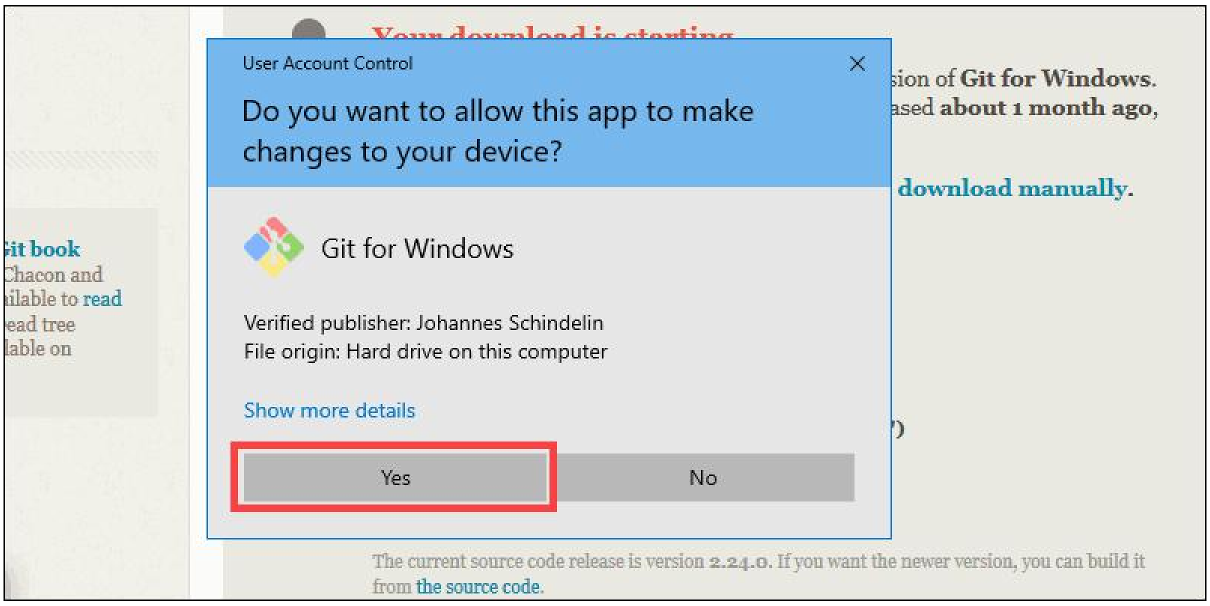
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

Extract and Launch Git Installer

3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.



4.Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.


User Account Control

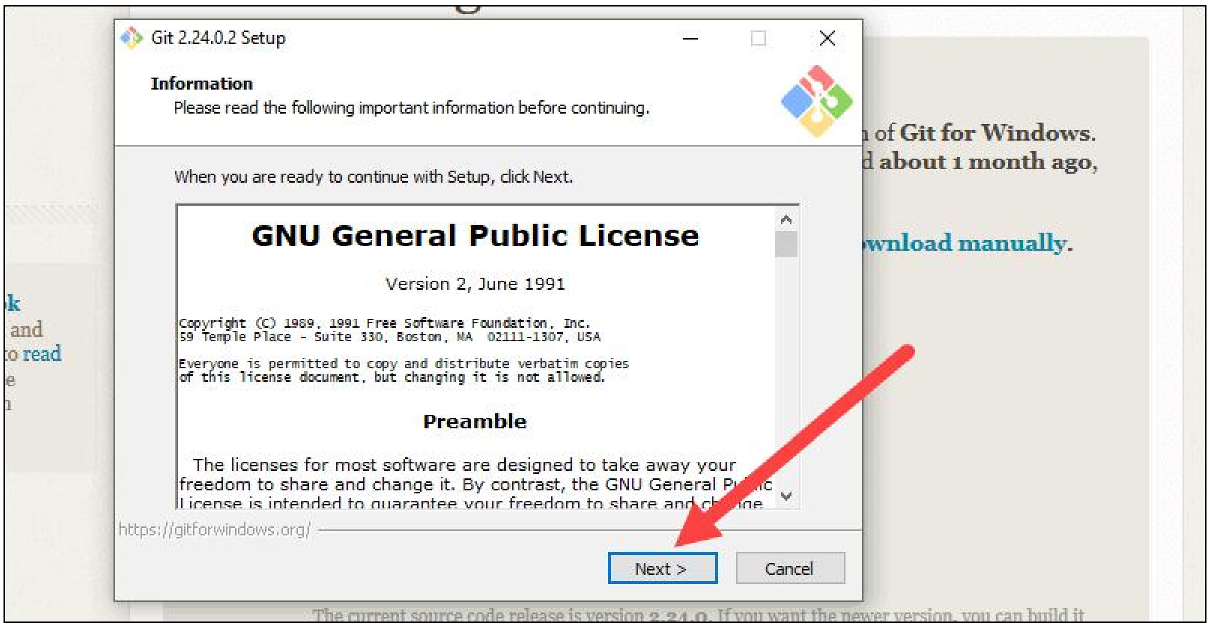
Do you want to allow this app to make changes to your device?

Git for Windows

Verified publisher: Johannes Schindelin
File origin: Hard drive on this computer

Show more details

5. Review the GNU General Public License, and when you're ready to install, click **Next**.


Git 2.24.0.2 Setup

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

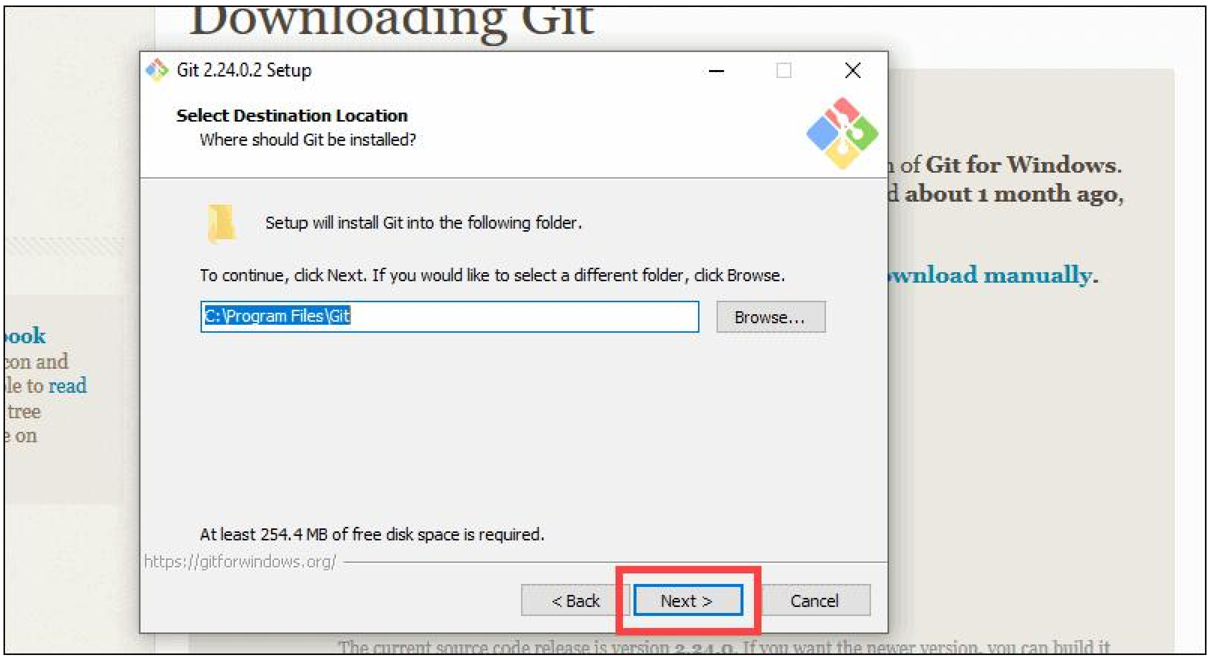
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

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/

6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.


Git 2.24.0.2 Setup

Select Destination Location

Where should Git be installed?

Setup will install Git into the following folder.

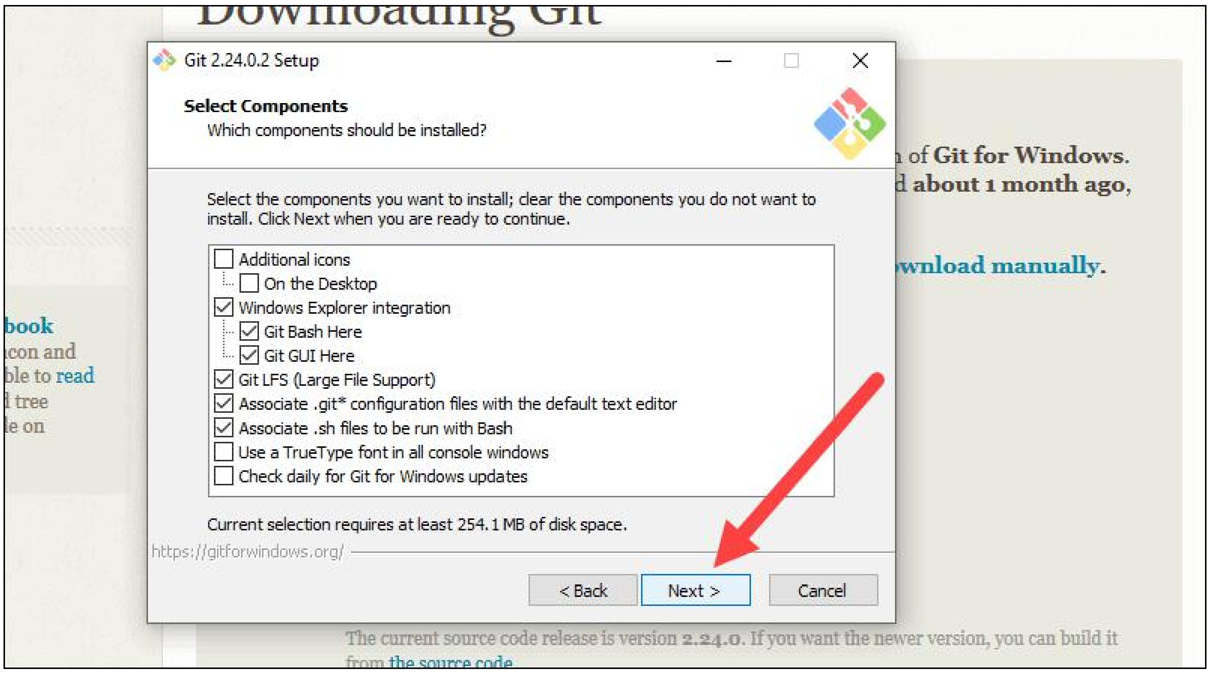
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 254.4 MB of free disk space is required.

https://gitforwindows.org/

7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.



Git 2.24.0.2 Setup

Select Components

Which components should be installed?

