How To Set Up Git

Installation git in ubuntu:
=====================
By far the easiest way of getting git installed and ready to use is by using Ubuntu's default repositories. This is the fastest method, but the version may be older than the newest version. If you need the latest release, consider following the steps to compile git from source.

You can use the apt package management tools to update your local package index. Afterwards, you can download and install the program:

```
$ sudo apt-get update

$ sudo apt-get install git
```

once installation is successful check the git version are you installed for confirmation.

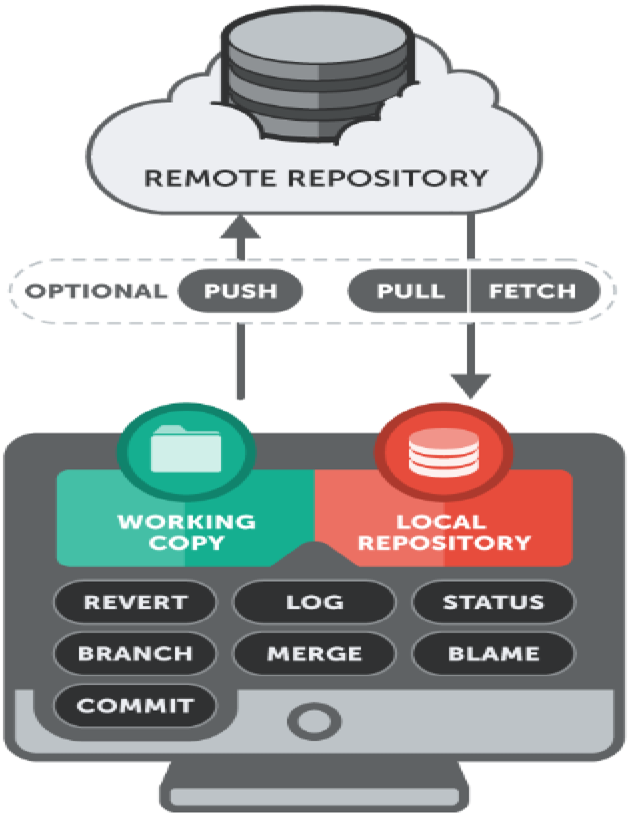
```
$git --version
```

Git installtion in CentOS:-

Use yum, CentOS's native package manager, to search for and install the latest git package available in CentOS's repositories:

```
$ yum install git -y
```

Git architecture:



Now that you have git installed, you need to do a few things so that the commit messages that will be generated for you will contain your correct information.

The easiest way of doing this is through the git config command. Specifically, we need to provide our name and email address because git embeds this information into each commit we do. We can go ahead and add this information by typing:

$ git config --global user.name "Your Name"
$ git config --global user.email "youremail@domain.com"

We can see all of the configuration items that have been set by typing:

$ git config --list

the output looks like this.

```
user.name=Your Name

user.email=youremail@domain.com
```

As you can see, this has a slightly different format. The information is stored in the configuration file, which you can optionally edit by hand with your text editor like this:

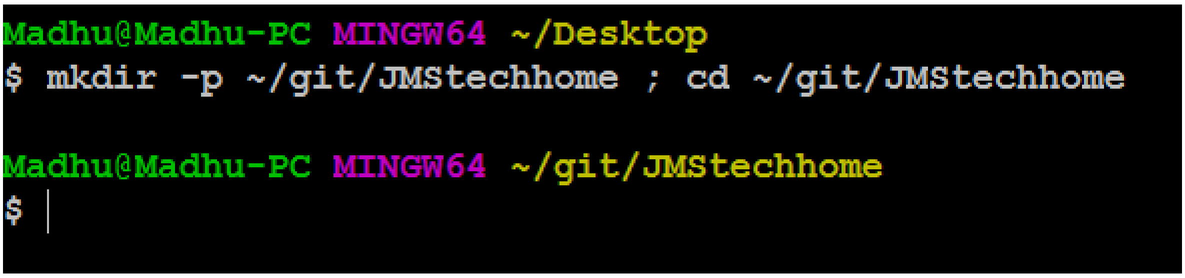
$ nano ~/.gitconfig

Creating your workspace

Just like you want to have a good, clean work environment, the same idea applies to where you do your coding, especially if you're going to contribute to a number of projects at the same time. A good suggestion might be to have a folder called git in your home directory which has subfolders for each of your individual projects.

The first thing we need to do is create our workspace environment:

$ mkdir -p ~/git/JMStechhome ; cd ~/git/JMStechhome



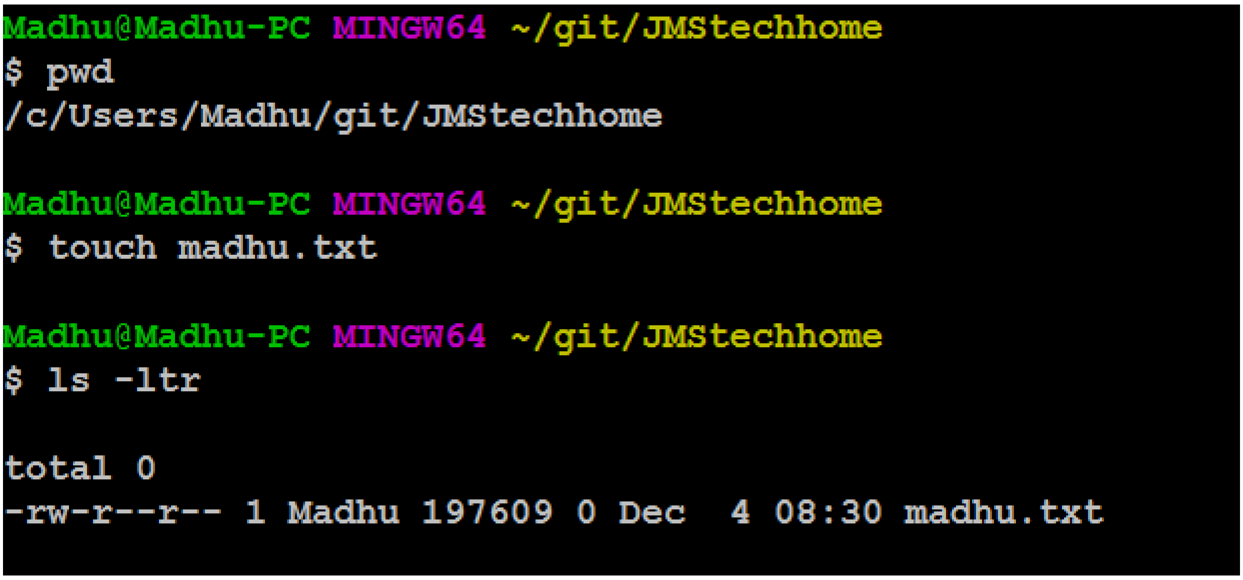
```
Madhu@Madhu-PC MINGW64 ~/Desktop
$ mkdir -p ~/git/JMStechhome ; cd ~/git/JMStechhome

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome
$
```

The above commands will accomplish two things: 1) It creates a directory called "git" in our home directory and then creates a subdirectory inside of that called "JMStechhome" (this is where our project will actually be stored). 2) It brings us to our project's base directory.

