

Git Contents

Git Get Started

1. Git New Files
2. Git Staging Environment
3. Git Commit
4. Git Help
5. Git Branch
6. Git Branch Merge

Git and GitHub

1. GitHub Get Started
2. GitHub Edit Code
3. Pull from GitHub
4. Push to GitHub
5. GitHub Branch
6. Pull Branch from GitHub
7. Push Branch to GitHub
8. GitHub Flow
9. GitHub Pages

Git Contribute

1. GitHub Fork
2. Git Clone from GitHub
3. GitHub Send Pull Request

Git Advanced

1. Git .gitignore
2. Git Security SSH
3. GitHub Add SSH

Git Undo

1. Git Revert
2. Git Reset
3. Git Amend

Git Get Started:

1) Get Started:-->

To check the Version of the Git use this command in cmd or Terminal.

---> git --version

Output---> git version 2.30.2.windows.1

To Configure Git

Now let Git know who you are.

This is important for version control systems, as each Git commit uses this information:

local user: git config --global user.name "Git-test"

local user: git config --global user.email "test@git.com"

Creating Git Folder:

local user: mkdir myproject —> Make a directory of folder name is myproject

local user: cd myproject

#Initialize Git

Once you have navigated to the correct folder, you can initialize Git on that folder:

--> git init

(Initialize empty Git repository in /Users/user/myproject/.git)

For Eg:- Create an Html file that contains the heading Hello World and with Paragraph.

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
</head>
<body>
<h1>Hello world!</h1>
<p>This is the first file in my new Git Repo.</p>
</body>
</html>
```

And Save it as `index.html`

Let's go back to the terminal and list the files in our current working directory:

ls - List

Steps:-

- a) `git init` —> This is the step of initializing the file
- b) `git ls` —> Showing the list of the files or folder
- c) `git status` —> Status of the process

2) Git Staging Environment:--->

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:

Steps:-

- a) `git add index.html`
- b) `git status`
- c) `git add -all` or `git add -A`

3) Git Commit:--->

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

Adding commits keeps 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**.

a) git commit -m "First release of Hello world"

Git status - - short

b) Git Commit without Stage:- It is possible to commit changes directly, skipping the staging environment.

The `-a` option will automatically stage every changed, already tracked file.

- -> git commit -a -m "Update index.html with a new line"

git log (To view the history of commits for a repository)

4) Git Help:--->

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

`git command -help` - See all the available options for the specific command

`git help --all` - See all possible commands

5) Git Branch:--->

In Git, a **branch** is a new/separate version of the main repository.

New Git Branch:- **Now we created a new branch called "hello-world-images". Let's confirm that we have created a new branch:**

Eg:- a) **git branch hello-world-images**

b) git branch

c) git checkout hello-world-images

d) The edit in and IDE or VSCode and edit the file (index.html) or add a new file(image.jpg).

e) git checkout hello-world-images Switched to branch 'hello-world-images' —> Checkout is the command used to check out a branch.

f) git status

g) git add - -all

h) git status

i) git commit -m "Added image to Hello World"

Switching Between Branches:-

Steps:-

- a) ls
- b) git checkout master
- c) ls

Emergency Branch:- If you don't want to mess with master directly and I do not want to mess with hello-world-images, then use this command:-

Steps:-

- a) **git checkout -b emergency-fix**
- b) git status (To Check The status)
- c) git add index.html

6) Git Branch Merge:-->

After emergency fix ready and then merge the master and emergency-fix branches.

Steps:-

- a) **git checkout master** - Checkout the master branch
- b) **git merge emergency-fix** - Then merge with emergency branch
- c) **git branch -d emergency-fix** (-d for delete operations)
 - As master and emergency-fix are essentially the same now, we can delete emergency-fix, as it is no longer needed:

d) **Merge Conflict:-**

It is a process in which github shows the duplicate file's data in the repository/branches and that have to be resolved by using this process are:-

1) git checkout hello-world-images -----(It will shift to this branch)

2) git add - - all

3) git commit -m "added new image"

4) git checkout master ----- (Swift to the master branch)

5) git merge hello-world-images

6) git status

7) Then after fixing the index.html file:

git add index.html

git status

git commit -m "merged with hello-world-images after fixing conflicts"

8) Then delete the hello-world-images branch:-

git branch -d hello-world-images

Git and GitHub

1) GitHub Get Started:--->

Steps:-

- 1) Create a new repository and use the HTTP Link of the repository.
- 2) git remote add origin <https://github.com/xyz.git> (Or - Link)
- 3) git push - -set-upstream origin master

2) GitHub Edit Code:--->

It is an easy step in GitHub you can do it in the repository section.

- a) Readme.md file
- b) and Commit Changes.

3) Pull from GitHub:--->

Pull is the combination of 2 Different commands:-

- 1) fetch
- 2) merge

Steps:-

- | | |
|----------------------------|--|
| 1) git fetch origin | --- It will fetch the origin data |
| 2) git log origin/master | --- It will obtain the history of branch |
| 3) git diff origin/master | --- Differentiate between them. |
| 4) git merge origin/master | --- It will merge the origin and master |
| 5) git pull origin | --- Git will pull in the origin |

4) Push to GitHub:--->

Let's try making some changes to our local git file (index.html) and pushing them to GitHub.

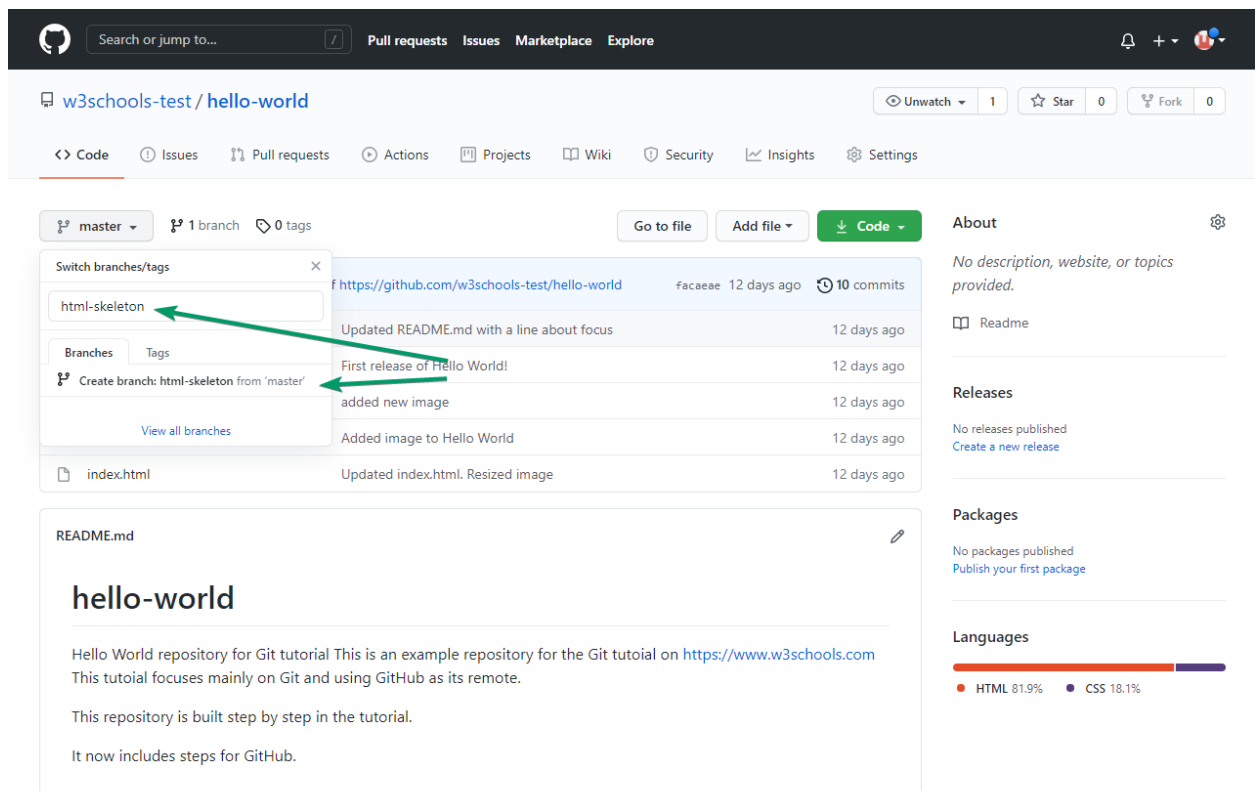
- 1) git commit -a -m "Updated index.html Resized image"
- 2) git push origin
- 3) git status

5) GitHub Branch:--->

Create a new Branch on Github using master

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:



The screenshot shows the GitHub interface for the repository `w3schools-test/hello-world`. The 'master' branch is selected. A dropdown menu is open for the 'master' branch, showing a list of branches including 'html-skeleton'. A green arrow points from the 'html-skeleton' branch in the dropdown to the 'Create branch: html-skeleton from 'master'' button. The commit history is visible, showing the first release of 'Hello World!' and subsequent updates to the README and index.html. The README content is displayed below the commit history, showing the 'hello-world' title and a description of the repository.

Switch branches/tags

html-skeleton

Branches

Tags

Create branch: html-skeleton from 'master'

View all branches

index.html

Updated index.html. Resized image

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

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.

About

No description, website, or topics provided.

Readme

Releases

No releases published

Create a new release

Packages

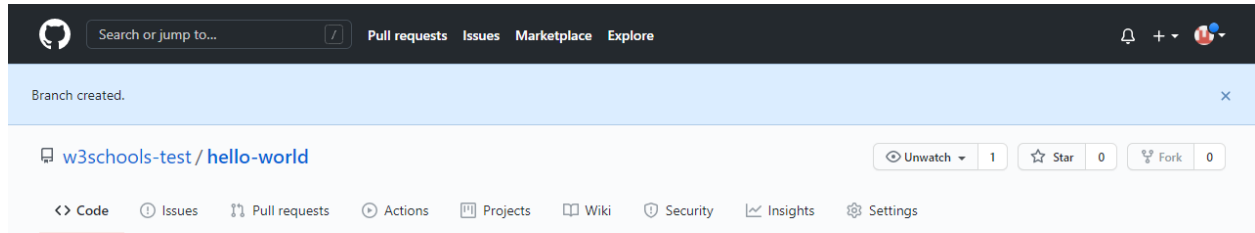
No packages published

Publish your first package

Languages

HTML 81.9% CSS 18.1%

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"?

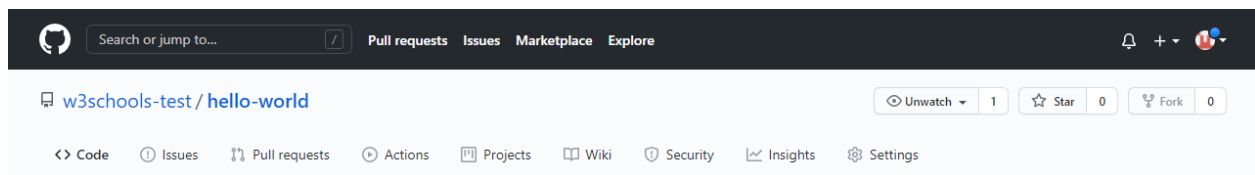


html-skeleton 2 branches 0 tags

This branch is even with master.

Pull request Compare

w3schools-test	Merge branch 'master' of https://github.com/w3schools-test/hello-world	faceae 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	



html-skeleton hello-world / index.html

Go to file ...

w3schools-test Updated index.html. Resized image

Latest commit e7de78f 12 days ago History

1 contributor

16 lines (14 sloc) 458 Bytes

Raw Blame Edit this file

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Hello World!</title>
5 <link rel="stylesheet" href="bluestyle.css">
6 </head>
7 <body>
8
9 <h1>Hello world!</h1>
10 <div></div>
11 <p>This is the first file in my new Git Repo.</p>
12 <p>This line is here to show how merging works.</p>
13 <div></div>
14
15 </body>
16 </html>
```

