**Git and Remote Repositories**

Git :

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

* + Tracking code changes
  + Tracking who made changes
  + Coding collaboration

To check the version : git --version

git version 2.30.2.windows.1

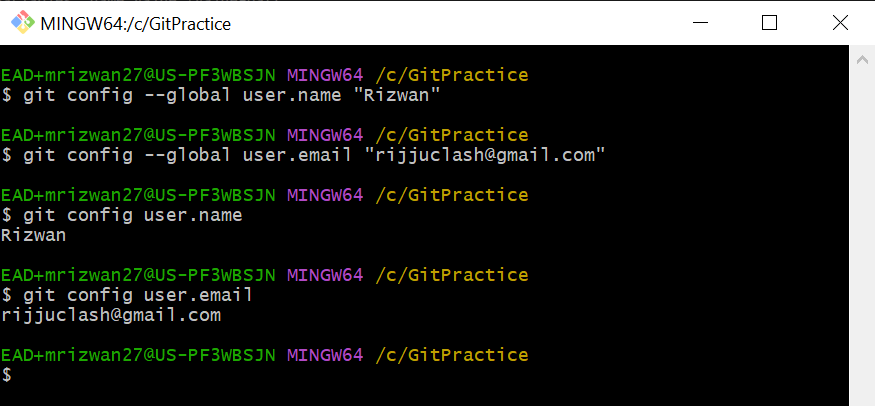
Configure Git : git config --global user.name "w3schools-test"

git config --global user.email "test@w3schools.com"

Creating Git Folder :

mkdir myproject

cd myproject



**Initialize Git :**

git init “Initialized empty Git repository in” /Users/user/myproject/.git/

Master ---🡪 is the main repository

Create a file : Index.html

Files in your Git repository folder can be in one of 2 states:

* Tracked - files that Git knows about and are added to the repository
* Untracked - files that are in your working directory, but not added to the repository

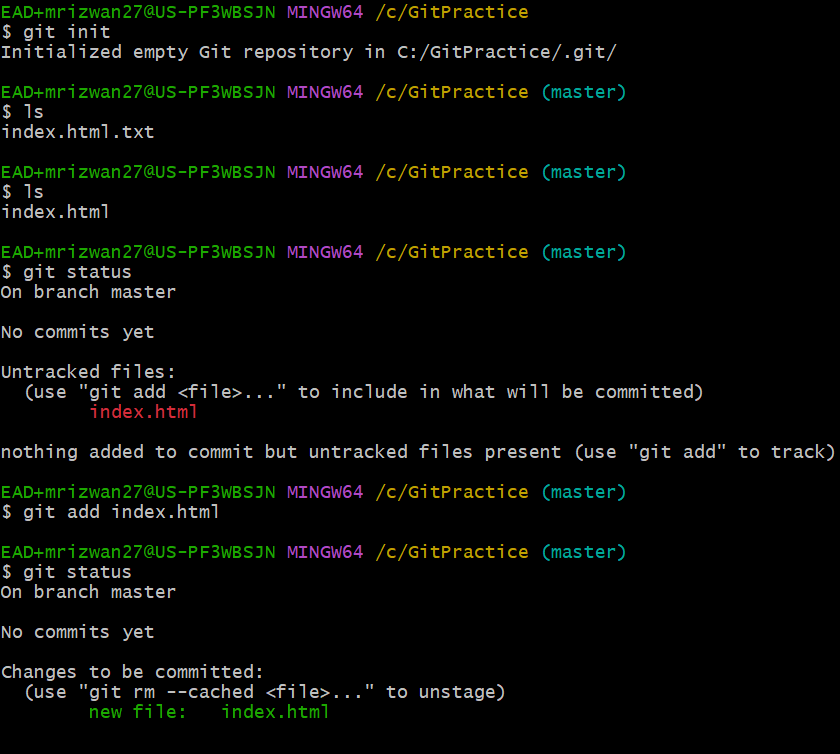
Git Staging Environment

One of the core functions of Git is the concepts of the Staging Environment, and the Commit.

