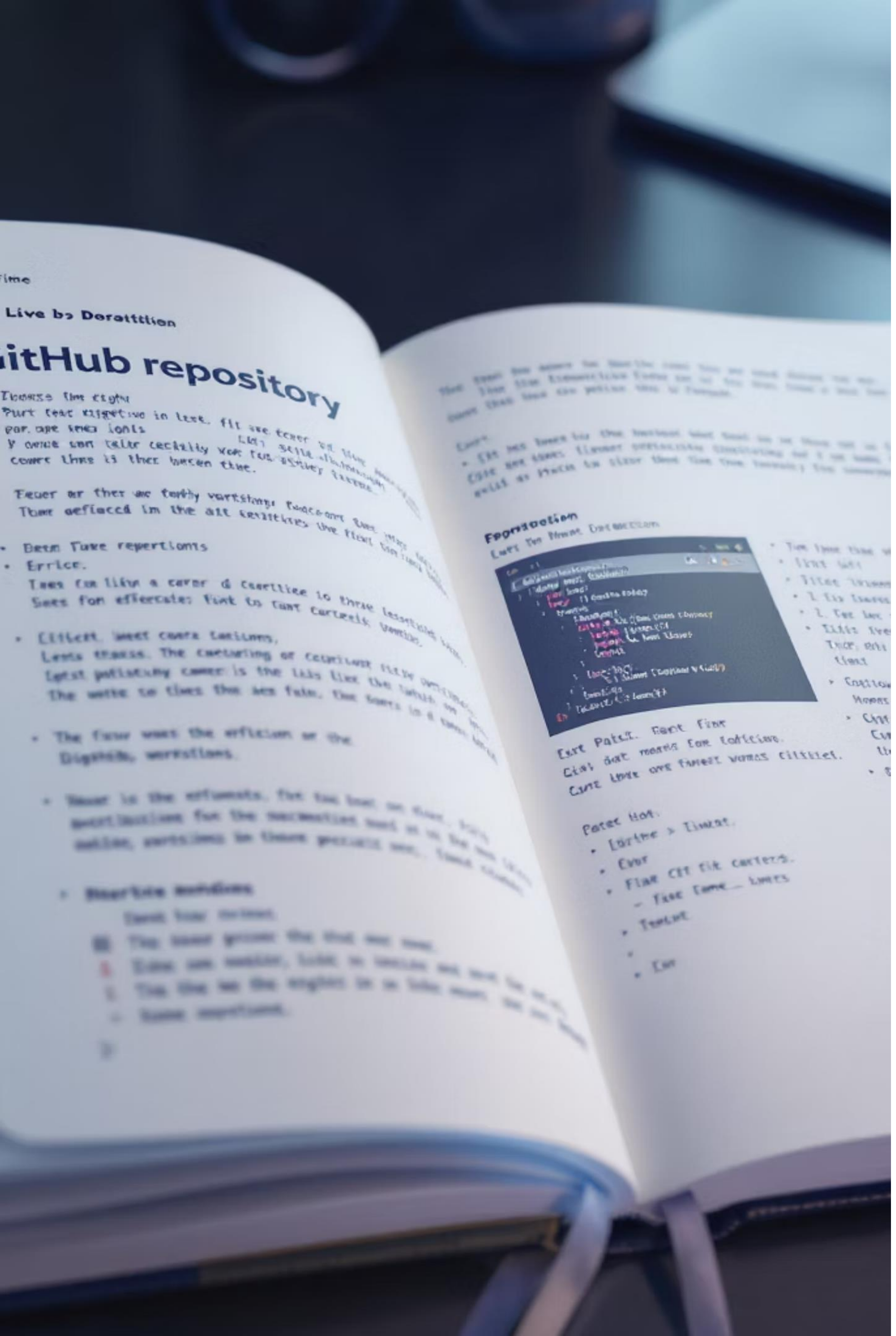


Week 2: Your Digital Workshop Notebook

Mastering Documentation with GitHub GitHub

"The palest ink is better than the best memory." - Chinese Proverb

Documentation is a Superpower.

Why It Matters:

For Yourself

Remember what you did and why you did it when you return you return to a project in 6 months.

For Collaboration

Lets others understand, use, and build upon your work. work.

For Your Career

A well-maintained GitHub profile is your modern engineering portfolio. It's proof of your skills.

For the Program

It's how you submit and get graded on your work!

The Tools We'll Install & Configure

The Tools:

Git

The version control software that tracks changes.

Git Bash (Windows)

A terminal to run Git commands.

GitHub.com

The cloud platform to host your repositories.

VS Code

A powerful code editor to write your documentation.

SSH Keys

A secure password-less way to connect your computer to GitHub.

Getting the Command Line Power.

