What is Git?

Git is a free and open source version control system which helps us to track changes to our files over time. With Git, we can revert to various states of the files. We can also make a copy of the file, make changes to that copy, and then merge these changes to the original copy.

What is Github?

GitHub is a web-based hosting service for Git repositories which allows us to create a remote copy of our local version-controlled project. This can be used as a backup or archive of the project or make it accessible to us and to our colleagues so that we can work collaboratively.

Version Control

The term version control system is a software tool that records all the changes made to a file or set of data, that make up a particular project and allows us to revert to previous versions of files if needed. This feature makes the process of collaboration so feasible with all team members, making it considerably more comfortable to work over a big project.

Why Version control

Version control automatically takes care of keeping a record of all the versions of a particular file and allows us to revert back to previous versions if needed. Version control also helps us to keep track of all the versions of files in a single place and it helps others (especially collaborators) review, contribute to and reuse the work through the GitHub website. Lastly, our files are always available from anywhere and on any computer, all that need is an internet connection.

A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together. As developers make changes to the project, any earlier version of the project can be recovered at any time.

Developers can review project history to find out:

- Which changes were made?
- Who made the changes?
- When were the changes made?
- Why were changes needed?

Benefits of Version Control System

Simplify code review

Modify code efficiently

Revert changes that introduce bugs

Maintaining multiple versions of the project

Better collaboration and improved team productivity

Easy integration with latest productivity tools

Git Repository Structure

It consists of 4 parts:

Working directory: This is our local directory where we can make the project (write code) and make changes to it.

Staging Area (or index): this is an area where we first need to put our project before committing. This is used for code review by other team members.

Local repository: The local repository is present on our computer and consists of all the files and folders. This Repository is used to make changes locally, review history, and commit when offline.

Remote repository: The remote repository refers to the server repository that may be present anywhere. This repository is used by all the team members to exchange the changes made.

Difference between Git and GitHub

S.No.	Git	GitHub
1.	Git is a software.	GitHub is a service.
2.	Git is a command-line tool	GitHub is a graphical user interface
3.	Git is installed locally on the system	GitHub is hosted on the web
4.	Git is maintained by linux.	GitHub is maintained by Microsoft.
5.	Git is focused on version control and code sharing.	GitHub is focused on centralized source code hosting.
6.	Git is a version control system to manage source code history.	GitHub is a hosting service for Git repositories.
7.	Git was first released in 2005.	GitHub was launched in 2008.

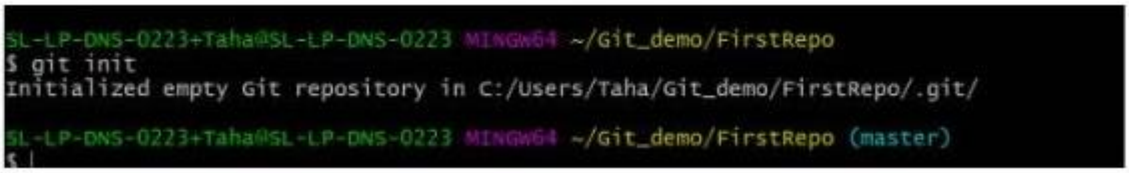
S.No.	Git	GitHub
8.	Git has no user management feature.	GitHub has a built-in user management feature.
9.	Git is open-source licensed.	GitHub includes a free-tier and pay-for-use tier.
10.	Git has minimal external tool configuration.	GitHub has an active marketplace for tool integration.
11.	Git provides a Desktop interface named Git Gui.	GitHub provides a Desktop interface named GitHub Desktop.
12.	Git competes with CVS, Azure DevOps Server, Subversion, Mercurial, etc.	GitHub competes with GitLab, Git Bucket, AWS Code Commit, etc.

Basic Git commands

To use Git, developers use specific commands to copy, create, change, and combine code. These commands can be executed directly from the command line or by using an application like GitHub Desktop. Here are some common commands for using Git:

git init - The command git init is used to create an empty Git repository. After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.

Syntax: git init



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo
$ git init
Initialized empty Git repository in C:/Users/Taha/Git_demo/FirstRepo/.git/

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ |
```

git config

- The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
- When git config is used with --global flag, it writes the settings to all repositories on the computer.

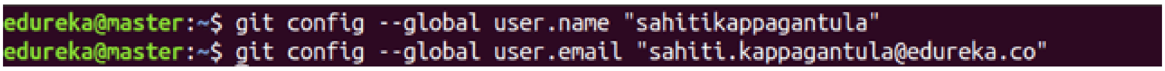
Syntax:

git config --global user.name “any user name” git config --global user.email <email id>

Example:

git config --global user.name “Anu”

git config --global user.email anu.123@gmail.com



```
edureka@master:~$ git config --global user.name "sahitikappagantula"
edureka@master:~$ git config --global user.email "sahiti.kappagantula@edureka.co"
```

git add - Add command is used to add the files to the staging area. Any changes that are staged will become a part of the next snapshot and a part of the project's history. Staging and committing separately gives developers complete control over the history of their project without changing how they code and work.

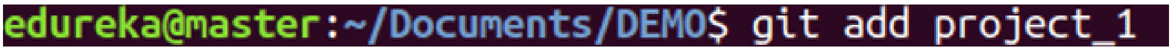
Different ways to use add command: Syntax:

git add –all or git add –A   - To add all files of current directory to staging area.

git add *.html – To add all the HTML files of current directory to staging area

git add filename -

Ex. git add project_1 - To add a specific file to staging area.

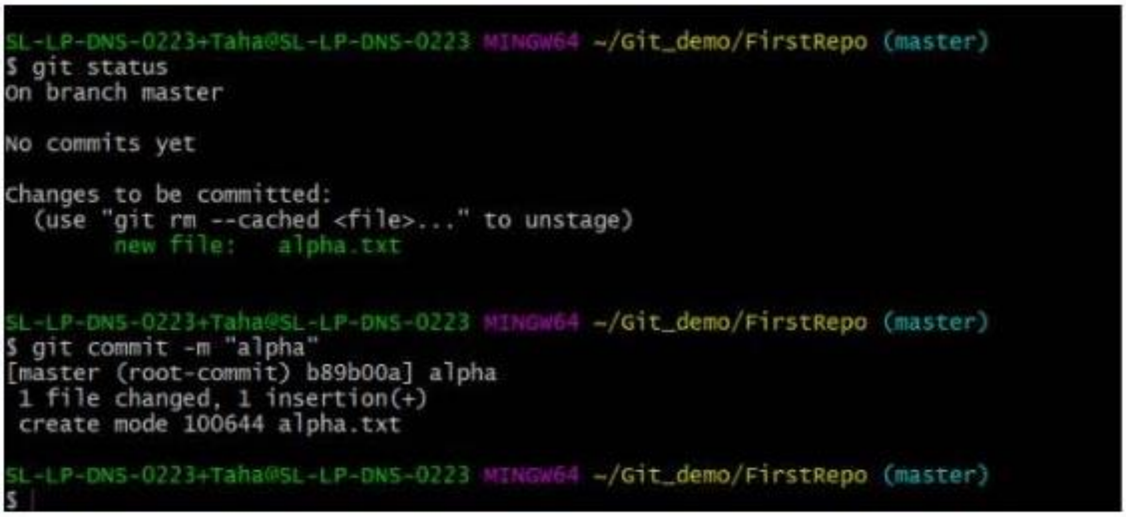


```
edureka@master:~/Documents/DEMO$ git add project_1
```

git commit   -   The commit command makes sure that the changes are saved to the local repository. It saves the snapshot to the project history and completes the change-tracking process.

Syntax: git commit –m <message>"   here the message allows us to describe everyone and
help them understand what has happened.

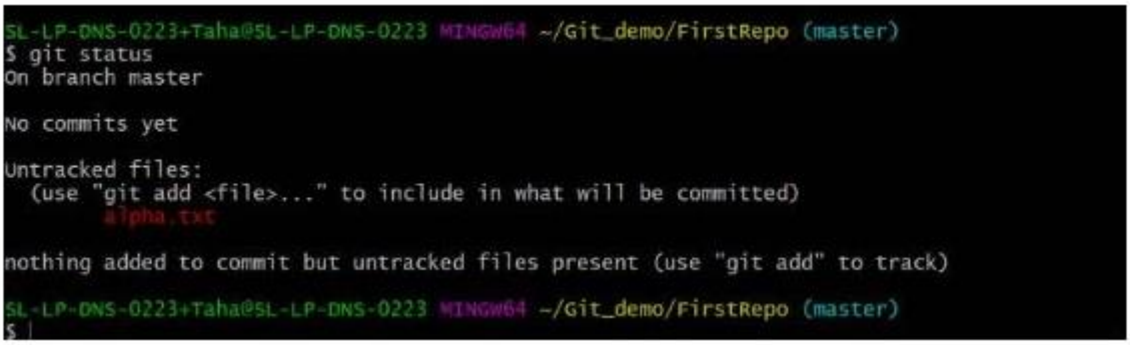
Example : git commit -m “alpha”



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master

No commits yet

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


SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git commit -m "alpha"
[master (root-commit) b89b00a] alpha
 1 file changed, 1 insertion(+)
 create mode 100644 alpha.txt

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ |
```

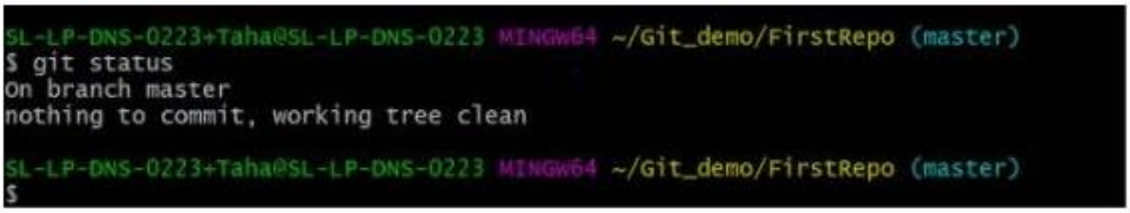
git status - The git status command tells the current state of the repository. as untracked, modified, or staged. The command provides status the current working branch. If the files are in the staging area, but not committed, it will show the status as" file yet to be commited". Also, if there are no changes, it will show the message no changes to commit, working directory clean.

Syntax: git status



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ |
```

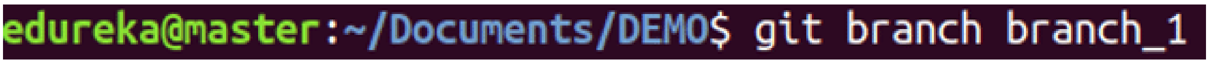
After committing all the changes: The output of status will be as follows:



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master
nothing to commit, working tree clean

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$
```

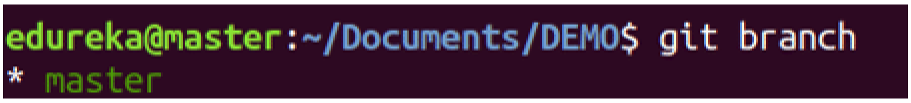
git branch – It is used to determine what branch the local repository is on.

git branch <branch_name> Ex. git branch branch_1 :: To create a new branch test



```
edureka@master:~/Documents/DEMO$ git branch branch_1
```

git branch :: List all local branches

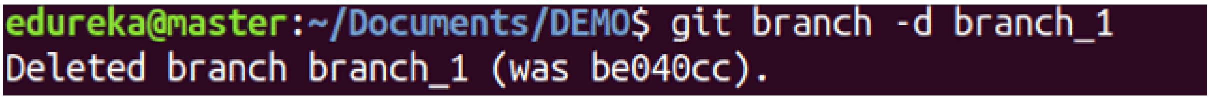


```
edureka@master:~/Documents/DEMO$ git branch
* master
```

git branch – a :: List all remote or local branches

git branch – r :: List all remote branches

git branch -d <branch_name> Ex. git branch -d branch_1 : To Delete a branch

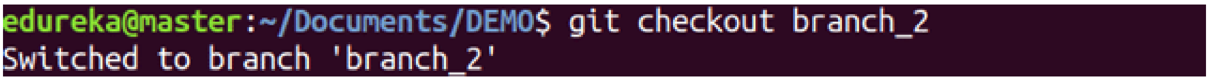


```
edureka@master:~/Documents/DEMO$ git branch -d branch_1
Deleted branch branch_1 (was be040cc).
```

git checkout : The git checkout command is used to switch branches, whenever the work is to be started on a different branch.

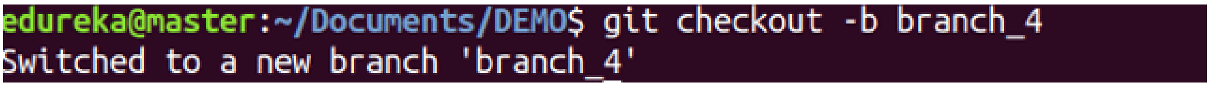
Syntax :

git checkout <branch_name> Ex. git checkout branch_2 : Checkout an existing branch



```
edureka@master:~/Documents/DEMO$ git checkout branch_2
Switched to branch 'branch_2'
```

git checkout -b <new_branch>   Ex.   git checkout –b branch_4 : create and Checkout a new branch with that name

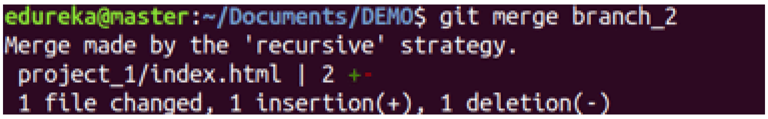


```
edureka@master:~/Documents/DEMO$ git checkout -b branch_4
Switched to a new branch 'branch_4'
```

git merge - merges lines of development together. This command is typically used to combine changes made on two distinct branches. For example, a developer would merge when they want to combine changes from a feature branch into the main branch for deployment.

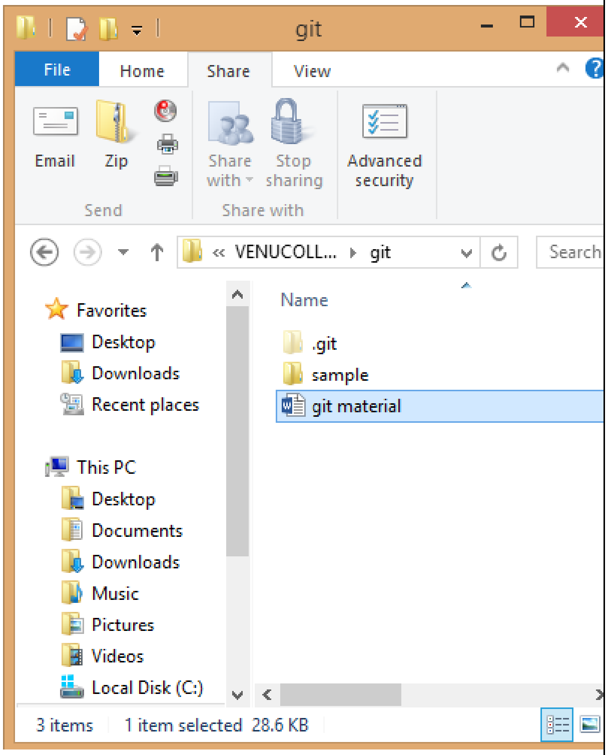
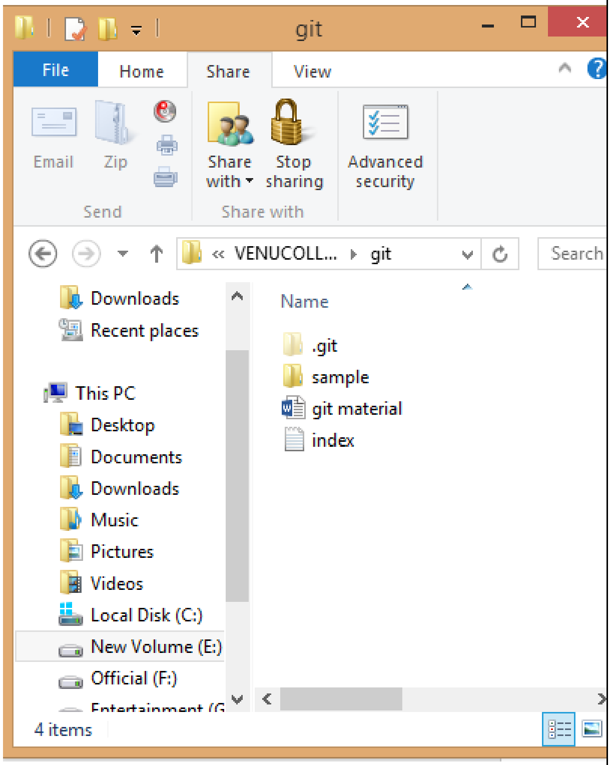
Syntax

