الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Configuration management

Lecture 02: Initialize your first repository

Eng. Raghad al-hossny

Install and create the first repository:

1. **Installing git from the main website:**

```
git --version
```

Check the installation

Ubuntu / Linux

```
- sudo apt update
- sudo apt install git
```

Windows

https://git-scm.com/downloads/win

Install and create the first repository:

2. add global configuration:

```
git config --global user.name "Raghad"
git config --global user.email raghad@email.com

git config --global --list
```

global configuration, stored once for your whole computer.
Used:
- To identify who made each change.
- To show your name correctly on GitHub , when pushing the code.
- By setting it *globally*, you don't have to retype your name and email for every project.

Install and create the first repository:

3. Let’s create a new folder to be our working directory:

Step 01: Make a new directory *mkdir class1*

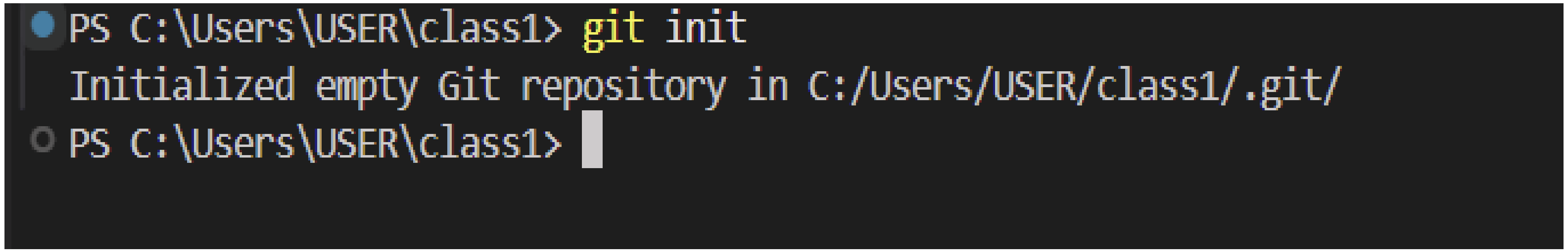
Step 02: Enter this directory *cd class1*

Step 03: Open the directory in your VS code (code editor) *code .*

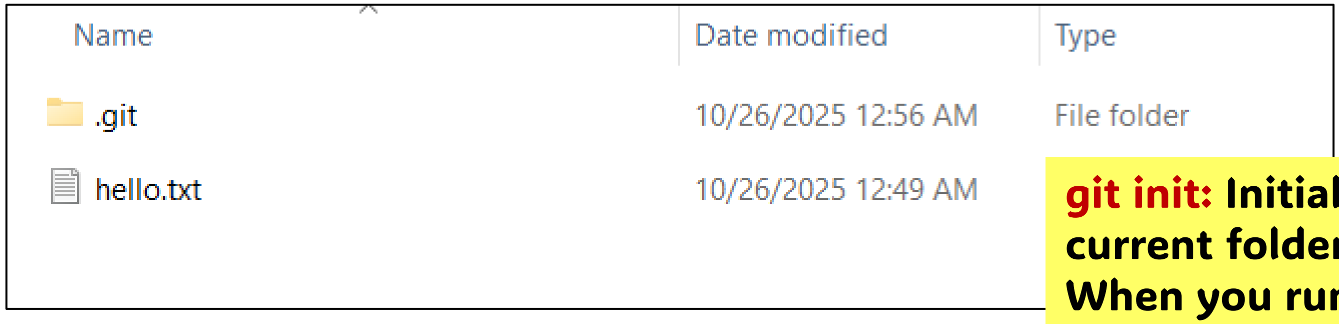
Step 04: Create a new text file hello.txt

Install and create the first repository:

4. Initialize a git repository in this folder:



```
PS C:\Users\USER\class1> git init
Initialized empty Git repository in C:/Users/USER/class1/.git/
PS C:\Users\USER\class1>
```



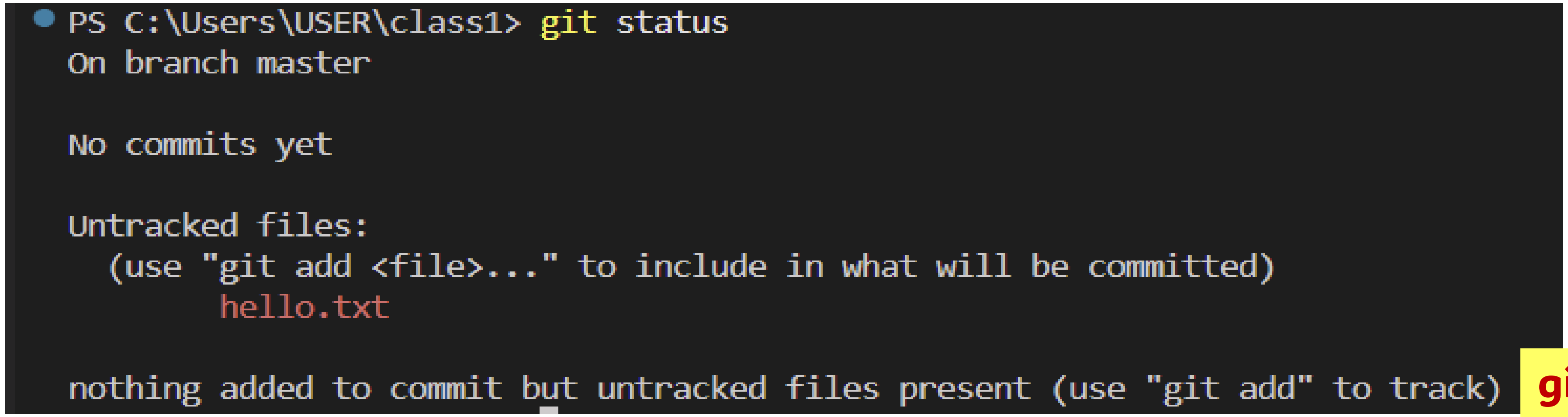
Name	Date modified	Type
.git	10/26/2025 12:56 AM	File folder
hello.txt	10/26/2025 12:49 AM	

git init: Initializes a new Git repository in your current folder.
When you run it inside any project folder, Git creates a hidden directory called **.git** that stores all version control information (commits, branches, logs, etc.).

git init my_first_repo
cd my_first_repo
This creates a new folder and starts Git tracking in it.

Install and create the first repository:

5. Check the status of the repo files:



```
PS C:\Users\USER\class1> git status
On branch master

No commits yet

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

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

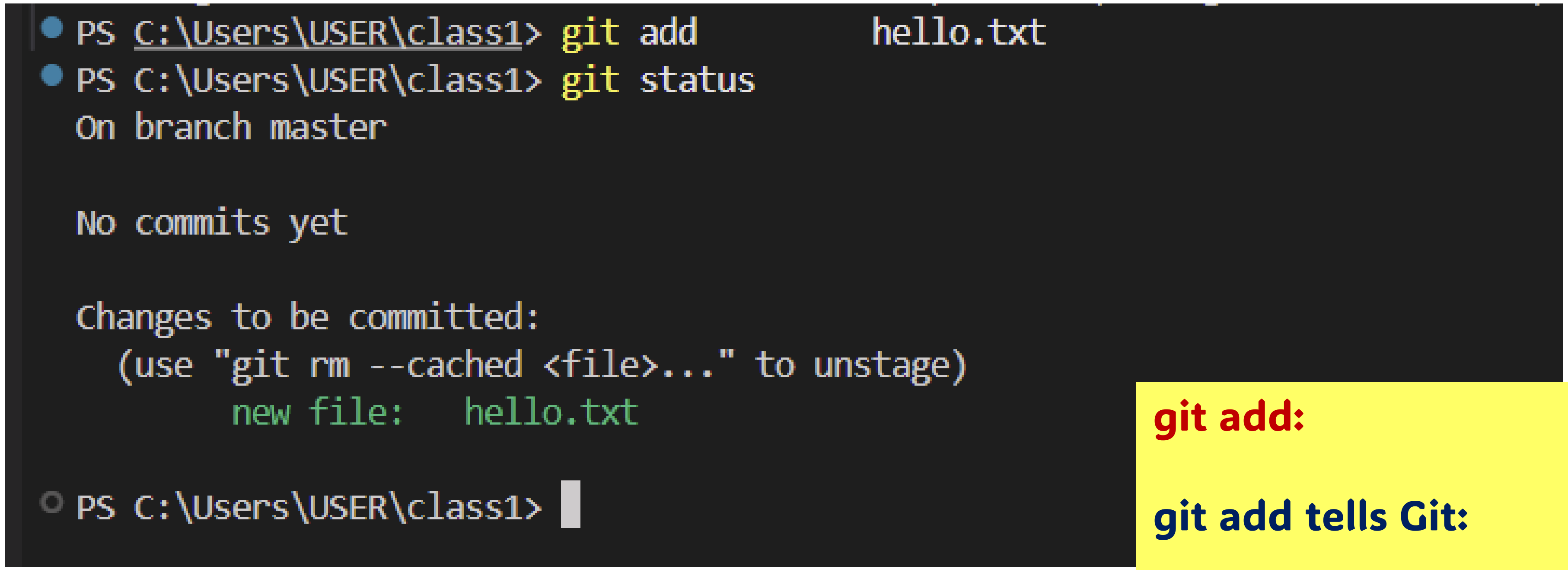
git status
tells you the current state of your working directory and staging area

Untracked files:

This means you created a file called hello.txt, but Git isn't tracking it yet — it's new to Git.
An untracked file is a file that exists in your project folder, but Git is not yet watching it (not tracking its changes).
Git knows the file *exists*, but it doesn't include it in version control until you explicitly tell it to.

Install and create the first repository:

6. Make the new file tracked by git:



```
PS C:\Users\USER\class1> git add        hello.txt
PS C:\Users\USER\class1> git status
On branch master

No commits yet

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

PS C:\Users\USER\class1>
```

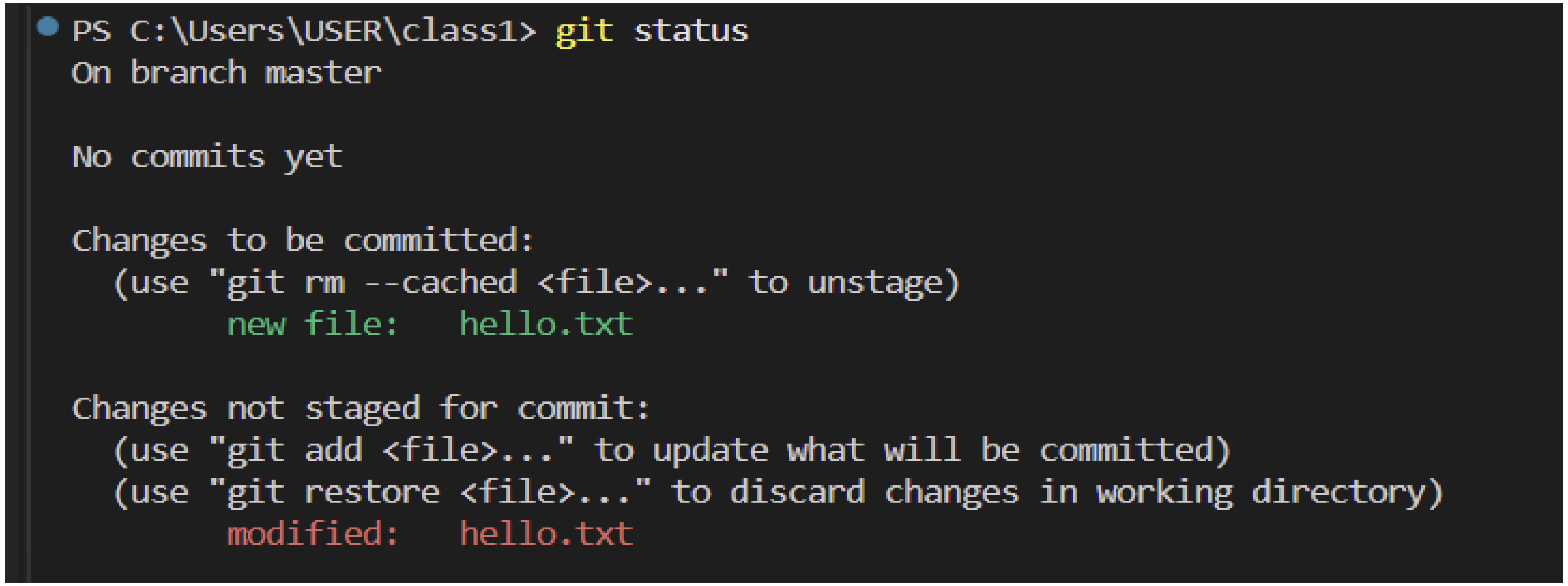
git add:

git add tells Git:

"Start tracking this file and prepare it for the next commit."

Install and create the first repository:

7. Try to make changes to this new tracked file:



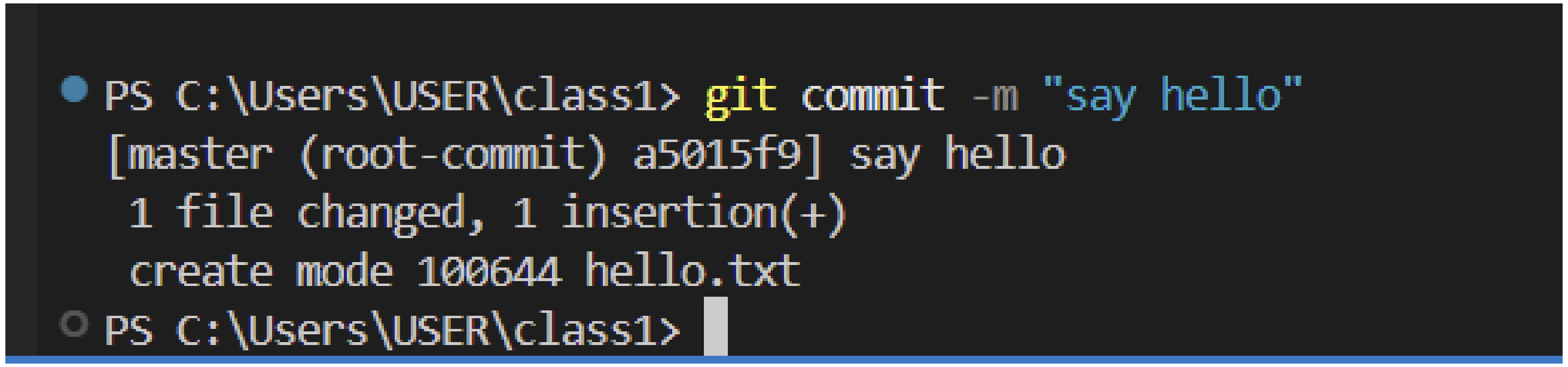
```
PS C:\Users\USER\class1> git status
On branch master

No commits yet

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

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:   hello.txt
```

After staging, we edited the file again:
Those new edits are not yet staged.

So, Git sees two versions:
- **The old one (staged, ready to commit)**
- **The new one (in working directory, not added yet)**

Install and create the first repository:

8. Commit files and changes:



```
● PS C:\Users\USER\class1> git commit -m "say hello"
  [master (root-commit) a5015f9] say hello
   1 file changed, 1 insertion(+)
   create mode 100644 hello.txt
○ PS C:\Users\USER\class1> 
```

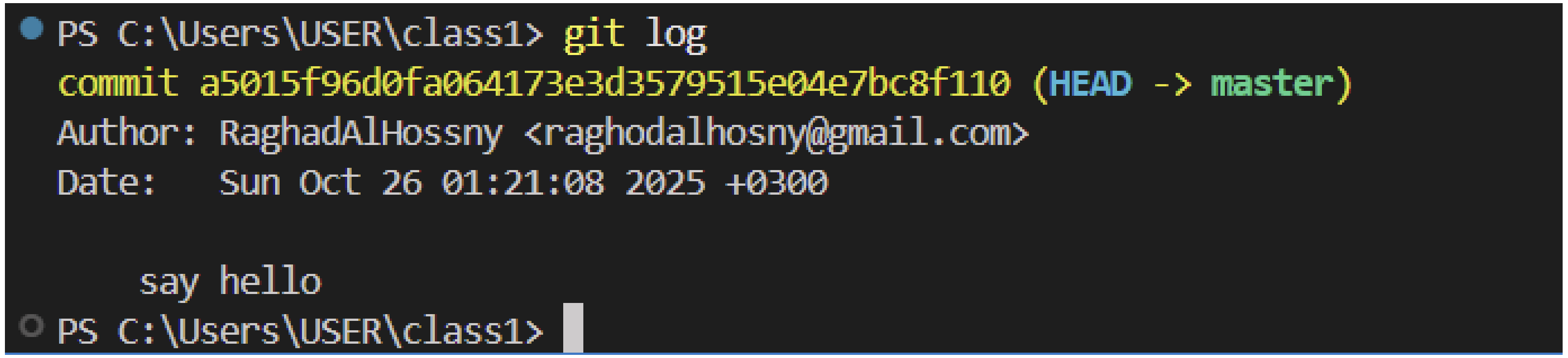
git commit:
tells Git to take a snapshot of your staged changes and save them permanently in the local repository history.

After committing, your file becomes tracked and stored safely in the .git folder (your local repo database).

- **master (root-commit) - This is the first ever commit in this repository.**
- **a5015f9 - The unique ID (hash) of your commit.**
- **1 file changed, 1 insertion(+) - You added one new line to a file.**
- **create mode 100644 hello.txt - Git created a new tracked file.**

Install and create the first repository:

9. See the history of commits:



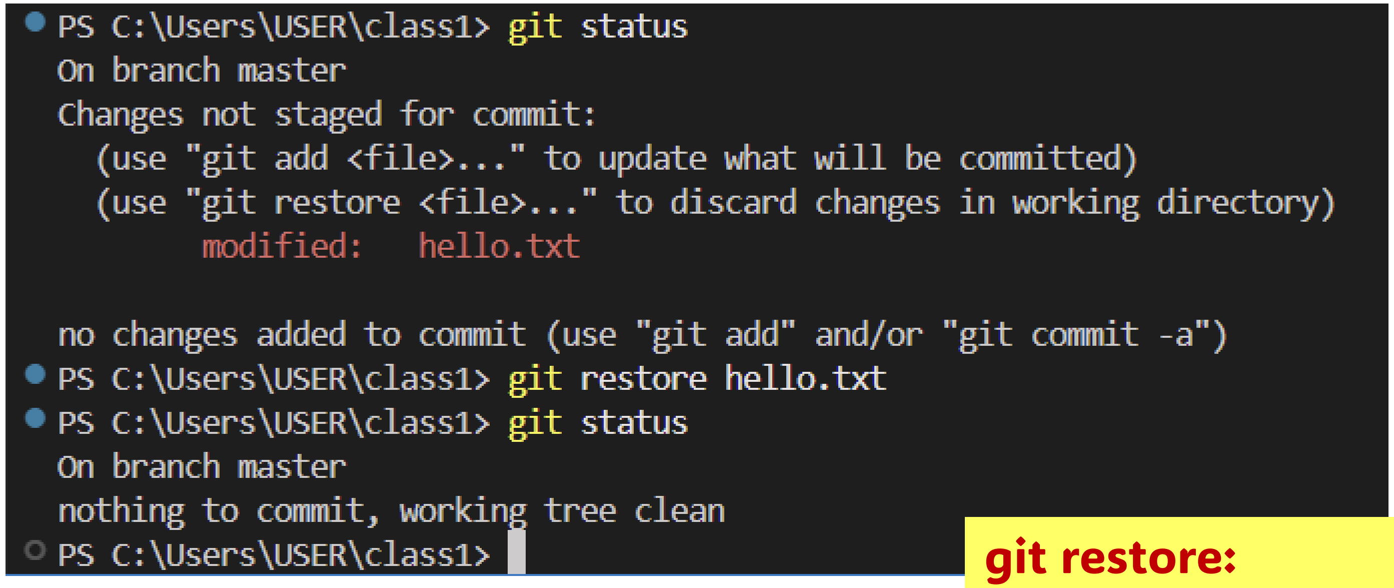
```
PS C:\Users\USER\class1> git log
commit a5015f96d0fa064173e3d3579515e04e7bc8f110 (HEAD -> master)
Author: RaghadAlHossny <raghodalhosny@gmail.com>
Date:   Sun Oct 26 01:21:08 2025 +0300

    say hello
PS C:\Users\USER\class1> 
```

git log:

The command shows the history of commits in your repository —
basically a list of every saved snapshot you've made with git commit.

Install and create the first repository:

10. What if we want to rollback changes?



