## 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.

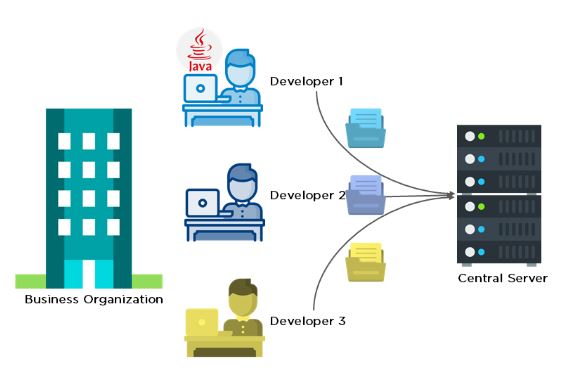
# What is GitHub?

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



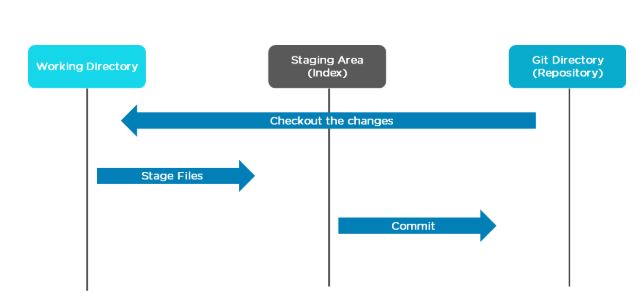
Now let’s look at the scenario after Git:

* 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?

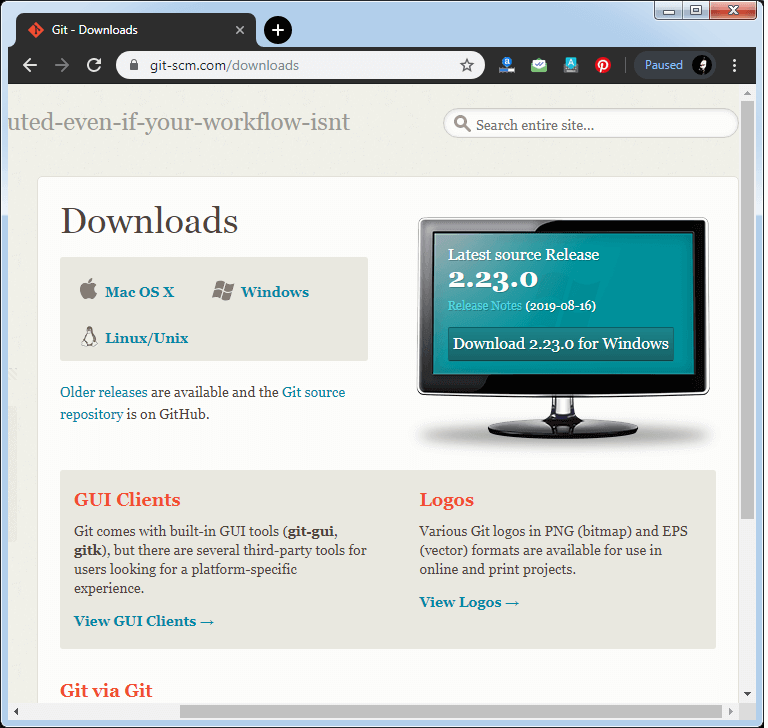
**Step1**

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



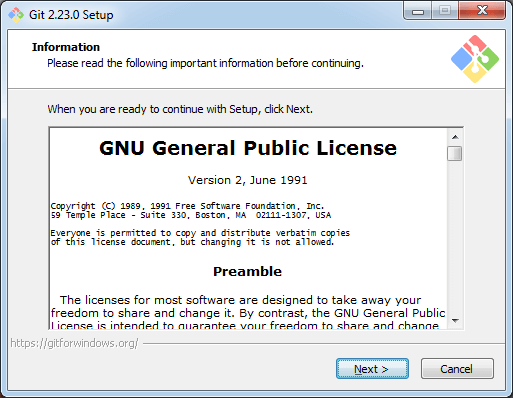
Click on the package given on the page as **download 2.23.0 for windows**. The download will start after selecting the package.

Now, the Git installer package has been downloaded.

## Install Git

**Step2**

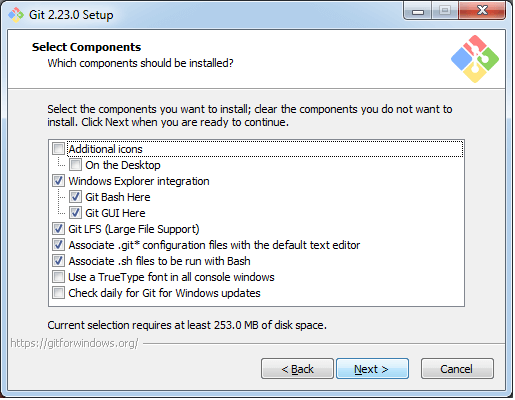
Click on the downloaded installer file and select **yes** to continue. After the selecting **yes** the installation begins, and the screen will look like as



Click on **next** to continue.

**Step3**

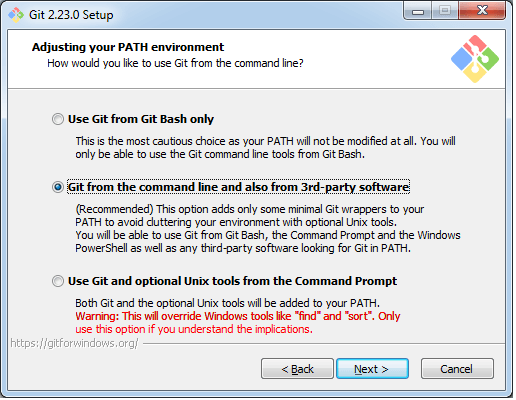
Default components are automatically selected in this step. You can also choose your required part.



Click next to continue.

