How to start
with git and create a
repository in Github??

GitHub

Bitbucket

Mithilesh Singh

GIT

- Git is a version control system. It has distributed architecture.
- It provides a flexible environment to work collaboratively and securely.
- You can create and merge repositories using Git.
- Git facilitate the branching and merging of files within the main repository.
- Git allows every user to have a copy of the project files locally, thus making the files in the server secure.
- There are three level of Configuration in git, local, system and global level.
- User settings in Git are mainly defined at global level, because they correspond user level settings for all repositories.

Linus Torvalds

Version Control System

It is also known as "**revision control**", "**source control**", or "**source code management**".

Why **Version Control System(Git)**?

- Have you ever made changes to a file or code by mistake that you wanted to revert later?
- Ever lost a code or file?
- Do you have a backup of code which you write or change every day.
- Do you ever have maintain multiple versions of a code or file and analyze the differences.
- Do you ever have to share your code or let other people work on it?

- Do you ever need to track the work being done in details?
- Do you need to experiment with your codes without messing up the current work?

Benefits of Version Control System

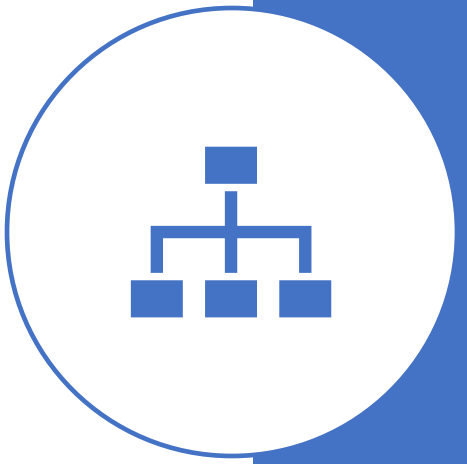
Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change.

Enhances the project development speed by providing efficient collaboration.

For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is **Git, Helix core,**

Types of Version Control Systems

- There are two main types of version control systems:
  a. **centralized** and b. **distributed** version control systems.



- **a**. Centralized version control systems(CVCS)
- **b**. Distributed version control systems(DVCS)

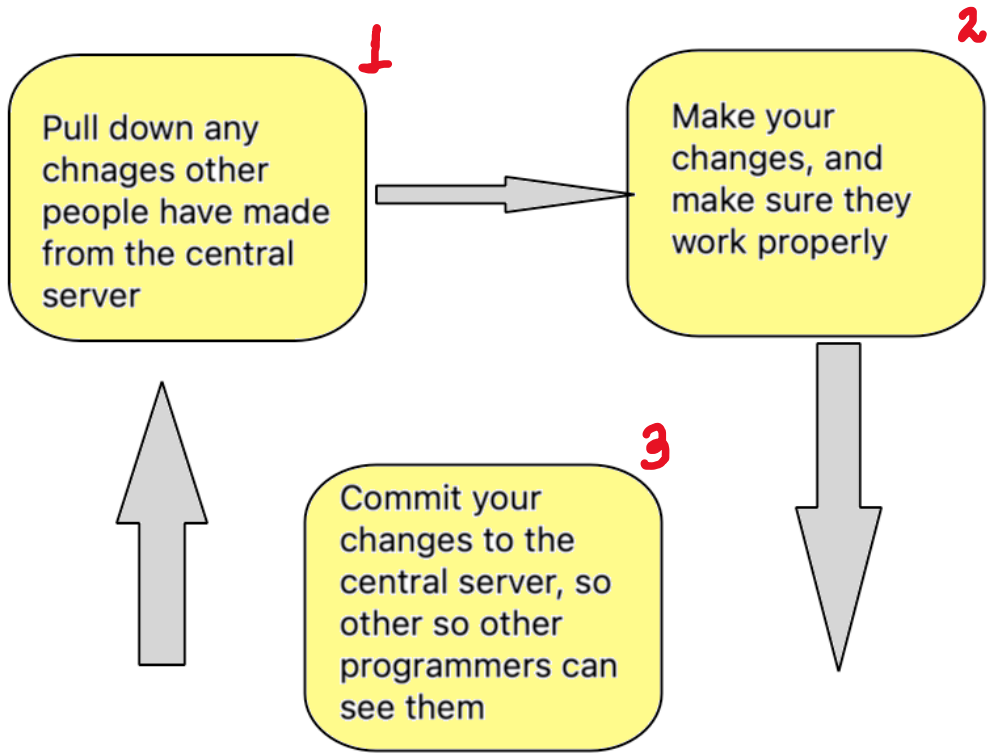
Centralized version control systems:

It is a type of version control system with a single server containing all version of the code files.

This set up helps every contributor(Programmer) to know what the others are working on.

Programmer will "commit" their changes to this central server.

Note:
Commiting a change simply means recording the changes in the central system. Other programmers can then see this change.They can also pull down changes.



Centralized Version Control Workflow

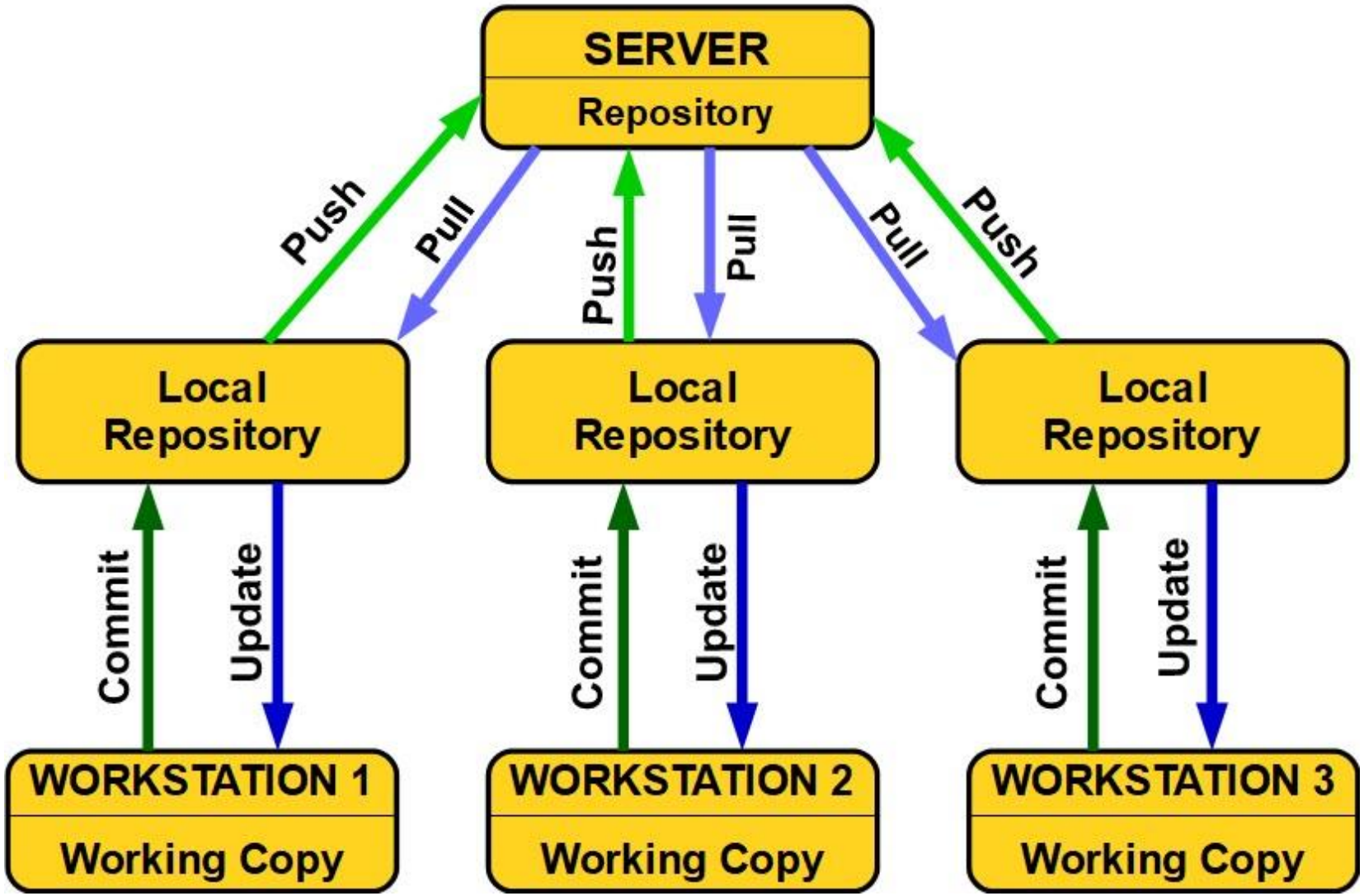
Downside of CVCS

- It has some **downsides** as well which led to the development of DVS. The most obvious is the single point of failure that the centralized repository represents if it goes down during that period collaboration and saving versioned changes is not possible. What if the hard disk of the central database becomes corrupted, and proper backups haven't been kept? You lose absolutely everything

Distributed Version Control System

- Distributed version control systems contain multiple repositories. Each user has their own repository and working copy. Just committing your changes will not give others access to your changes. This is because commit will reflect those changes in your local repository and you need to push them in order to make them visible on the central repository. Similarly, When you update, you do not get other's changes unless you have first pulled those changes into your repository.

Distributed Version Control System



Key Points

It is a type of version control system, this system is not rely on the central server to store the project files version, instead every developer clones a copy of the repository and has the full history of project.

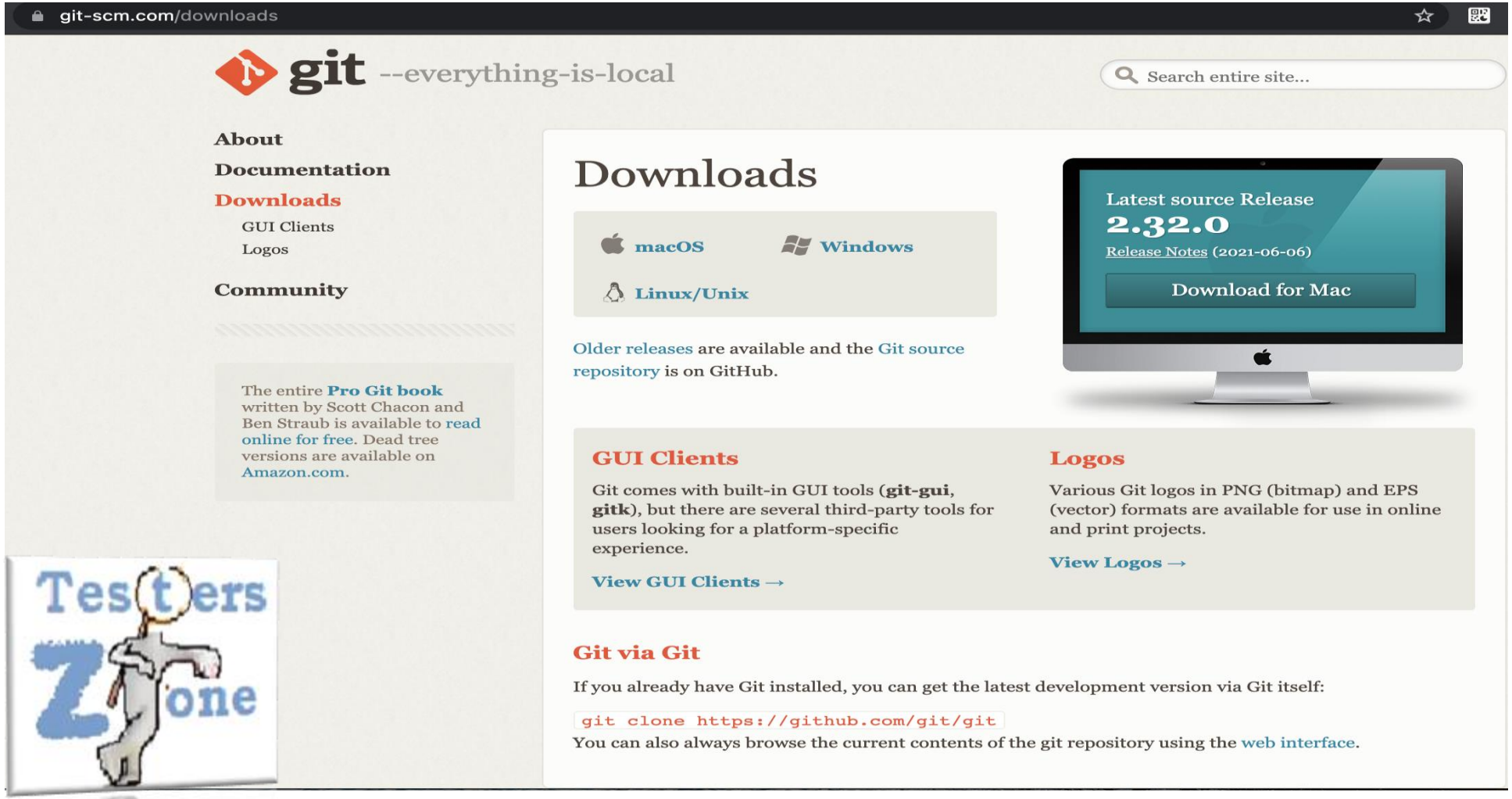
This copy has all the metadata of the original
Note: Metadata summarizes basic information about data. Or provide information about other data.

Advantages of DCVS:

- Performing actions other than pushing and pulling changesets is extremely fast because the tool only needs to access the hard drive, not a remote server.
- Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.
- Since each programmer has a full copy of the project repository, they can share changes with one or two other people at a time if they want to get some feedback before showing the changes to everyone.

