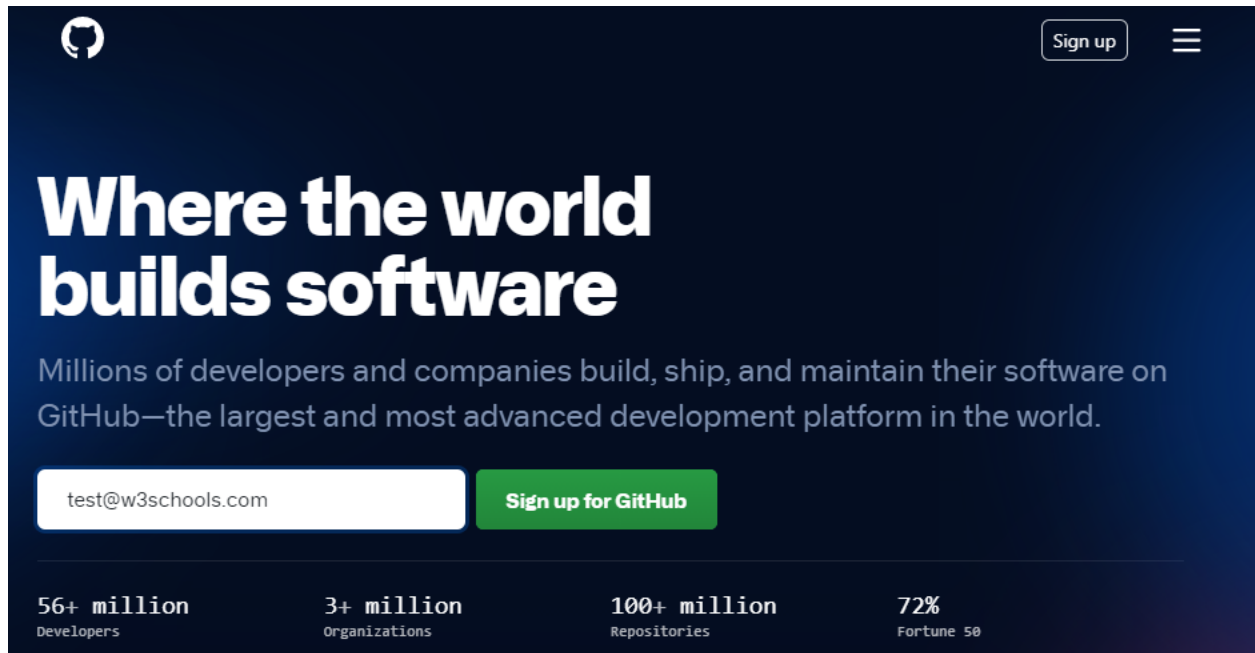


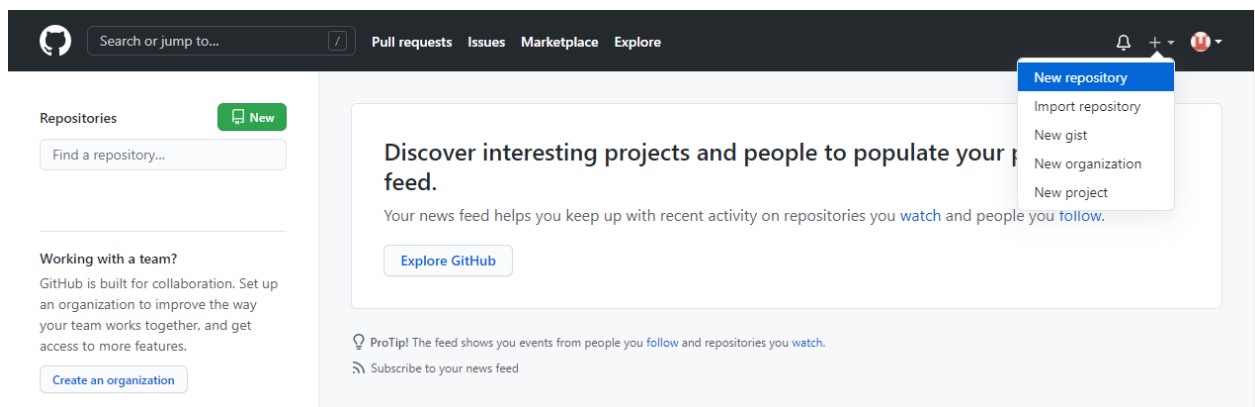
GitHub Account

Go to [GitHub](#) and sign up for an account:

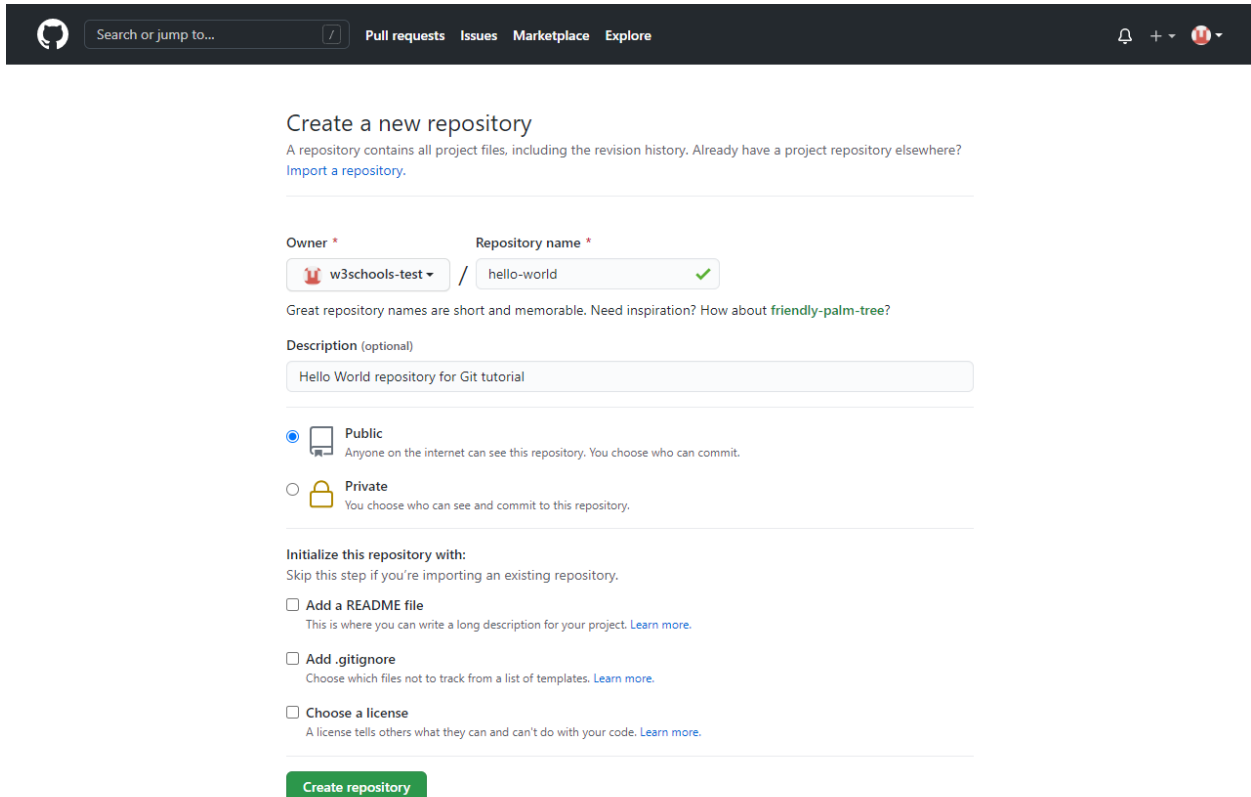


Create a Repository on GitHub

Now that you have made a GitHub account, sign in, and create a new Repo:



And fill in the relevant details:



