

We've released a brand new React course!

[Learn more →](#)



Git Basics

Foundations Course

Introduction

In this lesson, we'll cover common Git commands used to manage your projects and to upload your work onto GitHub. We refer to these commands as the **basic Git workflow**. When you're using Git, these are the commands that you'll use 70-80% of the time. So if you can get these down, you'll be more than halfway done mastering Git!

Lesson overview

This section contains a general overview of topics that you will learn in this lesson.

- How to create a repository on GitHub.
- How to get files to and from GitHub.
- How to take "snapshots" of your code.

Assignment

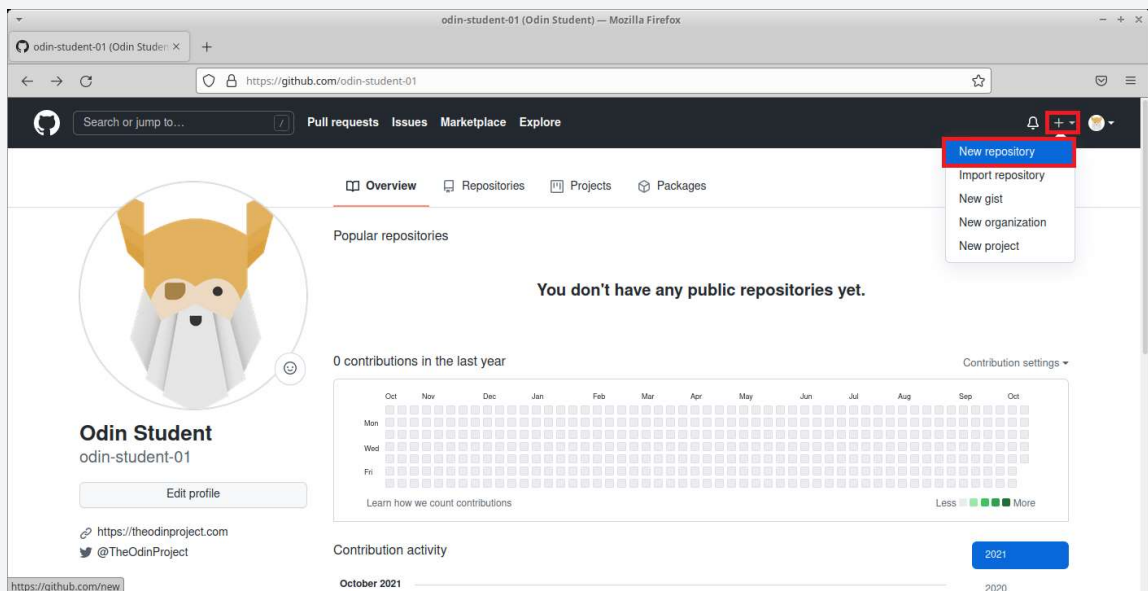
Before you start!

- Github recently updated the way it names the default branch. This means you need to make sure you are using a recent version of git (2.28 or later). You can check your version by running: `git --version`

- If you haven't already, set your local default git branch to `main`. You can do so by running: `git config --global init.defaultBranch main`
- For more information on the change from `master` to `main` see [GitHub's Renaming Repository](#).

Create the repository

1. You should have already created a GitHub account in the [Setting Up Git](#) lesson. If you haven't done that yet, you can sign up [here](#).
2. Create a new repository by clicking the button shown in the screenshot below.




3. Give your repository the name "git_test" in the repository name input field. Check "Add a README file". And then create the repository by clicking the "Create repository" button at the bottom of the page.

Create a new repository

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

Owner *

 odin-student-01 ▾

Repository name *

git_test ✓

Great repository names are short and memorable. Need inspiration? How about [curly-happiness?](#)

Description (optional)

My first GitHub repo!

☒  **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.](#)

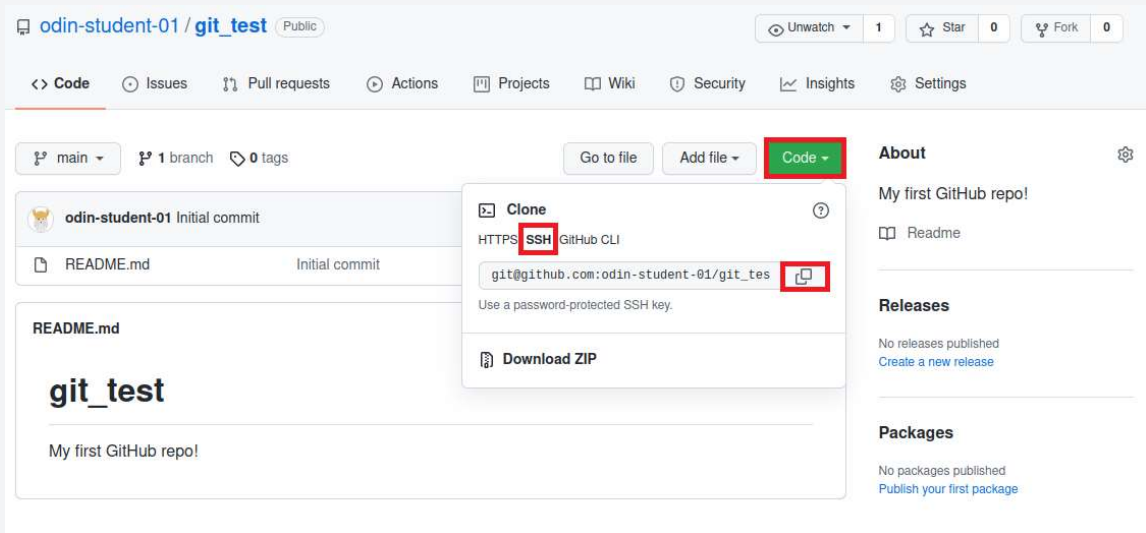
☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

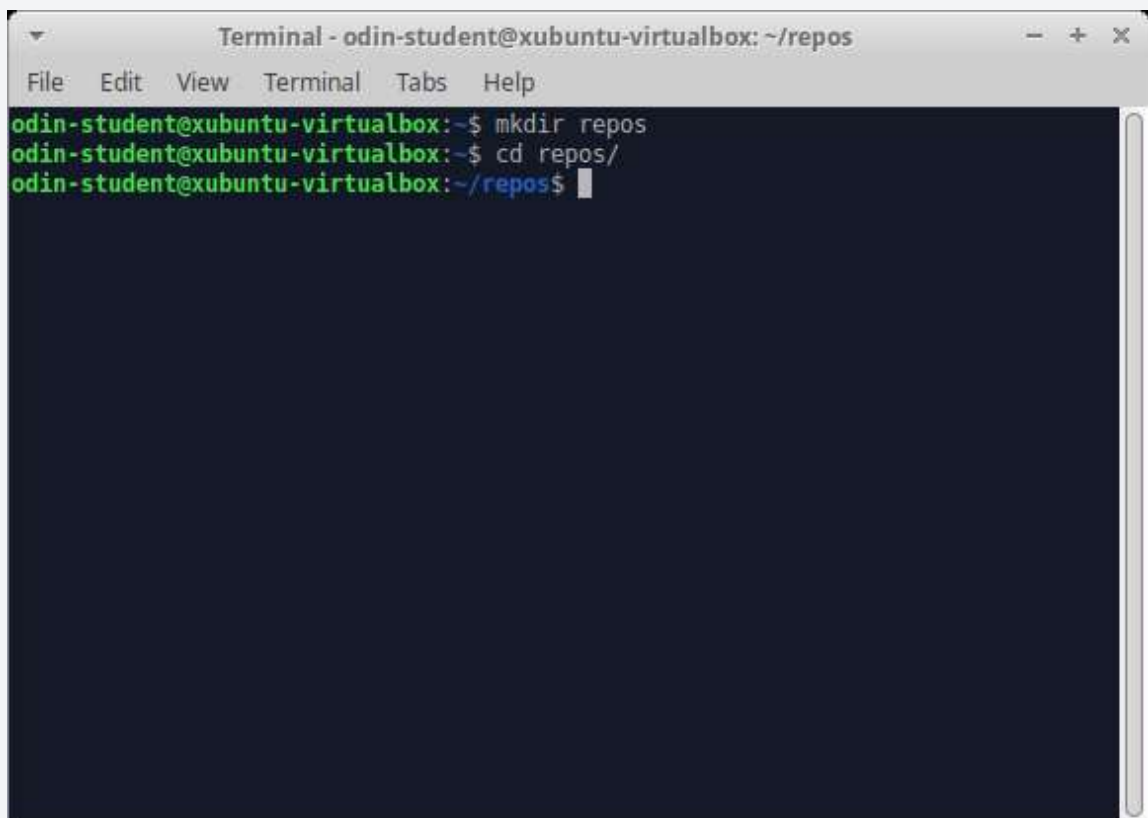
This will set  `main` as the default branch. Change the default name in your [settings](#).