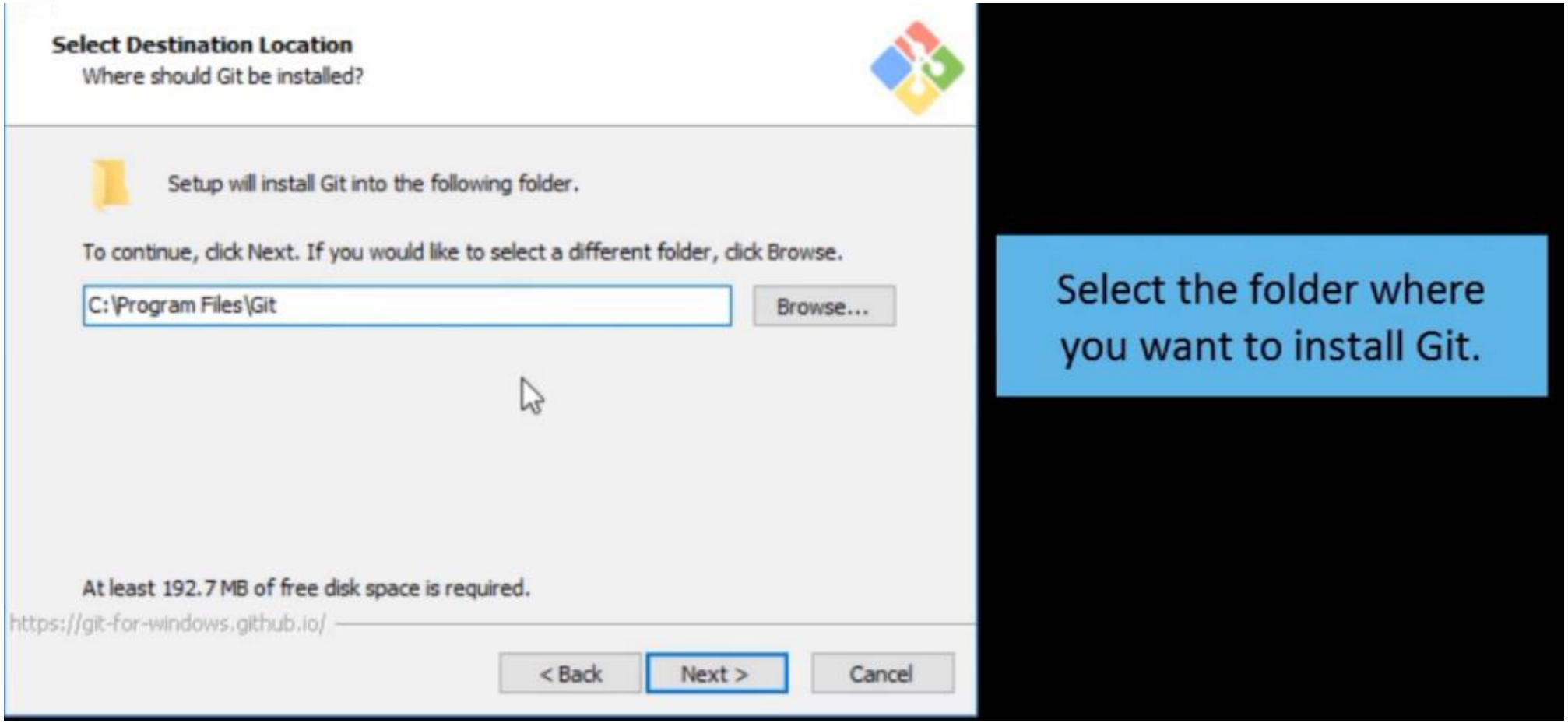
GIT Installation in the Window

- Access the URL: https://git-scm.com/downloads

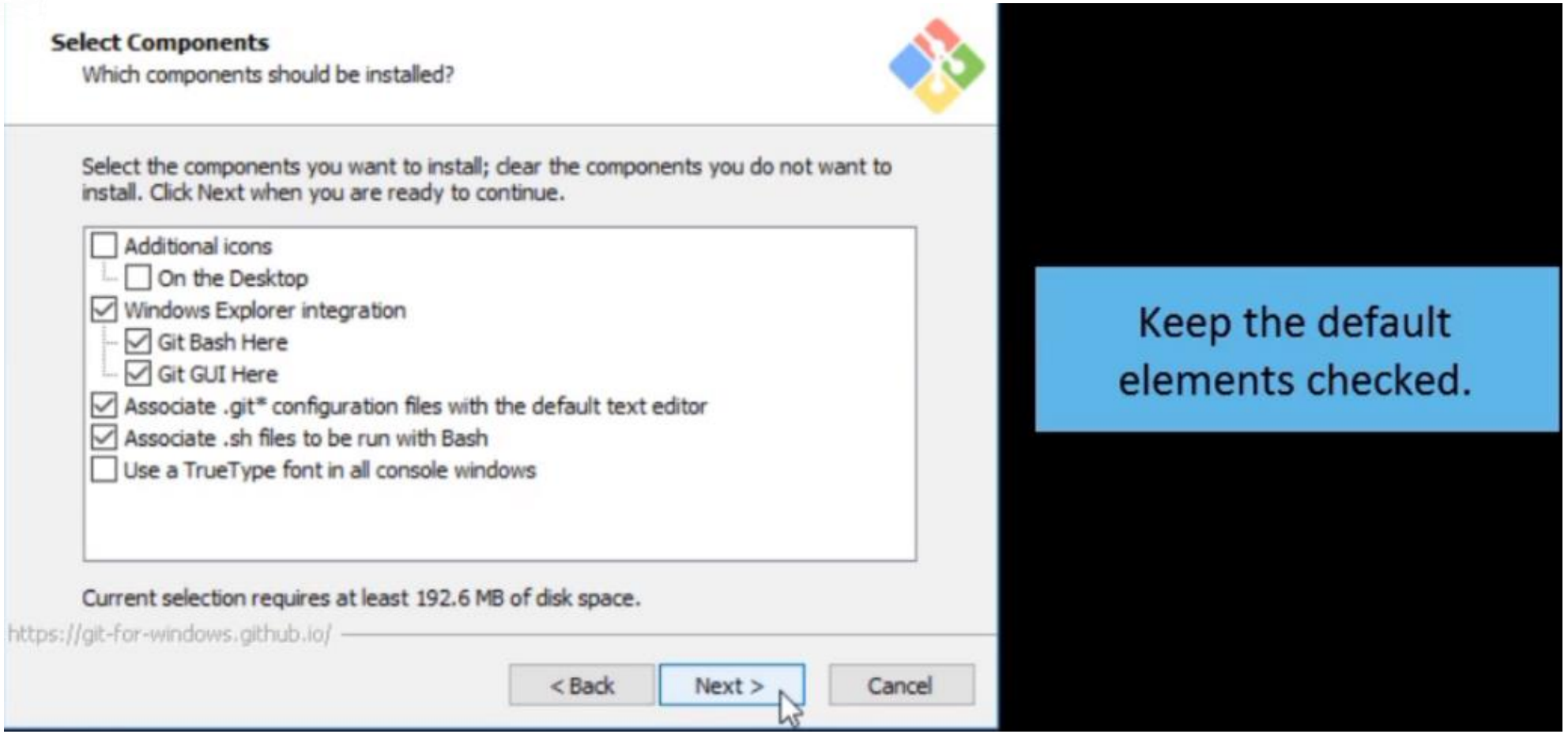


Open the downloaded .exe file and click next

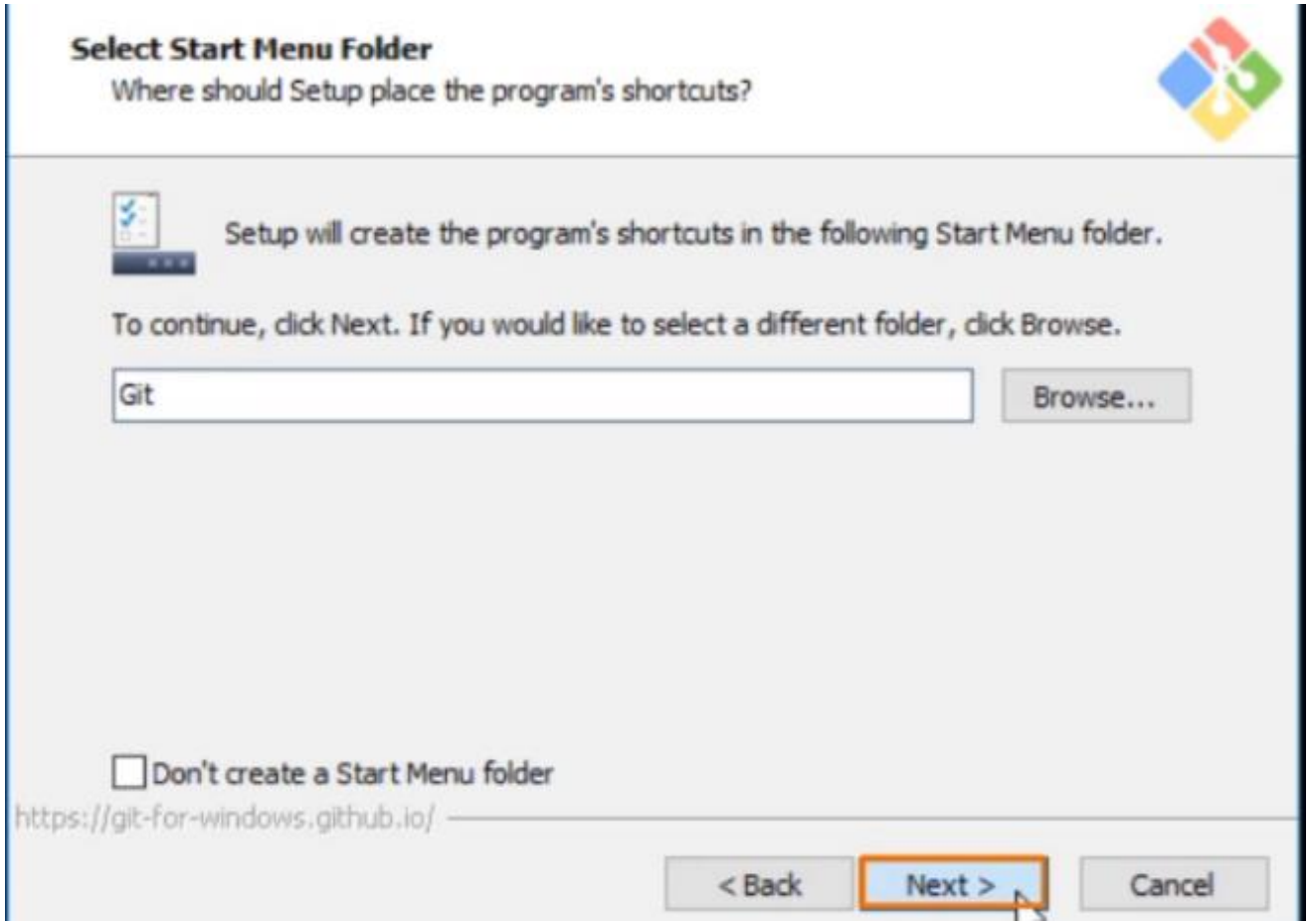
Step: 2


Select Destination Location
Where should Git be installed?
Setup will install Git into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Git
Browse...
At least 192.7 MB of free disk space is required.
https://git-for-windows.github.io/
< Back
Next >
Cancel
Select the folder where you want to install Git.

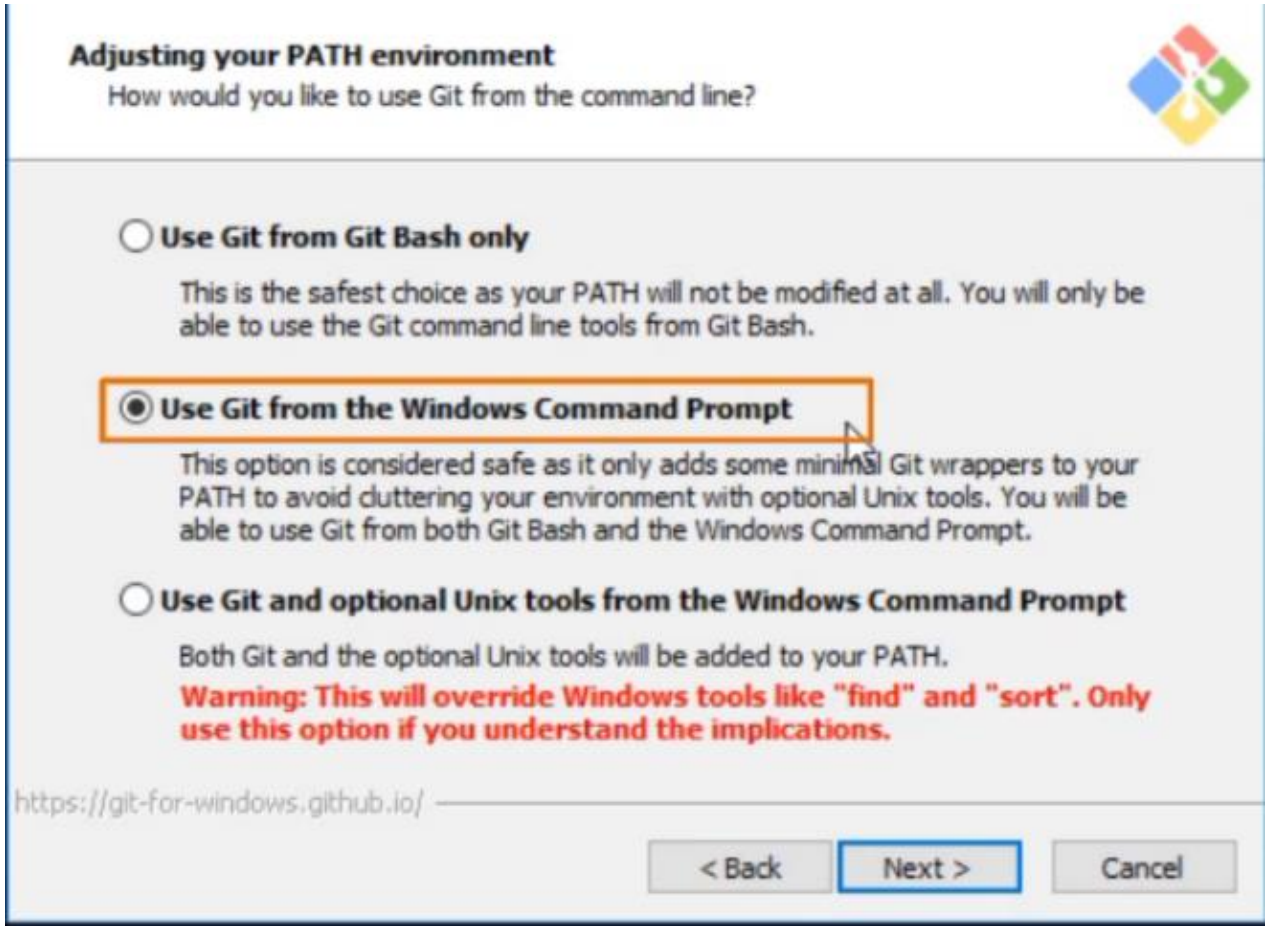
Step : 3


Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Current selection requires at least 192.6 MB of disk space.
https://git-for-windows.github.io/
< Back
Next >
Cancel
Keep the default elements checked.

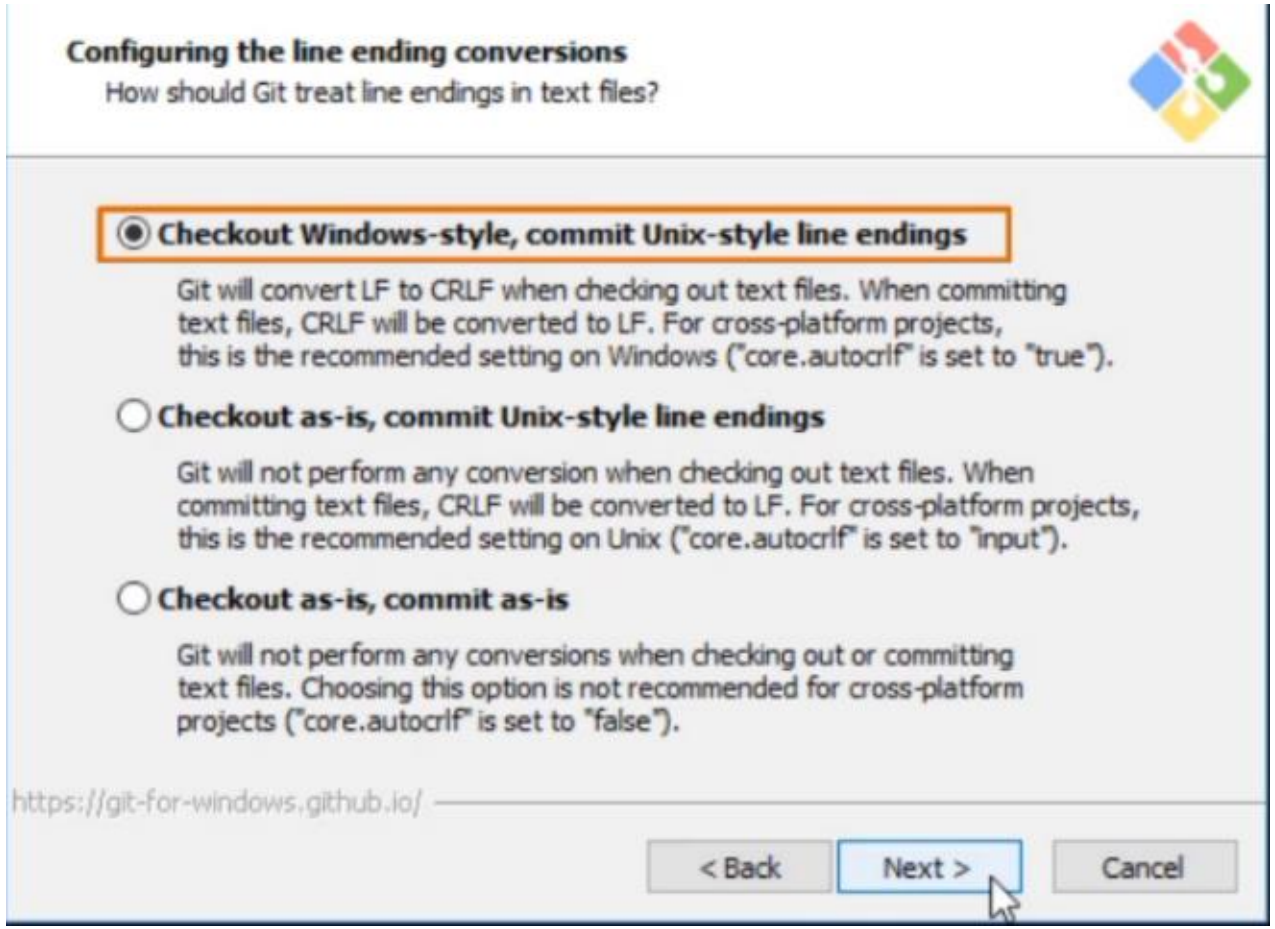
Step : 4


Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
https://git-for-windows.github.io/
< Back
Next >
Cancel

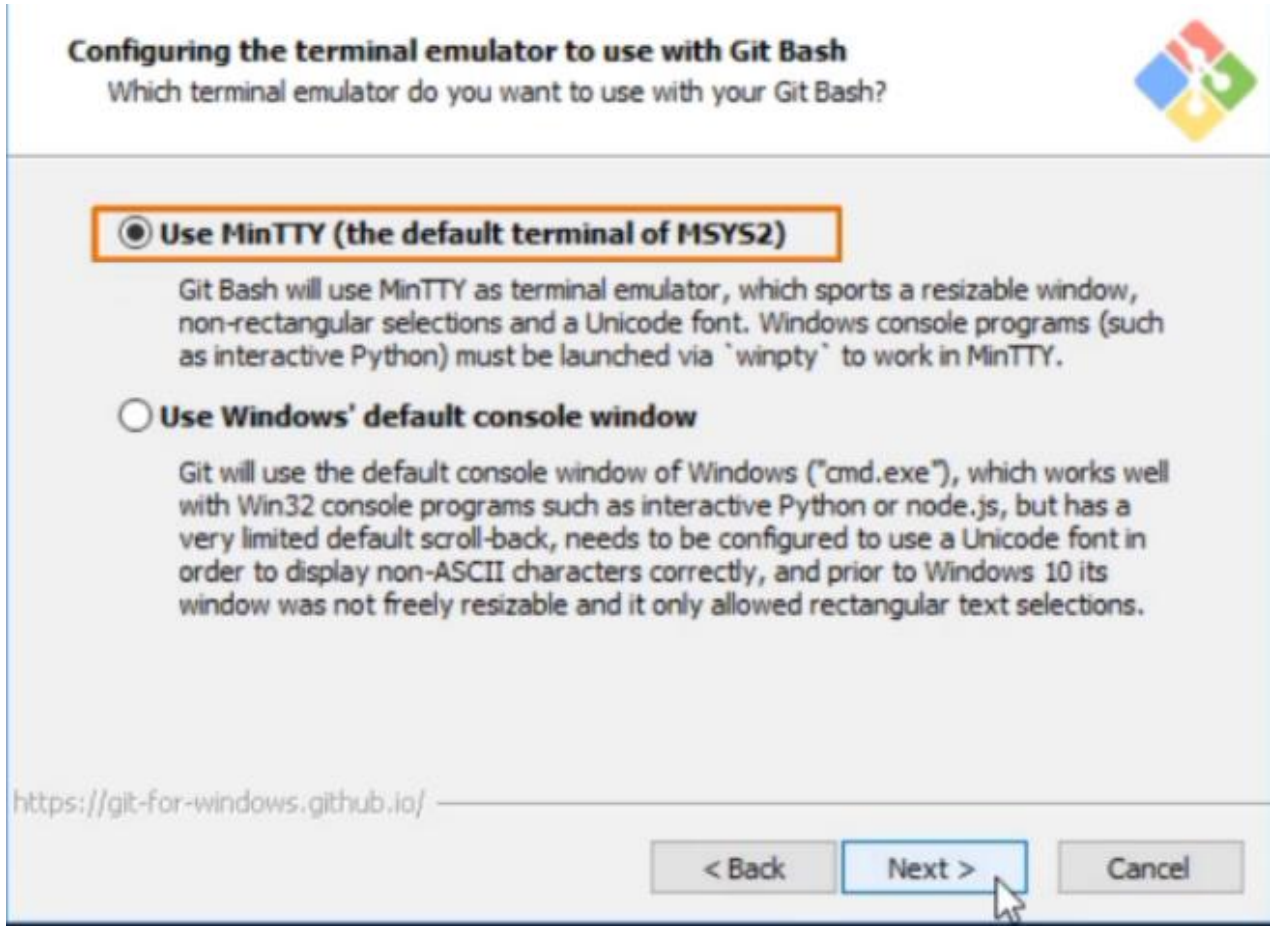
Step : 5


Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Use Git from the Windows Command Prompt
This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.
Use Git and optional Unix tools from the Windows Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://git-for-windows.github.io/
< Back
Next >
Cancel

