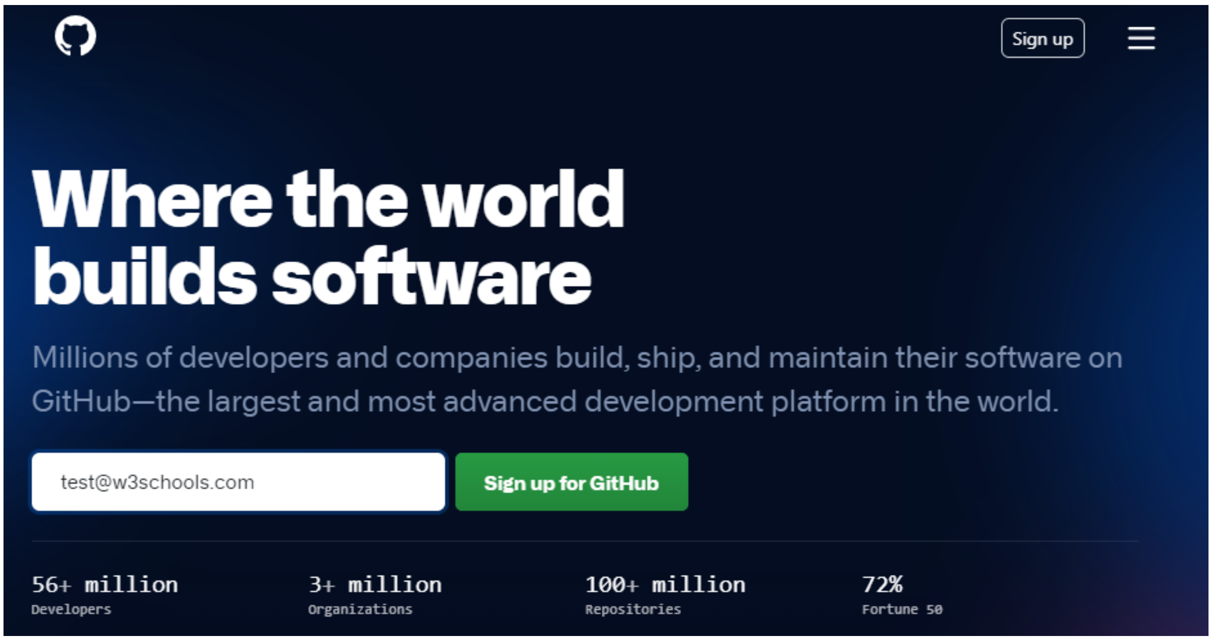
GitHub Account

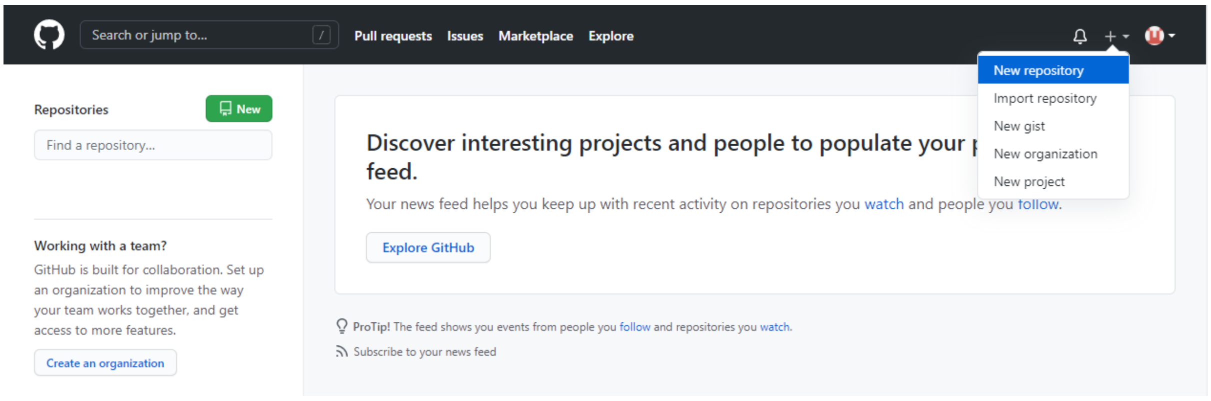
Go to GitHub and sign up for an account:


Sign up
Where the world builds software
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.
test@w3schools.com
Sign up for GitHub
56+ million
Developers
3+ million
Organizations
100+ million
Repositories
72%
Fortune 50