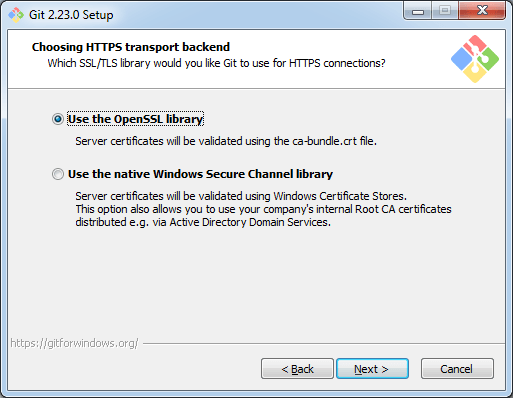
**Step4**

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



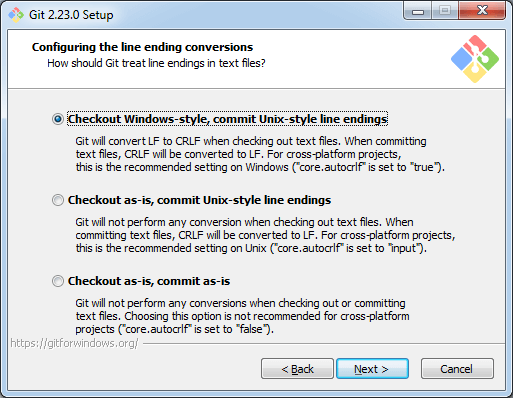
**Step5**

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



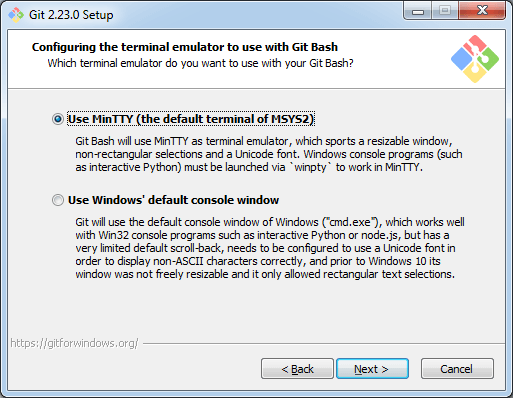
**Step6**

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



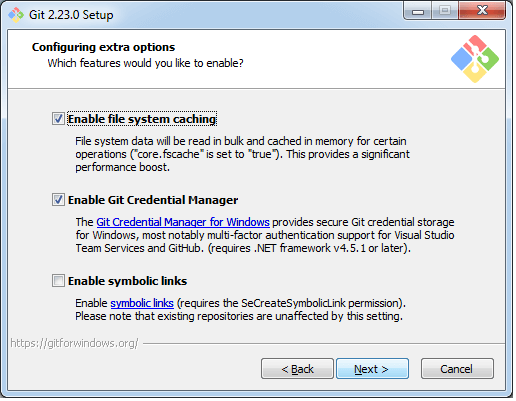
**Step7**

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



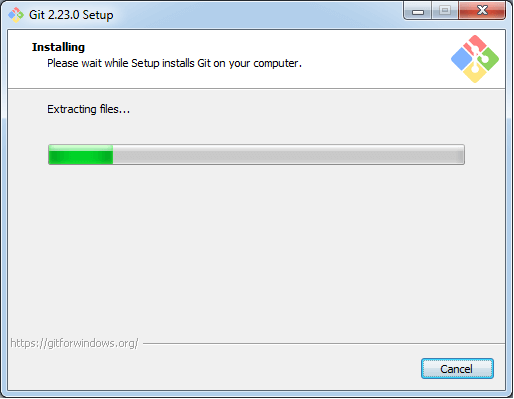
**Step8**

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.



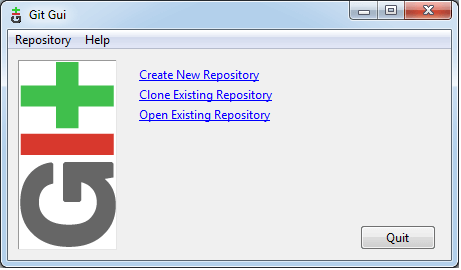
**Step9**

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



Different Commands in Git

So far, in the how Git works tutorial you read all about [Git and GitHub](https://www.simplilearn.com/tutorials/git-tutorial/git-vs-github). Next in the how Git works tutorial comes [Git commands](https://www.simplilearn.com/tutorials/git-tutorial/git-commands).

* Git config
* Git init
* Git add
* Git diff
* Git commit
* Git reset
* Git status
* [Git merge](https://www.simplilearn.com/tutorials/git-tutorial/merge-conflicts-in-git)
* Git push
* [Git pull](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request)

Git Commands:

## Git Commands: Working With Local Repositories

### git init

* The command git init is used to create an empty Git repository.
* After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.

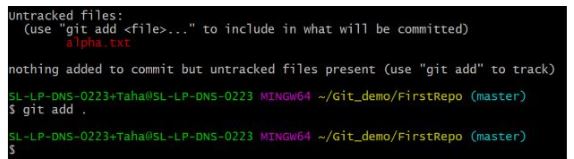
|  |
| --- |
| git init |



### git add

* Add command is used after checking the status of the files, to add those files to the staging area.
* Before running the commit command, "git add" is used to add any new or modified files.

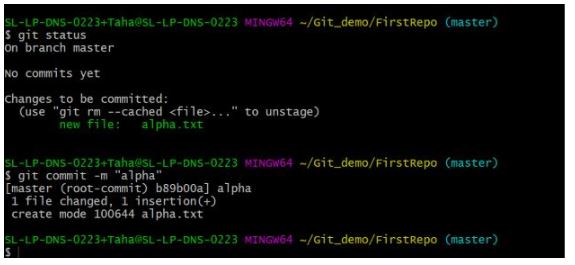
|  |
| --- |
| git add . |



### git commit

* The commit command makes sure that the changes are saved to the local repository.
* The command "git commit –m <message>" allows you to describe everyone and help them understand what has happened.

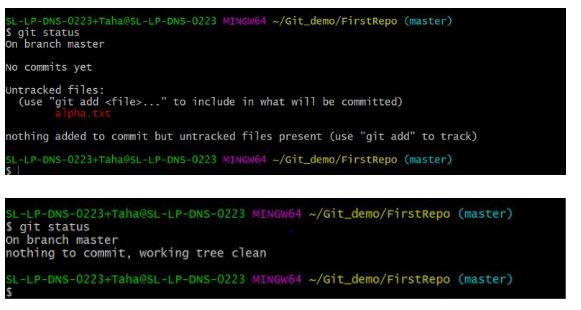
|  |
| --- |
| git commit -m “commit message” |



### git status

* The git status command tells the current state of the repository.
* The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the git status. Also, if there are no changes, it will show the message no changes to commit, working directory clean.

|  |
| --- |
| git status |



#### Free Course: Introduction to DevOps Tools

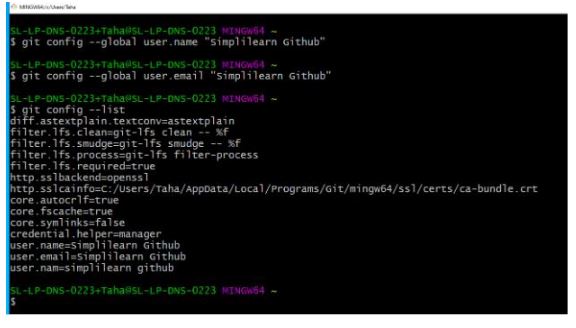
Master the Fundamentals of DevOps Tools[ENROLL NOW](https://www.simplilearn.com/devops-tools-free-course-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

Free Course: Introduction to DevOps Tools

### git config

* The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
* When git config is used with --global flag, it writes the settings to all repositories on the computer.

|  |
| --- |
| git config --global user.name “any user name”  git config --global user.email <email id> |



### git branch

* The git branch command is used to determine what branch the local repository is on.
* The command enables adding and deleting a branch.

|  |
| --- |
| # Create a new branch   git branch <branch\_name> |
| # List all remote or local branches   git branch -a |
| # Delete a branch   git branch -d <branch\_name> |

### git checkout

* The git checkout command is used to switch branches, whenever the work is to be started on a different branch.
* The command works on three separate entities: files, commits, and branches.

|  |
| --- |
| # Checkout an existing branch   git checkout <branch\_name> |
| # Checkout and create a new branch with that name   git checkout -b <new\_branch> |

### git merge

* The [git merge](https://www.simplilearn.com/tutorials/git-tutorial/merge-conflicts-in-git) command is used to integrate the branches together. The command combines the changes from one branch to another branch.
* It is used to merge the changes in the staging branch to the stable branch.

|  |
| --- |
| git merge <branch\_name> |

#### DevOps Engineer Master's Program

Bridge between software developers and operations[EXPLORE COURSE](https://www.simplilearn.com/devops-engineer-masters-program-certification-training?source=GhPreviewCTABanner)

DevOps Engineer Master's Program

However, these are popular and basic git commands used by developers.