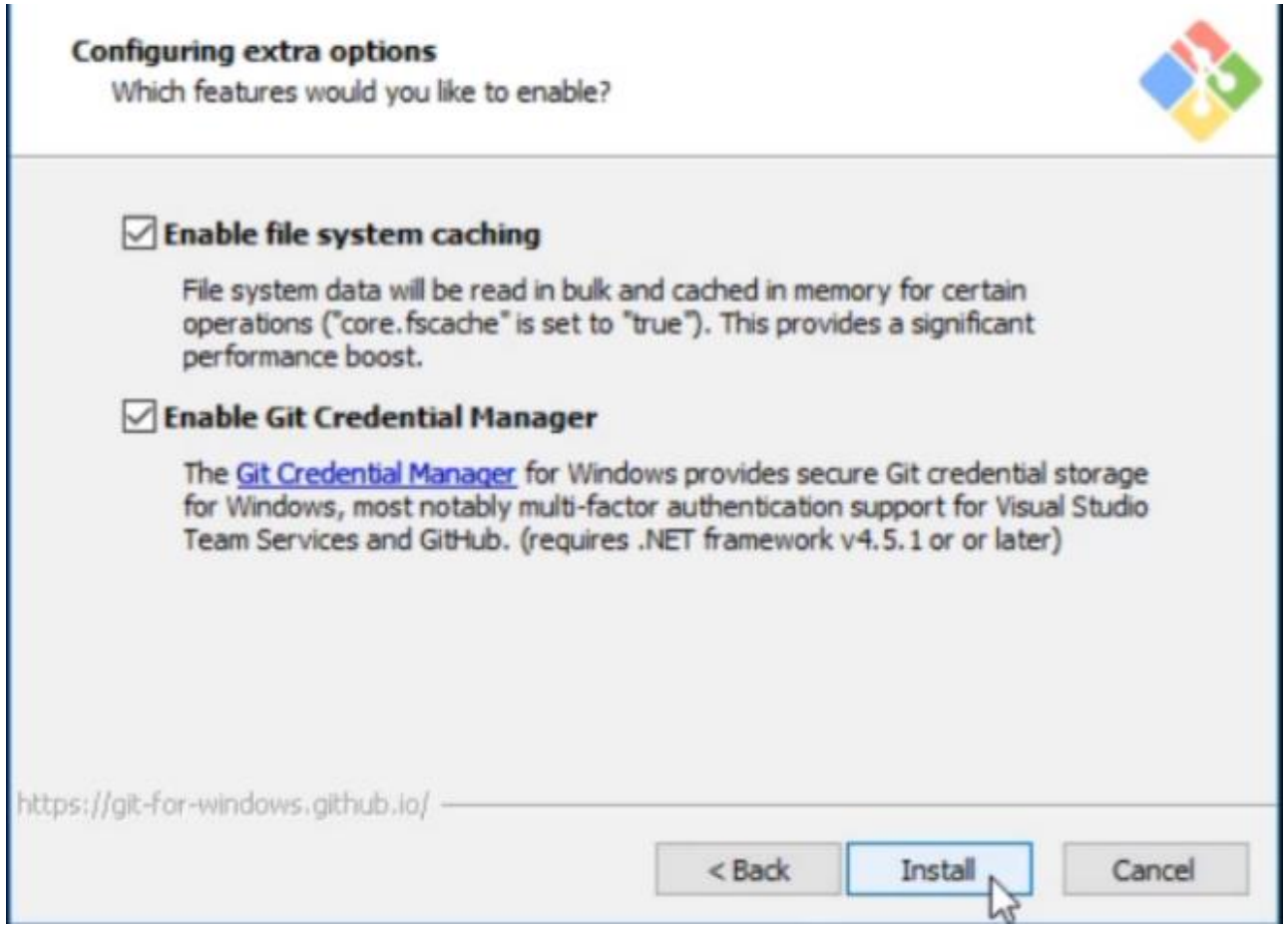
Step : 6


Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://git-for-windows.github.io/
< Back
Next >
Cancel

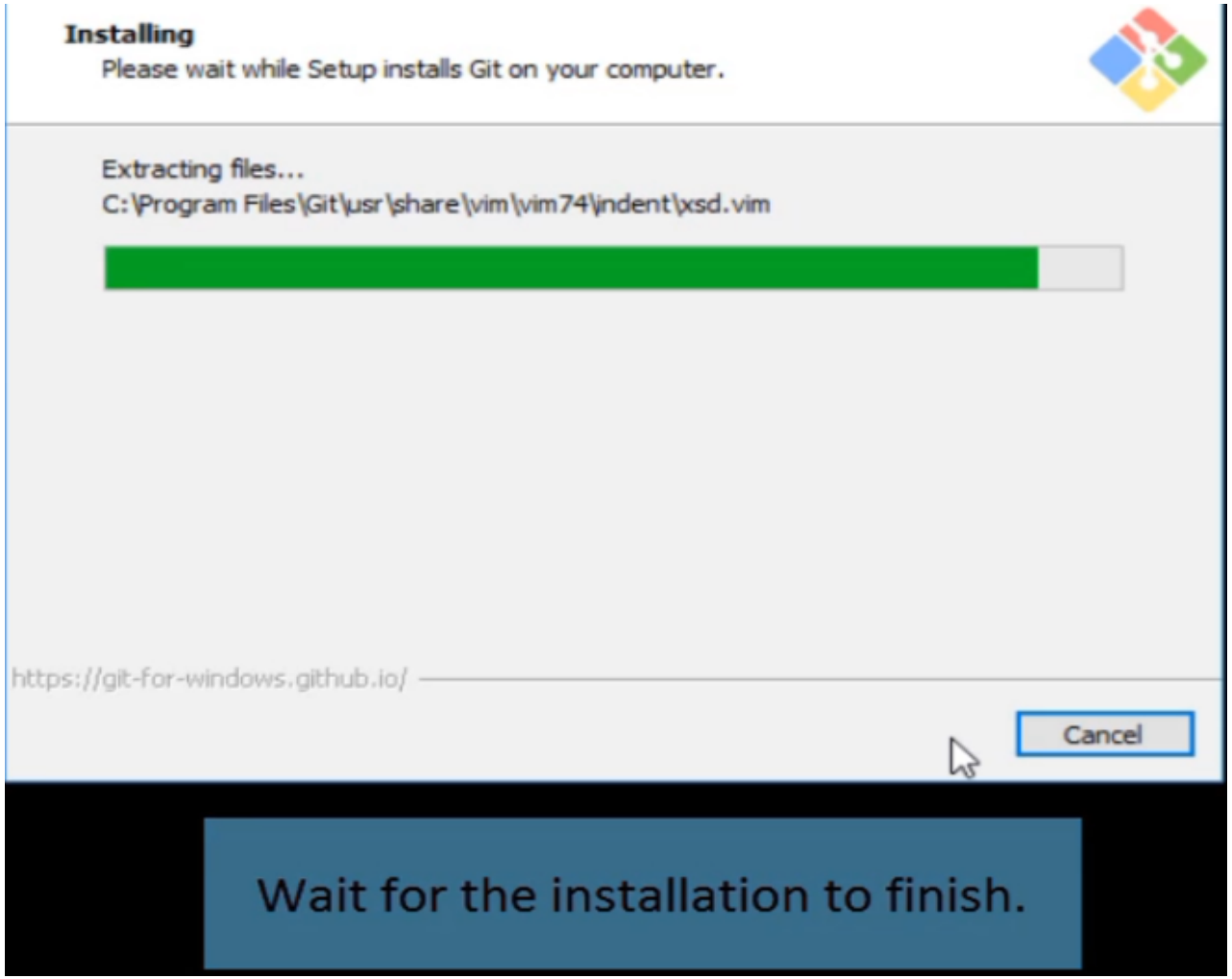
Step : 7


Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://git-for-windows.github.io/
< Back
Next >
Cancel

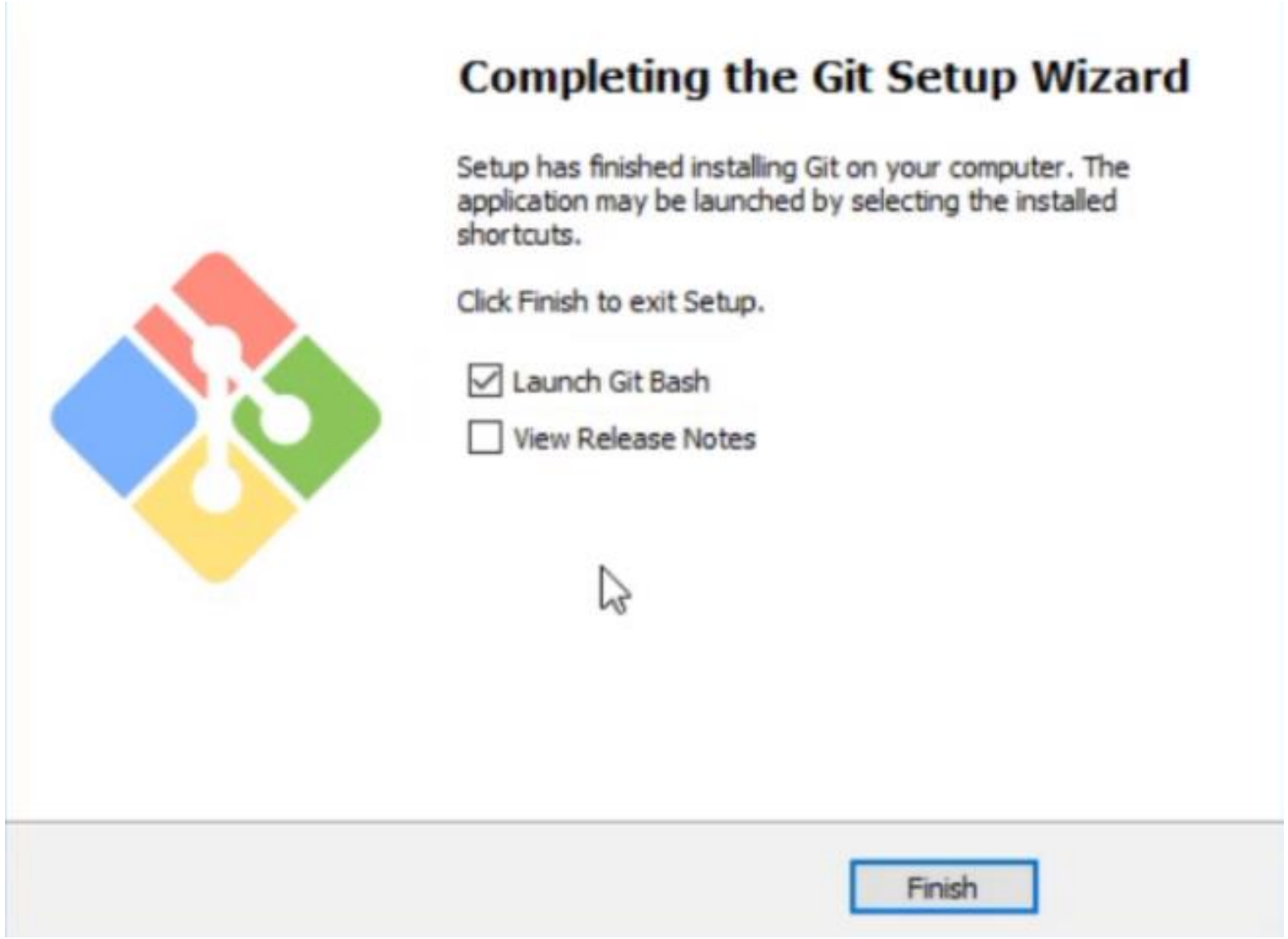
Step : 8


Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable Git Credential Manager
The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or or later)
https://git-for-windows.github.io/
< Back
Install
Cancel

Step : 9


Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files\Git\usr\share\vim\vim74\indent\xsd.vim
https://git-for-windows.github.io/
Cancel
Wait for the installation to finish.

Step : 10

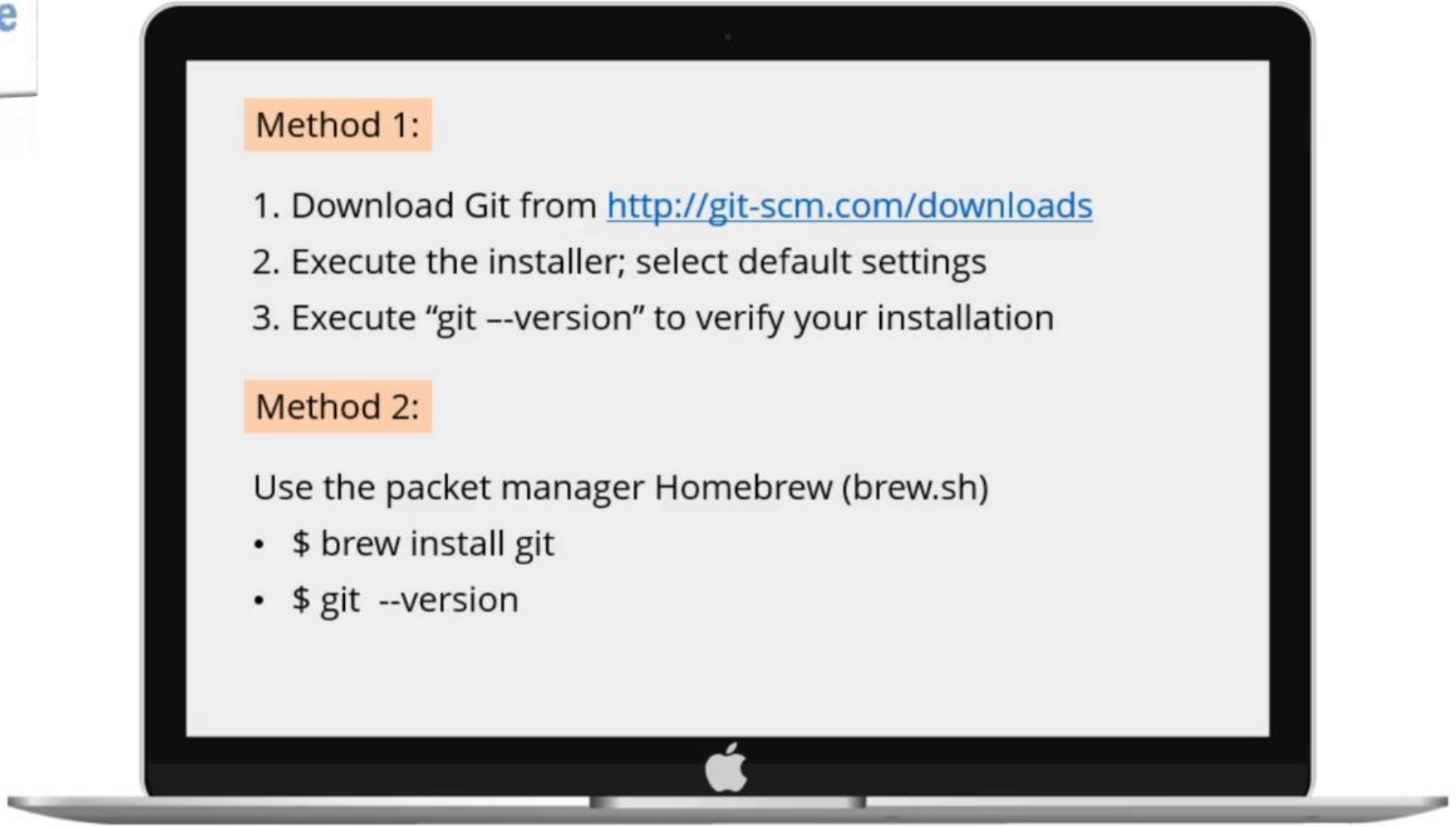

Completing the Git Setup Wizard
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Git Bash
View Release Notes
Finish

Step : 11

Note: You should be able to see git installed version in the console

- GIT Installation in the Mac

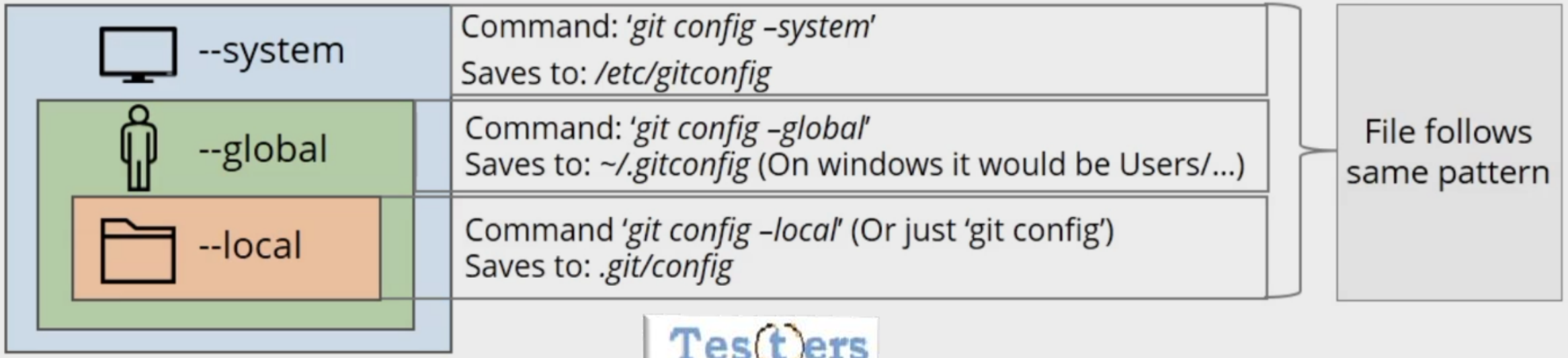

Testers
Zone


Method 1:
1. Download Git from http://git-scm.com/downloads
2. Execute the installer; select default settings
3. Execute “git –-version” to verify your installation
Method 2:
Use the packet manager Homebrew (brew.sh)
• $ brew install git
• $ git --version

Note

Git Configuration Levels

There are three configuration levels in Git.



NOTE

Local overrides Global and Global overrides System Level.

What is minimum Config to Configure Git?