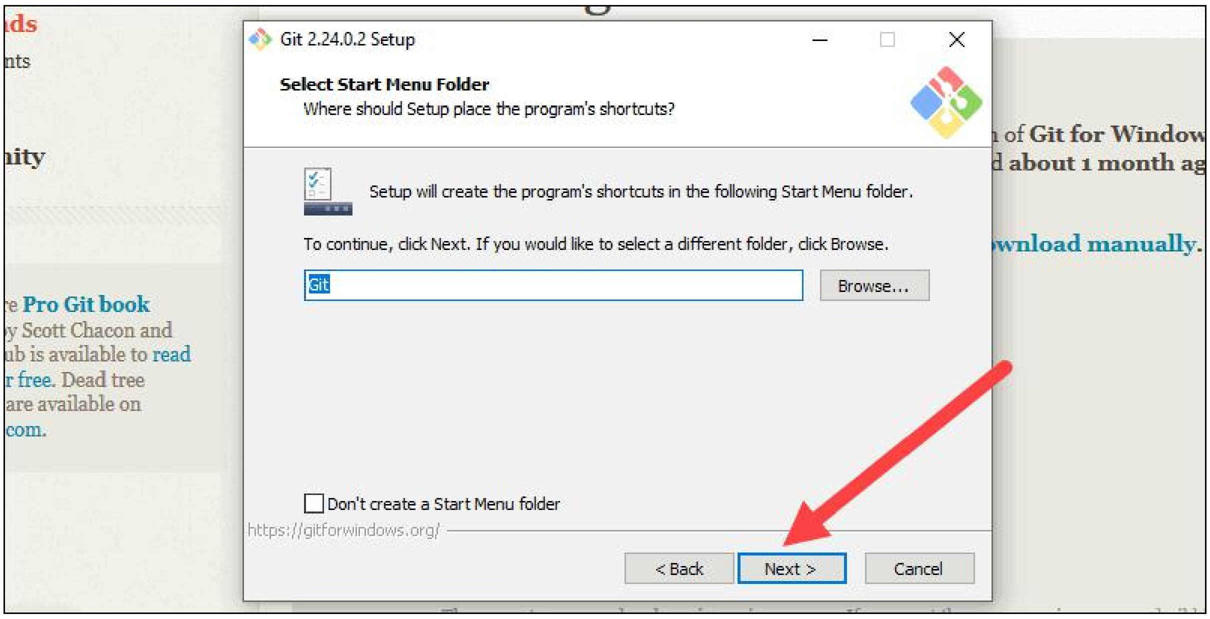
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Additional icons
  - [ ] On the Desktop
- [x] Windows Explorer integration
  - [x] Git Bash Here
  - [x] Git GUI Here
- [x] Git LFS (Large File Support)
- [x] Associate .git* configuration files with the default text editor
- [x] 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 254.1 MB of disk space.

https://gitforwindows.org/

8. The installer will offer to create a start menu folder. Simply click **Next**.



Git 2.24.0.2 Setup

Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

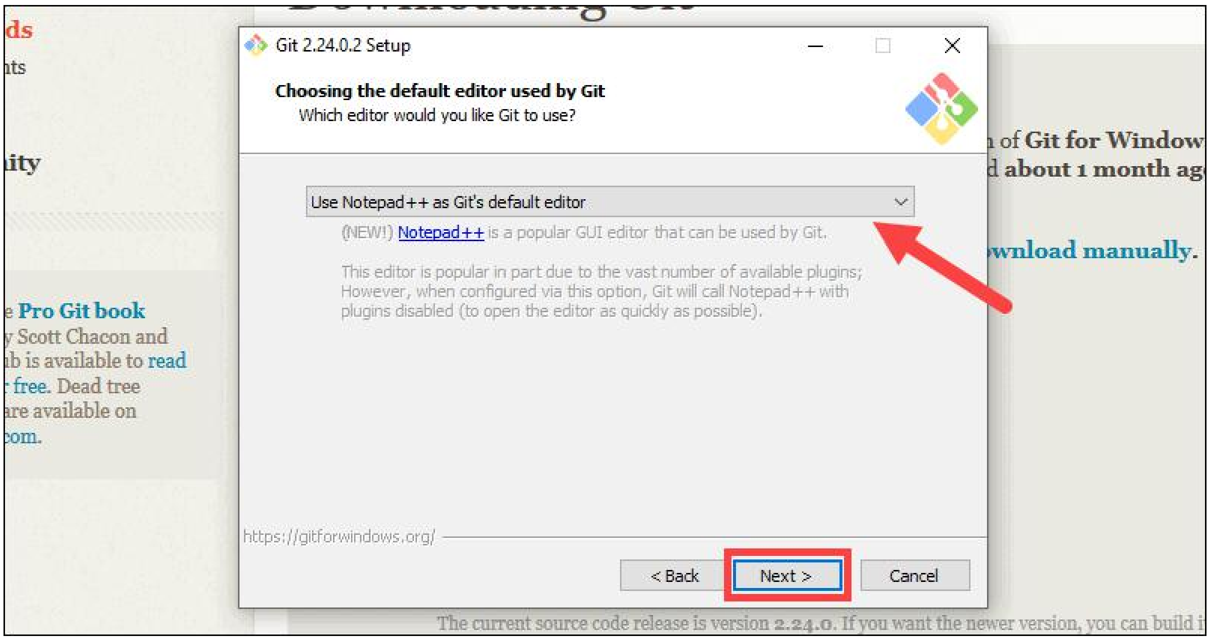
To continue, click Next. If you would like to select a different folder, click Browse.

Git

- [ ] Don't create a Start Menu folder

https://gitforwindows.org/

9. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.


Git 2.24.0.2 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

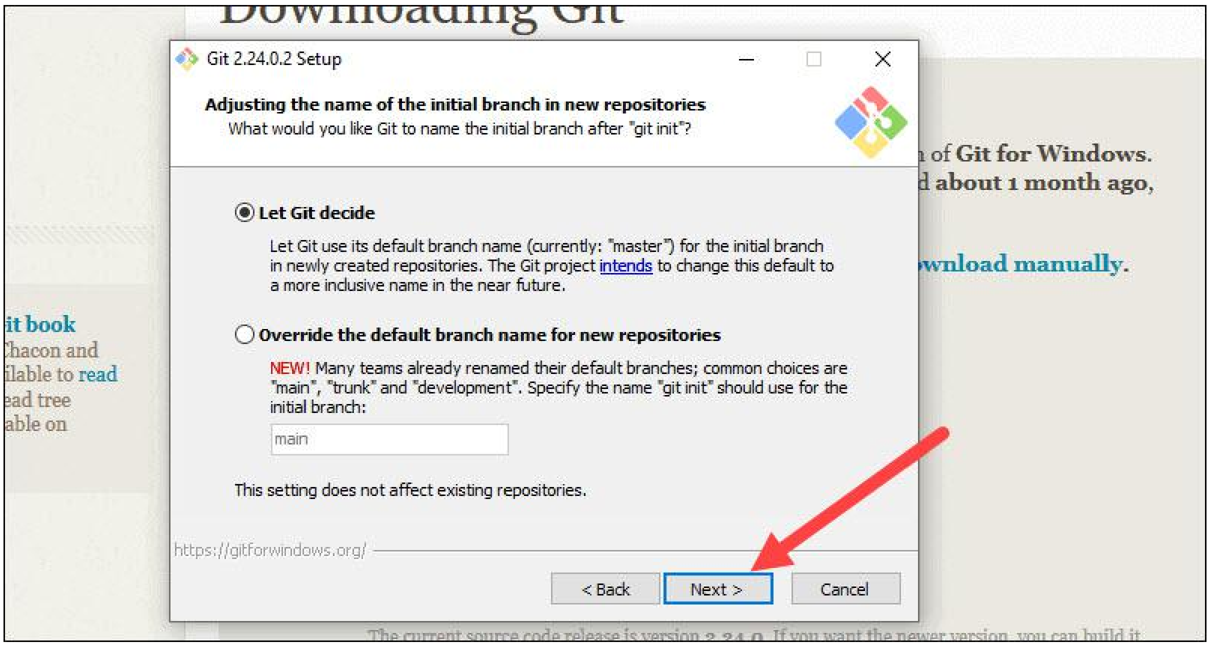
Use Notepad++ as Git's default editor

(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/

10. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click **Next.**


Git 2.24.0.2 Setup

Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

[✓ **Let Git decide**]

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to change this default to a more inclusive name in the near future.

Override the default branch name for new repositories

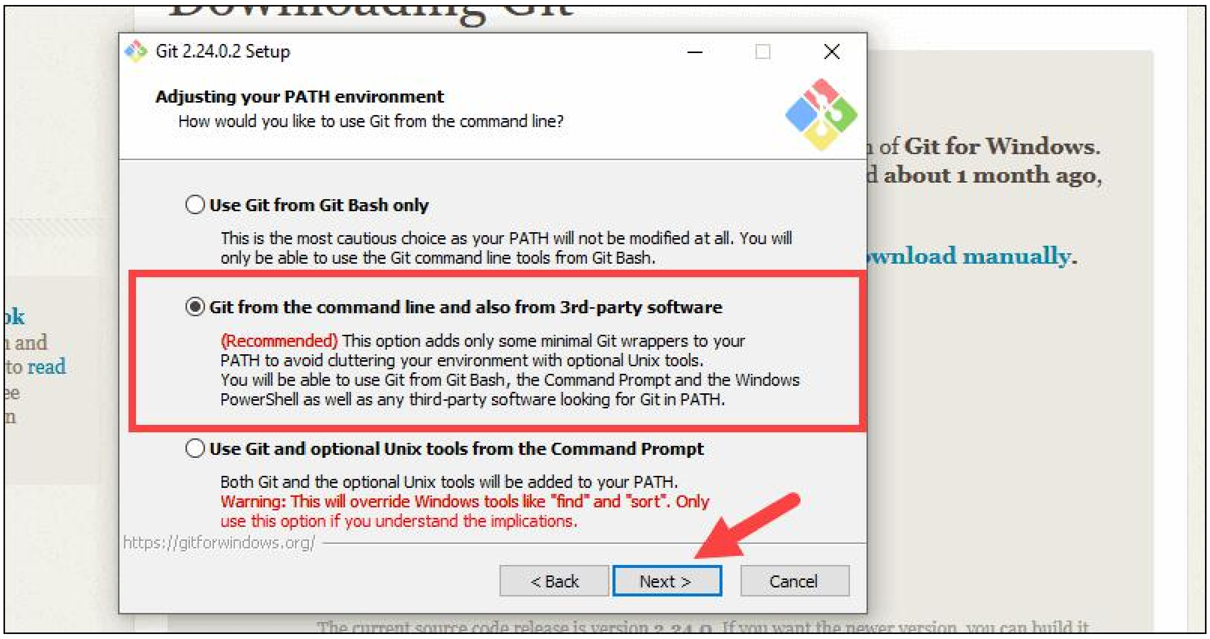
NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

main

This setting does not affect existing repositories.

https://gitforwindows.org/

11. This installation step allows you to change the **PATH environment**. The **PATH** is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.


Git 2.24.0.2 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 will 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 Windows PowerShell as well as any third-party software looking for Git in PATH.

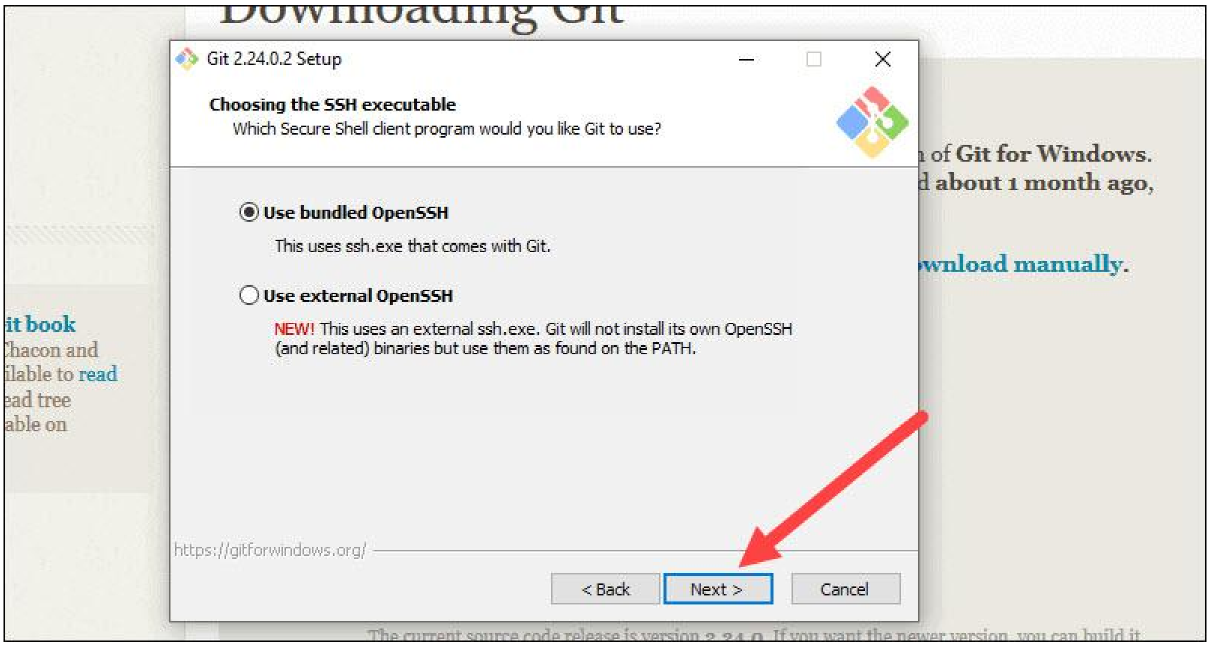
( ) **Use Git and optional Unix tools from the Command Prompt**

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/

Server Certificates, Line Endings and Terminal Emulators

12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**


Git 2.24.0.2 Setup

Choosing the SSH executable

Which Secure Shell client program would you like Git to use?

