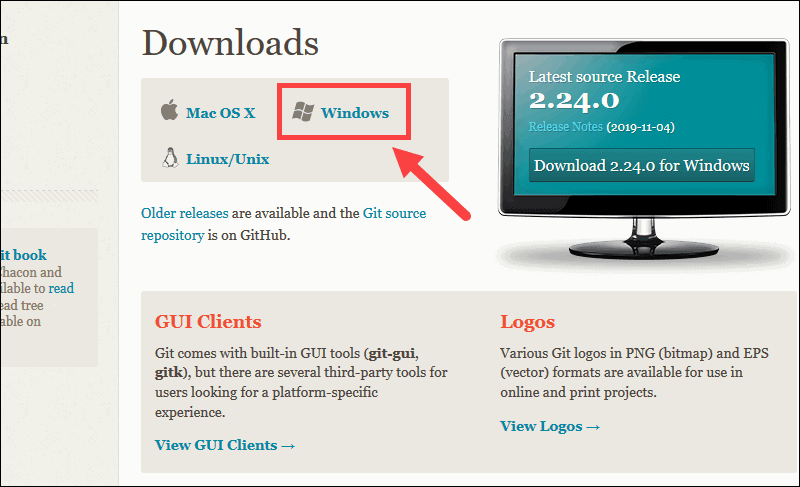
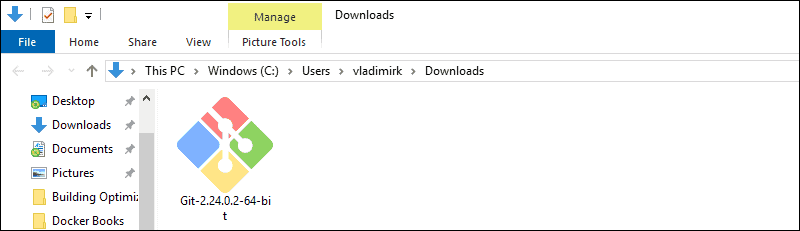
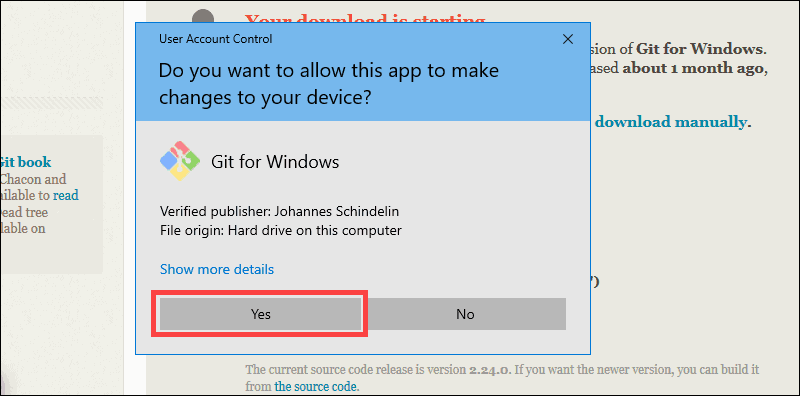
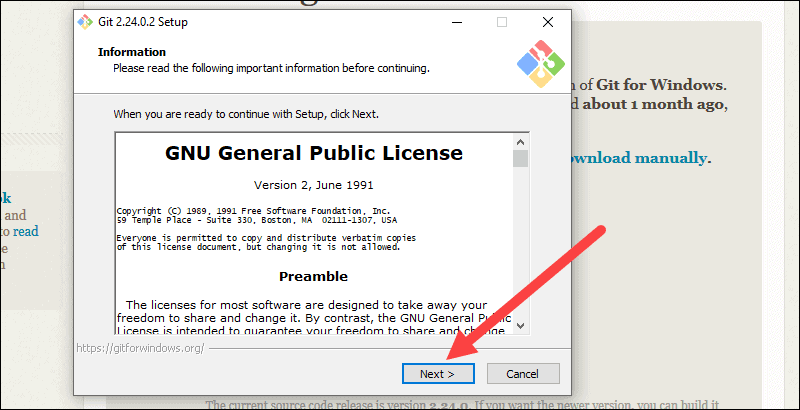