```
PS C:\Users\USER\class1> git status
On branch master
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:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\USER\class1> git restore hello.txt
PS C:\Users\USER\class1> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\USER\class1>
```

git restore:

is used to **undo changes** in your working directory or staging area.

Only work for tracked files…

Install and create the first repository:

Exercise: Git Basic Workflow

1. Initialize a new Git repository called my_first_repo.
2. Create a new text file named group.txt.
3. Check the status of your repository.
4. Track this new file using git
5. Add your group members’ names to the text file and save it.
6. Check the status again to see the modification.
7. Add your group members’ phone numbers to the same file.
8. Roll back the last change (remove the phone numbers).
9. Commit the clean version with a message like "Add group names".
10. Check the history of commits.

Create GitHub account:

Step 1: Go to Github in your web browser

Step 2: Click the Sign-up button in the top right of the screen.

Step 3: Enter your email address

Step 4: Choose a strong password

Step 5: Enter a username

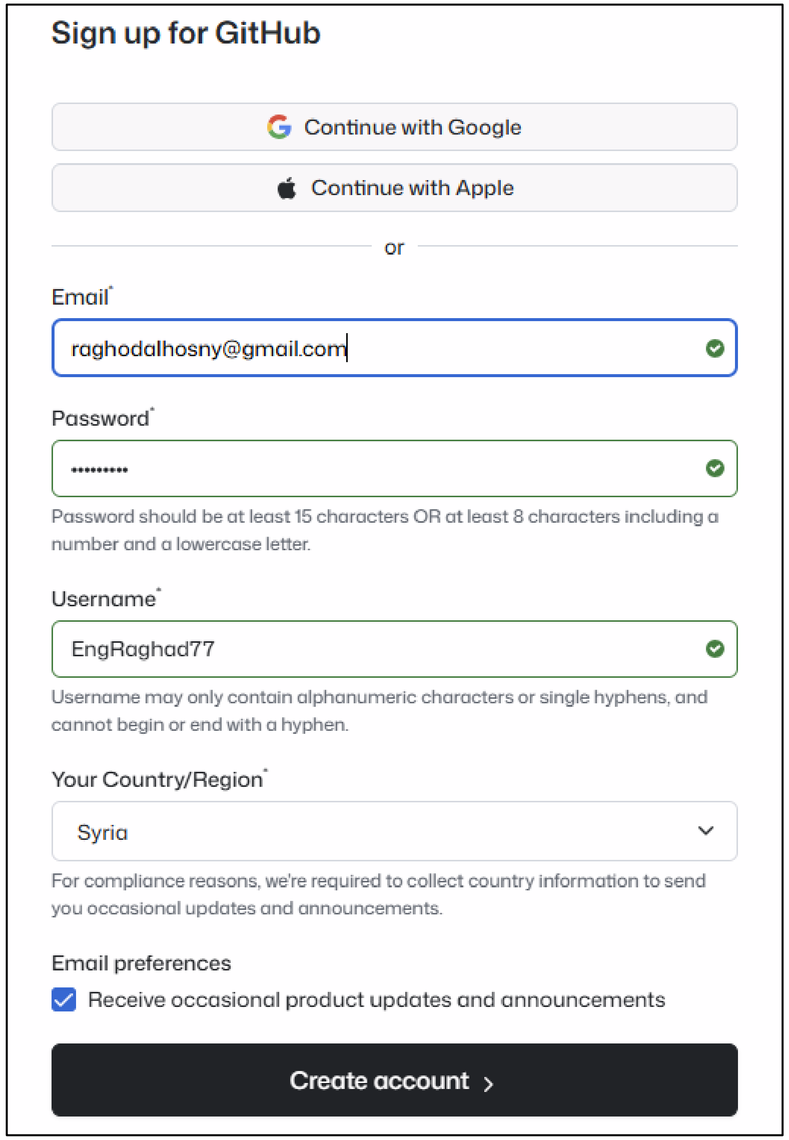
Step 6: Complete the security question

Step 7: Click Create Account

Step 8: You will then receive a confirmation email to confirm your address. The email will contain a code. Enter this code on the confirmation screen.

Create a GitHub account – sign up:

1



Sign up for GitHub

Email*

raghodalhosny@gmail.com

Password*

•••••••••

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*

EngRaghad77

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your Country/Region*

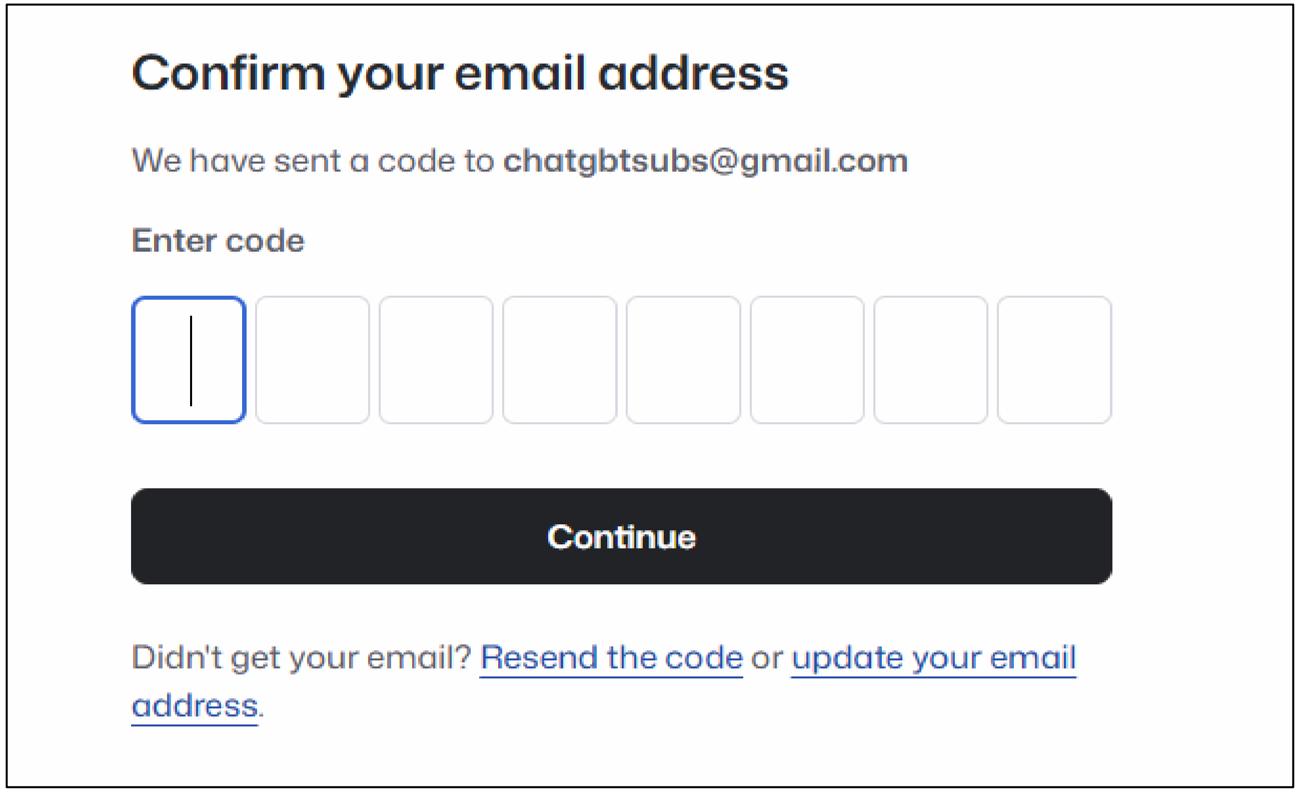
Syria

For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Email preferences

[✓] Receive occasional product updates and announcements

2



Confirm your email address

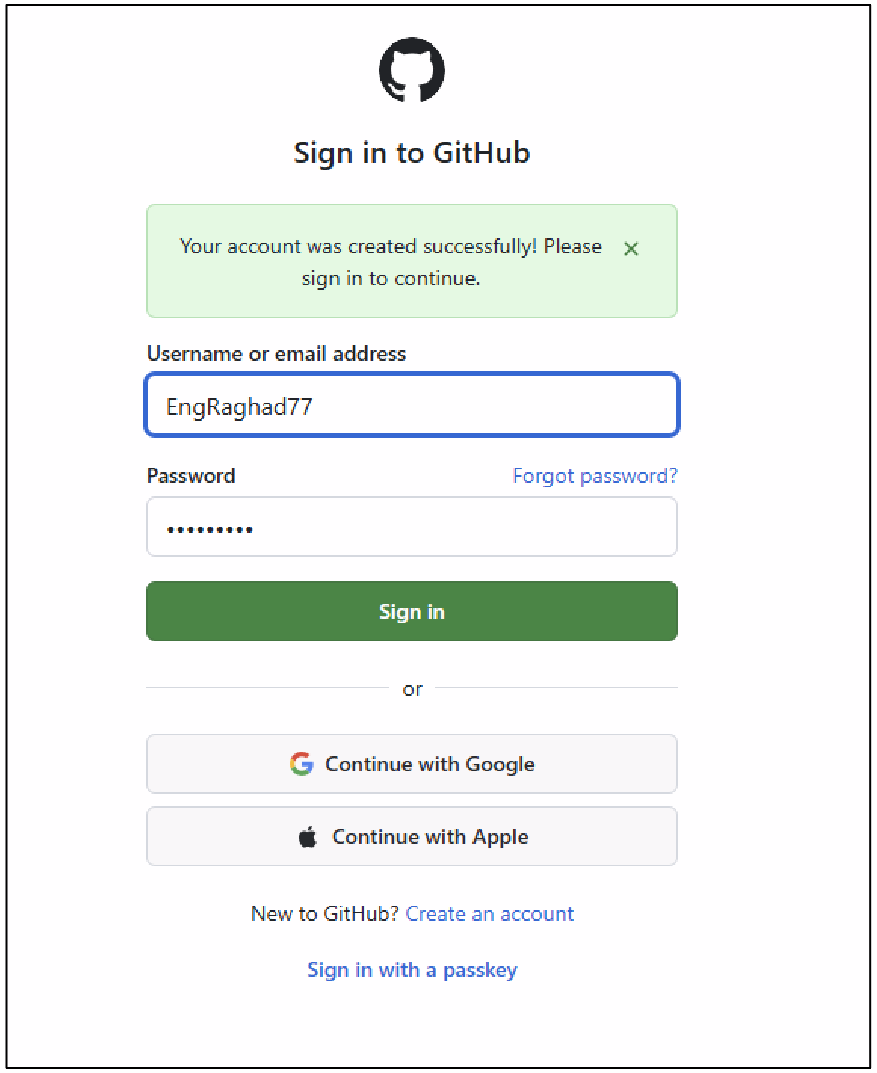
We have sent a code to **chatgbtsubs@gmail.com**

Enter code

Didn't get your email? Resend the code or update your email address.

Create a GitHub account – log in and homepage :

3



Sign in to GitHub

Your account was created successfully! Please sign in to continue. ×

Username or email address

EngRaghad77

Password Forgot password?

•••••••••

Sign in

or

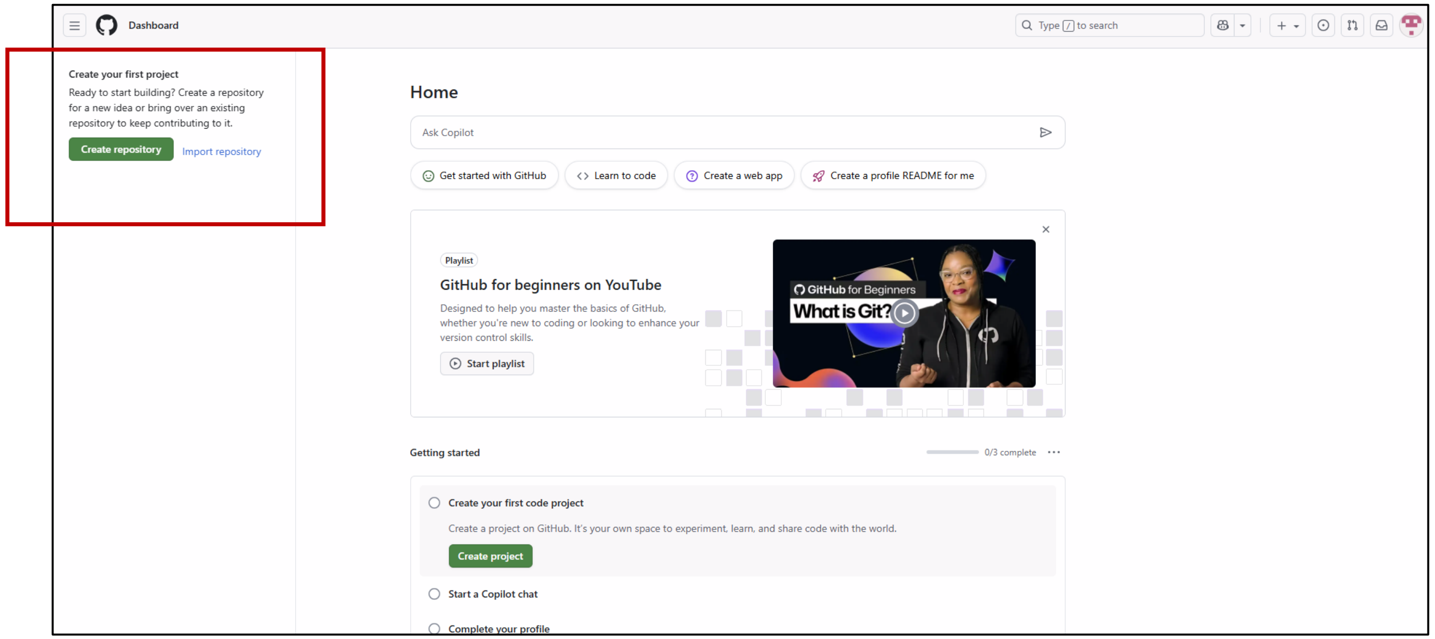
Continue with Google

Continue with Apple

New to GitHub? Create an account

Sign in with a passkey

4 Create repo



Dashboard