(●) **Use bundled OpenSSH**

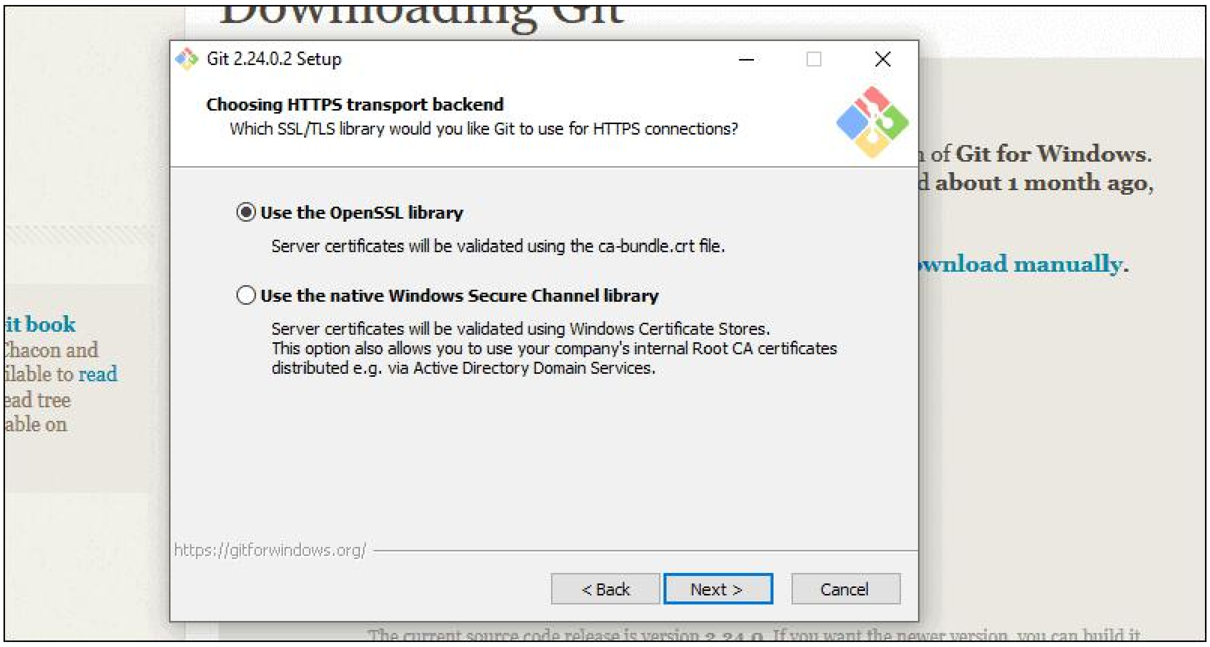
This uses ssh.exe that comes with Git.

( ) **Use external OpenSSH**

NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

https://gitforwindows.org/

13. The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.


Git 2.24.0.2 Setup

Choosing HTTPS transport backend

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

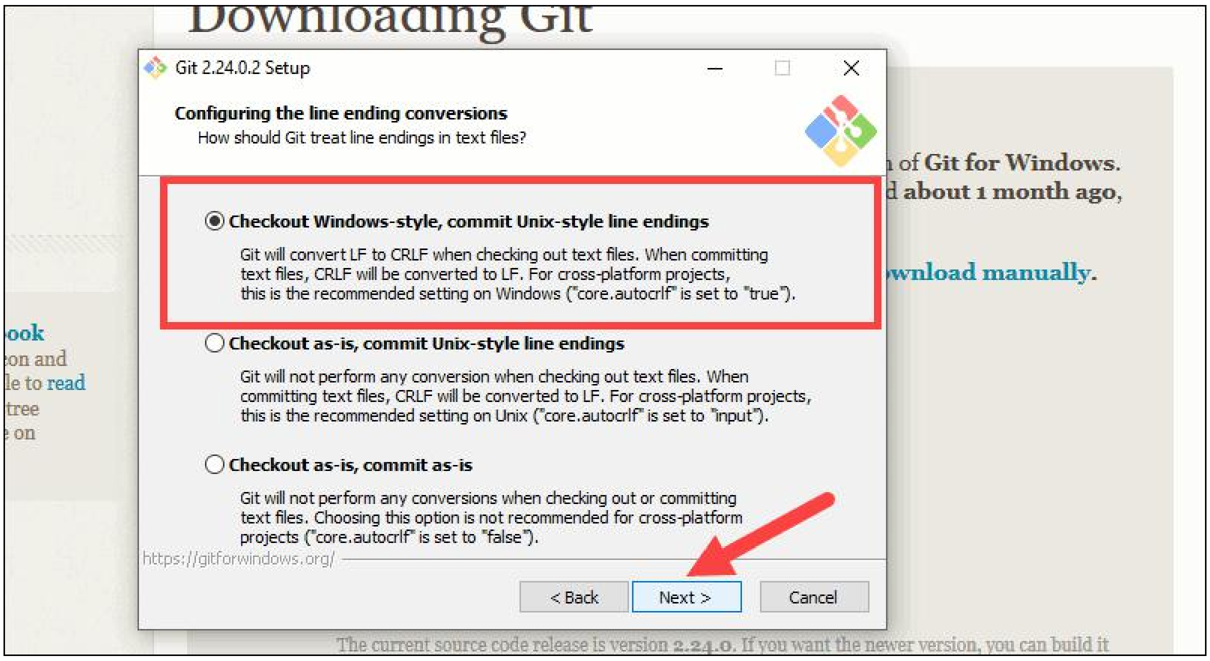
[✓ **Use the OpenSSL library**]

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.

14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.



Git 2.24.0.2 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

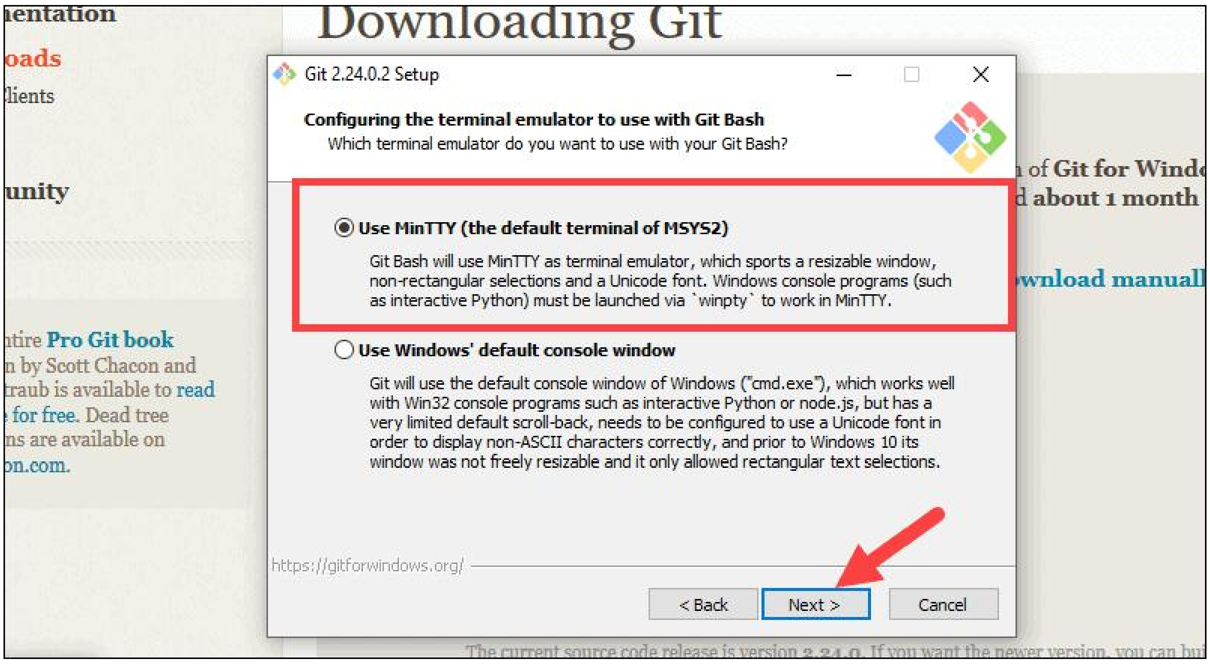
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

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/

15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.



Git 2.24.0.2 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)**]

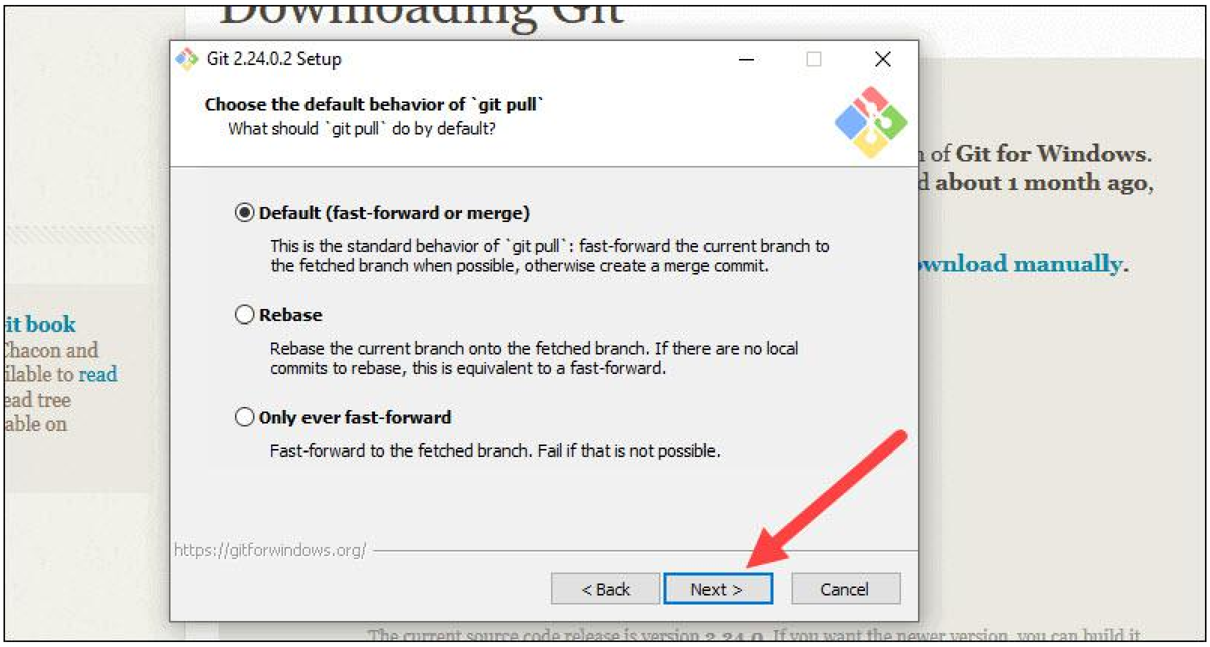
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

Use Windows' default console window

Git will use the default console window of Windows ("cmd.exe"), which works well 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/

16. The installer now asks what the `git pull` command should do. The default option is recommended unless you specifically need to change its behavior. Click **Next** to continue with the installation.


