While, **Git is a popular distributed version control system**, which means that you can clone your repository. Thus you can get a complete copy of your entire history of that project. This means you can access all your commits.

**Git has more advantages than SVN**. It is much better for those developers who are not always connected to the master repository. Also, it is much faster than SVN.

## What is Git?

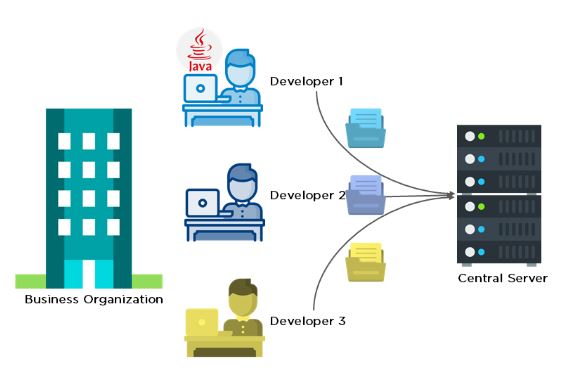
**Git** is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

# What is GitHub?

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface. GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

Before diving deep, let’s explain a scenario before Git:

* Developers used to submit their codes to the central server without having copies of their own
* Any changes made to the source code were unknown to the other developers
* There was no communication between any of the developers



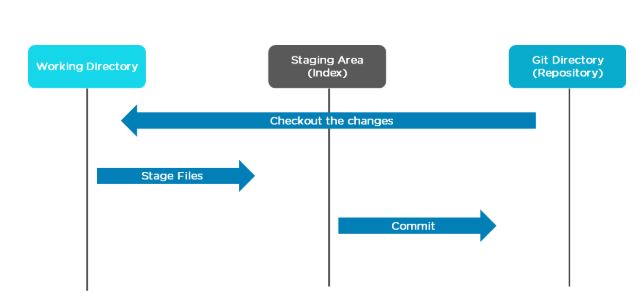
Now let’s look at the scenario after Git:

* Every developer has an entire copy of the code on their local systems
* Any changes made to the source code can be tracked by others
* There is regular communication between the developers



The Git workflow is divided into three states:

* Working directory - Modify files in your working directory
* Staging area (Index) - Stage the files and add snapshots of them to your staging area
* Git directory (Repository) - Perform a commit that stores the snapshots permanently to your Git directory. Checkout any existing version, make changes, stage them and commit.



# How to Install Git on Windows

To use Git, you have to install it on your computer. Even if you have already installed Git, it's probably a good idea to upgrade it to the latest version. You can either install it as a package or via another installer or download it from its official site.

Now the question arises that how to download the Git installer package. Below is the stepwise installation process that helps you to download and install the Git.

## How to download Git?

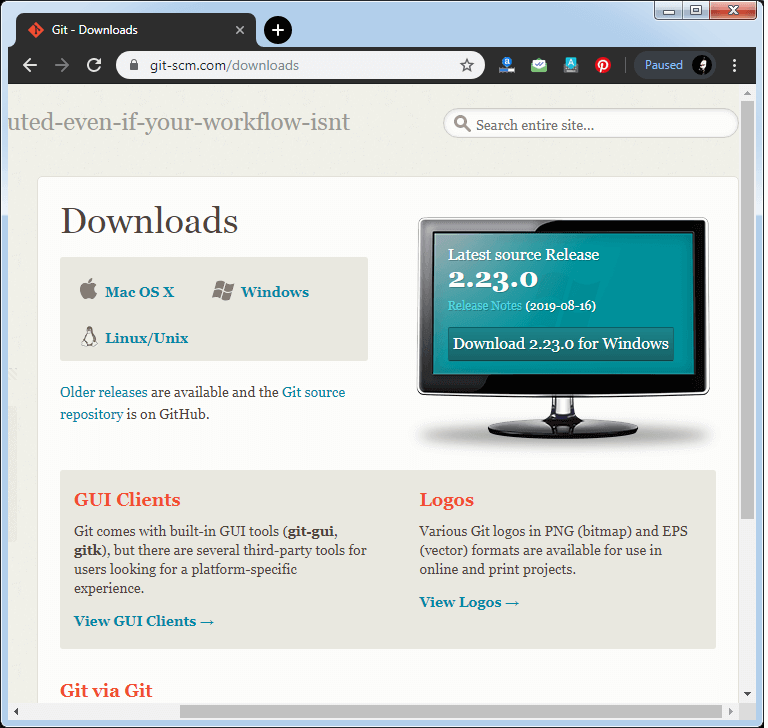
**Step1**

To download the Git installer, visit the Git's official site and go to download page. The link for the download page is <https://git-scm.com/downloads>. The page looks like as

64.9M

1.2K

C++ vs Java



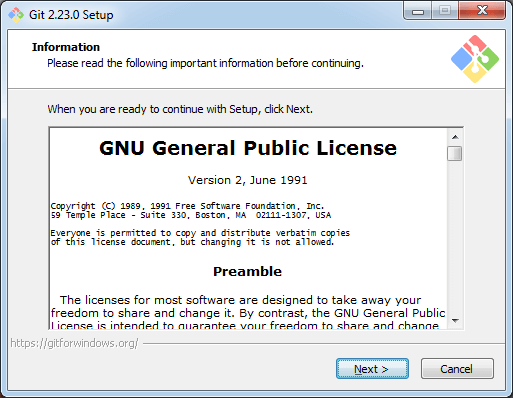
Click on the package given on the page as **download 2.23.0 for windows**. The download will start after selecting the package.