git merge <branch_name> Ex. git merge branch_2



```
edureka@master:~/Documents/DEMO$ git merge branch_2
Merge made by the 'recursive' strategy.
 project_1/index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Before Merge

Working Directory of master branch	**Working Directory of branch_2**
	

Working Directory of master branch:

.git
sample
git material

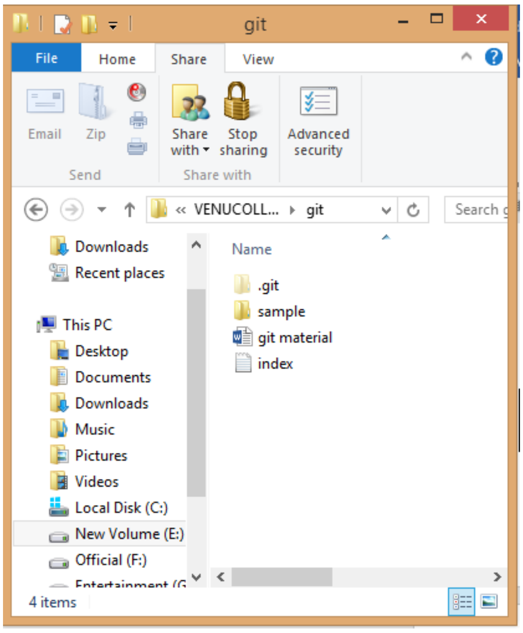
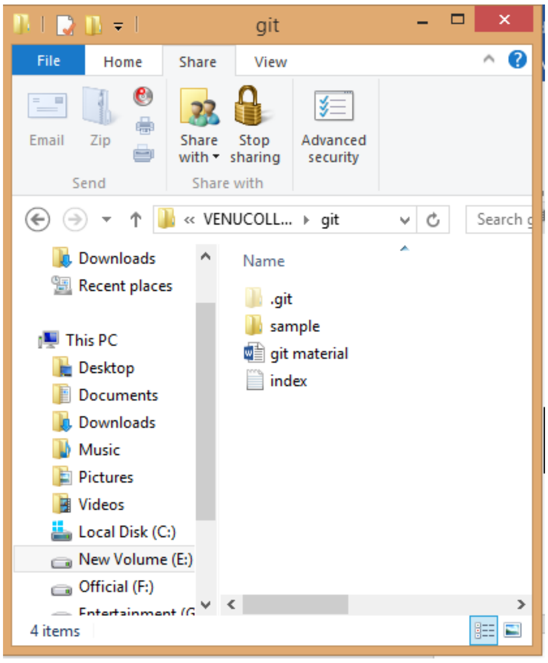
3 items   1 item selected 28.6 KB

Working Directory of branch_2:

.git
sample
git material
index

4 items

After Merge

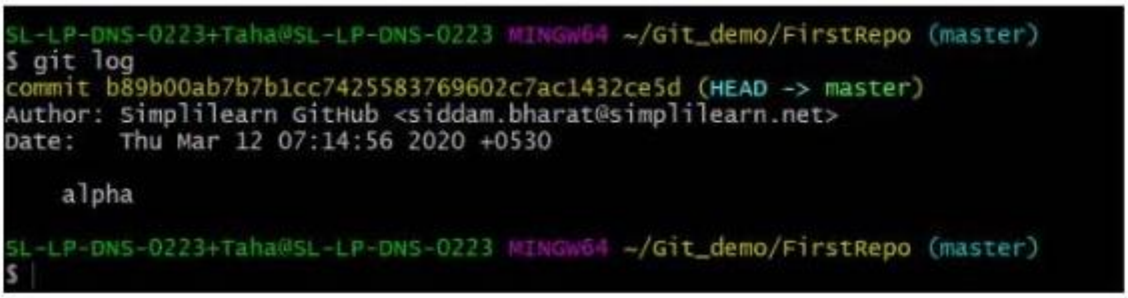
Working Directory of master branch	**Working Directory of branch_2**
 git File Home Share View Email Zip Send Share with Stop sharing Share with Advanced security VENUCOLL... git Search g Downloads Recent places This PC Desktop Documents Downloads Music Pictures Videos Local Disk (C:) New Volume (E:) Official (F:) Entertainment (G Name .git sample git material index 4 items	 git File Home Share View Email Zip Send Share with Stop sharing Share with Advanced security VENUCOLL... git Search g Downloads Recent places This PC Desktop Documents Downloads Music Pictures Videos Local Disk (C:) New Volume (E:) Official (F:) Entertainment (G Name .git sample git material index 4 items

git log

The git log command shows the order of the commit history for a repository.

The command helps in understanding the state of the current branch by showing the commits that lead to this state.

Syntax: git log



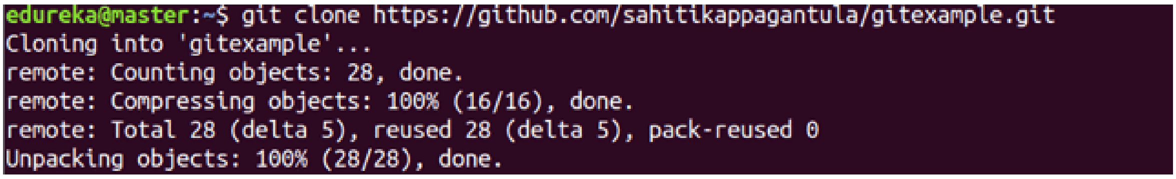
```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git log
commit b89b00ab7b7b1cc7425583769602c7ac1432ce5d (HEAD -> master)
Author: Simplilearn GitHub <siddam.bharat@simplilearn.net>
Date:   Thu Mar 12 07:14:56 2020 +0530

    alpha

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ 
```

git clone - creates a local copy of a project that already exists remotely .The command downloads the remote repository to the computer. The clone includes all the project's files, history, and branches.

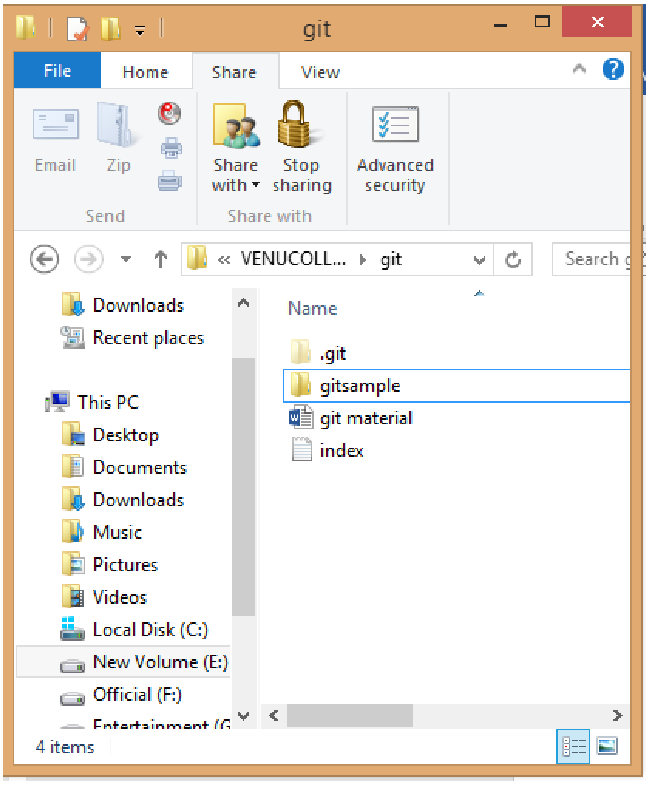
Syntax:

git clone <remote_URL>

Example: git clone https://github.com/sahitkappagantula/gitexample.git

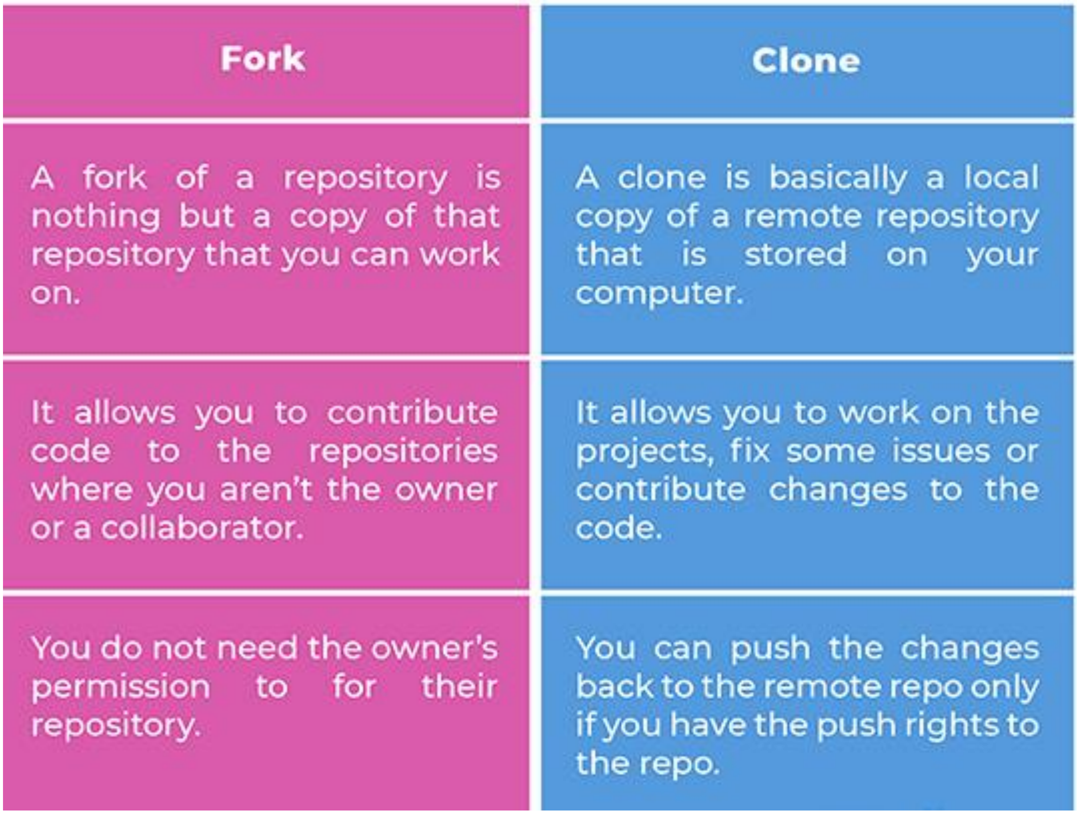


```
edureka@master:~$ git clone https://github.com/sahitikappagantula/gitexample.git
Cloning into 'gitexample'...
remote: Counting objects: 28, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 28 (delta 5), reused 28 (delta 5), pack-reused 0
Unpacking objects: 100% (28/28), done.
```



Git Fork

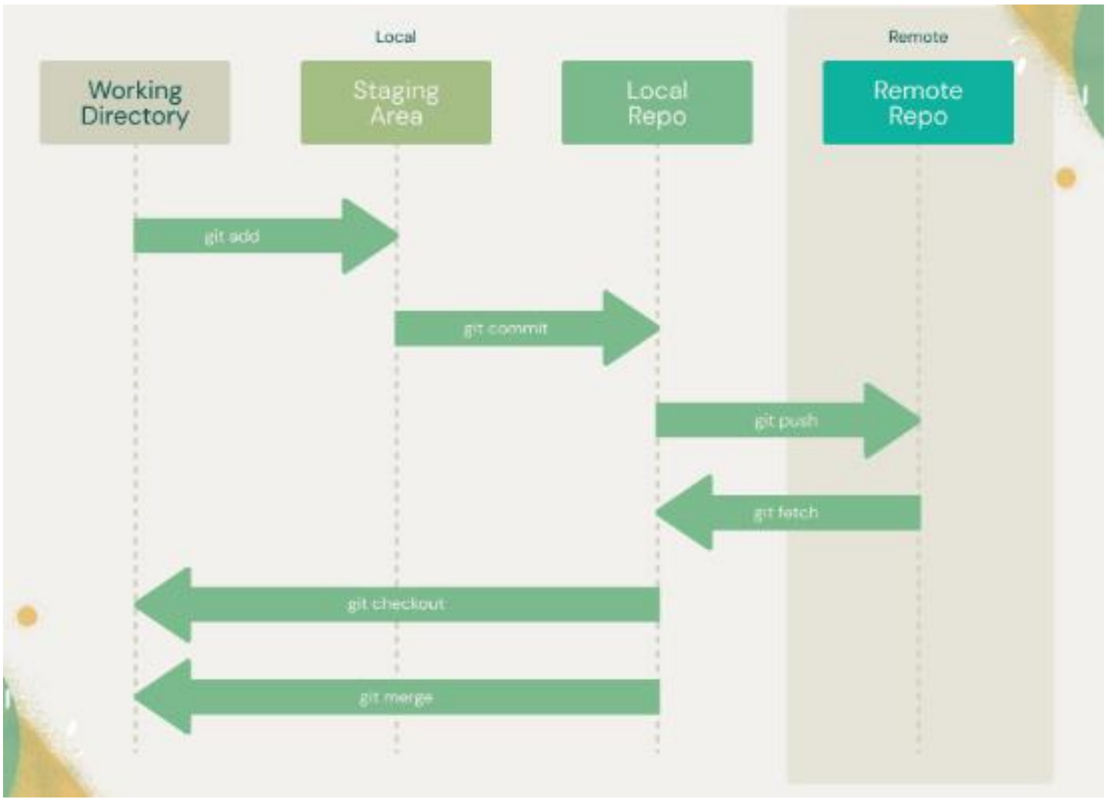
A fork is a new repository that shares code and visibility settings with the original “upstream” repository. Forks are often used to iterate on ideas or changes before they are proposed back to the upstream repository, such as in open source projects or when a user does not have write access to the upstream repository.



Fork	Clone
A fork of a repository is nothing but a copy of that repository that you can work on.	A clone is basically a local copy of a remote repository that is stored on your computer.
It allows you to contribute code to the repositories where you aren't the owner or a collaborator.	It allows you to work on the projects, fix some issues or contribute changes to the code.
You do not need the owner's permission to for their repository.	You can push the changes back to the remote repo only if you have the push rights to the repo.

Basic Git Workflow

1. When you browse and work on files in your repository, you are on a working tree, and all of your files are untracked at first.
2. The files you want to record are then staged and moved to index.
3. The staged files are then committed and saved in the local repository.
4. When you're ready to make them public, add them to a remote repository hosting service such as Github.



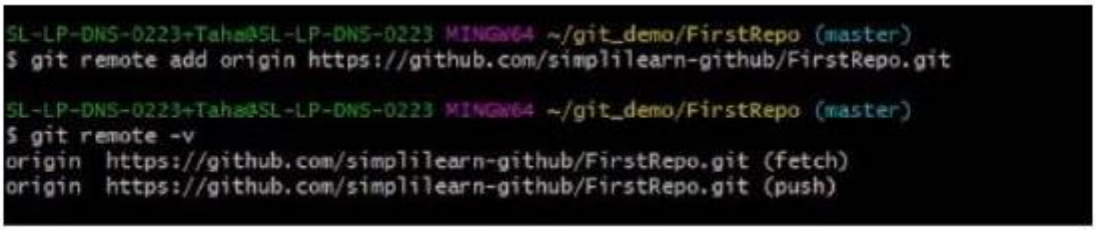
Git Commands: Working With Remote Repositories

git remote

- The git remote command is used to create, view, and delete connections to other global repositories.
- The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

Syntax git remote add [variable name] [Remote Server Link]

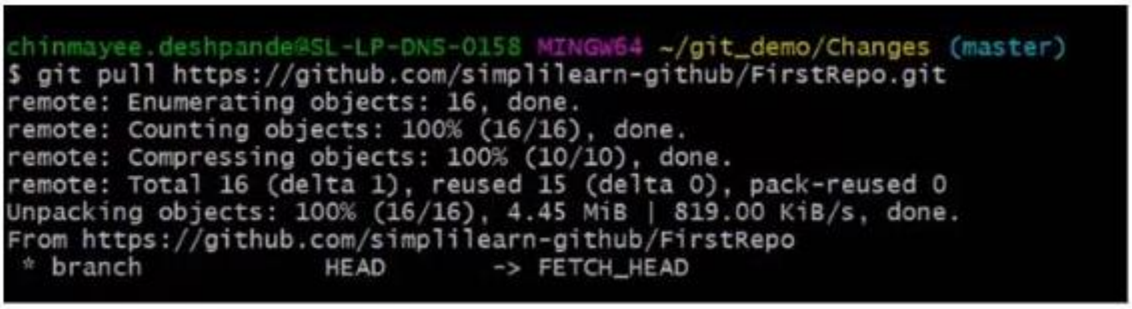
Example git remote add origin https://github.com/sahitkappagantula/gitDemo.git



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git remote add origin https://github.com/simplilearn-github/FirstRepo.git

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git remote -v
origin  https://github.com/simplilearn-github/FirstRepo.git (fetch)
origin  https://github.com/simplilearn-github/FirstRepo.git (push)
```

git pull

- The git pull command is used to fetch and merge changes from the remote repository to the local repository. Developers use this command if a teammate has made commits to a branch on a remote, and they would like to reflect those changes in their local environment.
- The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.



```
chinmayee.deshpande@SL-LP-DNS-0158 MINGW64 ~/git_demo/Changes (master)
$ git pull https://github.com/simplilearn-github/FirstRepo.git
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 16 (delta 1), reused 15 (delta 0), pack-reused 0
Unpacking objects: 100% (16/16), 4.45 MiB | 819.00 KiB/s, done.
From https://github.com/simplilearn-github/FirstRepo
 * branch            HEAD       -> FETCH_HEAD
```

git push

- The command git push is used to transfer the commits or pushing the content from the local repository to the remote repository.
- The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

Syntax: git push [variable name] master

Example: git push origin master

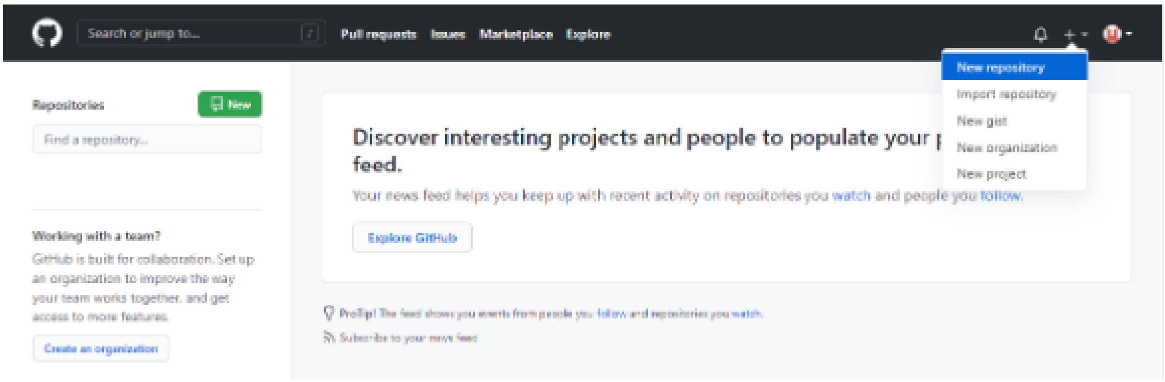
Syntax: git push [variable name] [branch]

Example: git push origin test

This command sends the branch commits to the remote repository.

Create a Repository on GitHub

Create a GitHub account, sign in, and create a new Repo and fill in the relevant details



Discover interesting projects and people to populate your p feed.