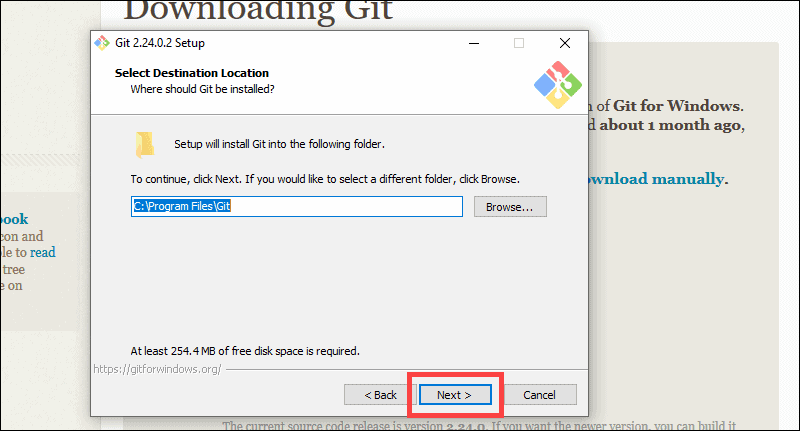
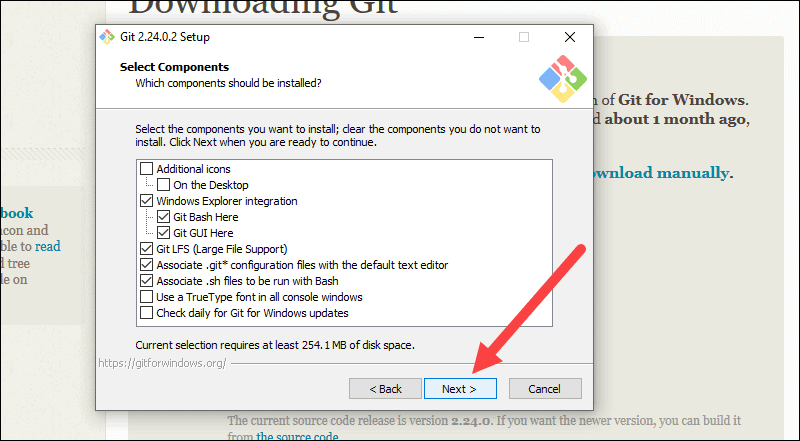
### **Experiment No 1 AIM:** Setting up the git client. Download Git for Windows