Create repository

4. This will redirect you to your new repository on GitHub. To get ready to copy (clone) this repository onto your local machine, click the green “Code” button. Then select the SSH option, and copy the line below it. **NOTE: You MUST click the SSH option to get the correct URL.**

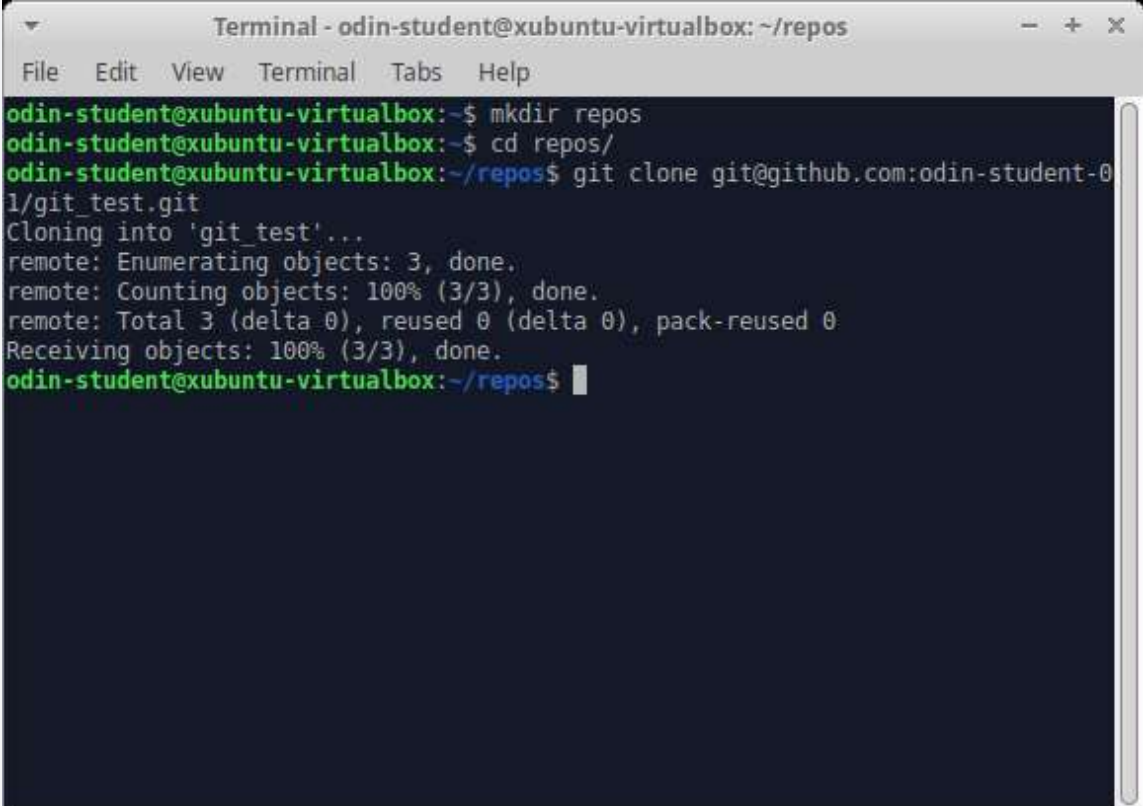


5. Let's use the command line on your local machine to create a new directory for all of your Odin projects. Create a directory called `repos` with the `mkdir` command in your home folder. Your home folder is represented by `~`. [Navigating Files and Directories](#) covered variations of home folders - sometimes `~` stands for `/Users/your_username` and sometimes it stands for `/home/your_username`. If you're not sure if you're in your home folder, just type `cd ~`. Once it's made, move into it with the `cd` command.



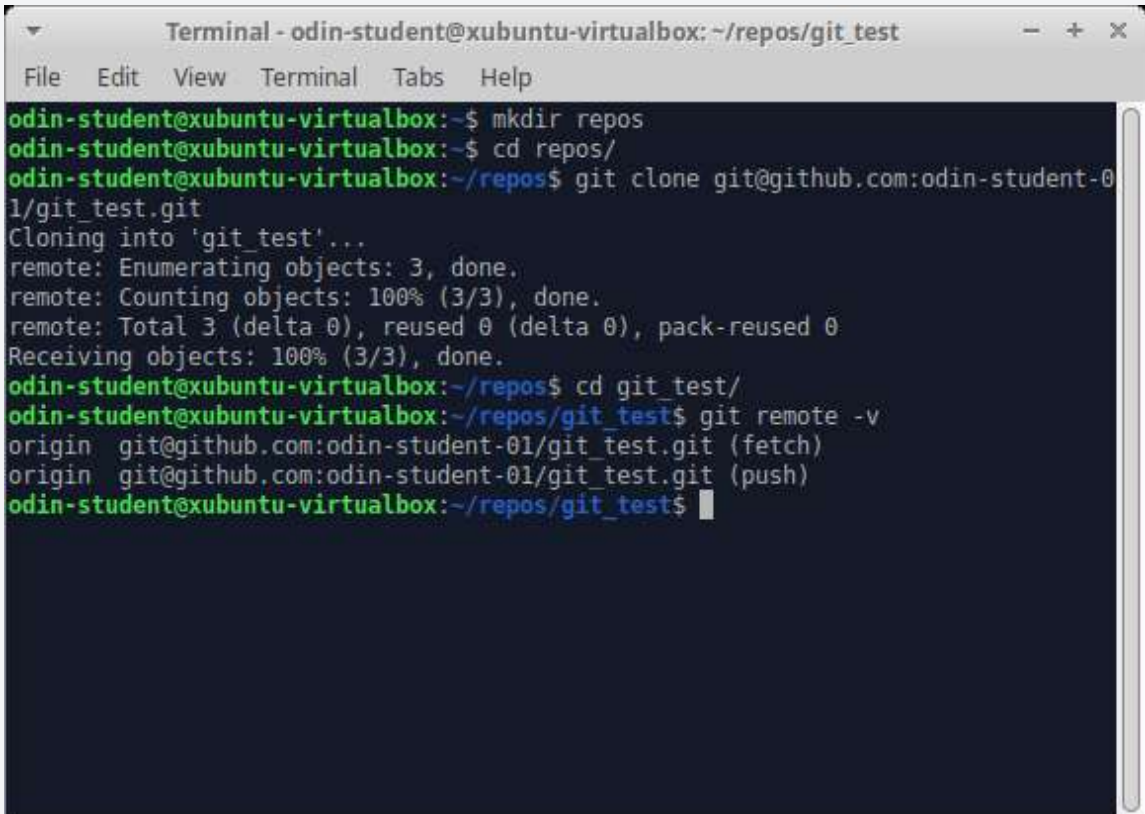
6. Now it's time to clone your repository from GitHub onto your computer with `git clone` followed by the URL you copied in the last step. The full

command should look similar to `git clone git@github.com:USER-NAME/REPOSITORY-NAME.git`. If your URL looks like `https://github.com/USER-NAME/REPOSITORY-NAME.git`, you have selected the HTTPS option, not the required SSH option.

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
odin-student@xubuntu-virtualbox:~$ mkdir repos
odin-student@xubuntu-virtualbox:~$ cd repos/
odin-student@xubuntu-virtualbox:~/repos$ git clone git@github.com:odin-student-01/git_test.git
Cloning into 'git_test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
odin-student@xubuntu-virtualbox:~/repos$
```

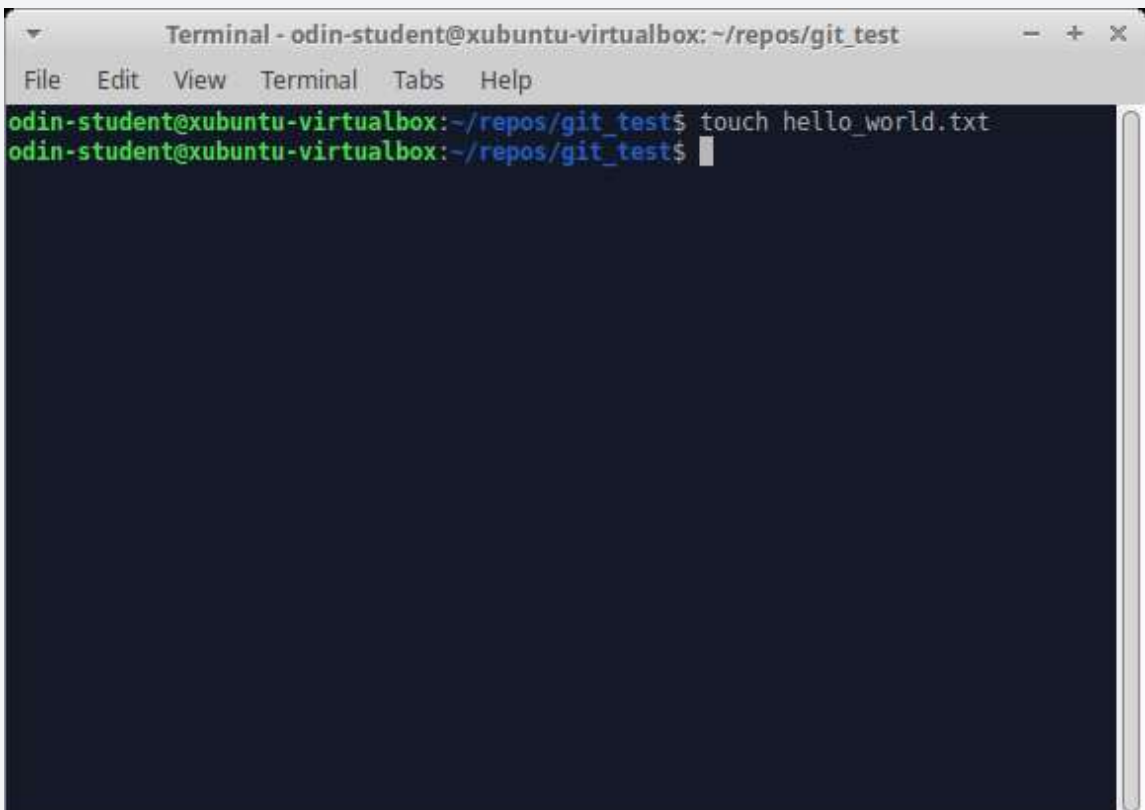
7. That's it! You have successfully connected the repository you created on GitHub to your local machine. To test this, you can `cd` into the new **git_test** folder that was downloaded and then enter `git remote -v` on your command line. This will display the URL of the repository you created on GitHub, which is the remote for your local copy. You may have also noticed the word **origin** at the start of the `git remote -v` output, which is the name of your remote connection. The name "origin" is both the default and the convention for the remote repository. But it could have just as easily been named "party-parrot" or "dancing-banana". (Don't worry about the details of origin for now; it will come up again near the end of this tutorial.)

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test". The window contains the following commands and output:

```
odin-student@xubuntu-virtualbox:~$ mkdir repos
odin-student@xubuntu-virtualbox:~$ cd repos/
odin-student@xubuntu-virtualbox:~/repos$ git clone git@github.com:odin-student-01/git_test.git
Cloning into 'git_test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
odin-student@xubuntu-virtualbox:~/repos$ cd git_test/
odin-student@xubuntu-virtualbox:~/repos/git_test$ git remote -v
origin  git@github.com:odin-student-01/git_test.git (fetch)
origin  git@github.com:odin-student-01/git_test.git (push)
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

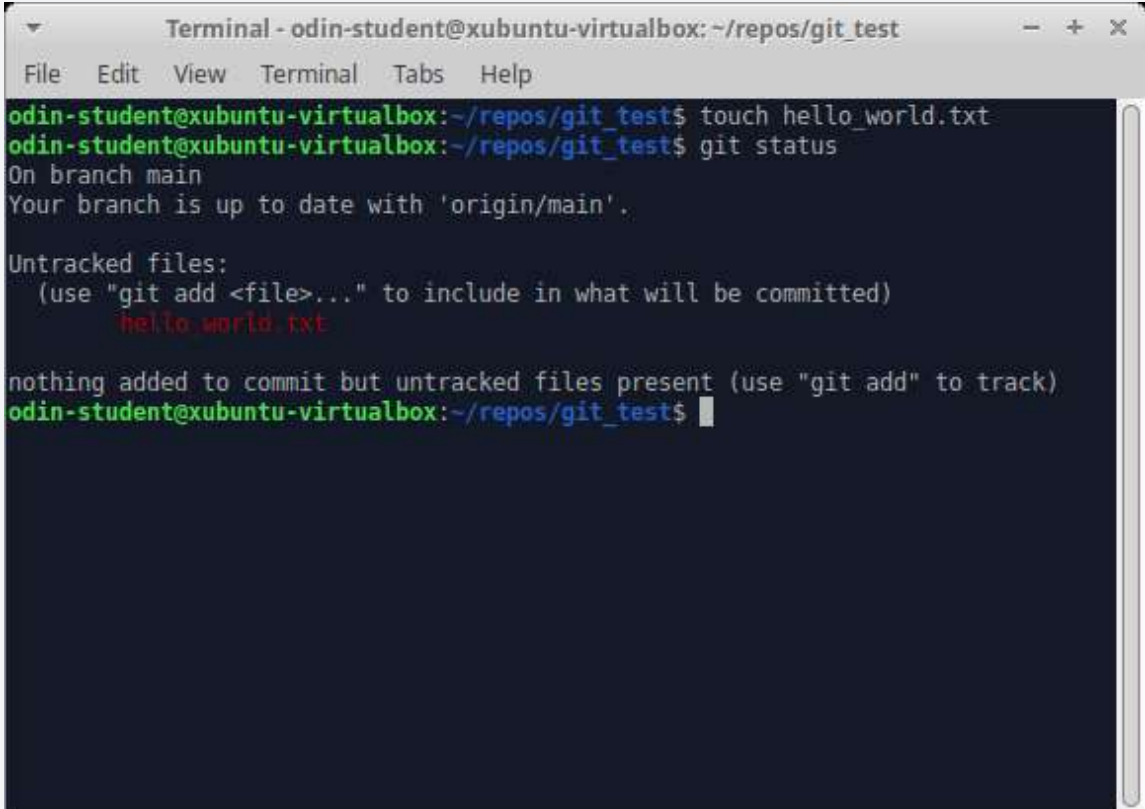
Use the Git workflow

1. Create a new file in the `git_test` folder called "hello_world.txt" with the command `touch hello_world.txt`.

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test". The window contains the following commands and output:

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ touch hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

2. Type `git status` in your terminal. In the output, notice that your `hello_world.txt` file is shown in red, which means that this file is not staged.

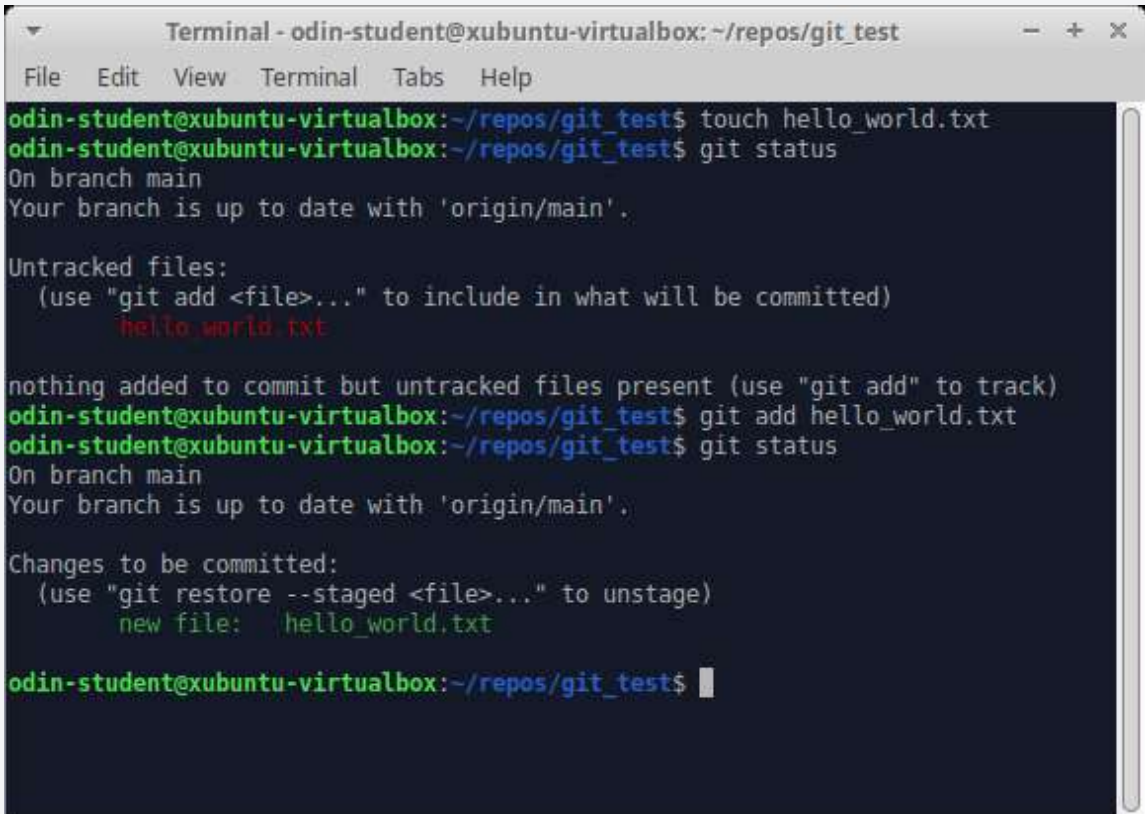
A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ touch hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

3. Type `git add hello_world.txt`. This command adds your `hello_world.txt` file to the staging area in Git. The staging area is part of the two-step process for making a commit in Git. Think of the staging area as a “waiting room” for your changes until you commit them. Now, type `git status` again. In the output, notice that your file is now shown in green, which means that this file is now in the staging area.