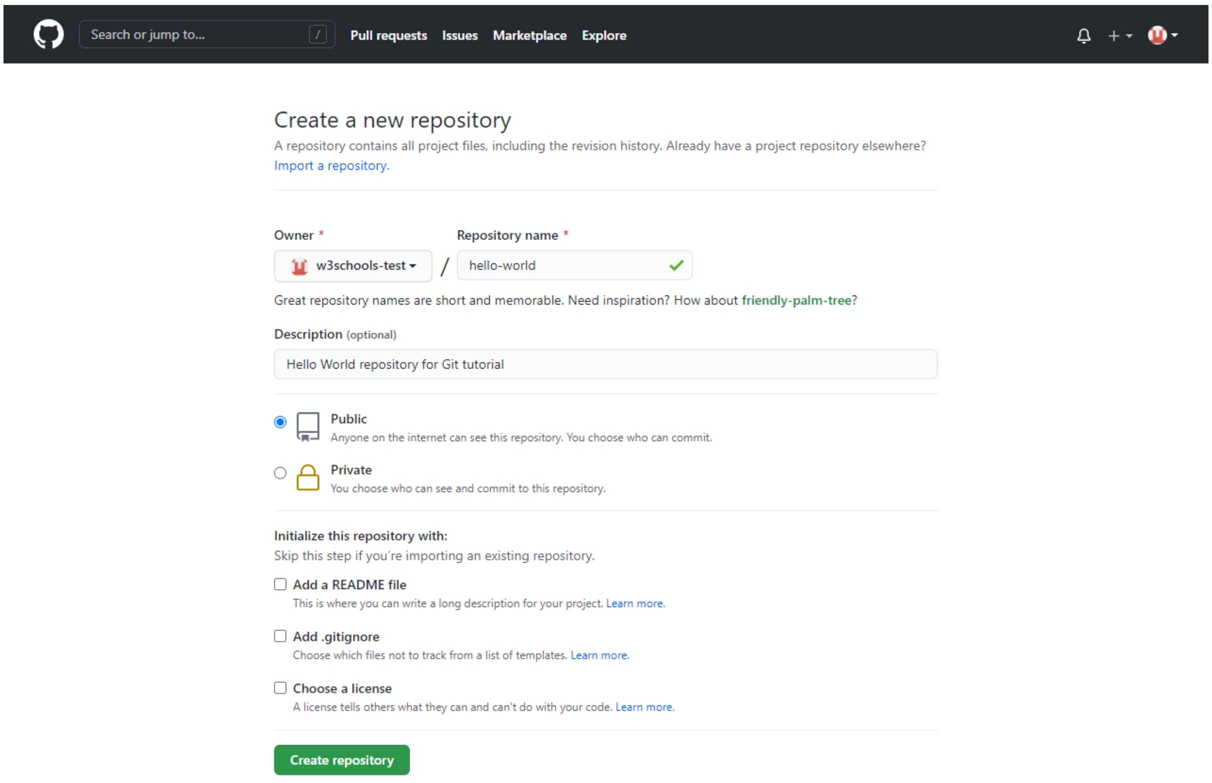
Note: Remember to use the same e-mail address you used in the Git config.

Create a Repository on GitHub

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


Search or jump to...
Pull requests
Issues
Marketplace
Explore
New repository
Import repository
New gist
New organization
New project
Repositories
New
Find a repository...
Discover interesting projects and people to populate your feed.
Your news feed helps you keep up with recent activity on repositories you watch and people you follow.
Explore GitHub
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.
Create an organization
ProTip! The feed shows you events from people you follow and repositories you watch.
Subscribe to your news feed

And fill in the relevant details:



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.

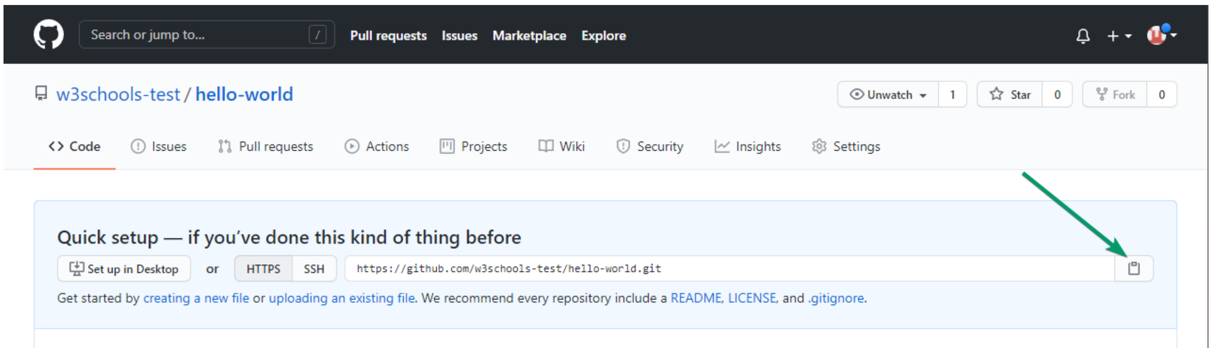
Create repository

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:



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:

Example

```
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:

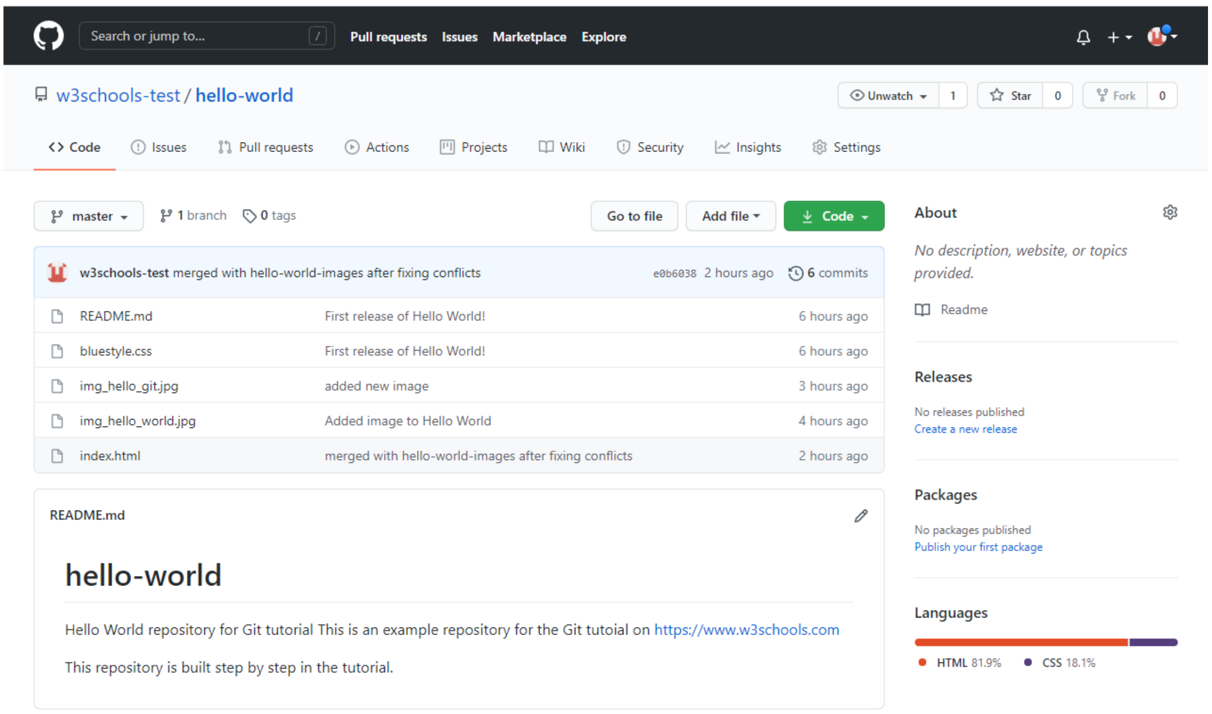
Example