## Git Commands: Working With Remote Repositories

### git remote

* The git remote command is used to create, view, and delete connections to other repositories.
* The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

|  |
| --- |
| git remote add origin <address> |



### git clone

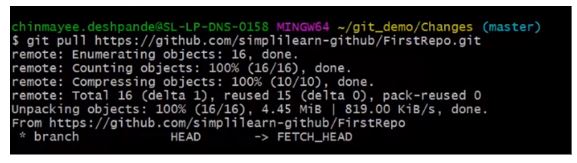
* The git clone command is used to create a local working copy of an existing remote repository.
* The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

|  |
| --- |
| git clone <remote\_URL> |

### git pull

* The [git pull command](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request) is used to fetch and merge changes from the remote repository to the local repository.
* The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

|  |
| --- |
| git pull <branch\_name> <remote URL> |



### git push

* The command [git push](https://www.simplilearn.com/tutorials/git-tutorial/git-push-command) is used to transfer the commits or pushing the content from the local repository to the remote repository.
* The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

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



## 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.



Now that we have learned about the Git push command in this Git Bash tutorial let us take a look at a Git push command demo.

#### FREE DevOps Certification Training

Master the fundamnetals of DevOps[ENROL NOW](https://www.simplilearn.com/learn-devops-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

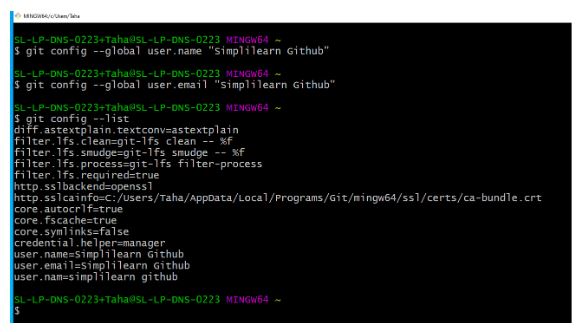
FREE DevOps Certification Training

### 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:

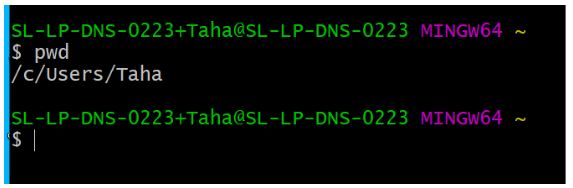
To configure, we use the following commands:

|  |
| --- |
| 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:

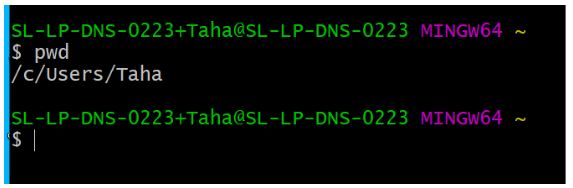
|  |
| --- |
| pwd |



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

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

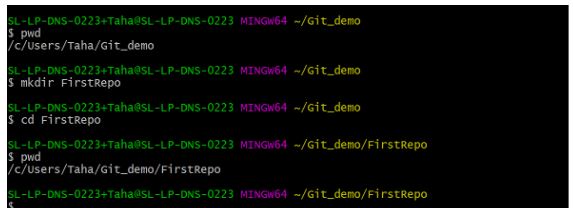
We can go to the directory location and check the Git\_demo folder.



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

Let's create a folder for the repository.

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



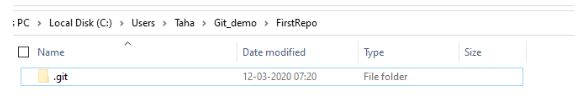
The folder "FirstRepo" is empty. We will now initialize a repository to our folder.

|  |
| --- |
| 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.

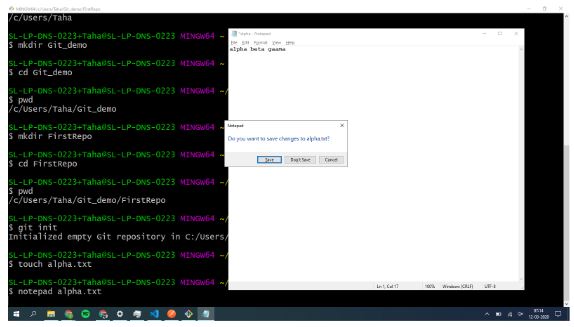


This is created when a repository is initialized.

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.



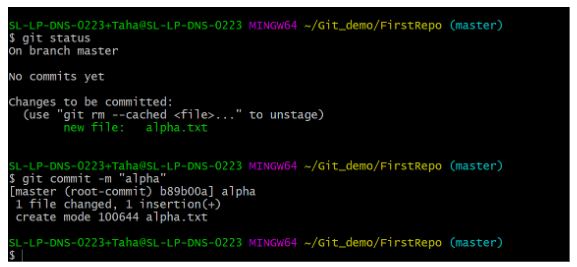
For Git to track that file, add command is used. If you know the exact name of the file, you can specify it and simply type the following command:

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

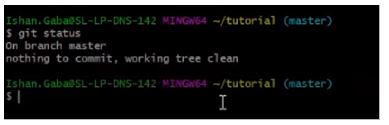


The next step is to commit the file.

|  |
| --- |
| 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 |

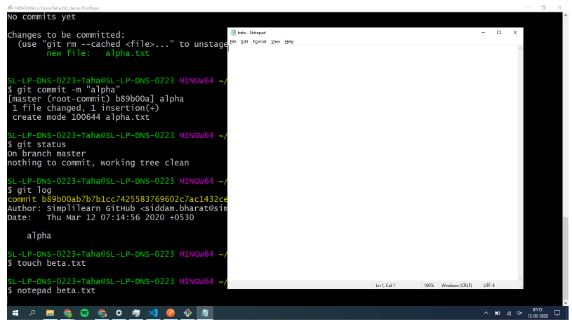


This displays the commit ID, author's name, and email ID used. You can also find the date and commit message on the screen.

Let's make one more commit.

Repeat the same process again. I will make a notepad, add something to it, and close it.