As you are working, you may be adding, editing and removing files. But whenever you hit a milestone or finish a part of the work, you should add the files to a Staging Environment.

**Staged** files are files that are ready to be **committed** to the repository you are working on. You will learn more about commit shortly.

* For now, we are done working with index.html. So we can add it to the Staging Environment:



## Git Add More than One File

You can also stage more than one file at a time. Let's add 2 more files to our working folder. Use the text editor again.

A README.md file that describes the repository

**README.md**

# hello-world  
Hello World repository for Git tutorial  
This is an example repository for the Git tutoial on https://www.dxc.com  
  
This repository is built step by step in the tutorial.

**bluestyle.css :**

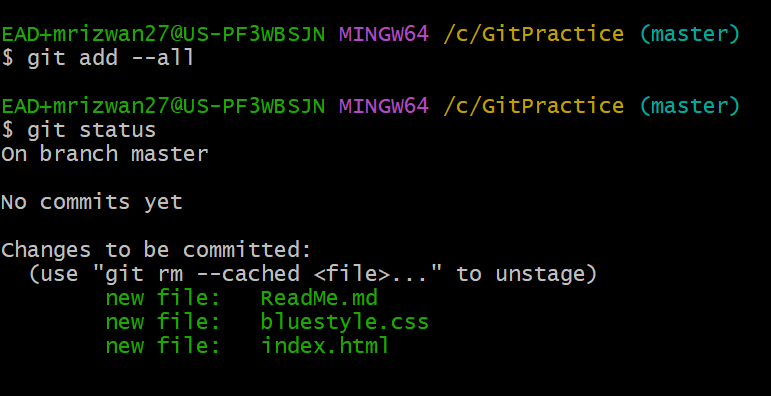
body {  
background-color: lightblue;  
}  
  
h1 {  
color: navy;  
margin-left: 20px;  
}

**index.html:**

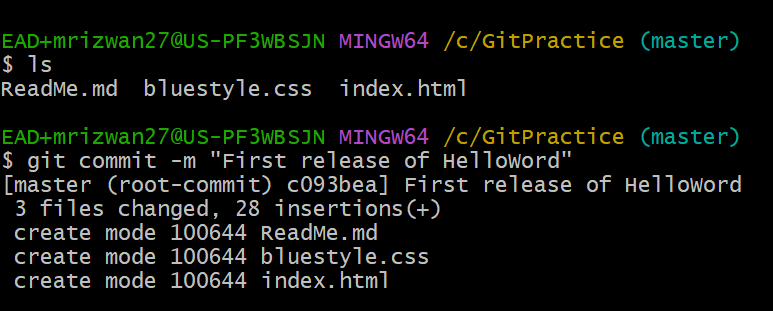
<!DOCTYPE html>  
<html>  
<head>  
<title>Hello World!</title>  
<link rel="stylesheet" href="bluestyle.css">  
</head>  
<body>  
  
<h1>Hello world!</h1>  
<p>This is the first file in my new Git Repo.</p>  
  
</body>  
</html>

Now add all files in the current directory to the Staging Environment:

git add --all



**Note:** The shorthand command for git add --all is git add -A



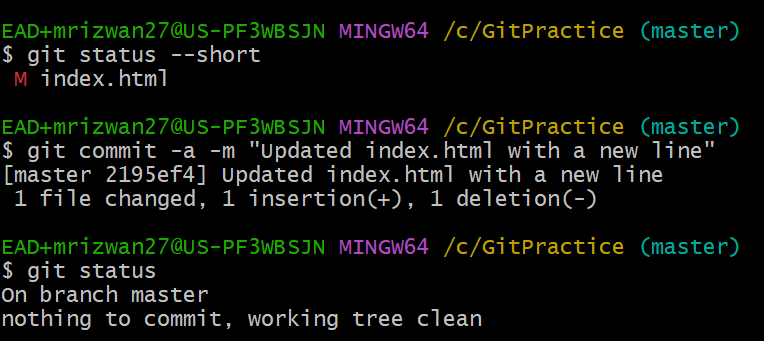
## Git Commit:

Since we have finished our work, we are ready move from stage to commit for our repo.