To get the proper answer we can use below git command once.
git commit –m "<Commit message>"

If you are doing first time then git will indicate two missing configuration,
1. user.name
2. user.email

That needs to be defined at a global level.

e.g.
git config –global user.email "<email.id>"
git config –global user.name "<user name>"

Note: to know about all configured list we can use command:
git config --list

Steps to create a Git Repository

- Create an empty directory: **$ mkdir <directory-name>**
- Convert the directory into repository: **$ git init**
- We can clubbed these two steps in a single step: command: **$ git init <directory_name>**

Note:

git init command creates a hidden directory(.git) that stores all the metadata required for it to function.(observe in next slide)

```
(base) ➜  ~ git init TestersZone
Initialized empty Git repository in /Users/mithilesh.qap.con/TestersZone/.git/
```

How to check .git directory ??

```
(base) ➜  ~ cd TestersZone
(base) ➜  TestersZone git:(master) ls -lrt
(base) ➜  TestersZone git:(master) ls -la
total 0
drwxr-xr-x   3 mithilesh.qap.con  1312973762    96 Aug 15 17:26 .
drwxr-xr-x+ 64 mithilesh.qap.con  1312973762  2048 Aug 15 17:32 ..
drwxr-xr-x  10 mithilesh.qap.con  1312973762   320 Aug 15 17:26 .git
(base) ➜  TestersZone git:(master)
```

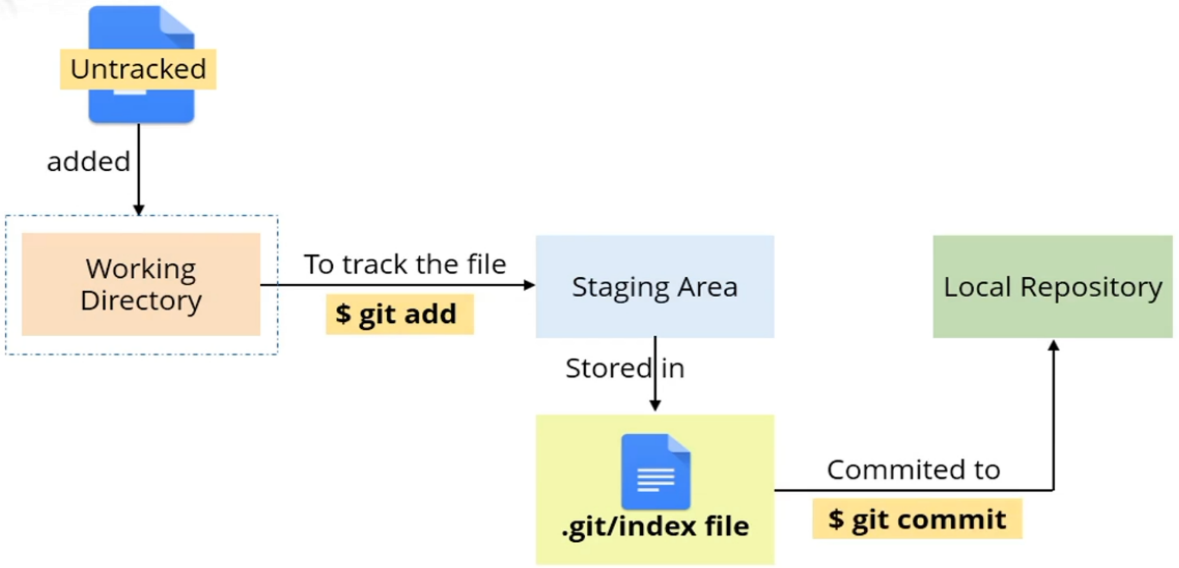
Main directory is empty

Checking hidden files

Default branch

Git's Three Stage Workflow

Git maintains three snapshots of a file in separate directories.



Working Directory is nothing but the area where we do code manupulation(write/delete/update) etc.

Note: working directory contains untracked files, means git does not track it. So if something goes wrong with your code, you will not be able to recover it.

Staging Area is tracked zone for the git once the changes have done by programmer in working directory they can add those in staging area using "git add" command.

$ git add <path of the file>.

Local Repository is the place where we can commit our all the changes with specific message, so that if anyone want to know about those changes they can easily get to know about changes.

$ git commit –m "<commit message>"

```
(base) ➜  TestersZone git:(master) touch demo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo.html

nothing added to commit but untracked files present (use "git add" to track)
(base) ➜  TestersZone git:(master) ✗ git add demo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   demo.html

(base) ➜  TestersZone git:(master) ✗ git commit -m "added demo.html"
[master (root-commit) 8fa2fd4] added demo.html
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo.html
(base) ➜  TestersZone git:(master) git status
On branch master
nothing to commit, working tree clean
(base) ➜  TestersZone git:(master)
```

Note: please follow the last slide's flow diagram to understand the git commands.

Step1: created one new file with the help of touch command. Since this file is created in working directory TestersZone it is untracked file.

Note: to verify it we can use git status command. It will be in red color

Step2: used git add command to convert it into trackable file(it is known as staging area).

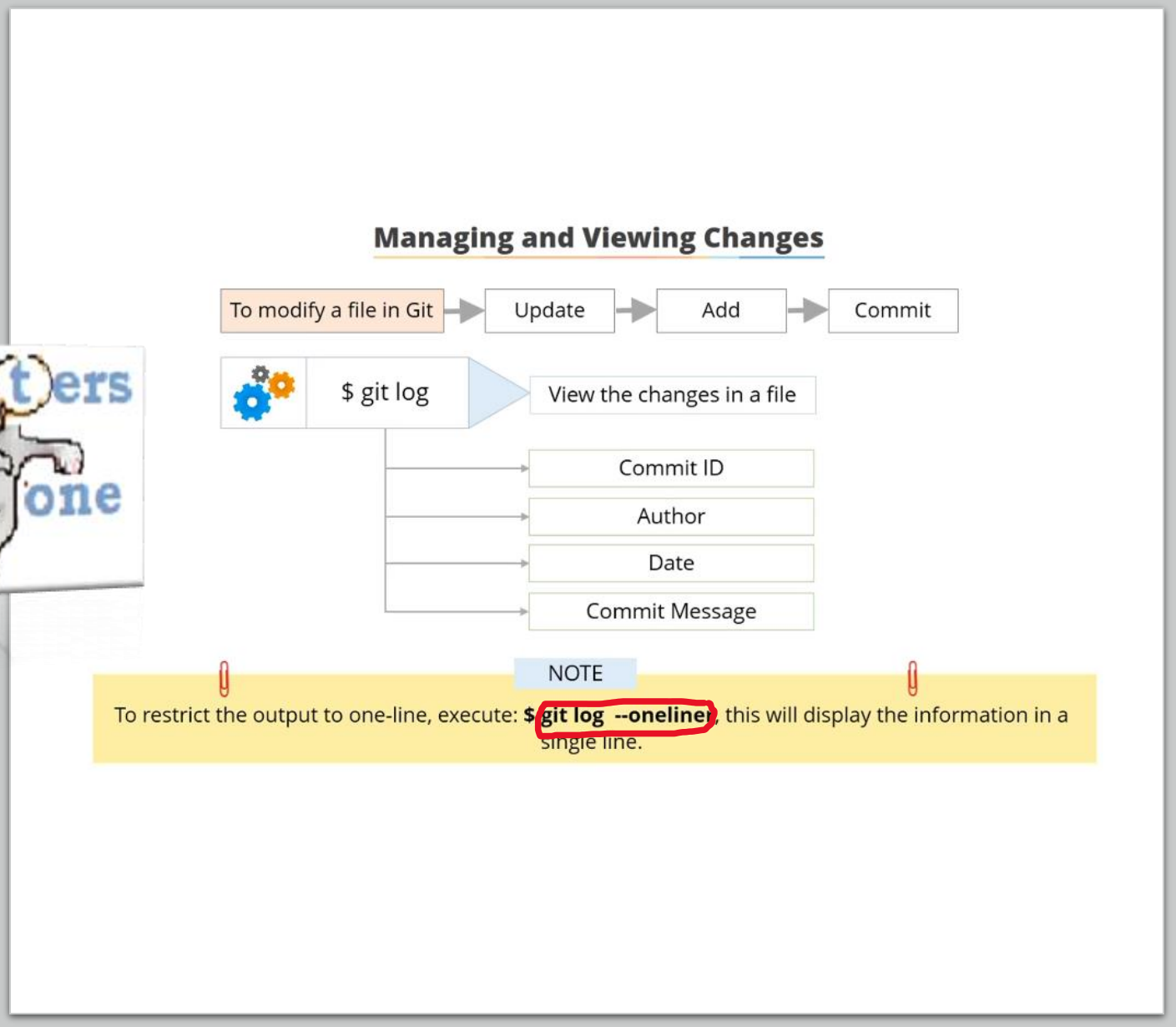
Note: file name will show in green color after moving to stage area.

Step3: to move the changes into local repository we can commit it with commit message.

Note: to verify it we can use git status command. We will get "nothing to commit, working tree clean"

What Information we get from git log?

- Add and commit files we had discussed in previous slide, so once we will do successful commit. Unique commit id will be generating with author name Date of commit and Commit message(as mentioned in the diagram) and that we can observe on console using this command
$ git log ---> see next slide for more clarity.



git log vs git log --oneline

```
(base) ➜  TestersZone git:(master) git log
commit 99b31ed67f680f1c97488c3f3d507f438db52212 (HEAD -> master)
Author: mithilesh.qap.con <mithilesh.qap.con@phonepe.com>
Date:   Sun Aug 15 22:17:20 2021 +0530

    modified demo.html file

commit 8fa2fd48a570931b9c191f0b9d51b8ab9f2bf736
Author: mithilesh.qap.con <mithilesh.qap.con@phonepe.com>
Date:   Sun Aug 15 21:54:40 2021 +0530

    added demo.html
(END)
```

```
(base) ➜  TestersZone git:(master) git log --oneline
99b31ed (HEAD -> master) modified demo.html file
8fa2fd4 added demo.html
(END)
```

In one liner log message you will be getting only commit message.

```
(base) ➜  TestersZone git:(master) ls -lrt
total 0
-rw-r--r--  1 mithilesh.qap.con  1312973762  0 Aug 15 21:52 demo.html
(base) ➜  TestersZone git:(master) echo "<html>   </html>" >> demo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.html

no changes added to commit (use "git add" and/or "git commit -a")
(base) ➜  TestersZone git:(master) ✗ git add demo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   demo.html

(base) ➜  TestersZone git:(master) ✗ git commit -m "modified demo.html file"
[master 99b31ed] modified demo.html file
 1 file changed, 1 insertion(+)
(base) ➜  TestersZone git:(master) git status
On branch master
nothing to commit, working tree clean
```

Viewing and modifying the files(class) inside framework:

1. If we are modifying something inside the files which is already added into git repo that will come under git tracked file only.

Q?: How will I come to know where I have added new file or modified into existing file

Ans: Observe diag 1 and diag2

All New files, git will show in untracked file and there will not be modified tag with file name

Modified files

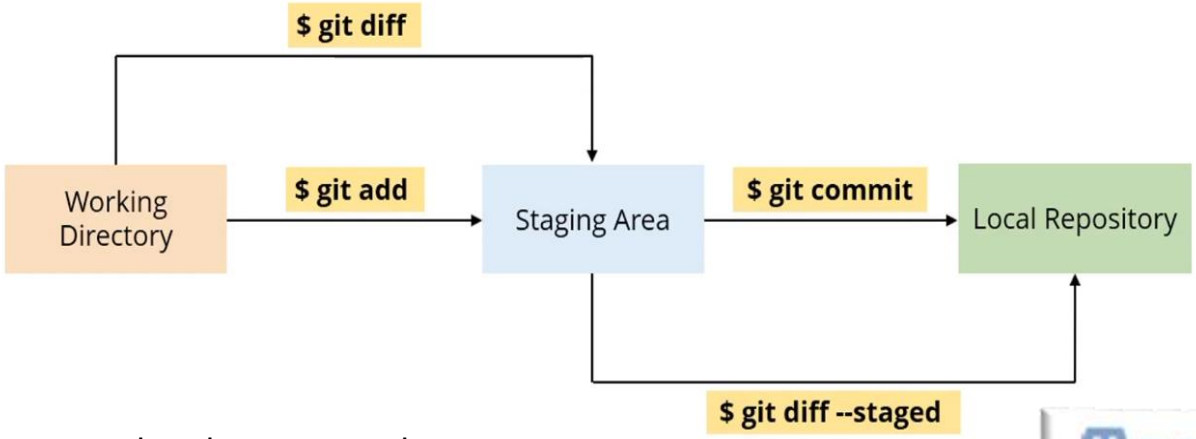
```
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.html
```

```
(base) ➜  TestersZone git:(master) ✗ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo.html
```

Can we track the changes inside the files in git?



What do you mean by tracking?

Tracking is the ability to identify changes between the different versions of the same file, spread across various repositories.


```
diff --git a/demo.html b/demo.html
index e69de29..12ea620 100644
--- a/demo.html
+++ b/demo.html
@@ -0,0 +1 @@
+<html>  </html>
(END)
```

+ sign indicates what actually changes are added in modified code, if something we are removing it will come with – sign

Note: in last slide we saw we have done some changes in the file and now it is under modified state.
It means we have one copy of file in our working directory(it is modified file) and one copy is in stage area(which we had added earlier and that is older version of file. That file does not have current changes. Changes will reflect when we will add changes.)

Conclusion: we have two diferent version of same file one is in working directory and other is in staging area.

Ques: I want to comapare changes between those two files?
Ans: **use git diff command.**

@@ -0, 0 +1 @@ --> left side of , tells about older file and right side of , tells about new file.
-0, --> - sign means old file and 0 means line starts will zero and end on same.
0 +1 --> it says line start with 0 and having total no of added lines count 1.
@@ is a part of format.

Key points:

1. If there is more than one files and we want to see the changes in particular file instead of "**git diff**" we can use "**git diff <file_name>**"

2. if we have already added our updated files to staging area then changes will not be reflecting using commands mentioned in point 1. in this case we say working directory is in synced with staging area.

3. **local repository**: once we commit the changes then only it will reflect into local repository.
4. If older changes are committed and latest changes are only added that means older changes are in local repo and latest changes are in staging area so we can differentiate the file versions available in stage and local repo using below command.
"**git diff --staged**".

5. If we have so many commits and want to differentiate the changes of any particular commit with updated file in staging then, we can use commit id like "**git diff <commit_id>"**. We can get commit id using "**git log"** command.(observe in next slide)

git diff vs git diff --staged

```
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.html

no changes added to commit (use "git add" and/or "git commit -a")
(base) ➜  TestersZone git:(master) ✗ git add .
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   demo.html

(base) ➜  TestersZone git:(master) ✗ git diff --staged
```

Steps:
1. Add the modified file to the git, now it is under staging area.
2. Now we have modified file in staging and older created file in local repo so if we want to diff between file's version we can either use commit id or HEAD with command git diff.

```
diff --git a/demo.html b/demo.html
index e69de29..12ea620 100644
--- a/demo.html
+++ b/demo.html
@@ -0,0 +1 @@
+<html>  </html>
(END)
```

Note:

If the number of commits increase beyond a certain count, use the **hashcode** command.	**$ git diff <commit hashcode>**

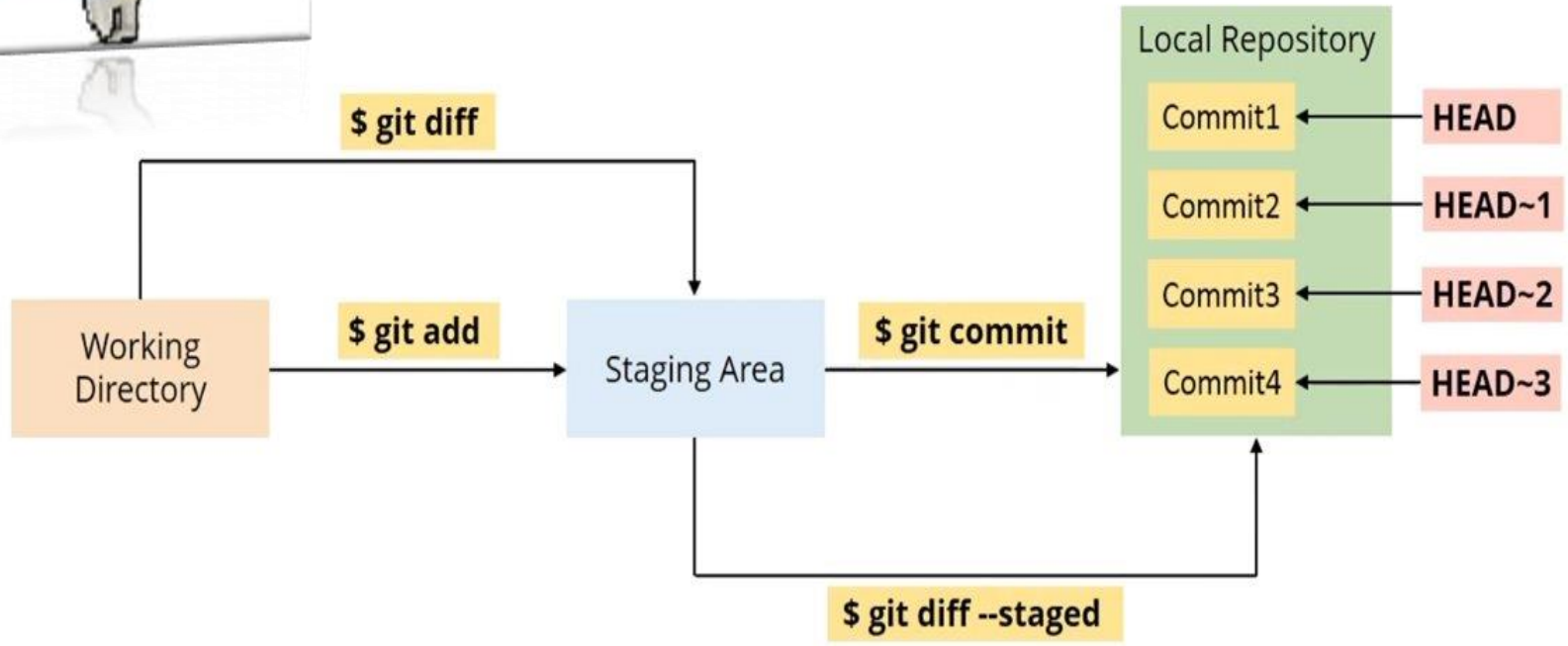
Command "**git diff --staged**" is similar to "**git diff HEAD**"

```
(base) ➜  TestersZone git:(master) ✗ git log --oneline
ee00303 (HEAD -> master) created empty file
e4e5fbd modified file in same format
99b31ed modified demo.html file
8fa2fd4 added demo.html
```

We can compare the code of staging area and local repo using these commit id's as well like

```
(base) ➜  TestersZone git:(master) ✗ git diff ee00303d2dbf64b5be87ae996fdd5f22e8884468
diff --git a/demo.html b/demo.html
index e69de29..12ea620 100644
--- a/demo.html
+++ b/demo.html
@@ -0,0 +1 @@
+<html>  </html>
(END)
```

Commit and HEAD both are same??




```
ee00303 (HEAD -> master) created empty file
e4e5fbd modified file in same format
99b31ed modified demo.html file
8fa2fd4 added demo.html
```

Commit id's is only we denote as HEAD as you can see in the above snaps

Please follow the previous slides. All slides are interrelated. Direct jumping to any slide will not be giving any idea

How can we revert the earlier commit?