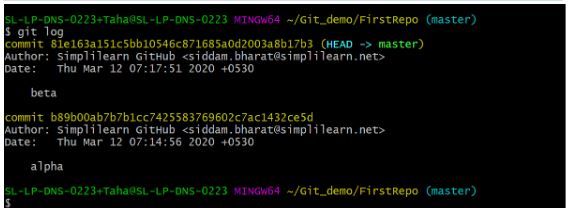
|  |
| --- |
| touch beta.txt  notepad beta.txt |



|  |
| --- |
| git add .  git commit -m "beta" |

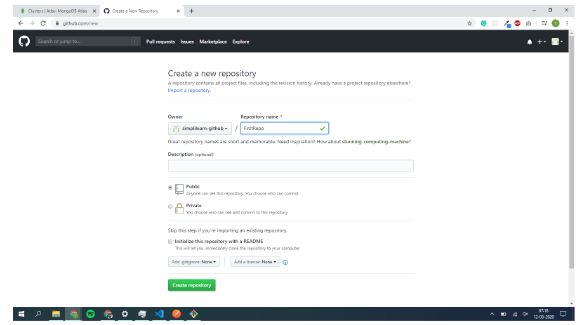


|  |
| --- |
| git log |



We can see the commit number and order of the commits.

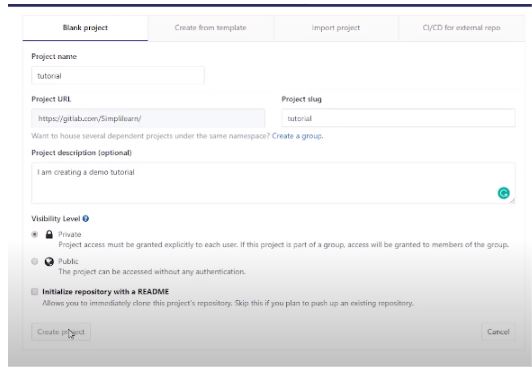
Now, let's push the two notepads on GitHub. Open your GitHub account, and create a new repository. The name of the repository will be "FirstRepo."



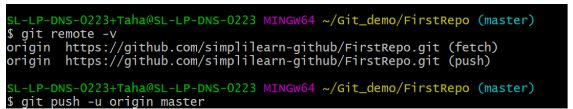
Copy the "git remote add origin" URL.



Paste the copied URL onto the Git Bash.

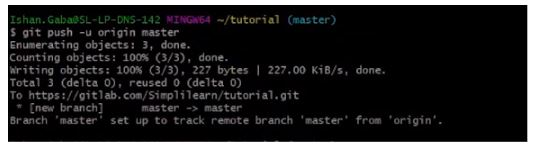


|  |
| --- |
| git remote -v |



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

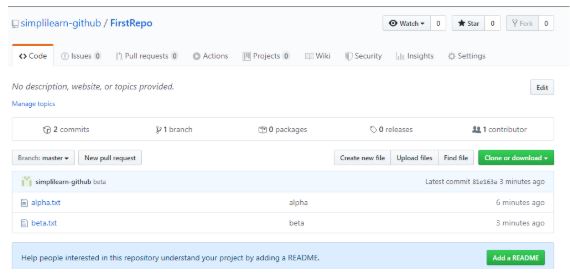
|  |
| --- |
| git push -u origin master |



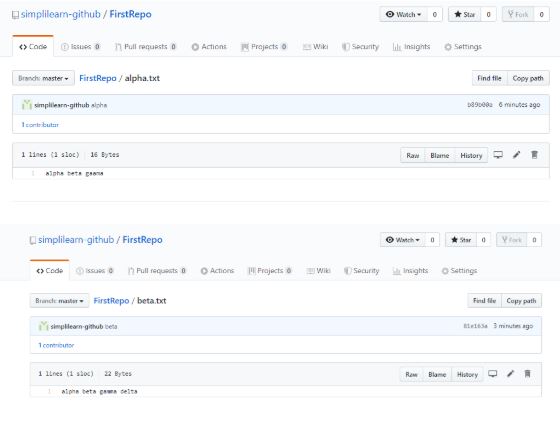
The repository is created on the server, and the content is pushed into that repository. It links the master branch on the local repository to the master branch on the server.

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.

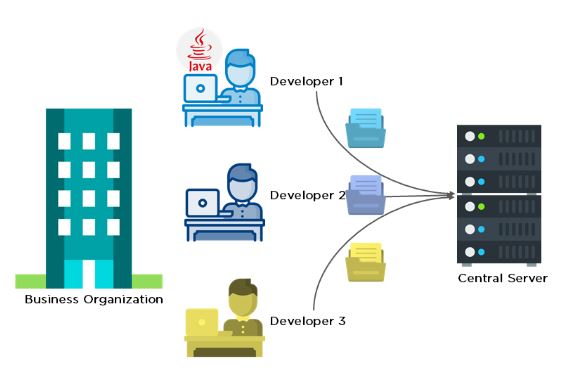


You can open each notepad and check the content inside.



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



#### 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

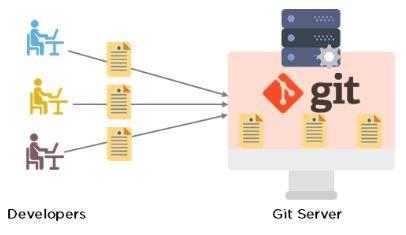
## What is Git?

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development.

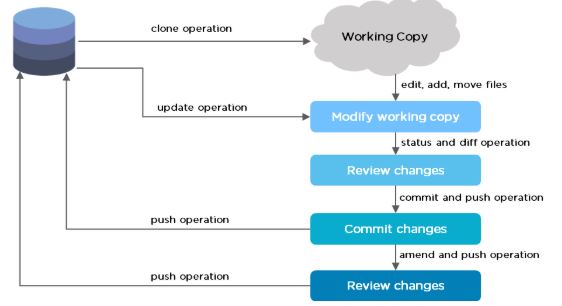
* Git is used to tracking changes in the source code
* The distributed version control tool is used for source code management
* It allows multiple developers to work together
* It supports non-linear development through its thousands of parallel branches

## Features of Git

* Tracks history
* Free and open source
* Supports non-linear development
* Creates backups
* Scalable
* Supports collaboration
* Branching is easier
* Distributed development

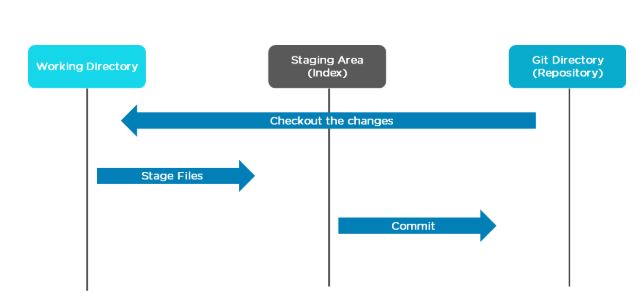


## Git Workflow



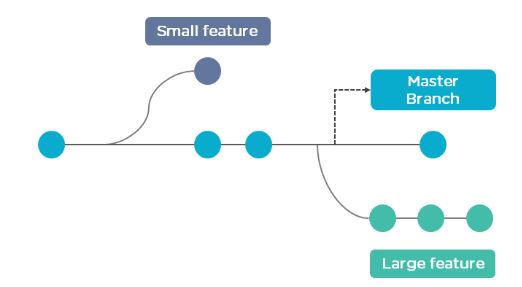
The Git workflow is divided into three states:

* 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.