```
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'.
```

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

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.

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

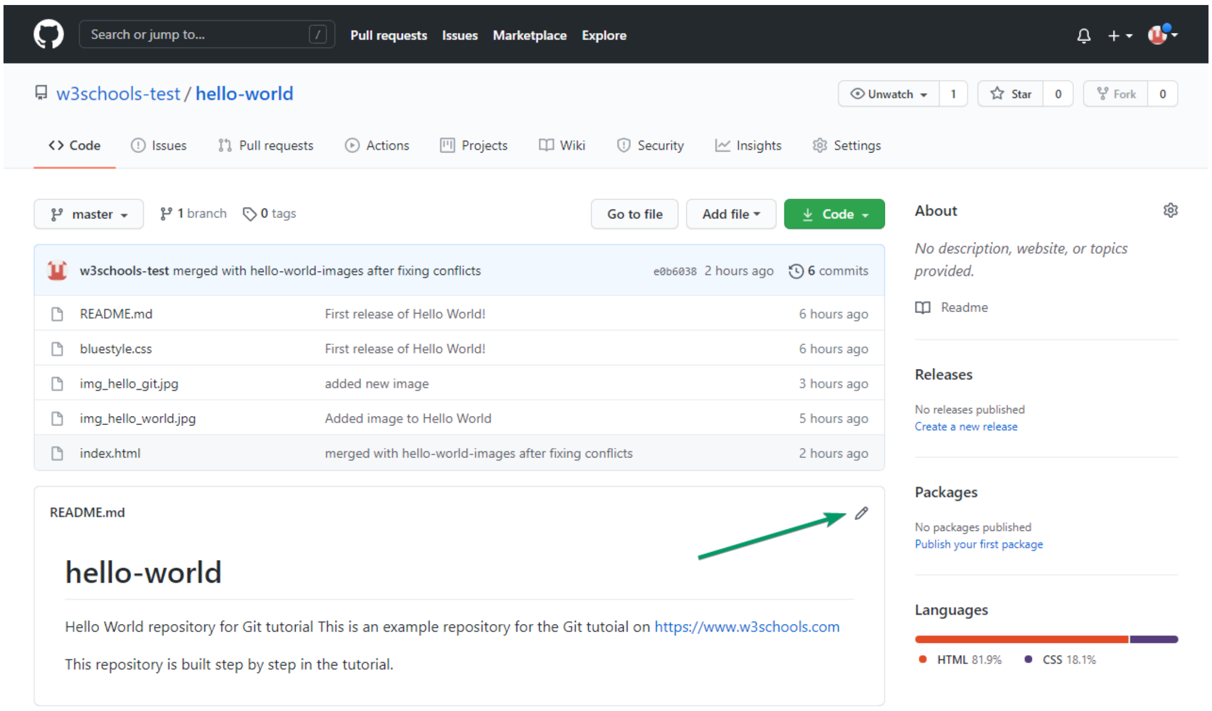
Languages

● HTML 81.9% ● CSS 18.1%

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 / hello-world

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.

About

No description, website, or topics provided.

Releases

No releases published
Create a new release

Packages

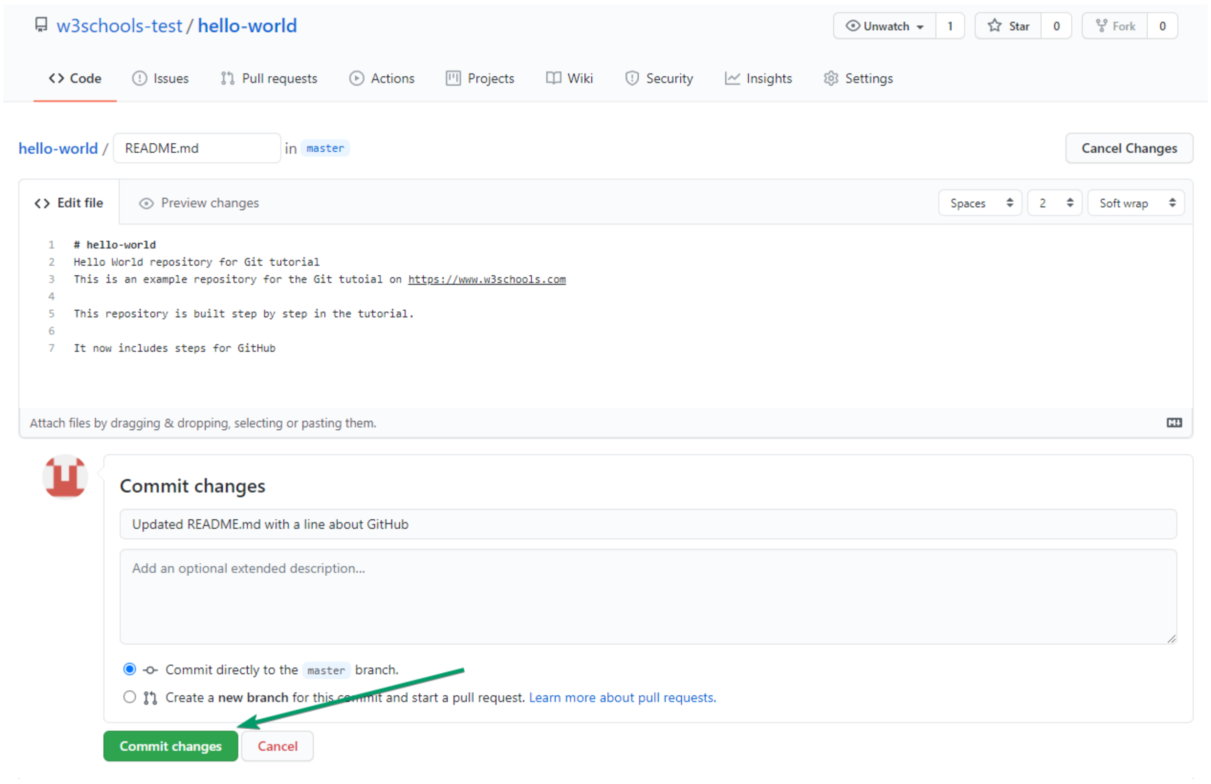
No packages published
Publish your first package

Languages

● 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".

Remember to add a description for the `commit`:


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.

That is how you edit code directly in 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
```