The screenshot shows a GitHub repository named 'w3schools-test/hello-world'. The file 'index.html' is being edited, and the changes are being previewed. The diff shows several additions and deletions, including meta tags and image links. A green arrow points from the 'Commit changes' button to the commit message field, which contains the text 'Added some meta tags to index.html'. Another green arrow points from the 'Commit changes' button to the 'Commit directly to the html-skeleton branch' radio button.

hello-world / index.html in html-skeleton Cancel changes

<> Edit file Preview changes

```
... @@ -1,16 +1,18 @@
1 1 <!DOCTYPE html>
2 - <html>
3 + <html lang="en">
4   <head>
5 -   <title>Hello World!</title>
6 -   <link rel="stylesheet" href="bluestyle.css">
7 +   <meta charset="UTF-8">
8 +   <title>Hello World!</title>
9 +   <meta name="viewport" content="width=device-width,initial-scale=1">
10 +   <link rel="stylesheet" href="bluestyle.css">
11 </head>
12 <body>
13 - <h1>Hello world!</h1>
14 - <div></div>
15 - <p>This is the first file in my new Git Repo.</p>
16 - <p>This line is here to show how merging works.</p>
17 - <div></div>
18 + <h1>Hello world!</h1>
19 + <div></div>
20 + <p>This is the first file in my new Git Repo.</p>
21 + <p>This line is here to show how merging works.</p>
22 + <div></div>
23 </body>
24 </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.](#)

Commit changes Cancel

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!

6) Pull Branch from GitHub:--->

Pulling a branch from GitHub:-

Now 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:

1) `git pull`

2) `git status`

3) `git branch`

4) `git branch -a` — > 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: (Note: `branch -r` is for remote branches only.)

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

5) `git checkout html-skeleton` —> switch the branch

6) `git pull` —> And check if it is all up to date:

7) `git branch` --- > Which branches do we have now, and where are we working from?

7) Push 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:

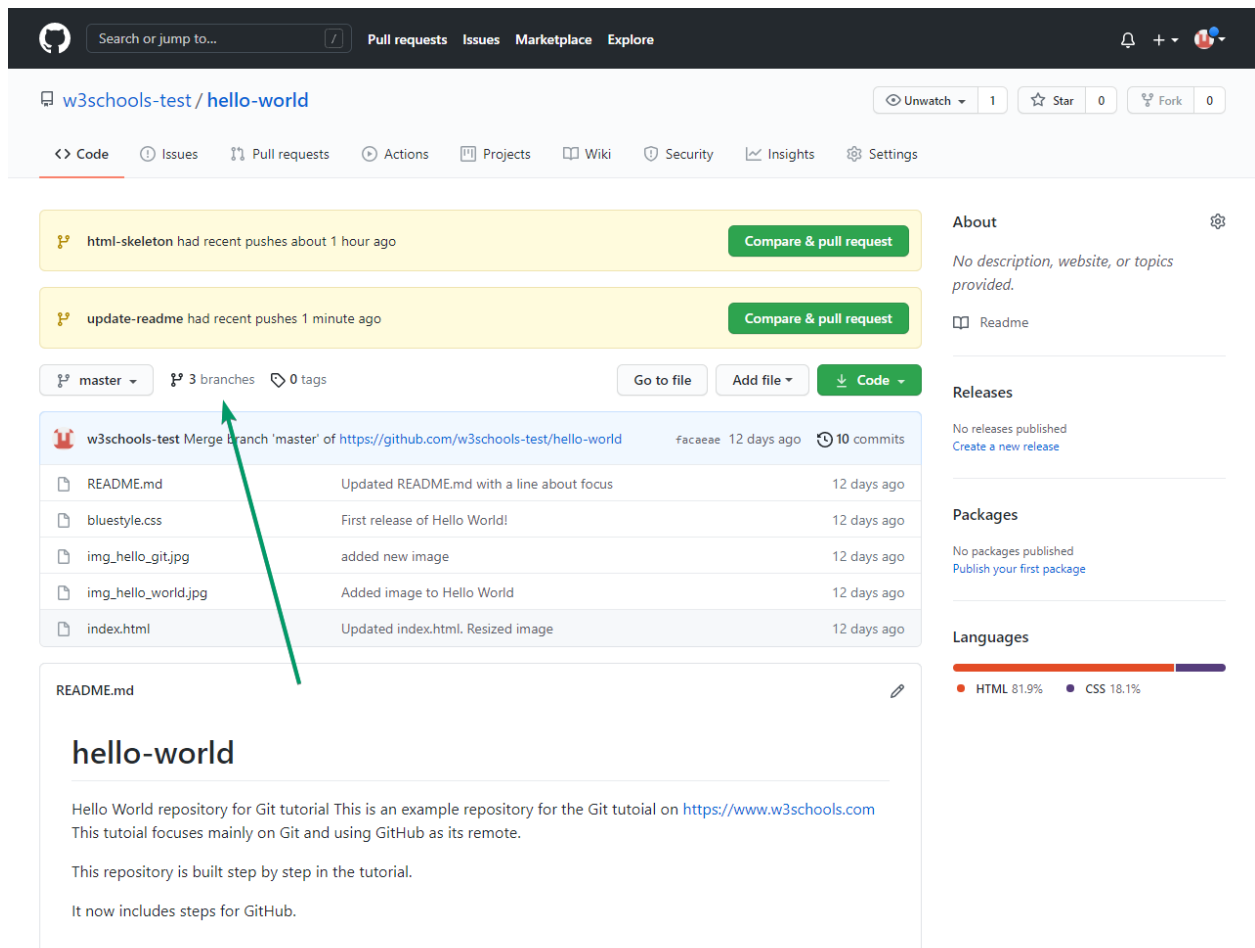
- 1) `git checkout -b update-readme` —> creating a new branch
- 2) `git status` —> checking the status
- 3) `git add README.md` —> is modified but not added to the Staging Environment.
- 4) `git status` —> to check the status of the branch
- 5) `git commit -m "updated readme for GitHub Branches"` —>
Then saves our changes
- 6) `git push origin update-readme` —>

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

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

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:



The screenshot displays the GitHub interface for the repository `w3schools-test/hello-world`. The repository has 1 star and 0 forks. The main navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository header shows the current branch as `master` with 3 branches and 0 tags. A green arrow points to the '3 branches' link. Below the header, a table lists the repository's files and their commit history:

File	Commit Message	Commit Hash	Time Ago
README.md	Updated README.md with a line about focus	facæææ	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

The repository also includes a 'Compare & pull request' button. The right sidebar shows the repository's 'About' section, which is currently empty, and the 'Releases' section, which also shows no releases published. The 'Languages' section indicates that the repository is primarily composed of HTML (81.9%) and CSS (18.1%).

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:

The screenshot displays a GitHub pull request interface. At the top, a summary bar indicates "2 commits", "2 files changed", "0 comments", and "2 contributors". Below this, the commit history for April 07, 2021, is shown, including "Updated index.html with basic meta" (verified, commit daf4f7c) and "Updated readme for GitHub Branches" (commit 836e5bf). A message states "Showing 2 changed files with 12 additions and 9 deletions." The interface shows two files: README.md and index.html. The diff view for index.html is expanded, showing a comparison between the current branch (left) and the base branch (right). The diff highlights changes in the HTML structure, including the addition of a language attribute to the HTML tag, a meta charset attribute, a viewport meta tag, and the addition of two image tags for "Hello World from Space" and "Hello Git". The diff uses a color-coded system: green for additions, red for deletions, and blue for context lines.