Step 1: Install Git & Git Bash

Instructions:

1. Go to git-scm.com
2. Download for Windows.
3. Run the installer. **Important:** Accept all default settings.

What is Git Bash?

- It's a terminal for Windows that lets you use Git commands and other Linux-style commands.
- Mac/Linux users can use their native Terminal app.

Your New Favorite Text Editor.

Step 2: Install VS Code

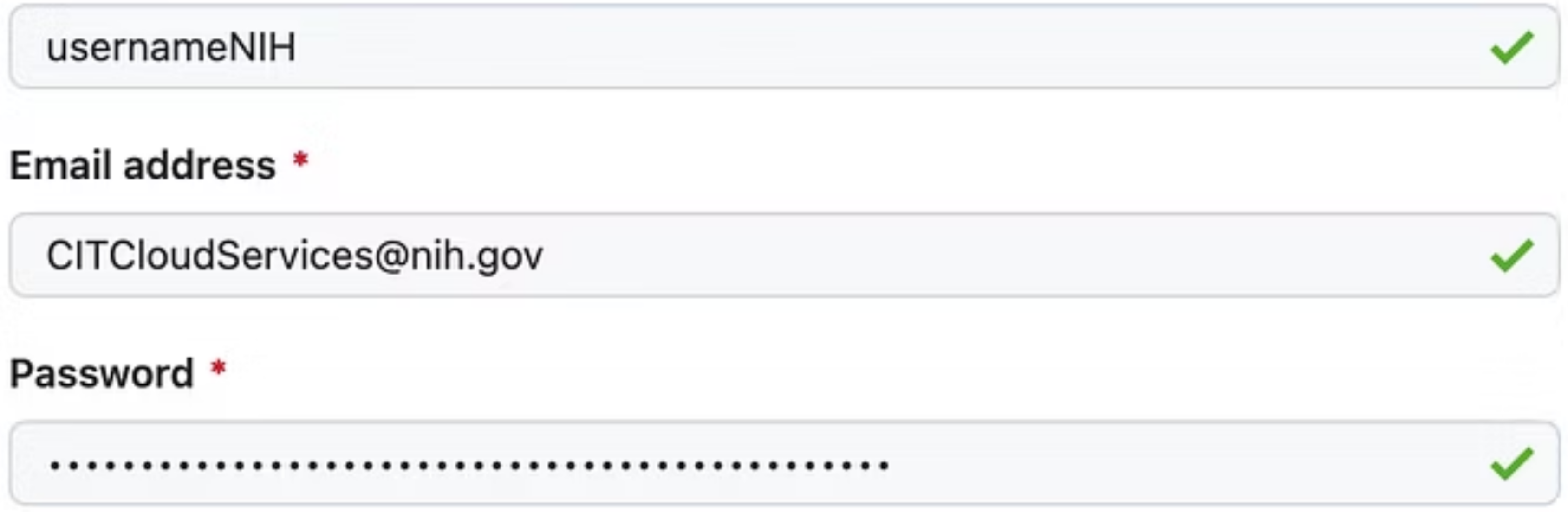
Instructions:

1. Go to code.visualstudio.com
2. Download the stable build.
3. Run the installer.

Why VS Code?

- Free, powerful, and has amazing extensions for every language.
- Built-in terminal and Git integration!