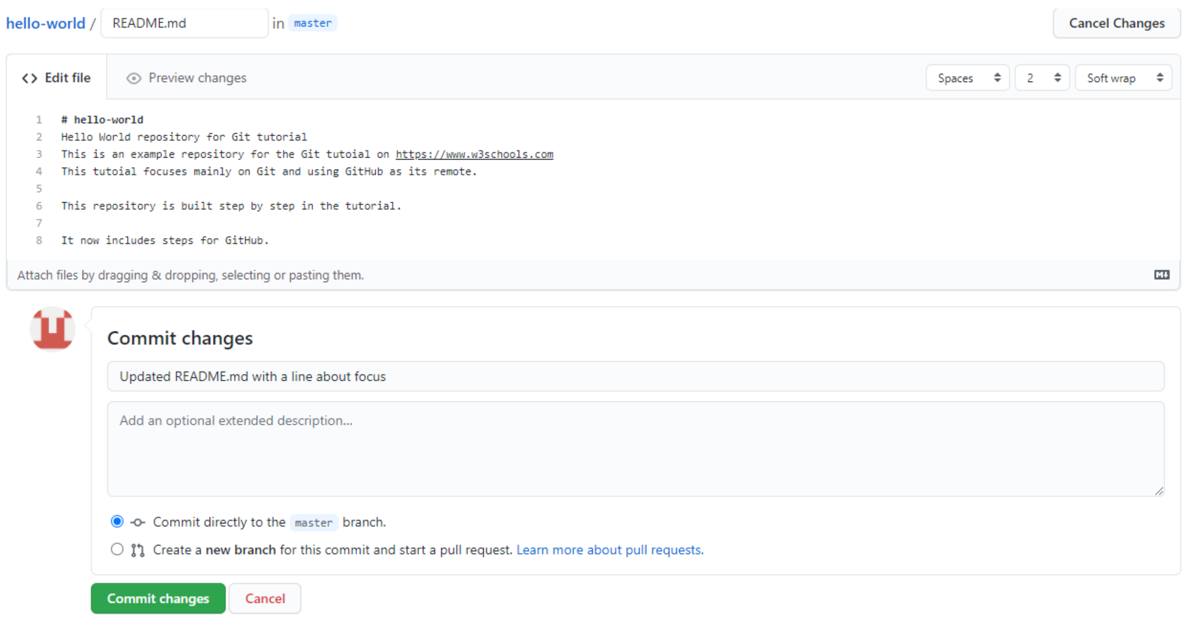
There! Your local git is up to date!

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.



```
# 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.
```

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

Commit changes

Updated README.md with a line about focus

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.

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. In the next chapter, we will look closer at how `push` works on GitHub.

What is Git?

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

It is used for:

Tracking code changes

Tracking who made changes

Coding collaboration

What does Git do?

Manage projects with Repositories

Clone a project to work on a local copy

Control and track changes with Staging and Committing

Branch and Merge to allow for work on different parts and versions of a project

Pull the latest version of the project to a local copy

Push local updates to the main project

Working with Git:

Initialize Git on a folder, making it a Repository

Git now creates a hidden folder to keep track of changes in that folder

When a file is changed, added or deleted, it is considered modified

You select the modified files you want to Stage

The Staged files are Committed, which prompts Git to store a permanent snapshot of the files

Git allows you to see the full history of every commit.

You can revert back to any previous commit.

Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

Change Platform:

Shift focus to GitHubGitHub

Shift focus to BitbucketBitbucket

Shift focus to GitLabGitLab

Why Git?

Over 70% of developers use Git!

Developers can work together from anywhere in the world.

Developers can see the full history of the project.

Developers can revert to earlier versions of a project.

What is GitHub?

Git is not the same as GitHub.

GitHub makes tools that use Git.

GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

In this tutorial, we will focus on using Git with GitHub.