Now, the Git installer package has been downloaded.

## Install Git

**Step2**

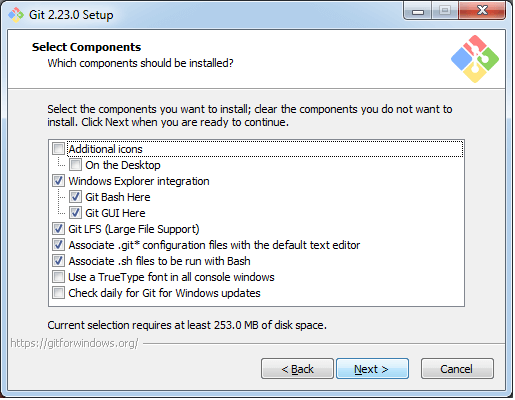
Click on the downloaded installer file and select **yes** to continue. After the selecting **yes** the installation begins, and the screen will look like as



Click on **next** to continue.

**Step3**

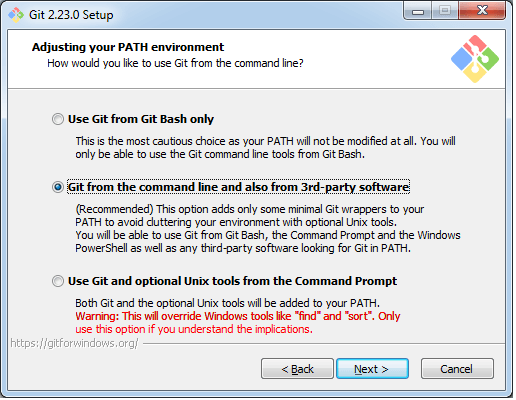
Default components are automatically selected in this step. You can also choose your required part.



Click next to continue.

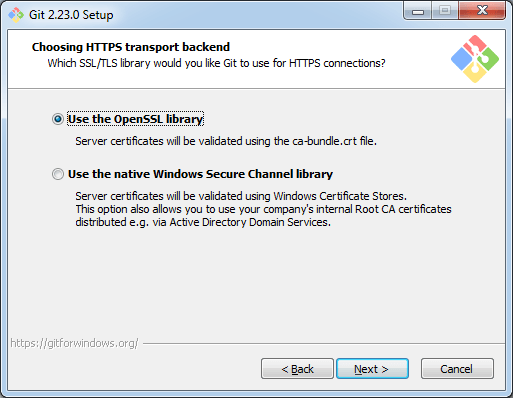
**Step4**

The default Git command-line options are selected automatically. You can choose your preferred choice. Click **next** to continue.



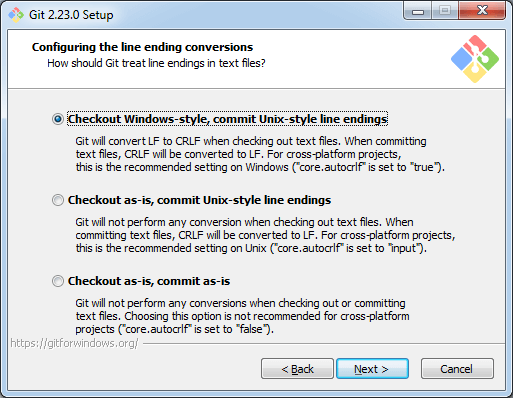
**Step5**

The default transport backend options are selected in this step. Click **next** to continue.



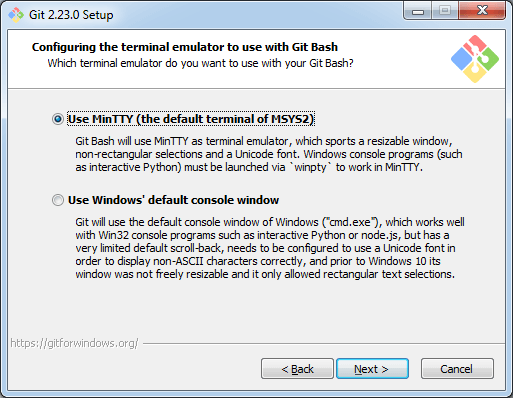
**Step6**

Select your required line ending option and click next to continue.



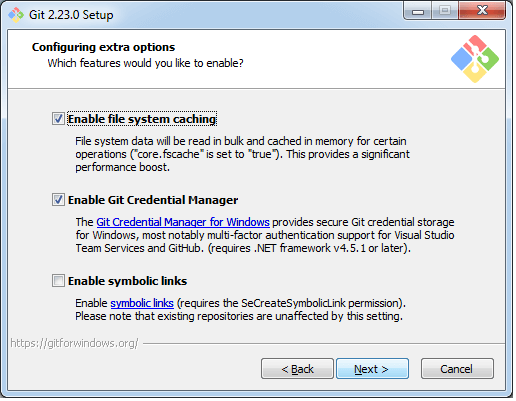
**Step7**

Select preferred terminal emulator clicks on the **next** to continue.