Type / to search

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository Import repository

Home

Ask Copilot

Get started with GitHub · Learn to code · Create a web app · Create a profile README for me

Playlist

GitHub for beginners on YouTube

Designed to help you master the basics of GitHub, whether you're new to coding or looking to enhance your version control skills.

Start playlist

Getting started 0/3 complete

Create your first code project

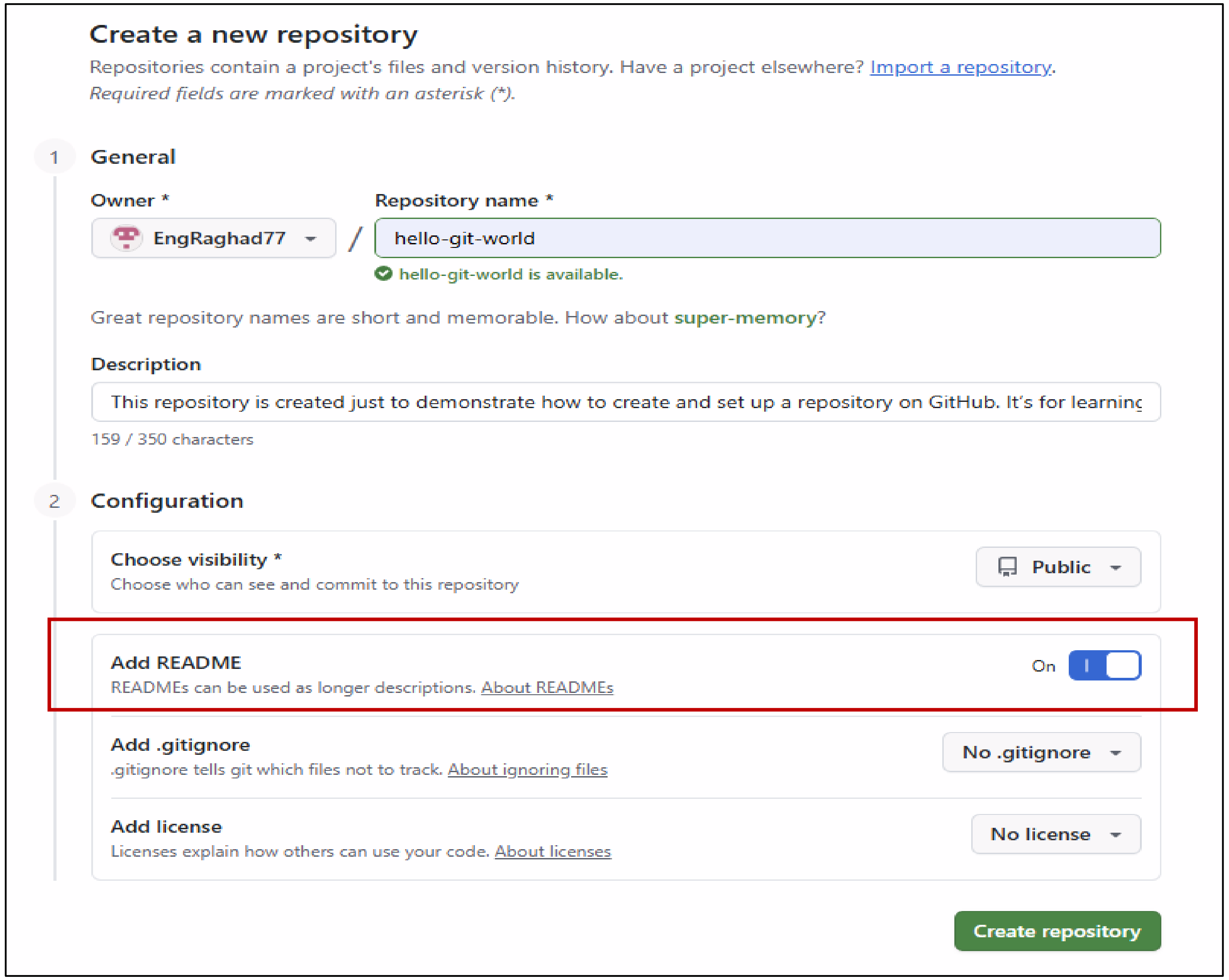
Create a project on GitHub. It's your own space to experiment, learn, and share code with the world.

Create project

Start a Copilot chat

Complete your profile

Create a GitHub account - create repository:



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner *: EngRaghad77 / Repository name *: hello-git-world

hello-git-world is available.

Great repository names are short and memorable. How about super-memory?

Description: This repository is created just to demonstrate how to create and set up a repository on GitHub. It's for learning

159 / 350 characters

2 Configuration

Choose visibility *: Public
Choose who can see and commit to this repository

Add README: On
READMEs can be used as longer descriptions. About READMEs

Add .gitignore: No .gitignore
.gitignore tells git which files not to track. About ignoring files

Add license: No license
Licenses explain how others can use your code. About licenses

Best practices with naming a repository (naming conventions):

General Naming Rules:

1. **Format: All repository names should be in lowercase and use slug-case (words separated by hyphens). This improves readability and consistency across different environments.**

2. **Naming Structure:**
{project}-{general|service}-{service-name}-{type}

Create a GitHub account - create repository:

Naming conventions – project type:

{project}-{general|service}-{service-name}-{type}

Backend projects

1. General Backend:
Pattern: {project}-general-api

Example: example-general-api

2. Microservice Backend:
Pattern: {project}-service-{service-name}-{api}

Examples:
example-service-user-api (API for user services)

frontend projects

1. Pattern: {project}-fe
Examples: example-{service}-fe

Mobile projects

Pattern: {project}-mobile-{platform}
Examples:
example-mobile-flutter (Flutter mobile application)
example-mobile-ios (iOS mobile application)

Create a GitHub account - create repository:

Naming conventions – project type:

{project}-{general|service}-{service-name}-{type}

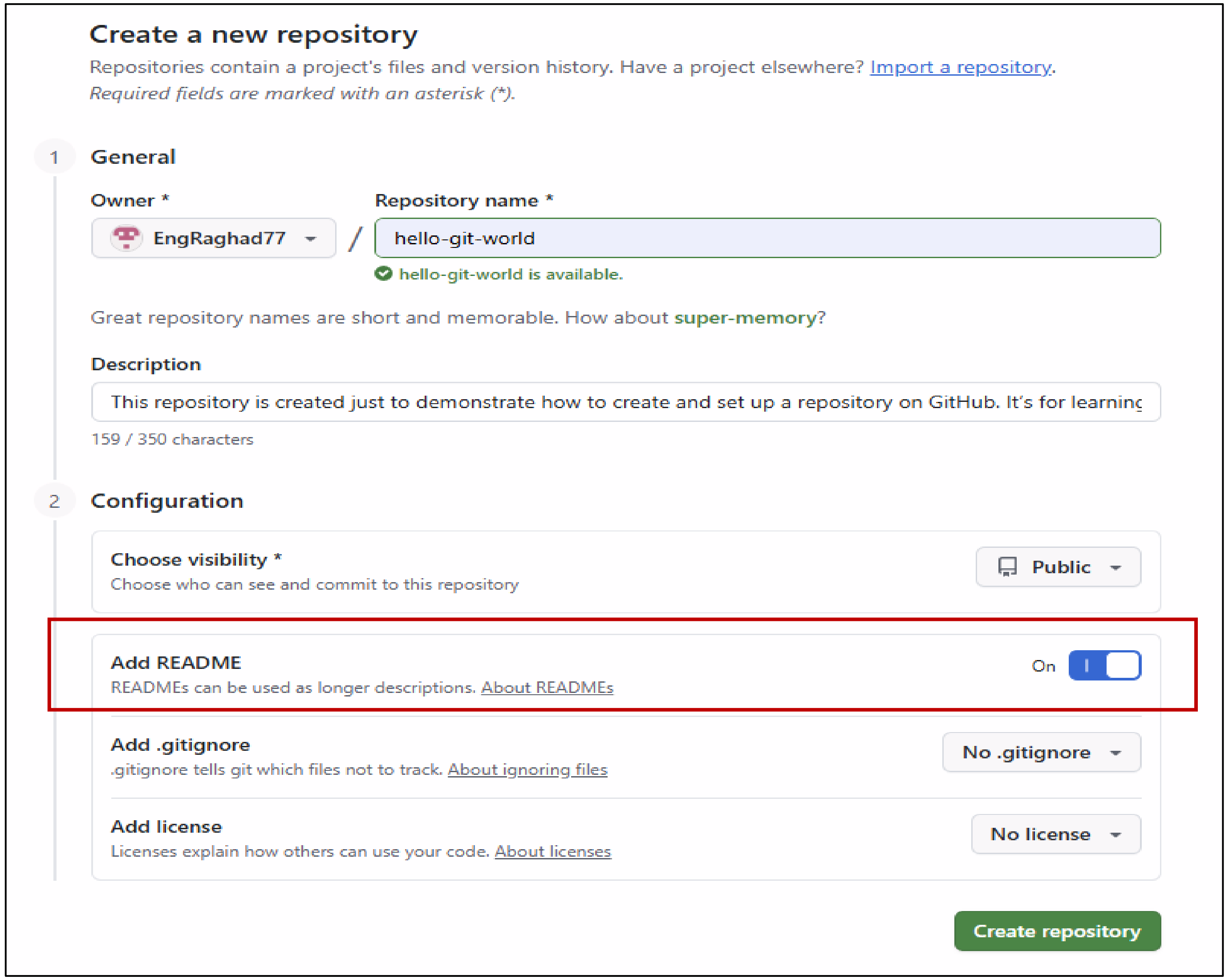