Once inside that directory, we need to create a few files that will be in our project. In this step, you can either follow along and create a few dummy files for testing purposes or you can create files/directories you wish that are going to be part of your project.

We are going to create a text file to use in our repository:

~/git/JMStechhome $ touch madhu.txt



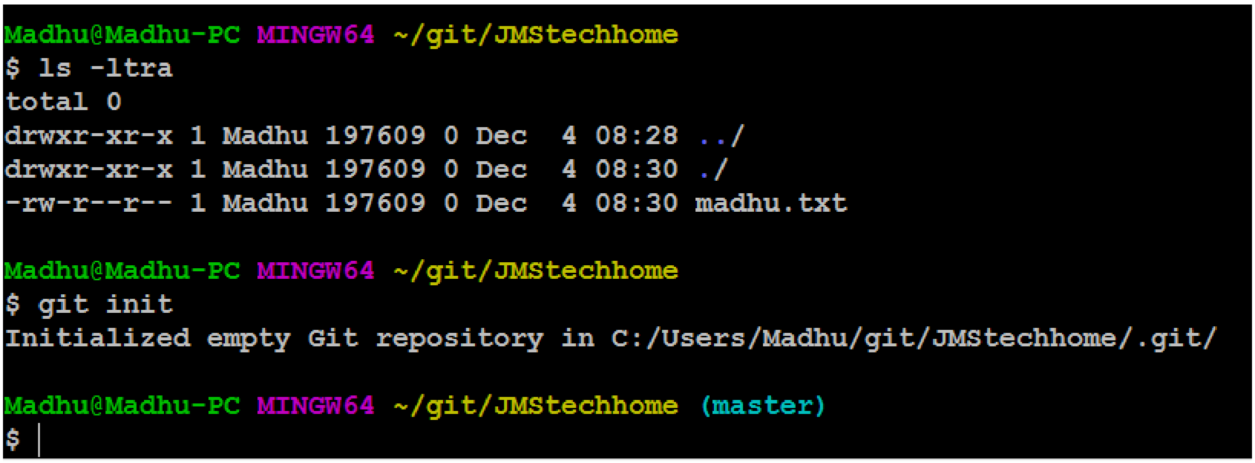
```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome
$ pwd
/c/Users/Madhu/git/JMStechhome

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome
$ touch madhu.txt

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome
$ ls -ltr

total 0
-rw-r--r-- 1 Madhu 197609 0 Dec  4 08:30 madhu.txt
```

Once all your project files are in your workspace, you need to start tracking your files with git. The next step explains that process.

Converting an existing project into a workspace environment

Once all the files are in your git workspace, you need to tell git that you want to use your current directory as a git environment.

~/git/JMStechhome $ git init



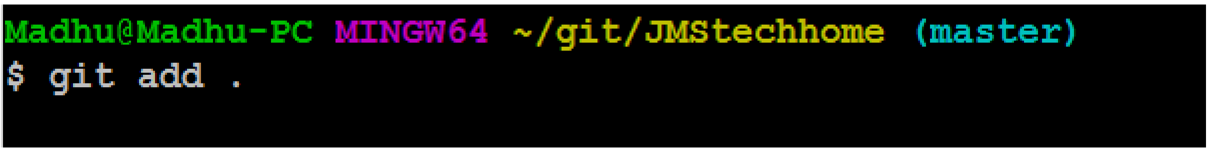
```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome
$ ls -ltra
total 0
drwxr-xr-x 1 Madhu 197609 0 Dec  4 08:28 ../
drwxr-xr-x 1 Madhu 197609 0 Dec  4 08:30 ./
-rw-r--r-- 1 Madhu 197609 0 Dec  4 08:30 madhu.txt

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome
$ git init
Initialized empty Git repository in C:/Users/Madhu/git/JMStechhome/.git/

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ |
```

After git initialization it will create a folder .git. Now our workspace is called git project.

Once you have initialized your new empty repository, you can add your files.

The following will add all files and directories to your newly created repository:

~/git/JMStechhome $ git add .



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git add .
```

In this case, no output is good output. Unfortunately, git does not always inform you if something worked---ahhaaaaaaaaaa

Every time you add or make changes to files, you need to write a commit message.

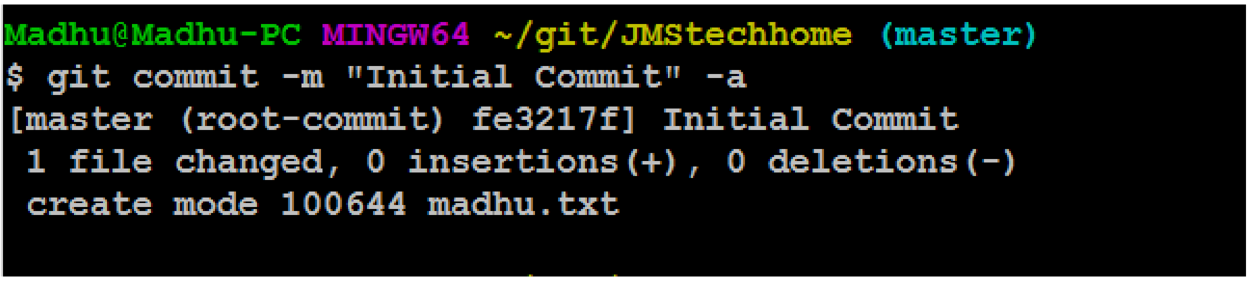
Creating a commit message

A commit message is a short message explaining the changes that you've made. It is required before sending your coding changes off (which is called a push) and it is a good way to communicate to your co-developers what to expect from your changes. This section will explain how to create them.

Commit messages are generally rather short, between one and two sentences explaining what your change did. It is good practice to commit each individual change before you do a push. You can push as many commits as you like. The only requirement for any commit is that it involves at least one file and it has a message. A push must have at least one commit.

Continuing with our example, we are going to create the message for our initial commit:

~/git/JMStechhome $ git commit -m "Initial Commit" –a



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git commit -m "Initial Commit" -a
[master (root-commit) fe3217f] Initial Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 madhu.txt
```

There are two important parameters of the above command. The first is **-m**, which signifies that our commit message (in this case "Initial Commit") is going to follow. Secondly, the **-a** signifies that we want our commit message to be applied to all added or modified files. This is okay for the first commit, but generally you should specify the individual files or directories that we want to commit.