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

When we commit, we should **always** include a **message**.

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.



## Git Commit without Stage

Sometimes, when you make small changes, using the staging environment seems like a waste of time. It is possible to commit changes directly, skipping the staging environment. The -a option will automatically stage every changed, already tracked file.

## Index.html add new line :

## <p>This is the first file in my new Git Repo.</p>

**Note:** Short status flags are:

* ?? - Untracked files
* A - Files added to stage
* M - Modified files
* D - Deleted files

## we see the file we expected is modified. So let's commit it directly:

## Git Commit Log

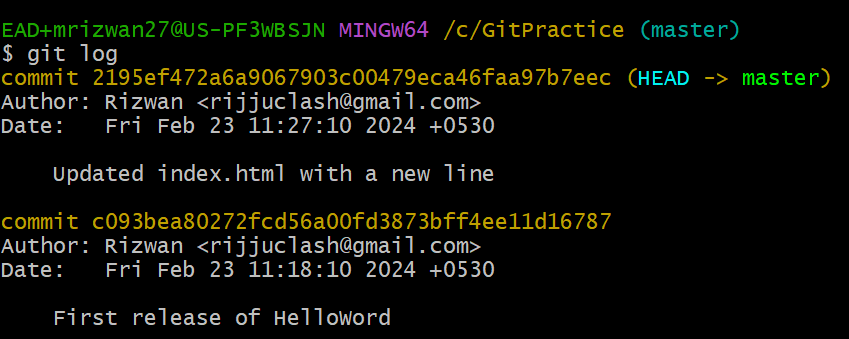
To view the history of commits for a repository, you can use the log command:

Git Help

If you are having trouble remembering commands or options for commands, you can use Git help.

There are a couple of different ways you can use the help command in command line:

* git *command* -help -  See all the available options for the specific command
* git help --all -  See all possible commands



## New Git Branch

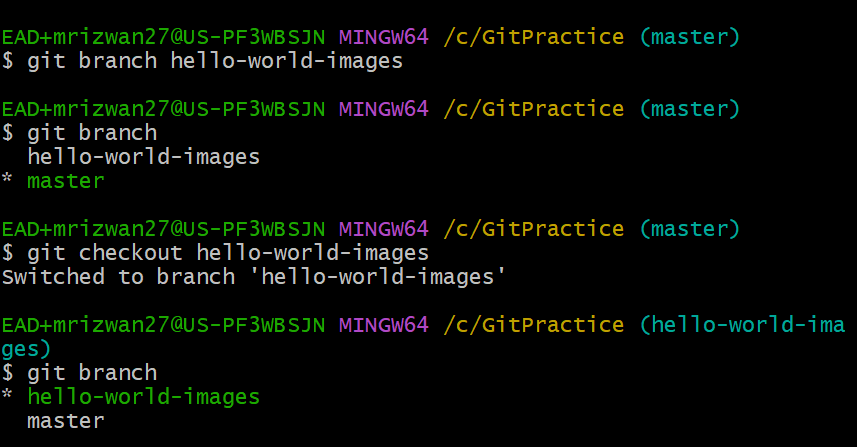
Let add some new features to our index.html page.

We are working in our local repository, and we do not want to disturb or possibly wreck the main project.

So we create a new branch:

We can see the new branch with the name "hello-world-images", but the \* beside master specifies that we are currently on that branch.

checkout is the command used to check out a branch. Moving us **from** the current branch, **to** the one specified at the end of the command:



Now we have moved our current workspace from the master branch, to the new branch

Open your favourite editor and make some changes.