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test". The window shows the following commands and output:

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ touch hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

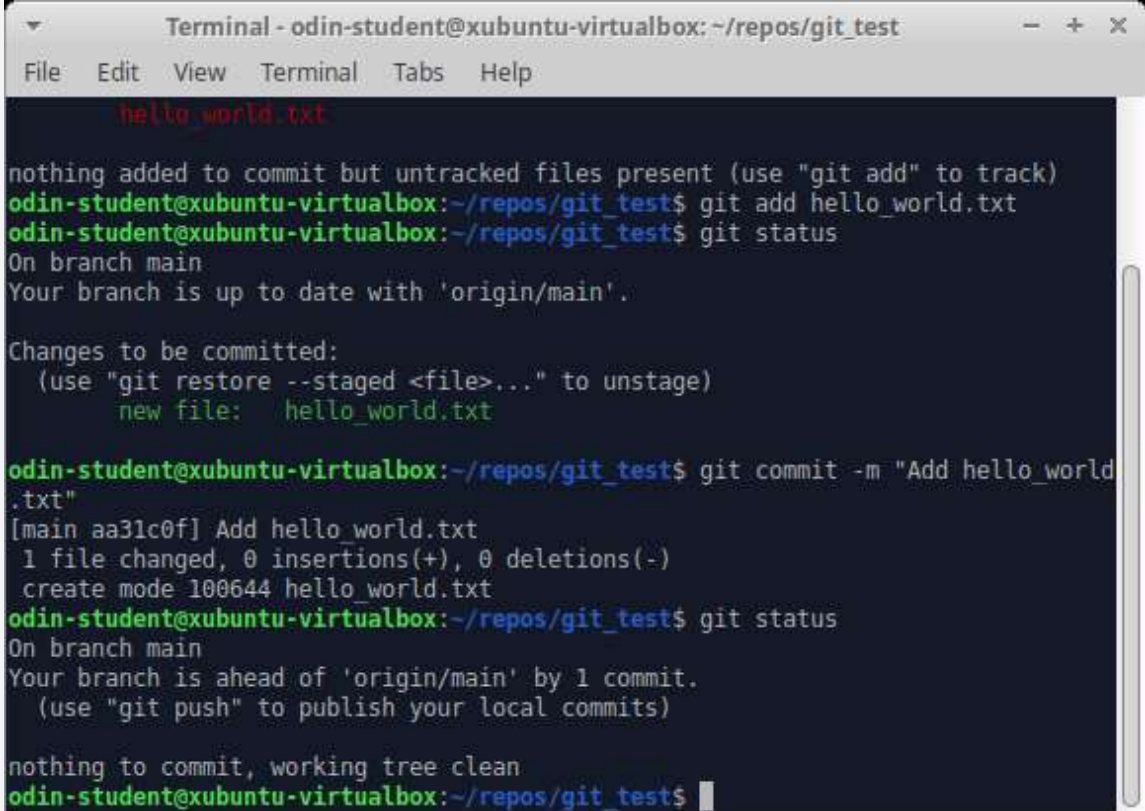
nothing added to commit but untracked files present (use "git add" to track)
odin-student@xubuntu-virtualbox:~/repos/git_test$ git add hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       new file:   hello_world.txt

odin-student@xubuntu-virtualbox:~/repos/git_test$
```

4. Type `git commit -m "Add hello_world.txt"` and then type `git status` once more. The output should now say: *"nothing to commit, working tree clean"*, indicating your changes have been committed. Don't worry if you get a message that says *"upstream is gone"*. This is normal and only shows when your cloned repository currently has no branches. It will be resolved once you have followed the rest of the steps in this project.

The message, *"Your branch is ahead of 'origin/main' by 1 commit"* just means that you now have newer snapshots than what is on your remote repository. You will be uploading your snapshots further down in this lesson.



```
Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test
File Edit View Terminal Tabs Help

hello_world.txt

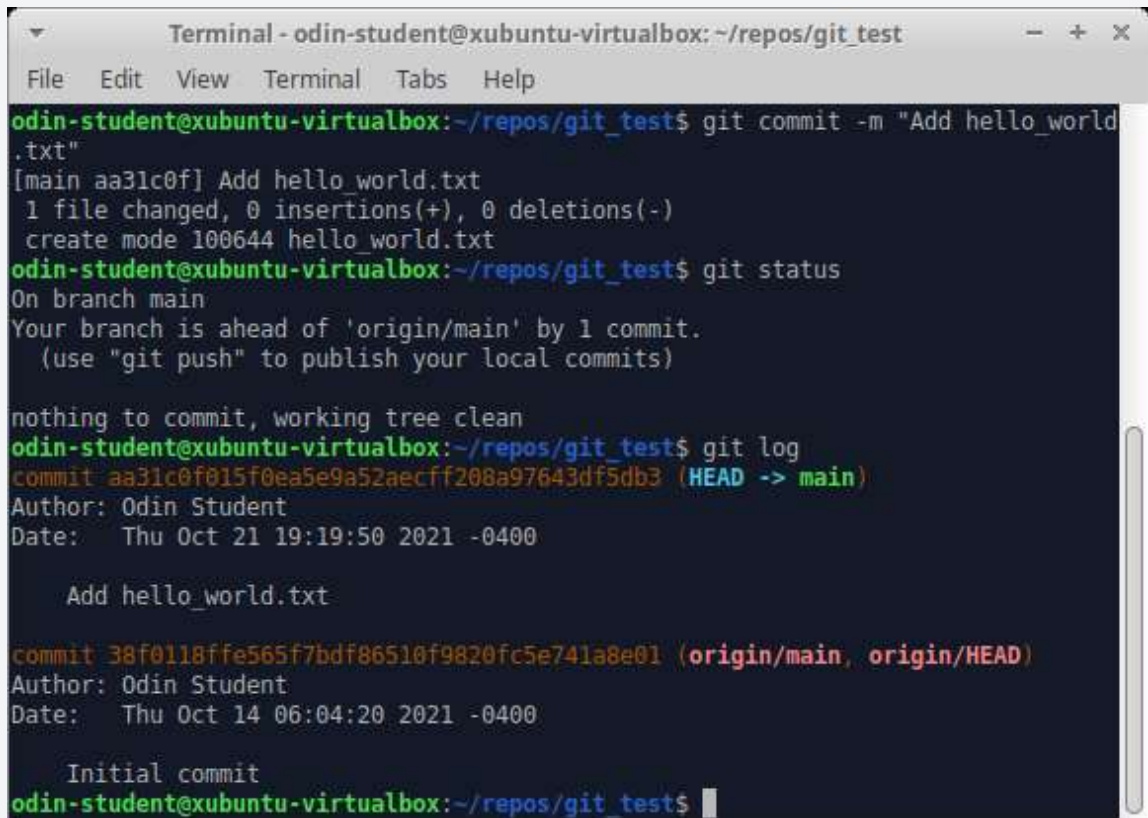
nothing added to commit but untracked files present (use "git add" to track)
odin-student@xubuntu-virtualbox:~/repos/git_test$ git add hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   hello_world.txt

odin-student@xubuntu-virtualbox:~/repos/git_test$ git commit -m "Add hello_world
.txt"
[main aa31c0f] Add hello_world.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