We could have also done:

~/git/JMStechhome $ git commit -m "Initial Commit" <file_name>

To specify a particular file to commit. To add additional files or directories, you just add a space separated list to the end of that command.

~/git/JMStechhome $ git commit -m "Initial Commit" <file_name1> <file_name2>

Pushing changes to a remote server

Up until this point, we have done everything on our local server. That's certainly an option to use git locally, if you want to have any easy way to have version control of your files. If you want to work with a team of developers, however, you're going to need to push changes to a remote server. This section will explain how to do that.

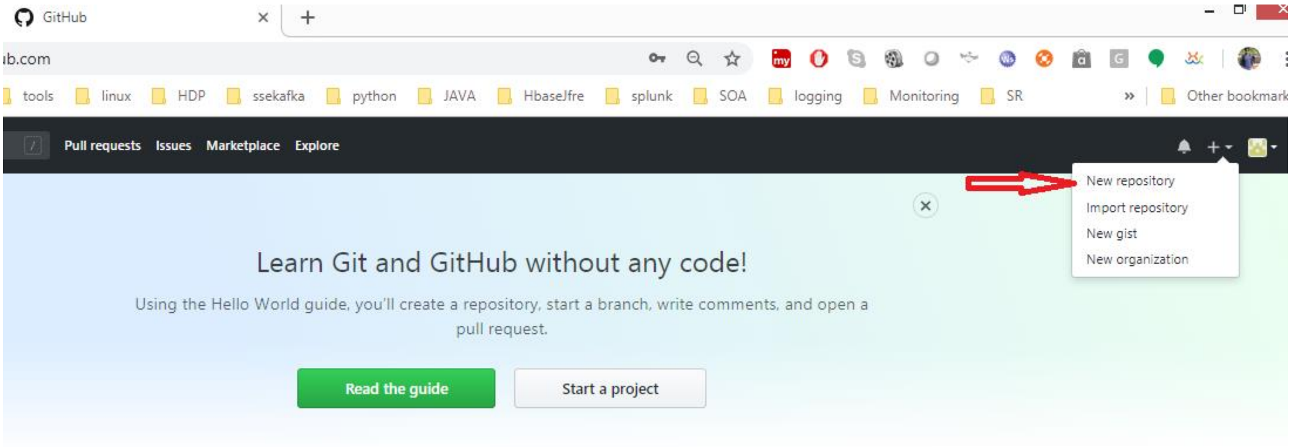
The first step to being able to push code to a remote server is providing the URL where the repository lives and giving it a name. To configure a remote repository to use and to see a list of all remotes (you can have more than one), type the following:

Setup GitHub account:

Step1 : sign-up the github account.

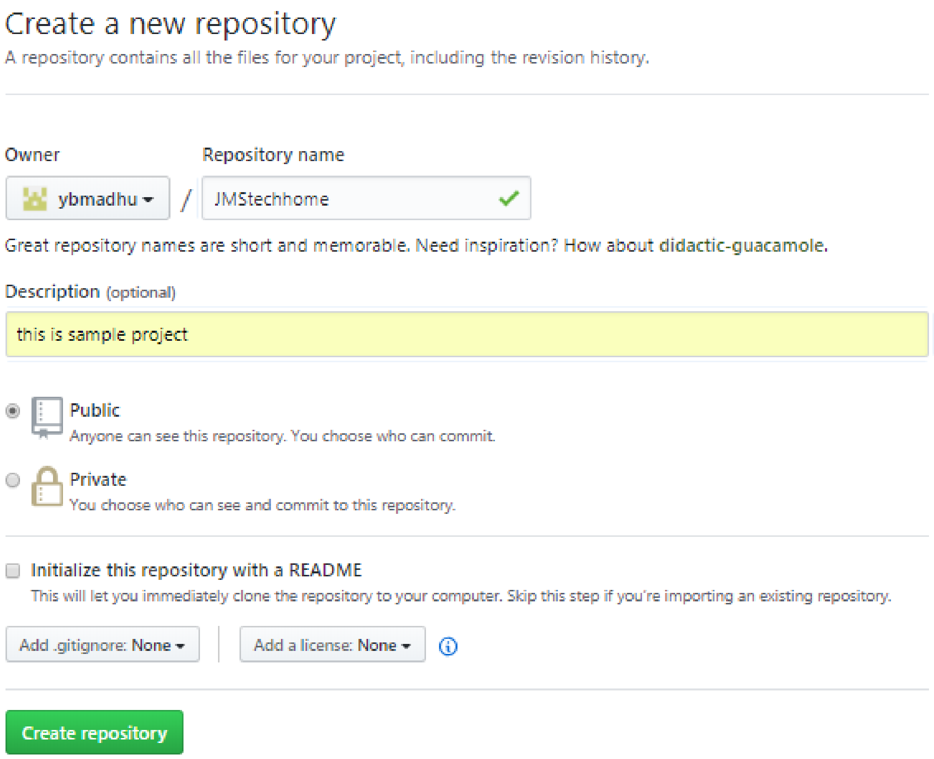
Step2: confirm your git hub account trough your mail.

Step3: then create a repository in github with our main folder name of the git project project.



Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: ybmadhu / Repository name: JMStechhome ✓

Great repository names are short and memorable. Need inspiration? How about didactic-guacamole.

Description (optional): this is sample project

● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

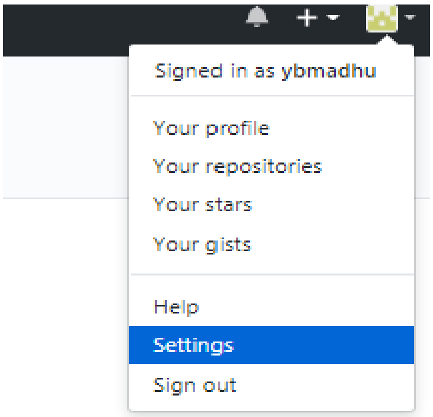
□ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