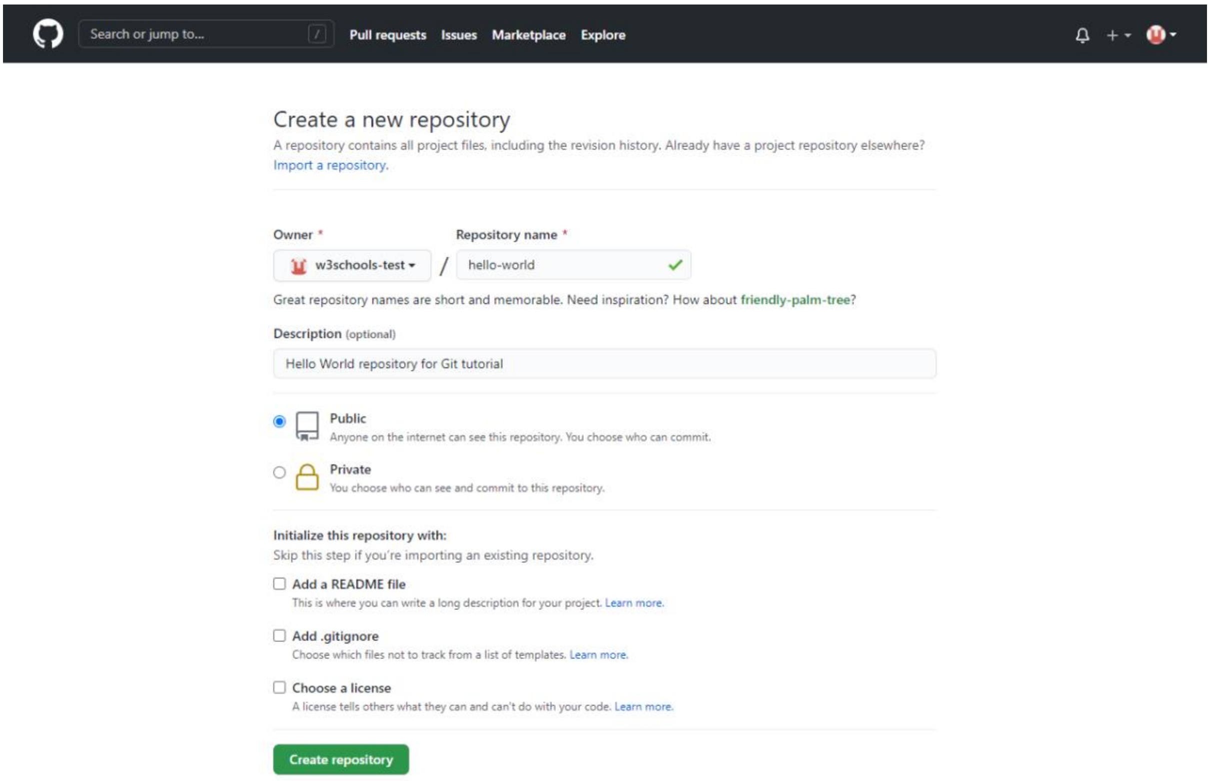
Your news feed helps you keep up with recent activity on repositories you watch and people you follow.

Working with a team?

GitHub is built for collaboration. Set up an organization to improve the way your team works together, and get access to more features.

ProTip! The feed shows you events from people you follow and repositories you watch.

Subscribe to your news feed



Create a new repository

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

Owner * w3schools-test / Repository name * hello-world

Great repository names are short and memorable. Need inspiration? How about friendly-palm-tree?

Description (optional): Hello World repository for Git tutorial

● 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.

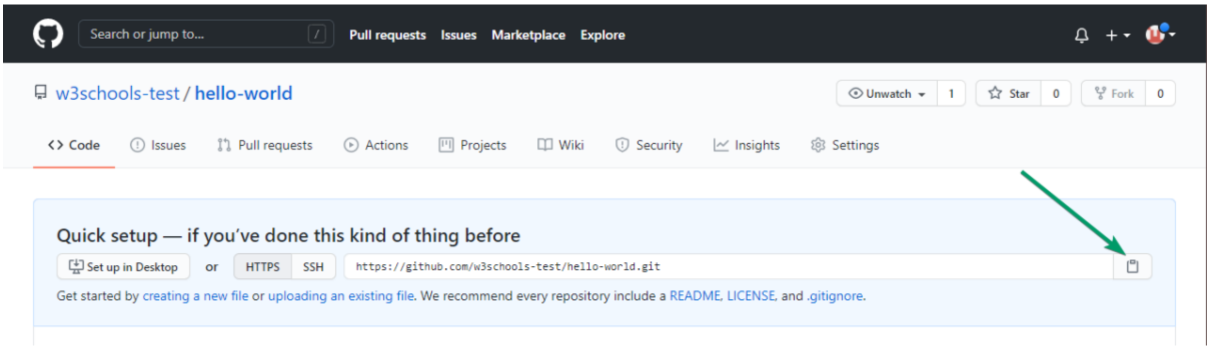
□ Add .gitignore
Choose which files not to track from a list of templates. Learn more.

□ Choose a license
A license tells others what they can and can't do with your code. Learn more.

`

Push Local Repository to GitHub

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



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

https://github.com/w3schools-test/hello-world.git

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

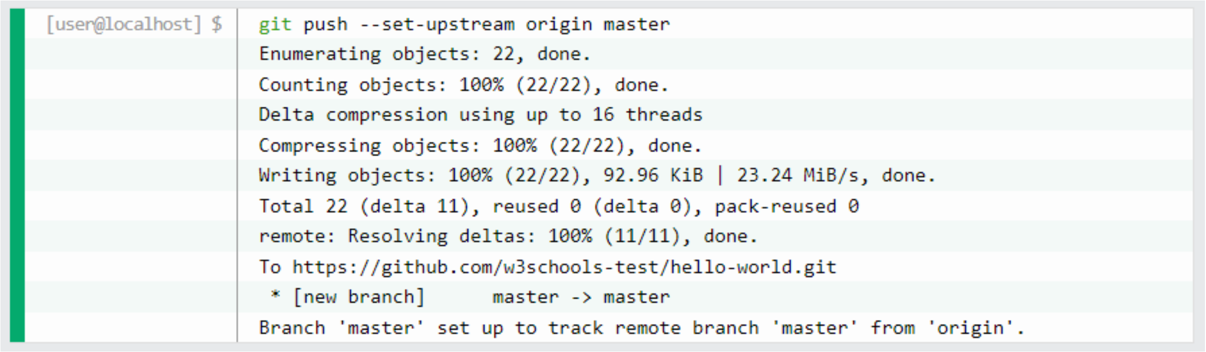
Copy the URL, or click the clipboard marked in the image above.

Now paste it the following command in local git bash to connect local and global repository

```
git remote add origin https://github.com/w3schools-test/hello-world.git
```

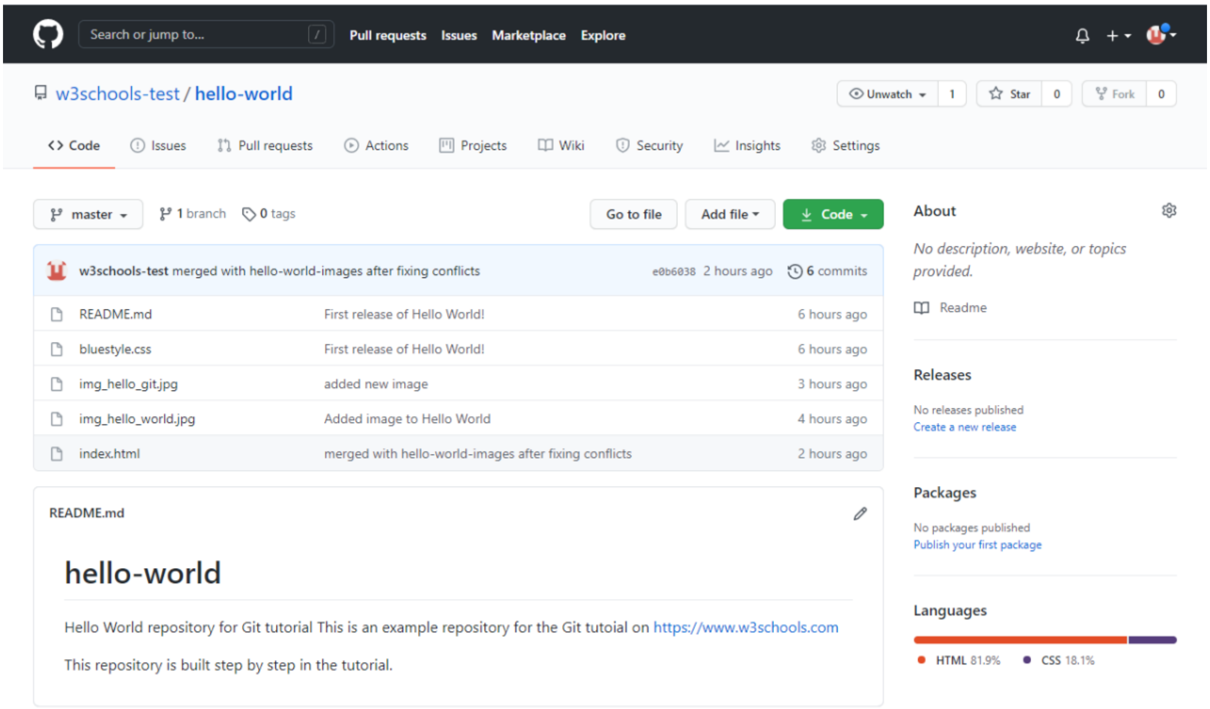
Now we are going to push our master branch to the origin url, and set it as the default remote branch:

```
git push --set-upstream origin master
```



```
[user@localhost] $   git push --set-upstream origin master
                     Enumerating objects: 22, done.
                     Counting objects: 100% (22/22), done.
                     Delta compression using up to 16 threads
                     Compressing objects: 100% (22/22), done.
                     Writing objects: 100% (22/22), 92.96 KiB | 23.24 MiB/s, done.
                     Total 22 (delta 11), reused 0 (delta 0), pack-reused 0
                     remote: Resolving deltas: 100% (11/11), done.
                     To https://github.com/w3schools-test/hello-world.git
                      * [new branch]      master -> master
                     Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Now, go back into GitHub and see that the repository has been updated:



w3schools-test merged with hello-world-images after fixing conflicts e0b6038 2 hours ago 6 commits

README.md	First release of Hello World!	6 hours ago
bluestyle.css	First release of Hello World!	6 hours ago
img_hello_git.jpg	added new image	3 hours ago
img_hello_world.jpg	Added image to Hello World	4 hours ago
index.html	merged with hello-world-images after fixing conflicts	2 hours ago

README.md

hello-world

Hello World repository for Git tutorial This is an example repository for the Git tutoial on https://www.w3schools.com

This repository is built step by step in the tutorial.

No description, website, or topics provided.

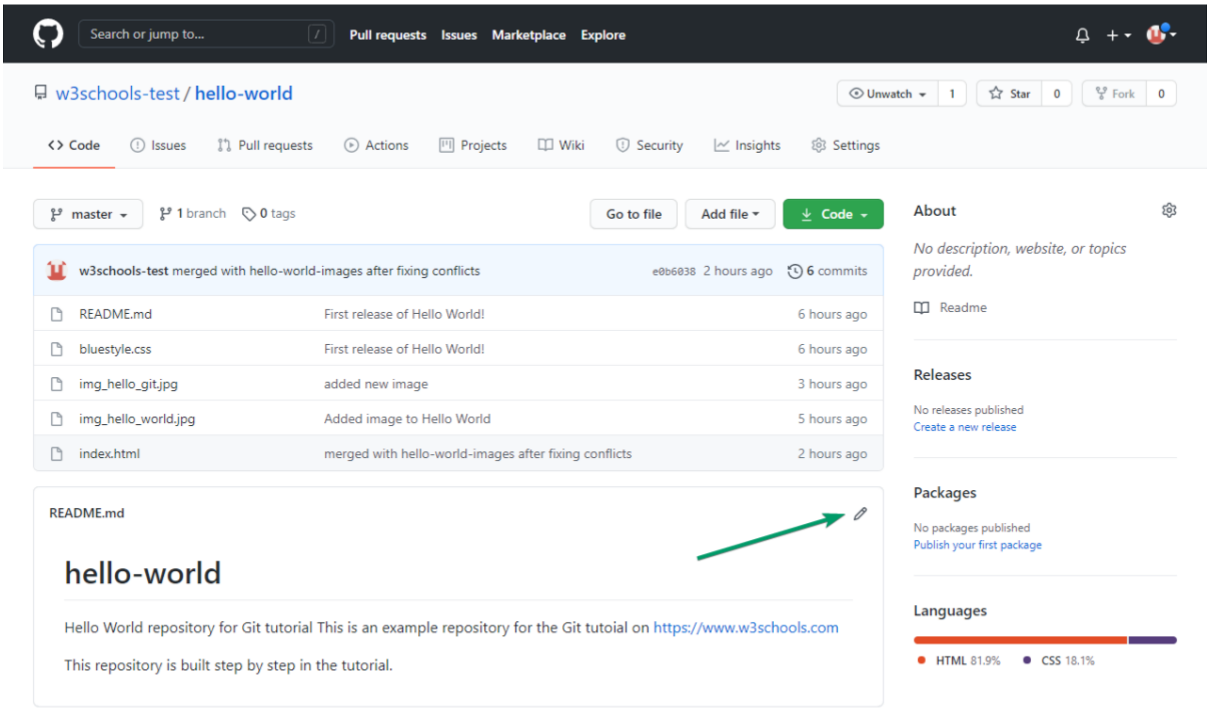
No releases published
Create a new release

No packages published
Publish your first package

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:



w3schools-test merged with hello-world-images after fixing conflicts e0b6038 2 hours ago 6 commits

README.md	First release of Hello World!	6 hours ago
bluestyle.css	First release of Hello World!	6 hours ago
img_hello_git.jpg	added new image	3 hours ago
img_hello_world.jpg	Added image to Hello World	5 hours ago
index.html	merged with hello-world-images after fixing conflicts	2 hours ago

README.md

hello-world

Hello World repository for Git tutorial This is an example repository for the Git tutoial on https://www.w3schools.com

This repository is built step by step in the tutorial.

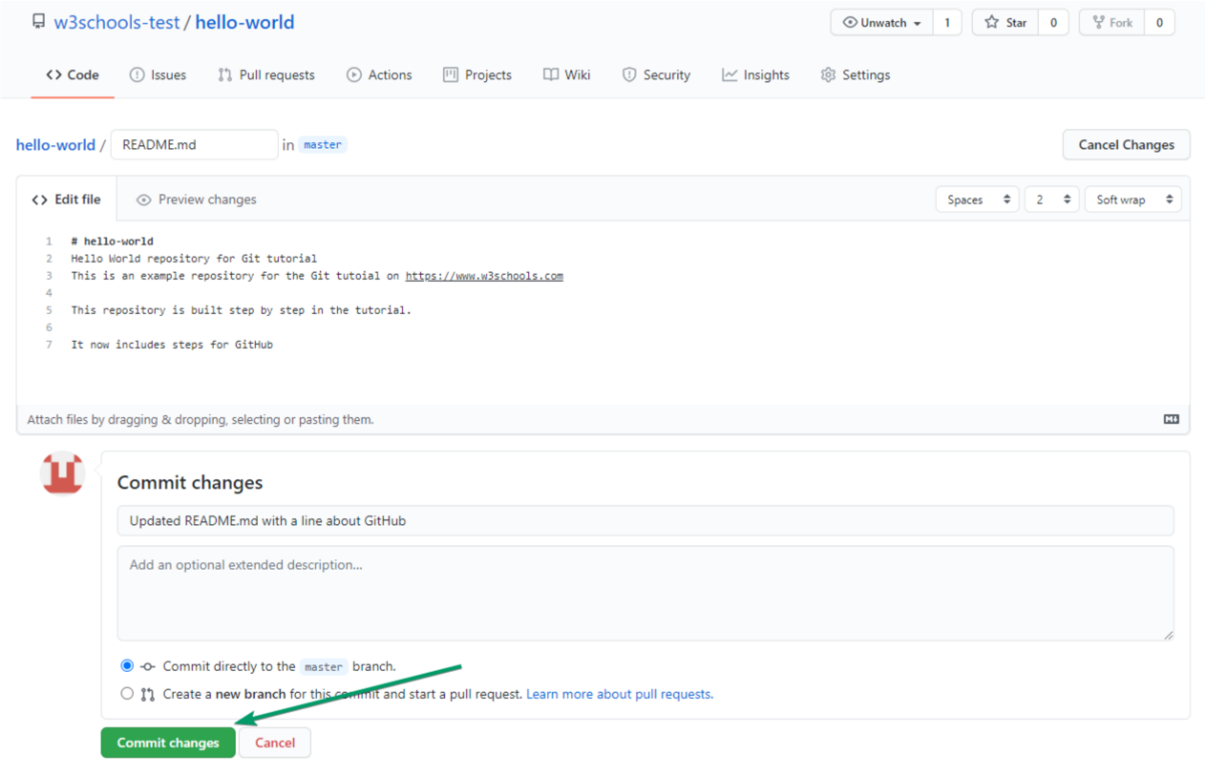
No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

● HTML 81.9% ● CSS 18.1%

Add some changes to the code, and then commit the changes. For now, we will "Commit directly to the master branch".



hello-world / README.md in master

```
# hello-world
Hello World repository for Git tutorial
This is an example repository for the Git tutoial on https://www.w3schools.com

This repository is built step by step in the tutorial.