Steps:

$ git log	List details of the commit associated with a file
$ git checkout	Stage the file
$ git status	Share the command to unstage the file
$ git reset	Remove the file from the staging area

1. cat <file_name> shows the contet of the file
2. It helps to verify the changes in the file.
3. In screeshot you can observe I have used cat demo.html twice to verify the content in two different versions of same file.
4. We can use HEAD or commit id both will work same. [see previous side for more clarity]
5. After checking out to any version of file we can use git status to get the command of unstaging the file. Or we can use git reset it will also un stage the stagged file.[observe pink text on the snap]

```
(base) ➜  TestersZone git:(master) cat demo.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta http-equiv="Content-Style-Type" content="text/css">
  <title></title>
  <meta name="Generator" content="Cocoa HTML Writer">
  <meta name="CocoaVersion" content="2022.44">
  <style type="text/css">
    p.p1 {margin: 0.0px 0.0px 0.0px 0.0px; font: 12.0px Helvetica}
  </style>
</head>
<body>
<p class="p1">“What is the purpose of today’s call, do you know?”</p>
</body>
</html>
(base) ➜  TestersZone git:(master) git checkout HEAD~1 demo.html
Updated 1 path from b863d21
(base) ➜  TestersZone git:(master) ✗ cat demo.html
<html>  </html>
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   demo.html

(base) ➜  TestersZone git:(master) ✗ git reset
Unstaged changes after reset:
M       demo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.html

no changes added to commit (use "git add" and/or "git commit -a")
(base) ➜  TestersZone git:(master) ✗
```

Yes it is possible using below commands

$ git rm <filename>	Deletes the file from staging area and working directory
Delete a file only from the staging area?	
$ git rm --cached <filename>	

```
(base) ➜  TestersZone git:(master) touch demo1.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo1.html

nothing added to commit but untracked files present (use "git add" to track)
(base) ➜  TestersZone git:(master) ✗ git add .
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   demo1.html
(base) ➜  TestersZone git:(master) git ls-files --stage
100644 12ea620de2b613468a3d0dc20959da8435d49e5c 0       demo.html
100644 e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 0       demo1.html
(base) ➜  TestersZone git:(master) git rm demo1.html
rm 'demo1.html'
(base) ➜  TestersZone git:(master) ✗ git ls-files --stage
100644 12ea620de2b613468a3d0dc20959da8435d49e5c 0       demo.html
(base) ➜  TestersZone git:(master) ✗ ls -lrt
total 8
-rw-r--r--  1 mithilesh.qap.con  1312973762  16 Aug 21 13:59 demo.html
(base) ➜  TestersZone git:(master) ✗ git rm --cached demo.html
rm 'demo.html'
(base) ➜  TestersZone git:(master) ✗ git ls-files --stage
(base) ➜  TestersZone git:(master) ✗ ls -lrt
total 8
-rw-r--r--  1 mithilesh.qap.con  1312973762  16 Aug 21 13:59 demo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    demo.html
        deleted:    demo1.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo.html

(base) ➜  TestersZone git:(master) ✗
```

Removing and restoring files in git

If you guys following the steps from the beginning, you will be awaring like we have demo.html file and now I am creating one more file demo1.html

Note:

1. To check the all stagging files we can use "**git ls-files --stage**"
2. To check the working directory files use "**ls -lrt**"

Steps followed:

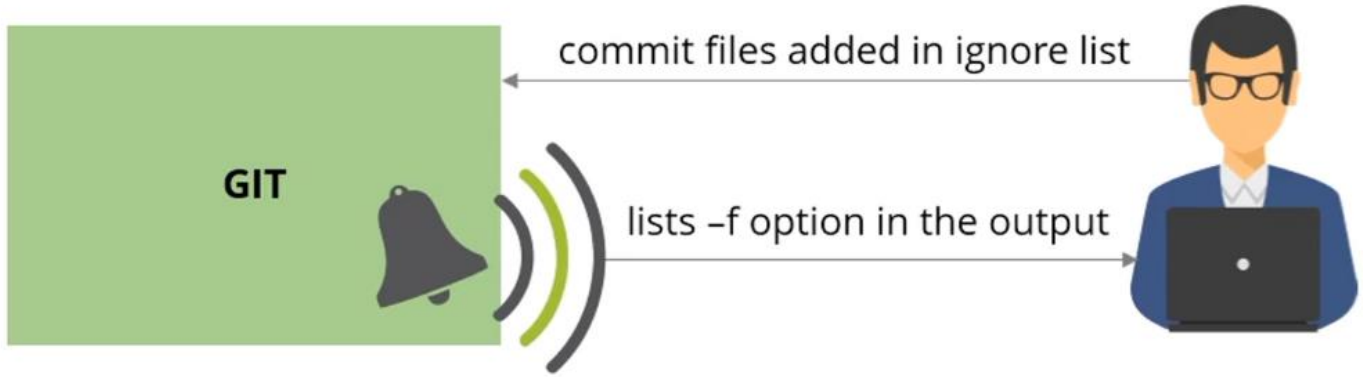
1. **git ls-files --stage** --> checked all the stagged files.
2. **git rm <file_name>** --> removed file from stage and working directory both.
3. **git ls-files --stage** --> verified deleted file from stagging.
4. **ls -lrt** --> verify deleted file from working directory.
5. **git rm --cached <file_name>** --> removed file from stagging only.
6. **git ls-files --stage** --> verify deleted file in stagging.
7. **ls -lrt** --> verify file is deleted only from stage not from working directory.
8. **git status** --> checked deleted files and untracked files.

You can restore the deleted files using version(commit id or file name)[observe pink line in screen shot]

e.g. **git checkout Head~1 <file_name>.**

How to ignore files in a git?

The file name and file pattern can be overridden using the **–f flag**



How to rename files in Git?

Steps:

$ git mv <filename> <new-filename>

$ git add	$ git commit

$ git status

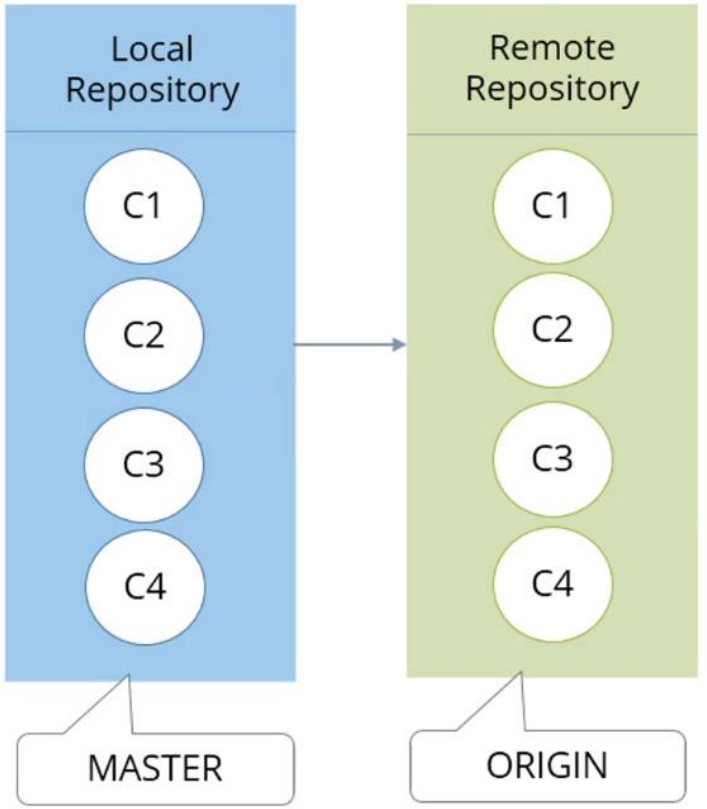

```
(base) ➜  TestersZone git:(master) git ls-files --stage
100644 12ea620de2b613468a3d0dc20959da8435d49e5c 0       demo.html
100644 e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 0       demo1.html
(base) ➜  TestersZone git:(master) ls -lrt
total 8
-rw-r--r--  1 mithilesh.qap.con  1312973762  16 Aug 21 13:59 demo.html
-rw-r--r--  1 mithilesh.qap.con  1312973762   0 Aug 21 22:21 demo1.html
(base) ➜  TestersZone git:(master) git mv demo.html newDemo.html
(base) ➜  TestersZone git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    demo.html -> newDemo.html

(base) ➜  TestersZone git:(master) ✗ git add .
(base) ➜  TestersZone git:(master) ✗ git commit -m "update the name of demo.html file"
[master 6bb2f16] update the name of demo.html file
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename demo.html => newDemo.html (100%)
(base) ➜  TestersZone git:(master) git status
On branch master
nothing to commit, working tree clean
(base) ➜  TestersZone git:(master) git ls-files --stage
100644 e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 0       demo1.html
100644 12ea620de2b613468a3d0dc20959da8435d49e5c 0       newDemo.html
(base) ➜  TestersZone git:(master)
```


Testers
Zone
GIT HUB

Introduction of Git Hub

- It provides the web based interface which helps to copy the entire commit history from the local repository to the remote repository.
- Local repository refers as master.
- Remote repository refers as origin.
- Git hub provides all the functionlaity of distribution control version tool(git) with some additional feature for publishing the project and collaboration.



How to create repository in Git hub?

Using https

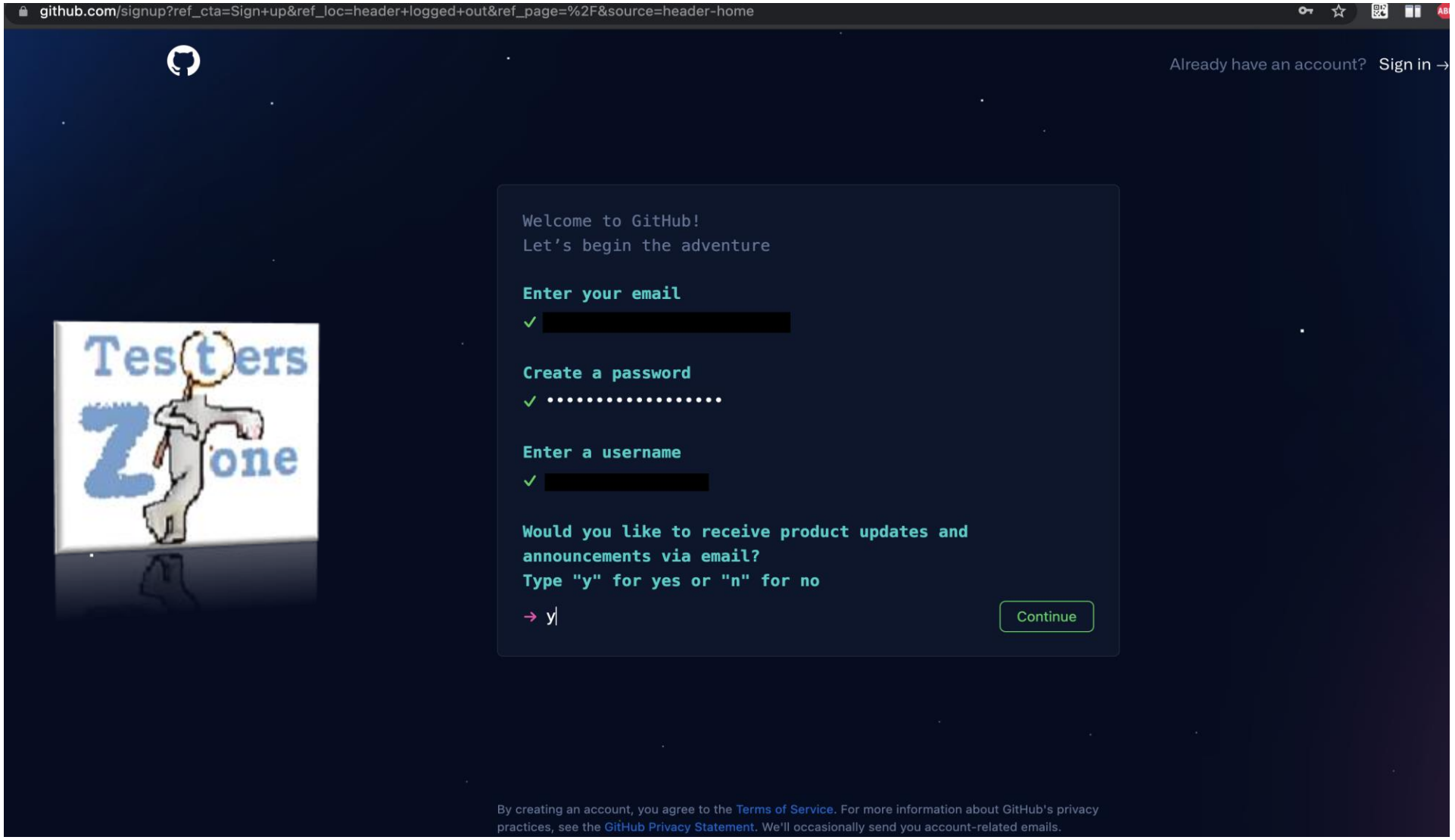
1. Go to GitHub and login
2. Create a new repository
3. Follow GitHub instructions
   - Create a symbolic link to GitHub
   - Push from local repository to GitHub

Using ssh

1. Ensure the local repository is ready
2. Create an SSH key (ssh-keygen)
3. Configure GitHub with SSH Public Key
4. Create a GitHub Repository
5. Push from local repository to GitHub

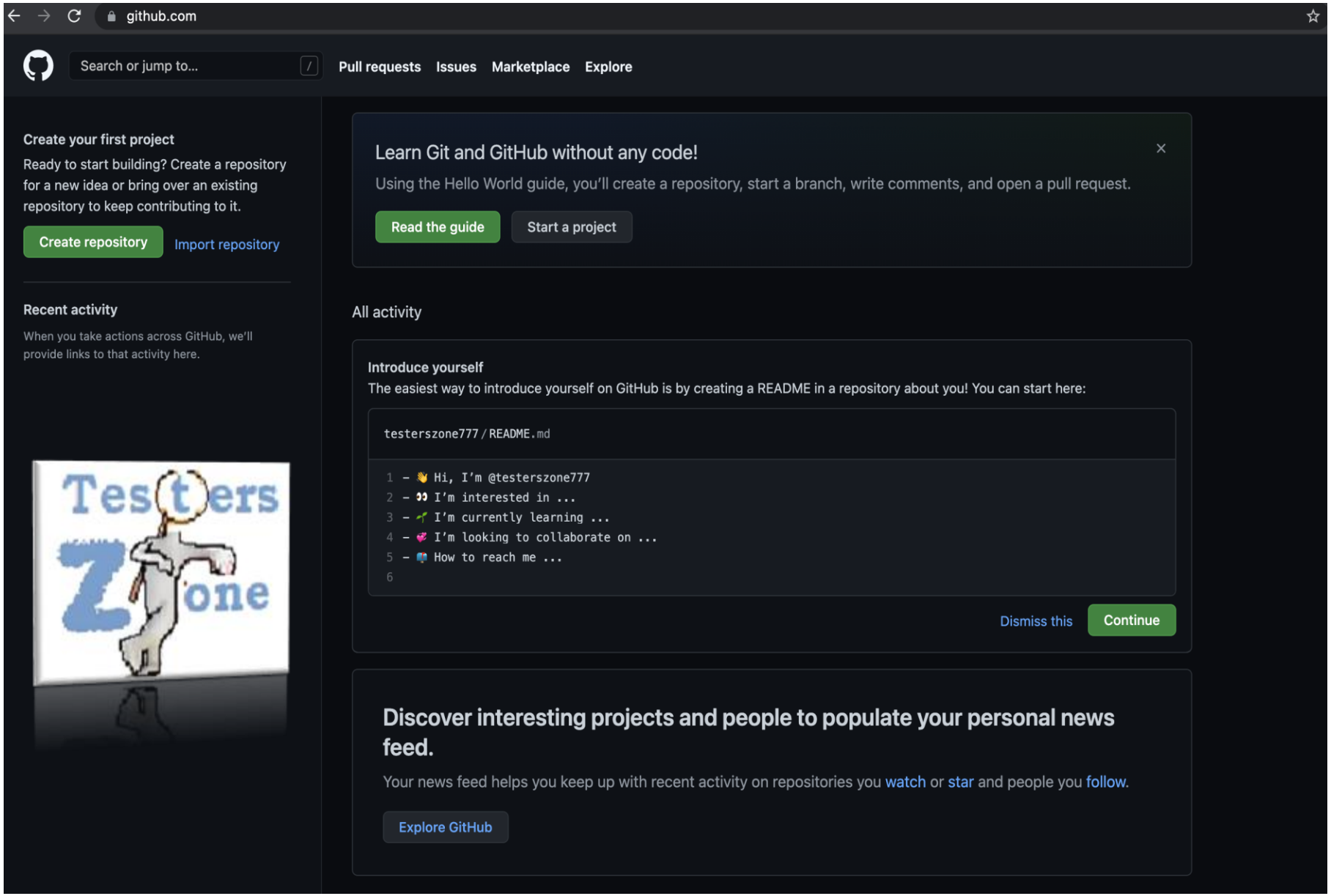
The SSH key helps you create a repository without a username and password.