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.  


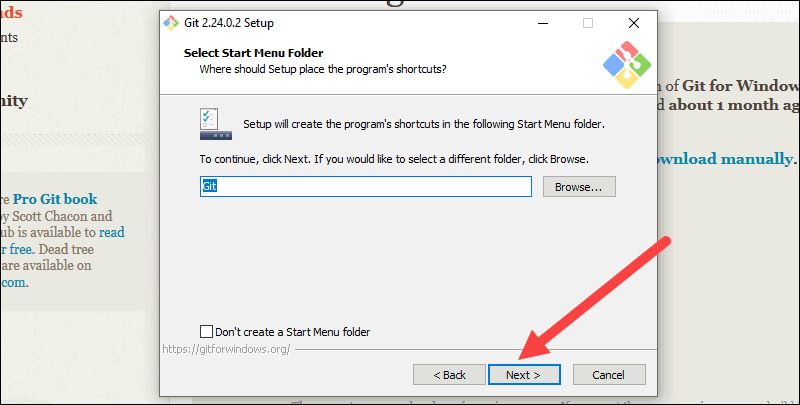
### Extract and Launch Git Installer

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

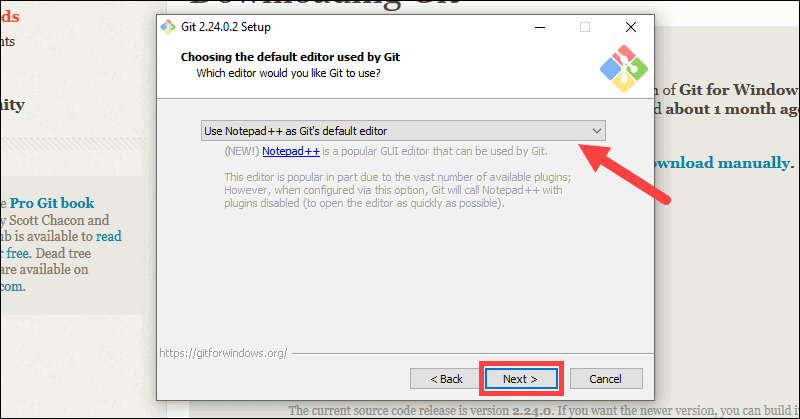


4. Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.  
****5. Review the GNU General Public License, and when you’re ready to install, click **Next**.  
****6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.  
****7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.  
 ****

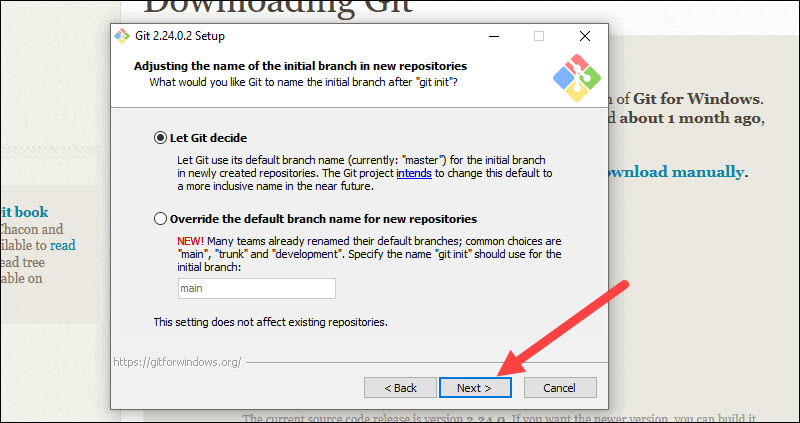
8. The installer will offer to create a start menu folder. Simply click **Next**.



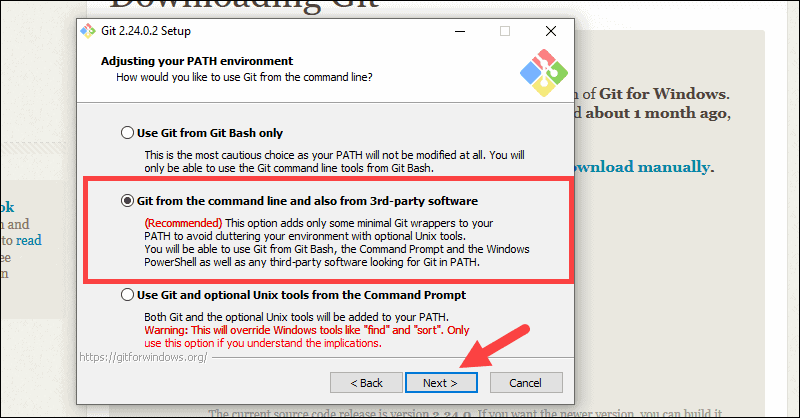
9. Select a text editor you’d like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.