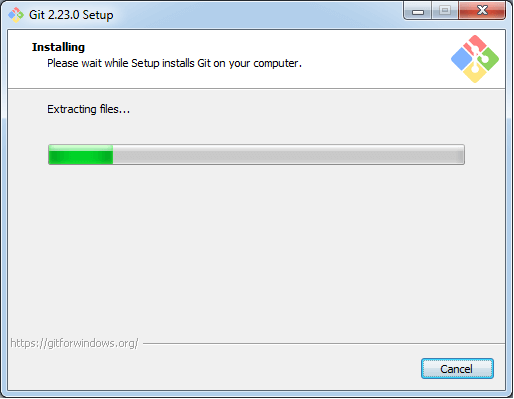
**Step8**

This is the last step that provides some extra features like system caching, credential management and symbolic link. Select the required features and click on the **next** option.



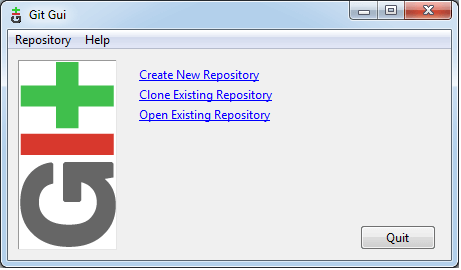
**Step9**

The files are being extracted in this step.



Therefore, The Git installation is completed. Now you can access the **Git Gui** and **Git Bash**.

The **Git Gui** looks like as



It facilitates with three features.

* Create New Repository
* Clone Existing Repository
* Open Existing Repository

The **Git Bash** looks like as



## Git Push Command

The [Git push command](https://www.simplilearn.com/tutorials/git-tutorial/git-push-command) is used to push the local repository content to a remote repository. After a local repository has been modified, a push is executed to share the modifications with remote team members. Pushing is the way commits are transferred from the local repository to the remote repository.

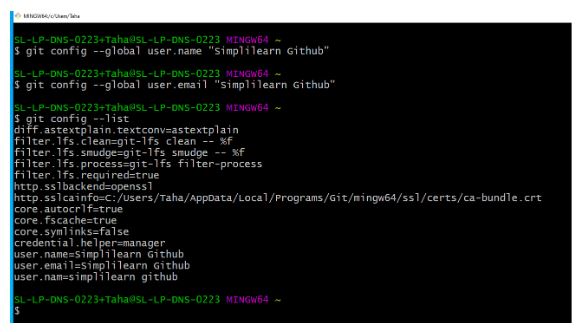


### Demo of Git Push Command

Next in the how Git works tutorial, let’s start with opening Git Bash and configuring it with a user name and email ID. To configure, we use the following commands:

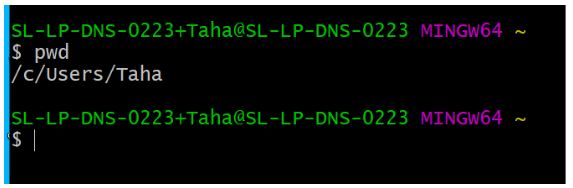
To configure, we use the following commands:

|  |
| --- |
| Git config --global user.name "Simplilearn GitHub"  Git config --global user.email siddam.bharat@simplilearn.net  Git config –-list |



Then, let's check the current working directory:

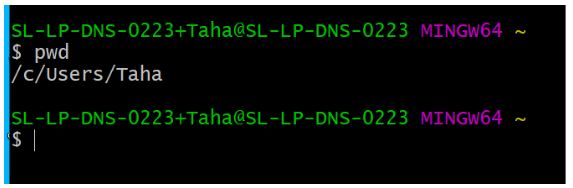
|  |
| --- |
| Pwd |



To create a repository in the working directory, use the following commands:

|  |
| --- |
| mkdir Git\_Demo  cd Git\_Demo  pwd |

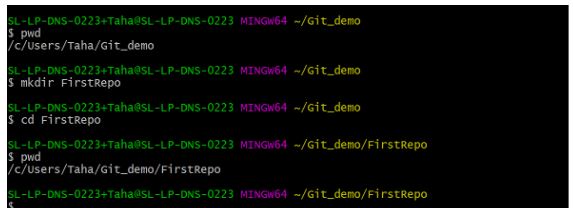
We can go to the directory location and check the Git\_demo folder.



The directory "Git\_demo" will be empty for now.

Let's create a folder for the repository.

|  |
| --- |
| mkdir FirstRepo  cd FirstRepo  pwd |



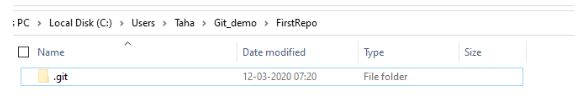
The folder "FirstRepo" is empty. We will now initialize a repository to our folder.

|  |
| --- |
| Git init |



Something called the "master" appears on the screen. Whenever a Git repository is created for the first time, it creates a branch, and the name of the branch is master. Navigate to the folder; you can find a hidden ".git" folder.

If you check the folder, you can see several directories and configurations. Make sure you don't make any changes to any of the directories.