Git 2.24.0.2 Setup

Choose the default behavior of `git pull`

What should `git pull` do by default?

(●) **Default (fast-forward or merge)**

This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

( ) **Rebase**

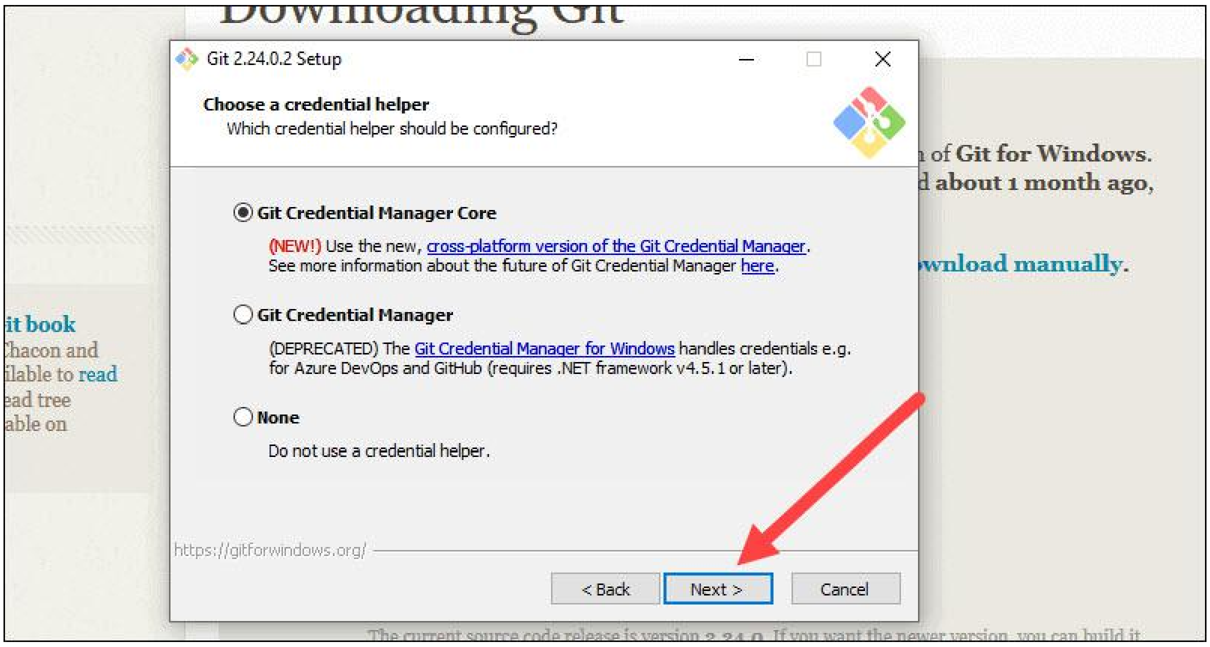
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

( ) **Only ever fast-forward**

Fast-forward to the fetched branch. Fail if that is not possible.

https://gitforwindows.org/

17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.


Git 2.24.0.2 Setup

Choose a credential helper

Which credential helper should be configured?

(●) **Git Credential Manager Core**

(NEW!) Use the new, cross-platform version of the Git Credential Manager. See more information about the future of Git Credential Manager here.

( ) **Git Credential Manager**

(DEPRECATED) The Git Credential Manager for Windows handles credentials e.g. for Azure DevOps and GitHub (requires .NET framework v4.5.1 or later).

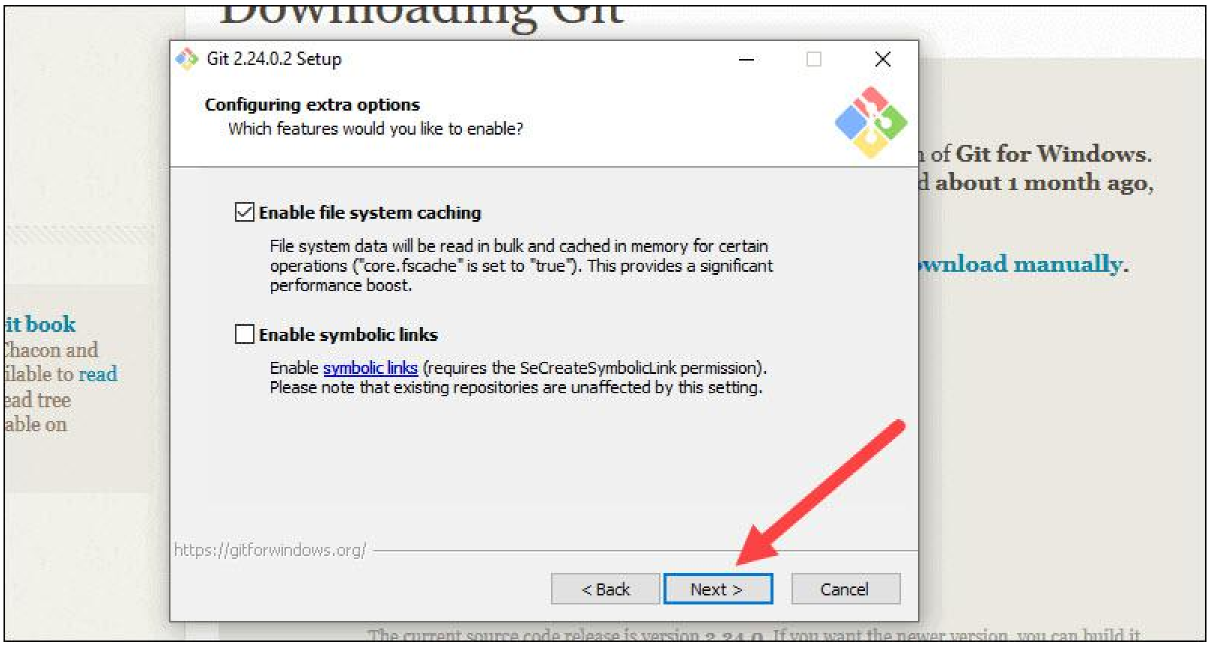
( ) **None**

Do not use a credential helper.

https://gitforwindows.org/

Additional Customization Options

18. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click **Next**.


Git 2.24.0.2 Setup

Configuring extra options

Which features would you like to enable?

[✓] **Enable file system caching**

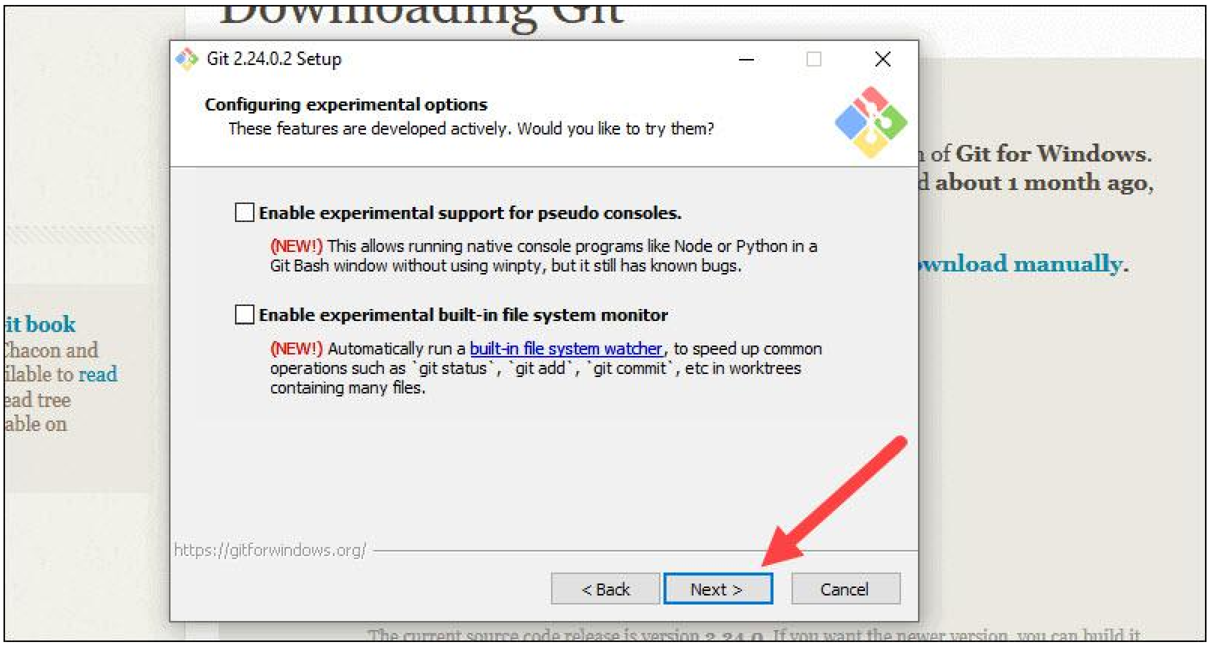
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

[ ] **Enable symbolic links**

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

https://gitforwindows.org/

19. Depending on the version of Git you're installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click **Install**.


Git 2.24.0.2 Setup

Configuring experimental options

These features are developed actively. Would you like to try them?

[ ] **Enable experimental support for pseudo consoles.**

(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

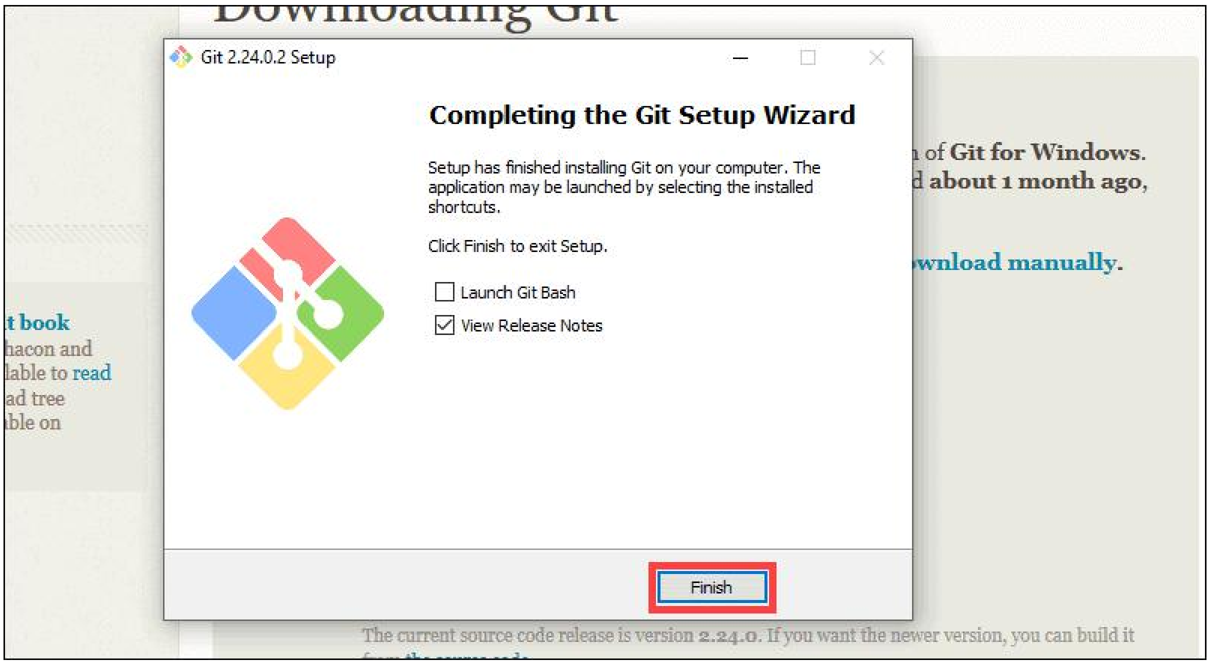
[ ] **Enable experimental built-in file system monitor**