It now includes steps for GitHub
```

Attach files by dragging & dropping, selecting or pasting them.

Commit changes

Updated README.md with a line about GitHub

Add an optional extended description...

● Commit directly to the master branch.

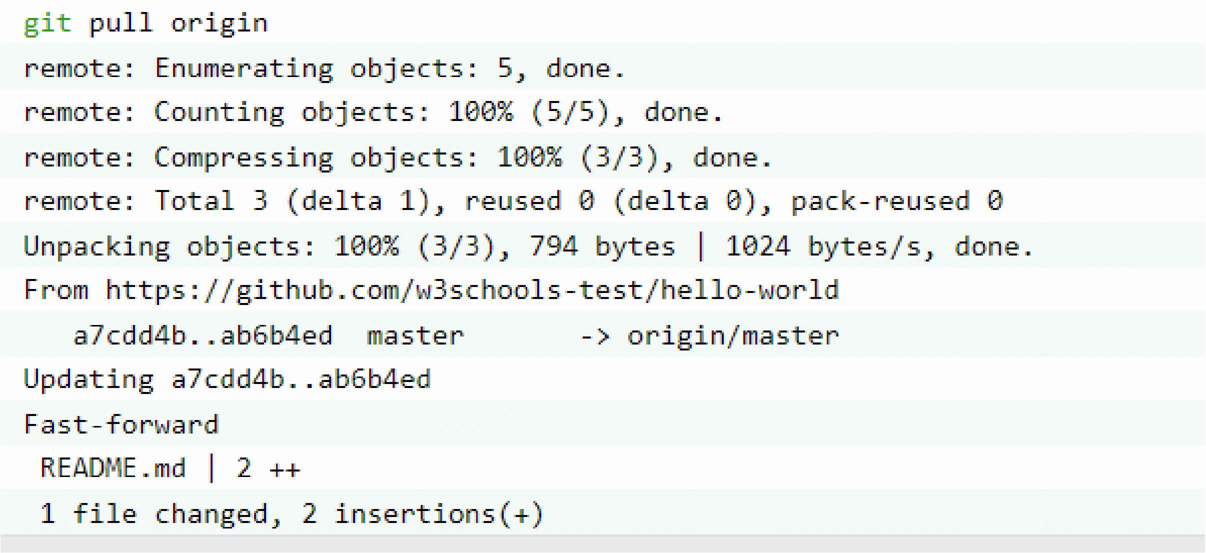
○ Create a new branch for this commit and start a pull request. Learn more about pull requests.

Pulling to Keep up-to-date with Changes in local host

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

Use pull to update our local Git:

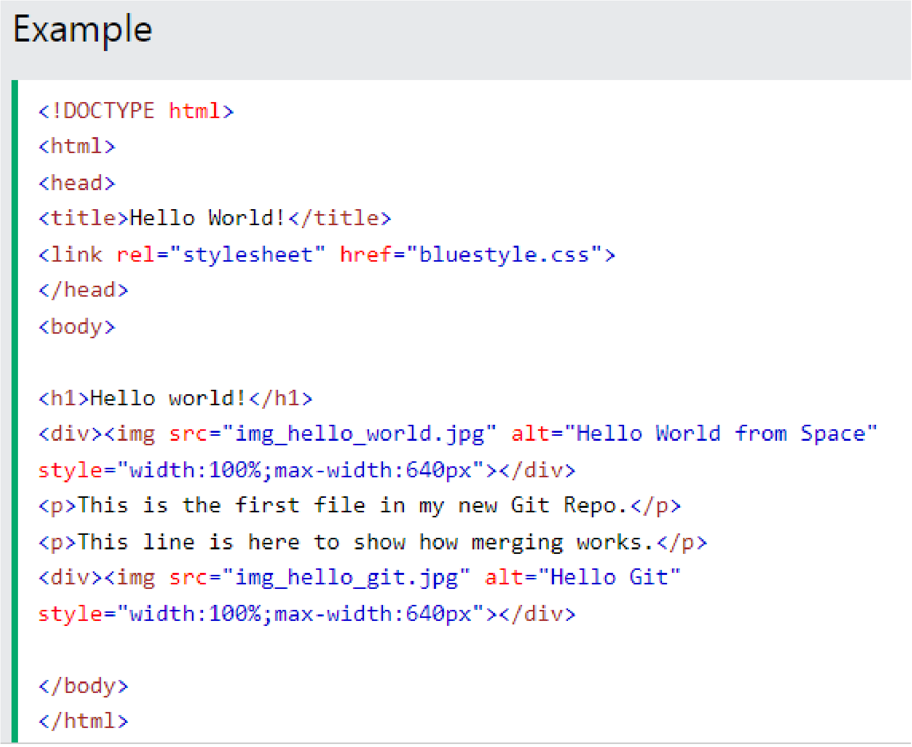
git pull origin



```
git pull origin
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), 794 bytes | 1024 bytes/s, done.
From https://github.com/w3schools-test/hello-world
   a7cdd4b..ab6b4ed  master       -> origin/master
Updating a7cdd4b..ab6b4ed
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
```

Push Changes to GitHub

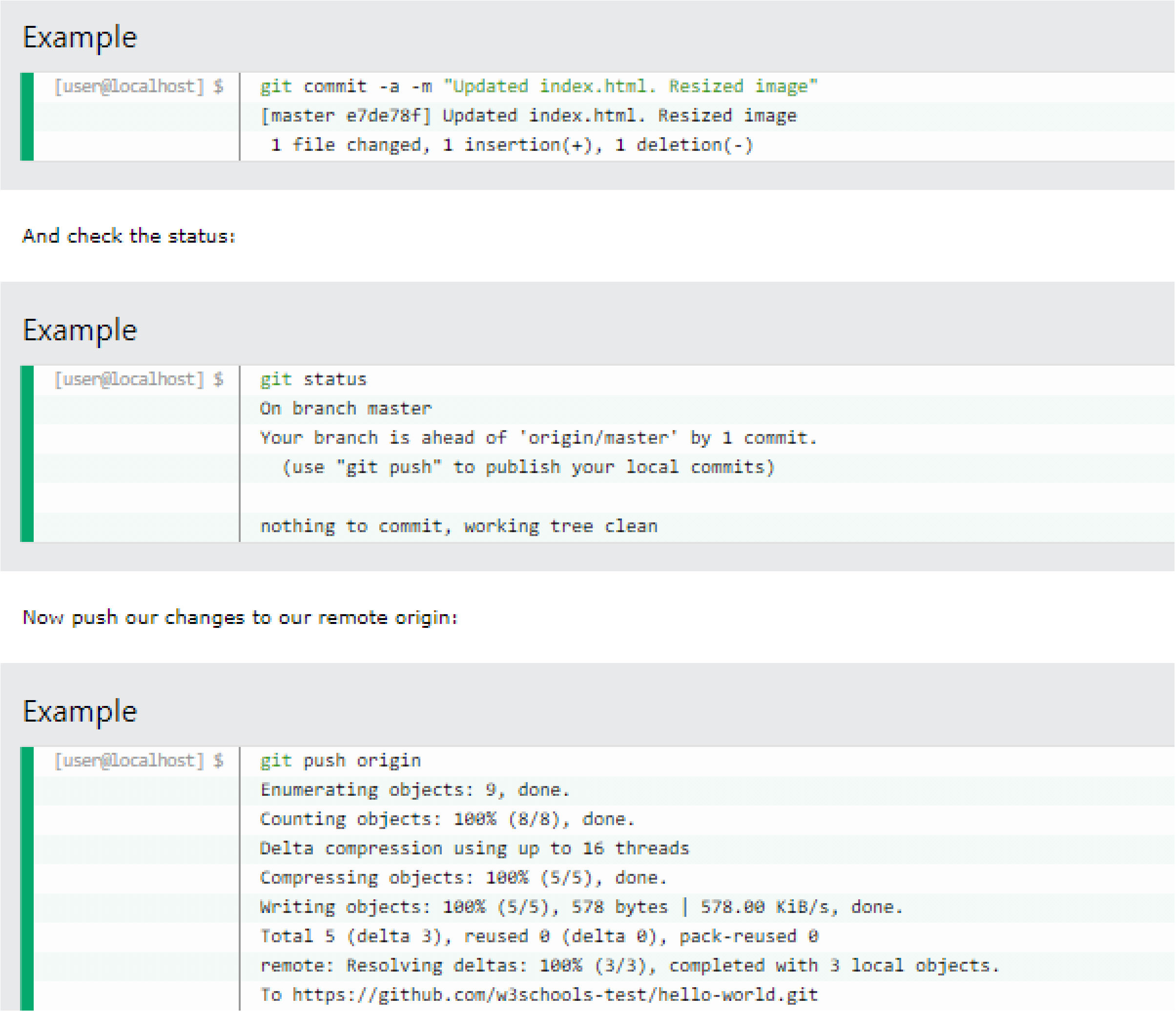
Let's try making some changes to our local git and pushing them to GitHub.



Example

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
<link rel="stylesheet" href="bluestyle.css">
</head>
<body>

<h1>Hello world!</h1>
<div><img src="img_hello_world.jpg" alt="Hello World from Space"
style="width:100%;max-width:640px"></div>
<p>This is the first file in my new Git Repo.</p>
<p>This line is here to show how merging works.</p>
<div><img src="img_hello_git.jpg" alt="Hello Git"
style="width:100%;max-width:640px"></div>

</body>
</html>
```

Commit the changes:



Example

```
[user@localhost] $    git commit -a -m "Updated index.html. Resized image"
                      [master e7de78f] Updated index.html. Resized image
                       1 file changed, 1 insertion(+), 1 deletion(-)
```

And check the status:

Example

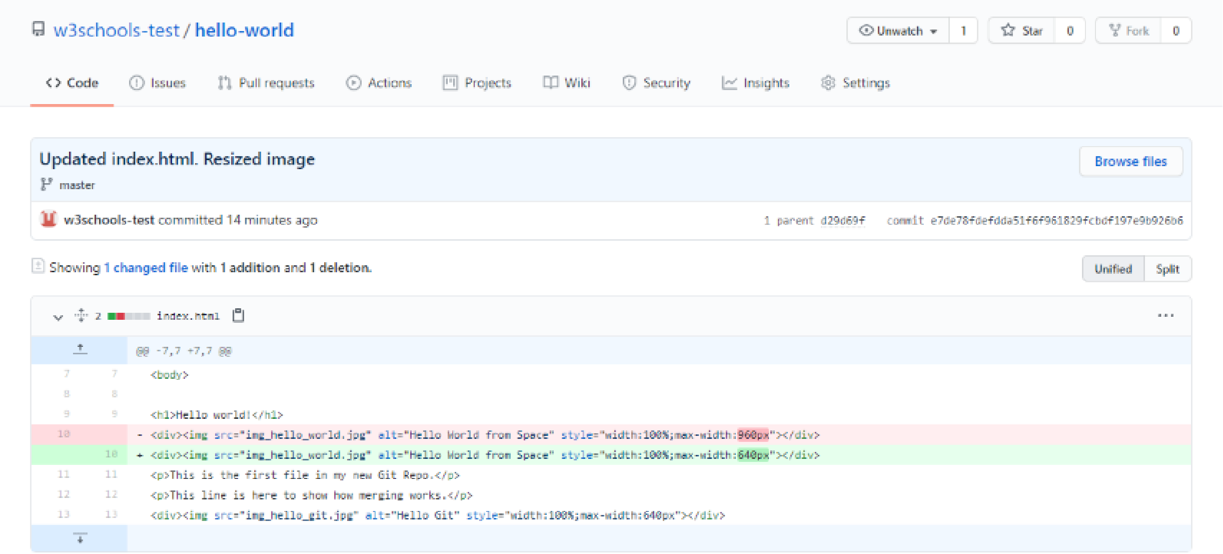
```
[user@localhost] $    git status
                      On branch master
                      Your branch is ahead of 'origin/master' by 1 commit.
                        (use "git push" to publish your local commits)

                      nothing to commit, working tree clean
```

Now push our changes to our remote origin:

Example

```
[user@localhost] $    git push origin
                      Enumerating objects: 9, done.
                      Counting objects: 100% (8/8), done.
                      Delta compression using up to 16 threads
                      Compressing objects: 100% (5/5), done.
                      Writing objects: 100% (5/5), 578 bytes | 578.00 KiB/s, done.
                      Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
                      remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
                      To https://github.com/w3schools-test/hello-world.git
```

Go to GitHub, and confirm that the repository has a new commit:



Updated index.html. Resized image

w3schools-test committed 14 minutes ago 1 parent d29d69f commit e7de78fdefdda51f6f961829fcbdf197e9b926b6

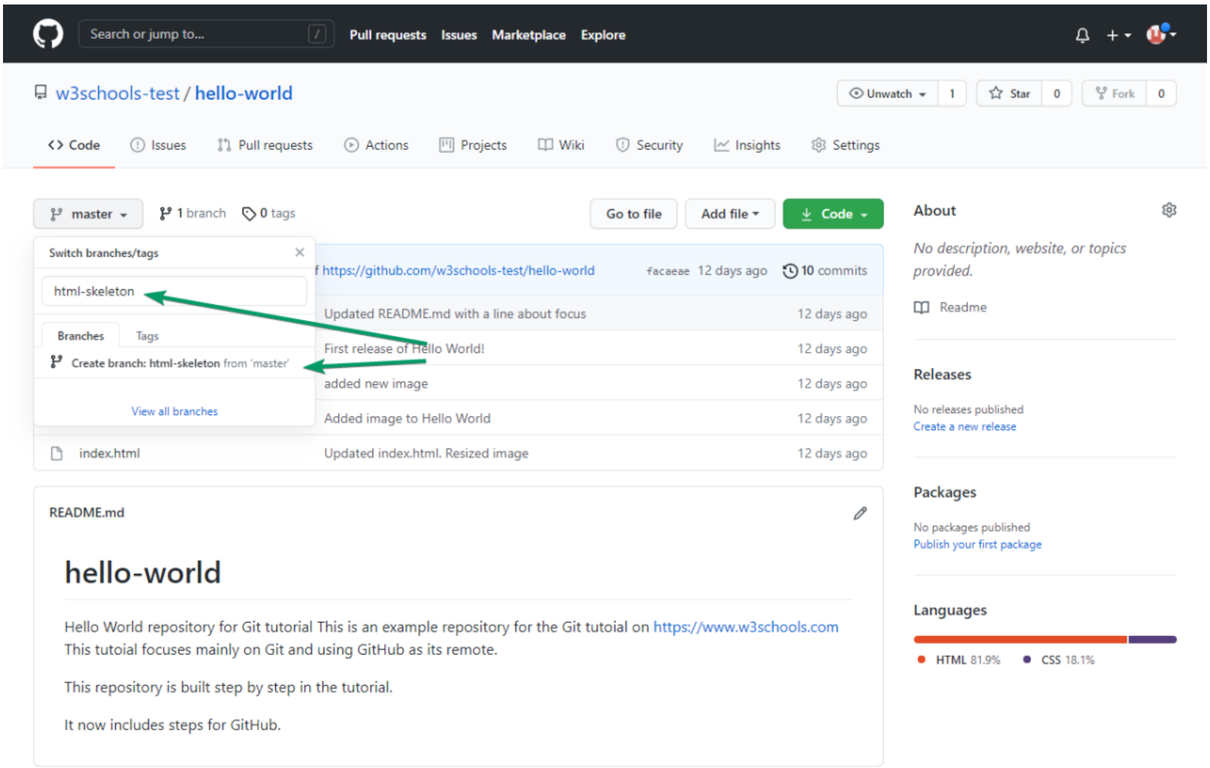
Showing 1 changed file with 1 addition and 1 deletion.

```
2 index.html
@@ -7,7 +7,7 @@
 7   7   <body>
 8   8
 9   9   <h1>Hello world!</h1>
10     - <div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:960px"></div>
    10 + <div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
11  11   <p>This is the first file in my new Git Repo.</p>
12  12   <p>This line is here to show how merging works.</p>
13  13   <div><img src="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>
```

Create a New Branch on GitHub

On GitHub, access your repository and click the "master" branch button.

There you can create a new Branch. Type in a descriptive name, and click Create branch:



html-skeleton

Create branch: html-skeleton from 'master'

https://github.com/w3schools-test/hello-world facaeae 12 days ago 10 commits

	Updated README.md with a line about focus	12 days ago
	First release of Hello World!	12 days ago
	added new image	12 days ago
	Added image to Hello World	12 days ago
index.html	Updated index.html. Resized image	12 days ago

No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

README.md

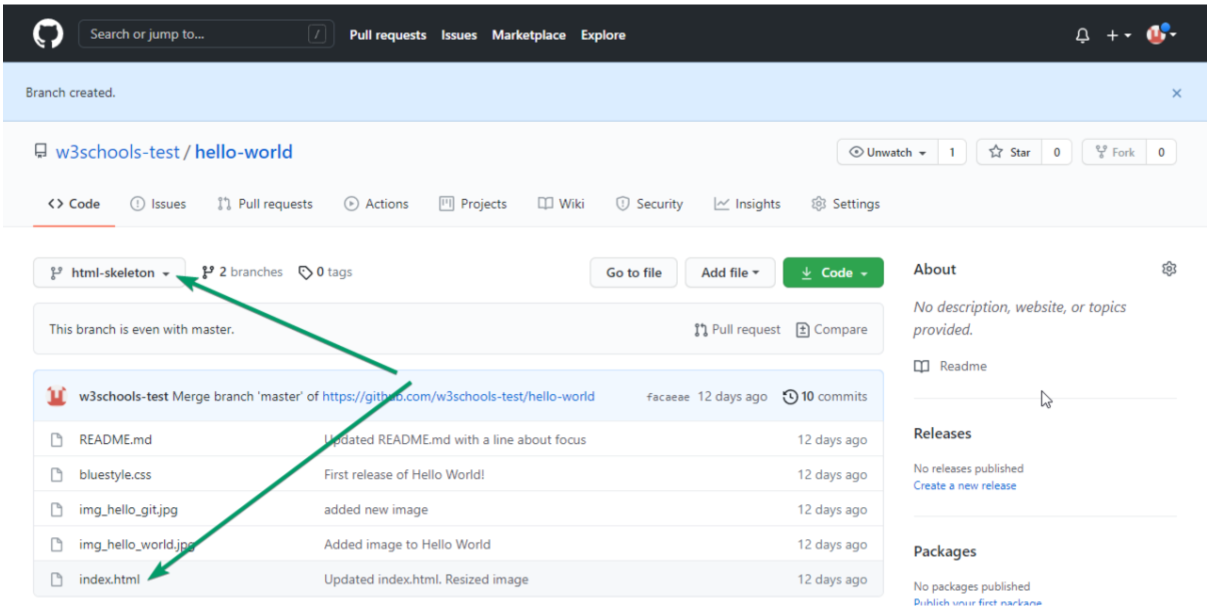
hello-world

Hello World repository for Git tutorial This is an example repository for the Git tutoial on https://www.w3schools.com This tutoial focuses mainly on Git and using GitHub as its remote.

This repository is built step by step in the tutorial.

It now includes steps for GitHub.

The branch should now be created and active. You can confirm which branch you are working on by looking at the branch button. See that it now says "html-skeleton" instead of "main"?



Branch created.