```
@@ -6,3 +6,4 @@ This tutorial focuses mainly on Git and using GitHub as its remote.
6 6 This repository is built step by step in the tutorial.
7 7
8 8 It now includes steps for GitHub.
9 + Including how to work with Branches on GitHub.

... @@ -1,16 +1,18 @@
1 1 <!DOCTYPE html>
2 - <html>
2 + <html lang="en">
3 3 <head>
4 - <title>Hello World!</title>
5 - <link rel="stylesheet" href="bluestyle.css">
4 + <meta charset="UTF-8">
5 + <title>Hello World!</title>
6 + <meta name="viewport" content="width=device-width,initial-scale=1">
7 + <link rel="stylesheet" href="bluestyle.css">
6 8 </head>
7 9 <body>
8 10
9 - <h1>Hello world!</h1>
10 - <div></div>
11 - <p>This is the first file in my new Git Repo.</p>
12 - <p>This line is here to show how merging works.</p>
13 - <div></div>
11 + <h1>Hello world!</h1>
12 + <div></div>
13 + <p>This is the first file in my new Git Repo.</p>
14 + <p>This line is here to show how merging works.</p>
15 + <div></div>
14 16 </body>
15 17 </html>
16 - </html>
18 + </html>
```

Note: This comparison shows both the changes from **update-readme** and **html-skeleton** because we created the new branch FROM **html-skeleton**.

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

The screenshot shows the GitHub interface for a pull request in the repository 'w3schools-test/hello-world'. The page title is 'Open a pull request'. Below the title, it says 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' The main area shows a comparison between 'base: master' and 'compare: update-readme', with a green checkmark indicating 'Able to merge. These branches can be automatically merged.' The pull request title is 'Update readme'. The description field contains the text 'Updated readme with branches info'. A green button labeled 'Create pull request' is visible, with a green arrow pointing to it. The right sidebar shows various settings: 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), and Helpful resources (GitHub [Community Guidelines](#)). The bottom bar shows '2 commits', '2 files changed', '0 comments', and '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 Actions Projects Wiki Security Insights Settings

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

Write Preview H B I i <> @ ↩

Updated readme with branches info

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

Create pull request

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

A pull request is how you propose changes. You can ask some to review your changes or pull your contribution and merge it into their branch.

Since this is your own repository, you can **merge** your pull request yourself:

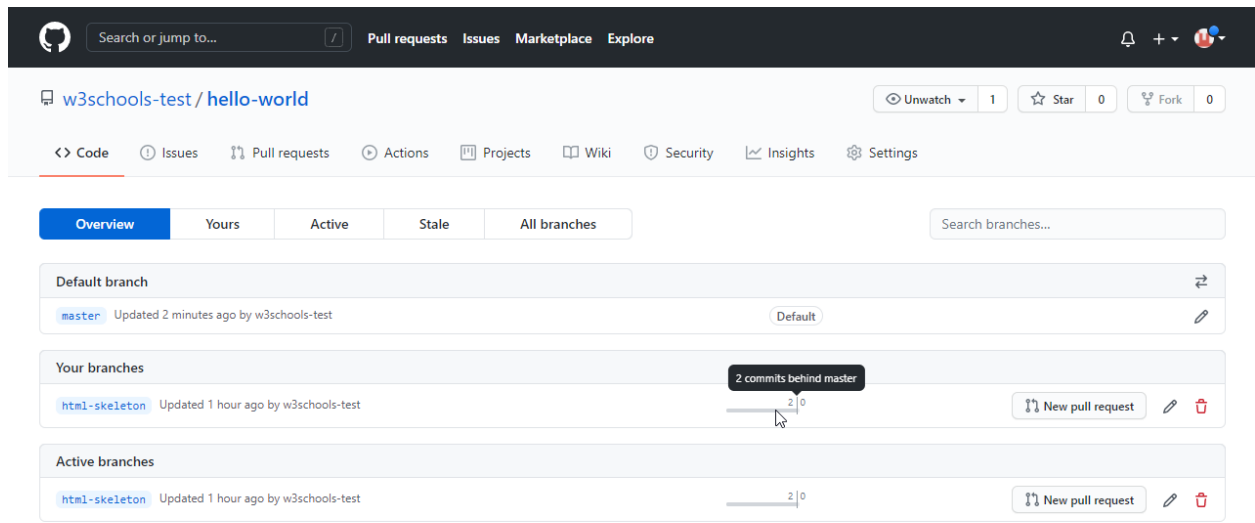
The screenshot shows a GitHub pull request interface for the repository `w3schools-test/hello-world`. The pull request is titled "Update readme #1" and is in the "Open" state. It shows that the user `w3schools-test` wants to merge 2 commits into the `master` branch from the `update-readme` branch. The commit history shows two commits: "Updated index.html with basic meta" and "Updated readme for GitHub Branches". A green arrow points to the "Merge pull request" button, which is highlighted in green. The right sidebar contains metadata such as "Reviews" (No reviews), "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), "Notifications" (Unsubscribe), and "1 participant".

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:

The screenshot shows a GitHub pull request interface for the repository 'w3schools-test/hello-world'. The pull request is titled 'Update readme #1' and is in the 'Merged' state. It shows a commit history with two commits: 'Updated index.html with basic meta' and 'Updated readme for GitHub Branches'. The pull request was merged into the 'master' branch. The interface includes a sidebar with navigation links (Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings) and a right-hand panel with metadata (Reviewers, Assignees, Labels, Projects, Milestone, Linked issues, Notifications). A comment box is visible at the bottom, and a 'Delete branch' button is present next to the 'Pull request successfully merged and closed' message.

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



8)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:

1. Create a new Branch
2. Make changes and add Commits
3. Open a Pull Request
4. Review
5. Deploy
6. Merge

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!

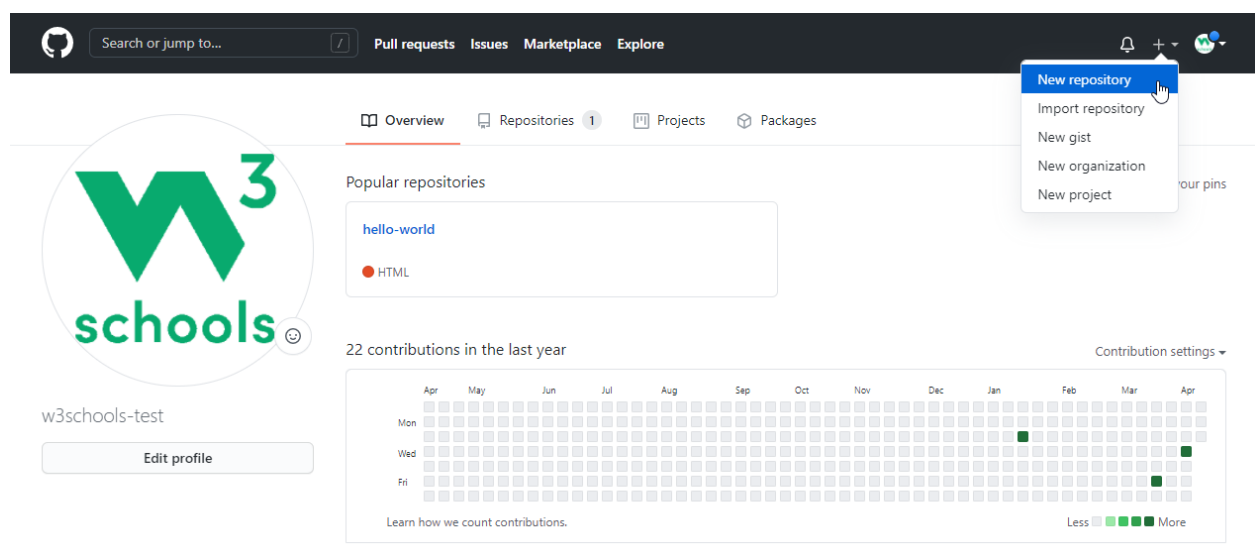
9)GitHub Pages:--->


Host Your Page on GitHub

With GitHub pages, GitHub allows you to host a webpage from your repository. Let's try to use GitHub Pages to host our repository.

Create a New Repository

Start by signing in to GitHub. GitHub pages need a special name and setup to work, so we start by creating a new repository:



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

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner ^{*}


Repository name ^{*}


w3schools-test / w3schools-test.github.io ✓

Great repository names are short and memorable. Need inspiration? How about [refactored-octo-garbanzo?](#)

Description (optional)

w3schools.com repository used in demonstrating GitHub and GitHub pages

☒  **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.](#)

Create repository

Push Local Repository to GitHub Pages

We add this new repository as a remote for our local repository, we are calling it **gh-page** (for GitHub Pages).

Copy the **URL** from here:


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

w3schools-test / w3schools-test.github.io

Unwatch 1 Star 0 Fork 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

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

 Set up in Desktop or ☐ HTTP ☒ SSH

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

And add it as a new **remote**:

```
git remote add gh-page
https://github.com/w3schools-test/w3schools-test.github.io.git
```

Make sure you are on the **master branch**, then push the **master branch** to the new **remote**:

User → **git push gh-page master**

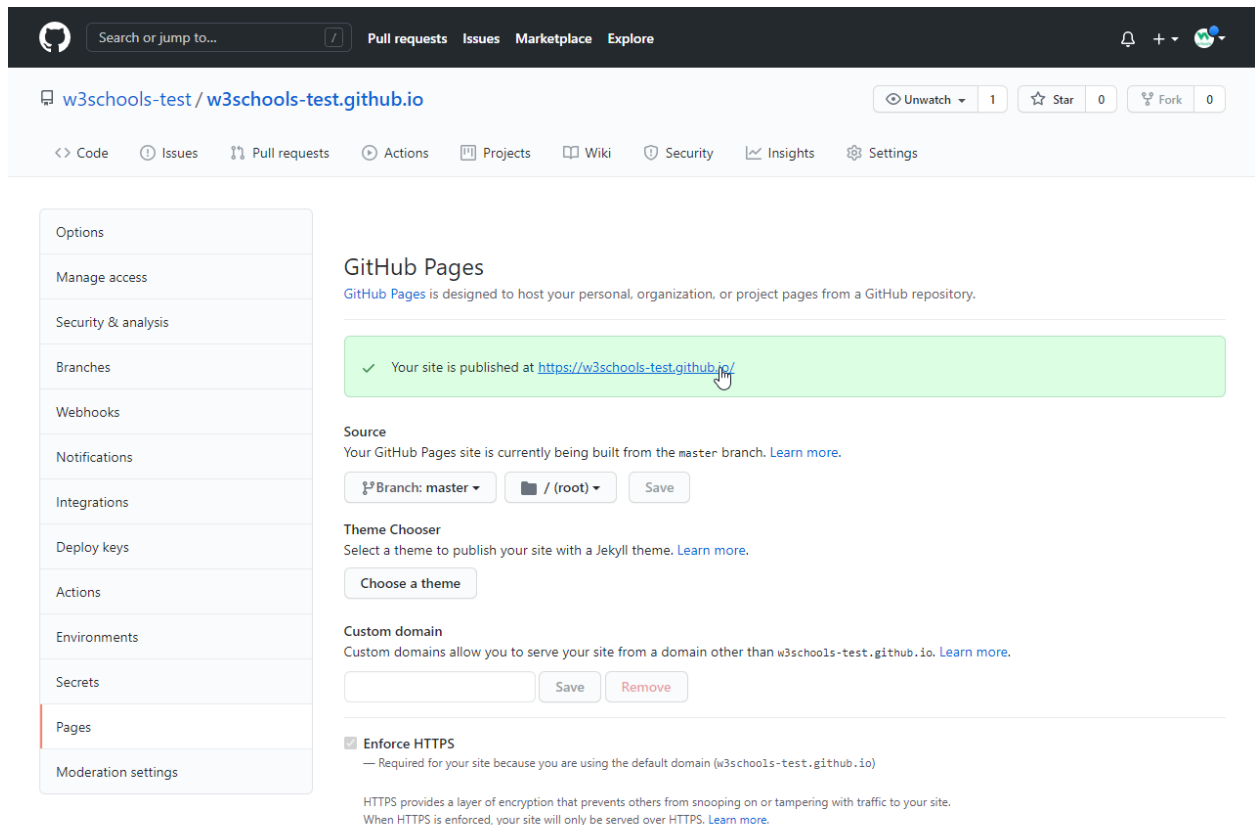
Note: If this is the first time you are connecting to GitHub, you will get some kind of notification to authenticate this connection.

Check that the new repository has received all the files:

The screenshot displays the GitHub interface for the repository `w3schools-test/w3schools-test.github.io`. The repository is on the `master` branch, which has 1 branch and 0 tags. It shows 10 commits, with the most recent commit by `faceae` 18 days ago. The file list includes `README.md` (updated with a line about focus), `bluestyle.css` (first release of Hello World!), `img_hello_git.jpg` (added new image), `img_hello_world.jpg` (added image to Hello World), and `index.html` (updated index.html. Resized image). The README.md content is visible, showing the title `hello-world` and a description: "Hello World repository for Git tutorial. This is an example repository for the Git tutorial on <https://www.w3schools.com>. This tutorial 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."

Check Out Your Own GitHub Page

That looks good, now click the Settings menu and navigate to the Pages tab:



The screenshot shows the GitHub repository settings page for the repository `w3schools-test` on the domain `w3schools-test.github.io`. The **Settings** menu is selected in the top navigation bar, and the **Pages** tab is selected in the left sidebar. The main content area displays the GitHub Pages configuration:

- GitHub Pages**: A green banner indicates that the site is published at <https://w3schools-test.github.io/>.
- Source**: The site is currently being built from the `master` branch. A `/ (root)` dropdown is visible.
- Theme Chooser**: A button labeled `Choose a theme` is present.
- Custom domain**: A text input field is shown, with `Save` and `Remove` buttons.
- Enforce HTTPS**: A checkbox is checked, indicating that HTTPS is enforced for the site.

Git Contribute

1)GitHub Fork:--->

Add to Someone Else's Repository

At the heart of Git is collaboration. However, Git does not allow you to add code to someone else's repository without access rights.

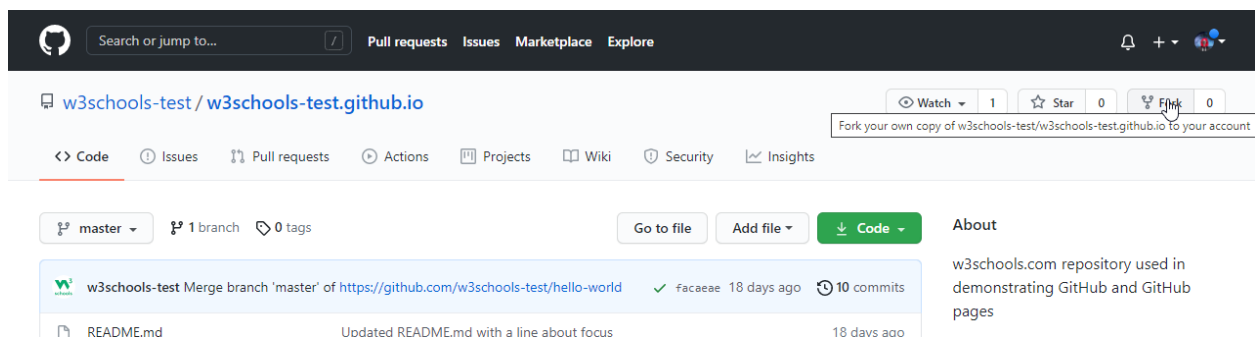
In these next 3 chapters we will show you how to copy a repository, make changes to it, and suggest those changes be implemented to the original repository.

At the end of these chapters, you will have the opportunity to add a message to our public GitHub page: <https://w3schools-test.github.io/>

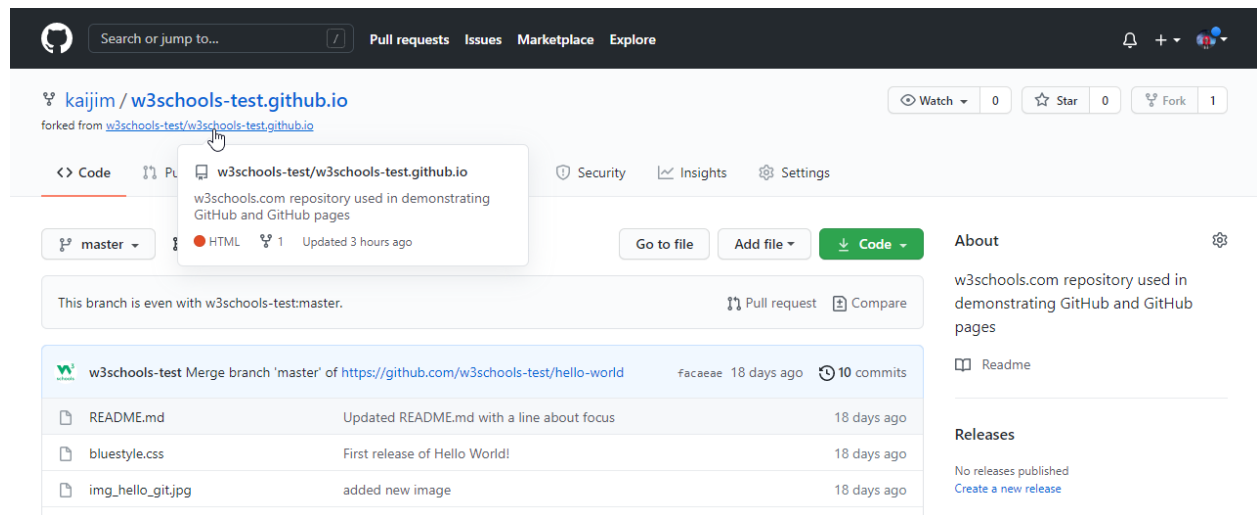
Fork a Repository

A **fork** is a copy of a repository. This is useful when you want to contribute to someone else's project or start your own project based on theirs.

fork is not a command in Git, but something offered in GitHub and other repository hosts. Let's start by logging in to GitHub, and **fork** our repository: <https://github.com/w3schools-test/w3schools-test.github.io>



Now we have our own copy of `w3schools-test.github.io`:



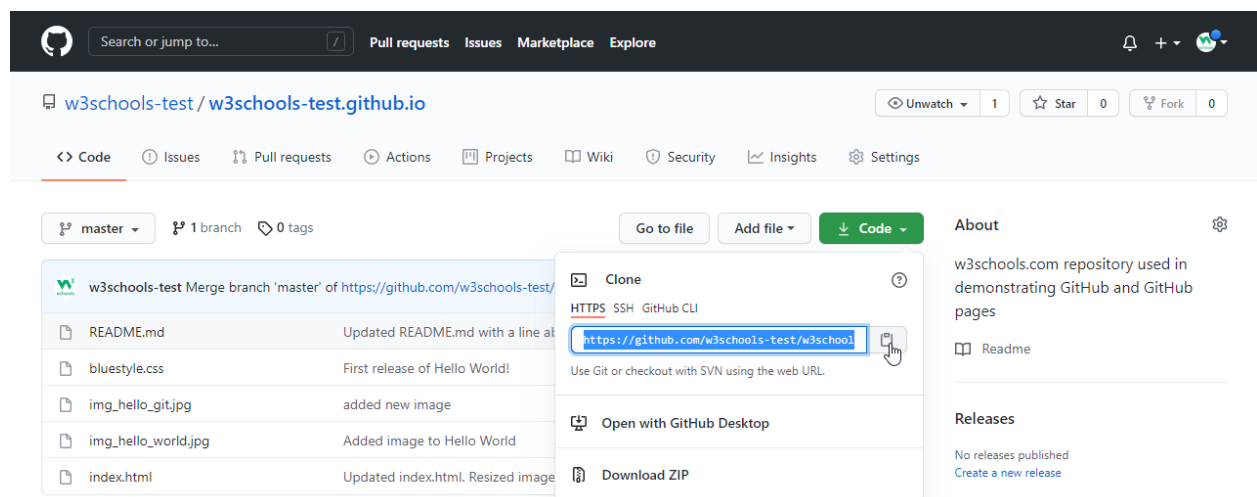
2)Git Clone from GitHub:--->

Clone a Fork from GitHub

Now we have our own `fork`, but only on GitHub. We also want a `clone` on our local Git to keep working on it.

A `clone` is a full copy of a repository, including all logging and versions of files.

Move back to the original repository, and click the green "Code" button to get the `URL` to `clone`:



Open your Git bash and `clone` the repository:

```
Git —> git clone https://github.com/w3schools-test/w3schools-test.github.io
```

Take a look in your file system, and you will see a new directory named after the cloned project:

```
Git—> ls
```

Navigate to the new directory, and check the `status`:

```
Git--> a) cd w3school-test-github.io  
b) git status  
c) git log
```

Configuring Remotes

Basically, we have a full copy of a repository, whose `origin` we are not allowed to make changes to.

Let's see how the `remotes` of this Git is set up:

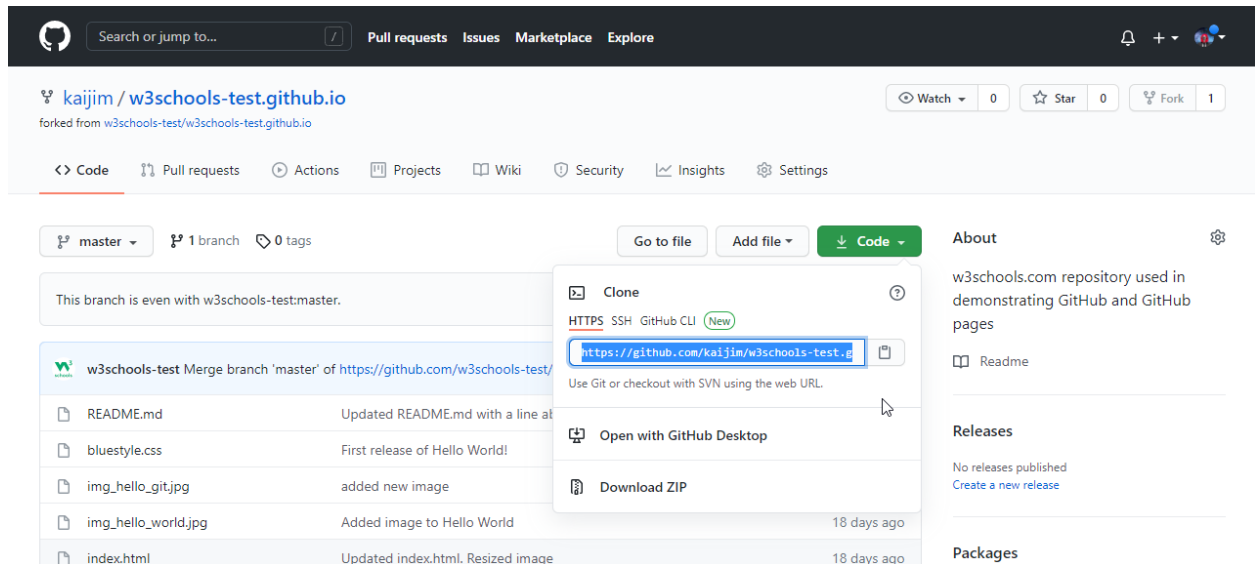
```
git remote -v
```

We see that `origin` is set up to the original "`w3schools-test`" repository, we also want to add our own `fork`.

First, we `rename` the original `origin` remote:

```
git remote rename origin upstream  
git remote -v
```

Then fetch the `URL` of our own `fork`:



And add that as **origin**:

- a) `git remote add origin https://github.com/xyz/xyz.github.io.git`
 - b) `git remote -v`

Now we have 2 remotes:

- **origin** - our own **fork**, where we have read and write access
- **upstream** - the original, where we have read-only access

Now we are going to make some changes to the code. In the next chapter, we will cover how we suggest those changes to the original repository.

3)GitHub Send Pull Request:--->

Push Changes to Our GitHub Fork

We have made a lot of changes to our local Git.

Now we push them to our GitHub fork:

commit the changes:

`git push origin`

Go to GitHub, and we see that the repository has a new commit. And we can send a Pull Request to the original repository:

The screenshot shows the GitHub interface for a repository named 'kaijim / w3schools-test.github.io'. The repository is a fork of 'w3schools-test/w3schools-test.github.io'. The main branch is 'master', which is 1 commit ahead of the upstream 'w3schools-test:master'. A 'Pull request' button is visible, which is the focus of the instruction. Below this, a commit by 'kaijim' is shown, titled 'Changed to serve as a public guestbook', with a commit hash of 'ebb1a5c' and a timestamp of '21 minutes ago'. The commit message is 'Changed to serve as a public guestbook'. The files changed in this commit are 'README.md', 'img_guestbook_wall.jpg', and 'index.html'. The README content is visible, showing a 'Guestbook for w3schools.com Git tutorial' and a message to 'Leave a message for us!'. The right sidebar shows repository statistics: 0 stars, 0 forks, and 1 watch. It also lists 'Releases' (none published) and 'Packages' (none published).

Click that and create a pull request:

The screenshot shows the 'Comparing changes' page on GitHub. The page is titled 'Comparing changes' and provides instructions on how to use the comparison tool. It shows the 'base repository' as 'w3schools-test/w3schools-test' and the 'head repository' as 'kaijim/w3schools-test.github.io'. The 'base' branch is 'master' and the 'compare' branch is 'master'. A green checkmark indicates that the branches are 'Able to merge'. A yellow box contains the text 'Discuss and review the changes in this comparison with others. Learn about pull requests' and a green 'Create pull request' button. Below this, a summary bar shows '1 commit', '6 files changed', '0 comments', and '1 contributor'. The commit history is shown, with a commit by 'kaijim' titled 'Changed to serve as a public guestbook' with a commit hash of 'ebb1a5c'.

Remember to add an explanation for the administrators.

The screenshot shows a GitHub pull request interface. At the top, the repository is identified as `w3schools-test / w3schools-test.github.io`. The pull request title is "Changed to serve as a public guestbook". The description states: "I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html". The pull request is from the `base: master` branch to the `head repository: kajjim/w3schools-test.github.io` branch, with a comparison to `compare: master`. A green checkmark indicates it is "Able to merge". The "Create pull request" button is highlighted with a mouse cursor. Below the description, there is a summary bar showing "1 commit", "6 files changed", "0 comments", and "1 contributor". The commit history shows a single commit on April 13, 2021, titled "Changed to serve as a public guestbook" by user `ebb1a5c`.

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 repository: w3schools-test/w3schools-test... base: master ← head repository: kajjim/w3schools-test.github.io compare: master

✓ Able to merge. These branches can be automatically merged.

Changed to serve as a public guestbook

Write Preview H B I i < > @ ↩ ↪

I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html

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

☒ Allow edits by maintainers ⓘ

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

1 commit 6 files changed 0 comments 1 contributor

Commits on Apr 13, 2021

Changed to serve as a public guestbook ebb1a5c

Pull Request is sent:

GitHub interface showing a Pull Request for repository `w3schools-test / w3schools-test.github.io`.

The Pull Request title is **Changed to serve as a public guestbook #1**. It is from `kaijim` to `w3schools-test:master` from `kaijim:master`.

Summary: `kaijim` wants to merge 1 commit into `w3schools-test:master` from `kaijim:master`.

Statistics: +109 -24 files changed.

Conversation:

- `kaijim` commented now: I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html
- `ebb1a5c` Changed to serve as a public guestbook

Add more commits by pushing to the `master` branch on `kaijim/w3schools-test.github.io`.

Status: **This branch has no conflicts with the base branch**. Only those with [write access](#) to this repository can merge pull requests.

Write / Preview tabs. Rich text editor with options: H, B, I, List, Link, Check, Mention, Image, Undo, Redo.

Leave a comment

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

Buttons: [Close pull request](#), [Comment](#)

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

ProTip! Add comments to specific lines under [Files changed](#).

Right sidebar (Metadata):

- Reviewers:** No reviews. Still in progress? Convert to draft
- Assignees:** No one assigned
- Labels:** None yet
- Projects:** None yet
- Milestone:** No milestone
- Linked issues:** Successfully merging this pull request may close these issues. None yet
- Notifications:** [Unsubscribe](#). You're receiving notifications because you authored the thread.

Approving Pull Requests

Now any member with access can see the Pull Request when they see the original repository:

The screenshot shows the GitHub interface for the repository `w3schools-test / w3schools-test.github.io`. The `Pull requests` tab is active, displaying a single pull request titled "w3schools-test Merge branch 'master' of https://github.com/w3schools-test/hello-world" by user `faceae`, submitted 18 days ago with 10 commits. Below the pull request list, the `README.md` file is open, showing the "hello-world" title and introductory text. The repository statistics on the right indicate 1 star, 0 forks, and 10 commits. The `Environments` section shows `github-pages` as active.

File	Commit Message	Time
README.md	Updated README.md with a line about focus	18 days ago
bluestyle.css	First release of Hello World!	18 days ago
img_hello_git.jpg	added new image	18 days ago
img_hello_world.jpg	Added image to Hello World	18 days ago
index.html	Updated index.html. Resized image	18 days ago

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.

Environments 1

- github-pages Active

Languages

- HTML 81.9%
- CSS 18.1%

And they can see the proposed changes:

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Dismiss

Filters

☐ **Changed to serve as a public guestbook #1**
I removed, added and updated files to make this able to serve as a public guest book....
w3schools-test:master ← kaijim:master

☐ **Changed to serve as a public guestbook**
#1 opened 1 minute ago by kaijim

Labels 9 Milestones 0 [New pull request](#)

Author Label Projects Milestones Reviews Assignee Sort

ProTip! Filter pull requests by the default branch with `base:master`.

Comment on the changes and **merge**:

Changed to serve as a public guestbook #1

[Open](#) kaijim wants to merge 1 commit into `w3schools-test:master` from `kaijim:master`

Conversation 0 Commits 1 Checks 0 Files changed 6 +109 -24

kaijim commented 2 minutes ago [First-time contributor](#)

I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html

Changed to serve as a public guestbook ebb1a5c

w3schools-test commented now [Owner](#)

This looks good! Changes approved!

Add more commits by pushing to the `master` branch on `kaijim/w3schools-test.github.io`.

[This branch has not been deployed](#)
No deployments

[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](#).

Reviewers

Suggestions

[w3schools-test](#) [Request](#)

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](#)

Confirm:

The screenshot shows a GitHub pull request interface. At the top, the repository name is `w3schools-test / w3schools-test.github.io`. The pull request title is "Changed to serve as a public guestbook #1". The pull request is from the `kaijim:master` branch to the `w3schools-test:master` branch. The pull request is open and has 1 commit. The pull request description is "I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html". The pull request is approved by the owner, `w3schools-test`, with the comment "This looks good! Changes approved!". The pull request is ready to be merged. The merge confirmation dialog is open, showing the merge title "Merge pull request #1 from kaijim/master" and the commit message "Changed to serve as a public guestbook". The dialog has a green "Confirm merge" button and a grey "Cancel" button. The right sidebar shows the pull request details, including the reviewer `w3schools-test` and the assignees, labels, projects, milestones, and linked issues.

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

w3schools-test / w3schools-test.github.io Unwatch 1 Star 0 Fork 1

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Changed to serve as a public guestbook #1

Edit Open with

Open kaijim wants to merge 1 commit into w3schools-test:master from kaijim:master

Conversation 0 Commits 1 Checks 0 Files changed 6 +109 -24

kaijim commented 3 minutes ago First-time contributor

I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html

Changed to serve as a public guestbook ebb1a5c

w3schools-test commented 34 seconds ago Owner

This looks good! Changes approved!

Add more commits by pushing to the master branch on kaijim/w3schools-test.github.io.

Merge pull request #1 from kaijim/master

Changed to serve as a public guestbook

Confirm merge Cancel

Reviewers Suggestions w3schools-test Request

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

And changes have been merged with master:

The screenshot shows a GitHub pull request interface. At the top, the repository is `w3schools-test / w3schools-test.github.io`. The pull request title is "Changed to serve as a public guestbook #1". A purple "Merged" badge is visible. The merge summary states: "w3schools-test merged 1 commit into w3schools-test:master from kaijim:master". Below this, a conversation thread shows a comment from `kaijim` (Contributor) stating: "I removed, added and updated files to make this able to serve as a public guest book. Future changes should be only to index.html". This is followed by a commit message "Changed to serve as a public guestbook" by `ebb1a5c`. Then, a comment from `w3schools-test` (Owner) states: "This looks good! Changes approved!". At the bottom, a merge summary shows: "w3schools-test merged commit #745935 into w3schools-test:master now". On the right side, there are sections for "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), and "Milestone" (No milestone). A "Revert" button is located at the bottom right of the merge summary.

Git Advanced

1)Git .gitignore:--->

Git Ignore

When sharing your code with others, there are often files or parts of your project, you do not want to share.

Examples

- log files
- temporary files
- hidden files
- personal files
- etc.

Git can specify which files or parts of your project should be ignored by Git using a `.gitignore` file.

Git will not track files and folders specified in `.gitignore`. However, the `.gitignore` file itself IS tracked by Git.

Create .gitignore

To create a `.gitignore` file, go to the root of your local Git, and create it:

```
touch .gitignore
```

Now open the file using a text editor.

We are just going to add two simple rules:

- Ignore any files with the `.log` extension
- Ignore everything in any directory named `temp`

Example:-

```
# ignore ALL .log files
*.log

# ignore ALL files in ANY directory named temp
temp/
```

Now all **.log** files and anything in **temp** folders will be ignored by Git.

Pattern	Explanation/Matches	Examples
	Blank lines are ignored	
<i># text comment</i>	Lines starting with # are ignored	
<i>name</i>	All <i>name</i> files, <i>name</i> folders, and files and folders in any <i>name</i> folder	/name.log /name/file.txt /lib/name.log
<i>name/</i>	Ending with / specifies the pattern is for a folder. Matches all files and folders in any <i>name</i> folder	/name/file.txt /name/log/name.log no match: /name.log
<i>name.file</i>	All files with the <i>name.file</i>	/name.file /lib/name.file
<i>/name.file</i>	Starting with / specifies the pattern matches only files in the root folder	/name.file no match: /lib/name.file
<i>lib/name.file</i>	Patterns specifying files in specific folders are always relative to root (even if you do not start with /)	/lib/name.file no match: name.file /test/lib/name.file
<i>**/lib/name.file</i>	Starting with ** before / specifies that it matches any folder in the repository. Not just on root.	/lib/name.file /test/lib/name.file
<i>**/name</i>	All <i>name</i> folders, and files and folders in any <i>name</i> folder	/name/log.file /lib/name/log.file /name/lib/log.file
<i>/lib/**/name</i>	All <i>name</i> folders, and files and folders in any <i>name</i> folder within the lib folder.	/lib/name/log.file /lib/test/name/log.file /lib/test/ver1/name/log.file no match: /name/log.file
<i>*.file</i>	All files with the <i>.file</i> extension	/name.file /lib/name.file
<i>*name/</i>	All folders ending with <i>name</i>	/lastname/log.file /firstname/log.file
<i>name?.file</i>	? matches a single non-specific character	/names.file /name1.file no match: /names1.file

<i>name</i> [a-z]. <i>file</i>	[<i>range</i>] matches a single character in the specified range (in this case a character in the range of a-z, and also be numeric.)	/names. <i>file</i> /nameb. <i>file</i> no match: /name1. <i>file</i>
<i>name</i> [abc]. <i>file</i>	[<i>set</i>] matches a single character in the specified set of characters (in this case either a, b, or c)	/namea. <i>file</i> /nameb. <i>file</i> no match: /names. <i>file</i>
<i>name</i> [!abc]. <i>file</i>	[! <i>set</i>] matches a single character, except the ones specified in the set of characters (in this case a, b, or c)	/names. <i>file</i> /namex. <i>file</i> no match: /namesb. <i>file</i>
*. <i>file</i>	All files with the . <i>file</i> extension	/name. <i>file</i> /lib/name. <i>file</i>
<i>name</i> / ! <i>name</i> /secret.log	! specifies a negation or exception. Matches all files and folders in any <i>name</i> folder, except name/secret.log	/name/ <i>file</i> .txt /name/log/name.log no match: /name/secret.log
*. <i>file</i> ! <i>name</i> . <i>file</i>	! specifies a negation or exception. All files with the . <i>file</i> extension, except name. <i>file</i>	/log. <i>file</i> /lastname. <i>file</i> no match: /name. <i>file</i>
. <i>file</i> ! <i>name</i> /. <i>file</i> junk.*	Adding new patterns after a negation will re-ignore a previous negated file All files with the . <i>file</i> extension, except the ones in <i>name</i> folder. Unless the file name is junk	/log. <i>file</i> /name/log. <i>file</i> no match: /name/junk. <i>file</i>

Local and Personal Git Ignore Rules

It is also possible to ignore files or folders but not show it in the distributed `.gitignore` file.

These kinds of ignores are specified in the `.git/info/exclude` file. It works the same way as `.gitignore` but are not shown to anyone else.

2)Git Security SSH:--->

Git Security

Up to this point, we have used HTTPS to connect to our remote repository.

HTTPS will usually work just fine, but you should use SSH if you work with unsecured networks. And sometimes, a project will require that you use SSH.

What is SSH

SSH is a secure shell network protocol that is used for network management, remote file transfer, and remote system access.

SSH uses a pair of SSH keys to establish an authenticated and encrypted secure network protocol. It allows for secure remote communication on unsecured open networks.

SSH keys are used to initiate a secure "handshake". When generating a set of keys, you will generate a "public" and "private" key.

The "public" key is the one you share with the remote party. Think of this more as the lock.

The "private" key is the one you keep for yourself in a secure place. Think of this as the key to the lock.

SSH keys are generated through a security algorithm. It is all very complicated, but it uses prime numbers, and large random numbers to make the public and private key.

It is created so that the public key can be derived from the private key, but not the other way around.

Generating an SSH Key Pair

In the command line for Linux, Apple, and in the Git Bash for Windows, you can generate an SSH key.

Let's go through it, step by step.

Start by creating a new key, using your email as a label:

Eg:-

```
ssh-keygen -t rsa -b 4096 -C "test@w3schools.com"
```

You will be prompted with the following through this creation:

```
Enter file in which to save the key (/c/Users/user/.ssh/id_rsa):
```

Select a file location, or press "Enter" to use the default file location.