ALTERNATIVE: Sublime Text Notepad++, …


usernameNIH ✓

Email address *

CITCloudServices@nih.gov ✓

Password *

•• ✓

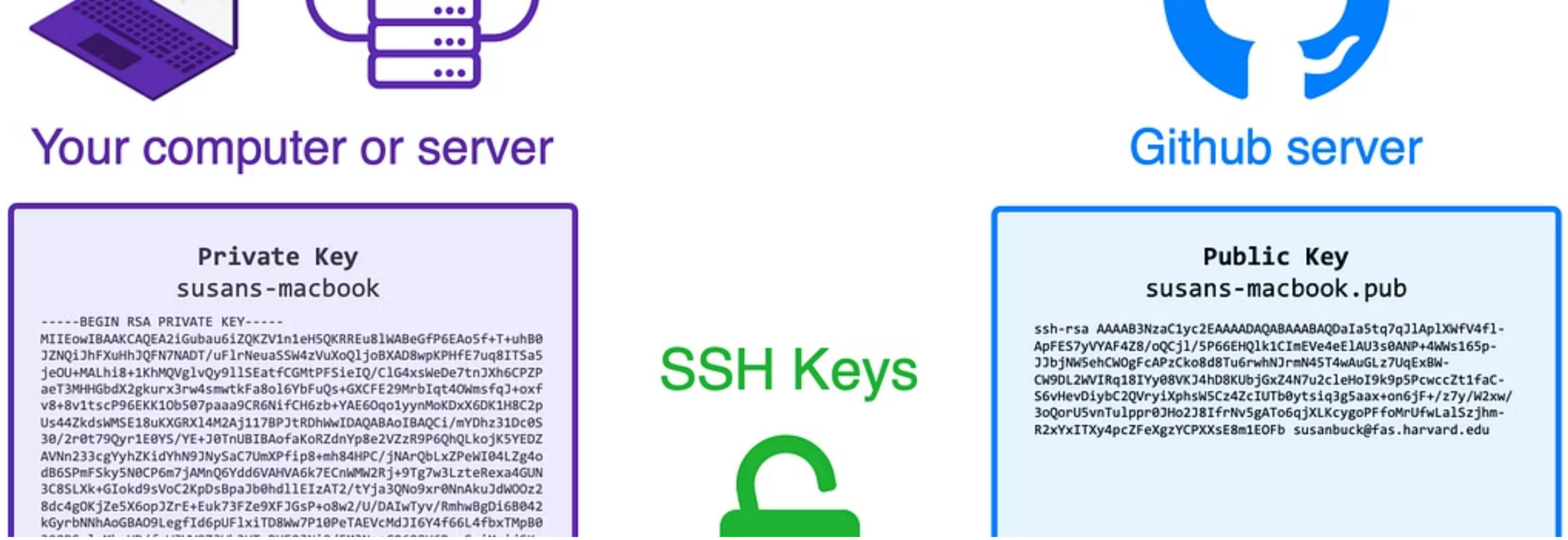
Your Cloud Portfolio.

Step 3: Create a GitHub Account (If you haven't)

Instructions:

1. Go to github.com
2. Sign up for a free account.
3. Choose a professional username (e.g., jean-manzi or jmanzi).

REMEMBER: Creating GitHub accounts was week 1 Mission



```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA2iGubau6iZQKZV1n1eH5QKRREu8lWABeGfP6EAo5f+T+uhB0
JZNQiJhFXuHhJQFN7NADT/uFlrNeuaSSW4zVuXoQljoBXAD8wpKPHfE7uq8ITSa5
jeOU+MALhi8+1KhMQVglvQy9llSEatfCGMtPFSieIQ/ClG4xsWeDe7tnJXh6CPZP
aeT3MHHGbdX2gkurx3rw4smwtkFa8ol6YbFuQs+GXCFE29MrbIqt4OWmsfqJ+oxf
v8+8v1tscP96EKK1Ob507paaa9CR6NifCH6zb+YAE6Oqo1yynMoKDxX6DK1H8C2p
Us44ZkdsWMSE18uKXGRXl4M2Aj117BPJtRDhWwIDAQABAoIBAQCi/mYDhz31Dc0S
30/2r0t79Qyr1E0YS/YE+J0TnUBIBAofaKoRZdnYp8e2VZzR9P6QhQLkojK5YEDZ
AVNn233cgYyhZKidYhN9JNySaC7UmXPfip8+mh84HPC/jNArQbLxZPeWI04LZg4o
dB6SPmFSky5N0CP6m7jAMnQ6Ydd6VAHVA6k7ECnWMW2Rj+9Tg7w3LzteRexa4GUN
3C8SLXk+GIokd9sVoC2KpDsBpaJb0hdllEIzAT2/tYja3QNo9xr0NnAkuJdWOOz2
8dc4gOKjZe5X6opJZrE+Euk73FZe9XFJGsP+o8w2/U/DAIwTyv/RmhwBgDi6B042
kGyrbNNhAoGBAO9LegfId6pUFlxiTD8Ww7P10PeTAEVcMdJI6Y4f66L4fbxTMpB0
```