## 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.

#### FREE GIT Training

Learn the basics of GIT[ENROLL NOW](https://www.simplilearn.com/learn-git-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

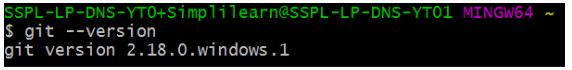
FREE GIT Training

## Commands in Git

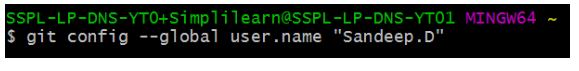
* Create Repositories  
  git init
* Make Changes  
  add  
  commit  
  status
* Parallel Development  
  branch  
  merge  
  rebase
* Sync Repositories  
  push  
  pull  
  add origin

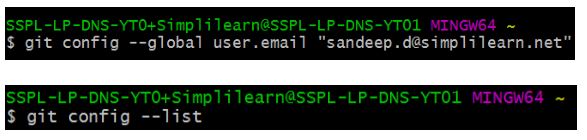
## Command

* Check the version of Git.

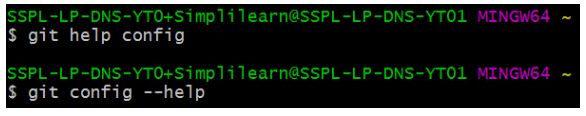


* Set up global config variables - If you are working with other developers, you need to know who is checking the code in and out, and to make the changes.

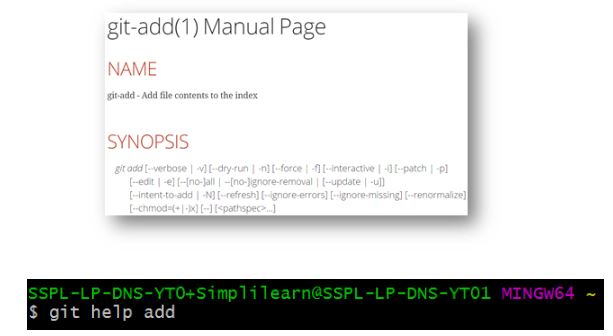




* If in case you need help, use the following [commands](https://www.simplilearn.com/tutorials/git-tutorial/git-commands):



This will lead you to the Git help page on the browser, which will display the following:

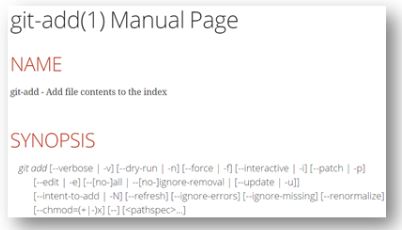


#### DevOps Certification Training Course

Gain expertise in the top DevOps tools[EXPLORE COURSE](https://www.simplilearn.com/pgp-devops-certification-training-course?source=GhPreviewCTABanner)

DevOps Certification Training Course

This will lead you to the Git help page on the browser, which will display the following:



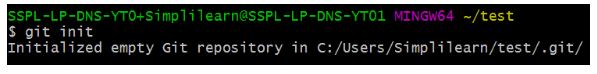
* Create a “test” repository in the local system.



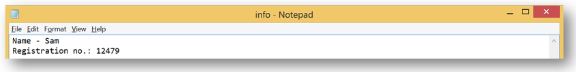
* Move to the test repository.



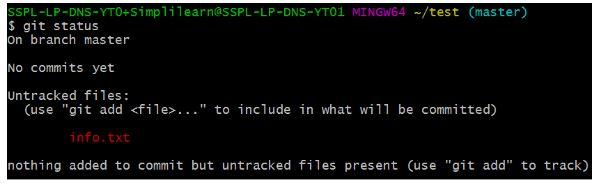
* Create a new git instance for a project.



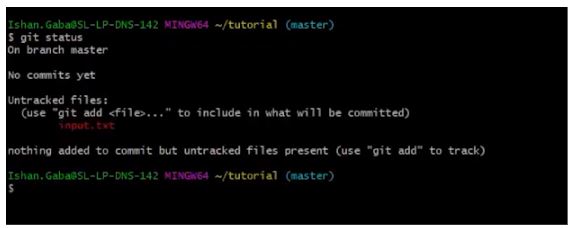
* Create a text file called info.txt in the test folder; write something and save it.



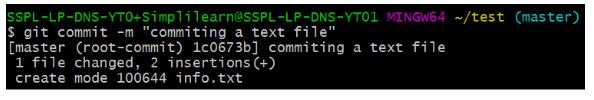
* Check the status of the repository.



* Add the file you created to make a commit.



* Commit those changes to the repository’s history with a short message.

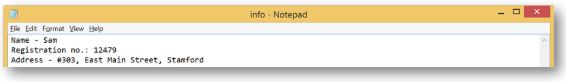


#### Free Course: Introduction to DevOps Tools

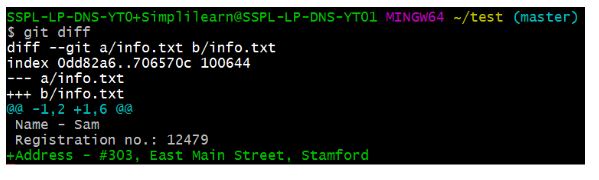
Master the Fundamentals of DevOps Tools[ENROLL NOW](https://www.simplilearn.com/devops-tools-free-course-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

Free Course: Introduction to DevOps Tools

* Make any necessary changes to the file and save.



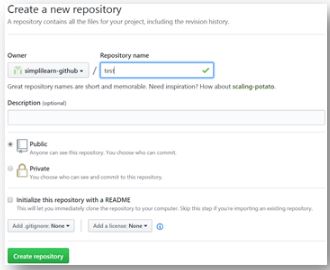
* Now that you’ve made changes to the file, you can compare the differences since your last commit.



* Add [GitHub](https://www.simplilearn.com/tutorials/git-tutorial/what-is-github" \o "GitHub" \t "_blank) username to Git Configuration.

username

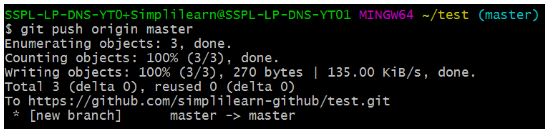
* Create a remote repository.



* Connect the local repository to your remote repository.

local-repo

* Push the file to the remote repository.