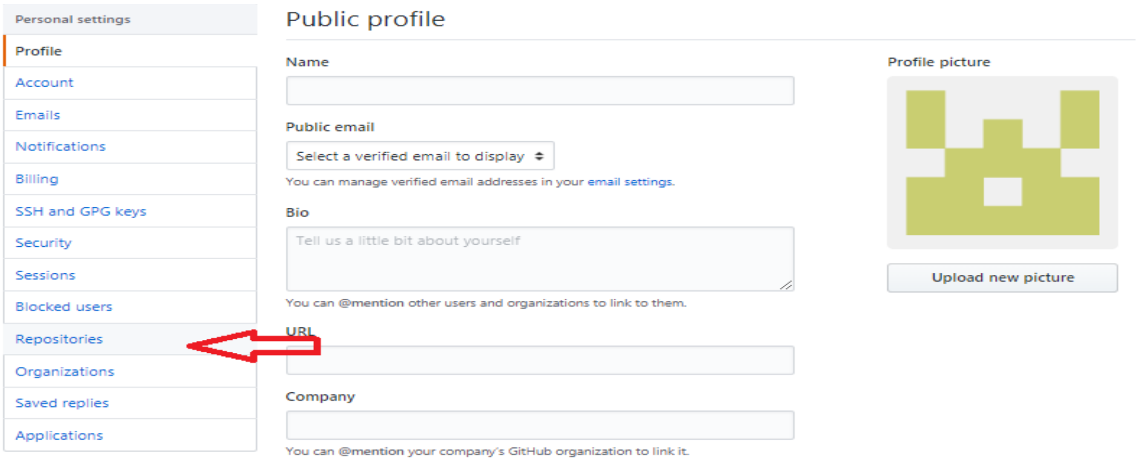
Add .gitignore: None | Add a license: None

Create repository

Once click the Crete repository button it will create a repository in your github account.

If you want to check your repository list goto your profile →settings → then see the right-side panel you will get a repository lists.





Public profile

Name

Public email: Select a verified email to display

You can manage verified email addresses in your email settings.

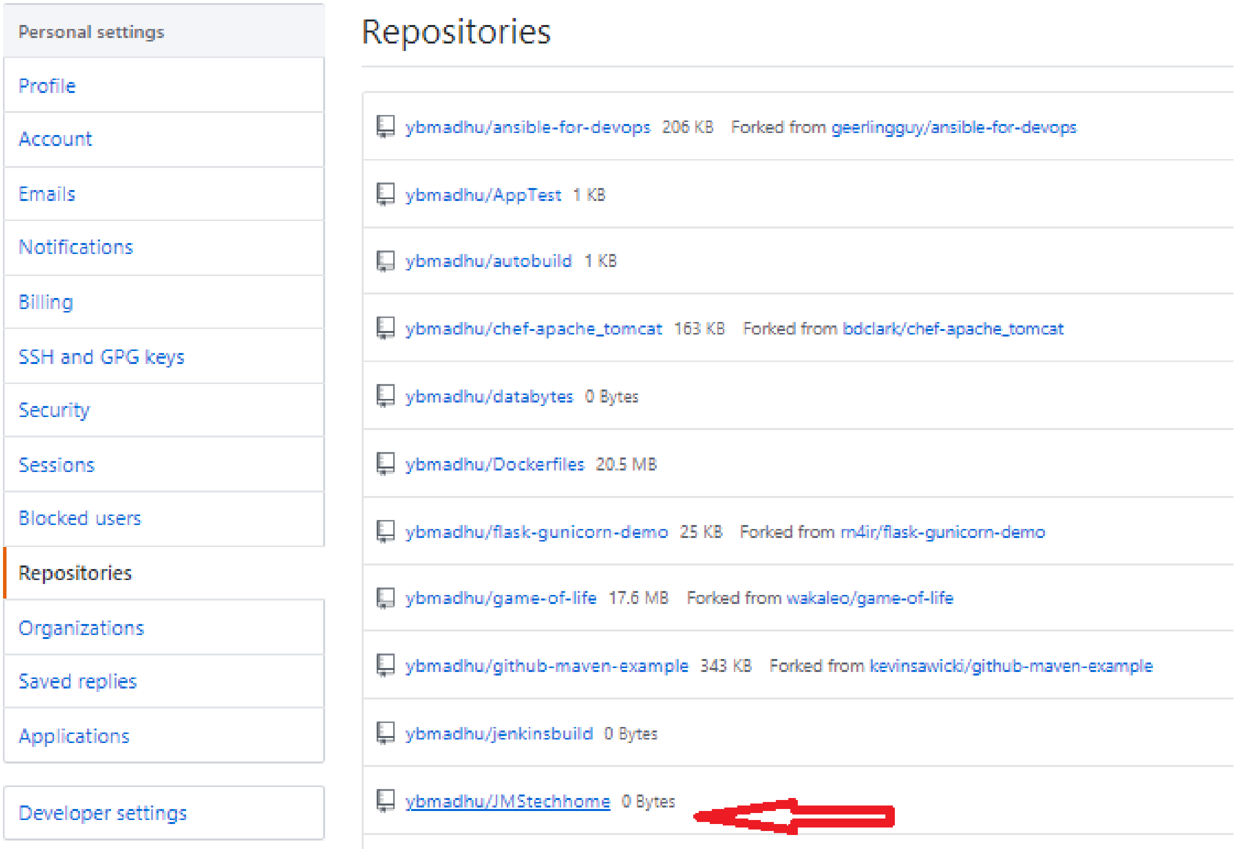
Bio: Tell us a little bit about yourself

You can @mention other users and organizations to link to them.

URL

Company

You can @mention your company's GitHub organization to link it.



Repositories

ybmadhu/ansible-for-devops 206 KB Forked from geerlingguy/ansible-for-devops

ybmadhu/AppTest 1 KB

ybmadhu/autobuild 1 KB

ybmadhu/chef-apache_tomcat 163 KB Forked from bdclark/chef-apache_tomcat

ybmadhu/databytes 0 Bytes

ybmadhu/Dockerfiles 20.5 MB

ybmadhu/flask-gunicorn-demo 25 KB Forked from m4ir/flask-gunicorn-demo

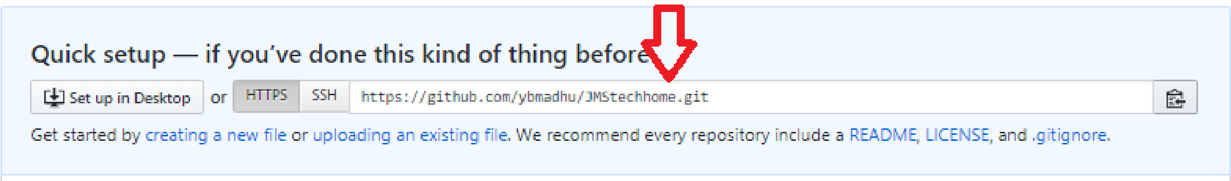
ybmadhu/game-of-life 17.6 MB Forked from wakaleo/game-of-life

ybmadhu/github-maven-example 343 KB Forked from kevinsawicki/github-maven-example

ybmadhu/jenkinsbuild 0 Bytes

ybmadhu/JMStechhome 0 Bytes

Click the repository and see the project structure and repository url. If its new repository it will give us some instructions what are the steps needs to be follow.