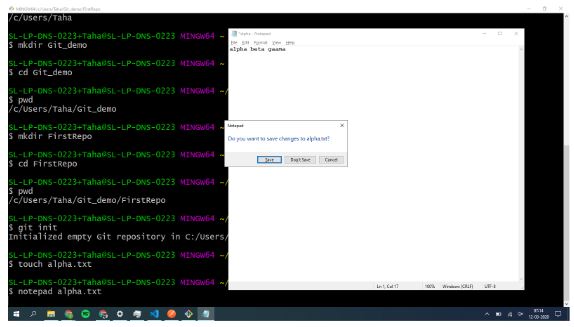


This is created when a repository is initialized.

Moving further, let's make some commits. For that, I will create two notepads and commit them one by one.

For the first notepad, the commands are as follows:

|  |
| --- |
| touch alpha.txt  notepad alpha.txt |

A notepad opens on the screen. Type anything inside it, save it and close it. 

Next, let's check the status of the file that was created.

|  |
| --- |
| git status |

This shows that there isn't a file committed yet, and there are untracked files. The untracked files can be seen in red.



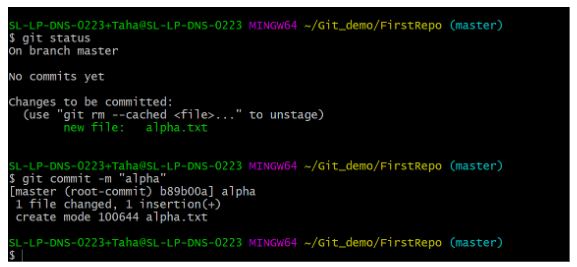
For Git to track that file, add command is used. If you know the exact name of the file, you can specify it and simply type the following command:

|  |
| --- |
| git add . |

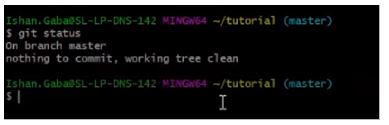


The next step is to commit the file.

|  |
| --- |
| git commit -m "alpha" |



Let's check the status of the file again.



|  |
| --- |
| git status |

You'll notice that there are no more commits to be made, as there was a single notepad and that was committed in the previous step.

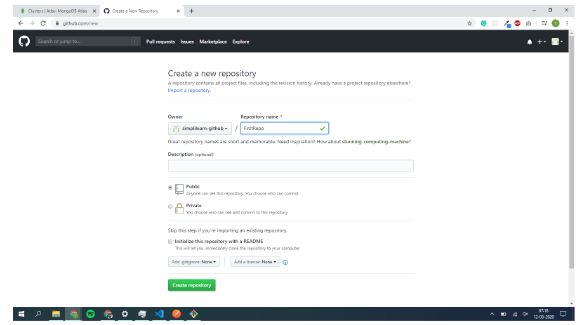
Next, check all the information regarding the commits that were made.

|  |
| --- |
| git log |



This displays the commit ID, author's name, and email ID used. You can also find the date and commit message on the screen.

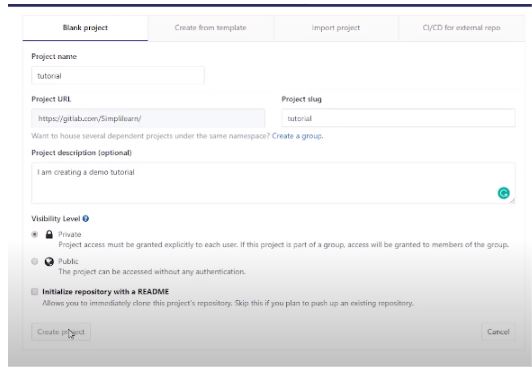
Now, let's push the two notepads on GitHub. Open your GitHub account, and create a new repository. The name of the repository will be "FirstRepo."



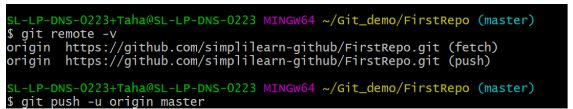
Copy the ------command name: git remote add origin URL(url of the repo in git hub)



Paste the copied URL onto the Git Bash.

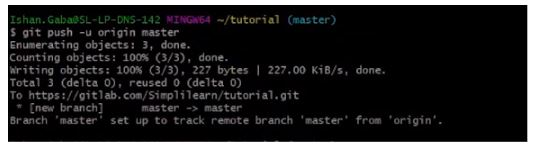


|  |
| --- |
| git remote –v |



Now, let's push the content on to the remote repository.