Using https



Steps:

1. Access the github.com url and click on sign up option.
2. Fill all the details shown in the screenshot and create an account.



Once you will logged in you can see this window and then you can click on Create Repository to create one new repository.

github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * testerszone777 / **Repository name** * demo

Great repository names are short and memorable. Need inspiration? How about probable-adventure?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:
Skip this step if you're importing an existing repository.

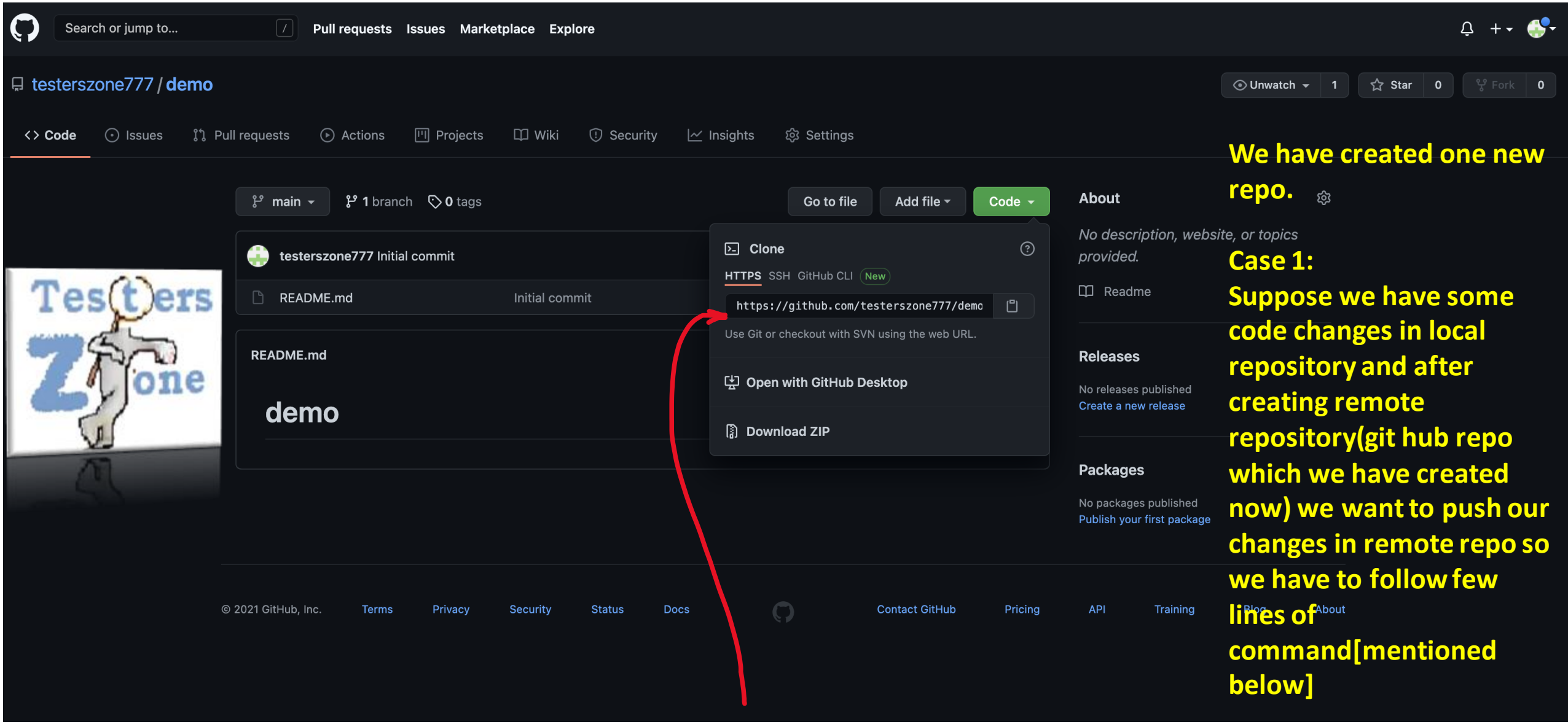
- [x] **Add a README file**
This is where you can write a long description for your project. Learn more.
- [ ] **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.
- [ ] **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

This will set `main` as the default branch. Change the default name in your settings.

Create repository

Soon you will click on create repository you will get this page.

1. Provide repository name.
2. Select type of repository either public or private. Public repo can be accessed by any one, for private repo you have to give the access.
3. Click on Create repository



We have created one new repo.

Case 1:
Suppose we have some code changes in local repository and after creating remote repository(git hub repo which we have created now) we want to push our changes in remote repo so we have to follow few lines of command[mentioned below]

```
(base) → TestersZone git:(master) git remote add origin https://github.com/testerszone777/demo.git
(base) → TestersZone git:(master) git push origin master
```

Perform these two lines of command one by one in the terminal. You will be asked to enter your credential. [observe the steps in next slide]

Note: repo link we can get from remote repo, exactly from where? You can observe in the above screenshot

```
(base) ➜  TestersZone git:(master) git status
On branch master
nothing to commit, working tree clean
(base) ➜  TestersZone git:(master) ls -lrt
total 8
-rw-r--r--  1 mithilesh.qap.con  1312973762  16 Aug 21 13:59 newDemo.html
-rw-r--r--  1 mithilesh.qap.con  1312973762   0 Aug 21 22:21 demo1.html
(base) ➜  TestersZone git:(master) git remote add origin https://github.com/testerszone777/demo.git
(base) ➜  TestersZone git:(master) git push origin master
Username for 'https://github.com': testerszone777
Password for 'https://testerszone777@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more inform
ation.
fatal: Authentication failed for 'https://github.com/testerszone777/demo.git/'
(base) ➜  TestersZone git:(master) git push origin master
Username for 'https://github.com': testerszone
Password for 'https://testerszone@github.com':
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (21/21), 2.12 KiB | 435.00 KiB/s, done.
Total 21 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/testerszone777/demo/pull/new/master
remote:
To https://github.com/testerszone777/demo.git
 * [new branch]      master -> master
(base) ➜  TestersZone git:(master)
```

How to generate token

If you use password here

Use generated token here

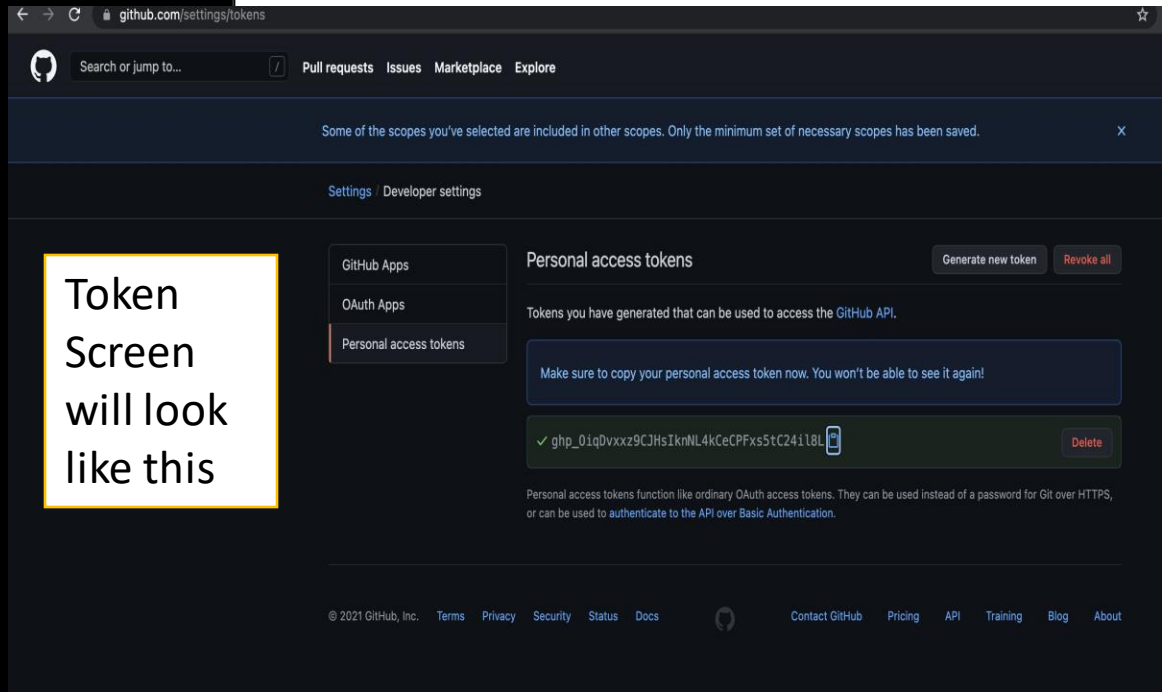
Create Personal Access Token on Github

From your Github account, go to **Settings** => **Developer Settings** => **Personal Access Token** => **Generate New Token** (Give your password) => **Fillup the form** => click **Generate token** => **Copy the generated Token**, it will be something like

`ghp_sFhFsSHhTzMDreGRLjmks4Tzuzgthdvfsrta`

More detail: https://stackoverflow.com/questions/68775869/support-for-password-authentication-was-removed-please-use-a-personal-access-to

Token Screen will look like this

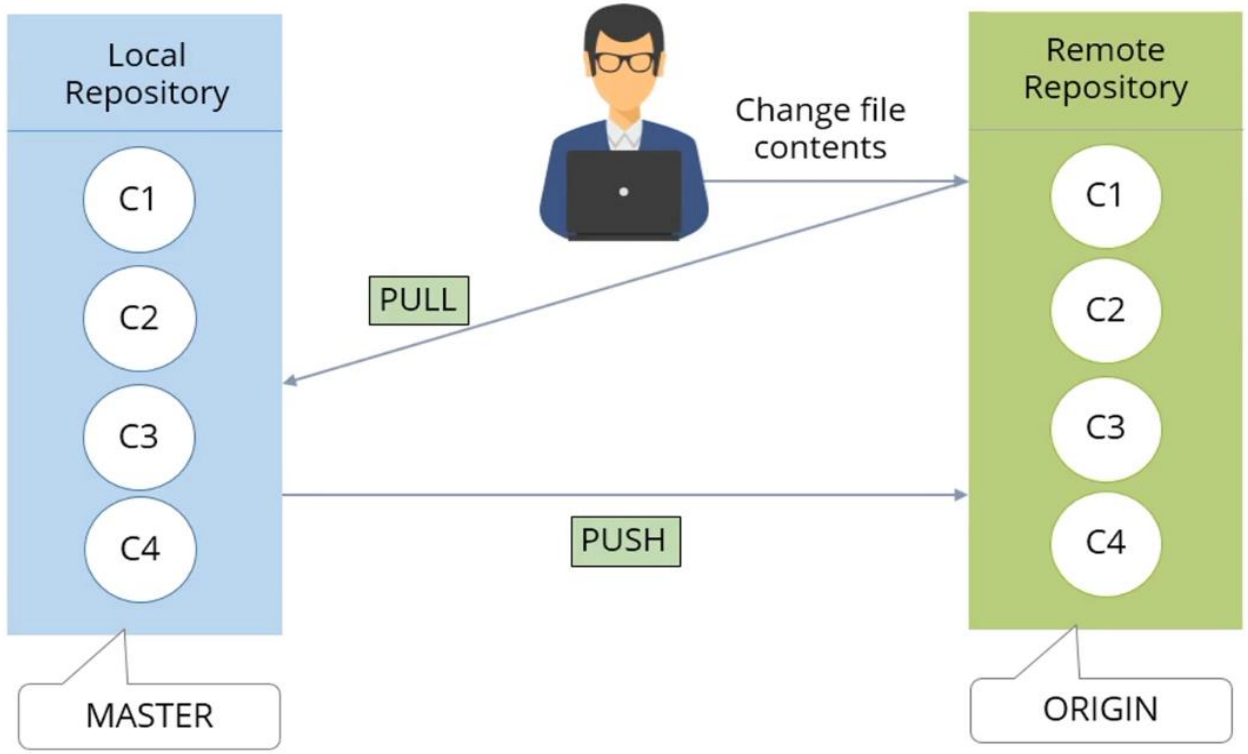


Pulling Changes from the origin repository?

- In real time scenario what happens like so many members will be working in the project and pushing their codes in the remote repository. So to get those changes in your local repo or to sync the local repository to the remote (origin) repository we need to pull the latest changes from remote repo always(look at diagram).

  e.g.
  suppose there is one file in the remote repository named demo1.html. And this file had few content. Someone made some changes in this file and push the changes to the git remote repo. Now in this case i will have older version of demo1.html file if I need new version of the same file we need to pull the changes from the remote repository

  Observe same scenario in the next 2 slides.



Search or jump to... / Pull requests Issues Marketplace Explore

testerszone777 / demo

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master demo / demo1.html Go to file ...

[redacted] added one line text Latest commit dfc3fa6 8 minutes ago History

0 contributors

Edit this file

16 lines (16 sloc) | 523 Bytes Raw Blame

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2  <html>
3  <head>
4    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5    <meta http-equiv="Content-Style-Type" content="text/css">
6    <title></title>
7    <meta name="Generator" content="Cocoa HTML Writer">
8    <meta name="CocoaVersion" content="2022.44">
9    <style type="text/css">
10     p.p1 {margin: 0.0px 0.0px 0.0px 0.0px; font: 12.0px Helvetica}
11   </style>
12 </head>
13 <body>
14 <p class="p1">“I am good”</p>
15 </body>
16 </html>
```

1 line

Older Version

testerszone777 / demo

Unwatch 1 | Star 0 | Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | demo / demo1.html | Go to file | ...

testerszone777 added one more line — Latest commit 7082f17 now — History

1 contributor

17 lines (17 sloc) | 565 Bytes — Raw | Blame

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2  <html>
3  <head>
4    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5    <meta http-equiv="Content-Style-Type" content="text/css">
6    <title></title>
7    <meta name="Generator" content="Cocoa HTML Writer">
8    <meta name="CocoaVersion" content="2022.44">
9    <style type="text/css">
10     p.p1 {margin: 0.0px 0.0px 0.0px 0.0px; font: 12.0px Helvetica}
11   </style>
12 </head>
13 <body>
14 <p class="p1">“I am good”</p>
15 <p class="p1">“I am really good?”</p>
16 </body>
17 </html>
```

2 lines

New Version

Follow the command mentioned in the screenshot to get the updated code from remote repo to local repo.

```
(base) ➜  TestersZone git:(master) git status
On branch master
nothing to commit, working tree clean
(base) ➜  TestersZone git:(master) git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/testerszone777/demo
 * branch            master     -> FETCH_HEAD
   dfc3fa6..7082f17  master     -> origin/master
Updating dfc3fa6..7082f17
Fast-forward
 demo1.html | 1 +
 1 file changed, 1 insertion(+)
(base) ➜  TestersZone git:(master)
```

Total new inserted lines of code.

Using ssh

To login with ssh key we need to create public key and add it in the repository. Follow the below steps.

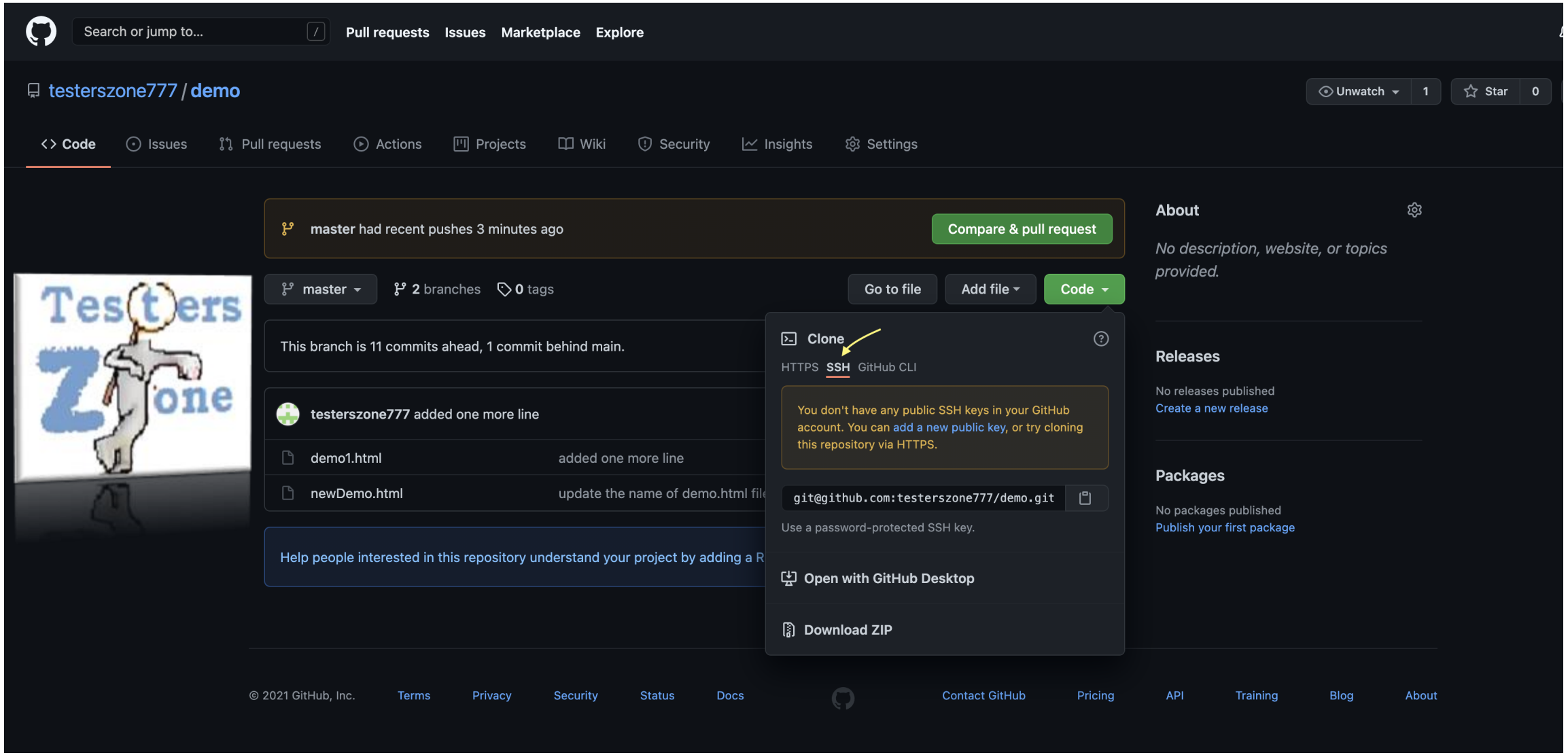
```
(base) ➜  TestersZone git:(master) ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/mithilesh.qap.con/.ssh/id_rsa):
/Users/mithilesh.qap.con/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/mithilesh.qap.con/.ssh/id_rsa.
Your public key has been saved in /Users/mithilesh.qap.con/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:IV50wR6J/RRw01rQqgv5tBTzGnaqbkfTvE2GfRj8TNk mithilesh.qap.con@PP-C02DFP2SMD6M.local
The key's randomart image is:
+---[RSA 3072]----+
|        .+++==   |
|       ...=...+  |
|      . o. + =  o|
|     . o .+ + o.E|
|      . S. B o * |
|        o O * = +|
|         B O = . |
|        . B . .  |
|       ooo       |
+----[SHA256]-----+
(base) ➜  TestersZone git:(master) cat /Users/mithilesh.qap.con/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCzvdSu4w4UaJlttY7mXjKwNUrXBcCMf9T/sgil7Fm57wQuJdSfRs89hAXoViLI07RN3hXun99bhDETmEz
vPshj9bMxnFX+tXAdmz+S7LsfoZGOAVJAIeEE82wu2x2s5AWllLg23Qp63aP5/tYDfDn+53HlRNxvGe2pFQ67llzo86XNa/EZoUAlCDnDWrf9BZntU9Jknz
yt0DoRWJZ4gWY3Ya52OgDsPDD4jp05vwfBjw7mcOc23PdLxVJllRa1Y2MC1vBOl/ySyEt017IEIJHiSDj3rPhItVMYXPCMEVJii5WhsZd+mEt+FgPxHd5m1
73CELh2NPYhqeaOy50Qnw52Ox+XMNAWPHQ+DzQOLo0xJlz0xuodit6gvFvFTpkyM8g8eHloAHyuhMHad88j3SNNrGrwtSJopHsfvocQbV0cg843B0jfNubL
ONe1ozc3e8IM1WD7utqrUZmshKaj6ZpTclUHf2jStg23t6WyFHuHxXkCGJwlTxPIGRV9HQ6pNas= mithilesh.qap.con@PP-C02DFP2SMD6M.local
(base) ➜  TestersZone git:(master)
```