(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.

https://gitforwindows.org/

Complete Git Installation Process

20. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click Finish**.**


Git 2.24.0.2 Setup

Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☐ Launch Git Bash

☑ View Release Notes

Experiment No. 02

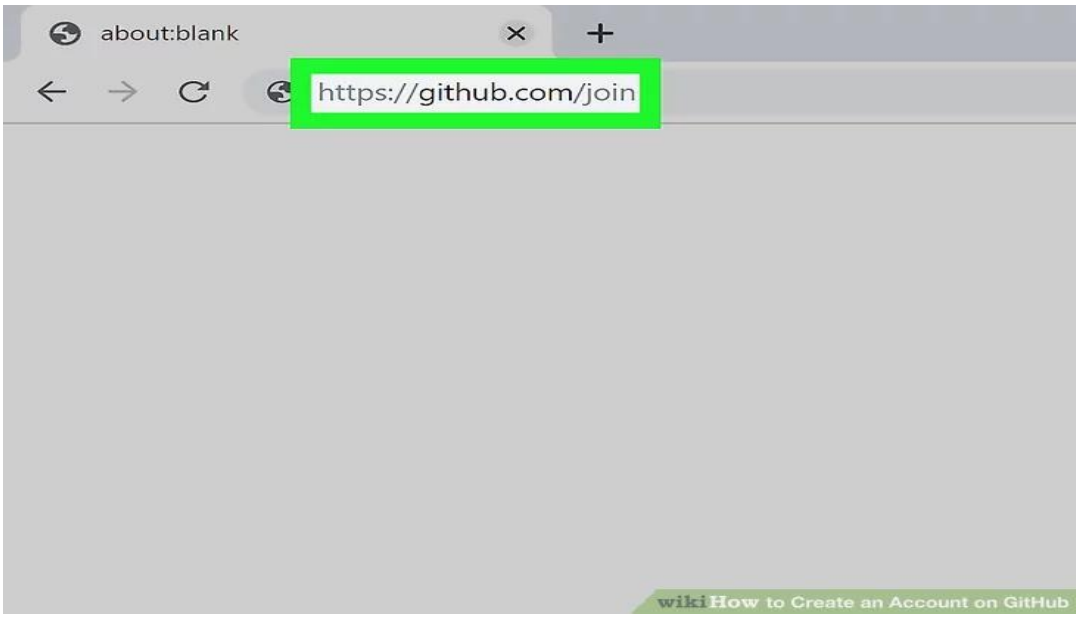
AIM:-Setting Up GitHub Account

Steps For Making an Account in Git Hub

1

God to https://github.com/join in a web browser. You can use any web browser on your computer, phone, or tablet to join.

- Some ad blockers, including u Block Origin, prevent GitHub's verification CAPTCHA puzzle from appearing. For best results, disable your web browser's ad blocker when signing up for GitHub.



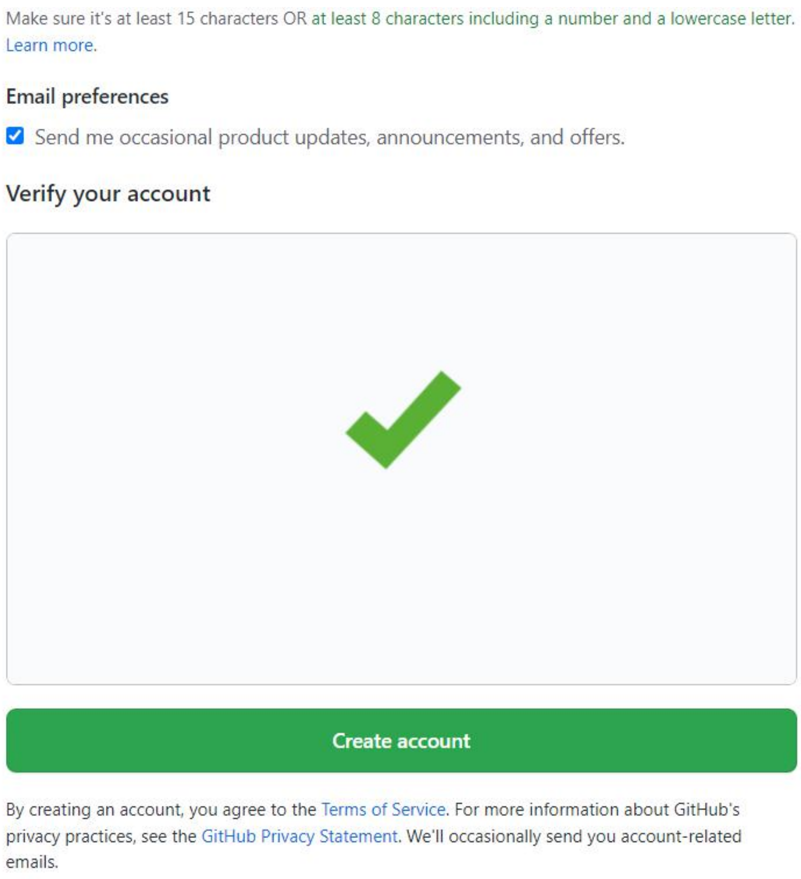
2

Enters your personal details. In addition to creating a username and entering an email address, you'll also have to create a password. Your password must be at least 15 characters in length *or* at least 8 characters with at least one number and lowercase letter.

- Carefully review the Terms of Service at https://help.github.com/en/articles/github-terms-of-service and the Privacy Statement at https://help.github.com/en/articles/github-privacy-statement before you continue. Continuing past the next step confirms that you agree to both documents.

3

Click the green Create an account **button.** It's below the form.


Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

Email preferences

- [x] Send me occasional product updates, announcements, and offers.

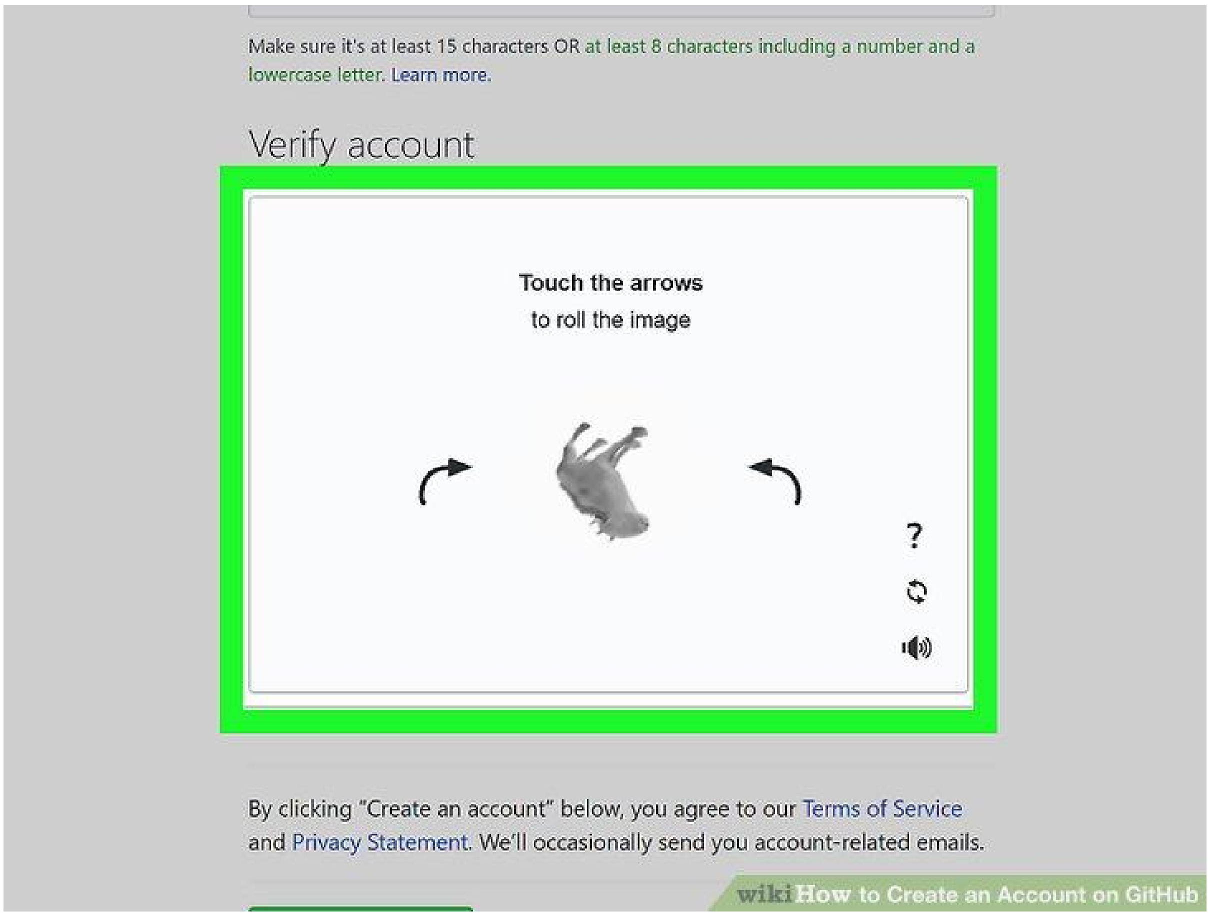
Verify your account

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

4

Complete the CAPTCHA puzzle. The instructions vary by puzzle, so just follow the on-screen instructions to confirm that you are a human.

- If you see an error that says "Unable to verify your captcha response," it's because your web browser's ad blocking extension prevented the CAPTCHA puzzle from appearing. Disable all ad-blocking extensions, refresh the page, and then click **VERIFY** to start the CAPTCHA.



Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

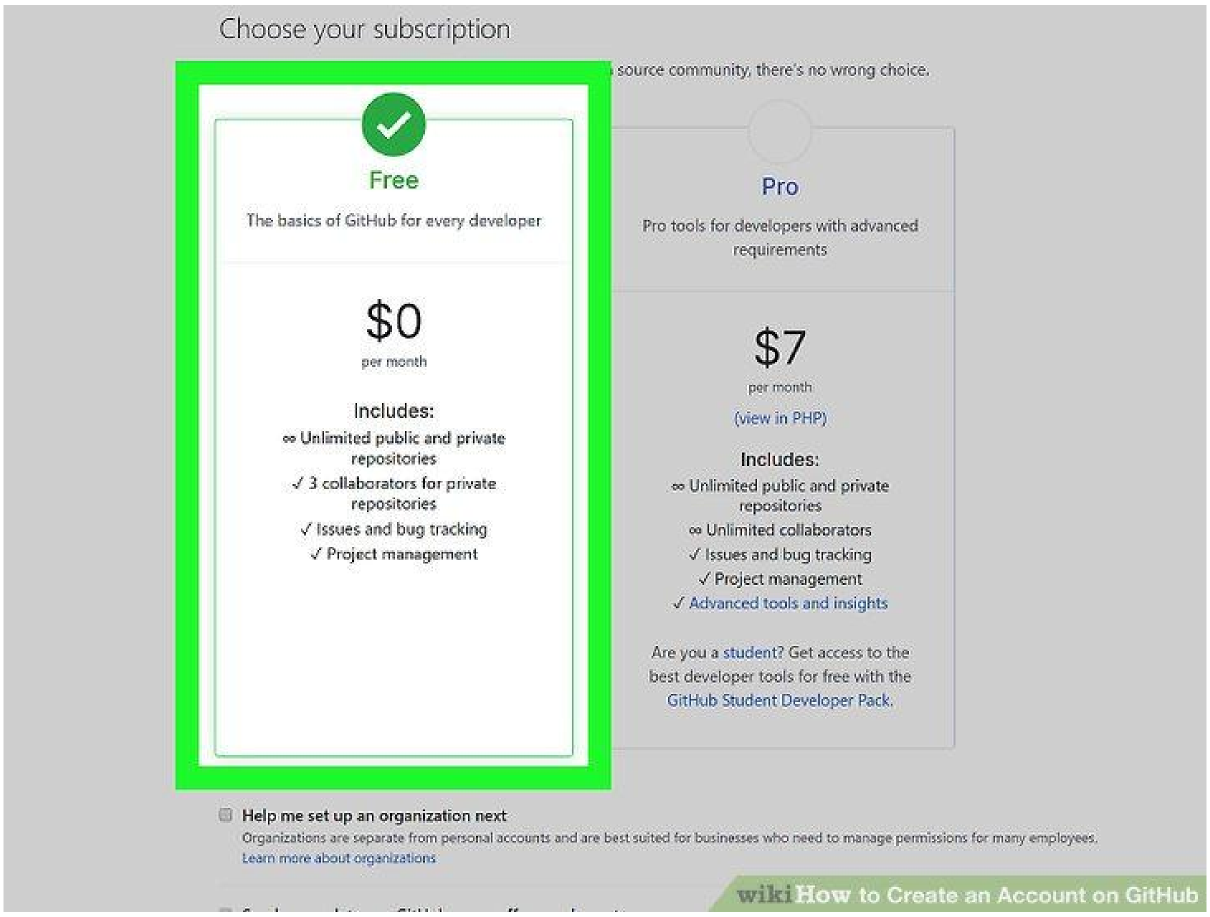
Verify account

By clicking "Create an account" below, you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account-related emails.

5

Click the Choose **button for your desired plan.** Once you select a plan, GitHub will send an email confirmation message to the address you entered. The plan options are:[2]

- **Free:** Unlimited public and private repositories, up to 3 collaborators, issues and bug tracking, and project management tools.
- **Pro:** Unlimited access to all repositories, unlimited collaborators, issue & bug tracking, and advanced insight tools.
- **Team:** All of the aforementioned features, plus team access controls and user management.
- **Enterprise:** All of the features of the Team plan, plus self-hosting or cloud hosting, priority support, single sign-on support, and more.


Choose your subscription

source community, there's no wrong choice.

Free

The basics of GitHub for every developer

$0
per month

Includes:
∞ Unlimited public and private repositories
✓ 3 collaborators for private repositories
✓ Issues and bug tracking
✓ Project management

Pro

Pro tools for developers with advanced requirements

$7
per month
(view in PHP)

Includes:
∞ Unlimited public and private repositories
∞ Unlimited collaborators
✓ Issues and bug tracking
✓ Project management
✓ Advanced tools and insights

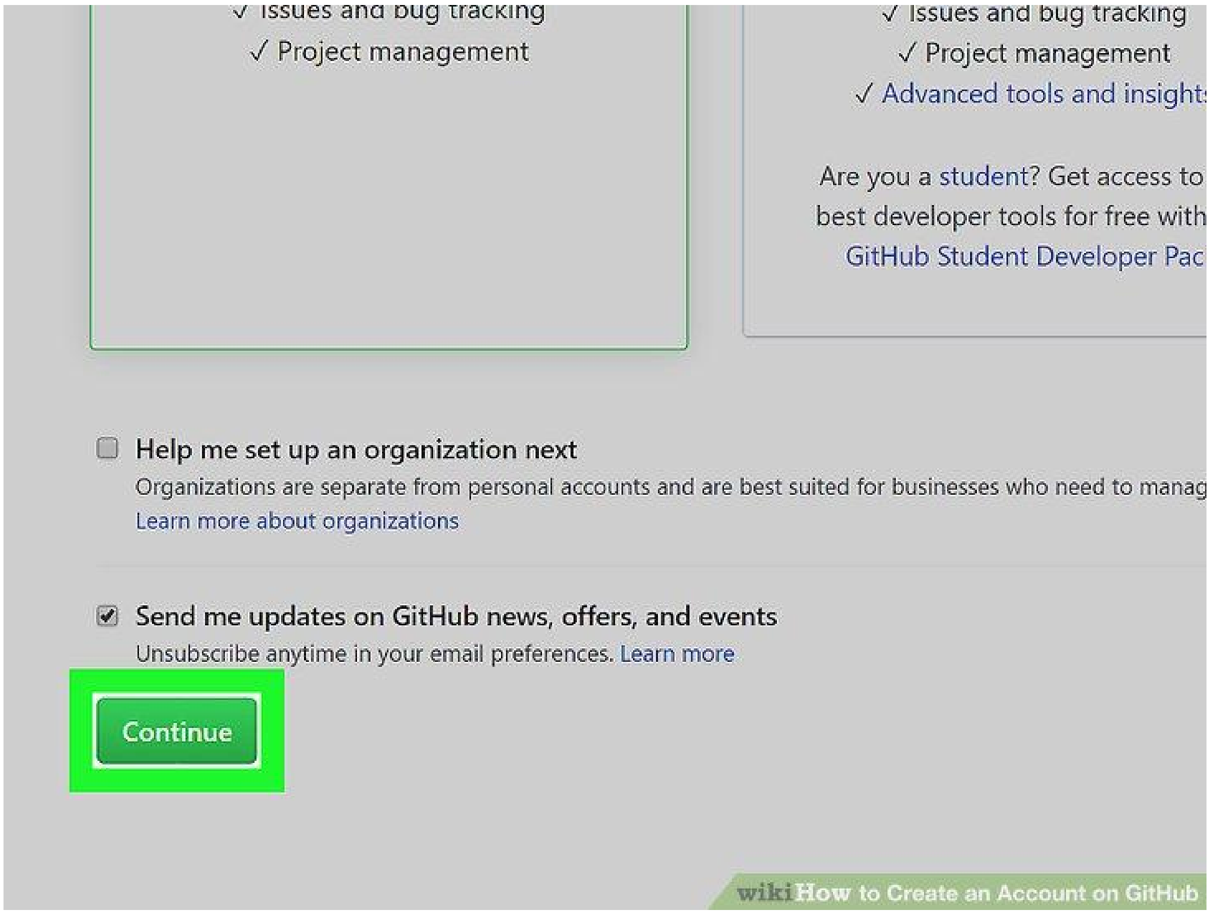
Are you a student? Get access to the best developer tools for free with the GitHub Student Developer Pack.

Help me set up an organization next
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
Learn more about organizations

6

Review your plan selection and click Continue**.** You can also choose whether you want to receive updates from GitHub via email by checking or unchecking the "Send me updates" box.

- If you chose a paid plan, you'll have to enter your payment information as requested before you can continue.



✓ Issues and bug tracking
✓ Project management

✓ Issues and bug tracking
✓ Project management
✓ Advanced tools and insight

Are you a student? Get access to
best developer tools for free with
GitHub Student Developer Pac

☐ Help me set up an organization next
Organizations are separate from personal accounts and are best suited for businesses who need to manag
Learn more about organizations

☑ Send me updates on GitHub news, offers, and events
Unsubscribe anytime in your email preferences. Learn more

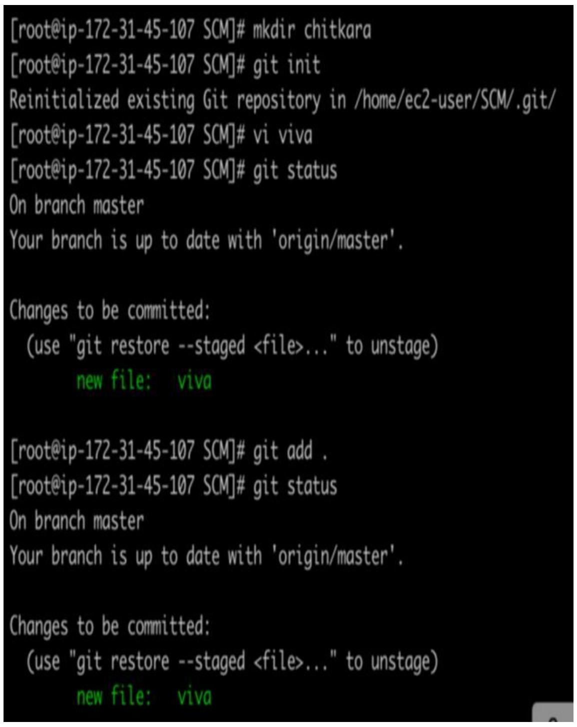
7

Select your preferences and click Submit**.** GitHub displays a quick survey that can help you tailor your experience to match what you're looking for. Once you make

AIM:-how to use Git Log

git status

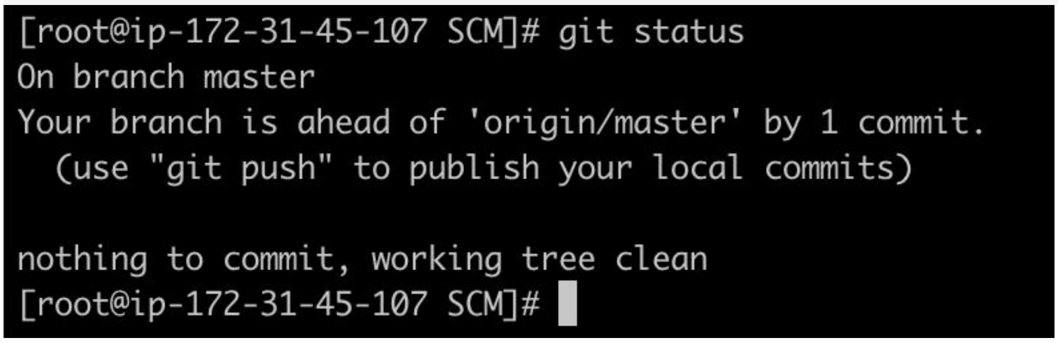
The `git status` command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does *not* show you any information regarding the committed project history. For this, you need to use `git log`.



```
[root@ip-172-31-45-107 SCM]# mkdir chitkara
[root@ip-172-31-45-107 SCM]# git init
Reinitialized existing Git repository in /home/ec2-user/SCM/.git/
[root@ip-172-31-45-107 SCM]# vi viva
[root@ip-172-31-45-107 SCM]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   viva

[root@ip-172-31-45-107 SCM]# git add .
[root@ip-172-31-45-107 SCM]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   viva
```

The `git add` command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, `git add` doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit.

In conjunction with these commands, you'll also need git status to view the state of the working directory and the staging area.



```
[root@ip-172-31-45-107 SCM]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[root@ip-172-31-45-107 SCM]#
```

Git Commit

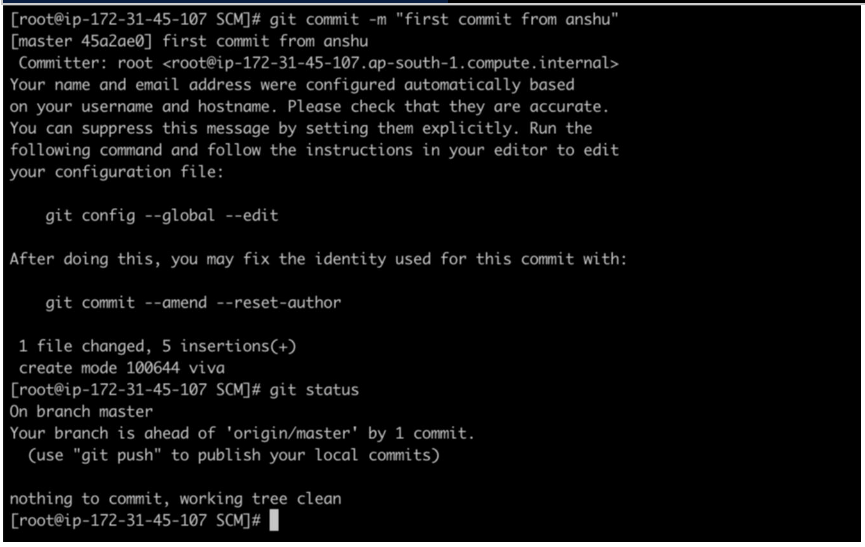
Since we have finished our work, we are ready move from stage to commit for our repo.