5. Type `git log` and look at the output. You should see an entry for your *"Add hello_world.txt"* commit. You will also see details on the author who made the commit and the date and time of when the commit was made. If your terminal is stuck in a screen with (END) at the bottom, just press "q" to escape. You can configure settings for this later, but don't worry about it too much for now.



```
Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test
File Edit View Terminal Tabs Help

odin-student@xubuntu-virtualbox:~/repos/git_test$ git commit -m "Add hello_world.txt"
[main aa31c0f] Add hello_world.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello_world.txt
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
odin-student@xubuntu-virtualbox:~/repos/git_test$ git log
commit aa31c0f015f0ea5e9a52aecff208a97643df5db3 (HEAD -> main)
Author: Odin Student
Date: Thu Oct 21 19:19:50 2021 -0400

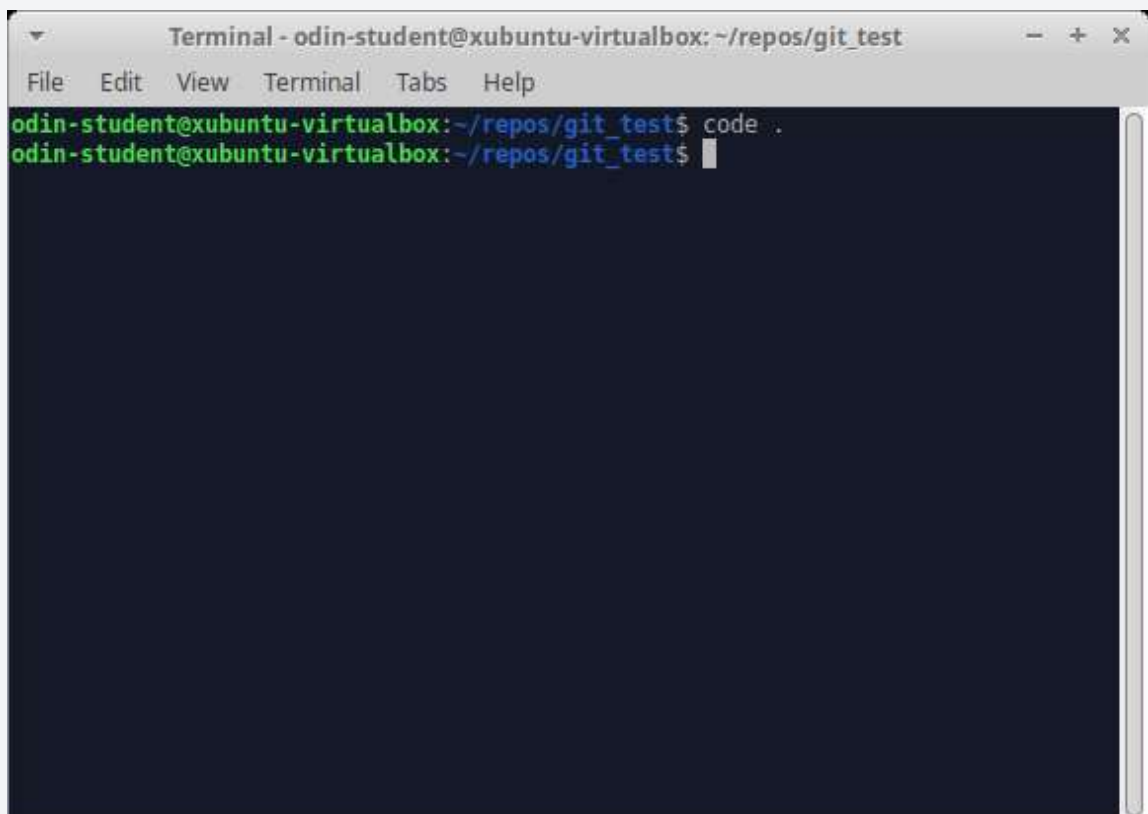
    Add hello_world.txt

commit 38f0118ffe565f7bdf86510f9820fc5e741a8e01 (origin/main, origin/HEAD)
Author: Odin Student
Date: Thu Oct 14 06:04:20 2021 -0400

    Initial commit
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

Modify a file or two

1. Open README.md in your text editor of choice. In this example, we will open the directory in Visual Studio Code by using the command `code .` inside your repository.

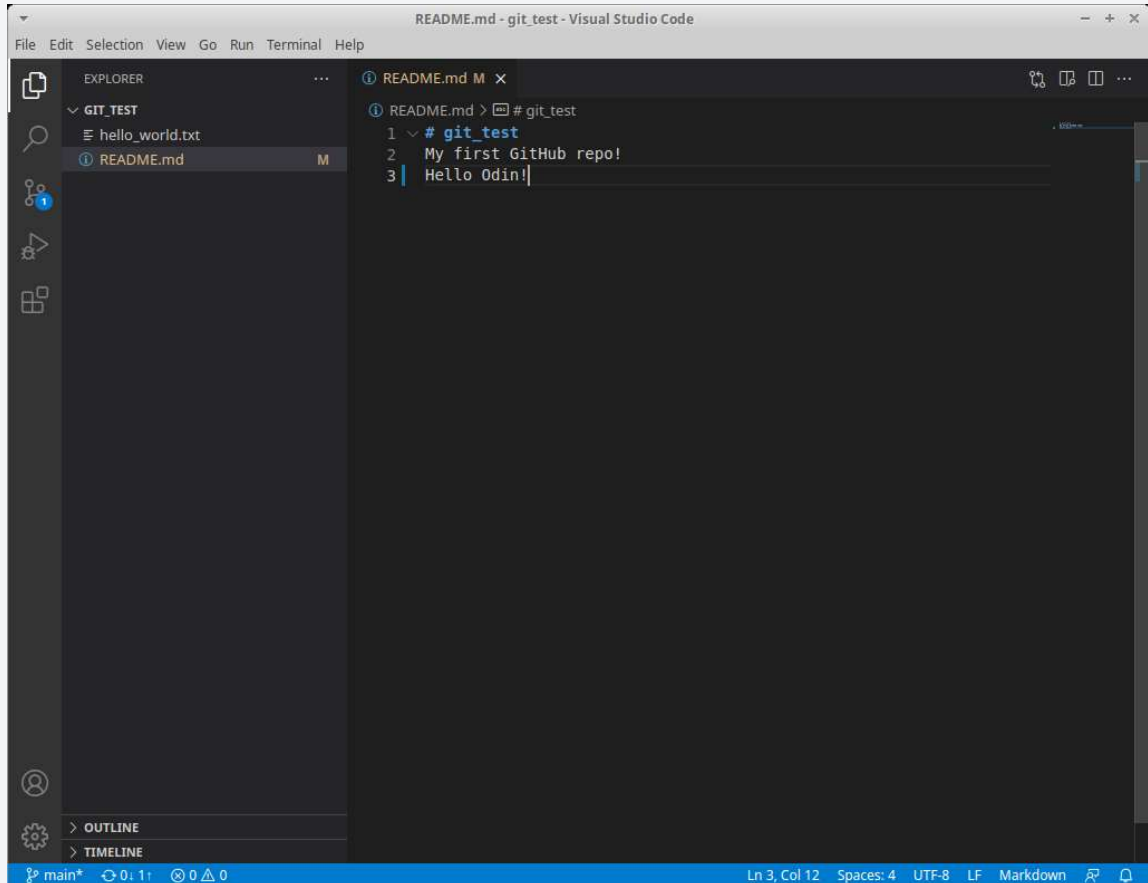


```
Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test
File Edit View Terminal Tabs Help

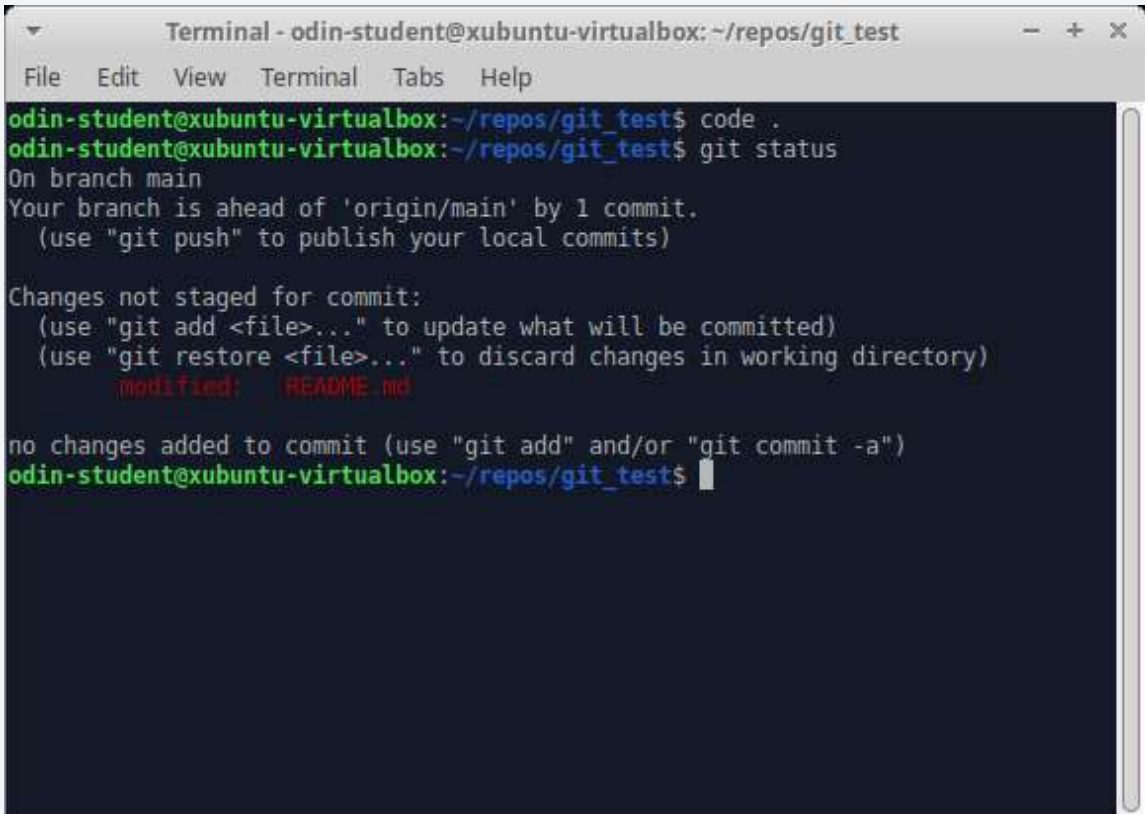
odin-student@xubuntu-virtualbox:~/repos/git_test$ code .
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

MacOS users: If your terminal reads *“command not found: code”*, you must head back to [Command Line Basics](#) and follow the instructions provided to allow this command to work.

2. Add “Hello Odin!” to line 3 of README.md and save the file with `ctrl` + `s` (Mac: `Cmd` + `s`).



1. Go back to your terminal or if you’re using Visual Studio Code you can open the built-in terminal by pressing `ctrl` + ``` (backtick). Then type `git status`. You’ll notice that README.md is now shown as not staged or committed.

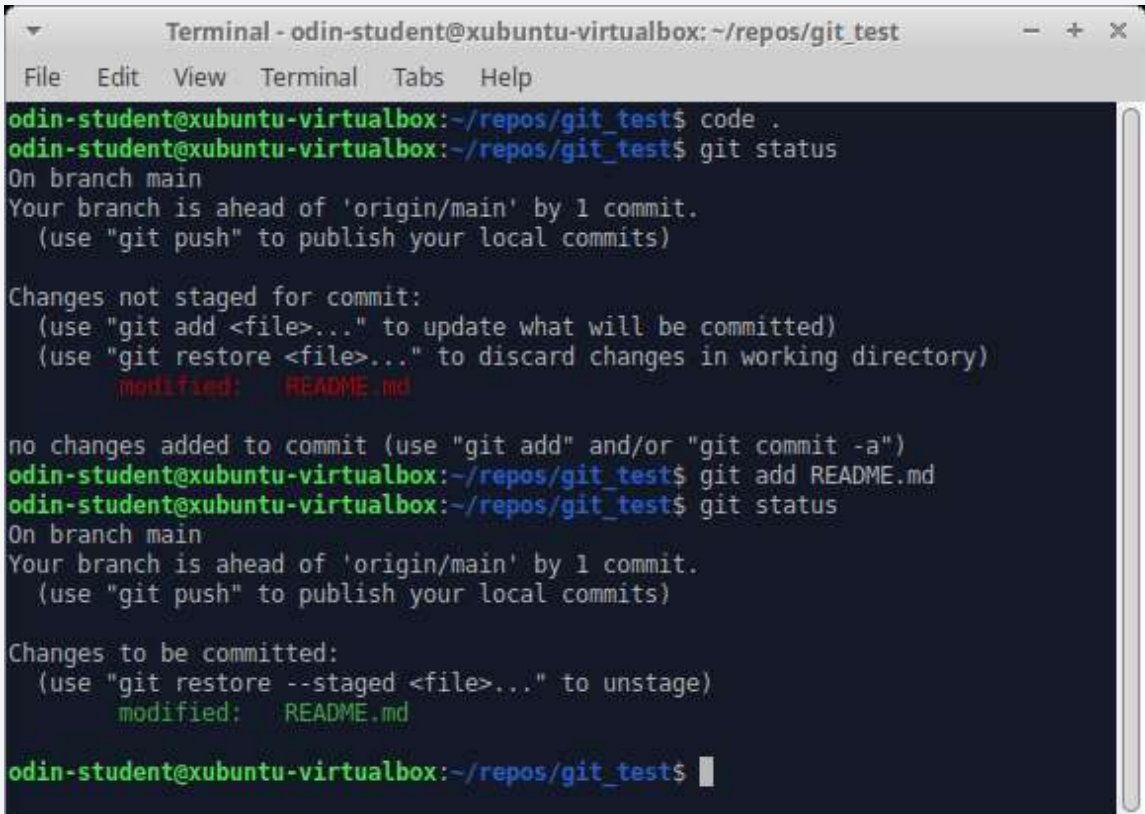
A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ code .
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