Desktop Projects

Pattern: {project}-desktop-{framework}
Examples:
example-desktop-electron (Electron-based desktop application)
example-desktop-dotnet (Desktop application built with .NET)

Full-stack projects

Pattern: {project}-app
Example: example-app

Create a GitHub account - create repository:



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner *: EngRaghad77

Repository name *: hello-git-world

hello-git-world is available.

Great repository names are short and memorable. How about super-memory?

Description: This repository is created just to demonstrate how to create and set up a repository on GitHub. It's for learning

159 / 350 characters

2 Configuration

Choose visibility *: Public

Choose who can see and commit to this repository

Add README: On

READMEs can be used as longer descriptions. About READMEs

Add .gitignore: No .gitignore

.gitignore tells git which files not to track. About ignoring files

Add license: No license

Licenses explain how others can use your code. About licenses

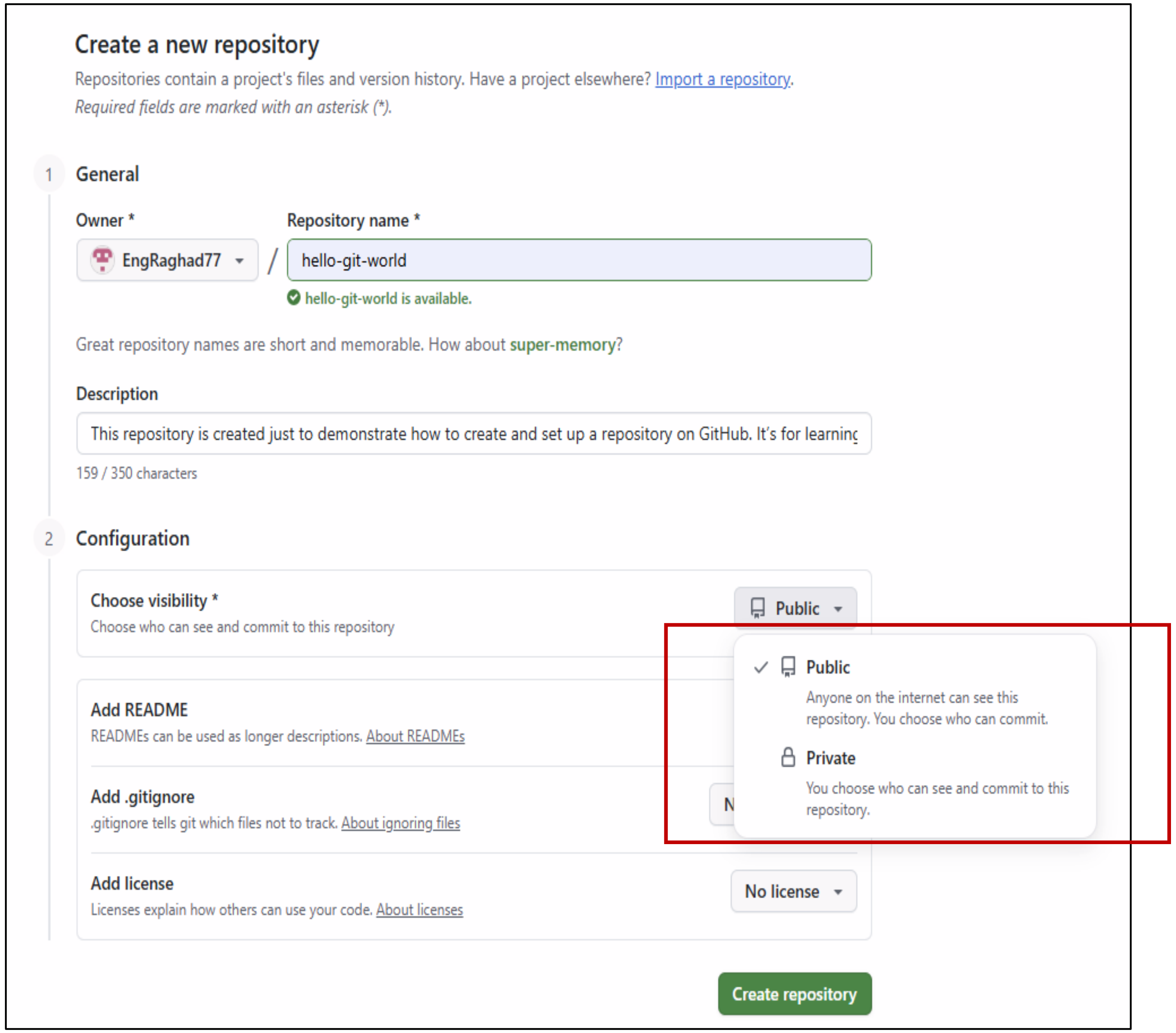
Readme file:

You can add a README file to a repository to communicate important information about your project.

A README is often the first item a visitor will see when visiting your repository. README files typically include information on:

- **What the project does**
- **Why the project is useful**
- **How users can get started with the project**
- **Where users can get help with your project**
- **Who maintains and contributes to the project**

Create a GitHub account - create repository:



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner * EngRaghad77 / Repository name * hello-git-world

hello-git-world is available.

Great repository names are short and memorable. How about **super-memory**?

Description

This repository is created just to demonstrate how to create and set up a repository on GitHub. It's for learninç

159 / 350 characters

2 Configuration

Choose visibility * — Public

Choose who can see and commit to this repository

✓ 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.

Add README — READMEs can be used as longer descriptions. About READMEs

Add .gitignore — .gitignore tells git which files not to track. About ignoring files

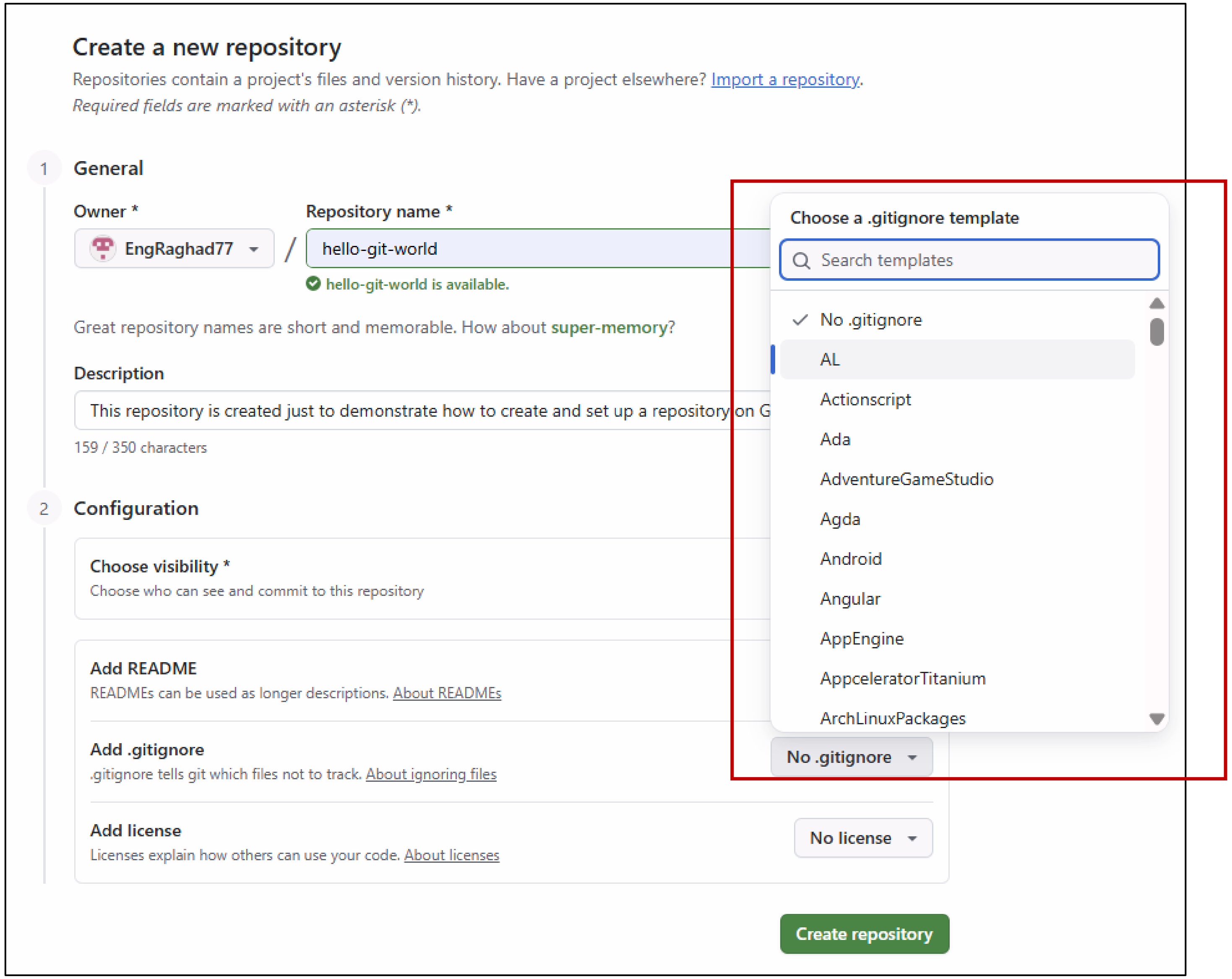
Add license — Licenses explain how others can use your code. About licenses — No license

Public repository vs. private repository:

Public just means that anyone on the Internet can see the repository. you still have control over who can make changes to it. It's just available on the viewable aspect of it on the Internet.

The next option is private, meaning it's not available for anyone to see.
You can only allow access by granting people access to the repository.

Create a GitHub account - create repository:



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner * EngRaghad77 / Repository name * hello-git-world

hello-git-world is available.

Great repository names are short and memorable. How about **super-memory**?

Description

This repository is created just to demonstrate how to create and set up a repository on G

159 / 350 characters

2 Configuration

Choose visibility *

Choose who can see and commit to this repository

Add README

READMEs can be used as longer descriptions. About READMEs

Add .gitignore

.gitignore tells git which files not to track. About ignoring files

No .gitignore

Add license

Licenses explain how others can use your code. About licenses

No license

Choose a .gitignore template

Search templates

✓ No .gitignore

AL

Actionscript

Ada

AdventureGameStudio

Agda

Android

Angular

AppEngine

AppceleratorTitanium

ArchLinuxPackages

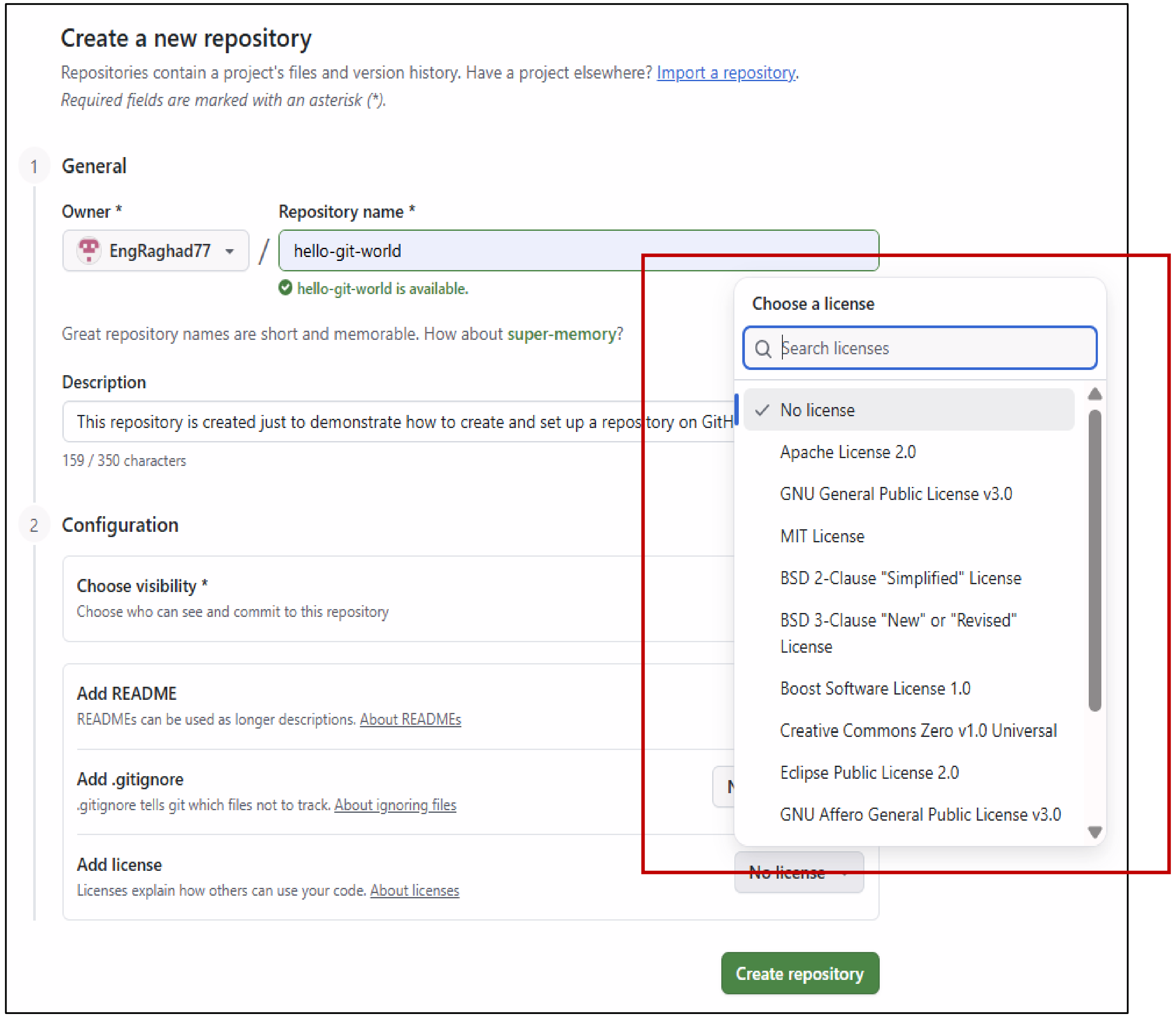
Create repository

.gitignore file

You can create a .gitignore file in your repository's root directory to tell Git which files and directories to ignore when you make a commit. To share the ignore rules with other users who clone the repository, commit the .gitignore file in to your repository.

GitHub maintains an official list of recommended .gitignore files for many popular operating systems, environments, and languages in the "github/gitignore" public repository.

Create a GitHub account - create repository:



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner * EngRaghad77 / Repository name * hello-git-world

hello-git-world is available.

Great repository names are short and memorable. How about super-memory?

Description

This repository is created just to demonstrate how to create and set up a repository on GitH

159 / 350 characters

2 Configuration

Choose visibility *
Choose who can see and commit to this repository

Add README
READMEs can be used as longer descriptions. About READMEs

Add .gitignore
.gitignore tells git which files not to track. About ignoring files

Add license
Licenses explain how others can use your code. About licenses

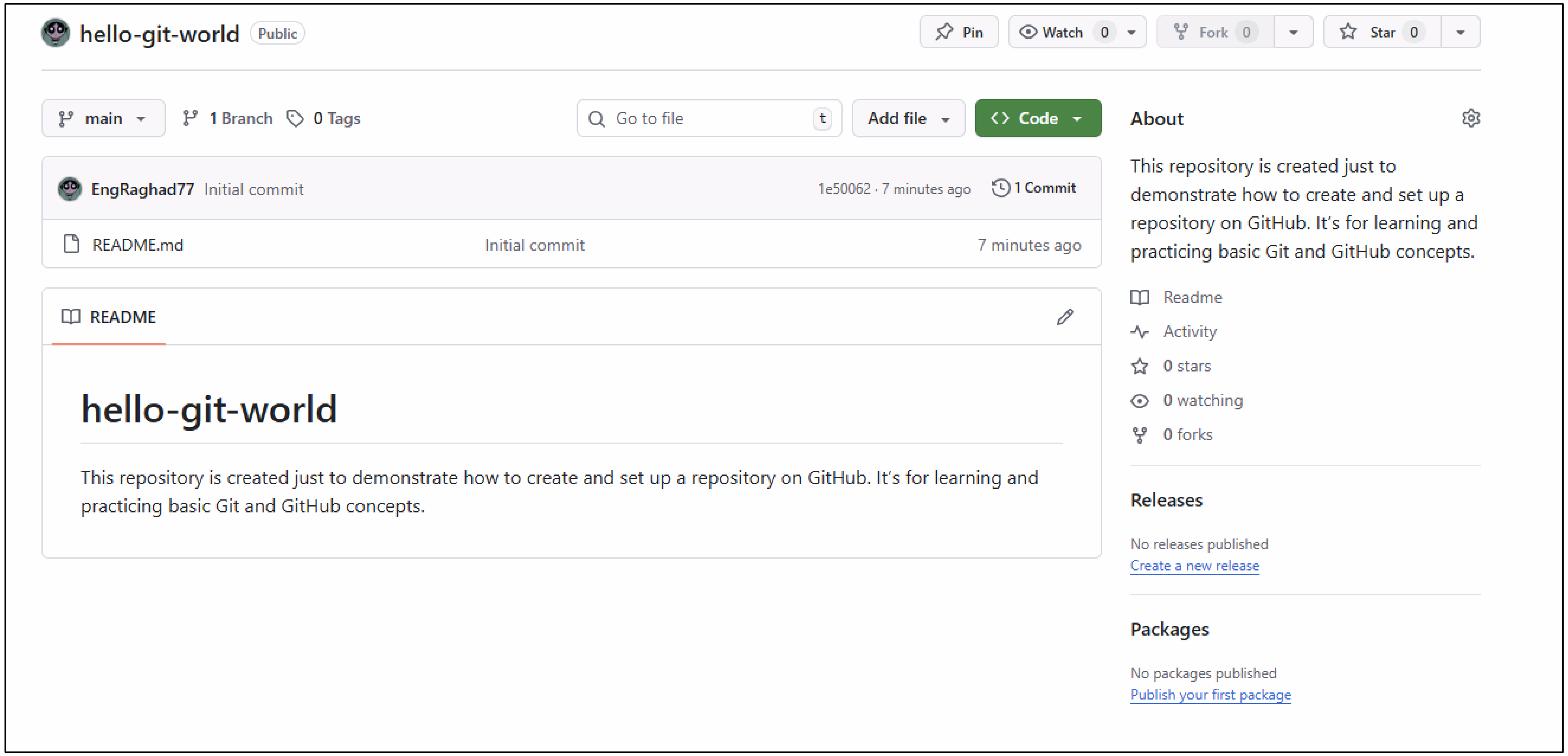
Choose a license
Search licenses
✓ No license
Apache License 2.0
GNU General Public License v3.0
MIT License