Public key location

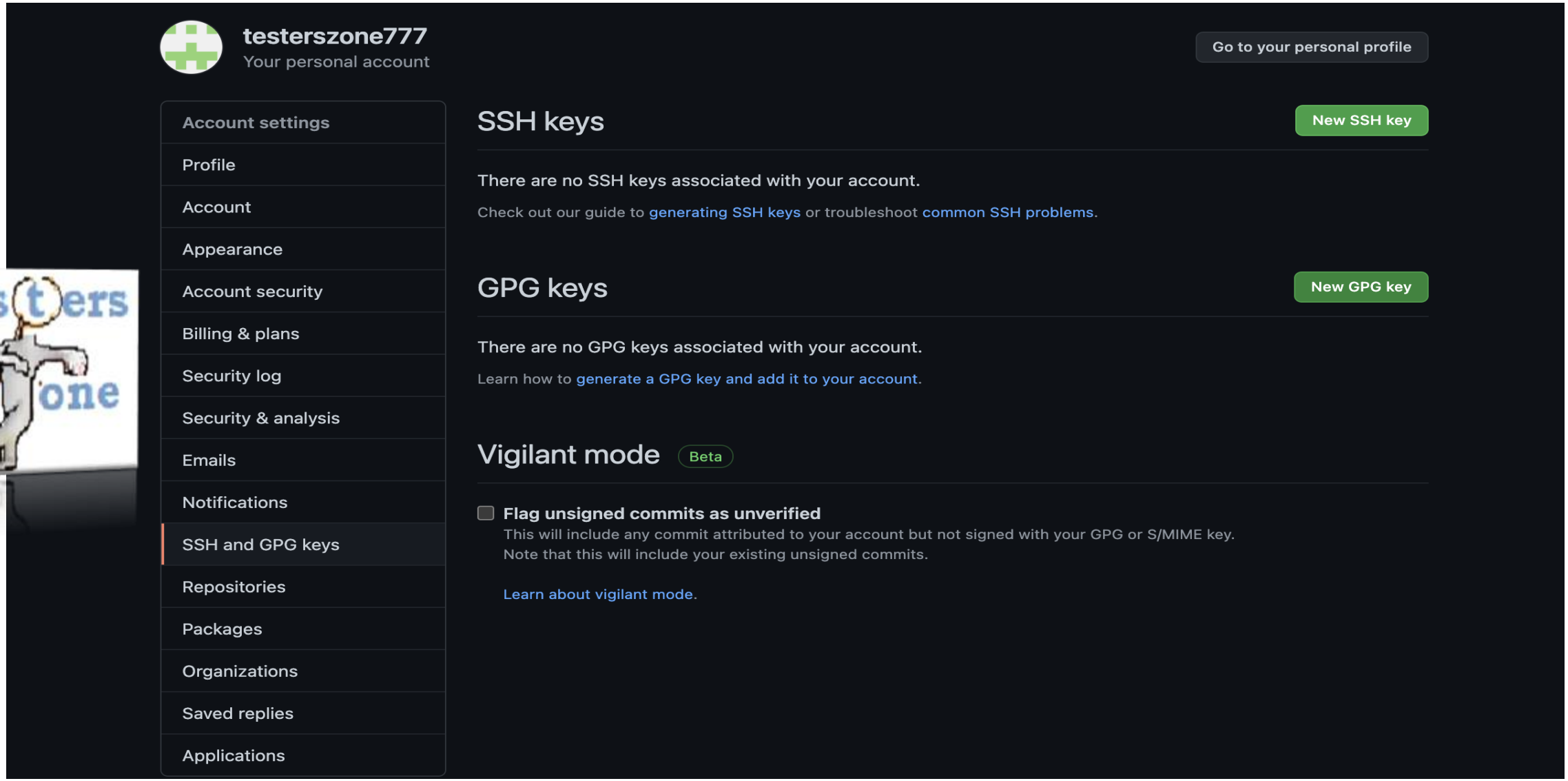
It shows key has been generated

Ssh public key

If you have not added any ssh key in your remote repo it will be showing message "you don't have any SSH keys". You can "add new public key".

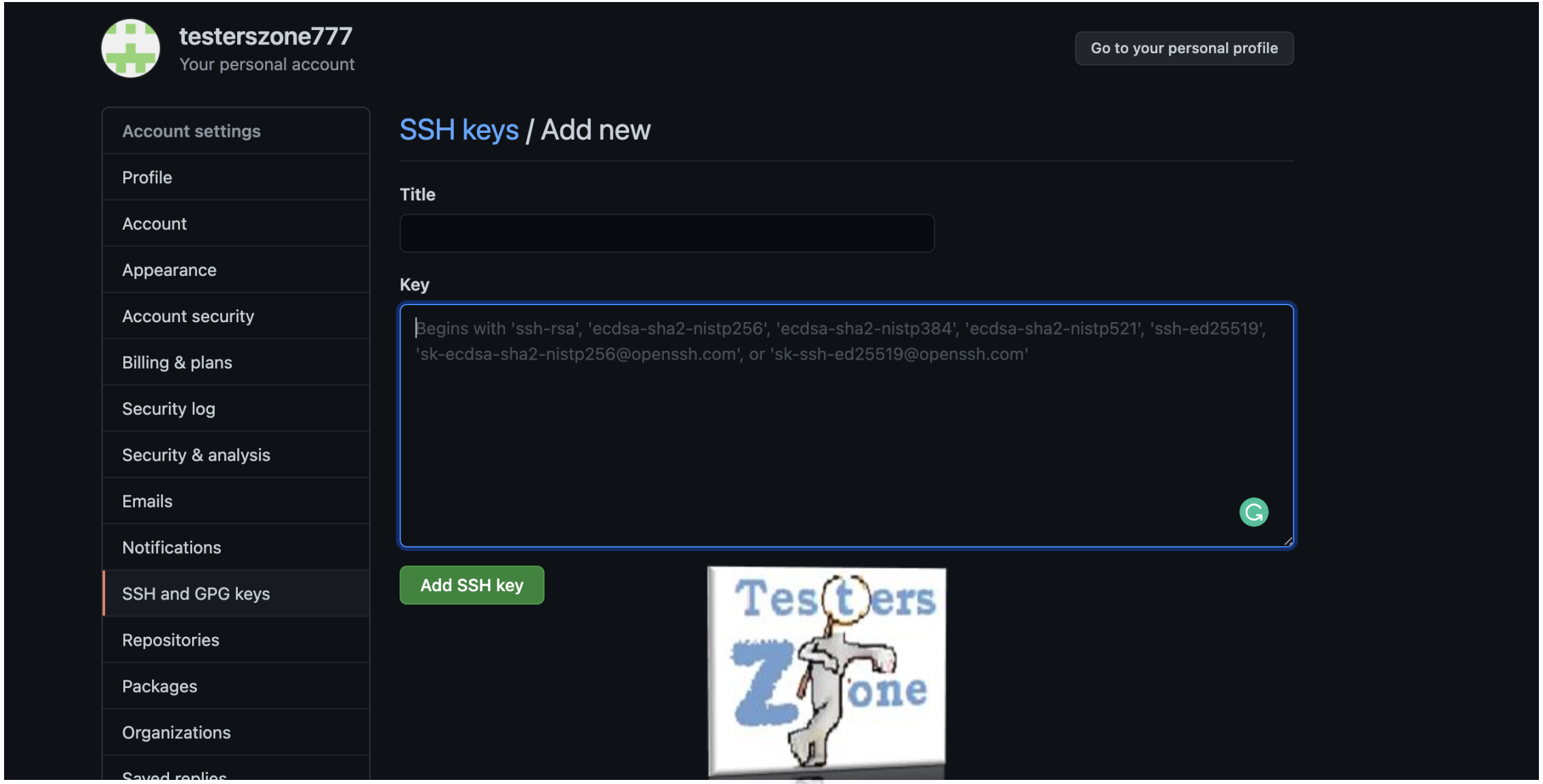


Go to repository setting--> SSH and GPG key--> New SSH key



Copy your ssh key from the terminal and paste it under key section and click on add ssh key

Note: Other process of adding local changes to remote repo will be same which we discussed in https case.
One minor change is like instead of https repo link use ssh repo link(copy it from remote repo, next to https)



Key Points:

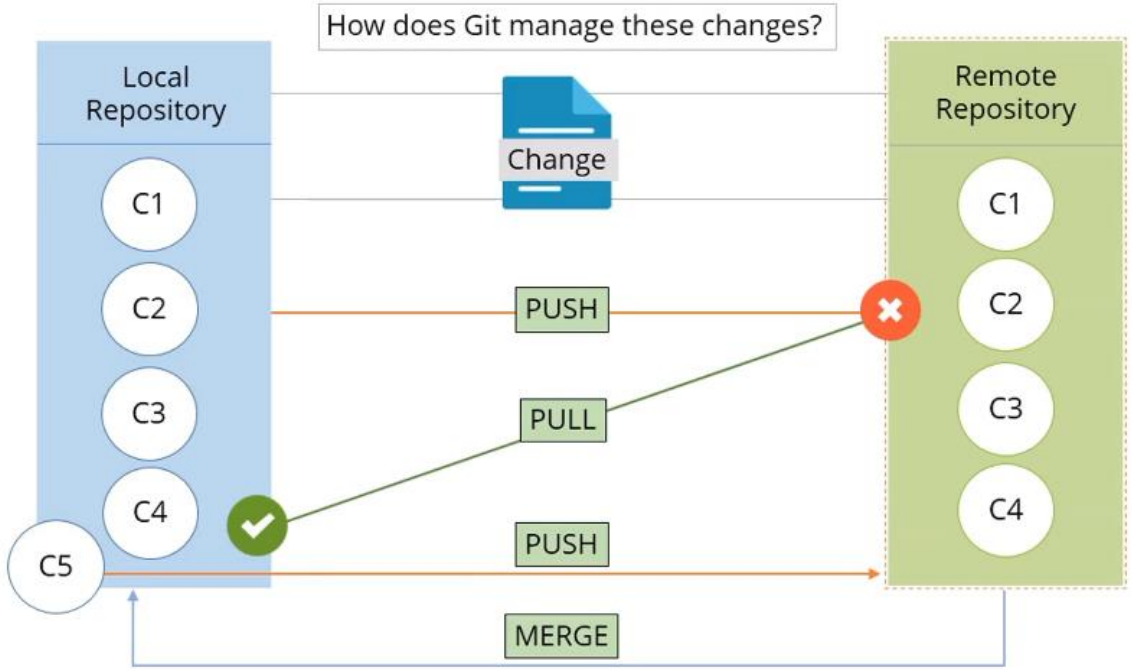
Collaboration between local and remote repository happens with the help of git pull and git push commands.
git pull command helps to get all the changes from remote repository to the local repository.
git push command needs to push all the local changes to the remote repository.
In this way we can use these two commands to keep our local and remote repositories in sync.

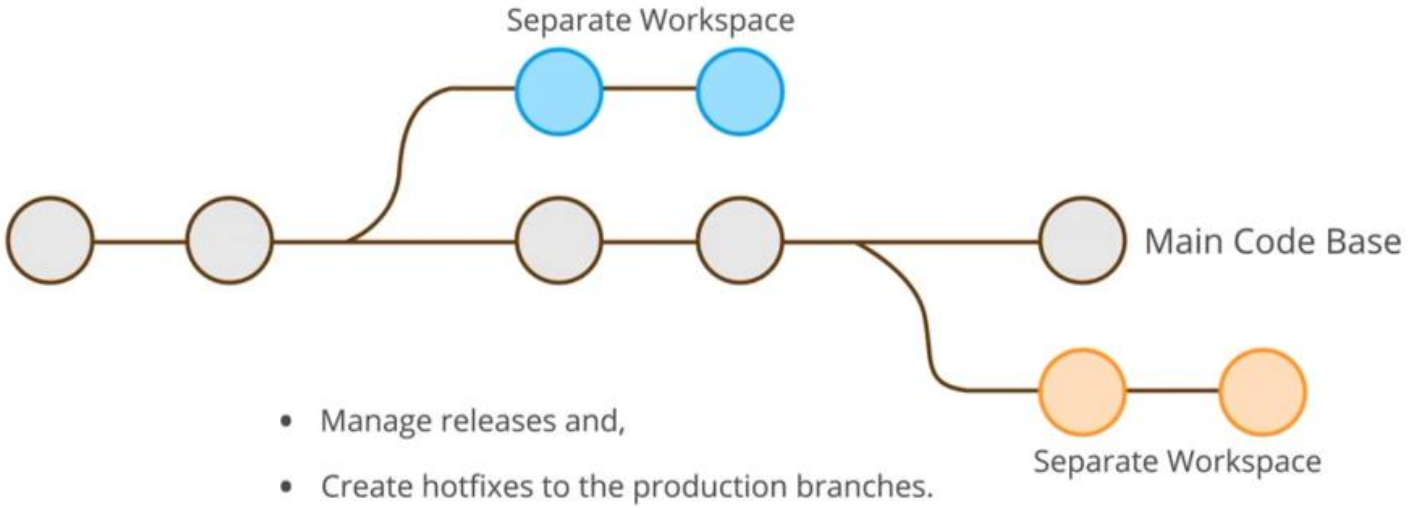
How to deal with multiple commits in the git?

- **Scenario**:

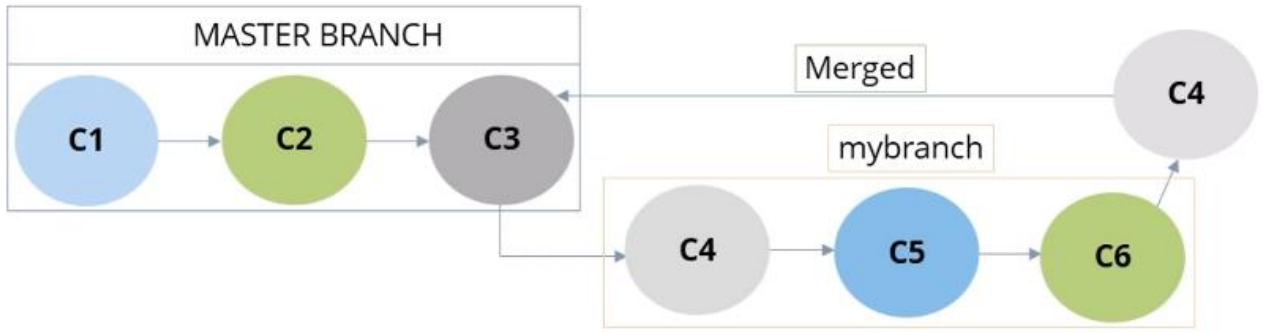
sometimes it might be the case two people are doing changes in the same file at their working directory and one of them pushed his/her changes to the repo. After that if you are trying to push you changes of the same file to the repo you will not be able to do so.

To overcome this we have to take pull from remote repo so all the changes will come to the local repo now we have to do force commit and then git push to push all your changes. In this way you can get other's update in our local repo also.





- **What is Branch concept in git?**



A branch is a method of requesting a new working directory and staging area.

When a new branch is created, the complete commit history of the master branch is replicated.

- **What Is need of the branch?**

Branch helps to keep our changes in our working directory till the time it is tested by programmer once they are done with the changes and want to share with the other members of the team they can merge their branches with the remote (origin) branch

Branch related commands

$ git branch <branch name>	Creates a new branch
$ git branch - a	Lists all branches in the current repository
$ git checkout <branch name>	Switch to branches
$ git merge <branch name>	Merges branches

NOTE

To create a new branch and switch to it, execute: **$ git checkout -b<branch-name>**

```
(base) ➜  TestersZone git:(master) git branch
* master
(base) ➜  TestersZone git:(master) git branch -a
* master
  remotes/origin/main
  remotes/origin/master
(base) ➜  TestersZone git:(master) git log --oneline
7082f17 (HEAD -> master, origin/master) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
(base) ➜  TestersZone git:(master) git branch demoBranch
(base) ➜  TestersZone git:(master) git branch -a
  demoBranch
* master
  remotes/origin/main
  remotes/origin/master
(base) ➜  TestersZone git:(master) git checkout demoBranch
Switched to branch 'demoBranch'
(base) ➜  TestersZone git:(demoBranch) git branch
* demoBranch
(base) ➜  TestersZone git:(master) git log --oneline
7082f17 (HEAD -> demoBranch, origin/master, master) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
(base) ➜  TestersZone git:(demoBranch) git checkout -b demo1Branch
Switched to a new branch 'demo1Branch'
(base) ➜  TestersZone git:(demo1Branch) git branch
* demo1Branch
  demoBranch
  master
```

It will show Current branch name.