Quick setup — if you've done this kind of thing before

https://github.com/ybmadhu/JMStechhome.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

If you click the 📋 it will copy the url. And we need to configure this url to in our local machine.

Then go to your git bash terminal of the project add the url like below. Mygithub repository url look like below syntax.

https://github.com/*USERNAME*/*REPOSITORY*.git (or)

ssh://git@github.com:*USERNAME*/*REPOSITORY*.git

$ git remote add origin https://github.com/*ybmadhu*/JMStechHome.git

to check the remote repositories use below command.

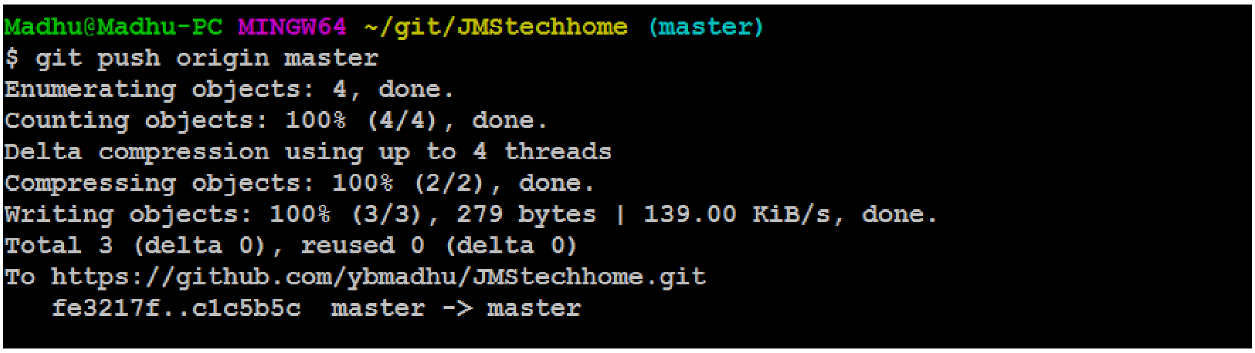
The first command adds a remote, called "origin", and sets the URL to

You can name your remote whatever you'd like, but the URL needs to point to an actual remote repository. For example, if you wanted to push code to GitHub, you would need to use the repository URL that they provide.

Once you have a remote configured, you are now able to push your code.

You can push code to a remote server by typing the following:

$ git push origin master



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 279 bytes | 139.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ybmadhu/JMStechhome.git
   fe3217f..c1c5b5c  master -> master
```

if you face the issue like below

$ git push origin master

The authenticity of host 'github.com (192.30.253.113)' can't be established.

RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'github.com,192.30.253.113' (RSA) to the list of known hosts.

Permission denied (publickey).

fatal: Could not read from remote repository.

Please make sure you have the correct access rights

and the repository exists.

ANSWER: you must provide authentication before you push the code use https url to push the code and it will ask user name and password of github account.

$ git remote set-url origin https://github.com/ybmadhu/JMStechhome.git

$ git remote -v

origin https://github.com/ybmadhu/JMStechhome.git (fetch)

origin https://github.com/ybmadhu/testing.git (push)

```
$ git push origin master

Username for 'https://github.com': ybmadhu

Password for 'https://ybmadhu@github.com':

Counting objects: 3, done.

Writing objects: 100% (3/3), 207 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/ybmadhu/JMStechhome.git

 * [new branch]      master -> master
```

then go and check your git hub account in that particular repository you files is pushed successfully or not.

"git push" tells git that we want to push our changes, "origin" is the name of our newly-configured remote server and "master" is the name of the first branch.

In the future, when you have commits that you want to push to the server, you can simply type "git push".

Next we have got new requirement then we need to change the same file or create new file in the same project.

Then edit the file or create new files in the same project.

Then check the status of the project and modified file using below command.

```
madhu@madhu-VirtualBox:~/git/testing$ git status

On branch master

Changes not staged for commit:

  (use "git add <file>..." to update what will be committed)

  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   fileno changes added to commit (use "git add" and/or "git commit -a").
```

Then we need to commit the changes in local workspace then after push to remote repository.

```
$ git add .

$ git commit -m "this is sencond change" -a

[master 1fcc24d] this is sencond change

 1 file changed, 1 insertion(+)
```

- To push the code to remote repository use below command.

```
$ git push origin master
Username for 'https://github.com': ybmadhu
Password for 'https://ybmadhu@github.com':
Counting objects: 5, done.
Writing objects: 100% (3/3), 257 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ybmadhu/JMStechhome.git
   9dd2953..1fcc24d  master -> master
madhu@madhu-VirtualBox:~/git/testing$ git status
On branch master
nothing to commit, working directory clean.
```

- To check the user activity using log command. Think of Git's log as a journal that remembers all the changes we've committed so far, in the order we committed them