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

When we commit, we should **always** include a **message**.

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.

```
git commit –m "commit message"
```

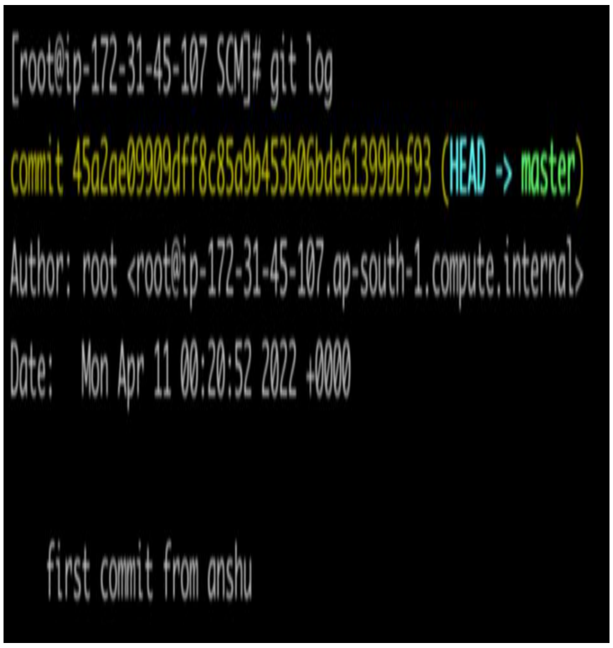


```
[root@ip-172-31-45-107 SCM]# git commit -m "first commit from anshu"
[master 45a2ae0] first commit from anshu
 Committer: root <root@ip-172-31-45-107.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 5 insertions(+)
 create mode 100644 viva
[root@ip-172-31-45-107 SCM]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[root@ip-172-31-45-107 SCM]#
```

The git log Command

The git log command shows a list of all the commits made to a repository. You can see the hash of each Git commit, the message associated with each commit, and more metadata. This command is useful for displaying the history of a repository.

Whereas the git status command is focused on the current working directory, git log allows you to see the history of your repository.



```
[root@ip-172-31-45-107 SCM]# git log
commit 45a2ae0909dff8c85a9b453b06bde61399bbf93 (HEAD -> master)
Author: root <root@ip-172-31-45-107.ap-south-1.compute.internal>
Date:   Mon Apr 11 00:20:52 2022 +0000


    first commit from anshu
```

Experiment No. 04

AIM:-Create and Visualize branch

How it works

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history. New commits are recorded in the history for the current branch, which results in a fork in the history of the project.

The `git branch` command lets you create, list, rename, and delete branches. It doesn’t let you switch between branches or put a forked history back together again. For this reason, `git branch` is tightly integrated with the git checkout and git merge commands.

Common Options

```
git branch
```

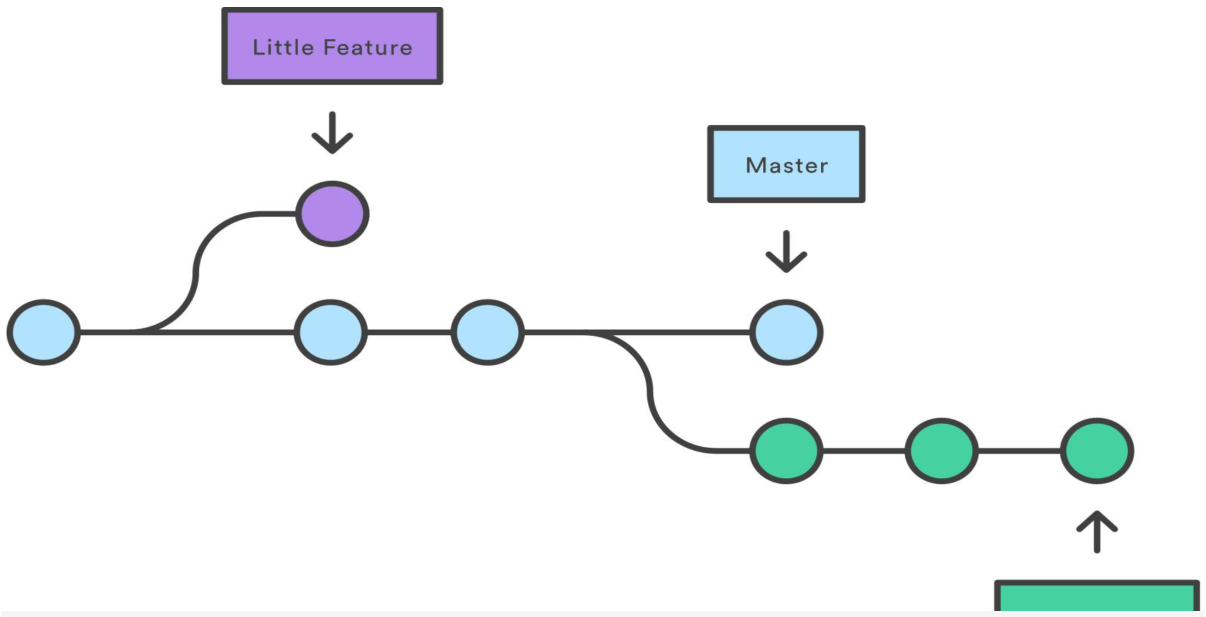
List all of the branches in your repository. This is synonymous with `git branch --list`.

```
git branch <branch>
```

Create a new branch called `<branch>`. Thisdoes *not* check out the new branch.

Creating Branches

It's important to understand that branches are just pointers to commits. When you create a branch, all Git needs to do is create a new pointer, it doesn’t change the repository in any other way. If you start with a repository that looks like this:

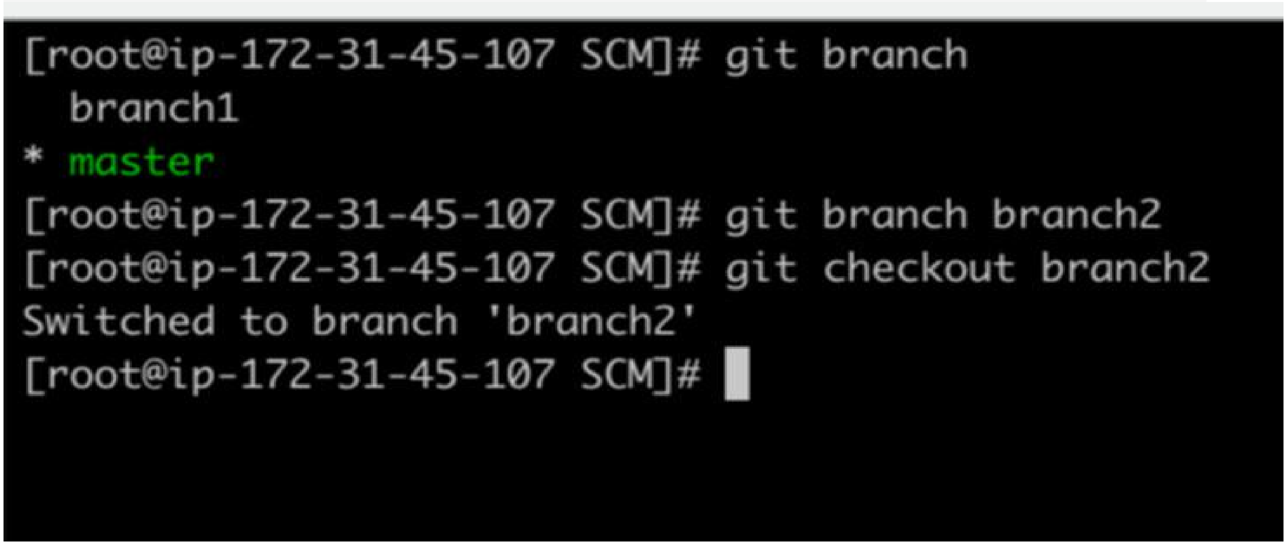


Then, you create a branch using the following command:

```
gitbranch crazy-experiment
```

The repository history remains unchanged. All you get is a new pointer to the current commit:

Note that this only *creates* the new branch. To start adding commits to it, you need to select it with `git checkout`, and then use the standard `git add` and `git commit` commands.



```
[root@ip-172-31-45-107 SCM]# git branch
  branch1
* master
[root@ip-172-31-45-107 SCM]# git branch branch2
[root@ip-172-31-45-107 SCM]# git checkout branch2
Switched to branch 'branch2'
[root@ip-172-31-45-107 SCM]# 
```

Merging into Master — Fun Part

This is the only step that is different from the steps followed in the rebase article.

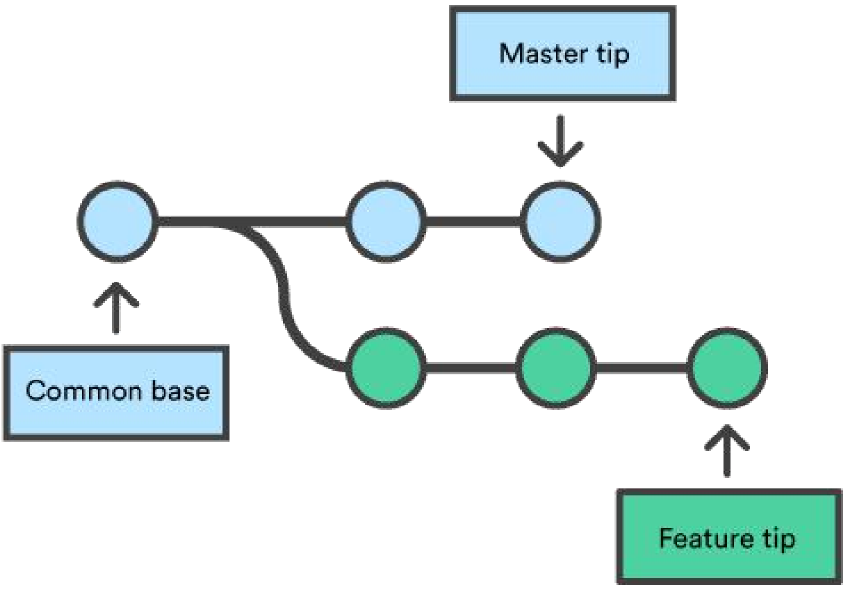
Note that the changes are added to the current branch from the selected branch when doing a merge. Therefore we should first move to the **master** branch to start the merge.

```
git checkout master
```

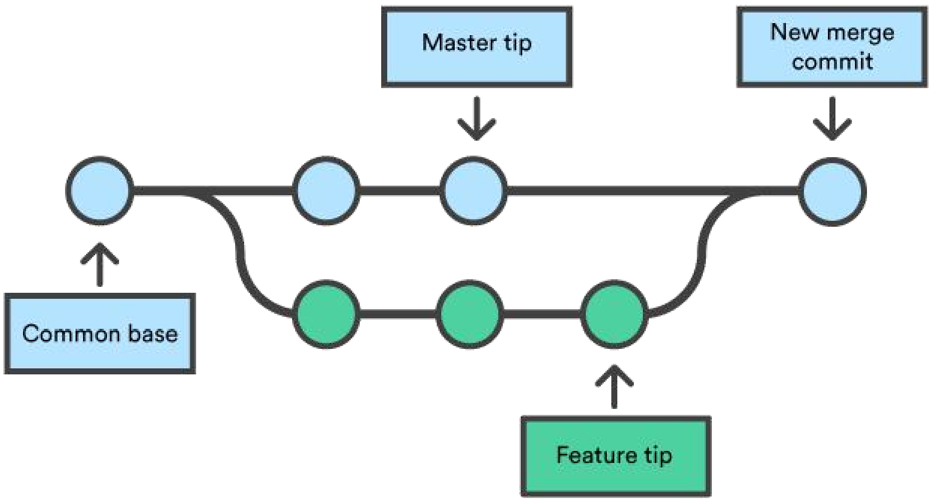
Now you can use the following command to move the changes from **dev** to **master**.