2. Add README.md to the staging area with `git add README.md`.
3. Can you guess what `git status` will output now? README.md will be displayed in green text. That means README.md has been added to the staging area. The file hello_world.txt will not show up because it has not been modified since it was committed.

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test". The window shows the following commands and output:

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ code .
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

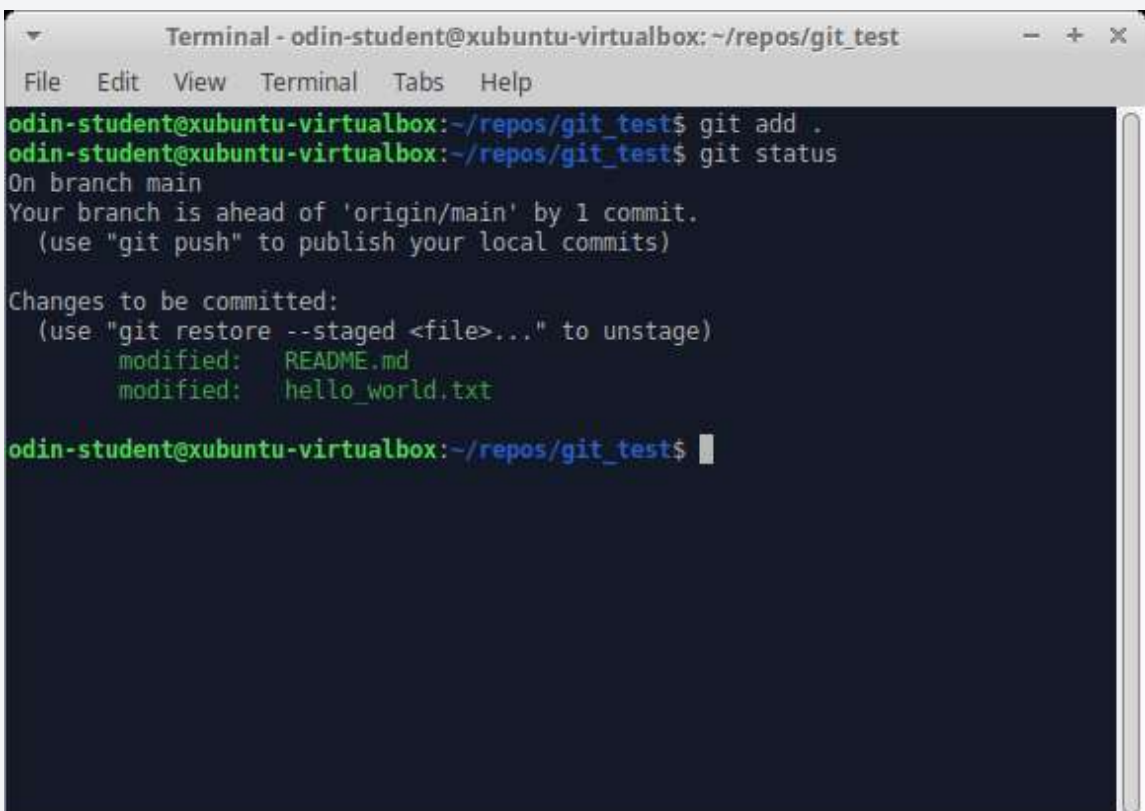
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
odin-student@xubuntu-virtualbox:~/repos/git_test$ git add README.md
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

odin-student@xubuntu-virtualbox:~/repos/git_test$
```

4. Open `hello_world.txt`, add some text to it, save it and stage it. You can use `git add .` to add all files in the current directory and all subsequent directories to the staging area. Then, type `git status` once more, and everything should now be in the staging area.

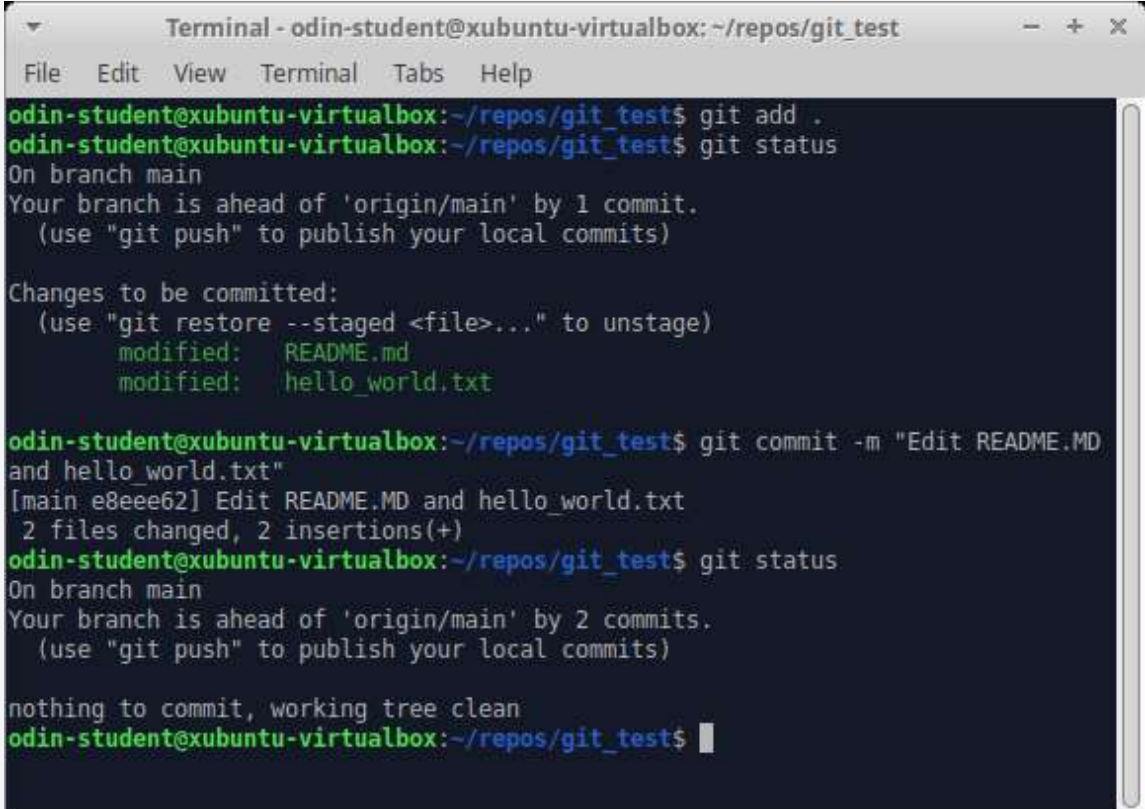
A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test". The window shows the following commands and output:

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ git add .
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        modified:   hello_world.txt

odin-student@xubuntu-virtualbox:~/repos/git_test$
```


5. Finally, let's commit all of the files that are in the staging area and add a descriptive commit message. `git commit -m "Edit README.md and hello_world.txt"`. Then, type `git status` once again, which will output *"nothing to commit"*.