```
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:
```

Entering a secure passphrase will create an additional layer of security. Preventing anyone who gains access to the computer to use that key without the passphrase. However, it will require you to supply the passphrase anytime the SSH key is used.

Now we add this SSH key pair to the SSH-Agent (using the file location from above):

Eg:-

```
ssh-add /Users/user/.ssh/id_rsa  
Enter passphrase for /Users/user/.ssh/id_rsa:  
Identity added: /Users/user/.ssh/id_rsa (test@w3schools.com)
```

You will be prompted to supply the passphrase if you added one.

Now the SSH key pair is ready to use.

3)GitHub Add SSH:--->

Copy the SSH Public Key

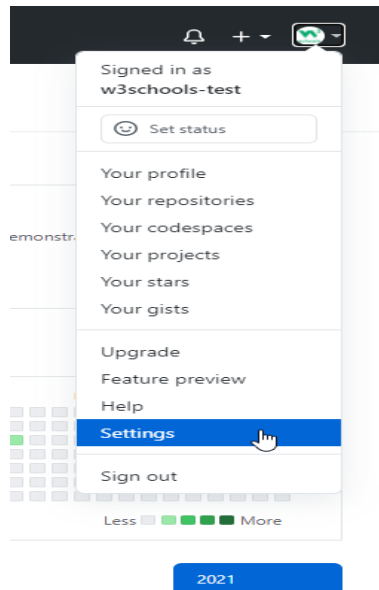
In the previous chapter, we created an SSH key pair.

Now we will use the `clip <` command to copy the public key to our clipboard:

Eg:-

```
clip < /Users/user/.ssh/id_rsa.pub
```

Go to GitHub, navigate to the top left corner, click your profile, and select: Settings:



Then select "SSH and GPG keys". and click the "New SSH key" button:

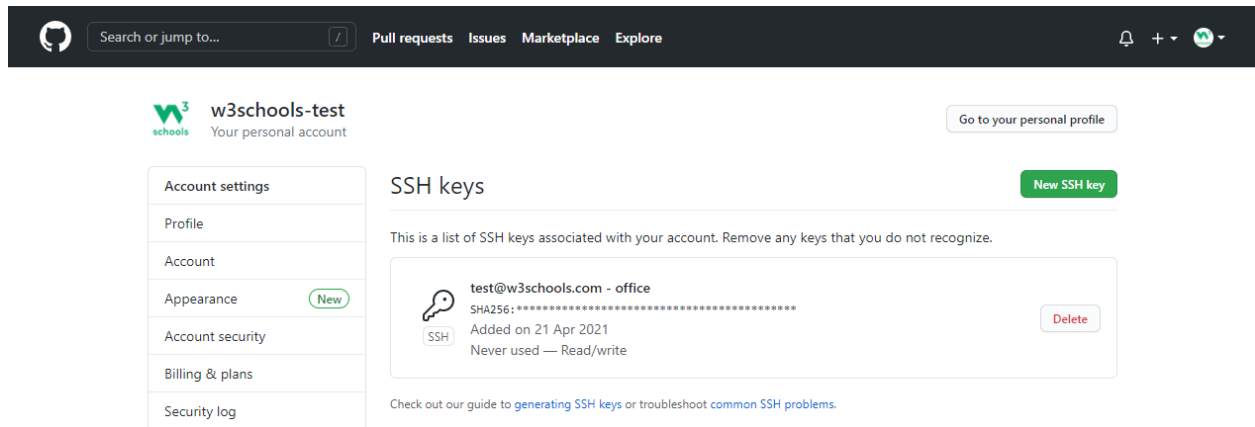
The screenshot shows the GitHub account settings page for 'w3schools-test'. The left sidebar contains a list of settings: Account settings, Profile, Account, Appearance (marked 'New'), Account security, Billing & plans, Security log, Security & analysis, Emails, Notifications, SSH and GPG keys (highlighted with a red bar), and Repositories. The main content area is titled 'SSH keys' and states 'There are no SSH keys associated with your account.' with a link to a guide. A green button 'New SSH key' is visible. Below this is the 'GPG keys' section, also stating 'There are no GPG keys associated with your account.' with a 'New GPG key' button. A green arrow originates from the 'SSH and GPG keys' menu item in the sidebar and points to the 'New SSH key' button.

Select a title, and paste the public SSH key into the "Key" field, and click "Add SSH Key":

The screenshot shows the 'Add new SSH key' form in the GitHub account settings. The left sidebar is the same as the previous screenshot, with 'SSH and GPG keys' highlighted. The main content area is titled 'SSH keys / Add new'. It has a 'Title' field with the value 'test@w3schools.com - office'. Below it is the 'Key' field, which contains a public SSH key: 'ssh-rsa' followed by a long string of asterisks. At the bottom of the form is a green button labeled 'Add SSH key'.

You will be prompted to supply your GitHub password.

You will see your new SSH key added:



Test SSH Connection to GitHub

Now we can test our connection via SSH to GitHub:

```
ssh -T git@github.com
```

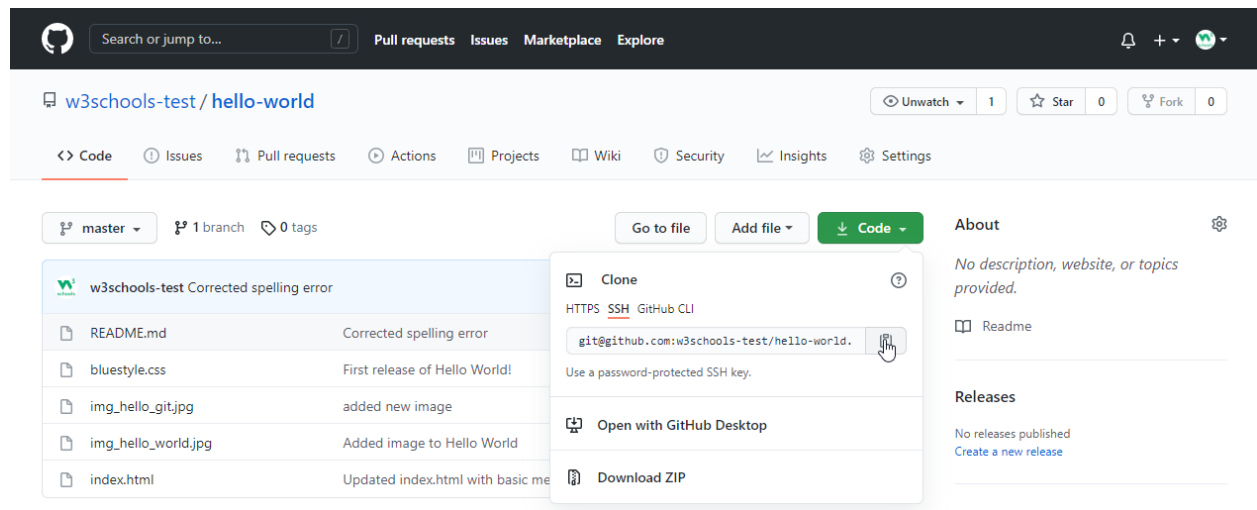
```
Output:- The authenticity of host 'github.com (140.82.121.3) '
can't be established. RSA key fingerprint is
SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8. Are you sure
you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.121.3' (RSA) to the
list of known hosts. Hi w3schools-test! You've successfully
authenticated, but GitHub does not provide shell access.
```

If the last line contains your username on GitHub, you are successfully authenticated!

Add New GitHub SSH Remote

Now we can add a new remote via SSH to our Git.

First, get the SSH address from our repository on GitHub:



Then use that address to add a new origin:

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

Note: You can change a remote origin from HTTPS to SSH with the command:

```
git remote set-url remote-name git@github.com:username/repository.git
```

```
git remote set-url origin git@github.com:w3schools-test/hello-world.git
```

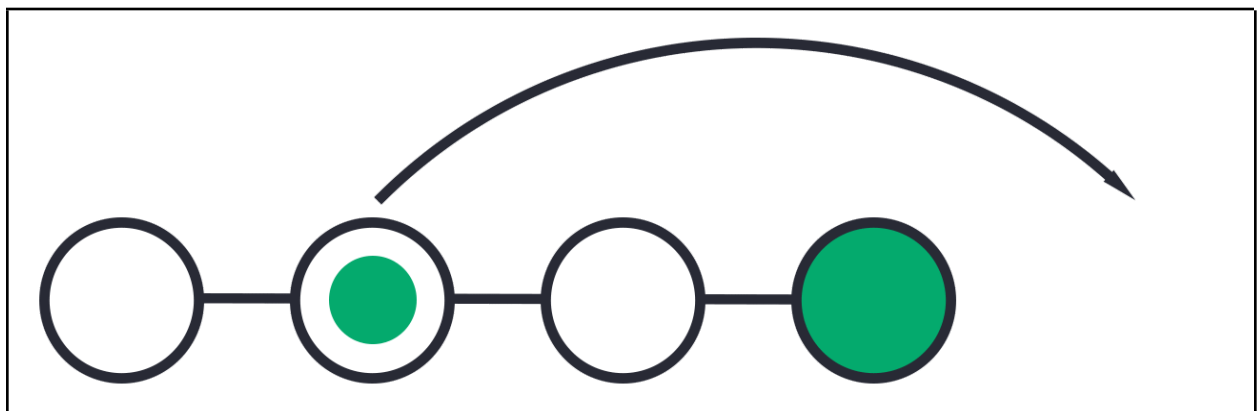
Git Undo

1) Git Revert:--->

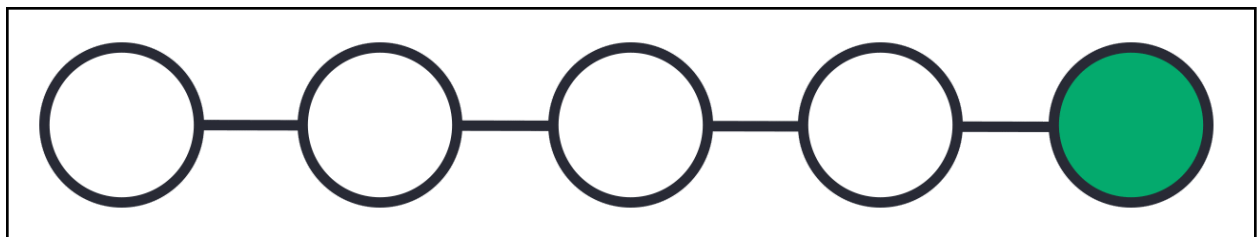
Git Revert

`revert` is the command we use when we want to take a previous `commit` and add it as a new `commit`, keeping the `log` intact.

Step 1: Find the previous `commit`:



Step 2: Use it to make a new `commit`:



Let's make a new `commit`, where we have "accidentally" deleted a file:

Eg:-

```
git commit -m "Just a regular update, definitely no accidents here..."
```

Now we have a part in our `commit` history we want to go back to. Let's try and do that with `revert`.

Git Revert Find Commit in Log

First thing, we need to find the point we want to return to. To do that, we need to go through the `log`.

To avoid the very long log list, we are going to use the `--oneline` option, which gives just one line per commit showing:

- The first seven characters of the `commit hash`
- the `commit message`

So let's find the point we want to `revert`:

```
git log - - oneline
```

We want to revert to the previous `commit: 52418f7 (HEAD -> master)` Just a regular update, definitely no accidents here..., and we see that it is the latest `commit`.

Git Revert HEAD

We revert the latest `commit` using `git revert HEAD` (`revert` the latest change, and then `commit`), adding the option `--no-edit` to skip the commit message editor (getting the default `revert` message):

```
a) git revert HEAD - - no-edit  
b) git log - - oneline
```

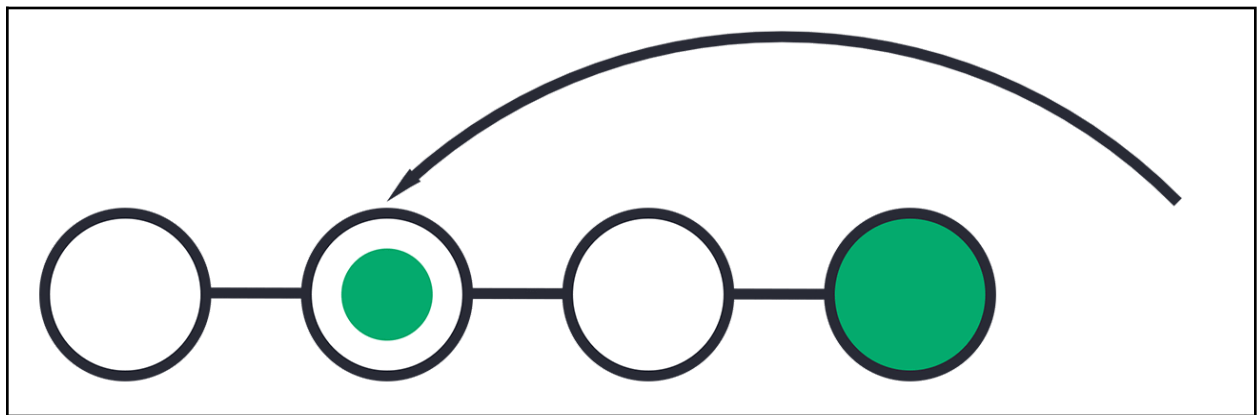

We want to revert to the previous `commit: 52418f7` (`HEAD -> master`) Just a regular update, definitely no accidents here..., and we see that it is the latest `commit`.

2) Git Reset:--->

Git Reset

`reset` is the command we use when we want to move the repository back to a previous `commit`, discarding any changes made after that `commit`.

Step 1: Find the previous `commit`:



Step 2: Move the repository back to that step:



After the previous chapter, we have a part in our `commit` history we could go back to. Let's try and do that with `reset`.

Git Reset Find Commit in Log

First thing, we need to find the point we want to return to. To do that, we need to go through the `log`.

To avoid the very long `log` list, we are going to use the `--oneline` option, which gives just one line per `commit` showing:

- The first seven characters of the `commit hash` - this is what we need to refer to in our reset command.
- the `commit message`

So let's find the point we want to `reset` to:

```
a) git log - - oneline
```

We want to return to the `commit: 9a9add8 (origin/master) Added .gitignore`, the last one before we started to mess with things.

Git Reset

We `reset` our repository back to the specific commit using `git reset commithash` (`commithash` being the first 7 characters of the commit hash we found in the `log`):

```
git reset 9a9add8
```

Now let's check the `log` again:

```
git log - - oneline
```

Warning: Messing with the `commit` history of a repository can be dangerous. It is usually ok to make these kinds of changes to your own local repository. However, you should avoid making changes that rewrite history to `remote` repositories, especially if others are working with them.

Git Undo Reset

Even though the commits are no longer showing up in the `log`, it is not removed from Git.

```
git reset e56ba1f
```

Now let's check the `log` again:

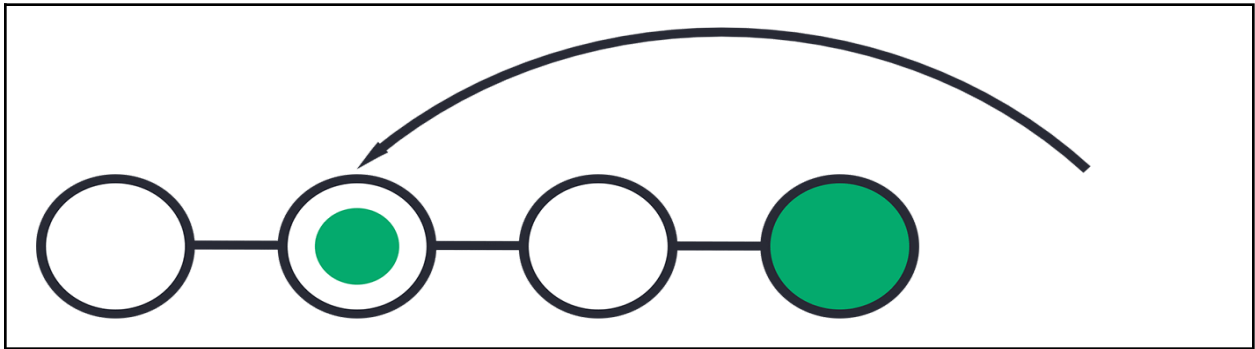
```
git log - - oneline
```

3) Git Amend:--->

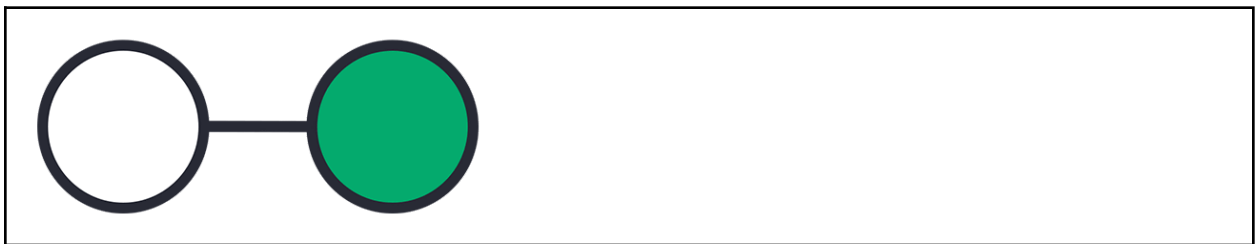
Git Reset

`reset` is the command we use when we want to move the repository back to a previous `commit`, discarding any changes made after that `commit`.

Step 1: Find the previous `commit`:



Step 2: Move the repository back to that step:



After the previous chapter, we have a part in our `commit` history we could go back to. Let's try and do that with `reset`.

Git Reset Find Commit in Log

First thing, we need to find the point we want to return to. To do that, we need to go through the `log`.

To avoid the very long `log` list, we are going to use the `--oneline` option, which gives just one line per `commit` showing:

- The first seven characters of the `commit hash` - this is what we need to refer to in our reset command.
- the `commit message`

So let's find the point we want to `reset` to:

Eg:-

```
git log - - oneline
```

We want to return to the `commit: 9a9add8 (origin/master) Added .gitignore`, the last one before we started to mess with things.

Git Reset

We `reset` our repository back to the specific commit using `git reset commithash` (`commithash` being the first 7 characters of the commit hash we found in the `log`):

Eg:-

```
git reset 9a9add8
```

Now let's check the `log` again:

```
git log - - oneline
```

Git Undo Reset

Even though the commits are no longer showing up in the `log`, it is not removed from Git.

If you know the commit hash you can `reset` to it:

```
Git reset e56ba1f
```

Now let's check the `log` again:

```
git log --oneline
```

Git - Cheat Sheet Developed by [Jyotirmay Chowdhury](#) 🍷

Content Credit to w3Schools