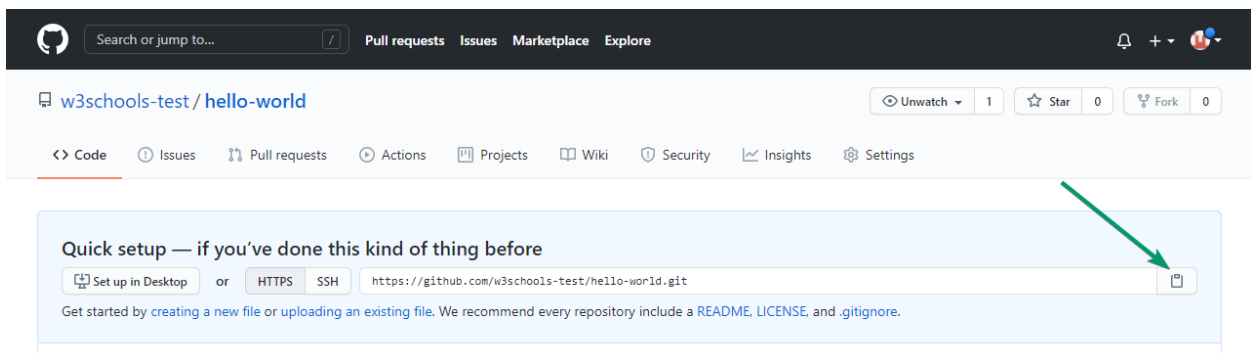
The screenshot shows the GitHub 'Create a new repository' page. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the page title is 'Create a new repository'. A sub-header explains that a repository contains all project files and provides a link to 'Import a repository'. The form fields are: 'Owner' (w3schools-test), 'Repository name' (hello-world), and 'Description' (Hello World repository for Git tutorial). There are two radio button options for visibility: 'Public' (selected) and 'Private'. Below these are three checkboxes for initialization: 'Add a README file', 'Add .gitignore', and 'Choose a license'. A green 'Create repository' button is at the bottom.

We will go over the different options and what they mean later. But for now, choose Public (if you want the repo to be viewable for anyone) or Private (if you want to choose who should be able to view the repo). Either way, you will be able to choose who can **contribute** to the repo.

Then click "Create repository".

Push Local Repository to GitHub

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



The screenshot shows the GitHub repository page for 'w3schools-test/hello-world'. The page header includes the repository name, 'Unwatch', 'Star' (1), and 'Fork' (0) buttons. Below the header is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. A blue box titled 'Quick setup — if you've done this kind of thing before' contains a 'Set up in Desktop' button, an 'or' separator, and 'HTTPS' and 'SSH' buttons. A text input field shows the URL 'https://github.com/w3schools-test/hello-world.git' with a copy icon on the right. A green arrow points to the copy icon. Below the input field, it says '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:

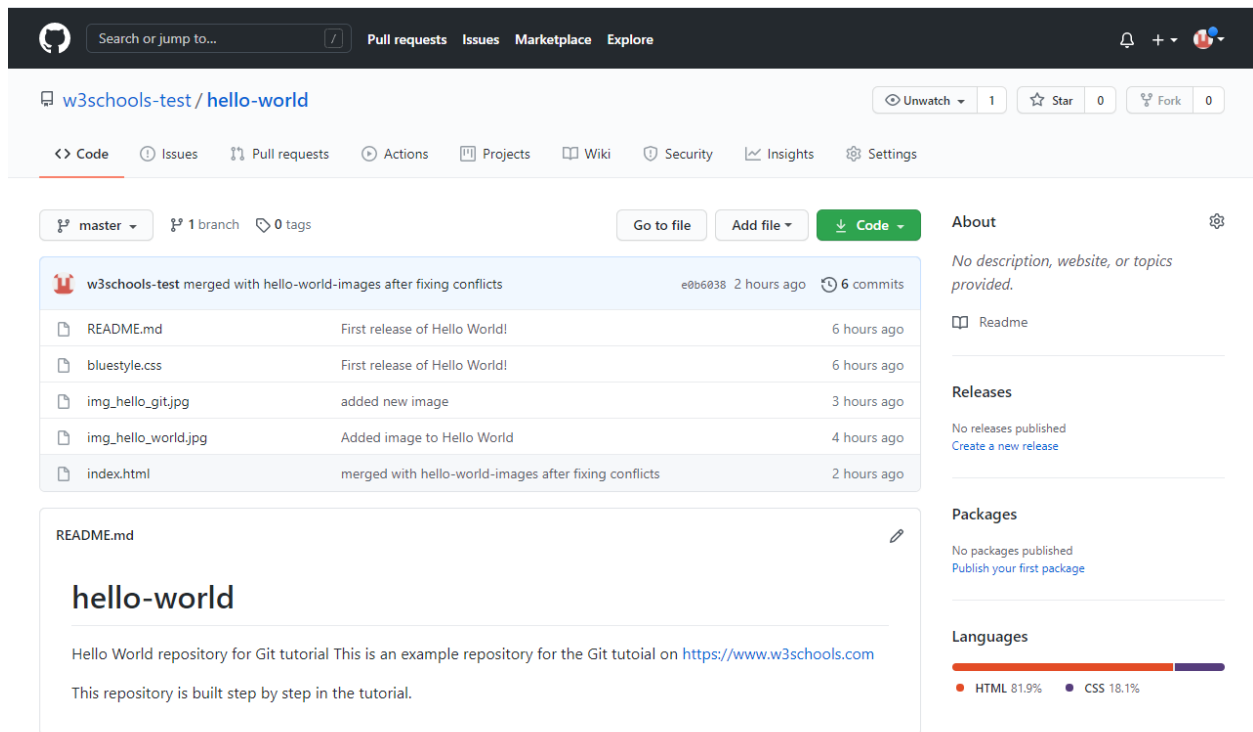
```
git remote add origin https://github.com/w3schools-test/hello-world.git
```