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following sequence of commands and output:

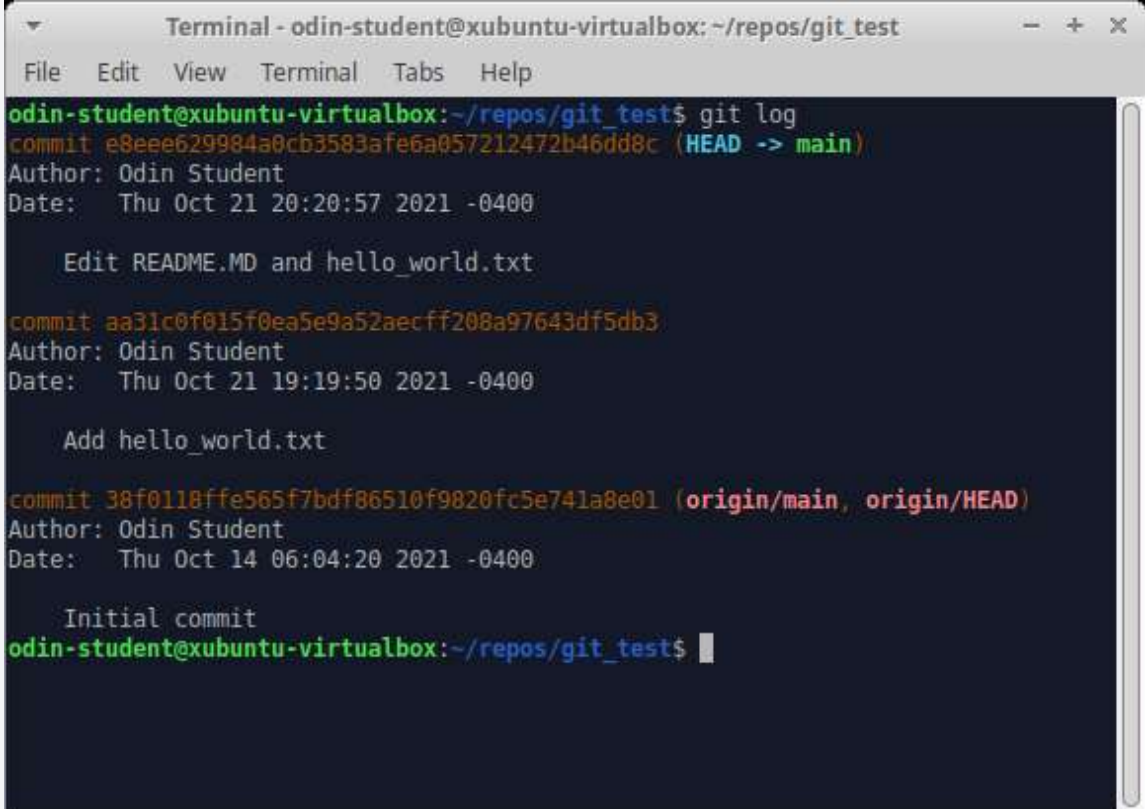
```
odin-student@xubuntu-virtualbox:~/repos/git_test$ git add .
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        modified:   hello_world.txt

odin-student@xubuntu-virtualbox:~/repos/git_test$ git commit -m "Edit README.MD
and hello_world.txt"
[main e8eee62] Edit README.MD and hello_world.txt
 2 files changed, 2 insertions(+)
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

6. Take one last look at your commit history by typing `git log`. You should now see three entries.

A terminal window titled "Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the output of the 'git log' command. It lists three commits: 1. 'Initial commit' with hash 38f0118ffe565f7bdf86510f9820fc5e741a8e01, dated Thu Oct 14 06:04:20 2021. 2. 'Add hello_world.txt' with hash aa31c0f015f0ea5e9a52aecff208a97643df5db3, dated Thu Oct 21 19:19:50 2021. 3. 'Edit README.MD and hello_world.txt' with hash e8eee629984a0cb3583afe6a057212472b46dd8c, dated Thu Oct 21 20:20:57 2021. The HEAD points to the main branch.

```
odin-student@xubuntu-virtualbox:~/repos/git_test$ git log
commit e8eee629984a0cb3583afe6a057212472b46dd8c (HEAD -> main)
Author: Odin Student
Date: Thu Oct 21 20:20:57 2021 -0400

    Edit README.MD and hello_world.txt

commit aa31c0f015f0ea5e9a52aecff208a97643df5db3
Author: Odin Student
Date: Thu Oct 21 19:19:50 2021 -0400

    Add hello_world.txt

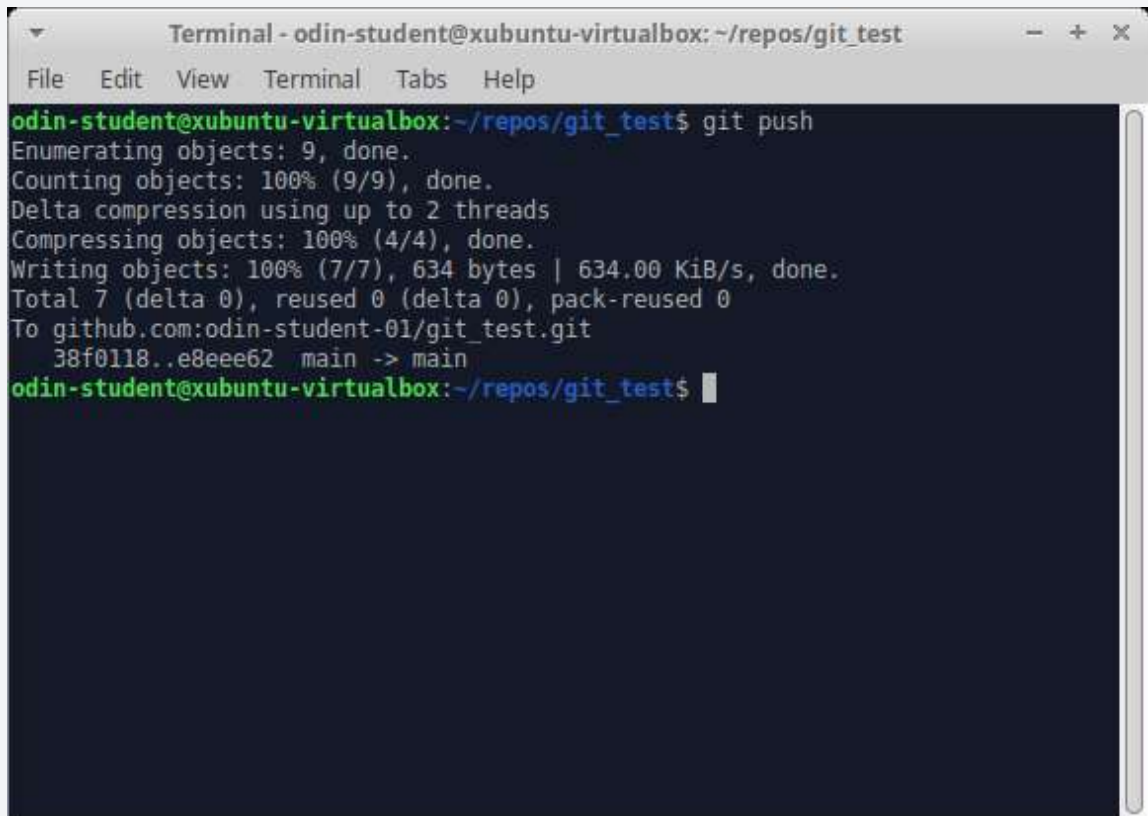
commit 38f0118ffe565f7bdf86510f9820fc5e741a8e01 (origin/main, origin/HEAD)
Author: Odin Student
Date: Thu Oct 14 06:04:20 2021 -0400

    Initial commit
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

Push your work to GitHub

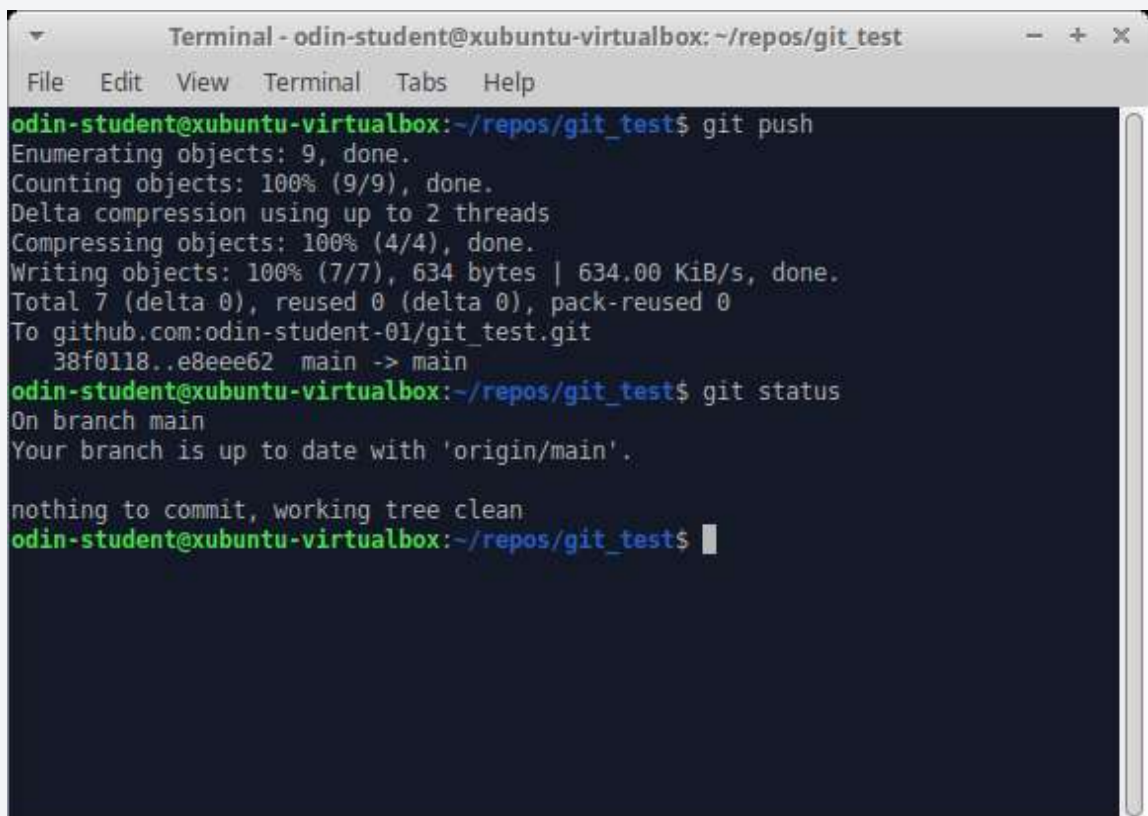
Finally, let's upload your work to the GitHub repository you created at the start of this tutorial.