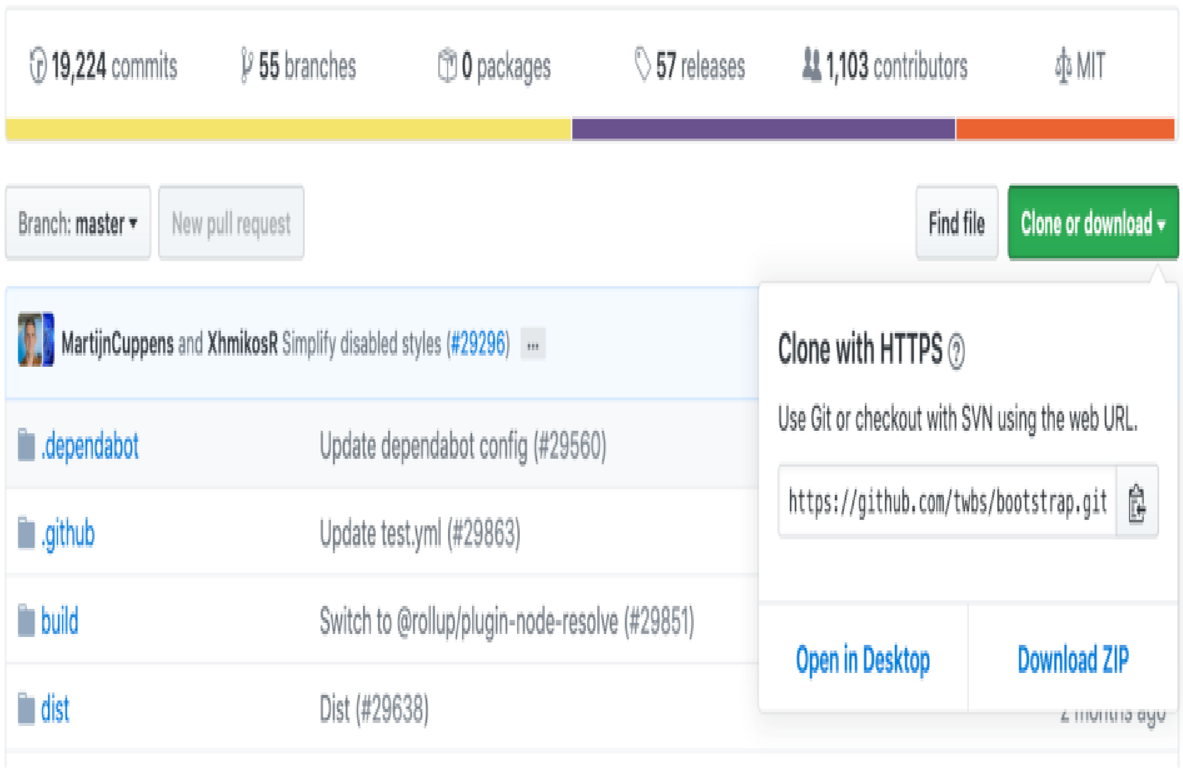
1. Git clone

Git clone is a command for downloading existing source code from a remote repository (like Github, for example). In other words, Git clone basically makes an identical copy of the latest version of a project in a repository and saves it to your computer.

There are a couple of ways to download the source code, but mostly I prefer the **clone with https** way:

```
git clone <https://name-of-the-repository-link>
```

For example, if we want to download a project from Github, all we need to do is click on the green button (clone or download), copy the URL in the box and paste it after the git clone command that I've shown right above.



MartijnCuppens and XhmikosR Simplify disabled styles (#29296)

.dependabot	Update dependabot config (#29560)
.github	Update test.yml (#29863)
build	Switch to @rollup/plugin-node-resolve (#29851)
dist	Dist (#29638)

Use Git or checkout with SVN using the web URL.

```
https://github.com/twbs/bootstrap.git
```

Bootstrap source code example from Github

This will make a copy of the project to your local workspace so you can start working with it.

2. Git branch

Branches are highly important in the git world. By using branches, several developers are able to work in parallel on the same project simultaneously. We can use the git branch command for creating, listing and deleting branches.

Creating a new branch:

```
git branch <branch-name>
```

This command will create a branch **locally**. To push the new branch into the remote repository, you need to use the following command:

```
git push -u <remote> <branch-name>
```

Viewing branches:

```
git branch or git branch --list
```

Deleting a branch:

```
git branch -d <branch-name>
```

3. Git checkout

This is also one of the most used Git commands. To work in a branch, first you need to switch to it. We use **git checkout** mostly for switching from one branch to another. We can also use it for checking out files and commits.

```
git checkout <name-of-your-branch>
```

There are some steps you need to follow for successfully switching between branches:

- The changes in your current branch must be committed or stashed before you switch
- The branch you want to check out should exist in your local

There is also a shortcut command that allows you to create and switch to a branch at the same time:

```
git checkout -b <name-of-your-branch>
```

This command creates a new branch in your local (-b stands for branch) and checks the branch out to new right after it has been created.