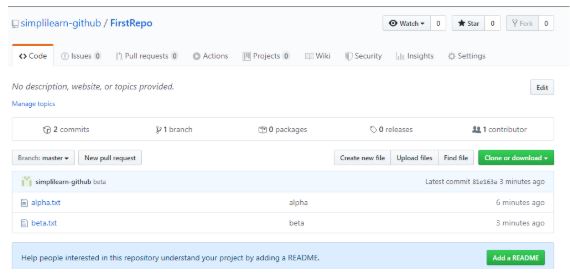
|  |
| --- |
| git push -u origin master |



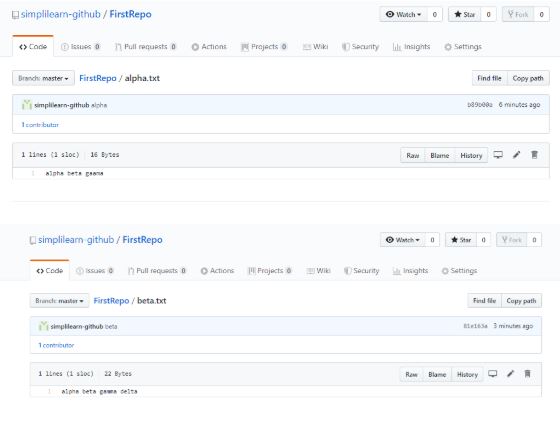
The repository is created on the server, and the content is pushed into that repository. It links the master branch on the local repository to the master branch on the server.

Next, refresh the GitHub page, and you can find all the commits there.

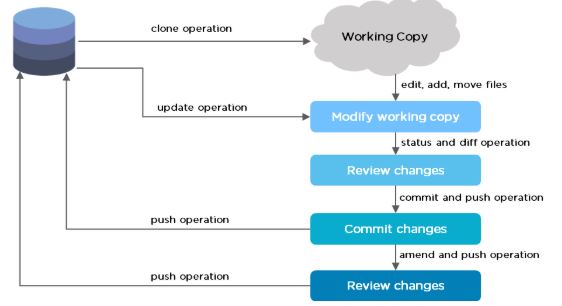
Each commit has a hash ID, which contains the details of each commit.



You can open each notepad and check the content inside.

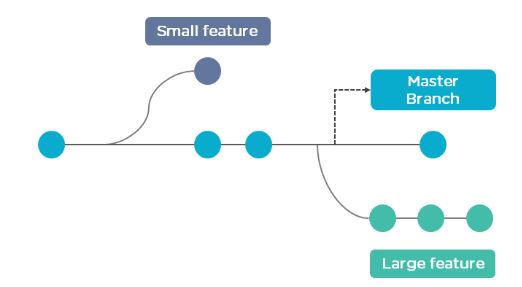


## Git Workflow



## Branch in Git

Branch in Git is used to keep your changes until they are ready. You can do your work on a branch while the main branch (master) remains stable. After you are done with your work, you can merge it with the main office.



The above diagram shows there is a master branch. There are two separate branches called “small feature” and “large feature.” Once you are finished working with the two separate branches, you can merge them and create a master branch.

* Create a branch “first\_branch” and merge it to the main (master) branch.

first-branch.

The above command creates a branch.



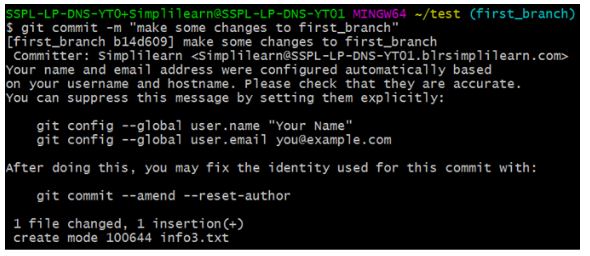
The above command switches to the new branch from the master branch.



The above command creates and adds “info3.txt” to the first\_branch.

info3text

* Create a branch “first\_branch” and merge it with the main (master) branch.



The above command makes a commit to the first\_branch.

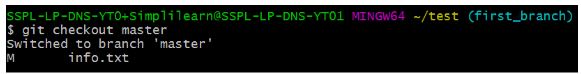
branch

The above command shows that the new branch has access to all the files.

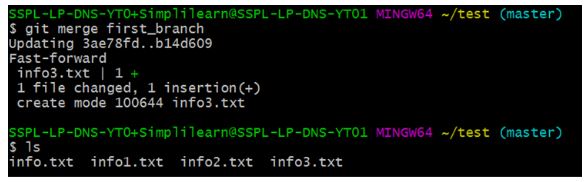
git-br

The above command shows that the master branch does not have an “info3.txt” file.

[https://www.simplilearn.com/ice9/free_resources_article_thumb/master-branch-info.JPG](https://s3.amazonaws.com/static2.simplilearn.com/ice9/free_resources_article_thumb/master-branch-info.JPG)



The above command is used to merge “first\_branch” with the master branch. Now, the master branch has “info3.txt” file.

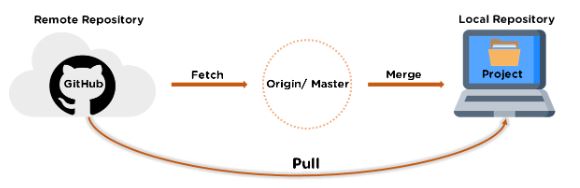


## Git Pull

The git pull command is used to retrieve and download content from a remote repository and update the local repository as soon as it has been downloaded. In Git-based collaboration workflows, it is common to merge remote upstream changes into your local repository.

The Git pull command is used to fetch and merge code changes from the remote repository to the local repository. Git pull is a combination of two commands, Git fetch followed by [Git merge.](https://www.simplilearn.com/tutorials/git-tutorial/merge-conflicts-in-git)

In the first stage, Git fetch is executed that downloads content from the required remote repository. Then, the Git merge command combines multiple sequences of commits into a single branch.