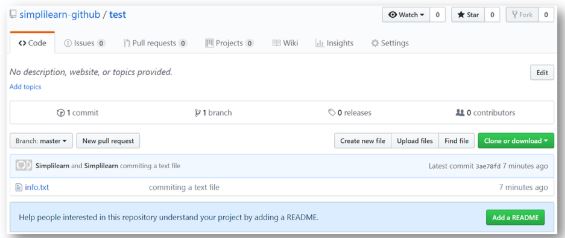


#### DevOps Engineer Master's Program

Bridge between software developers and operations[EXPLORE COURSE](https://www.simplilearn.com/devops-engineer-masters-program-certification-training?source=GhPreviewCTABanner)

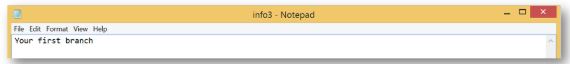
DevOps Engineer Master's Program

* Refresh your repository page on GitHub. You will get your local file on your remote [GitHub repository](https://www.simplilearn.com/tutorials/git-tutorial/git-vs-github" \o "GitHub repository" \t "_blank).



* Create three more text files in the local repository - “info1.txt”, “info2.txt”, “info3.txt”.





* 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.

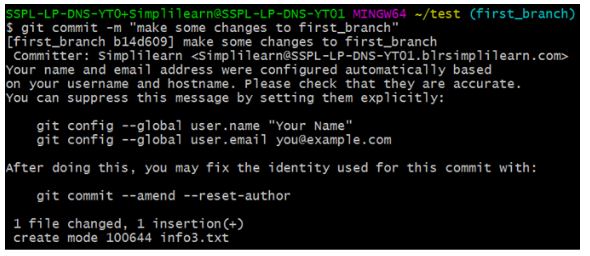
info3text

#### FREE DevOps Certification Training

Master in-demand DevOps tools and skills[ENROLL FOR FREE](https://www.simplilearn.com/learn-devops-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

FREE DevOps Certification Training

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



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

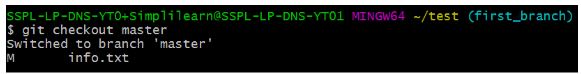
branch

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

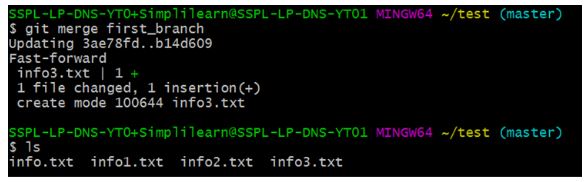
git-br

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.

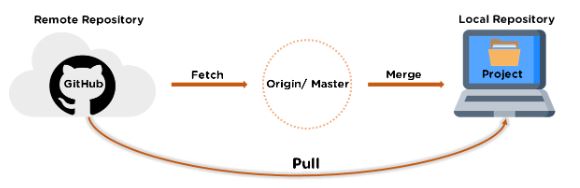


## Git Pull

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** |



The directory. as of now, is empty.



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.

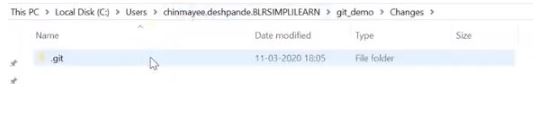
|  |
| --- |
| **Git init** |

now

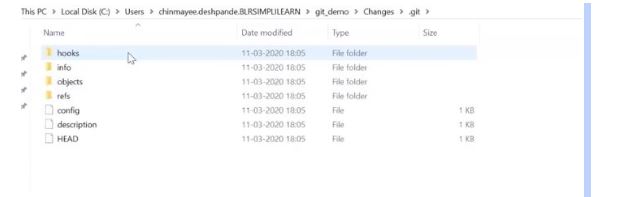
Now something called the “master” appears on the screen. Whenever a Git repository is created for the first time, it creates a branch. The name of the branch is master, and that is why we see master on the screen.

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

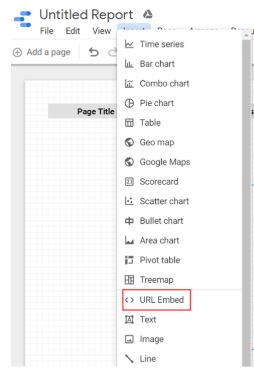
This is created when a repository is initialized.



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.



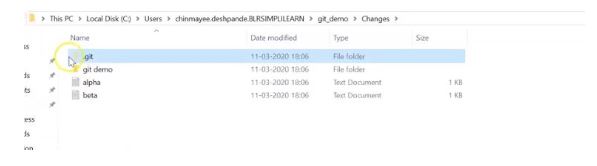
Coming back to GitHub, paste the URL with Git pull command.

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



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

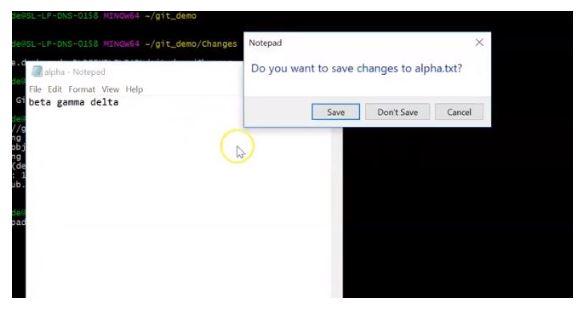
The contents can be found in the desired directory.



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

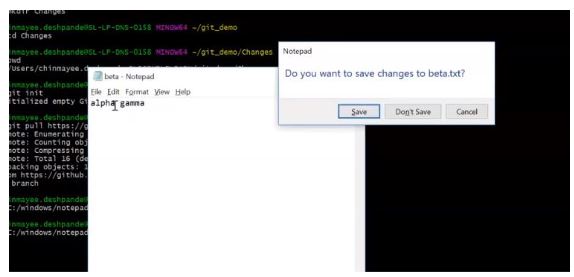
Master the Fundamentals of DevOps Tools[ENROLL NOW](https://www.simplilearn.com/devops-tools-free-course-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

Free Course: Introduction to DevOps Tools

Save it and close it.

Open the next notepad.

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

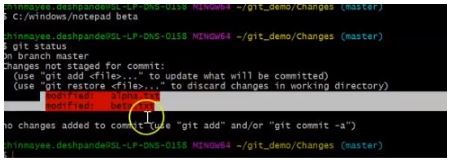


Save it and close it.

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.

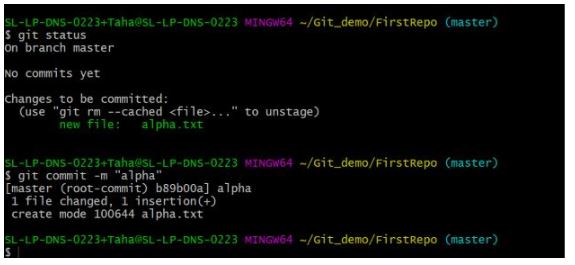


For Git to track that file, the add command is given. If you know the exact name of the file, you can specify that simply type the following command:

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

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

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

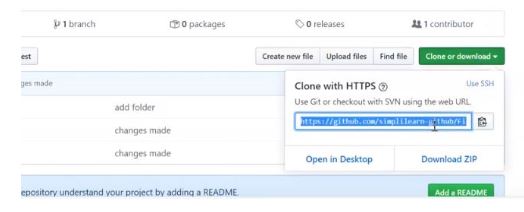


Let’s check the status of the file again.

|  |
| --- |
| **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\*** |

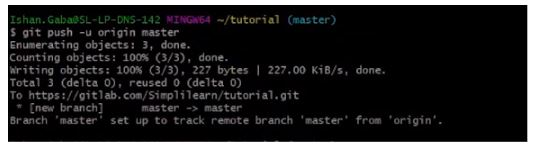


|  |
| --- |
| **Git remote -v** |

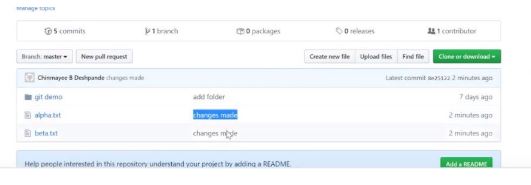


Now let's push the content back on to the remote repository.

