



WEB DEVELOPMENT AND DESIGN

MODULE 1A – GIT AND VERSION CONTROL



Table of Contents

Section 1. Getting started with Git	2
Section 2. Create your first repository, then add and commit	2
files.....	2
Lesson 1: Installing Git	2
Lesson 2: Setting your user name and email	4
Lesson 3: Creating Your First Repository	4

Section 1. Getting started with Git

In this first section, we're going to explore what you'll learn in this course. You'll learn the Basics of GIT and Version Control and we'll explore what's covered in this class.

Section 2. Create your first repository, then add and commit

Lesson 1: Installing Git

Git for Windows stand-alone installer

- Create and Account on [GitHub](#)
- Download the latest [Git for Windows installer](#).
- When you've successfully started the installer, you should see the Git Setup wizard screen.
- Follow the Next and Finish prompts to complete the installation. The default options are pretty sensible for most users.
- Open a Command Prompt (or Git Bash if during installation you elected not to use Git from the Windows Command Prompt).
- Run the following commands to configure your Git username and email using the following commands, replacing Emma's name with your own. These details will be associated with any commits that you create:

```
$ git config --global user.name "Emma Paris"
$ git config --global user.email "eparis@atlassian.com"
```

At the command line, first verify that you have Git installed:

On all operating systems:

```
git -version
```

```
git init
```

- This creates a hidden folder, .git, which contains the plumbing needed for Git to work. Next, check what files Git will add to your new repository; this step is worth special care:

git status

- Review the resulting list of files; you can tell Git which of the files to place into version control (avoid adding files with confidential information such as passwords, or files that just clutter the repo):

git add <file/directory name #1> <file/directory name #2> < ... >

- If all files in the list should be shared with everyone who has access to the repository, a single command will add everything in your current directory and its subdirectories:

git add .

- This will "stage" all files to be added to version control, preparing them to be committed in your first commit.
- For files that you want never under version control, create and populate a file named .gitignore before running the add command.

Commit all the files that have been added, along with a commit message:

git commit -m "Initial commit"

- This creates a new commit with the given message. A commit is like a save or snapshot of your entire project. You can now push, or upload, it to a remote repository, and later you can jump back to it if necessary.
- If you omit the -m parameter, your default editor will open and you can edit and save the commit message there.

Adding a remote

To add a new remote, use the git remote add command on the terminal, in the directory your repository is stored at.

The git remote add command takes two arguments:

1. A remote name, for example, origin
2. A remote URL, for example, `https://<your-git-service-address>/user/repo.git`

```
git remote add origin https://<your-git-service-address>/owner/repository.git
```

Lesson 2: Setting your user name and email

You need to set who you are **before** creating any commit. That will allow commits to have them.

To declare that identity for all repositories, use `git config --global`

This will store the setting in your user's `.gitconfig` file: e.g. `$HOME/.gitconfig` or for Windows,

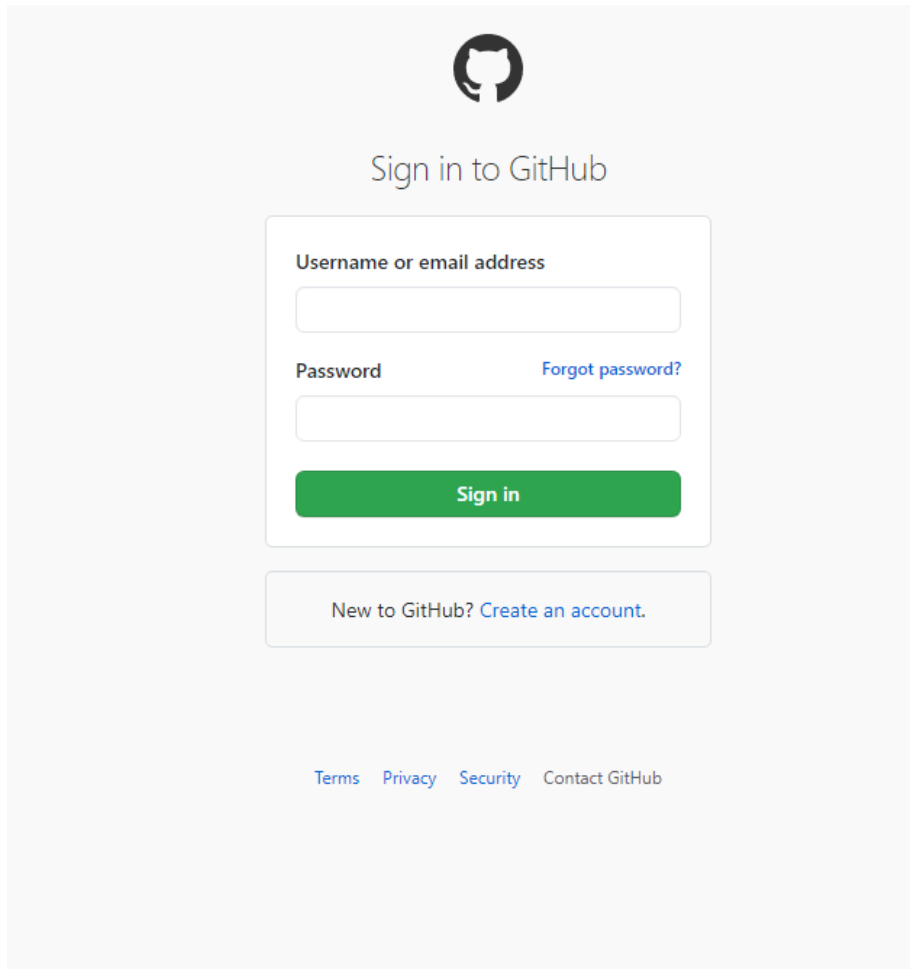
`%USERPROFILE%\gitconfig`.

```
git config --global user.name "Your Name"
```

```
git config --global user.email mail@example.com
```