BSD 2-Clause "Simplified" License
BSD 3-Clause "New" or "Revised" License
Boost Software License 1.0
Creative Commons Zero v1.0 Universal
Eclipse Public License 2.0
GNU Affero General Public License v3.0
No license
Create repository

License

Public repositories on GitHub are often used to share open source software. For your repository to truly be open source, you'll need to license it so that others are free to use, change, and distribute the software.

A software license tells others what they can and can't do with your source code, so it's important to make an informed decision.

Create a GitHub account - create repository done:



EngRaghad77 Initial commit 1e50062 · 7 minutes ago 1 Commit

README.md Initial commit 7 minutes ago

README

hello-git-world

This repository is created just to demonstrate how to create and set up a repository on GitHub. It's for learning and practicing basic Git and GitHub concepts.

About

This repository is created just to demonstrate how to create and set up a repository on GitHub. It's for learning and practicing basic Git and GitHub concepts.

Readme
Activity
0 stars
0 watching
0 forks

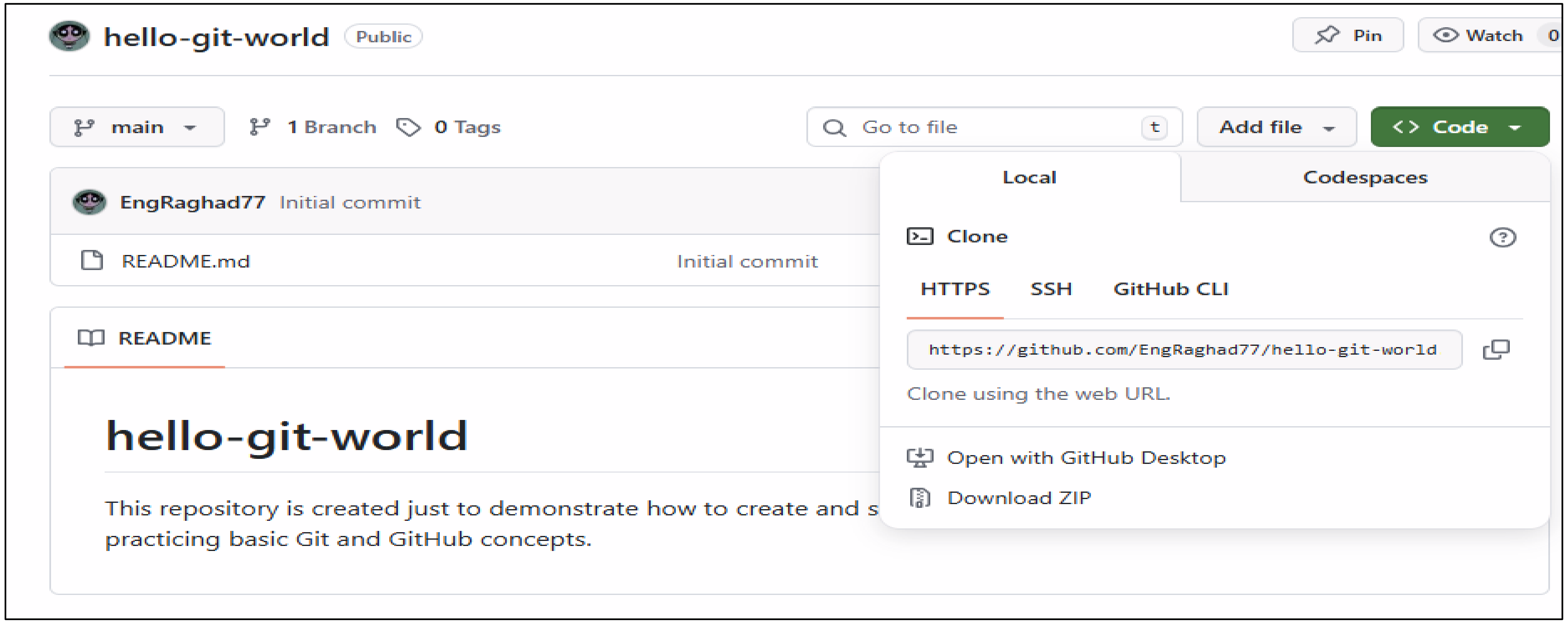
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Create a GitHub account - clone the public repo with HTTPs:



https://github.com/EngRaghad77/hello-git-world

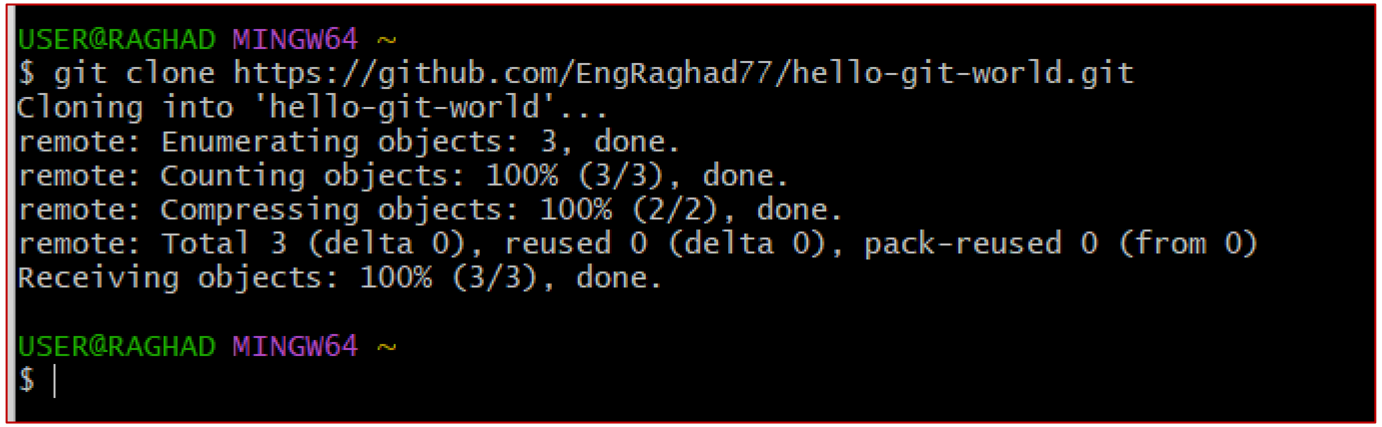
Clone using the web URL.

Open with GitHub Desktop

Download ZIP

hello-git-world

This repository is created just to demonstrate how to create and s
practicing basic Git and GitHub concepts.



```
USER@RAGHAD MINGW64 ~
$ git clone https://github.com/EngRaghad77/hello-git-world.git
Cloning into 'hello-git-world'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

USER@RAGHAD MINGW64 ~
$ |
```

Task 02 :)

- Create a new folder named python_project and initialize a Git repository inside it.
- Create a file file1.py that prints the message Hello, World!
- Stage and commit only file1.py with a clear commit message.
- Create a file file2.py that prints your full name and a short sentence about you.
- Stage and commit only file2.py with a meaningful commit message.
- Create a file file3.py that prints today's date.
- Stage and commit only file3.py with a descriptive commit message.
- Verify that your repository shows exactly three commits in the history.
- Run each file individually to confirm they produce the expected output text.
- Take a screenshot of your commit history (compact view) and submit it with your three files.