This branch is even with master.

w3schools-test Merge branch 'master' of https://github.com/w3schools-test/hello-world facaeae 12 days ago 10 commits

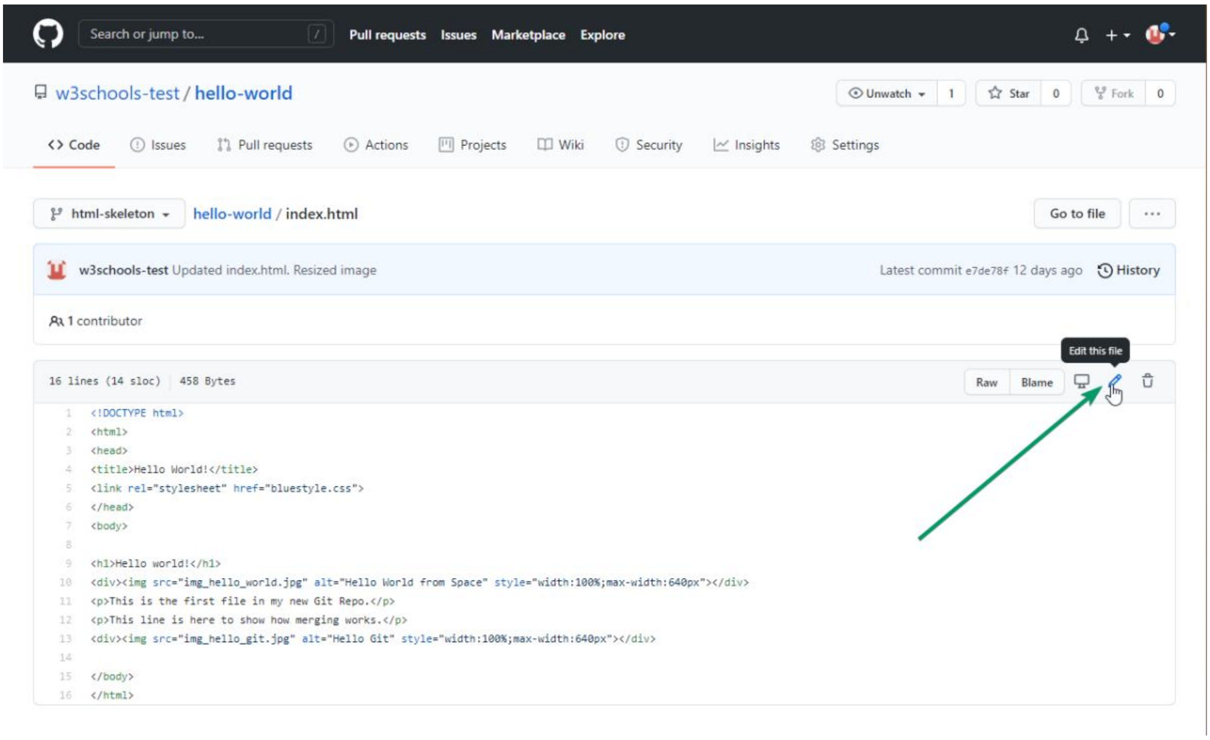
README.md	Updated README.md with a line about focus	12 days ago
bluestyle.css	First release of Hello World!	12 days ago
img_hello_git.jpg	added new image	12 days ago
img_hello_world.jpg	Added image to Hello World	12 days ago
index.html	Updated index.html. Resized image	12 days ago

No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

Start working on an existing file in this branch. Click the "index.html" file and start editing:



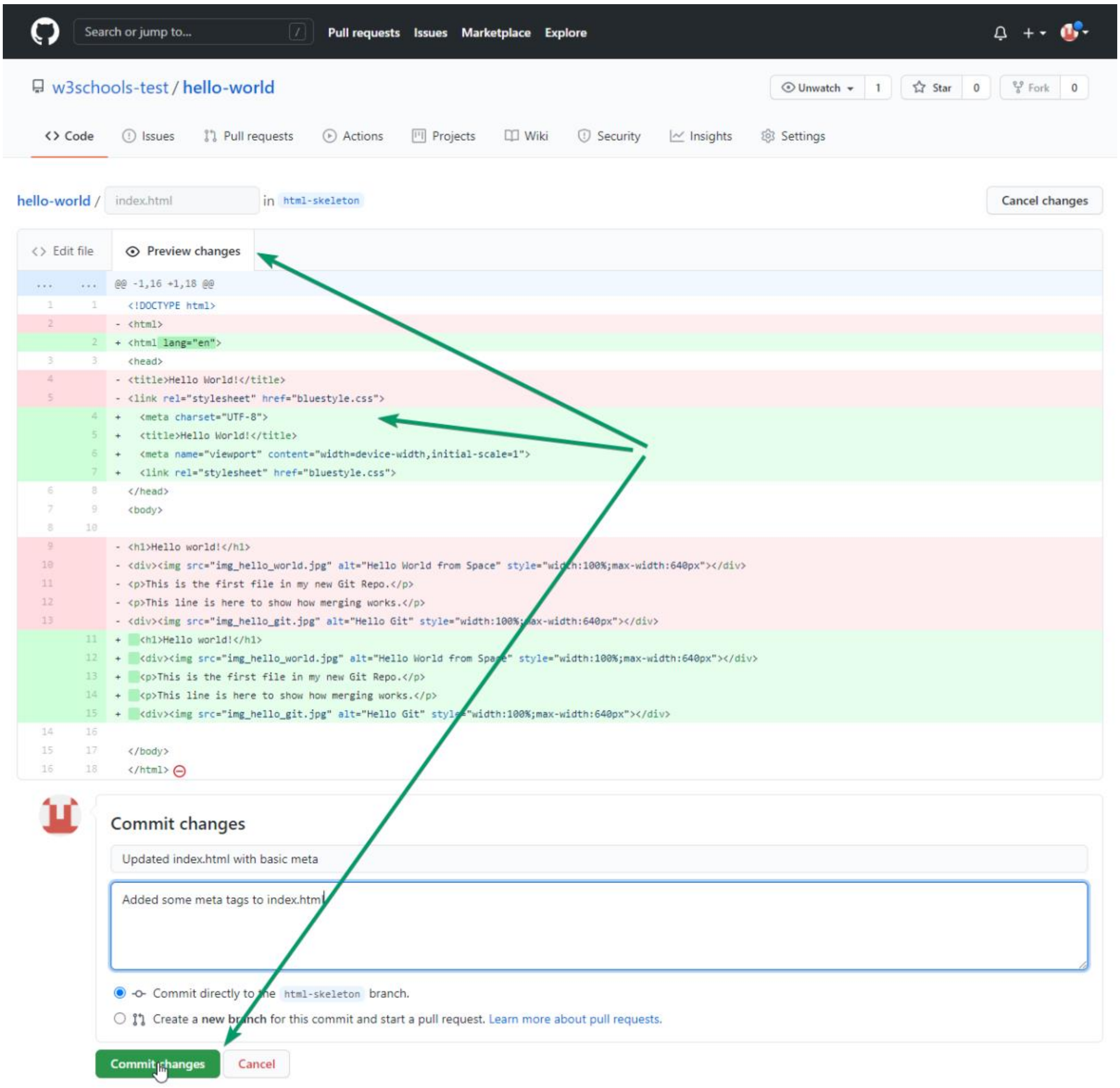
w3schools-test Updated index.html. Resized image Latest commit e7de78f 12 days ago

1 contributor

16 lines (14 sloc) | 458 Bytes

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
<link rel="stylesheet" href="bluestyle.css">
</head>
<body>

<h1>Hello world!</h1>
<div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
<p>This is the first file in my new Git Repo.</p>
<p>This line is here to show how merging works.</p>
<div><img src="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>

</body>
</html>
```

After you have finished editing the file, you can click the "Preview changes" tab to see the changes you made highlighted:



```
@@ -1,16 +1,18 @@
  <!DOCTYPE html>
- <html>
+ <html lang="en">
  <head>
- <title>Hello World!</title>
- <link rel="stylesheet" href="bluestyle.css">
+   <meta charset="UTF-8">
+   <title>Hello World!</title>
+   <meta name="viewport" content="width=device-width,initial-scale=1">
+   <link rel="stylesheet" href="bluestyle.css">
  </head>
  <body>

- <h1>Hello world!</h1>
- <div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
- <p>This is the first file in my new Git Repo.</p>
- <p>This line is here to show how merging works.</p>
- <div><img src="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>
+   <h1>Hello world!</h1>
+   <div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
+   <p>This is the first file in my new Git Repo.</p>
+   <p>This line is here to show how merging works.</p>
+   <div><img src="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>

  </body>
  </html>
```

Commit changes

Updated index.html with basic meta

Added some meta tags to index.html

● Commit directly to the html-skeleton branch.

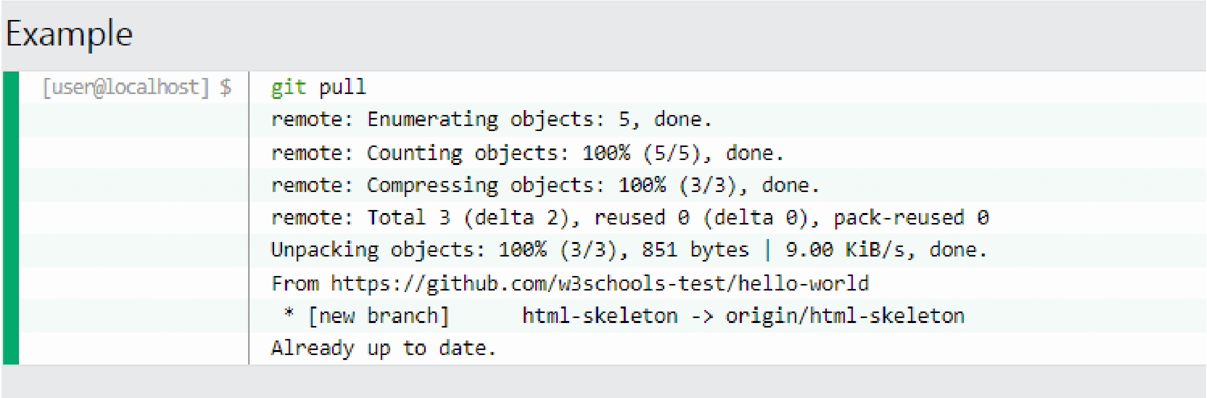
○ Create a new branch for this commit and start a pull request. Learn more about pull requests.

If you are happy with the change, add a comment that explains what you did, and click Commit changes.

You now have a new `branch` on GitHub, updated with some changes!

Pulling a Branch from GitHub

Now to continue working on our new branch in our local Git. Lets pull from our GitHub repository again so that our code is up-to-date

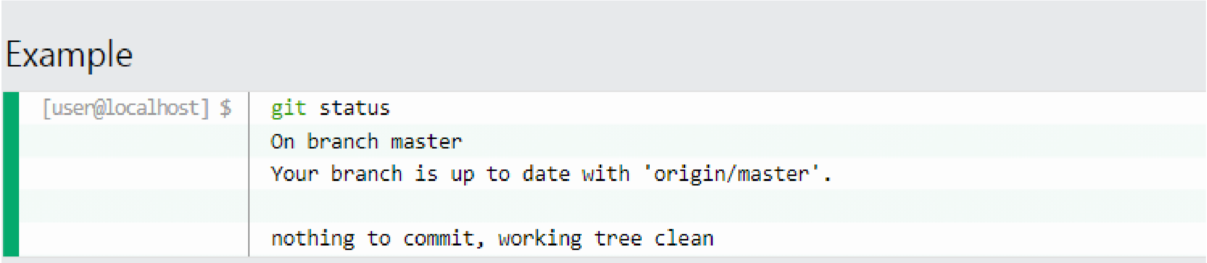


Example

```
[user@localhost] $   git pull
                     remote: Enumerating objects: 5, done.
                     remote: Counting objects: 100% (5/5), done.
                     remote: Compressing objects: 100% (3/3), done.
                     remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
                     Unpacking objects: 100% (3/3), 851 bytes | 9.00 KiB/s, done.
                     From https://github.com/w3schools-test/hello-world
                      * [new branch]      html-skeleton -> origin/html-skeleton
                     Already up to date.
```

Now our main `branch` is up todate. And we can see that there is a new `branch` available on GitHub.

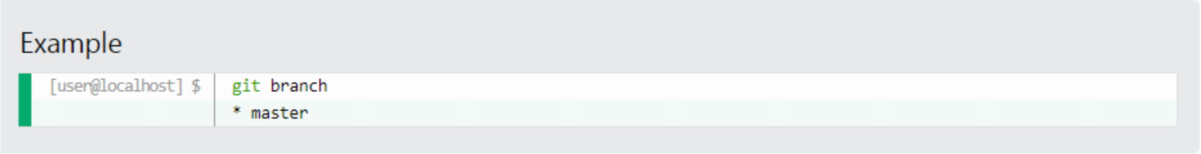
Do a quick `status` check:



Example

```
[user@localhost] $   git status
                     On branch master
                     Your branch is up to date with 'origin/master'.

                     nothing to commit, working tree clean
```

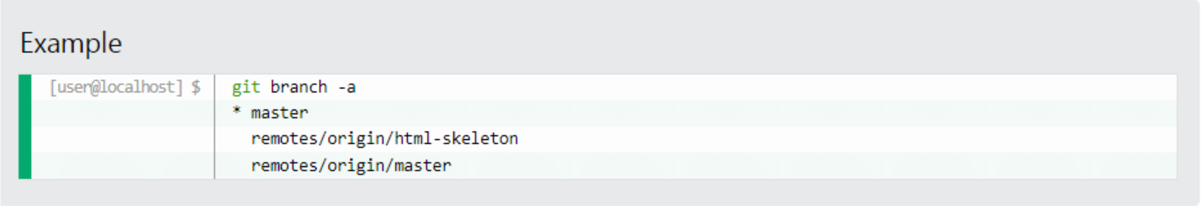
And confirm which branches we have, and where we are working at the moment:



Example

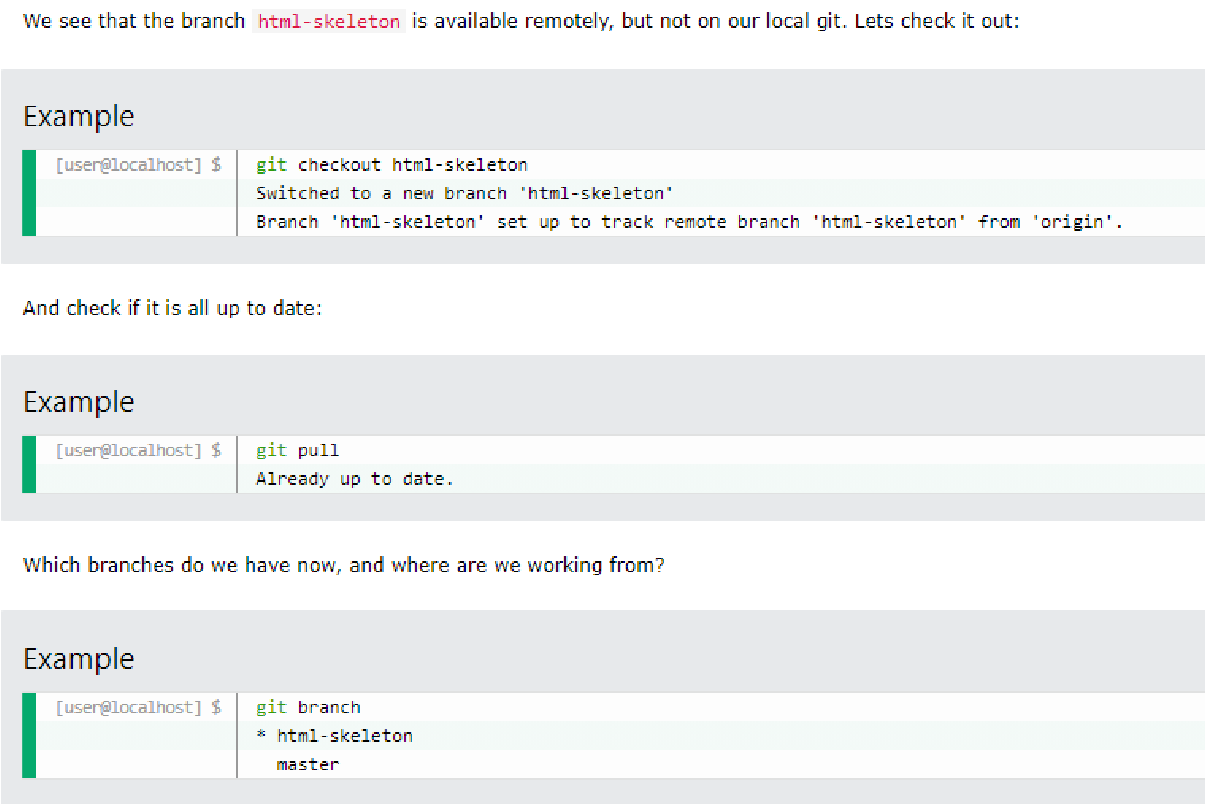
```
[user@localhost] $   git branch
                     * master
```

So, we do not have the new `branch` on our local Git. But we know it is available on GitHub. So we can use the `-a` option to see all local and remote branches:



Example

```
[user@localhost] $   git branch -a
                     * master
                       remotes/origin/html-skeleton
                       remotes/origin/master
```



We see that the branch `html-skeleton` is available remotely, but not on our local git. Lets check it out:

Example

```
[user@localhost] $ git checkout html-skeleton
Switched to a new branch 'html-skeleton'
Branch 'html-skeleton' set up to track remote branch 'html-skeleton' from 'origin'.
```

And check if it is all up to date:

Example

```
[user@localhost] $ git pull
Already up to date.
```

Which branches do we have now, and where are we working from?

Example

```
[user@localhost] $ git branch
* html-skeleton
  master
```

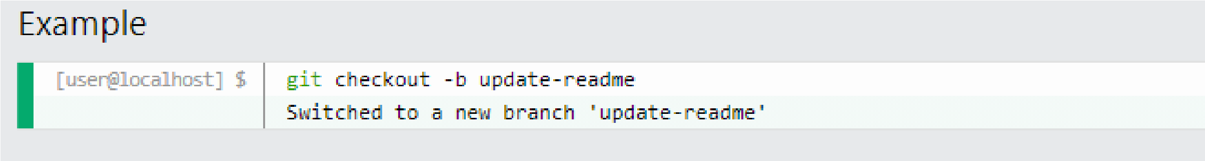
Now, open your favourite editor and confirm that the changes from the GitHub branch carried over.

That is how you pull a GitHub branch to your local Git.

Push a Branch to GitHub

Let's try to create a new local branch, and push that to GitHub.

Start by creating a branch, like we did earlier:

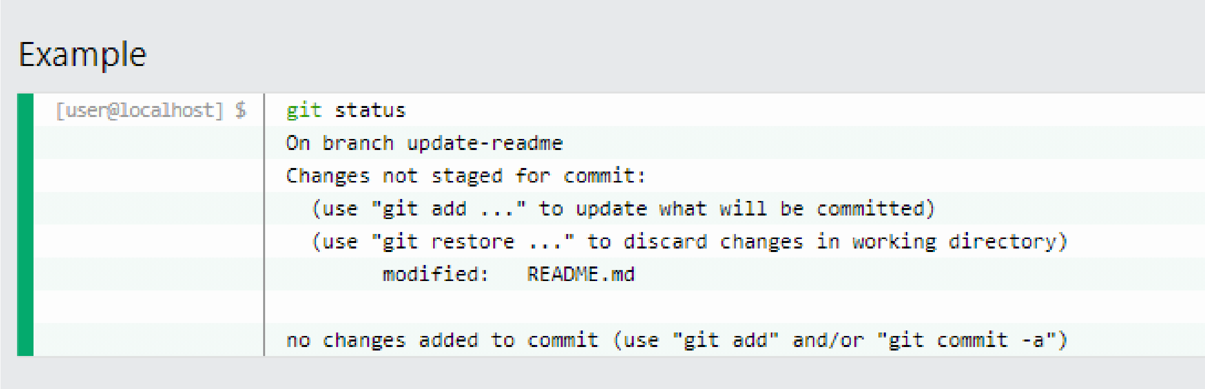


Example

```
[user@localhost] $ git checkout -b update-readme
Switched to a new branch 'update-readme'
```

And we make some changes to the README.md file. Just add a new line.

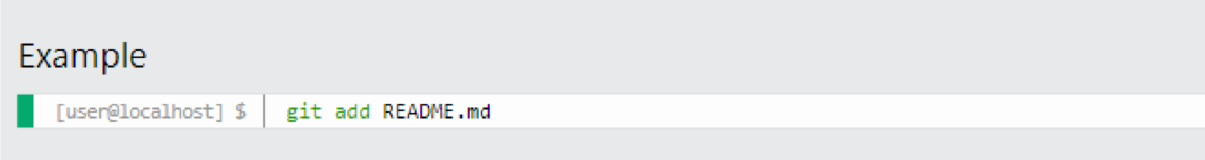
So now we check the `status` of the current branch.



Example