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.**

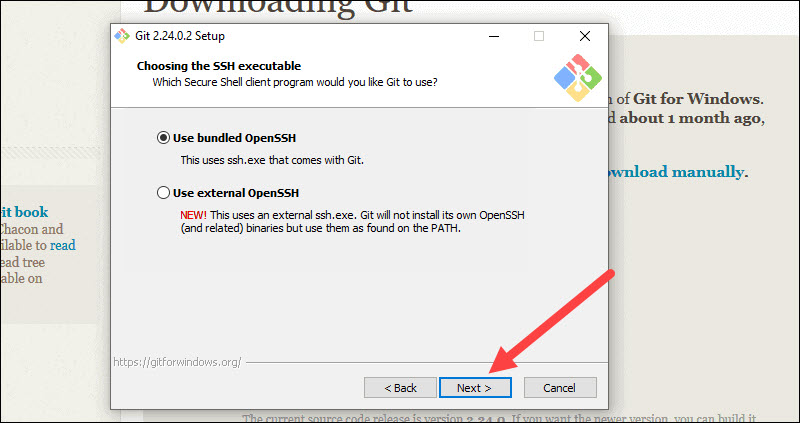


11. This installation step allows you to change the **PATH environment**. The **PATH**is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.

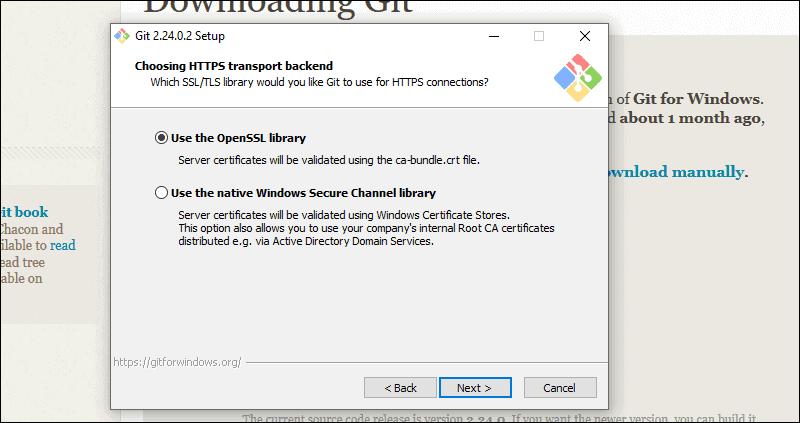


### Server Certificates, Line Endings and Terminal Emulators

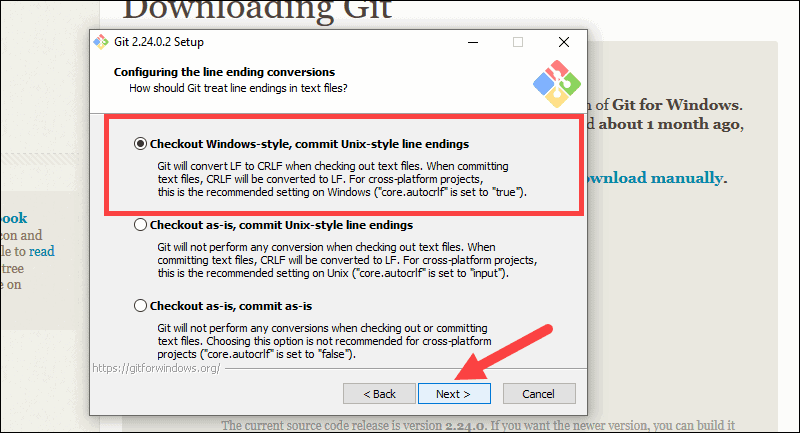
12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**