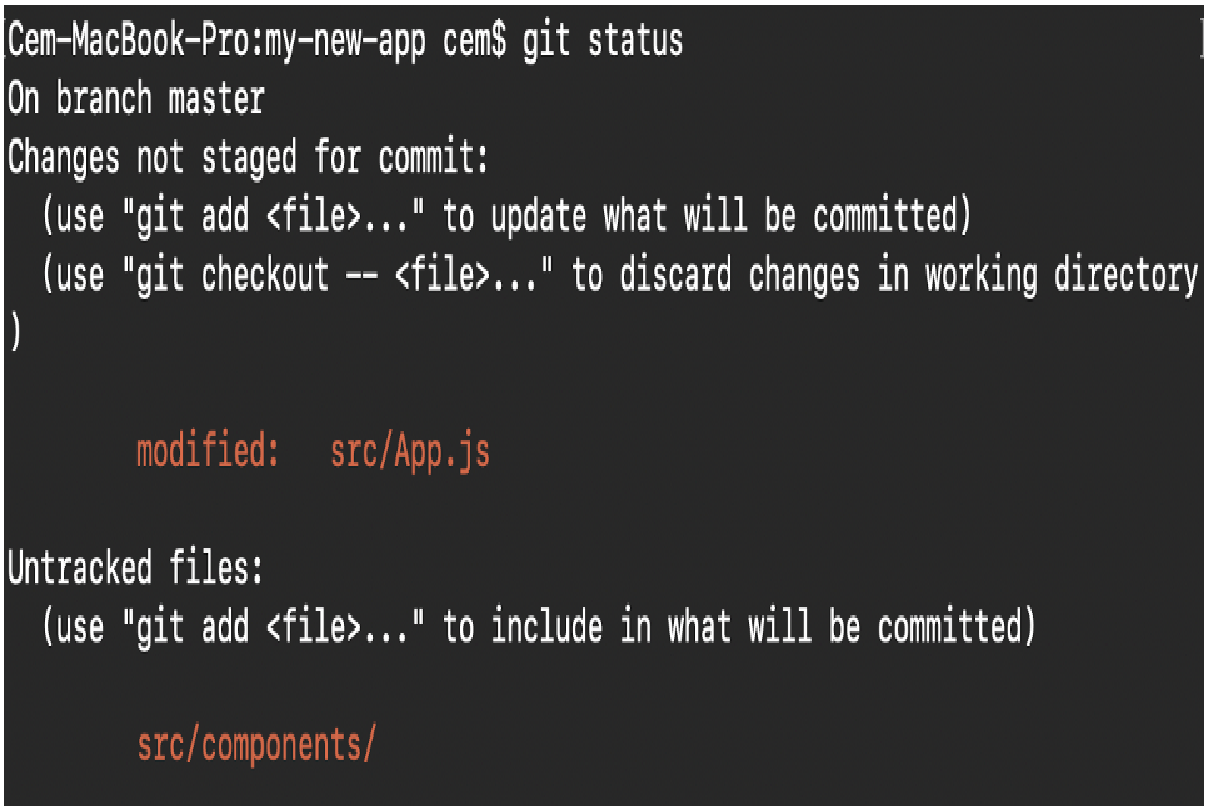
4. Git status

The Git status command gives us all the necessary information about the current branch.

```
git status
```

We can gather information like:

- Whether the current branch is up to date
- Whether there is anything to commit, push or pull
- Whether there are files staged, unstaged or untracked
- Whether there are files created, modified or deleted



```
Cem-MacBook-Pro:my-new-app cem$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory
)

        modified:   src/App.js

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

        src/components/
```

Git status gives information about the branch & files

5. Git add

When we create, modify or delete a file, these changes will happen in our local and won't be included in the next commit (unless we change the configurations).

We need to use the git add command to include the changes of a file(s) into our next commit.

To add a single file:

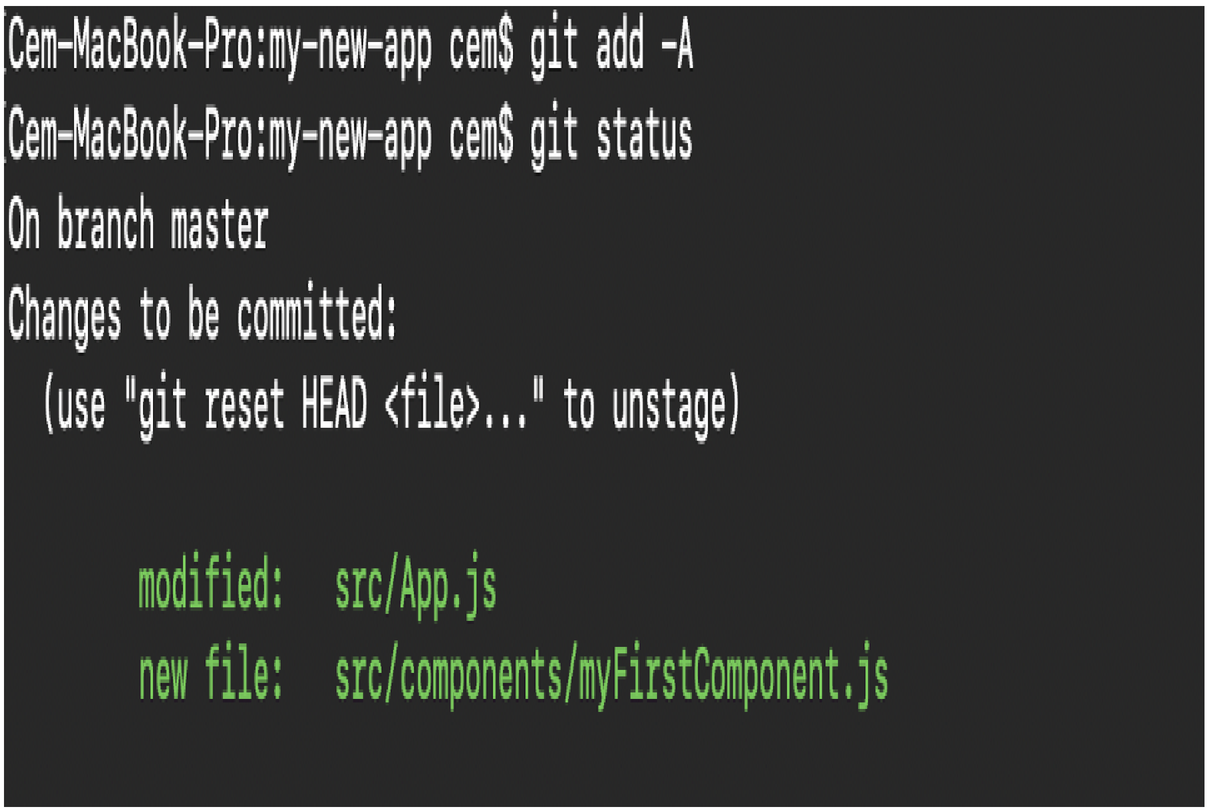
```
git add <file>
```

To add everything at once:

```
git add -A
```

When you visit the screenshot above in the 4th section, you will see that there are file names that are red - this means that they're unstaged files. The unstaged files won't be included in your commits.