## Git Pull Demo

Let’s begin with making a directory.

|  |
| --- |
| **mkdir Git\_Demo**  **cd Git\_Demo**  **pwd** |



Let’s create a folder for the repository.

|  |
| --- |
| **mkdir Changes**  **cd Changes**  **pwd** |



The folder “Changes” is empty. We will now initialize a repository to our folder.

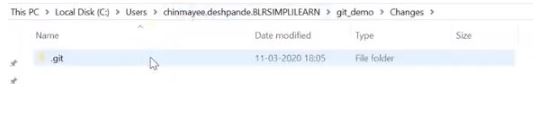
|  |
| --- |
| **Git init** |

now

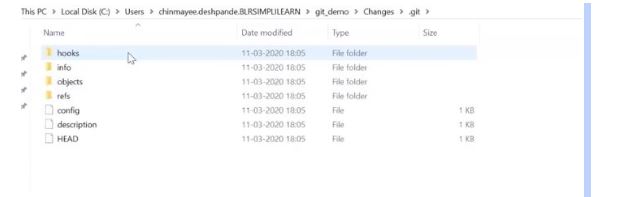
Now something called the “master” appears on the screen. Whenever a Git repository is created for the first time, it creates a branch. The name of the branch is master, and that is why we see master on the screen.

Navigate to the folder to find a hidden “.Git” folder.

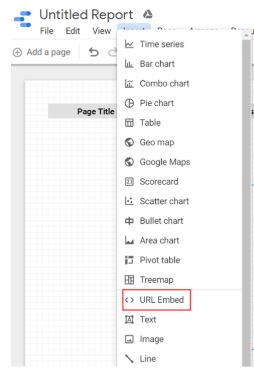
This is created when a repository is initialized.



If you go on to check the folder, you can see a bunch of directories and configurations. Make sure you don't make any changes to any of the directories.



Next, let's pull our files from the remote repository (GitHub).  
For that, go to your GitHub, go to the repository and then to the clone or download option, and copy the URL.



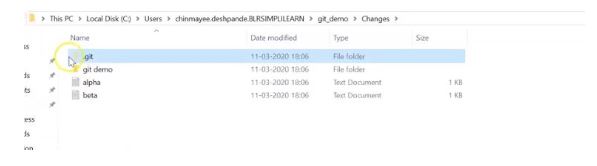
Coming back to GitHub, paste the URL with Git pull command.

|  |
| --- |
| **Git pull \*clone or download URL\*** |



All the contents from the repository have been pulled to our local repository.

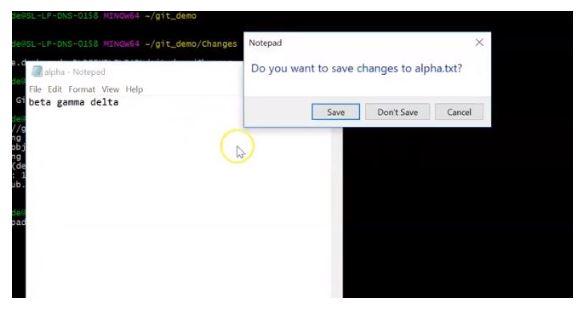
The contents can be found in the desired directory.



Now, let's make changes in the pulled files, and later push them back on the GitHub.

Come back to Git bash, open the notepad, and edit the contents.

|  |
| --- |
| **C:/windows/notepad alpha** |



#### Free Course: Introduction to DevOps Tools

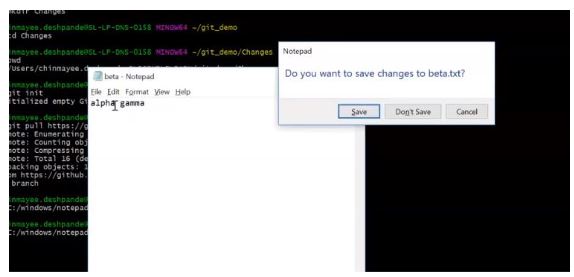
Master the Fundamentals of DevOps Tools[ENROLL NOW](https://www.simplilearn.com/devops-tools-free-course-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

Free Course: Introduction to DevOps Tools

Save it and close it.

Open the next notepad.

|  |
| --- |
| **C:/windows/notepad alpha** |

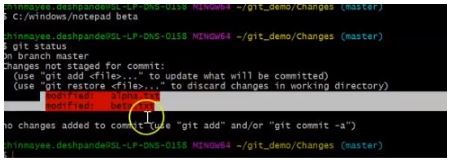


Save it and close it.

Next is to check the status of all the files and directories.

|  |
| --- |
| **Git status** |

It shows that no file is yet committed, and there are untracked files. The untracked files can be seen in red color.