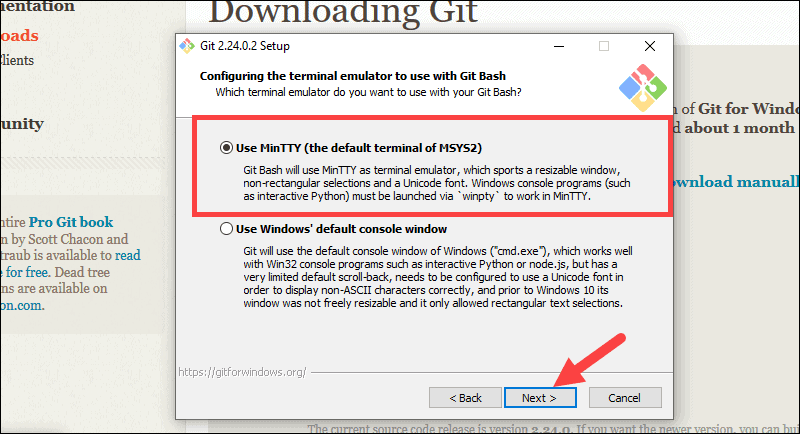
13. The next option relates to server certificates. Most users should use the default. If you’re working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.



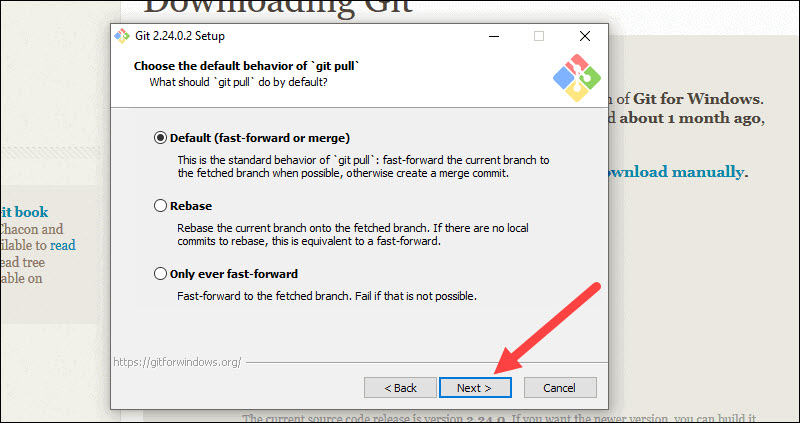
14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.



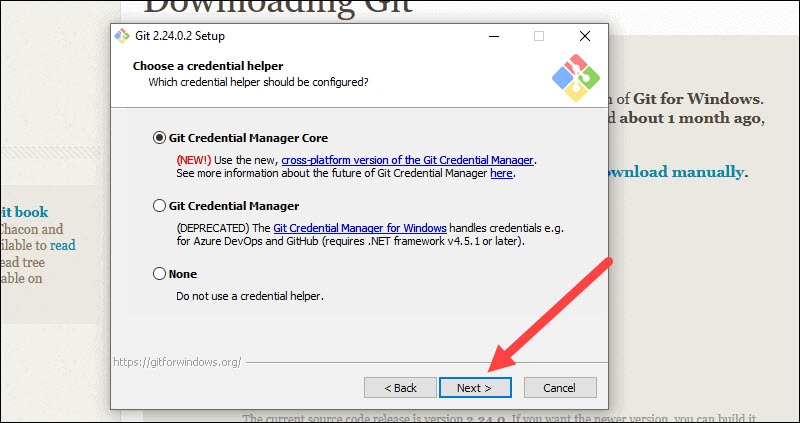
15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.



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.

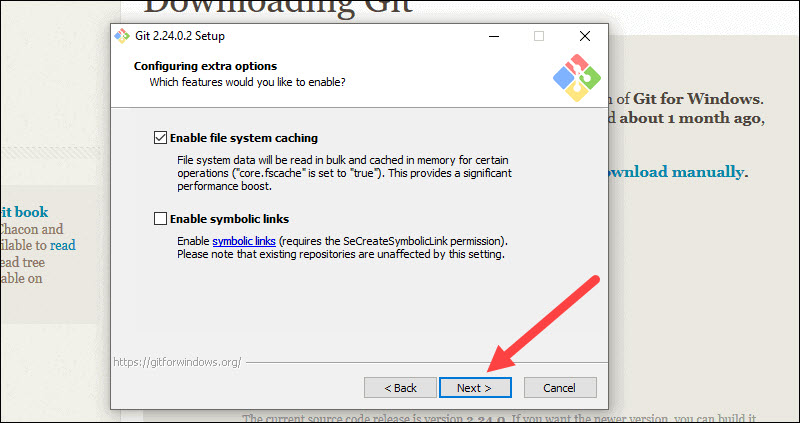


17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.