`git remote add origin URL` specifies that you are adding a remote repository, with the specified `URL`, as an `origin` to your local Git repo.

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
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:



The screenshot displays the GitHub interface for the repository `w3schools-test/hello-world`. At the top, there are navigation links for Pull requests, Issues, Marketplace, and Explore. Below the repository name, there are statistics for Unwatch (1), Star (0), and Fork (0). The main content area shows a list of files and their commit history:

File	Commit Message	Time
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

The README content is as follows:

```
hello-world

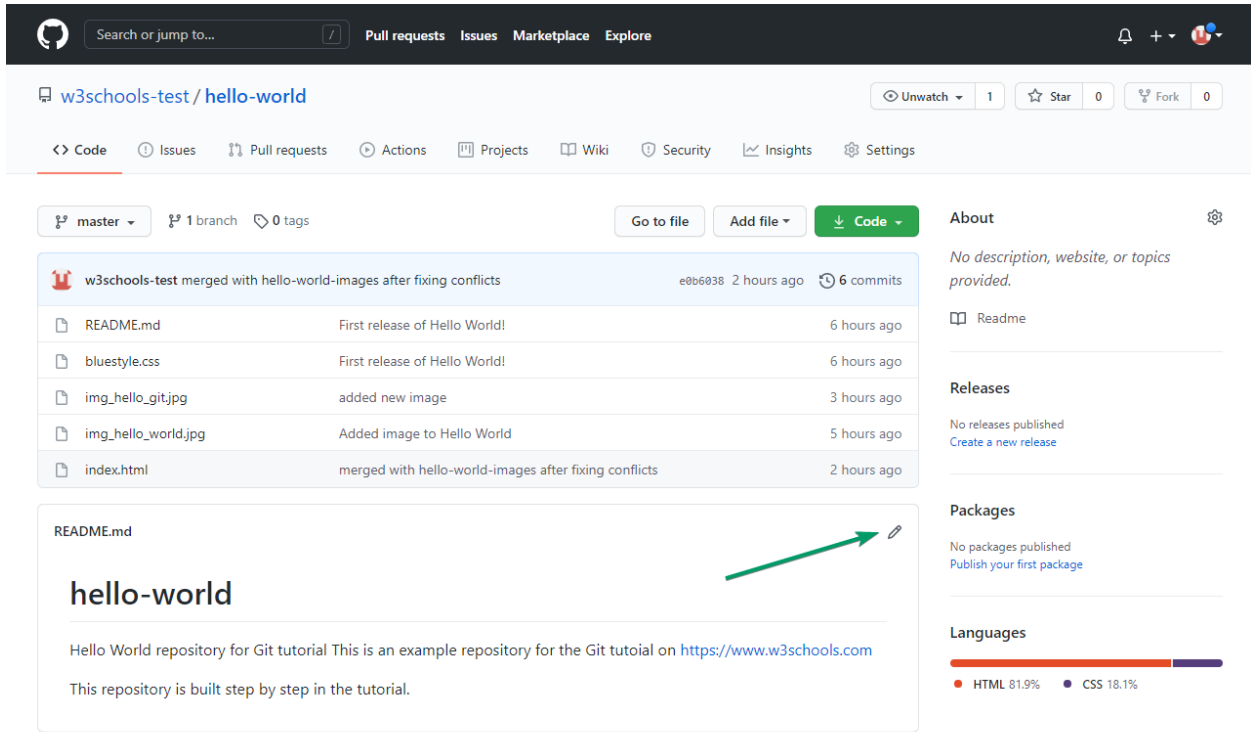
Hello World repository for Git tutorial This is an example repository for the Git tutorial on https://www.w3schools.com

This repository is built step by step in the tutorial.
```