```
$ git log
commit 1fcc24deeb90ba6bfd2af01e05805e622d7f3c84
Author: ybmadhu<ybmadhu707@gmail.com>
Date:   Thu Dec 14 15:50:42 2017 +0530
this is sencond change
commit 9dd29534376ca19714ae649e098435b5d0ed4927
Author: ybmadhu<ybmadhu707@gmail.com>
Date:   Thu Dec 14 13:41:14 2017 +0530
    Initial Commit
```

Pulling Remotely

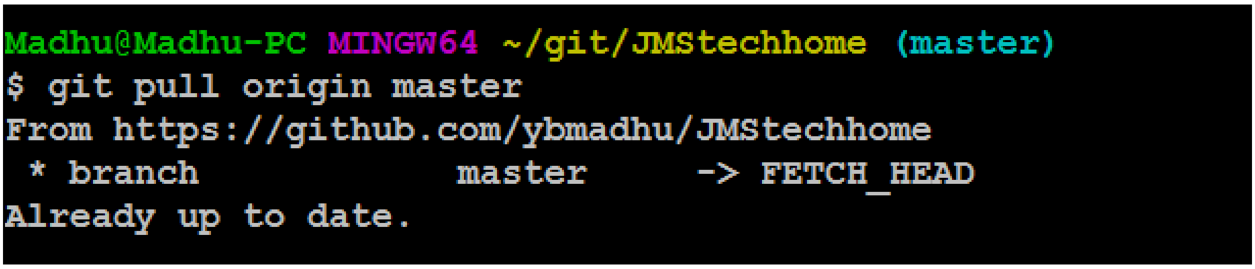
Let's pretend some time has passed. We've invited other people to our GitHub project who have pulled your changes, made their own commits, and pushed them.

We can check for changes on our GitHub repository and pull down any new changes by running:

- if there is no changes in remote repository the out is look like this

$ git pull origin master

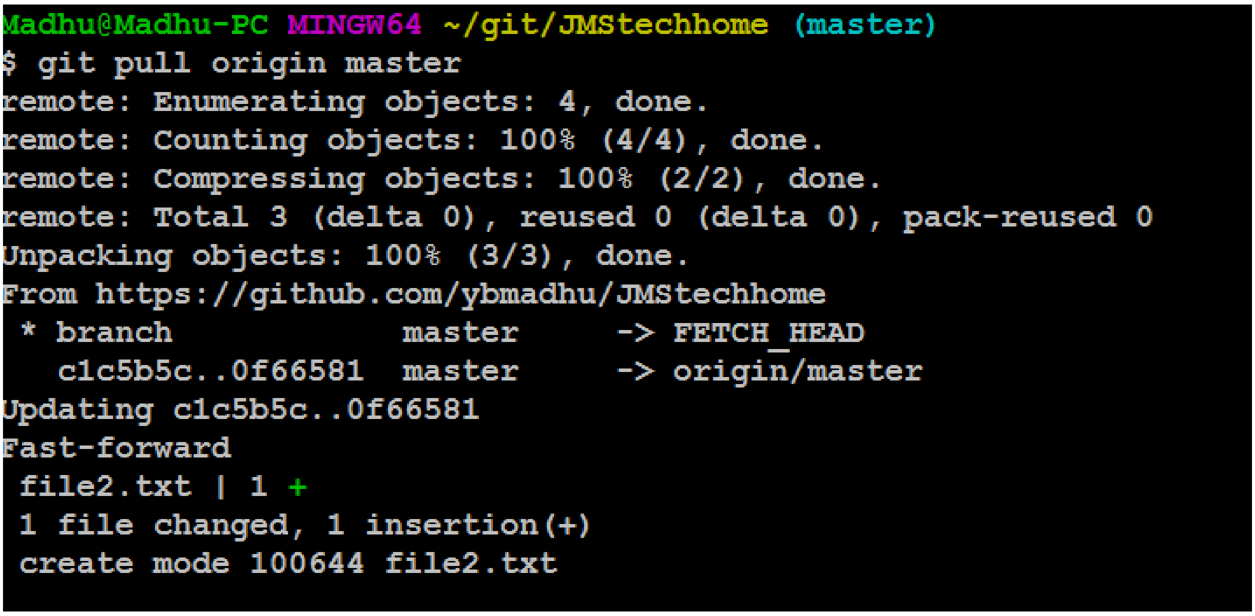
$ git pull origin master



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git pull origin master
From https://github.com/ybmadhu/JMStechhome
 * branch            master     -> FETCH_HEAD
Already up to date.
```

- ❖ if any changes in remote repository the output is look like below.

madhu@madhu-VirtualBox:~/git/testing$ git pull origin master



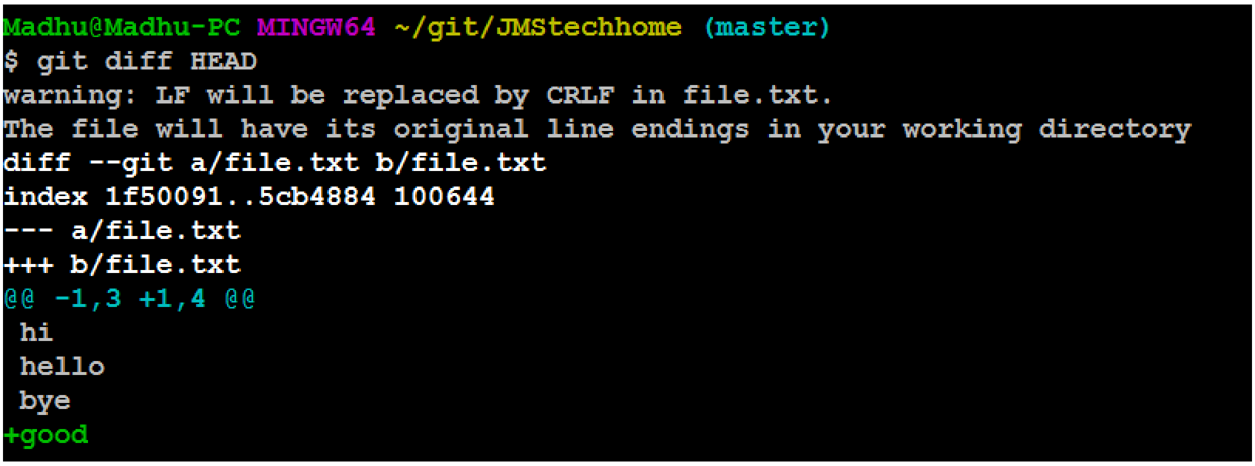
```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/ybmadhu/JMStechhome
 * branch            master     -> FETCH_HEAD
   c1c5b5c..0f66581  master     -> origin/master
Updating c1c5b5c..0f66581
Fast-forward
 file2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt
```

Git Differences

Let's take a look at what is different from our last commit by using the git diff command.

In this case we want the diff of our most recent commit, which we can refer to using the HEAD pointer.