```
[user@localhost] $ git status
On branch update-readme
Changes not staged for commit:
  (use "git add ..." to update what will be committed)
  (use "git restore ..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

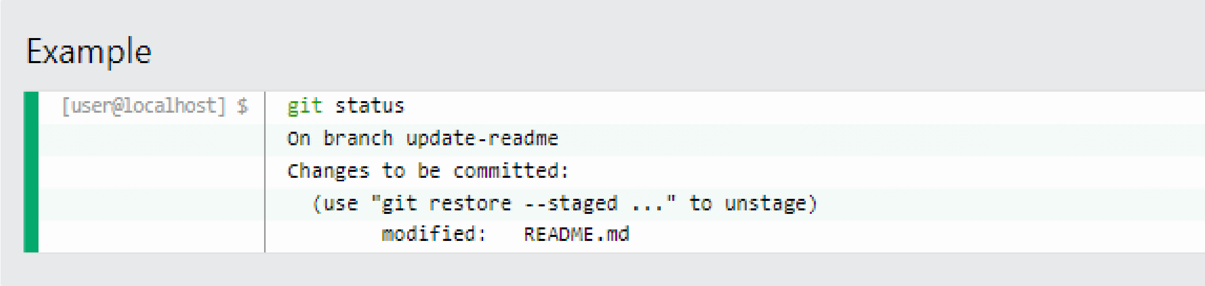
We see that `README.md` is modified but not added to the Staging Environment:



Example

```
[user@localhost] $ git add README.md
```

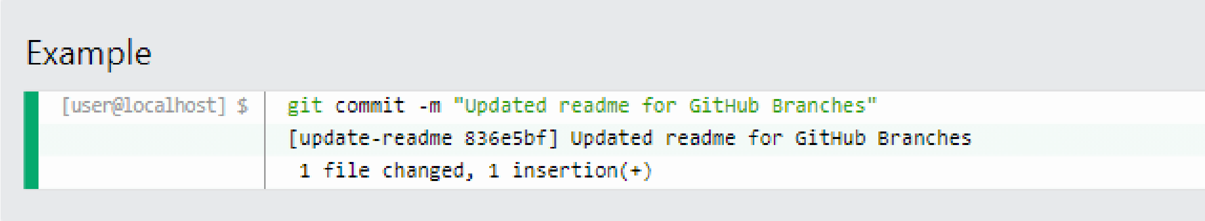
Check the `status` of the branch:



Example

```
[user@localhost] $ git status
On branch update-readme
Changes to be committed:
  (use "git restore --staged ..." to unstage)
        modified:   README.md
```

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



Example

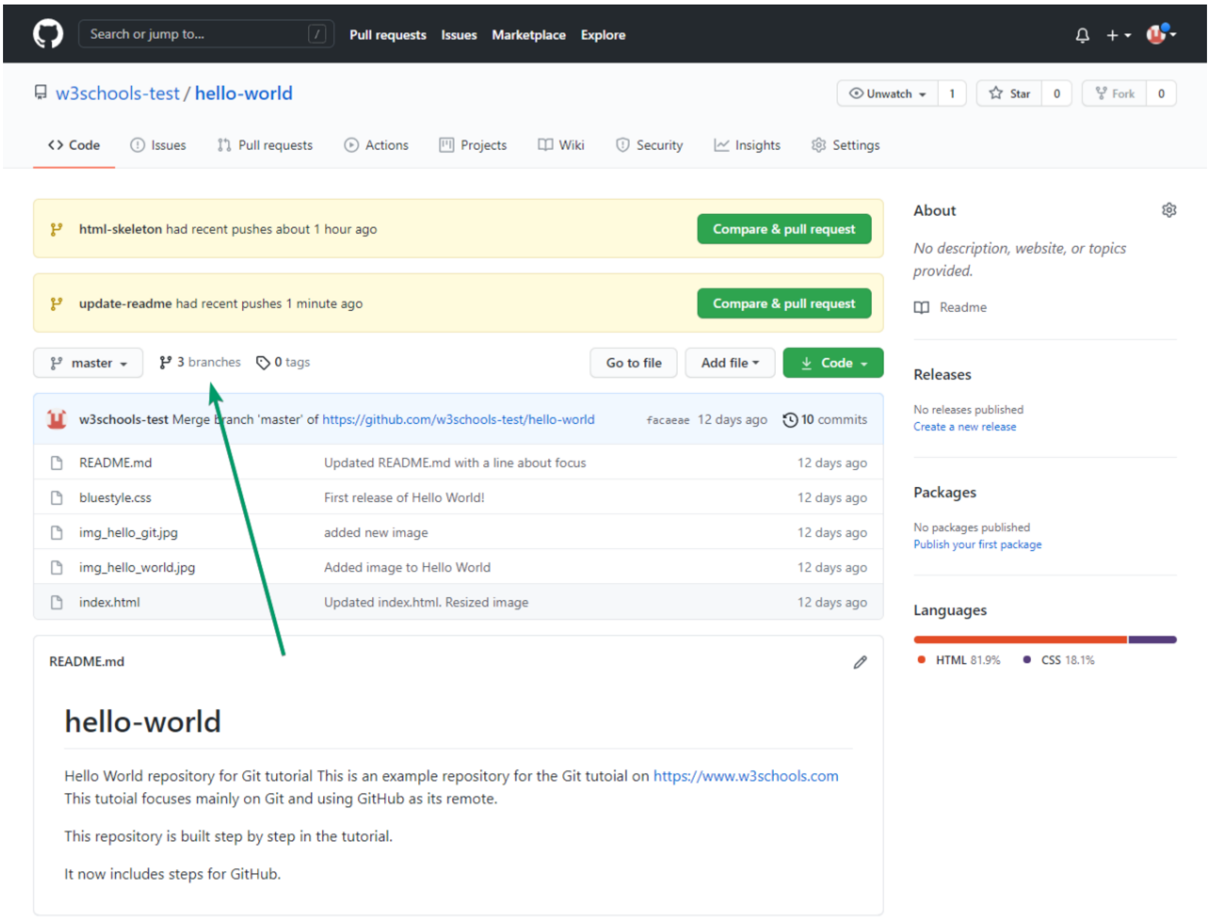
```
[user@localhost] $ git commit -m "Updated readme for GitHub Branches"
[update-readme 836e5bf] Updated readme for GitHub Branches
 1 file changed, 1 insertion(+)
```

Now `push` the `branch` from our local Git repository, to GitHub, where everyone can see the changes:

Example

```
[user@localhost] $   git push origin update-readme
                     Enumerating objects: 5, done.
                     Counting objects: 100% (5/5), done.
                     Delta compression using up to 16 threads
                     Compressing objects: 100% (3/3), done.
                     Writing objects: 100% (3/3), 366 bytes | 366.00 KiB/s, done.
                     Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
                     remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
                     remote:
                     remote: Create a pull request for 'update-readme' on GitHub by visiting:
                     remote:      https://github.com/w3schools-test/hello-world/pull/new/update-readme
                     remote:
                     To https://github.com/w3schools-test/hello-world.git
                      * [new branch]      update-readme -> update-readme
```

Go to GitHub, and confirm that the repository has a new `branch`



html-skeleton had recent pushes about 1 hour ago

update-readme had recent pushes 1 minute ago

w3schools-test Merge branch 'master' of https://github.com/w3schools-test/hello-world facaeae 12 days ago 10 commits

README.md	Updated README.md with a line about focus	12 days ago
bluestyle.css	First release of Hello World!	12 days ago
img_hello_git.jpg	added new image	12 days ago
img_hello_world.jpg	Added image to Hello World	12 days ago
index.html	Updated index.html. Resized image	12 days ago

No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

README.md

hello-world

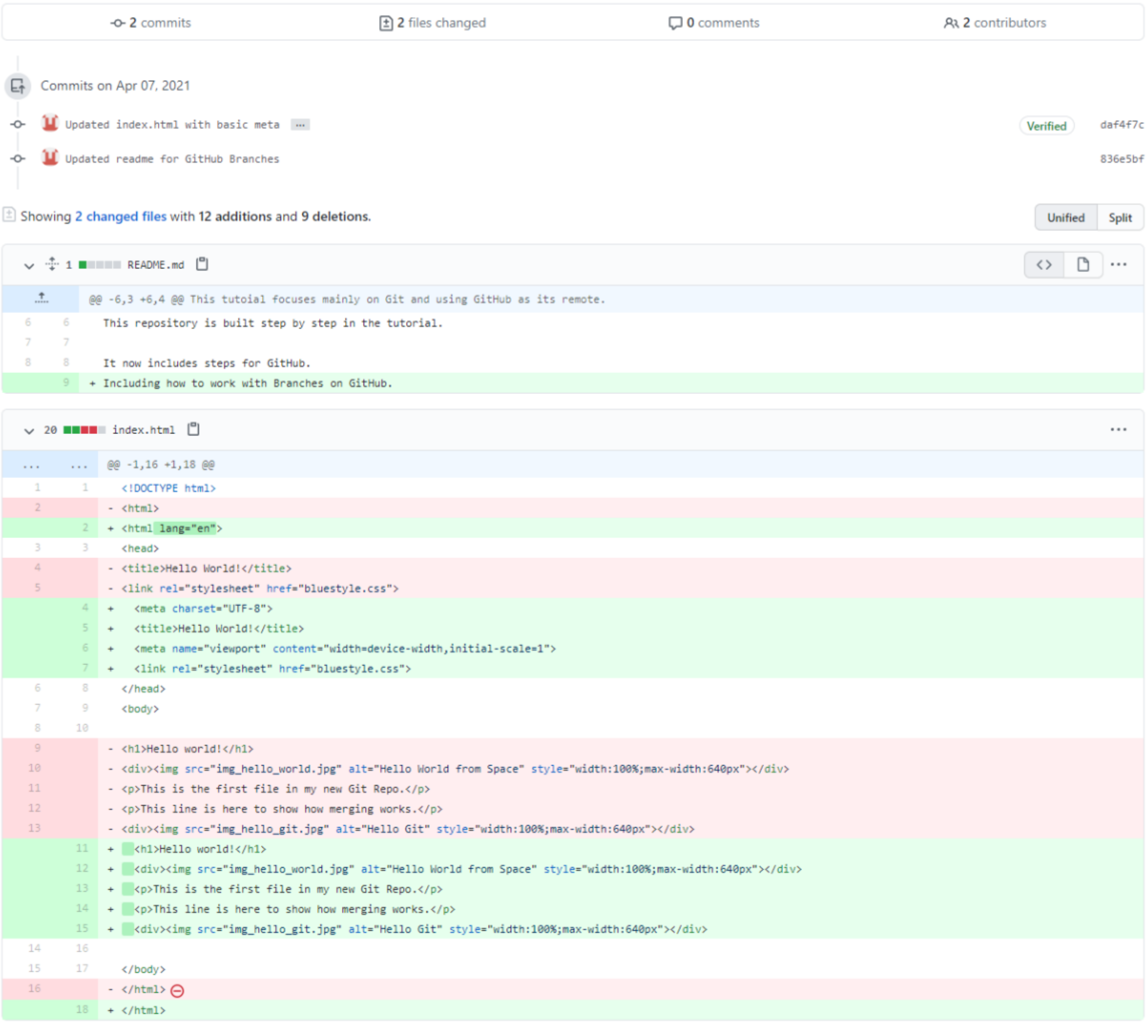
Hello World repository for Git tutorial This is an example repository for the Git tutoial on https://www.w3schools.com This tutoial focuses mainly on Git and using GitHub as its remote.

This repository is built step by step in the tutorial.

It now includes steps for GitHub.

In GitHub, we can now see the changes and `merge` them into the master `branch` if we approve it.

If you click the "Compare & pull request", you can go through the changes made and new files added:



Commits on Apr 07, 2021

```
Updated index.html with basic meta       daf4f7c
Updated readme for GitHub Branches       836e5bf
```

Showing 2 changed files with 12 additions and 9 deletions.

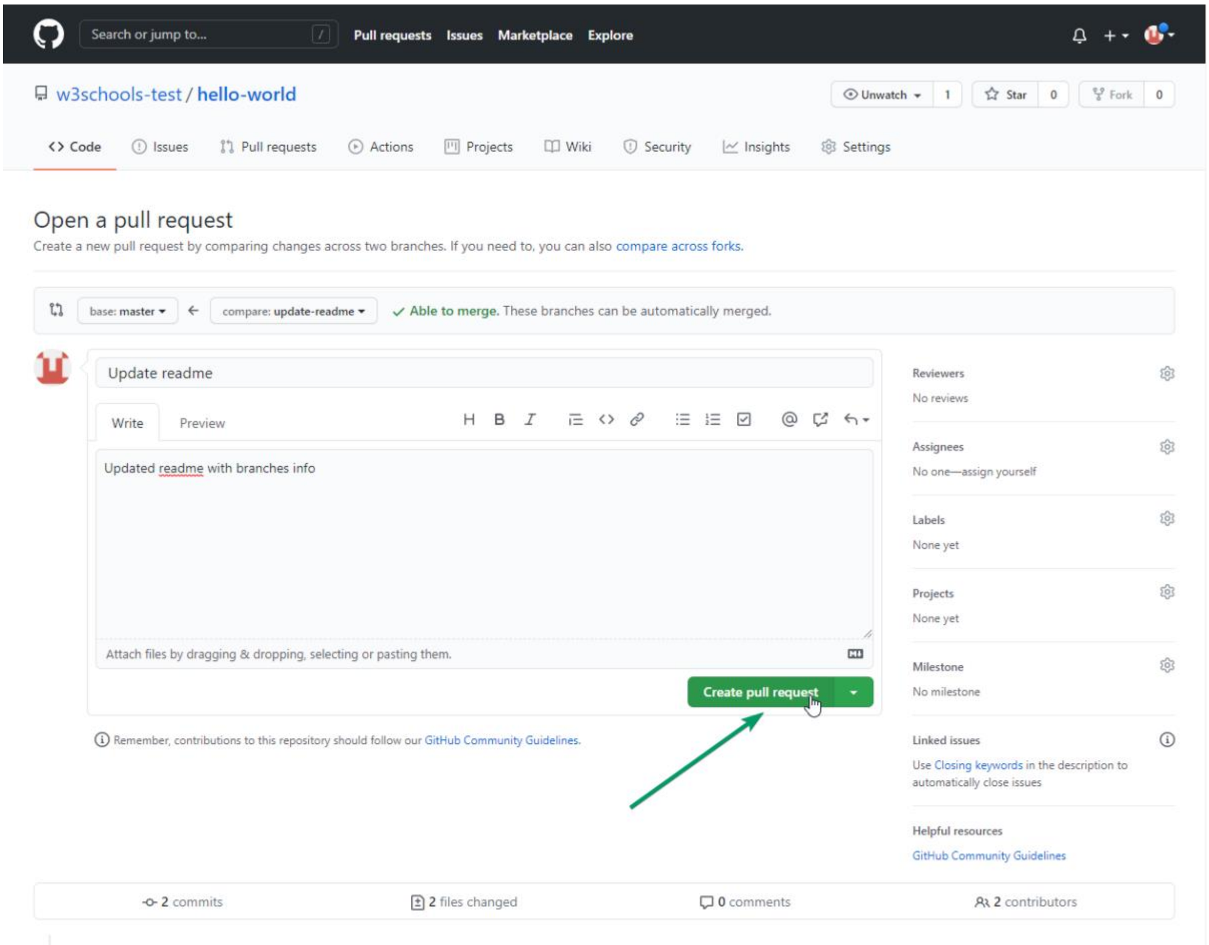
1 README.md

```
@@ -6,3 +6,4 @@ This tutoial focuses mainly on Git and using GitHub as its remote.
  This repository is built step by step in the tutorial.

  It now includes steps for GitHub.
+ Including how to work with Branches on GitHub.
```

20 index.html

```
@@ -1,16 +1,18 @@
  <!DOCTYPE html>
- <html>
+ <html lang="en">
  <head>
- <title>Hello World!</title>
- <link rel="stylesheet" href="bluestyle.css">
+   <meta charset="UTF-8">
+   <title>Hello World!</title>
+   <meta name="viewport" content="width=device-width,initial-scale=1">
+   <link rel="stylesheet" href="bluestyle.css">
  </head>
  <body>

- <h1>Hello world!</h1>
- <div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
- <p>This is the first file in my new Git Repo.</p>
- <p>This line is here to show how merging works.</p>
- <div><img src="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>
+  <h1>Hello world!</h1>
+  <div><img src="img_hello_world.jpg" alt="Hello World from Space" style="width:100%;max-width:640px"></div>
+  <p>This is the first file in my new Git Repo.</p>
+  <p>This line is here to show how merging works.</p>
+  <div><img src="img_hello_git.jpg" alt="Hello Git" style="width:100%;max-width:640px"></div>

  </body>
- </html>
+ </html>
```

If the changes look good, you can go forward, creating a `pull request`:



w3schools-test / hello-world

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ← compare: update-readme ✓ Able to merge. These branches can be automatically merged.