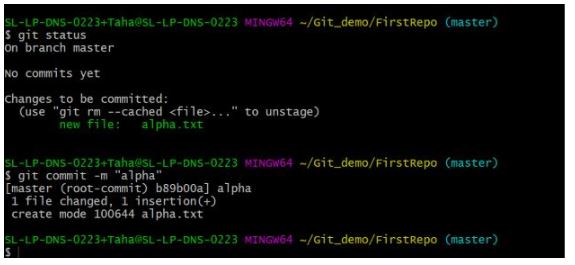


For Git to track that file, the add command is given. If you know the exact name of the file, you can specify that simply type the following command:

|  |
| --- |
| **Git add .** |

After add, the next step is to commit these files.

|  |
| --- |
| **Git commit -m “changes made”** |

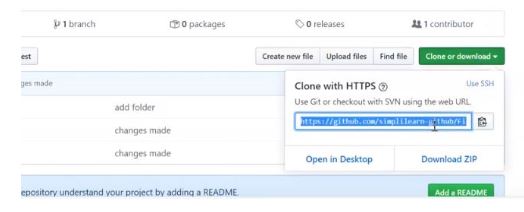


Let’s check the status of the file again.

|  |
| --- |
| **Git status** |



It can be seen that there are no more commits to be made. After the commits are made, let’s push the edited files back to the remote repository.  
Again go back to your GitHub and copy the URL.



Come back to Git bash, type the Git remote command and paste the URL.

|  |
| --- |
| **Git remote add origin \*URL\*** |

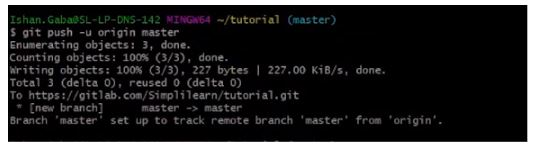


|  |
| --- |
| **Git remote –v** |

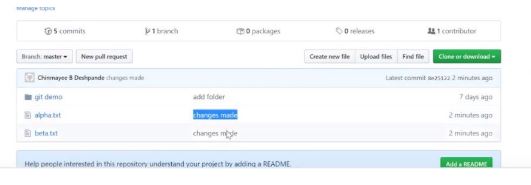


Now let's push the content back on to the remote repository.

|  |
| --- |
| **Git push -u origin master** |



The edited content has been sent back to the remote repository. Let’s go and check the content there.



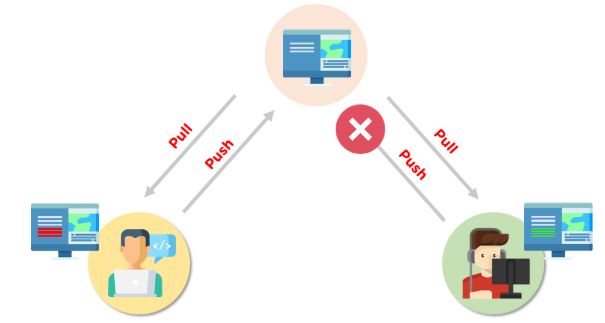
The contents of the notepad can be seen, and it is evident that the content has been changed. The commit “changes made” may also be seen.



## What is a Git Merge Conflict?

A 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.

The following is an example of how a Git merge conflict works:



Let’s assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.

To prevent such conflicts, developers work in separate isolated branches. The Git merge command combines separate branches and resolves any conflicting edits.

FREE GIT Training

## How to Resolve Merge Conflicts in Git?

There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

1. The easiest way to resolve a conflicted file is to open it and make any necessary changes
2. After editing the file, we can use the git add a command to stage the new merged content
3. The final step is to create a new commit with the help of the git commit command
4. Git will create a new merge commit to finalize the merge

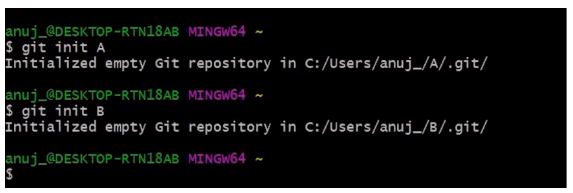
Let us now look into the Git commands that may play a significant role in resolving conflicts.

FREE DevOps Certification Training

## Demo: Resolving Git Merge Conflicts

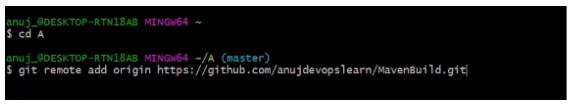
First, initialize two repositories:

|  |
| --- |
| git init A  git init B |



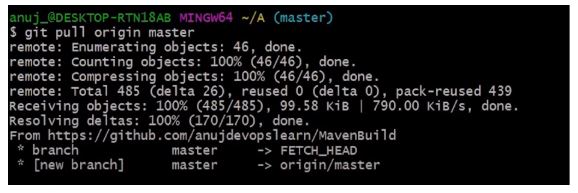
Add the remote address in the A repository:

|  |
| --- |
| git remote add origin \*address\* |



The next step is to pull all the changes in the central repository to the local repository.

|  |
| --- |
| git pull origin master |



Follow the same process to add the origin in the B repository.

|  |
| --- |
| git remote add origin \*address\* |

the-pull

The pull command is executed again to retrieve all the content from the remote repository and move it to the local repository.

|  |
| --- |
| git pull origin master |



Both of these repositories represent two different repositories of two different developers.