$ git diff HEAD



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git diff HEAD
warning: LF will be replaced by CRLF in file.txt.
The file will have its original line endings in your working directory
diff --git a/file.txt b/file.txt
index 1f50091..5cb4884 100644
--- a/file.txt
+++ b/file.txt
@@ -1,3 +1,4 @@
 hi
 hello
 bye
+good
```

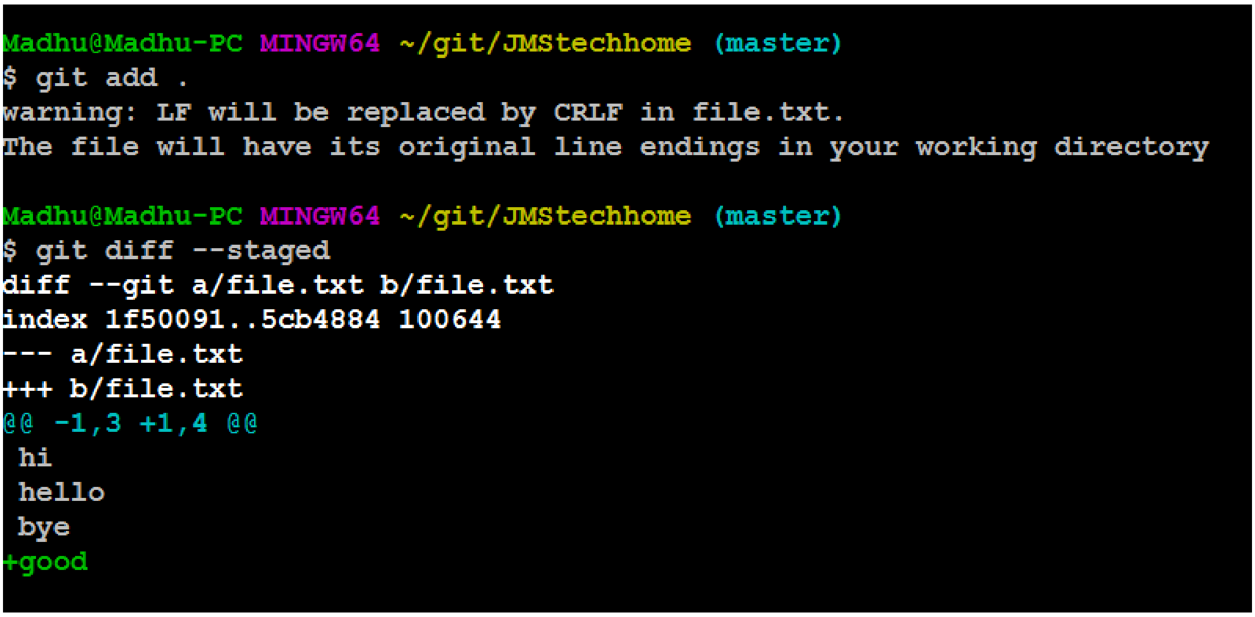
Staged Differences

Another great use for diff is looking at changes within files that have already been staged. Remember, staged files are files we have told git that are ready to be committed.

Staged Differences (cont'd)

Good, now go ahead and run git diff with the --staged option to see the changes you just staged.

$ git diff --staged



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git add .
warning: LF will be replaced by CRLF in file.txt.
The file will have its original line endings in your working directory

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git diff --staged
diff --git a/file.txt b/file.txt
index 1f50091..5cb4884 100644
--- a/file.txt
+++ b/file.txt
@@ -1,3 +1,4 @@
 hi
 hello
 bye
+good
```

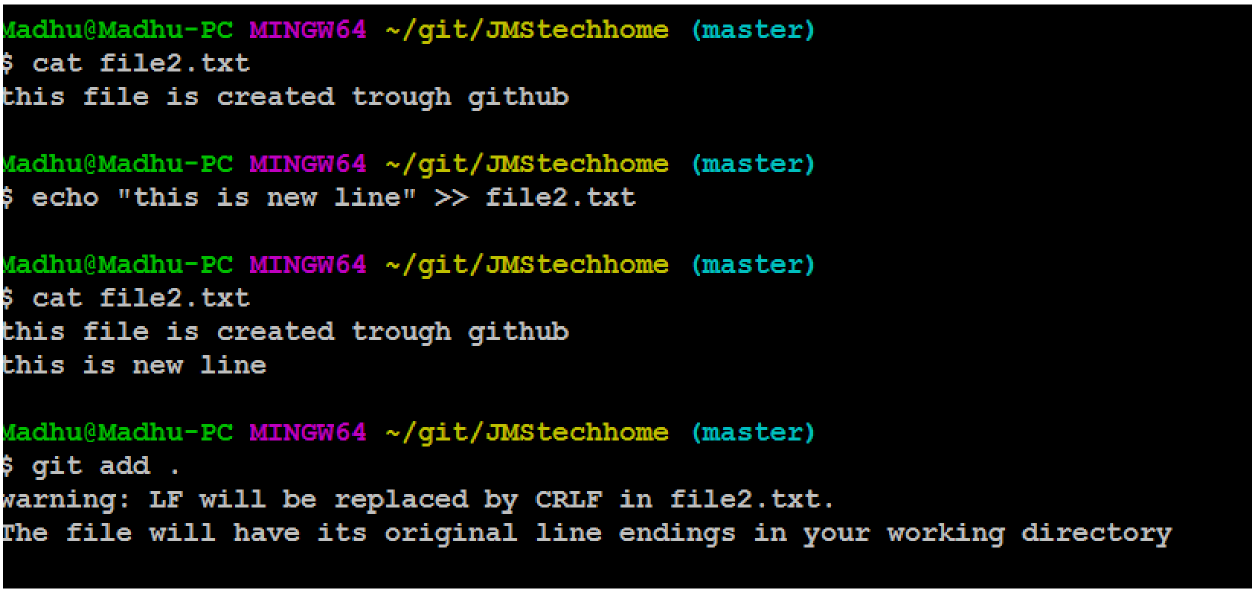
Undoing the Last Commit:

Suppose I want delete my last commit due to some wrong code committed. Now I want undo my committed code. In this scenario we will use **git reset** commands like below.

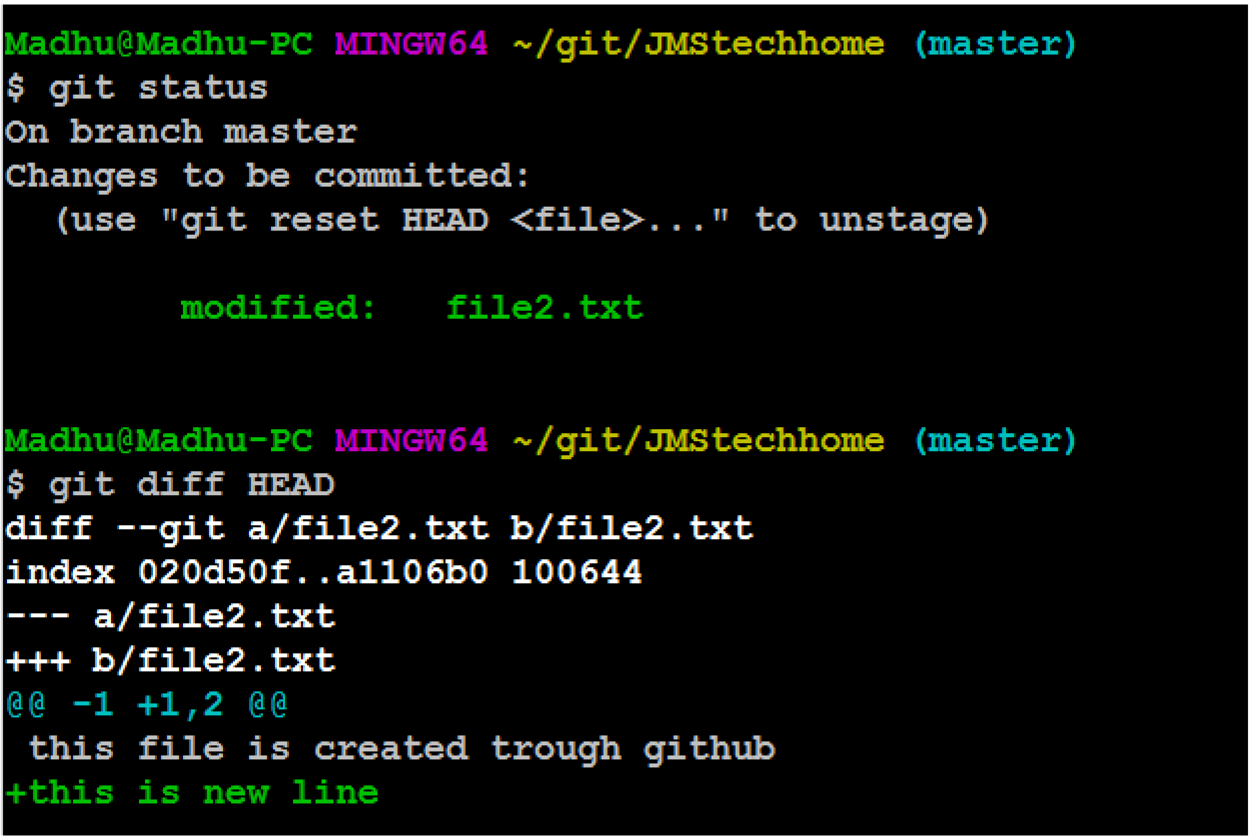
There are two option are available for undo 1.--soft and –hard.