Edit Code in GitHub

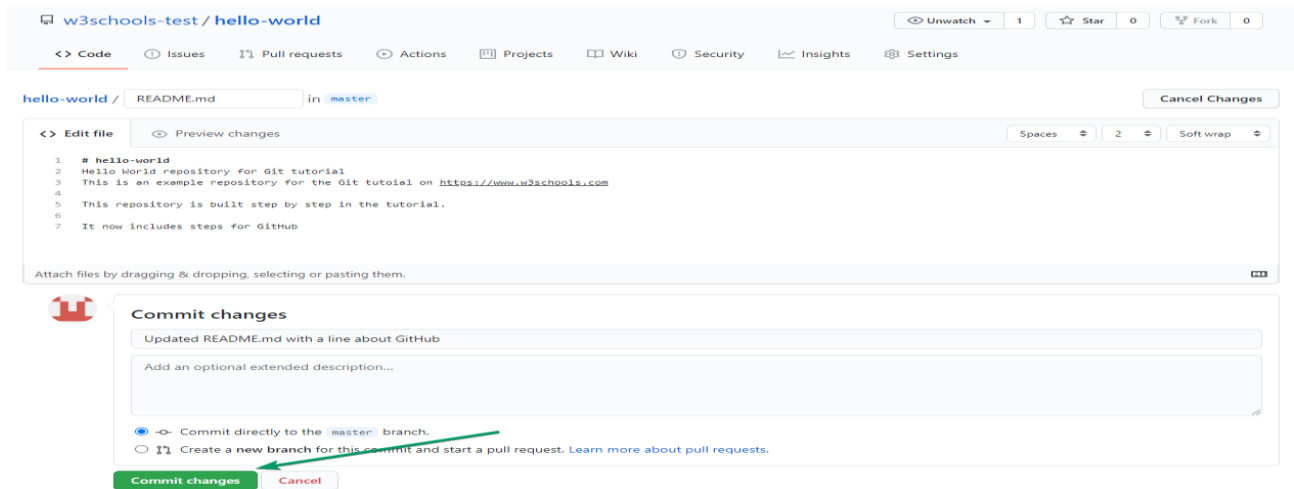
In addition to being a host for Git content, GitHub has a very good code editor.

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



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

Remember to add a description for the **commit**:



Git Pull from GitHub

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`

Let's take a closer look into how `fetch`, `merge`, and `pull` works.

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:

Example

```
git fetch origin
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), 733 bytes | 3.00 KiB/s, done.
From https://github.com/w3schools-test/hello-world
   e0b6038..d29d69f  master    -> origin/master
```

Now that we have the recent `changes`, we can check our `status`:

Example

```
git status
On branch master
```

Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.