Update readme

Updated readme with branches info

Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

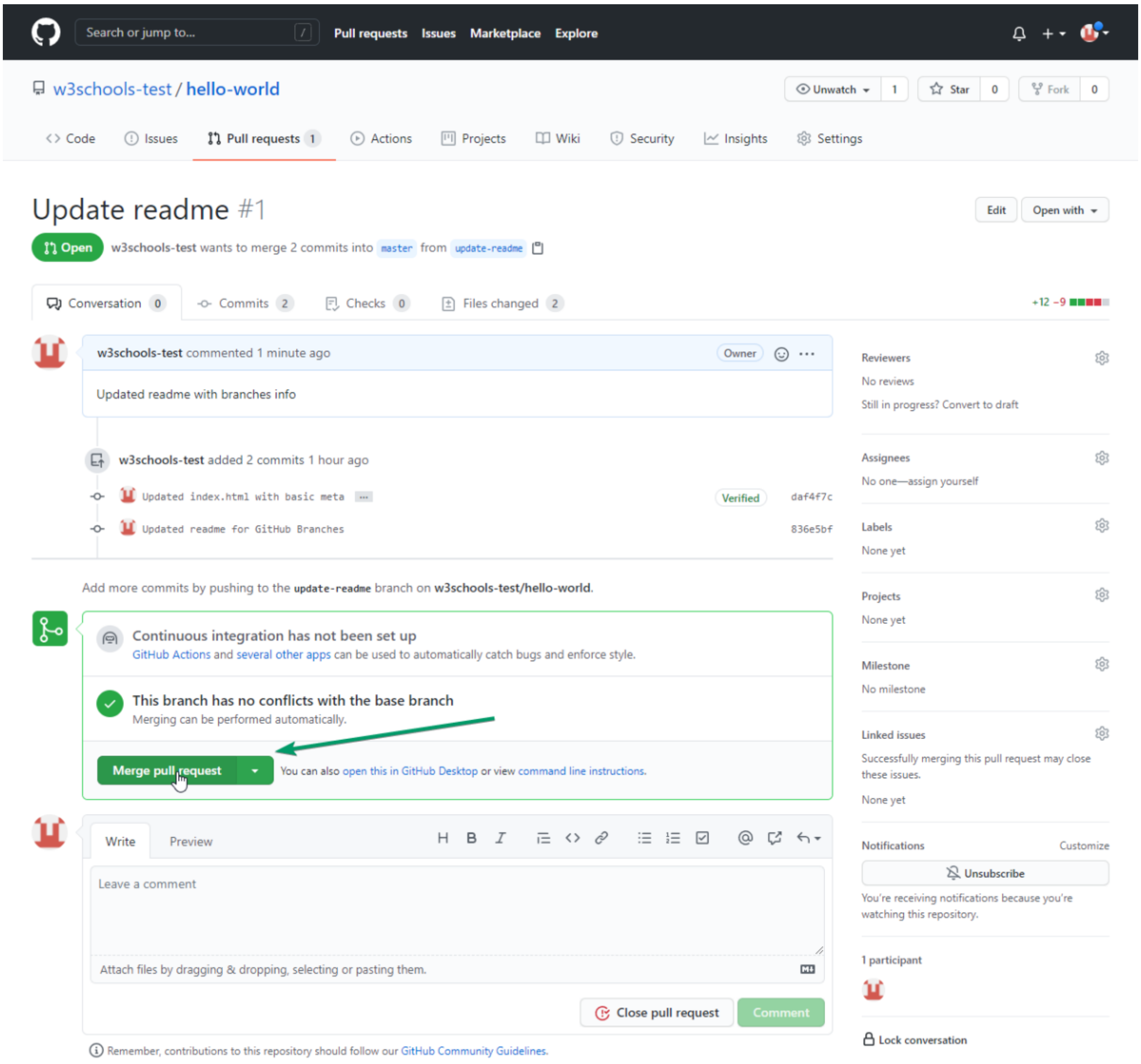
Projects
None yet

Milestone
No milestone

Linked issues
Use Closing keywords in the description to automatically close issues

Helpful resources
GitHub Community Guidelines

2 commits | 2 files changed | 0 comments | 2 contributors



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

w3schools-test / hello-world

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Update readme #1

Edit Open with

Open w3schools-test wants to merge 2 commits into `master` from `update-readme`

Conversation 0 Commits 2 Checks 0 Files changed 2 +12 −9

w3schools-test commented 1 minute ago Owner

Updated readme with branches info

w3schools-test added 2 commits 1 hour ago

`Updated index.html with basic meta` Verified `daf4f7c`

`Updated readme for GitHub Branches` `836e5bf`

Add more commits by pushing to the `update-readme` branch on w3schools-test/hello-world.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
None yet

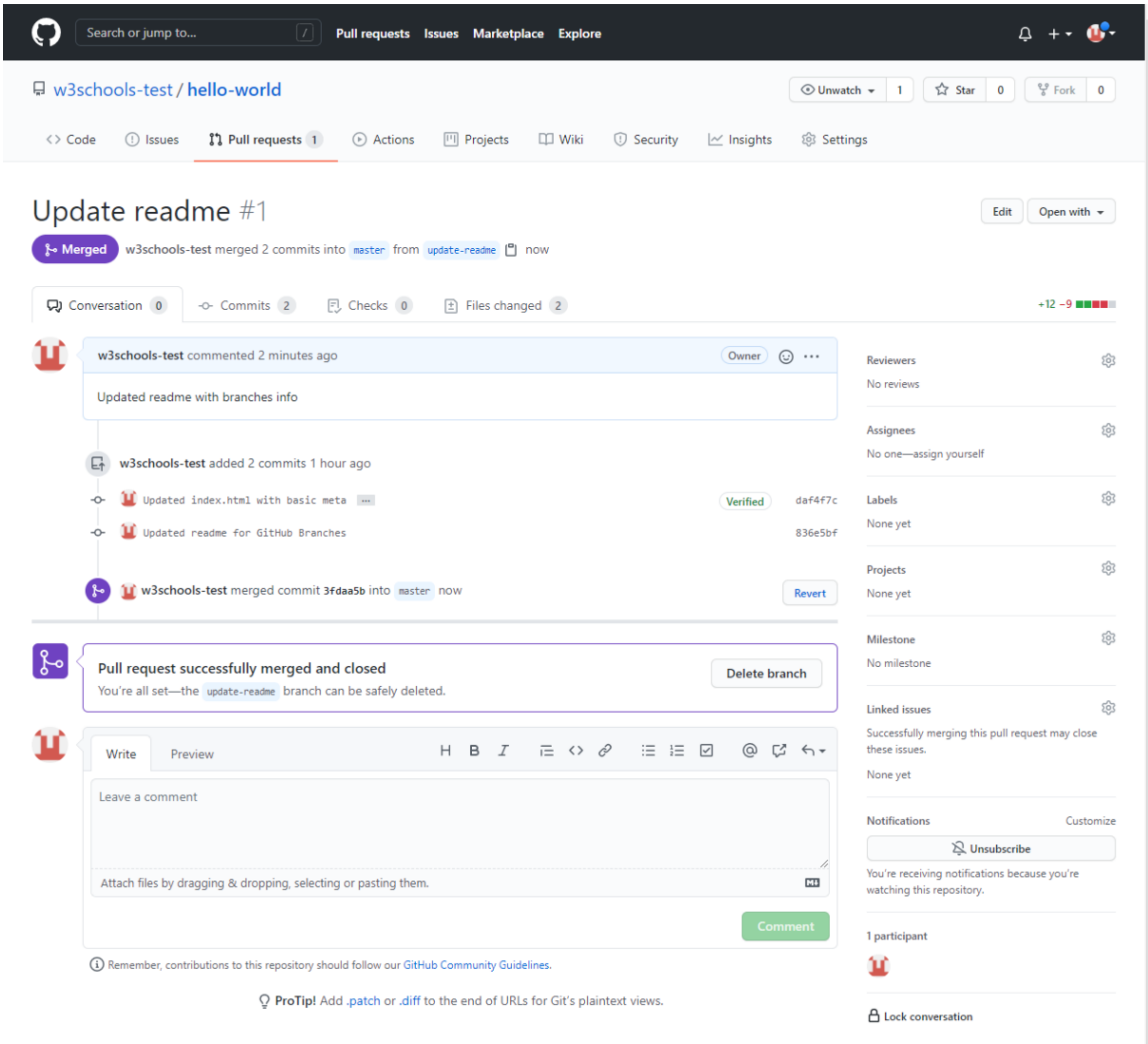
Notifications Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

1 participant

Lock conversation

The pull request will record the changes, which means you can go through them later to figure out the changes made.

The result should be something like this:



Update readme #1

w3schools-test merged 2 commits into master from update-readme now

w3schools-test commented 2 minutes ago

Updated readme with branches info

w3schools-test added 2 commits 1 hour ago

```
Updated index.html with basic meta   daf4f7c
Updated readme for GitHub Branches   836e5bf
```

w3schools-test merged commit 3fdaa5b into master now

Pull request successfully merged and closed

You're all set—the update-readme branch can be safely deleted.

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.

ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

No reviews

No one—assign yourself

None yet

None yet

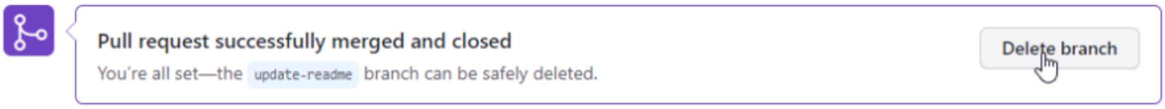
No milestone

Successfully merging this pull request may close these issues.

None yet

You're receiving notifications because you're watching this repository.

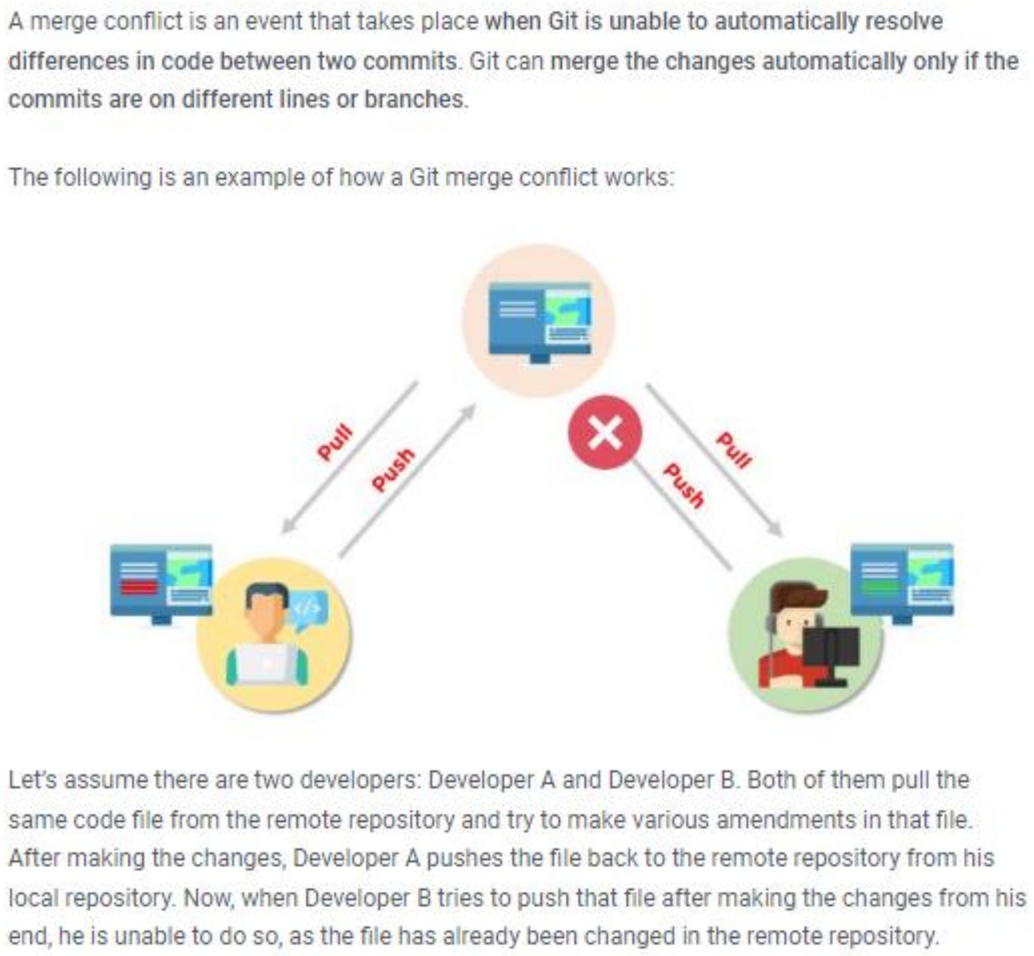
To keep the repo from getting overly complicated, you can delete the now unused branch by clicking "Delete branch".



Pull request successfully merged and closed

You're all set—the update-readme branch can be safely deleted.

Merge Conflict



A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

The following is an example of how a Git merge conflict works:

Let's assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.

Git Merge And Merge Conflict

Git Merge:

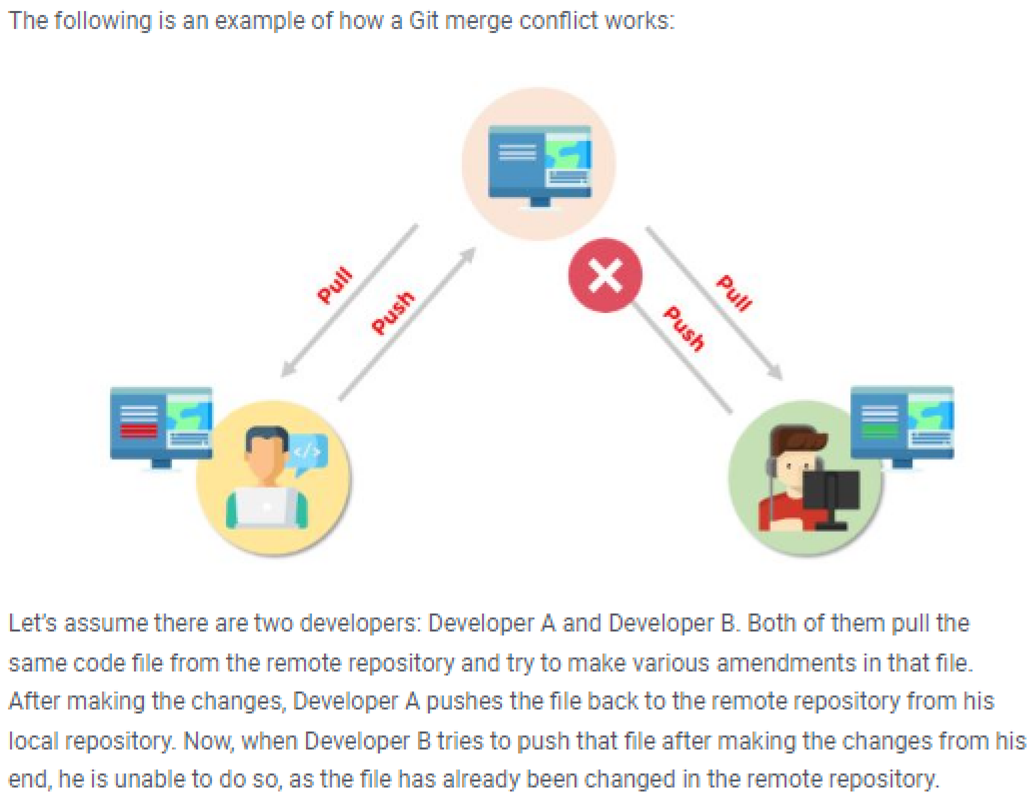
Git merge is a Git command used to combine changes from multiple branches into a single branch. This operation is used to bring changes from different branches into the current branch. The merge operation can be performed on any two branches within a repository, such as local and remote branches, or two local branches.

When merging two branches, Git will compare the changes made in each branch and combine them into a single branch. If there are conflicting changes, Git will mark them as a merge conflict, and the user will need to resolve these conflicts manually.

Git Merge Conflict:

A Git merge conflict occurs when two or more branches have made conflicting changes to the same line(s) of code. During the merge operation, Git will identify these conflicting changes and stop the merge process, requiring the user to resolve the conflicts manually.

To resolve a merge conflict, the user must open the file with the conflict, identify the conflicting changes, and choose which changes to keep or discard. After resolving the conflicts, the user must stage and commit the changes, completing the merge operation.



The following is an example of how a Git merge conflict works:

Let's assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.

Examples:

Merging two local branches:

```
$ git checkout branch1
$ git merge branch2
```

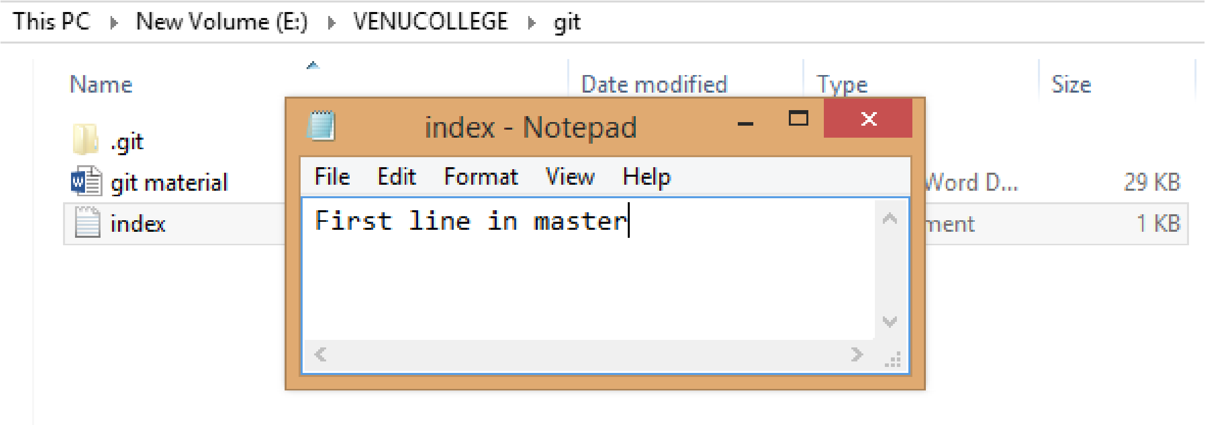
Merging a remote branch into a local branch:

```
$ git checkout local-branch
$ git merge origin/remote-branch
```

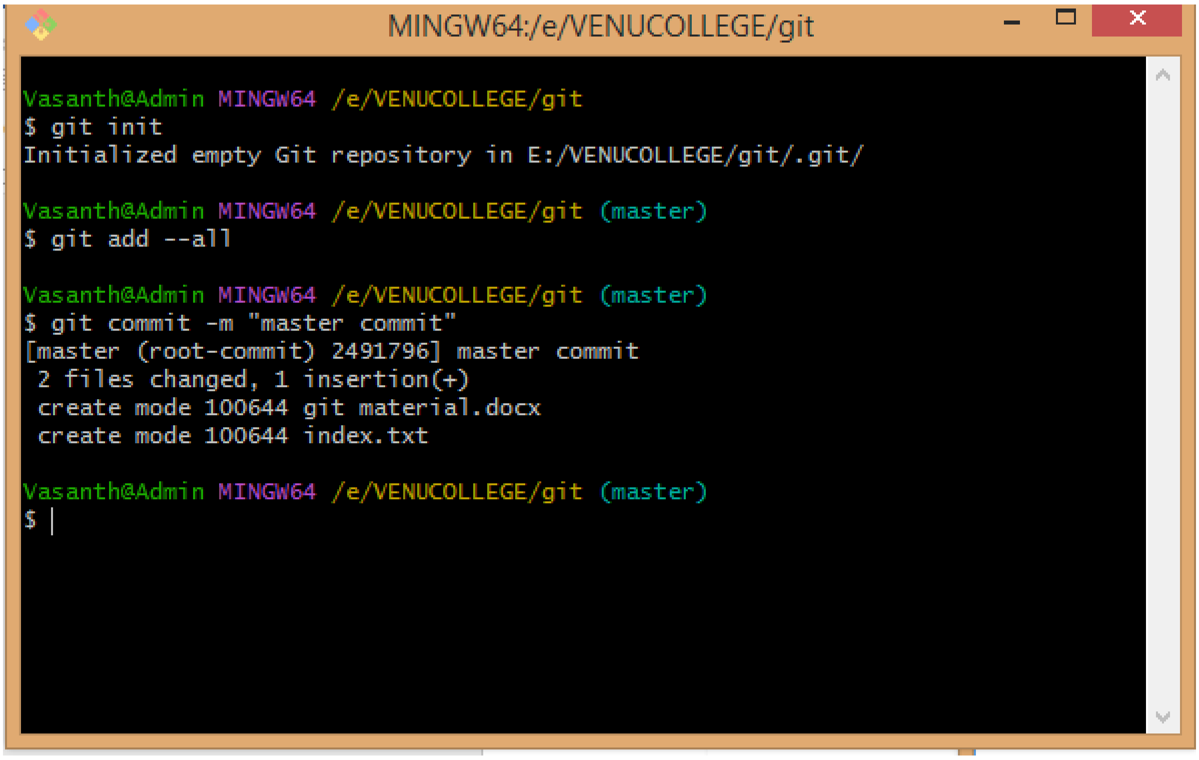
Resolving a merge conflict:

```
$ git checkout branch1
$ git merge branch2
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Step 1: Create a local repository in master and added two files in it.

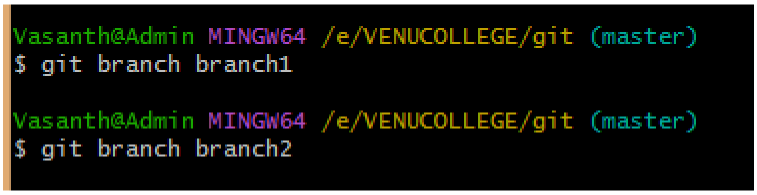