For this example, we added an image (img\_hello\_world.jpg) to the working folder and a line of code in the index.html file:

## Index.html :

## <!DOCTYPE html>

## <html>

## <head>

## <title>Hello World!</title>

## <link rel="stylesheet" href="bluestyle.css">

## </head>

## <body>

## <h1>Hello world!</h1>

## <div><img src="C:\Users\ypavani\OneDrive - DXC Production\Pictures\DXC Logo\DXC Logo 2C Horiz\_White RGB.png" alt="DXC LOGO Bangalore"

## style="width:100%;max-width:960px"></div>

## <p>A new line in our file!</p>

## <p>This is the first file in my new Git Repo.</p>

## </body>

## </html>

## 

## 

## We are happy with our changes. So we will commit them to the branch:

## 

## **Note:** Using the -b option on checkout will create a new branch, and move to it, if it does not exist

## Switching Between Branches

## 

## 

## Emergency Branch

Now imagine that we are not yet done with hello-world-images, but we need to fix an error on master.

I don't want to mess with master directly, and I do not want to mess with hello-world-images, since it is not done yet.

So we create a new branch to deal with the emergency:

## 

## Index.html :

## <!DOCTYPE html> <html> <head> <title>Hello World!</title> <link rel="stylesheet" href="bluestyle.css"> </head> <body> <h1>Hello world!</h1> <p>This is the first file in my new Git Repo.</p> <p>This line is here to show how merging works.</p> </body> </html>

## 

## Merge Branches

We have the emergency fix ready, and so let's merge the master and emergency-fix branches.

First, we need to change to the master branch:

## 

## Delete Branches :

## 

## Merge Conflict:

## first checkout to hello-world-image

## Add image to index.html

## Index.html:

## <!DOCTYPE html>

## <html>

## <head>

## <title>Hello World!</title>

## <link rel="stylesheet" href="bluestyle.css">

## </head>

## <body>

## <h1>Hello world!</h1>

## <div><img src="C:\GitPractice\DXC Logo 2C Horiz\_White RGB.png" alt="DXC TECHNOLOGY "

## style="width:100%;max-width:960px"></div>

## <p>This is the first file in my new Git Repo.</p>

## <p>A new line in our file!</p>

## <div><img src="C:\GitPractice\jerry.webp" alt="DXC" style="width:100%;max-width:640px"></div>

## </body>

## </html>

## 

## And delete the hello-world-images branch:

## 

## Push Local Repository to GitHub

## Since we have already set up a local Git repo, we are going to push that to GitHub:

## 

## Edit Code in GitHub

In addition to being a host for Git content, GitHub has a very good code editor.

Let's try to edit the README.md file in GitHub. Just click the edit button:

## 

## 

## Finally we edited the code

Pulling to Keep up-to-date with Changes

When working as a team on a project, it is important that everyone stays up to date.

Any time you start working on a project, you should get the most recent changes to your local copy.

With Git, you can do that with pull.

pull is a combination of 2 different commands:

* fetch
* merge

## Git Fetch

fetch gets all the change history of a tracked branch/repo.

So, on your local Git, fetch updates to see what has changed on GitHub:

## 

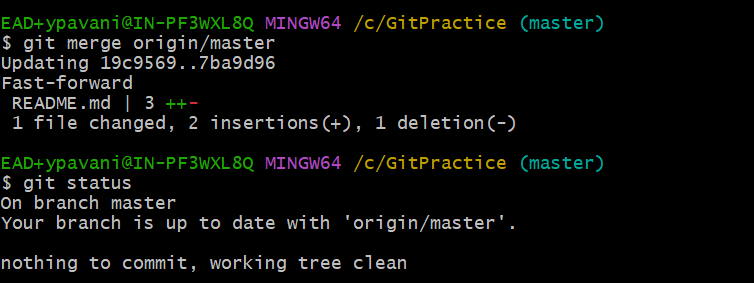
## 

## To see the difference between our local master and origin/master

## Git Merge

merge combines the current branch, with a specified branch.