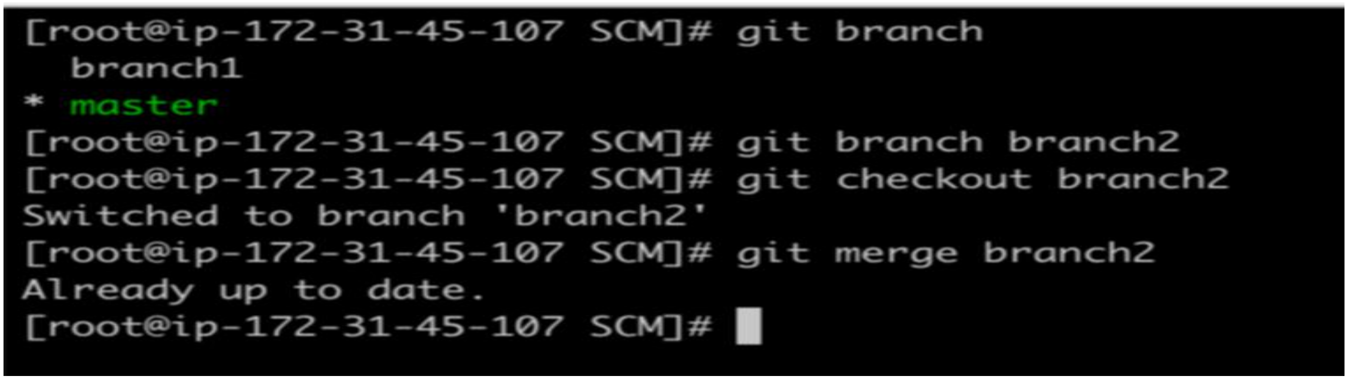
```
git merge <Branch name>
```

When we use this command, Git will find a common commit between the 2 branches and create a new *merge commit* on the **master** combining the queued changes in the **dev** branch.



Before Merge(Image from Git Documentation)





```
[root@ip-172-31-45-107 SCM]# git branch
  branch1
* master
[root@ip-172-31-45-107 SCM]# git branch branch2
[root@ip-172-31-45-107 SCM]# git checkout branch2
Switched to branch 'branch2'
[root@ip-172-31-45-107 SCM]# git merge branch2
Already up to date.
[root@ip-172-31-45-107 SCM]# 
```

AIM:- Git Life Cycle Description

Introduction to Git Life Cycle

Git is one of the premier distributed version control systems available for programmers and corporates. In this article, we will see details about how a project that is being tracked by git proceeds with workflow i.e Git Life Cycle. As the name suggests is regarding different stages involved after cloning the file from the repository. It covers the git central commands or main commands that are required for this particular version control system

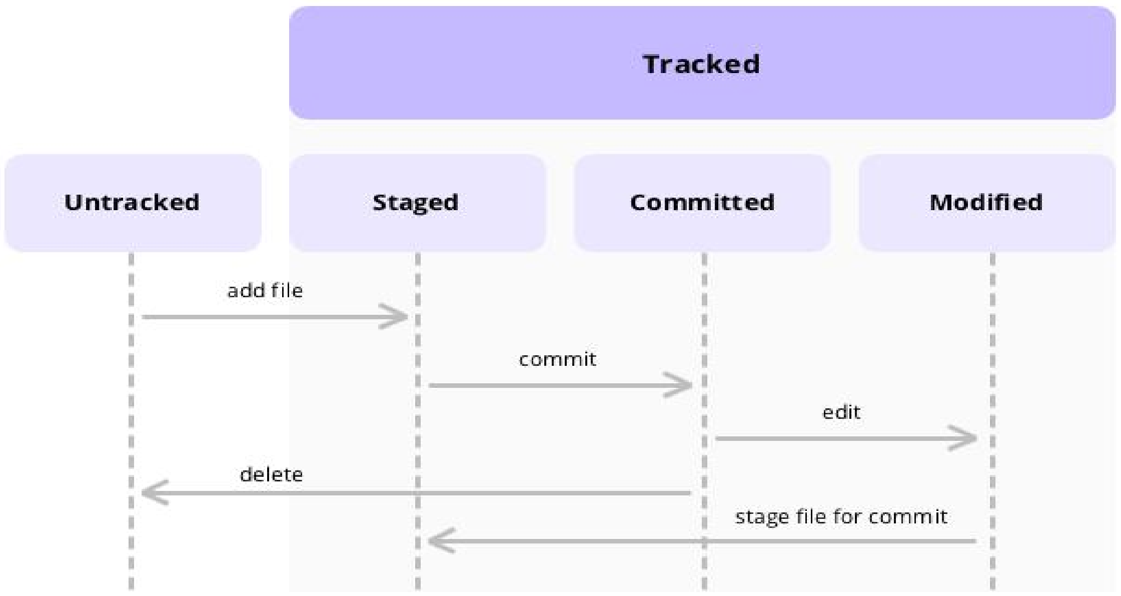
Workflow of Git Life Cycle

The workflow of the Git as follows:

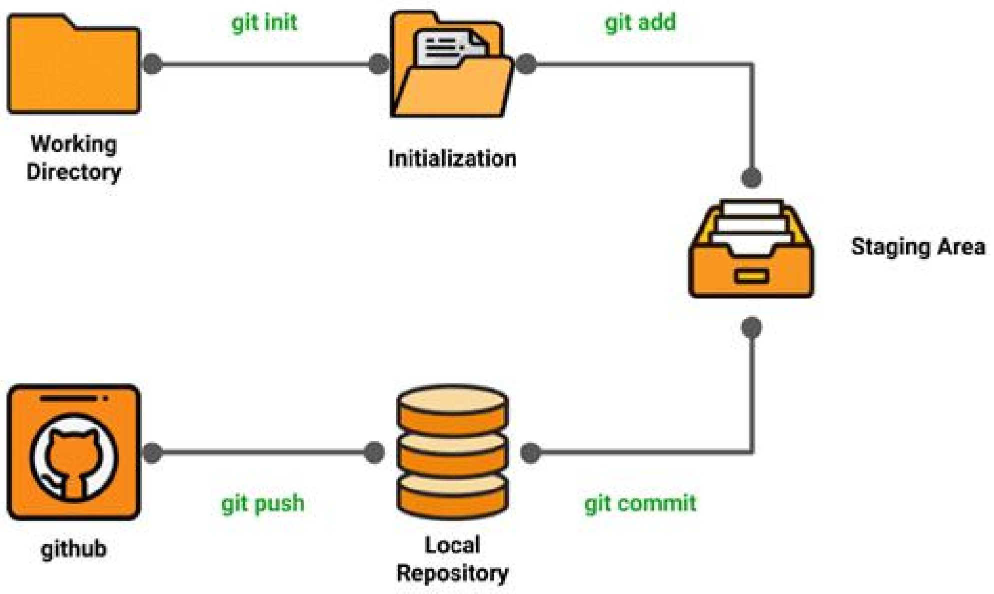
- We will create a branch on which we can work on and later we will merge it with master

- Clone: First, when we have code present in the remote repository, we clone to local to form something called a local repository.

- Modifications/Adding Files: we perform several developments on the existing files or may as well add new files. Git will monitor all these activities and will log them.



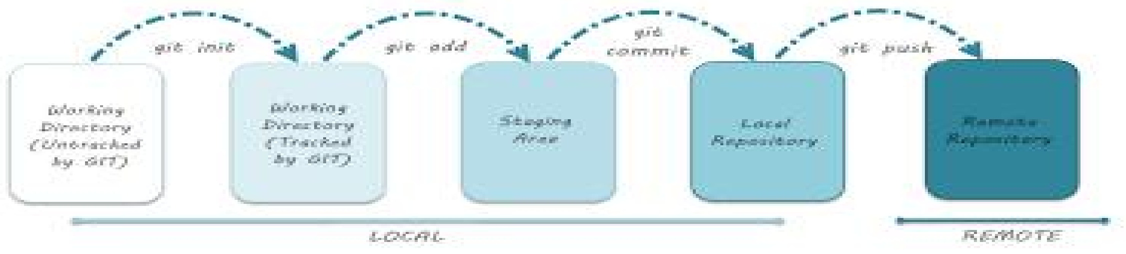
- We need to move the content that we require to transform to the master to the staging area by using git commands and the snapshot of staged files will be saved in the git staging area.
- We need to perform commits on the files that are staged and the recorded snapshot from the above steps will be permanently saved on the local repo and this particular is recorded by commit message for future referrals.



- Once we commit the code is available on the local repo but to send it to the master repo we need to perform PUSH operation
- If someone else is working on the same branch then there will be a possibility that he might have added his changes to the master by push. So we need to perform PULL operation before the PUSH operation if multiple people are working on the same branch and this workflow as shown below.

- Once the target branch is updated we need to get all the required approvals so that merge operation with the master is allowed.

This is the basic workflow of git was lots of intermediate commands like git add, git status, git commit, git push origin, git rebase, git merge, git diff, etc will be used depending upon the requirement of the user.



Stages of Git Life Cycle

So we have seen the workflow of the git life cycle above. But we need to know that we have a project linked with git then that project can reside in there of the following areas. Below mentioned areas are ingredients to the recipe of Git and having an idea of them will help you a lot to track the files that you are working on.

The stages are as discussed:

- Working Directory
- Staging Area
- Git Directory

These Three Stages are explained below:

1. Working Directory

- If you have your project residing on to your local machines then basically it is called even though it is linked to git or not. In either case, it will be called as the working directory. But when the available project is linked with git then basically there will be .git folder hidden in the so-called working directory. So the presence of the .git folder is enough to say that the folder is working copy on the machine and it is tracked by the git.
- At this stage, git knows what are the files and folders that it's tracking that's it. No other info will be available regarding this. To make sure that the newly added files get tracked in the working copy we need to make sure that those files are staged and this is our second residence for the files.

2. Staging Area

- When we make changes to the existing files in the working repo or if we add any folder of files and if we want these changes to need to be tracked and also need to be moved to the local repo for tracking then we need to move these changed files or newly added folder of file to the staging area. Git add is the basic command which will be used to move the modified files to the staged area.
- It's ticked that been give to modified files or newly added folder of file to travel to the local repo for further traction. Those files that don't have that ticket will be tracked by the git but they won't be able to move to the target easily. Here index plays a critical role. GIT Index is something that comes in between local repo and working directory and it is the one that decides what

needs to be sent to the local repo and in fact, it decides what needs to be sent to the central repo.

3. GIT Directory

- When we have done the modifications or addition of files or folder and want them to be part of the repository they first we do is to move them to the staging area and they will commit ready. When we commit then provide the appropriate commit message and files will be committed and get updated in the working directory.
- Now git tracks the commits and commit messages and preserves the snapshot of commit files and this is done in the Git specific directory called Git Directory. Information related to all the files that were committed and their commit messages will be stored in

this directory. We can say that this git directory stores the

metadata of the files that were committed.