#### Post Graduate Program in DevOps

Designed in collaboration with Caltech CTME[EXPLORE COURSE](https://www.simplilearn.com/pgp-devops-certification-training-course?source=GhPreviewCTABanner)

Post Graduate Program in DevOps

Let's get back to the A repository.

|  |
| --- |
| cd ../A |



In the A repository, a readme file is opened in order to make various changes.

|  |
| --- |
| vi README.md |



Make the necessary changes in the file, and then save it.

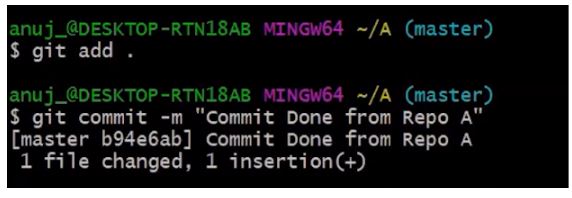
The git status command is then executed in order to see the reflected changes.

|  |
| --- |
| git status |



The next step is to add these changes to the staging area and commit them.

|  |
| --- |
| git add.  git commit -m \*commit message\* |



#### Free Course: Introduction to DevOps Tools

Master the Fundamentals of DevOps Tools[ENROLL NOW](https://www.simplilearn.com/devops-tools-free-course-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)

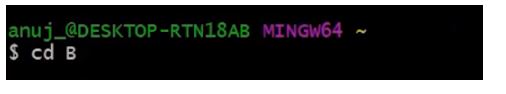
Free Course: Introduction to DevOps Tools

After the commit is finished, the changed file is pushed to the remote repository.

|  |
| --- |
| git push origin master |

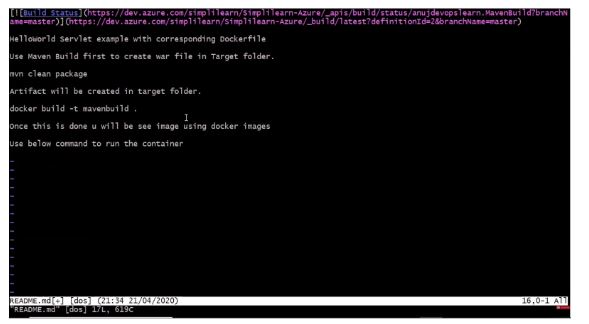
Now, return to the B repository.

|  |
| --- |
| cd B |



Open a readme file

|  |
| --- |
| vi README.md |



Make changes to the file, save it, and close it. After that, add the changed file and commit it.

|  |
| --- |
| git add.  git commit -m \*commit message\* |



The next step is to push the file to the remote repository.

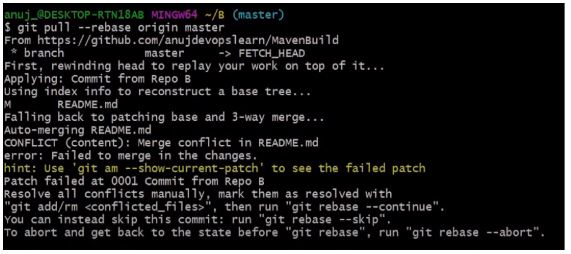
|  |
| --- |
| git push |



An error is shown, meaning that the updates are rejected.

Next, we need to execute:

|  |
| --- |
| git -- rebase origin master |

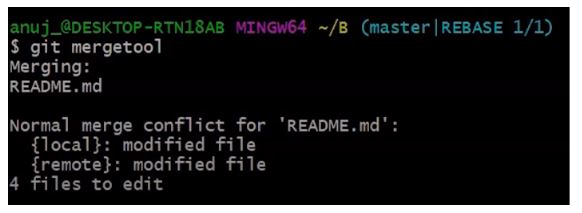


Currently, there are visible conflicts that need to be resolved manually.

If you want to skip this commit, you can type git rebase --skip, or if you want to abort this rebase, you can type git rebase --abort.

After managing this conflict manually, we will open the merge tool.

|  |
| --- |
| git mergetool |



After we input this command, all of the files will be processed.



These are all the processes and the modifications done in the file.

You can see three different files there, and you can see everything that was added or removed.

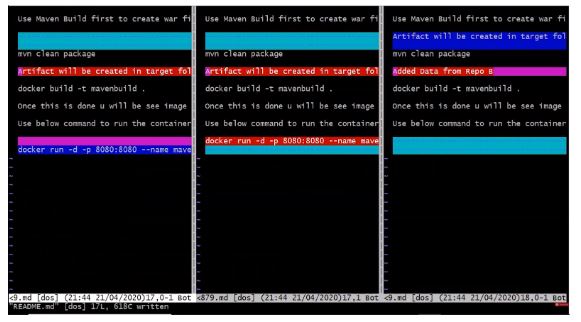
After scrolling, you can verify where exactly the conflict happened.



You can then decide if you want to continue with this particular file or not. I will proceed with removing that line.

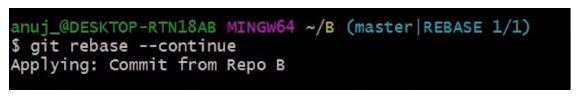


Manual modifications have allowed us to resolve file conflicts. Save the file and close the final file.



Next, we will run:

|  |
| --- |
| git rebase –continue |



Now, when the conflict is resolved, we must be able to push the file to the remote repository.