(use "git pull" to update your local branch)

nothing to commit, working tree clean

We are behind the `origin/master` by 1 `commit`. That should be the updated `README.md`, but lets double check by viewing the `log`:

Example

```
git log origin/master
```

```
commit d29d69ffe2ee9e6df6fa0d313bb0592b50f3b853 (origin/master)
```

```
Author: w3schools-test <77673807+w3schools-test@users.noreply.github.com>
```

```
Date:   Fri Mar 26 14:59:14 2021 +0100
```

```
    Updated README.md with a line about GitHub
```

```
commit e0b6038b1345e50aca8885d8fd322fc0e5765c3b (HEAD -> master)
```

```
Merge: dfa79db 1f1584e
```

```
Author: w3schools-test
```

```
Date:   Fri Mar 26 12:42:56 2021 +0100
```

```
    merged with hello-world-images after fixing conflicts
```

```
...
```

```
...
```

That looks as expected, but we can also verify by showing the differences between our local `master` and `origin/master`:

Example

```
git diff origin/master
```

```
diff --git a/README.md b/README.md
```

```
index 23a0122..a980c39 100644
```

```
--- a/README.md
+++ b/README.md
@@ -2,6 +2,4 @@
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
+This repository is built step by step in the tutorial.
\ No newline at end of file
```

That looks precisely as expected! Now we can safely **merge**.

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**:

Example

```
git merge origin/master
```

```
Updating e0b6038..d29d69f
```

```
Fast-forward
```

```
README.md | 4 +++-
```

```
1 file changed, 3 insertions(+), 1 deletion(-)
```

Check our **status** again to confirm we are up to date:

Example

```
git status
```

On branch master

Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

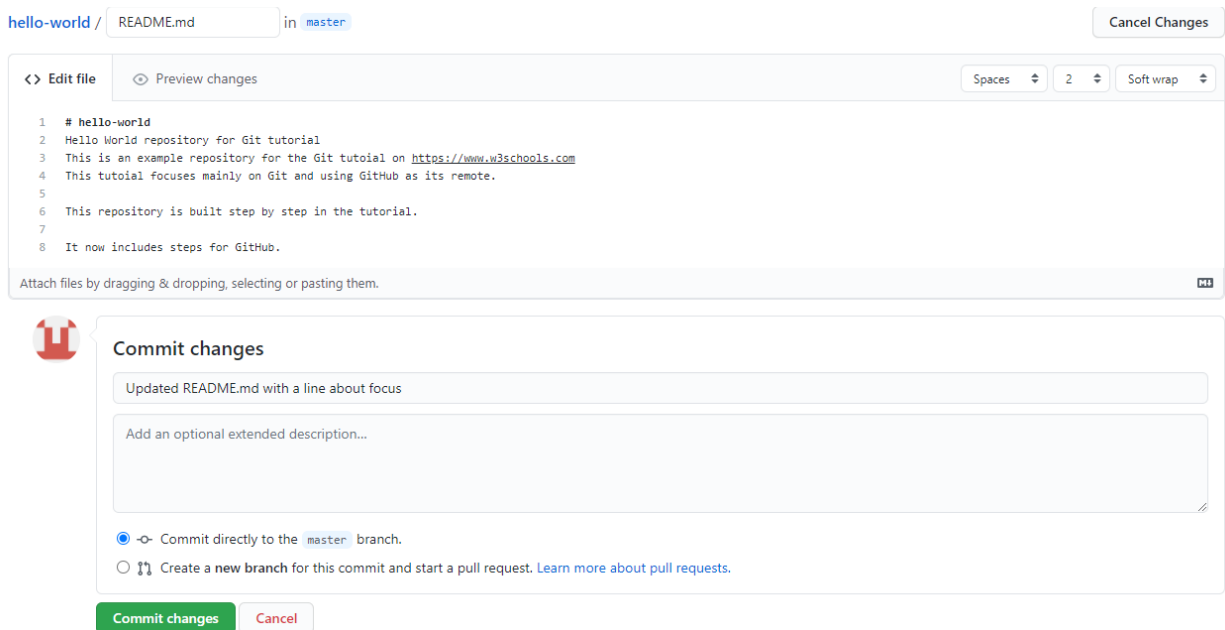
There! Your local git is up to date!

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.