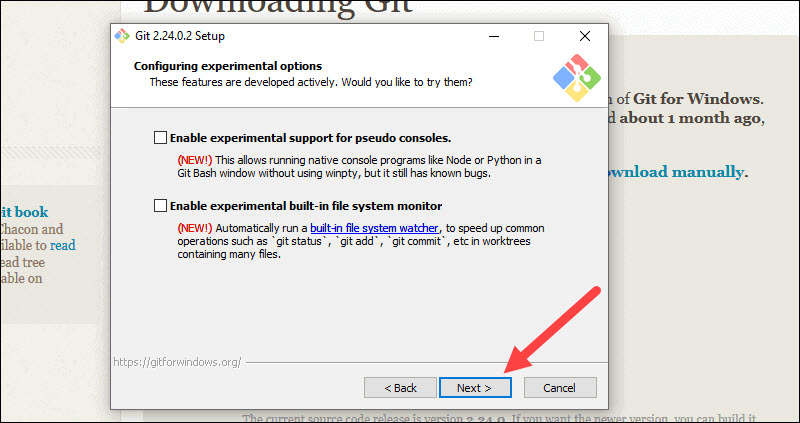


### Additional Customization Options

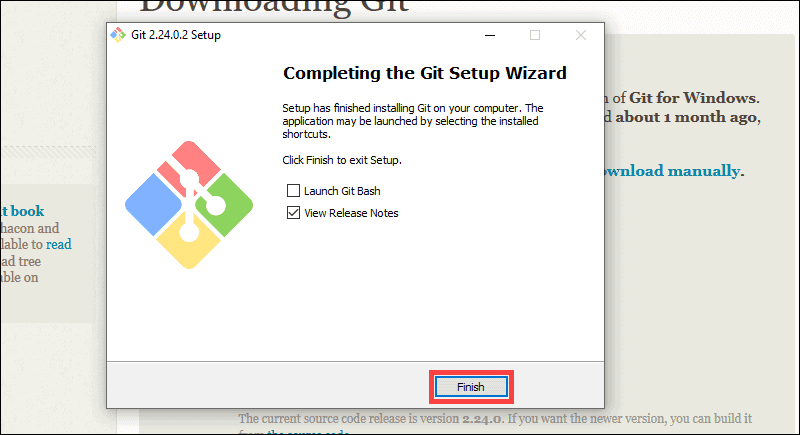
18. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click **Next**.



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**.



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

 **Expermient No 2**Aim**:** Setting up GitHub Account   
  
**Go to**[**https://github.com/join**](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 uBlock 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**

**Enter 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.[[1]](https://www.wikihow.com/Create-an-Account-on-GitHub#_note-1)

* + 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.

**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](https://www.wikihow.com/Disable-Your-Ad-Blocker), refresh the page, and then click **VERIFY** to start the CAPTCHA.

**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]](https://www.wikihow.com/Create-an-Account-on-GitHub#_note-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.

**6**

**Click the Verify email address button in the message from GitHub.** This confirms your email address and returns you to the sign-up process.

**7**

**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.

**8**

**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 your selection, you'll be taken to a screen that allows you to set up your first repository.

**Expermient No 3**Aim: Program to generate logs  
  
Basic Git init  
  
Git init command creates a new Git repository. It can be used to convert an existing, undersigned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialize repository, so this is usually the first command you’ll run in a new project.  
  