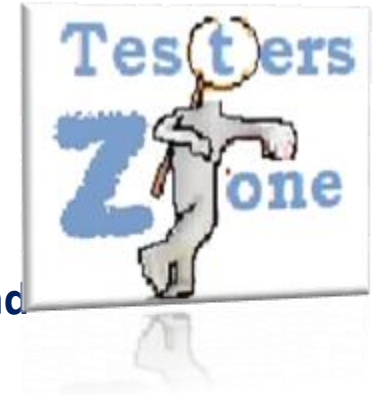
It will show all the branch name. * will represent current branch

It is representing all the commits performed in the master branch

Create a new branch with this command

Verifying newly created branch

Move to specific branch using this command



Verify the landed branch

Verify all the master branch commits are there in local branch also

This single command can create and checkout to the new branch

Verifying above command

Note: Our local changes(commits) will not be impecting master branch until code is not merged.

```
base) ➜  TestersZone git:(demo1Branch) git branch
* demo1Branch
  demoBranch
  master
(base) ➜  TestersZone git:(demo1Branch) ls -lrt
total 16
-rw-r--r--  1 mithilesh.qap.con  1312973762   16 Aug 21 13:59 newDemo.html
-rw-r--r--  1 mithilesh.qap.con  1312973762  565 Aug 22 03:19 demo1.html
(base) ➜  TestersZone git:(demo1Branch) echo "Updated line" >> demo1.html
(base) ➜  TestersZone git:(demo1Branch) ✗ git status
On branch demo1Branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo1.html

no changes added to commit (use "git add" and/or "git commit -a")
(base) ➜  TestersZone git:(demo1Branch) ✗ git add .
(base) ➜  TestersZone git:(demo1Branch) ✗ git commit -m "updated file in local branch"
[demo1Branch 032c5ce] updated file in local branch
 1 file changed, 1 insertion(+)
(base) ➜  TestersZone git:(demo1Branch) git log --oneline

032c5ce (HEAD -> demo1Branch) updated file in local branch
7082f17 (origin/master, master, demoBranch) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
(base) ➜  TestersZone git:(demo1Branch) git checkout master
Switched to branch 'master'
(base) ➜  TestersZone git:(master) git log --oneline
7082f17 (HEAD -> master, origin/master, demoBranch) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
```

I have added one extra line in the demo1.html file and commit the changes. This changes will not be impacting master branch. To verify it just use git log –oneline command in demo1Branch and master branch.

In demo1Branch we can observe we have one extra commit which is not there in master branch.

it indicates our changes will be in local repo until we push it to the remote repository and merged.

How to merge branches in Git?

Switch to the branch you want to merge

Execute: **$ git merge <branch name>**

Steps to merge the Branches

- Check the difference between two branches using command
  git diff <first_branch_name>..<second_branch_name>
- It will show, if there is any content difference between these two branches.
  Note: if there are more changes in more than one files then all will be reflecting on console using above command.
- Other way to identify the changes are using git log command, it will show the extra commits in case of additional changes.
- Observe all these practically in next slide.

```
(base) ➜  TestersZone git:(demo1branch) git diff master..demo1branch
diff --git a/demo1.html b/demo1.html
index 412f905..6cc8a8d 100644
--- a/demo1.html
+++ b/demo1.html
@@ -15,3 +15,4 @@
 <p class="p1">“I am really good?”</p>
 </body>
 </html>
+Updated line
diff --git a/newDemo.html b/newDemo.html
index 12ea620..51b8cd0 100644
--- a/newDemo.html
+++ b/newDemo.html
@@ -1 +1,2 @@
 <html>  </html>
+My name is Mithilesh Singh
(base) ➜  TestersZone git:(demo1branch) git log --oneline
65979e4 (HEAD, demo1Branch) added new changes in newDemo.html file
032c5ce updated file in local branch
7082f17 (origin/master, master, demoBranch) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
5ca9252 removed from staging and added few things
f17859a reverted few changes
fa38c4c added one line text
312383d modified demo.html file with one liner test
ee00303 created empty file
e4e5fbd modified file in same format
99b31ed modified demo.html file
8fa2fd4 added demo.html
(base) ➜  TestersZone git:(master) git log --oneline
7082f17 (HEAD -> master, origin/master, demoBranch) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
5ca9252 removed from staging and added few things
f17859a reverted few changes
fa38c4c added one line text
312383d modified demo.html file with one liner test
ee00303 created empty file
e4e5fbd modified file in same format
99b31ed modified demo.html file
8fa2fd4 added demo.html
```

git merge command

1. If you will observe here we are using **git log --oneline** command for the **master** branch and **demo1branch**.

In **demo1branch** we have two more commit id's as compare to master branch. It shows we have more changes in local branch(**demo1branch**) compare to **master** branch.
If we want to look at exact content changes rather than commit message, we can use git **diff branch1..branch2** command.

Note: since we have extra contents in **demo1branch** we can merge that with the **master** branch.
Extra changes should me added with master branch so we have to checkout to the master branch first.

Then use **git merge <branch_name>** --> **branch_name** means branch whose code we need to add with master branch.

Once we merge the master branch with the demo1branch there will not be any difference between those two branches.[observe next]


```
(base) ➜  TestersZone git:(master) git diff master..demo1branch
(base) ➜  TestersZone git:(master) git merge demo1branch
Updating 7082f17..65979e4
Fast-forward
 demo1.html   | 1 +
 newDemo.html | 1 +
 2 files changed, 2 insertions(+)
```

1

After Merge??

```
(base) ➜  TestersZone git:(master) git log --oneline
65979e4 (HEAD -> master, demo1Branch) added new changes in newDemo.html file
032c5ce updated file in local branch
7082f17 (origin/master, demoBranch) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
5ca9252 removed from staging and added few things
f17859a reverted few changes
fa38c4c added one line text
312383d modified demo.html file with one liner test
ee00303 created empty file
e4e5fbd modified file in same format
99b31ed modified demo.html file
8fa2fd4 added demo.html
```

2

```
(base) ➜  TestersZone git:(demo1branch) git log --oneline
65979e4 (HEAD, master, demo1Branch) added new changes in newDemo.html file
032c5ce updated file in local branch
7082f17 (origin/master, demoBranch) added one more line
dfc3fa6 added one line text
6bb2f16 update the name of demo.html file
5ca9252 removed from staging and added few things
f17859a reverted few changes
fa38c4c added one line text
312383d modified demo.html file with one liner test
ee00303 created empty file
e4e5fbd modified file in same format
99b31ed modified demo.html file
8fa2fd4 added demo.html
```

3

In step 1 I merged master branch with demo1branch.
And now you can observe both the branches having same commit messages. Means all changes are available in both the branches so local branch and master branch both are in sync.

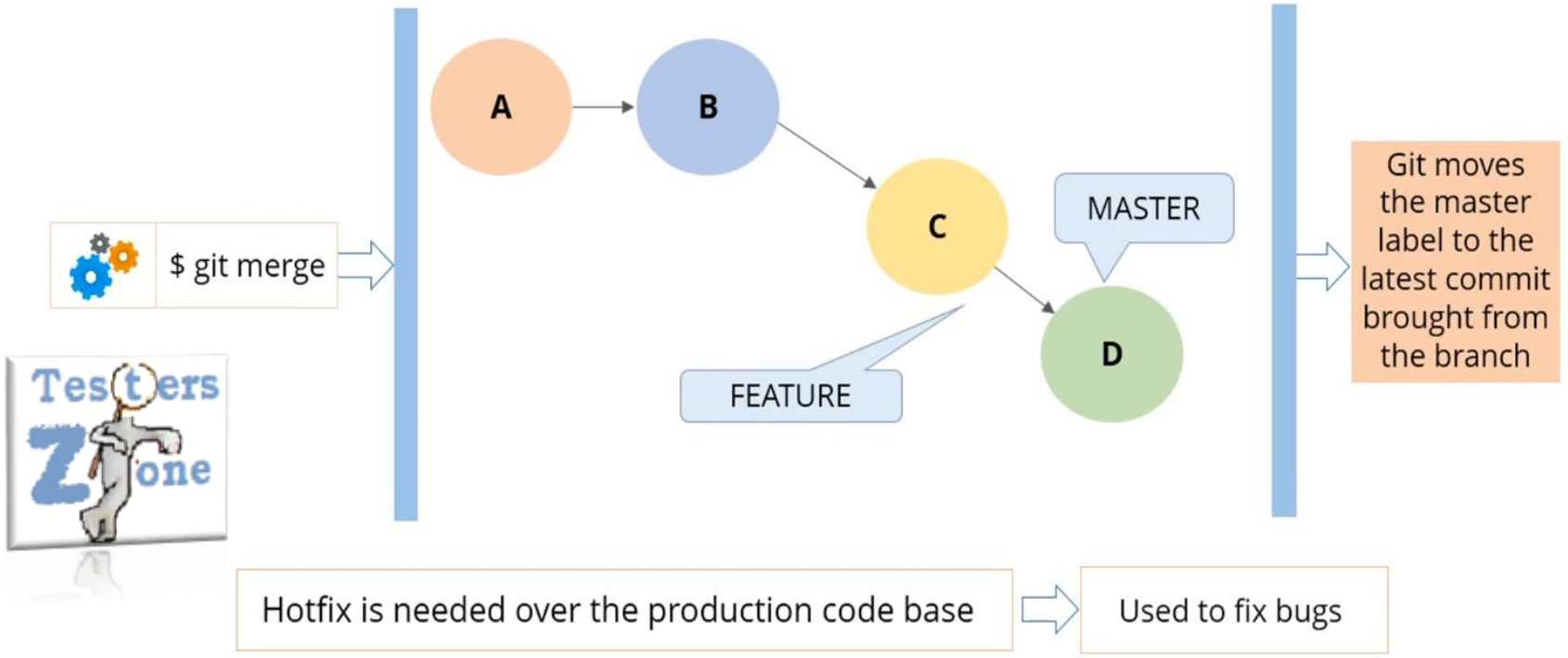
Note: In Real time projects we need to create PR(pull request) or MR(Merge request) against our changes to get it reviewed by lead or tech head before merging it with master. Once they approved you can pull all new changes from master.

TYPES OF GIT MERGE

Fast Forward Merge

Recursive merge

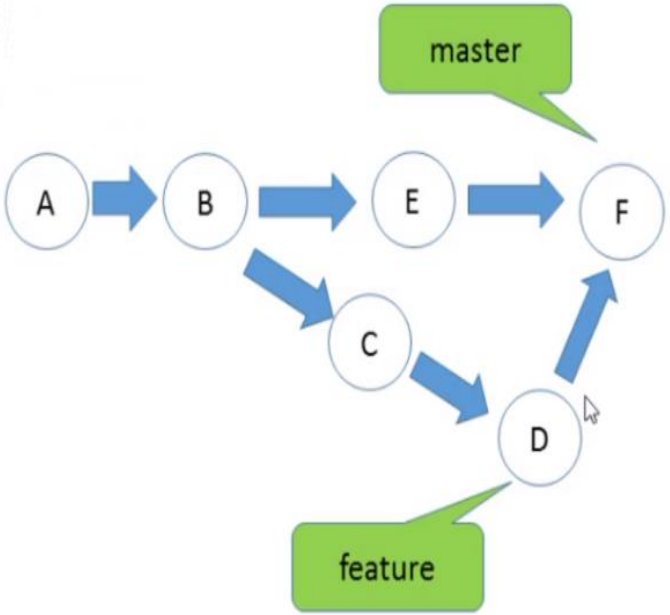
Fast forward merge is employed when you merge into a branch, the latest commit of which is your parent.



As you can observe in the screenshot, B is the master branch and we have two new commits (features/bug fixes) and there is no any other commits to the master branch so we can easily merge the latest commit with master branch and shift master tag to D as you can observe in diag. So no need to create extra commit for that. This is fast forward merge

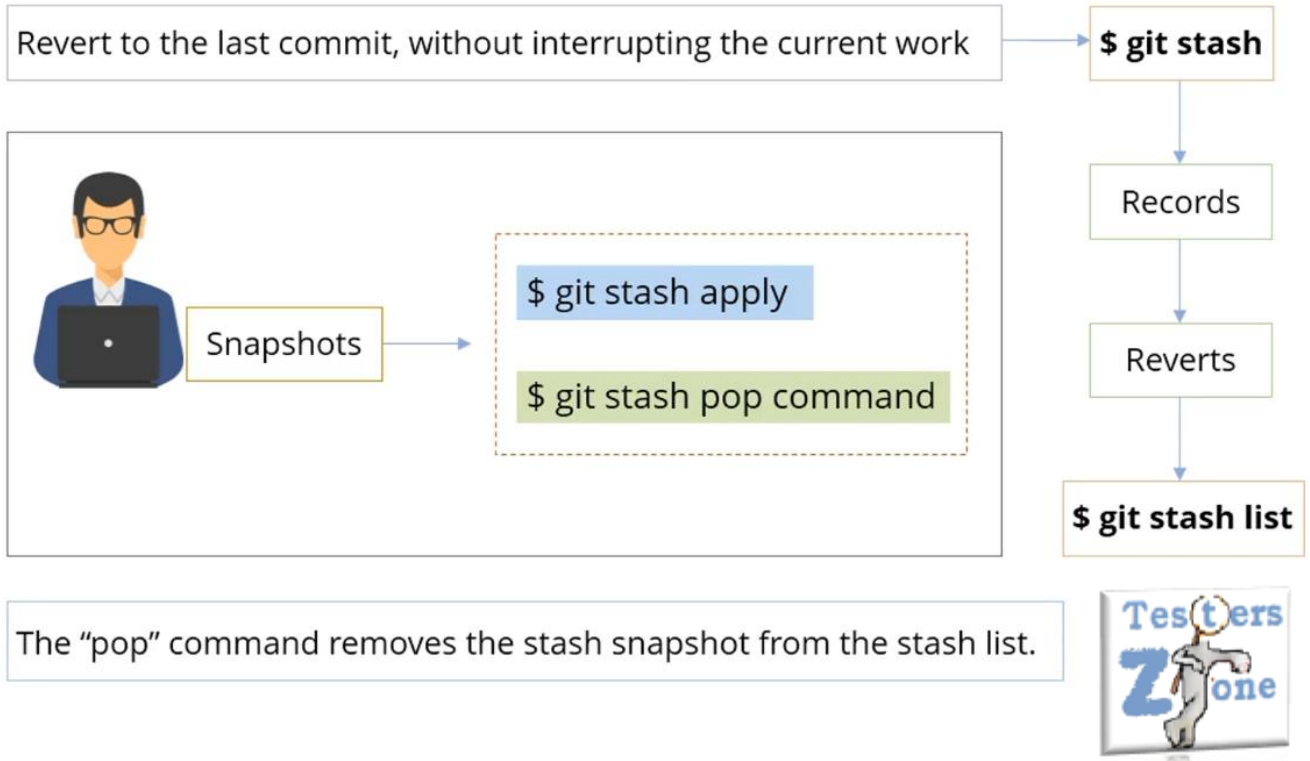
Fast forward merge can be performed when there is a direct linear path from the source branch to the target branch. In fast-forward merge, git **simply moves the source branch pointer to the target branch pointer without creating an extra merge commit**. ... In git a branch is nothing but a pointer to a commit.

In recursive merge, Git creates another commit called F and brings all the changes into this additional commit.



Note: look at previous slide first to get the understanding on Fast forward merge first.

In this merge process as you can observe in the diagram B is master branch and we would have feature changes or fixes at D but same time there might be some other changes which pushed to master B so in this case we can't make D as master branch or else other's changes will be lost [pointed by E]. so to avoid this we can pull all the master branch changes to our local and on top of that we can add our changes and create one new commit that is F in this case. This F now can be marked as master branch because it will have all the changes.



Git Stash??

git stash command helps to checkout the other branches without commit the current changes.
Take the scenario like you are working in a local branch and having some updates there but you need to checkout master branch for some reason so in this case you can stash your all changes using git stash command and then easily checkout.
We have to commit the changes or stash before checking out the branch. Git stash create a snapshot of your current changes and move your branch to last commit so when ever we want we can get those changes using **git stash apply** command.
git stash save command helps to stash current changes with user's message. User can paas any messgae to identify stashed code easily.
Basic difference between **git stash apply** and **git stash pop** command is, pop will remove the current changes snapshot once it is used so that you can not use again and again.
git stash list provides all the stashed list.

What is git clone?

- Cloning is the process to create a copy of original git repository in our local and link with the original git repository to work with collaboration.
- Let's understand the steps:
  1. Create a new folder in your system.
  2. Navigate to the folder and open command line inteface.
  3. Copy the git repository url from the git repo and use git clone <repo link> in CLI and enter.
  4. You might have asked credential, so type your credential and enter
  5. You will have copy of original git repository now you can create your own branch and add your changes.

  Note: how to create branch and add changes and merge it with original repository all have covered in previous slides.

We have completed most of the part of Git and Git repository but one question where most of the people feel trouble I.e.

What is difference between git, gitlab, git hub and bitbucket?

The **main difference** between Git and Bitbucket, gitlasb, github is that **Git is a distributed version control system while** Bitbucket, gitlasb, github **is a web-based version control repository hosting service for development projects that use Git.**

The version control system allows the software developer to share code and to maintain the history of their work. It can store the modification on files and source code. Version control system saves the state of the project and saves them each time the user does a modification on the project. Each saved state of the project is called a version. Overall, Git is a version control system, whereas Bitbucket, gitlasb, github is a version control hosting service.

For deeper insights you can visit below links:

https://www.geeksforgeeks.org/bitbucket-vs-github-vs-gitlab/
https://stackshare.io/stackups/bitbucket-vs-github-vs-gitlab

Mithilesh Singh