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



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



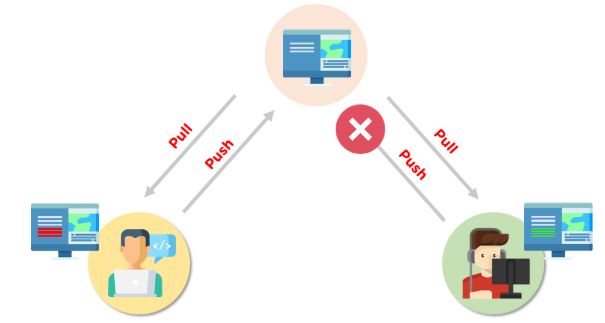
The contents of the notepad can be seen, and it is evident that the content has been changed. The commit “changes made” may also be seen.



## 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.

Now that we have gone through the basics of the Git merge conflict, let’s look at the various types of conflicts next.

#### FREE GIT Training

Learn the basics of GIT[ENROLL NOW](https://www.simplilearn.com/learn-git-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)

FREE GIT Training

## Types of Git Merge Conflicts

There are two points when a merge can enter a conflicted state:

### 1. Starting the Merge Process

If there are changes in the working directory’s stage area for the current project, merging won’t start.

In this case, conflicts happen due to pending changes that need to be stabilized using different [Git commands](https://www.simplilearn.com/tutorials/git-tutorial/git-commands).

### 2. During the Merge Process

The failure during the merge process indicates that there is a conflict between the local branch and the branch being merged.

In this case, Git resolves as much as possible, but there are things that have to be resolved manually in the conflicted files.

We will now go over resolving merge conflicts in Git.

## 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.

## Git Commands to Resolve Conflicts

### 1. git log --merge

The git log --merge command helps to produce the list of commits that are causing the conflict

### 2. git diff

The git diff command helps to identify the differences between the states repositories or files

### 3. git checkout

The git checkout command is used to undo the changes made to the file, or for changing branches

### 4. git reset --mixed

The git reset --mixed command is used to undo changes to the working directory and staging area

### 5. git merge --abort

The git merge --abort command helps in exiting the merge process and returning back to the state before the merging began

### 6. git reset

The git reset command is used at the time of merge conflict to reset the conflicted files to their original state

#### FREE DevOps Certification Training

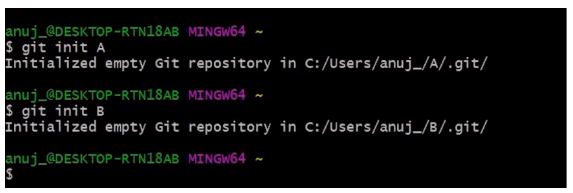
Master the fundamentals of DevOps[START LEARNING](https://www.simplilearn.com/learn-devops-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)

FREE DevOps Certification Training

## Demo: Resolving Git Merge Conflicts

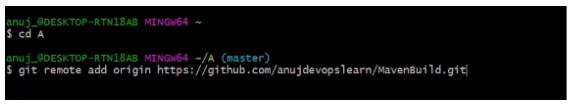
First, initialize two repositories:

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



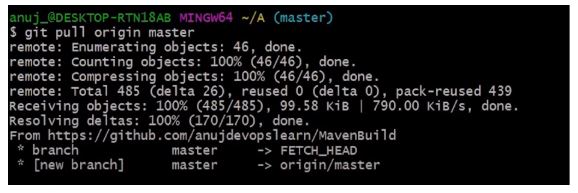
Add the remote address in the A repository:

|  |
| --- |
| 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.

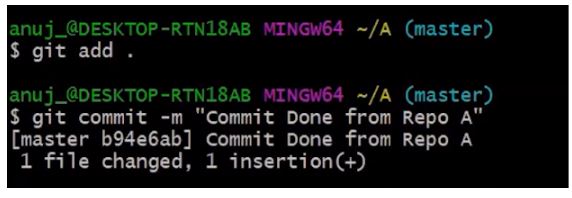
The git status command is then executed in order to see the reflected changes.

|  |
| --- |
| git status |



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

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



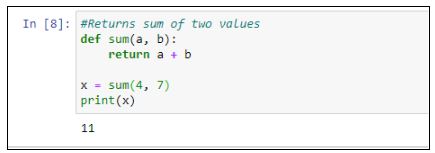
#### Free Course: Introduction to DevOps Tools

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

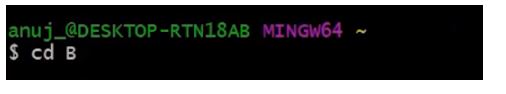
After the commit is finished, the changed file is pushed to the remote repository.

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



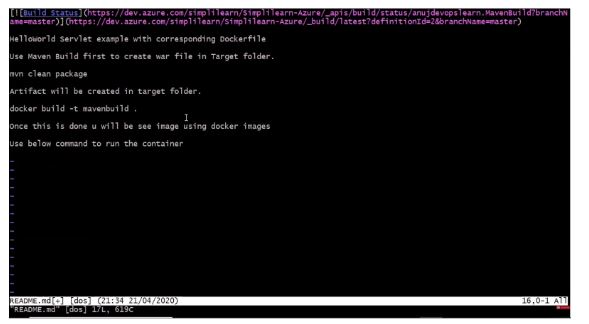
Now, return to the B repository.

|  |
| --- |
| 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.

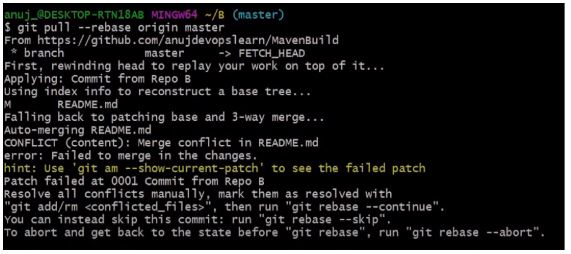
|  |
| --- |
| git push |



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

Next, we need to execute:

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

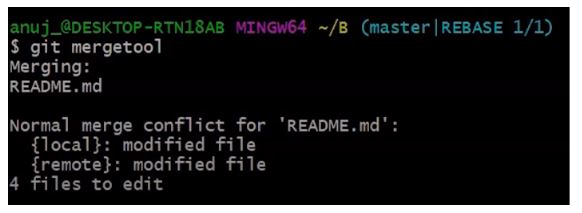


Currently, there are visible conflicts that need to be resolved manually.