1. Type `git push`. To be more specific, type `git push origin main`. Since you are not dealing with another branch (other than *main*) or a different remote (as mentioned above), you can leave it as `git push` to save a few keystrokes. **NOTE: If at this point you receive a message that says "Support for password authentication was removed on August 13, 2021. Please use a personal access token instead."**, you have followed the steps incorrectly and cloned with HTTPS, not SSH. Please follow [these steps](#) to change your remote to SSH, then attempt to push to Github.



```
Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test
File Edit View Terminal Tabs Help
odin-student@xubuntu-virtualbox:~/repos/git_test$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 634 bytes | 634.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:odin-student-01/git_test.git
   38f0118..e8eee62  main -> main
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

2. Type `git status` one final time. It should output *"Your branch is up to date with 'origin/main'. nothing to commit, working tree clean"*.

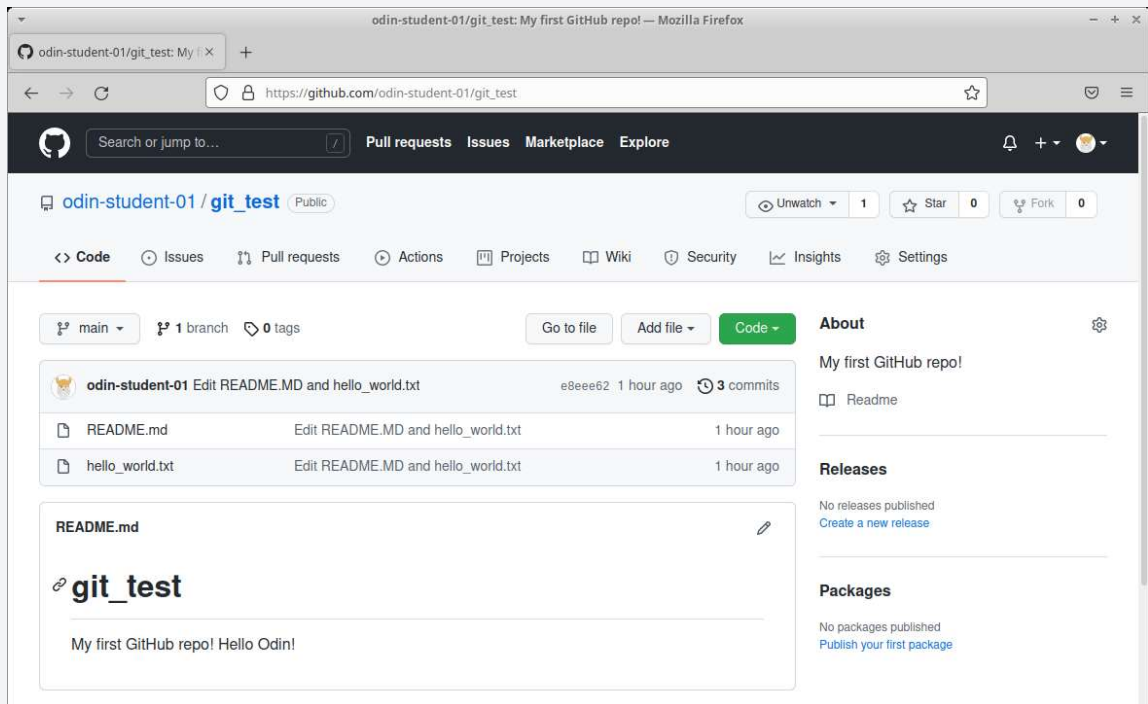


```
Terminal - odin-student@xubuntu-virtualbox: ~/repos/git_test
File Edit View Terminal Tabs Help
odin-student@xubuntu-virtualbox:~/repos/git_test$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 634 bytes | 634.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:odin-student-01/git_test.git
   38f0118..e8eee62  main -> main
odin-student@xubuntu-virtualbox:~/repos/git_test$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
odin-student@xubuntu-virtualbox:~/repos/git_test$
```

3. When you reload the repository on GitHub, you should see the README.md and hello_world.txt files that you just pushed there from your local

machine.



Note/Warning

When trying to make simple changes to the files in your repo, such as attempting to fix a typo in your README.md you might be tempted to make this change directly via Github. However, it is best to avoid this as it will cause issues that require more advanced Git knowledge than we want to go over at this stage (it is covered in a future lesson), for now it is advised to make any changes via your local files then commit and push them using Git commands in your terminal once ready.

Cheatsheet

This is a reference list of the most commonly used Git commands. (You might consider bookmarking this handy page.) Try to familiarize yourself with the commands so that you can eventually remember them all:

- Commands related to a remote repository:
 - `git clone git@github.com:USER-NAME/REPOSITORY-NAME.git`
 - `git push` or `git push origin main` (Both accomplish the same goal in this context)

- Commands related to the workflow:
 - `git add .`
 - `git commit -m "A message describing what you have done to make this snapshot different"`
- Commands related to checking status or log history
 - `git status`
 - `git log`

The basic Git syntax is `program | action | destination`.

For example,

- `git add .` is read as `git | add | .`, where the period represents everything in the current directory;
- `git commit -m "message"` is read as `git | commit -m | "message"`; and
- `git status` is read as `git | status | (no destination)`.

Git best practices

There's a lot to learn about using Git. But it is worth taking the time to highlight some best practices so that you can be a better collaborator. Git is not only helpful when collaborating with others. It's also useful when working independently. You will be relying more and more on your own commit history in the future when revisiting old code.

Two helpful best practices to consider are **atomic commits** and leveraging those atomic commits to make your commit messages more useful to future collaborators.

An atomic commit is a commit that includes changes related to only one feature or task of your program. There are two main reasons for doing this: first, if something you change turns out to cause some problems, it is easy to revert the specific change without losing other changes; and second, it enables you to write better commit messages. You'll learn more about what a good commit message looks like in a future lesson!

Changing the Git commit message editor

If you are using *Visual Studio Code* (and you should be if you're following this curriculum), there's a way to ensure that if you use `git commit` without the message flag (`-m`), you won't get stuck writing your commit message in [Vim](#).

Changing the default message editor is a good idea in case you accidentally omit the flag, unless you prefer using Vim. There is no downside to changing it, because you will have the option to write your commit messages in the terminal or in the comfort of VS Code.

The following command will set this configuration. Type (or copy & paste) this command into your terminal and hit `Enter`.

```
1 | git config --global core.editor "code --wait"
```

There will be no confirmation or any output on the terminal after entering this command.

With that done, you can now choose to use either `git commit -m <your message here>` or `git commit` to type your message with Visual Studio Code!

To make a commit with Visual Studio Code as the text editor, just type `git commit`. After you hit `Enter` a new tab in VS Code will open for you to write your commit message. You may provide more details on multiple lines as part of your commit message. After typing your commit message, save it `ctrl + s` (Mac: `cmd + s`) and close the tab. If you return to the command line, you will see your commit message and a summary of your changes.

Conclusion

You may not feel completely comfortable with Git at this point, which is normal. It's a skill that you will get more comfortable with as you use it.

The main thing to take away from this lesson is the **basic workflow**. The commands you've learned here are the ones you will be using the most often with Git.

Don't worry if you don't know all the commands yet or if they aren't quite sticking in your memory yet. They will soon be seared into your brain as you use them over and over in future Odin projects.

In later Git lessons, we will cover some of the more advanced Git features, such as branches. They will further expand your abilities and make you more productive.

For now, concentrate on using the basics of Git that you've learned here for all of your projects from now on. You will soon know each of the basic Git commands from memory!

Knowledge check

This section contains questions for you to check your understanding of this lesson on your own. If you're having trouble answering a question, click it and review the material it links to.

- [How do you create a new repository on GitHub?](#)
- [How do you copy a repository onto your local machine from GitHub?](#)
- [What is the default name of your remote connection?](#)
- [Explain what `origin` is in `git push origin main`.](#)
- [Explain what `main` is in `git push origin main`.](#)
- [Explain the two-stage system that Git uses to save files.](#)
- [How do you check the status of your current repository?](#)
- [How do you add files to the staging area in git?](#)
- [How do you commit the files in the staging area and add a descriptive message?](#)
- [How do you push your changes to your repository on GitHub?](#)
- [How do you look at the history of your previous commits?](#)


Additional resources

This section contains helpful links to related content. It isn't required, so consider it supplemental.

- [Complete Git and GitHub Tutorial from Basics to Advanced](#) - by Kunal Kushwaha
- [Git - Reference](#)
- [This article about adding locally hosted code to GitHub](#) will walk you through creating a git repository from a local folder and adding it to GitHub.

 View Course

Mark Complete

 Next Lesson [Improve this lesson on GitHub](#)**THE ODIN PROJECT**

High quality coding education maintained by an open source community.

**About us**[About](#)[Blog](#)[Success Stories](#)**Support**[FAQ](#)**Guides**[Community guides](#)[Installation guides](#)**Legal**[Terms](#)[Privacy](#)

[Contribute](#)

[Contact us](#)

© 2023 The Odin Project. All rights reserved.