To include them, we need to use git add:



```
Cem-MacBook-Pro:my-new-app cem$ git add -A
Cem-MacBook-Pro:my-new-app cem$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/App.js
        new file:   src/components/myFirstComponent.js
```

Files with green are now staged with git add

Important: The git add command doesn't change the repository and the changes are not saved until we use git commit.

6. Git commit

This is maybe the most-used command of Git. Once we reach a certain point in development, we want to save our changes (maybe after a specific task or issue).

Git commit is like setting a checkpoint in the development process which you can go back to later if needed.

We also need to write a short message to explain what we have developed or changed in the source code.

```
git commit -m "commit message"
```

Important: Git commit saves your changes only locally.

7. Git push

After committing your changes, the next thing you want to do is send your changes to the remote server. Git push uploads your commits to the remote repository.

```
git push <remote> <branch-name>
```

However, if your branch is newly created, then you also need to upload the branch with the following command:

```
git push --set-upstream <remote> <name-of-your-branch>
```

or

```
git push -u origin <branch_name>
```

Important: Git push only uploads changes that are committed.

8. Git pull

The **git pull** command is used to get updates from the remote repo. This command is a combination of **git fetch** and **git merge** which means that, when we use git pull, it gets the updates from remote repository (git fetch) and immediately applies the latest changes in your local (git merge).

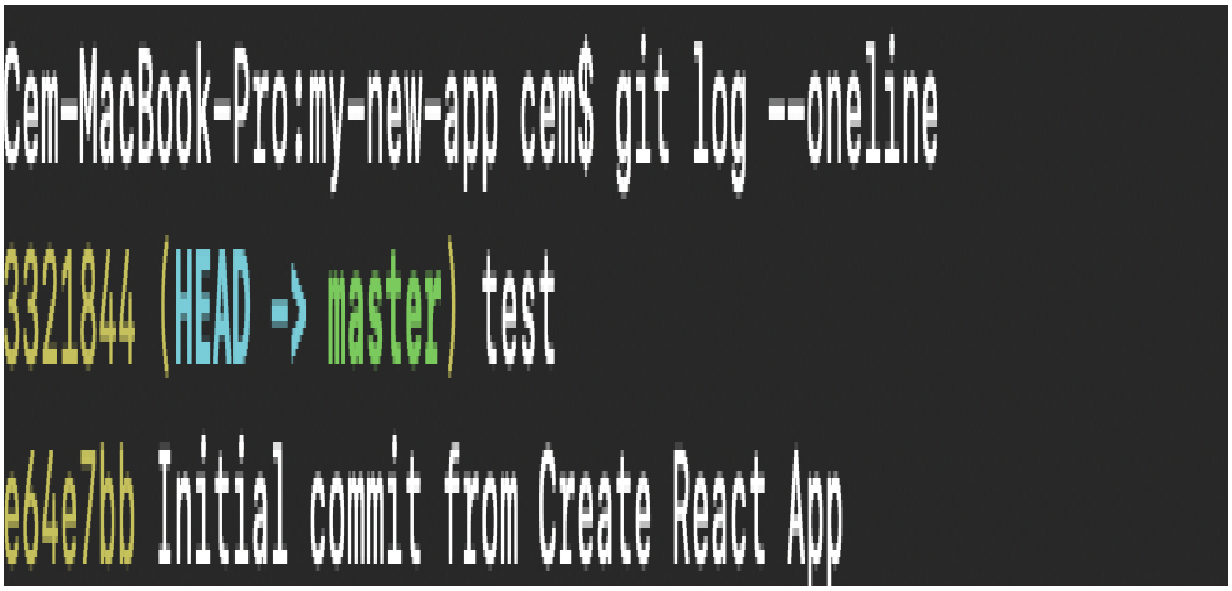
```
git pull <remote>
```

This operation may cause conflicts that you need to solve manually.

9. Git revert

Sometimes we need to undo the changes that we've made. There are various ways to undo our changes locally or remotely (depends on what we need), but we must carefully use these commands to avoid unwanted deletions.

A safer way that we can undo our commits is by using **git revert**. To see our commit history, first we need to use **git log --oneline:**



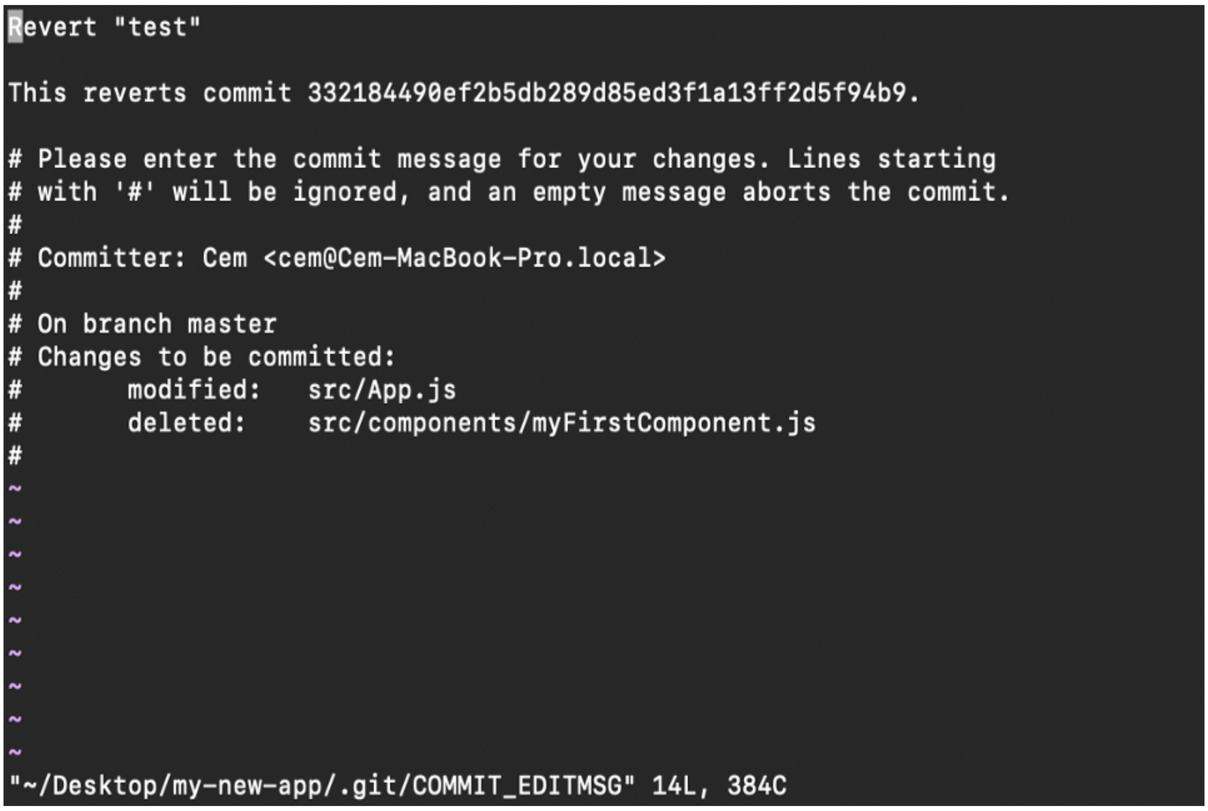
```
Cem-MacBook-Pro:my-new-app cem$ git log --oneline
3321844 (HEAD -> master) test
e64e7bb Initial commit from Create React App
```