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.

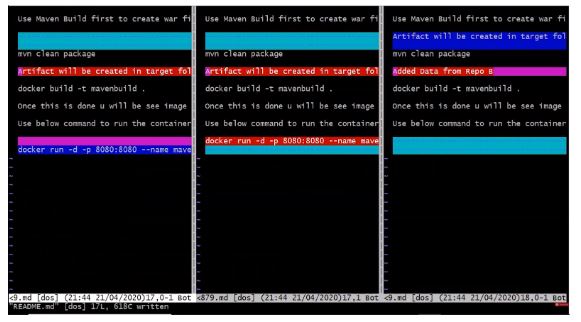
After scrolling, you can verify where exactly the conflict happened.



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

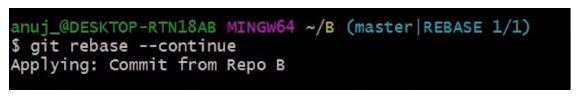


Manual modifications have allowed us to resolve file conflicts. Save the file and close the final file.

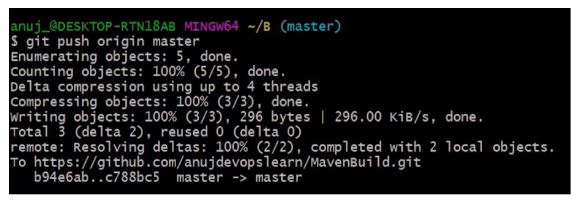


Next, we will run:

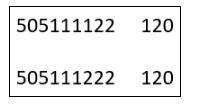
|  |
| --- |
| git rebase --continue |



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



You can now check the commits in your remote repository.



# What is GitHub And How To Use It?

## A GitHub Use Case

Let’s consider the case of Decathlon, the world’s largest sporting goods retail brand. The company has over 1600 stores in 57 countries, with more than 87,000 employees.

Every company, no matter how large or small, inevitably experiences challenges and obstacles. We could best summarize Decathlon’s problems as:

1. How would the company maintain workflow visibility and avoid redundancies in such a large workforce?
2. How would the company hire developers for so many diverse locations?

[GitHub](https://www.simplilearn.com/tutorials/git-tutorial/git-vs-github) to the rescue! GitHub is not only an affordable resource but also features a great open-source community. Since it is a cloud-based tool, the code is conveniently visible across the entire client organization, facilitating every participant's contributions.

GitHub allows collaboration with developers from all over the world. Open-source solutions like GitHub enable potential developers to contribute and share their knowledge to benefit the global community.

The version control system, or VCS, is the element in [Git](https://www.simplilearn.com/tutorials/git-tutorial/what-is-git) that is best suited for tackling Decathlon’s two problems. So, let’s expand our knowledge of GitHub by taking a closer look at the Git [version control system](https://www.simplilearn.com/tutorials/devops-tutorial/version-control) and see why it’s such a game-changer.

## What is a Version Control System?

The Git version control system, as the name suggests, is a system that records all the modifications made to a file or set of data so that a specific version may be called up later if needed. The system makes sure that all the team members are working on the file’s latest version, and everyone can work simultaneously on the same project.

Before we dig deeper into what GitHub is, we must examine first what the ‘Git’ part is all about.

#### FREE GIT Training

Learn the basics of GIT[ENROLL NOW](https://www.simplilearn.com/learn-git-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-skillup-course-promotion)

FREE GIT Training

## What is Git?

[Git](https://www.simplilearn.com/tutorials/git-tutorial/git-tutorial-for-beginner) is a version control system used for tracking changes in computer files, making it a top-rated utility for programmers world-wide. Git can handle projects of any size.

Git is used to coordinate the workflow among project team members and track their progress over time. It also benefits both programmers and non-technical users by keeping track of their project files. Git allows multiple users to work together without disrupting each other’s work.

Now that you've been introduced to Git, you have the foundation needed to understand what is GitHub better.

## What is GitHub?

GitHub is a Git repository hosting service that provides a web-based graphical interface. It is the world’s largest coding community. Putting a code or a project into GitHub brings it increased, widespread exposure. Programmers can find source codes in many different languages and use the command-line interface, Git, to make and keep track of any changes.

GitHub helps every team member work together on a project from any location while facilitating collaboration. You can also review previous versions created at an earlier point in time.

So now we know what Git and GitHub are. Time to gain a better understanding of the importance and relevance of what is GitHub by exploring its features.

## What are GitHub’s Features?

### 1. Easy Project Management

GitHub is a place where project managers and developers come together to coordinate, track, and update their work so that projects are transparent and stay on schedule.

### 2. Increased Safety With Packages

Packages can be published privately, within the team, or publicly to the open-source community. The packages can be used or reused by downloading them from GitHub.

### 3. Effective Team Management

GitHub helps all the team members stay on the same page and organized. Moderation tools like Issue and Pull Request Locking help the team to focus on the code.

### 4. Improved Code Writing

[Pull requests](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request) help the organizations to review, develop, and propose new code. Team members can discuss any implementations and proposals through these before changing the source code.

### 5. Increased Code Safety

GitHub uses dedicated tools to identify and analyze vulnerabilities to the code that other tools tend to miss. Development teams everywhere work together to secure the software supply chain, from start to finish.

### 6. Easy Code Hosting

All the code and documentation are in one place. There are millions of repositories on GitHub, and each repository has its own tools to help you host and release code.

Now that we’ve gained some familiarity with GitHub let’s check out the competition.

#### 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

## So How Do You Get Started With GitHub?

It’s easy to get things going with GitHub. For starters, click onto the [GitHub site](https://github.com/" \o "GitHub site" \t "_blank) and create an account. Then, consider installing [Git](https://git-scm.com/downloads)on your system, especially if you plan on using your local computer. Then, go to your terminal and make yourself [known to Git](https://www.simplilearn.com/tutorials/git-tutorial/git-installation-on-windows) by setting up your user name in every repository. Use this command:

git config --global user.name "<your\_name\_here>"

Make sure the “your name here” parameter is your own name. Pick any name you’d like.

Next, share your email address with Git. It should be the same address you entered when you joined GitHub.

git config --global user.email "<[your\_email@email.com](mailto:your_email@email.com)>"

You’re now ready to use Git!

## How Do You Use Git and GitHub?

Here’s a very broad overview of the steps you need to use both Git and GitHub. You can find more details regarding the specific commands and syntax here on [opensource.com](https://opensource.com/article/18/1/step-step-guide-git).

1. Create your GitHub account, which you should have already done, thanks to the previous section!
2. Create a repository or “repo” for short. This is where you store your code.
3. Build a file.
4. Make a commit. Whenever you create a file or change it, you create a Git commit to store the new version.
5. Connect your repo with your computer system.

## GitHub’s Competitors

# https://www.simplilearn.com/tutorials/git-tutorial/what-is-gitlab