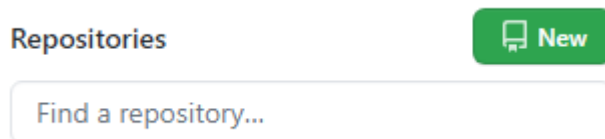
Lesson 3: Creating Your First Repository

1. Login your GitHub Account here:



The image shows the GitHub sign-in page. At the top center is the GitHub logo (an octocat). Below it, the text "Sign in to GitHub" is centered. The main form is a white box with a light gray border. It contains two input fields: "Username or email address" and "Password". To the right of the password field is a link "Forgot password?". Below the password field is a green button labeled "Sign in". At the bottom of the form is a link "New to GitHub? Create an account.". At the very bottom of the page, there are links for "Terms", "Privacy", "Security", and "Contact GitHub".


2. Click on New Repository



The image shows the GitHub "Repositories" page. The word "Repositories" is on the left. On the right is a green button with a computer icon and the word "New". Below these is a search bar with the placeholder text "Find a repository...".

3. Give your New Repository a name (e.g. portfolio-project)
4. Add a description (e.g. My Personal Portfolio Project)
5. Tick Add a README File.
6. Click Create Repository


Owner * Repository name *


 thehungrycoder225 / portfolio-projec ✓

Great repository names are short and memorable. Need inspiration? How about **ubiquitous-parakeet**?

Description (optional)

My Personal Portfolio Project

☒  **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.](#)

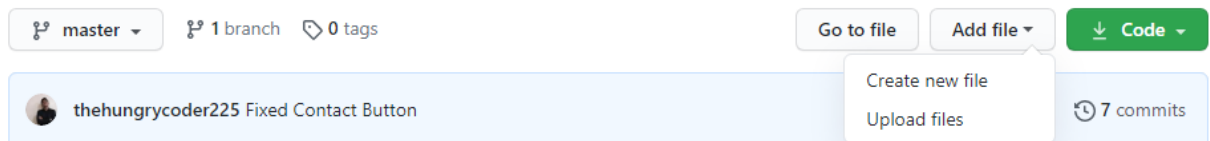
☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

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

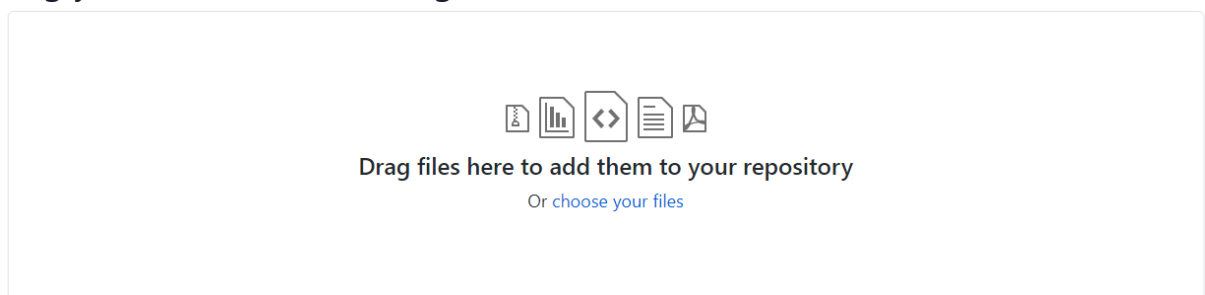
This will set  master as the default branch. Change the default name in your [settings](#).

Create repository

7. Select your repository and click on Add File > Upload Files



8. Drag your files here including folders



9. Add Commit Message (e.g. Initial Commit) > click on Commit changes

Commit changes

Add files via upload

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.](#)

Commit changes Cancel

10. Go to settings > Scroll Down to GitHub Pages

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾

Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme using the `gh-pages` branch. [Learn more.](#)

Choose a theme

11. From source select Master > /(root) > save

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

🔗 Branch: master ▾ 📁 /(root) ▾ Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme using the `gh-pages` branch. [Learn more.](#)

Choose a theme

12. GitHub will now Publish your webpage wait for at least 5mins (may take longer depends on the server load)

Your site is ready to be published at <https://thehungrycoder225.github.io/portfolio/>.

13. If you see this message it means you have successfully published your page on Git Hub

✓ Your site is published at <https://thehungrycoder225.github.io/portfolio/>

14. Click on the link and check whether your page has been published

15. Congrats on publishing your first webpage 🖥️.