Basic 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.  
  
Basic Git commit  
  
The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used  
  
Basic Git add command  
 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  
  
Basic Git log Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It will use as:

**Experiment No 4**Aim: Create And Visulaize Branches In Git

## Create a New Git Branch

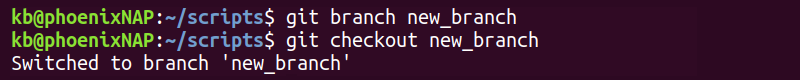
There are many ways to create a new Git branch. In most cases it comes down to whether you are creating a branch from the main branch or, for example, a new commit or tag.

One common method of creating a new branch is with the command:

git branch <new\_branch\_name>

This doesn’t automatically switch to that branch. To [switch Git branches](https://phoenixnap.com/kb/git-switch-branch), enter the following command:

git checkout <new\_branch\_name>



**Note:** Instead of **<new\_branch\_name>** type the name for the new branch.

### Create New Git Branch From Current Branch

The easiest and most popular way of creating a Git branch is:

git checkout -b <new\_branch\_name>

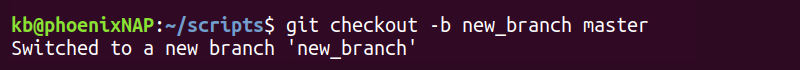
git checkout -b new branch

This creates a new branch from the current branch. It also automatically switches to the new branch.

### Create New Git Branch From a Different Branch

To create a new branch from a different branch, run the following command:

git checkout -b <new\_branch\_name> <specific\_different\_branch>



Instead of **<new\_branch\_name>** type the name for the new branch, and instead of **<specific\_different\_branch>** type the name of the existing branch from which the new one shall be created.

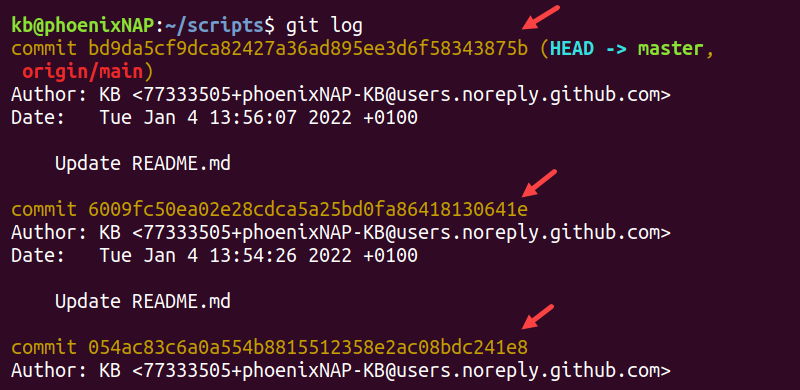
### Create a Branch from a Commit

A commit is a command that saves the changes made in the code. A project may have multiple commits as it's revised and improved.

Find the hash key for a specific commit:

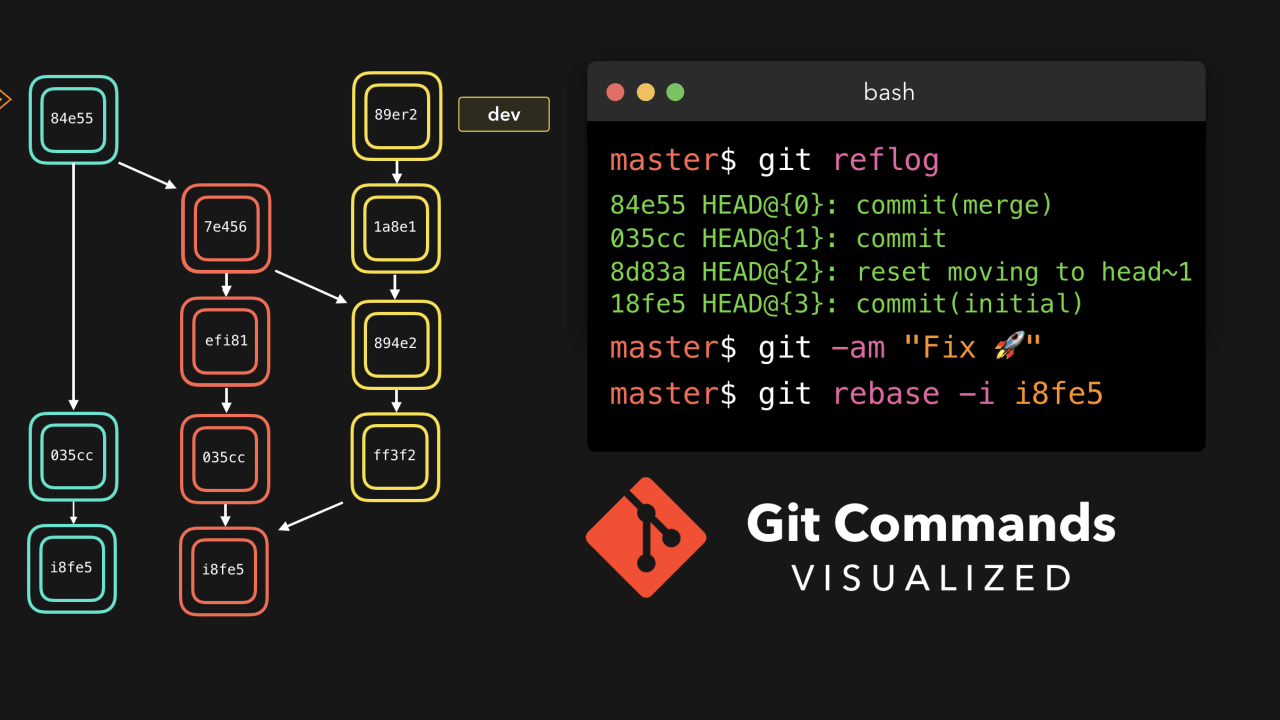
git log

The log contains the hash key.



Create a branch from an older commit:

git branch <new\_branch\_name> 6009fc

Visualizing Branches:  
 To visualize, we have to create a new file in the new branch “activity1” instead of the master branch. After this we have to do three step architecture i.e. working directory, staging area and git repository. After this I have done the 3 Step architecture which is tracking the file, send it to stagging area and finally we can rollback to any previously saved version of this file. After this we will change the branch from activity1 to master, but when we switch back to master branch the file we created i.e “hello” will not be  
there. Hence the new file will not be shown in the master branch. In this way we can create and change different branches. We can also merge the branches by using the git merge command. In this way we can create and change different branches. We can also merge the branches by using git merge command.  
 ****

**Expermient No 5**Aim: Git Lifecycle description

Git is used in our day-to-day work, we use Git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Lifecycle description that git has and understand more about its life cycle. Let us see some of the basic steps that we have to follow while working with Git.

* You clone the Git repository as a working copy.
* You modify the working copy by adding/editing files.
* If necessary, you also update the working copy by taking other developers' changes.
* You review the changes before committing.
* You commit changes. If everything is fine, then you push the changes to the repository.
* After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.  
    
  

When a directory is made into a git repository, there are mainly 3 states which make up the essence of the Git Version Control System. The three states are:

* Working Directory
* Staging Area
* Git Directory

### **1. Working Directory**

Whenever we want to initialize our local project directory to make it a git repository, we use the **git init** command. After this command, git becomes aware of the files in the project although it doesn’t track the files yet. The files are further tracked in the staging area.

### **2. Staging Area**

Now, to track the different versions of our files we use the command ***git add*.** We can term a staging area as a place where different versions of our files are stored. ***git add*** command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don’t want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the ***.git*** folder inside the ***index*** file.

### **3. Git Directory**

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the ***git commit*** command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.  
  