We have confirmed that the updates are as expected, and we can merge our current branch (master) with origin/master:

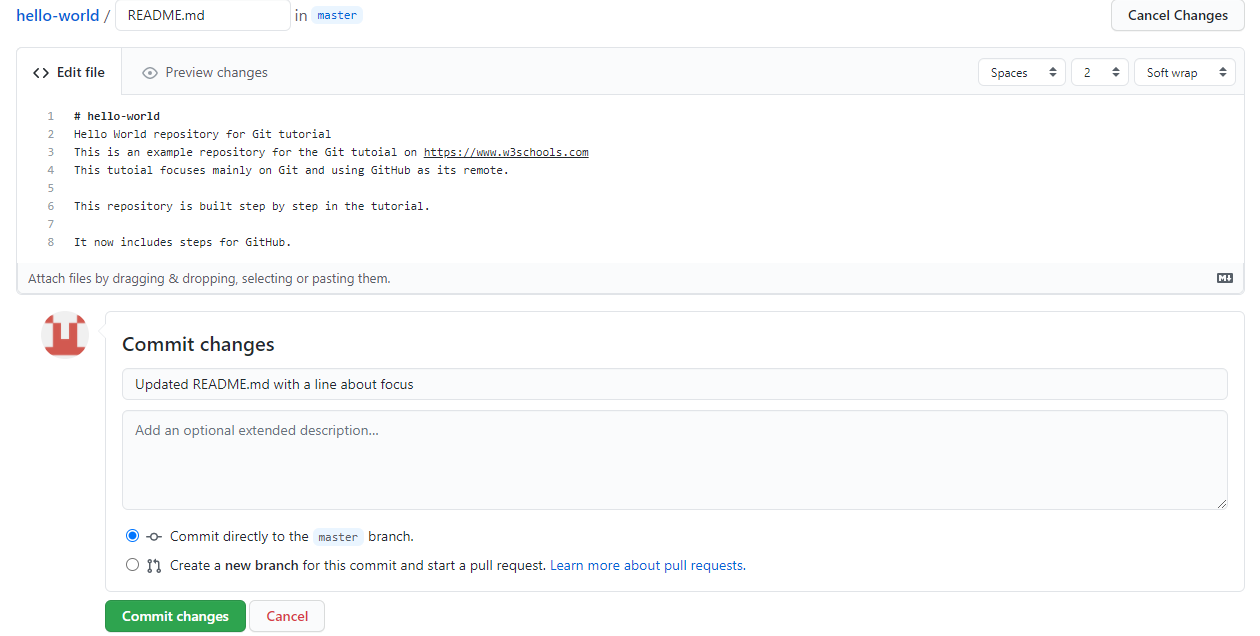


## Git Pull

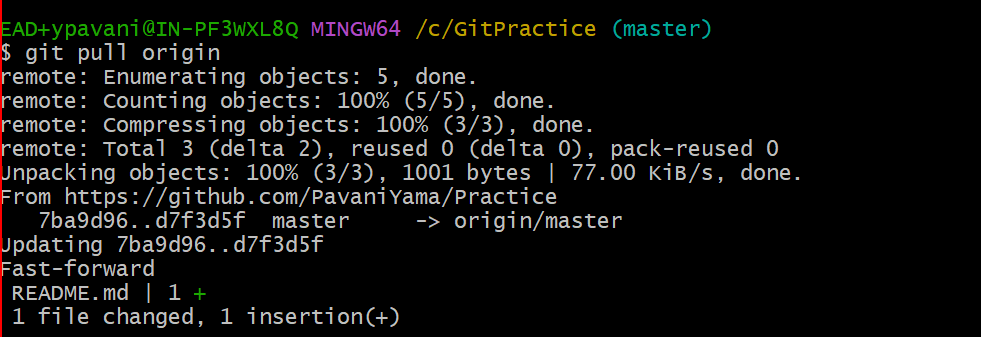
But what if you just want to update your local repository, without going through all those steps?

pull is a combination of fetch and merge. It is used to pull all changes from a remote repository into the branch you are working on.

Make another change to the Readme.md file on GitHub.



Use pull to update our local Git:



## Push Changes to GitHub

Let's try making some changes to our local git and pushing them to GitHub.

Index.html : <div><img src="img\_hello\_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>