commit history of my master branch

Then we just need to specify the hash code next to our commit that we would like to undo:

```
git revert 3321844
```

After this, you will see a screen like below - just press **shift + q** to exit:

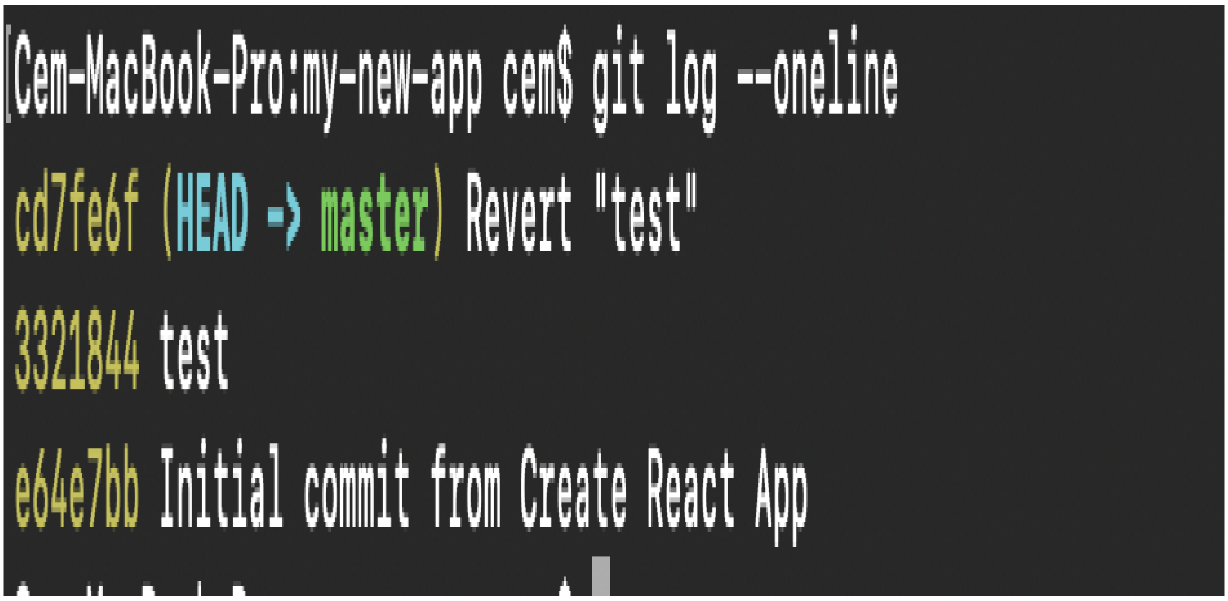


```
Revert "test"

This reverts commit 332184490ef2b5db289d85ed3f1a13ff2d5f94b9.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Committer: Cem <cem@Cem-MacBook-Pro.local>
#
# On branch master
# Changes to be committed:
#	modified:   src/App.js
#	deleted:    src/components/myFirstComponent.js
#
~
~
~
~
~
~
~
~
~
"~/Desktop/my-new-app/.git/COMMIT_EDITMSG" 14L, 384C
```

The Git revert command will undo the given commit, but will create a new commit without deleting the older one:



```
Cem-MacBook-Pro:my-new-app cem$ git log --oneline
cd7fe6f (HEAD -> master) Revert "test"
3321844 test
e64e7bb Initial commit from Create React App
```

new "revert" commit

The advantage of using **git revert** is that it doesn't touch the commit history. This means that you can still see all of the commits in your history, even the reverted ones.

Another safety measure here is that everything happens in our local system unless we push them to the remote repo. That's why git revert is safer to use and is the preferred way to undo our commits.

10. Git merge

When you've completed development in your branch and everything works fine, the final step is merging the branch with the parent branch (dev or master). This is done with the `git merge` command.

Git merge basically integrates your feature branch with all of its commits back to the dev (or master) branch. It's important to remember that you first need to be on the specific branch that you want to merge with your feature branch.

For example, when you want to merge your feature branch into the dev branch:

First you should switch to the dev branch:

```
git checkout dev
```

Before merging, you should update your local dev branch:

```
git fetch
```

Finally, you can merge your feature branch into dev:

```
git merge <branch-name>
```