```
First line in master
```



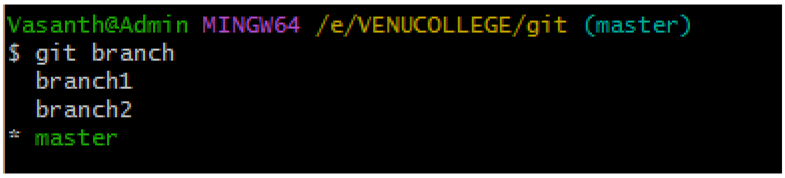
```
Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git
$ git init
Initialized empty Git repository in E:/VENUCOLLEGE/git/.git/

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git add --all

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git commit -m "master commit"
[master (root-commit) 2491796] master commit
 2 files changed, 1 insertion(+)
 create mode 100644 git material.docx
 create mode 100644 index.txt

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ 
```

Creating two branches



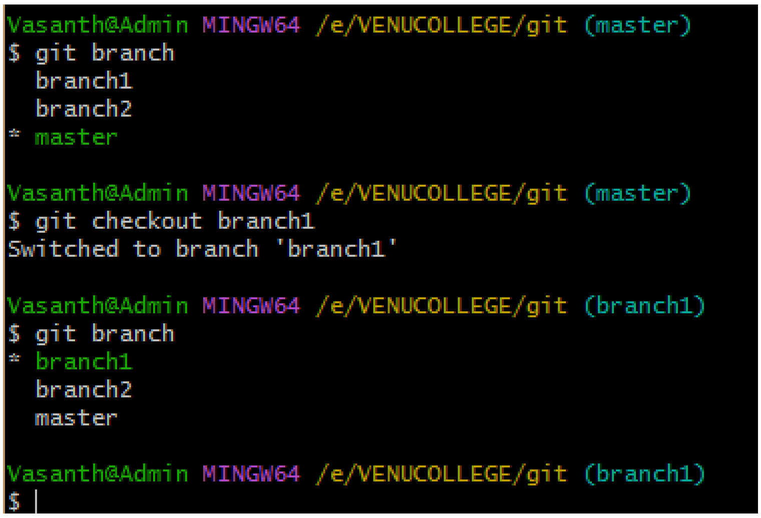
```
Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git branch branch1

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git branch branch2
```

Listing two branches

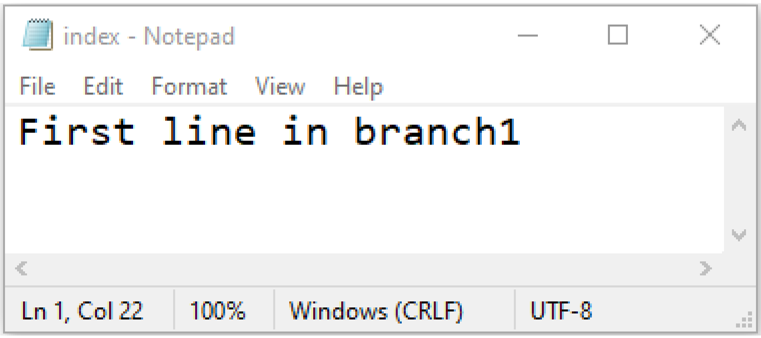


```
Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git branch
  branch1
  branch2
* master
```

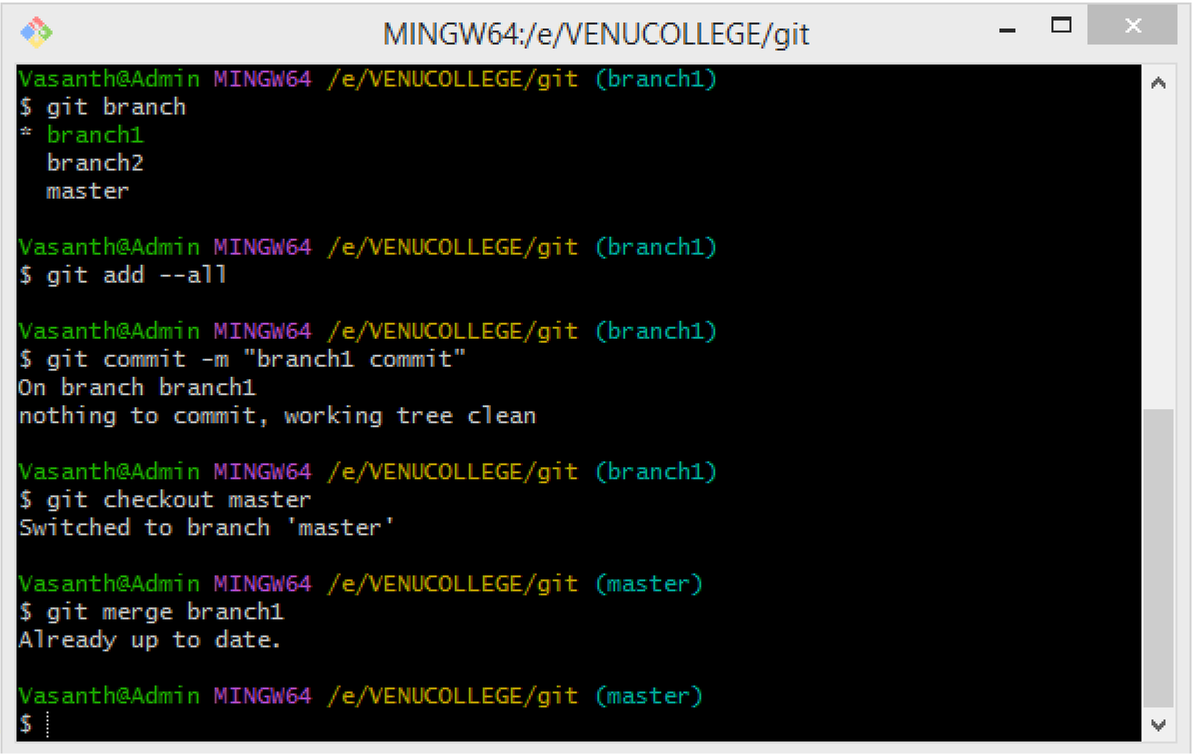
Moving to branch1 and making changes in index file & commit & merge with master



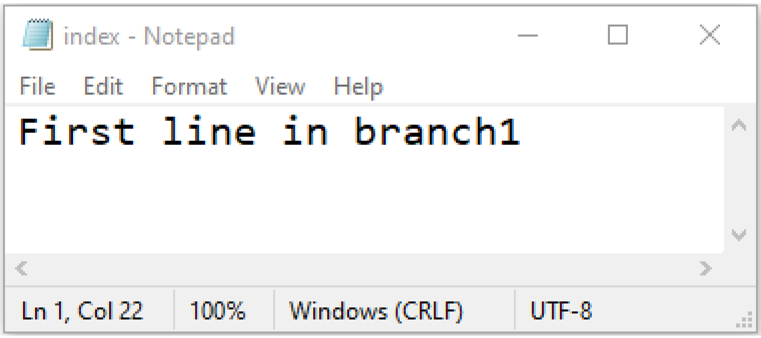
```
Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git branch
  branch1
  branch2
* master

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git checkout branch1
Switched to branch 'branch1'

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (branch1)
$ git branch
* branch1
  branch2
  master

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (branch1)
$ |
```



```
First line in branch1
```

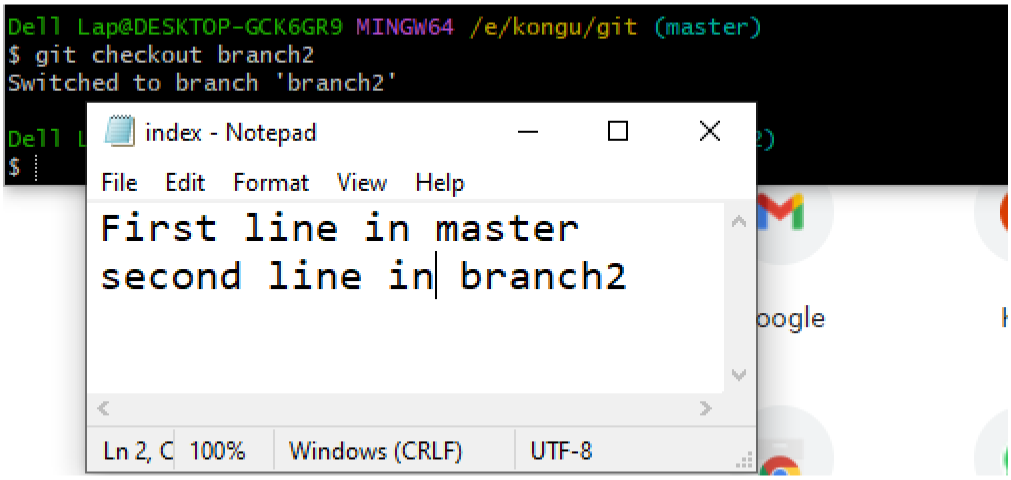


```
Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (branch1)
$ git branch
* branch1
  branch2
  master

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (branch1)
$ git add --all

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (branch1)
$ git commit -m "branch1 commit"
On branch branch1
nothing to commit, working tree clean

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (branch1)
$ git checkout master
Switched to branch 'master'

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$ git merge branch1
Already up to date.

Vasanth@Admin MINGW64 /e/VENUCOLLEGE/git (master)
$
```



```
First line in branch1
```

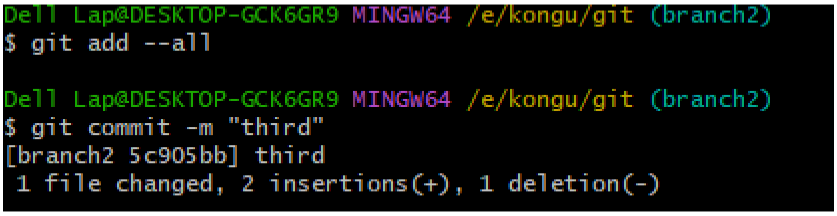
Checkout to branch2 and modify the index file



```
Dell Lap@DESKTOP-GCK6GR9 MINGW64 /e/kongu/git (master)
$ git checkout branch2
Switched to branch 'branch2'

Dell L
$
```

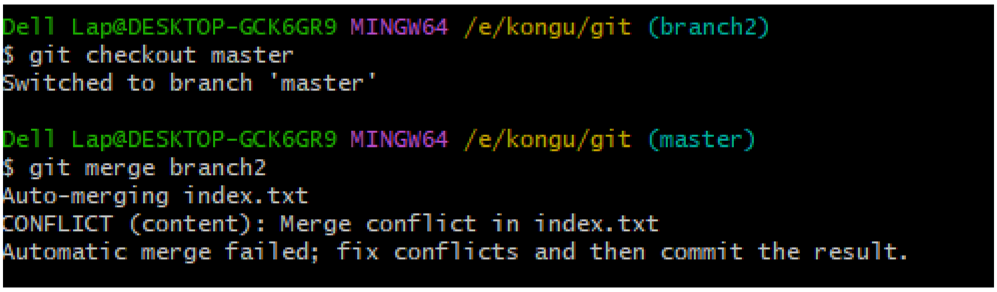
```
First line in master
second line in branch2
```

Add & commit the changes in the branch2



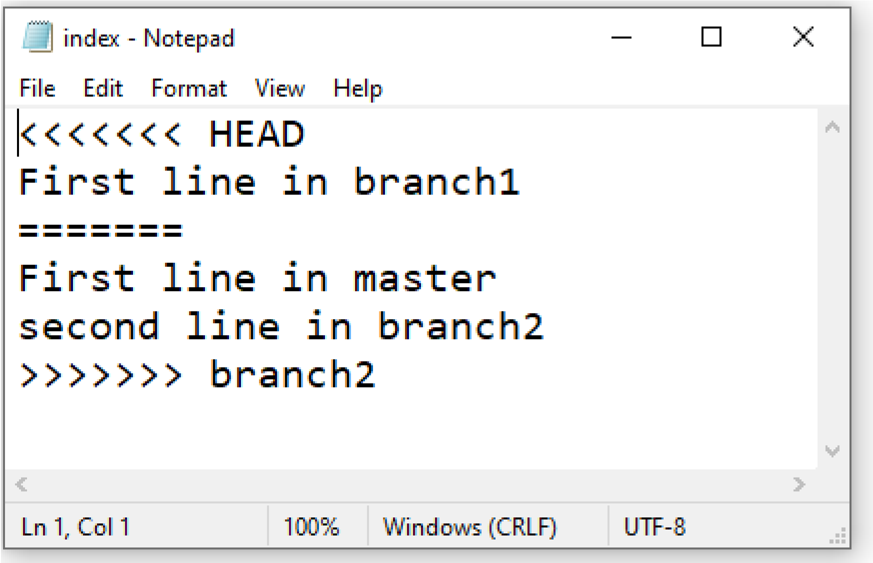
```
Dell Lap@DESKTOP-GCK6GR9 MINGW64 /e/kongu/git (branch2)
$ git add --all

Dell Lap@DESKTOP-GCK6GR9 MINGW64 /e/kongu/git (branch2)
$ git commit -m "third"
[branch2 5c905bb] third
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Checkout to master and merge



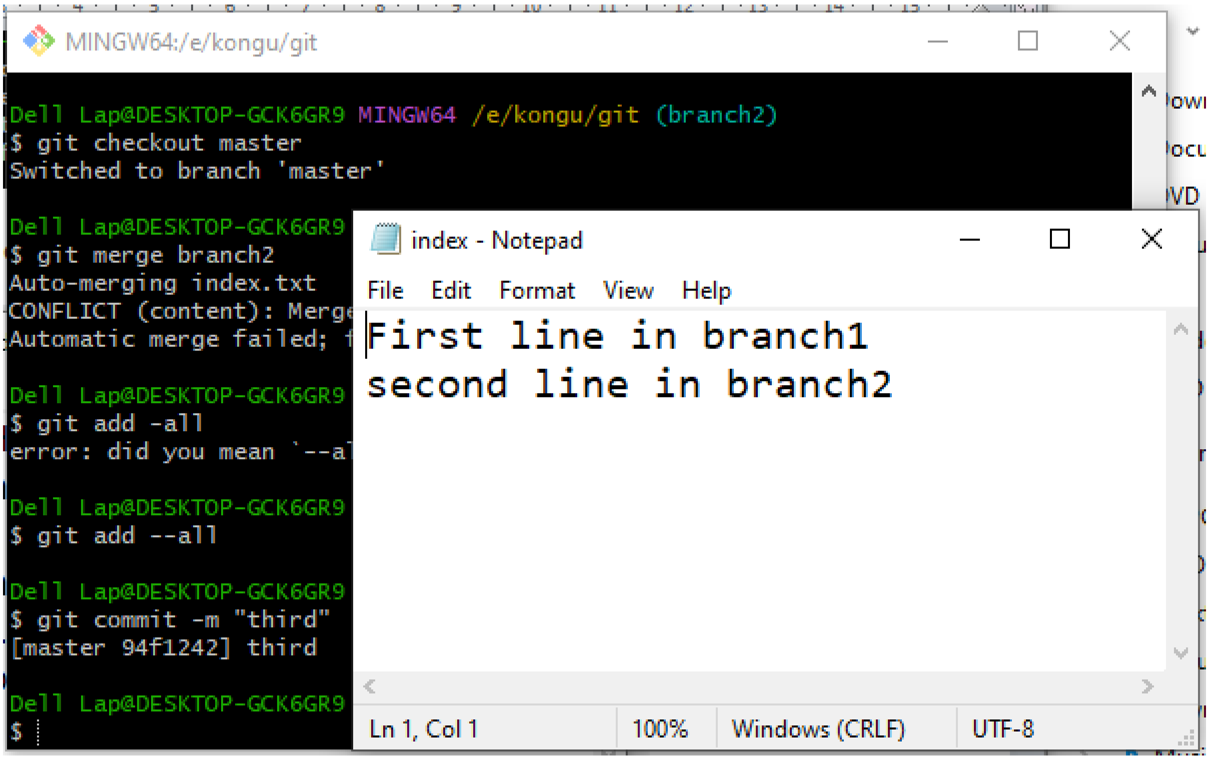
```
Dell Lap@DESKTOP-GCK6GR9 MINGW64 /e/kongu/git (branch2)
$ git checkout master
Switched to branch 'master'

Dell Lap@DESKTOP-GCK6GR9 MINGW64 /e/kongu/git (master)
$ git merge branch2
Auto-merging index.txt
CONFLICT (content): Merge conflict in index.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflict occurs



```
<<<<<<< HEAD
First line in branch1
=======
First line in master
second line in branch2
>>>>>>> branch2
```

Resolve the conflict and commit



```
Dell Lap@DESKTOP-GCK6GR9 MINGW64 /e/kongu/git (branch2)
$ git checkout master
Switched to branch 'master'

Dell Lap@DESKTOP-GCK6GR9
$ git merge branch2
Auto-merging index.txt
CONFLICT (content): Merge
Automatic merge failed; f

Dell Lap@DESKTOP-GCK6GR9
$ git add -all
error: did you mean `--al

Dell Lap@DESKTOP-GCK6GR9
$ git add --all

Dell Lap@DESKTOP-GCK6GR9
$ git commit -m "third"
[master 94f1242] third

Dell Lap@DESKTOP-GCK6GR9
$
```

```
First line in branch1
second line in branch2
```