1. If I use --soft option we can do undo last committed changes but the code still available in your local.

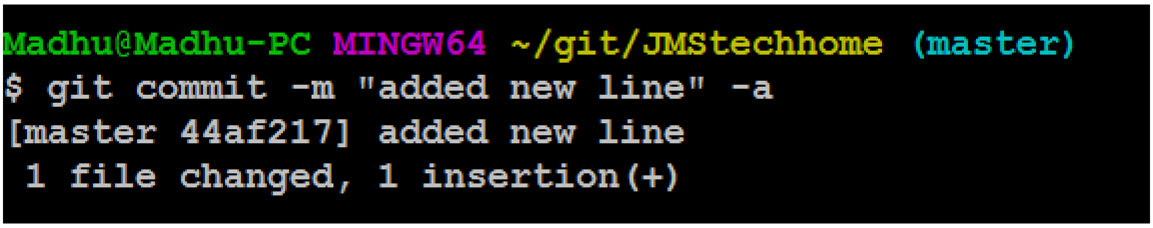
2. If I use --hard option we can do undo last committed changes and latest code also deleted in your local.



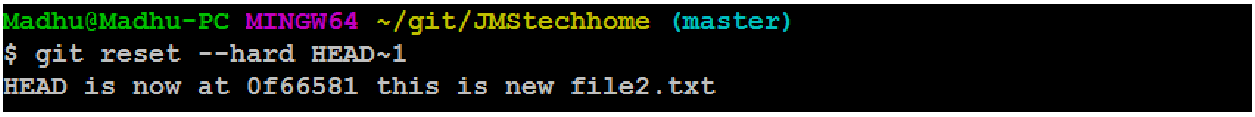
```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ cat file2.txt
this file is created trough github

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ echo "this is new line" >> file2.txt

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ cat file2.txt
this file is created trough github
this is new line

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git add .
warning: LF will be replaced by CRLF in file2.txt.
The file will have its original line endings in your working directory
```



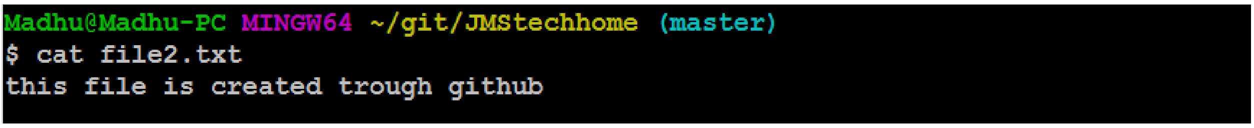
```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file2.txt


Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git diff HEAD
diff --git a/file2.txt b/file2.txt
index 020d50f..a1106b0 100644
--- a/file2.txt
+++ b/file2.txt
@@ -1 +1,2 @@
 this file is created trough github
+this is new line
```



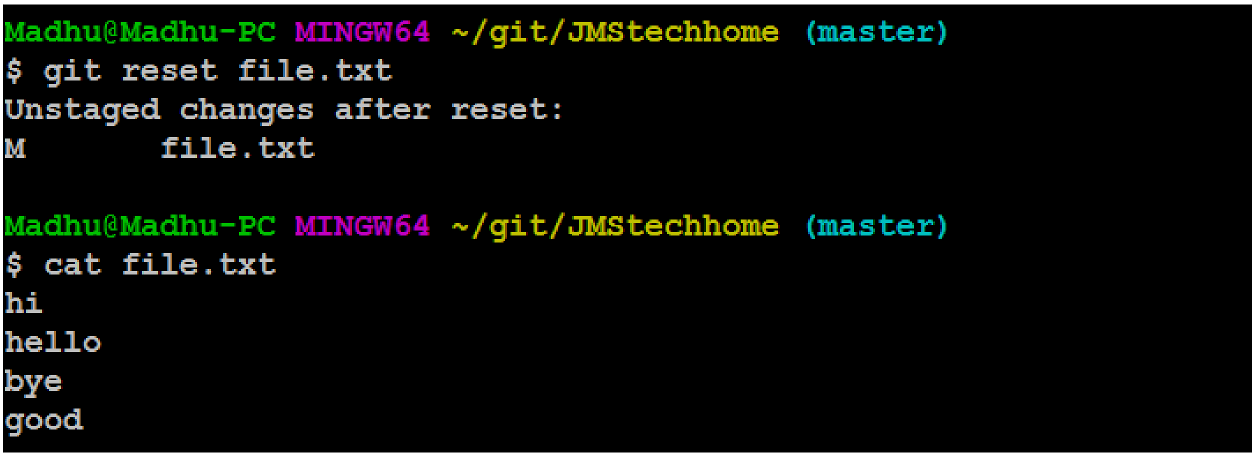
```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git commit -m "added new line" -a
[master 44af217] added new line
 1 file changed, 1 insertion(+)
```



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git reset --hard HEAD~1
HEAD is now at 0f66581 this is new file2.txt
```



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ cat file2.txt
this file is created trough github
```



```
Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ git reset file.txt
Unstaged changes after reset:
M       file.txt

Madhu@Madhu-PC MINGW64 ~/git/JMStechhome (master)
$ cat file.txt
hi
hello
bye
good
```

Undo

Files can be changed back to how they were at the last commit by using the command: git checkout -- <target>.

madhu@madhu-VirtualBox:~/git/testing$ git checkout -- file

reference

========

https://itnext.io/become-a-git-pro-in-just-one-blog-a-thorough-guide-to-git-architecture-and-command-line-interface-93fbe9bdb395