The screenshot shows a GitHub web editor interface. At the top, it indicates the file path 'hello-world / README.md' and the current branch 'in master'. A 'Cancel Changes' button is visible in the top right. Below the path, there are tabs for 'Edit file' and 'Preview changes'. The 'Edit file' tab is active, showing a code editor with 8 lines of text: 1 # hello-world, 2 Hello World repository for Git tutorial, 3 This is an example repository for the Git tutorial on <https://www.w3schools.com>, 4 This tutorial focuses mainly on Git and using GitHub as its remote., 5, 6 This repository is built step by step in the tutorial., 7, 8 It now includes steps for GitHub. Below the editor, there is a section for 'Commit changes' with a GitHub logo. It contains a text input field with the text 'Updated README.md with a line about focus', a larger text area for an optional extended description, and two radio button options: 'Commit directly to the master branch.' (selected) and 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' At the bottom of the commit dialog are two buttons: 'Commit changes' and 'Cancel'.

Use **pull** to update our local Git:

Example

```
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(+)
```

That is how you keep your local Git up to date from a remote repository.

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></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></div>

</body>
</html>
```

Commit the changes:

Example

```
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

```
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

```
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
```

```
5a04b6f..facaee master -> master
```

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

Updated index.html. Resized image

Browse files

master

w3schools-test committed 14 minutes ago

1 parent d29d69f commit e7de78fdefdda51f6f961829fcbdf197e9b926b6

Showing 1 changed file with 1 addition and 1 deletion.

Unified Split

```
index.html 2
```

```
@@ -7,7 +7,7 @@
7 7 <body>
8 8
9 9 <h1>Hello world!</h1>
10 10 - <div></div>
11 11 + <div></div>
12 12 <p>This is the first file in my new Git Repo.</p>
13 13 <p>This line is here to show how merging works.</p>
14 14 <div></div>
```

Git GitHub Flow

Working using the GitHub Flow

The GitHub flow is a workflow designed to work well with Git and GitHub.

It focuses on branching and makes it possible for teams to experiment freely, and make deployments regularly.

The GitHub flow works like this:

- Create a new Branch
- Make changes and add Commits
- Open a Pull Request
- Review
- Deploy
- Merge

You should already have a good understanding of how this works from the previous chapters. This chapter focuses on understanding how the flow makes it easy for you to work together.

Create a New Branch

Branching is the key concept in Git. And it works around the rule that the master branch is ALWAYS deployable.

That means, if you want to try something new or experiment, you create a new branch! Branching gives you an environment where you can make changes without affecting the main branch.

When your new branch is ready, it can be reviewed, discussed, and merged with the main branch when ready.

When you make a new branch, you will (almost always) want to make it from the master branch.

Note: Keep in mind that you are working with others. Using descriptive names for new branches, so everyone can understand what is happening.

Make Changes and Add Commits

After the new branch is created, it is time to get to work. Make changes by adding, editing and deleting files. Whenever you reach a small milestone, add the changes to your branch by commit.

Adding commits keeps track of your work. Each commit should have a message explaining what has changed and why. Each commit becomes a part of the history of the branch, and a point you can revert back to if you need to.

Note: commit messages are very important! Let everyone know what has changed and why. Messages and comments make it so much easier for yourself and other people to keep track of changes.

Open a Pull Request

Pull requests are a key part of GitHub. A Pull Request notifies people you have changes ready for them to consider or review.

You can ask others to review your changes or pull your contribution and merge it into their branch.

Review

When a Pull Request is made, it can be reviewed by whoever has the proper access to the branch. This is where good discussions and review of the changes happen.

Pull Requests are designed to allow people to work together easily and produce better results together!

If you receive feedback and continue to improve your changes, you can push your changes with new commits, making further reviews possible.

Note: GitHub shows new commit and feedback in the "unified Pull Request view".

Deploy

When the pull request has been reviewed and everything looks good, it is time for the final testing. GitHub allows you to deploy from a branch for final testing in production before merging with the master branch.

If any issues arise, you can undo the changes by deploying the master branch into production again!

Note: Teams often have dedicated testing environments used for deploying branches.

Merge

After exhaustive testing, you can merge the code into the master branch!

Pull Requests keep records of changes to your code, and if you commented and named changes well, you can go back and understand why changes and decisions were made.

Note: You can add keywords to your pull request for easier searching!