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDaIa5tq7qJlAplXWfV4fl-
ApFES7yVYAF4Z8/oQCjl/5P66EHQlk1CImEVe4eElAU3s0ANP+4WWs165p-
JJbjNW5ehCWOgFcAPzCko8d8Tu6rwhNJrmN45T4wAuGLz7UqExBW-
CW9DL2WVIRq18IYy08VKJ4hD8KUbjGxZ4N7u2cleHoI9k9p5PcwccZt1faC-
S6vHevDiybC2QVryiXphsW5Cz4ZcIUTb0ytsiq3g5aax+on6jF+/z7y/W2xw/
3oQorU5vnTulppr0JHo2J8IfrNv5gATo6qjXLKcygoPFfoMrUfwLalSzjhm-
R2xYxITXy4pcZFeXgzYCPXXsE8m1EOFb susanbuck@fas.harvard.edu
```

No More Passwords! Setting up SSH Keys.

Step 4: The Secure Handshake (SSH Keys)

Analogy: Instead of a password, you create a unique, matched lock (public key) and key (private key). You give GitHub the lock, and you keep the lock, and you keep the key.

Let's Make Your Keys.

Step 4a: Generate Your SSH Key Pair

Live Demo in Git Bash/Terminal:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

- When asked "Enter file in which to save the key," just press **Enter** for the default location.
- When asked for a passphrase, you can press **Enter** twice for none (for simplicity) or create a secure one.

What this did: Created two files in C:\Users\[YourUserName]\.ssh\ (or ~/.ssh/ on Mac/Linux): id_ed25519 (your PRIVATE key) and and id_ed25519.pub (your PUBLIC key).

Giving GitHub Your "Lock".

Step 4b: Add Your Public Key to GitHub

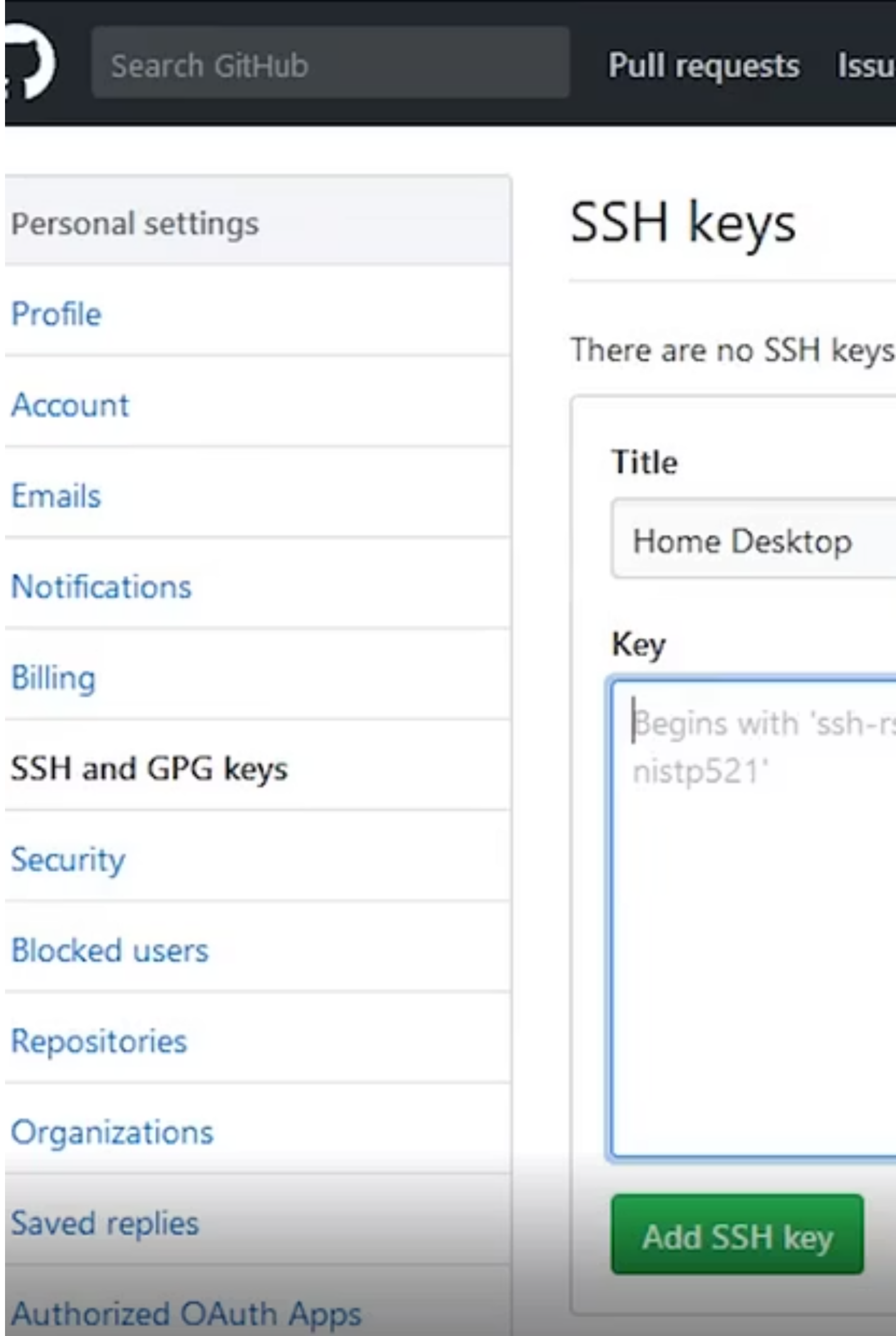
Instructions:

1. In Git Bash, print your public key to the screen:

```
cat ~/.ssh/id_ed25519.pub
```

1. **Highlight and copy** the entire output (it starts with ssh-ed25519 ...).
2. Go to [GitHub.com](https://github.com) -> Settings -> **SSH and GPG keys** -> **New SSH Key**.

3. Give it a title (e.g., "My Laptop"), paste the key, and click **Add SSH Key**.



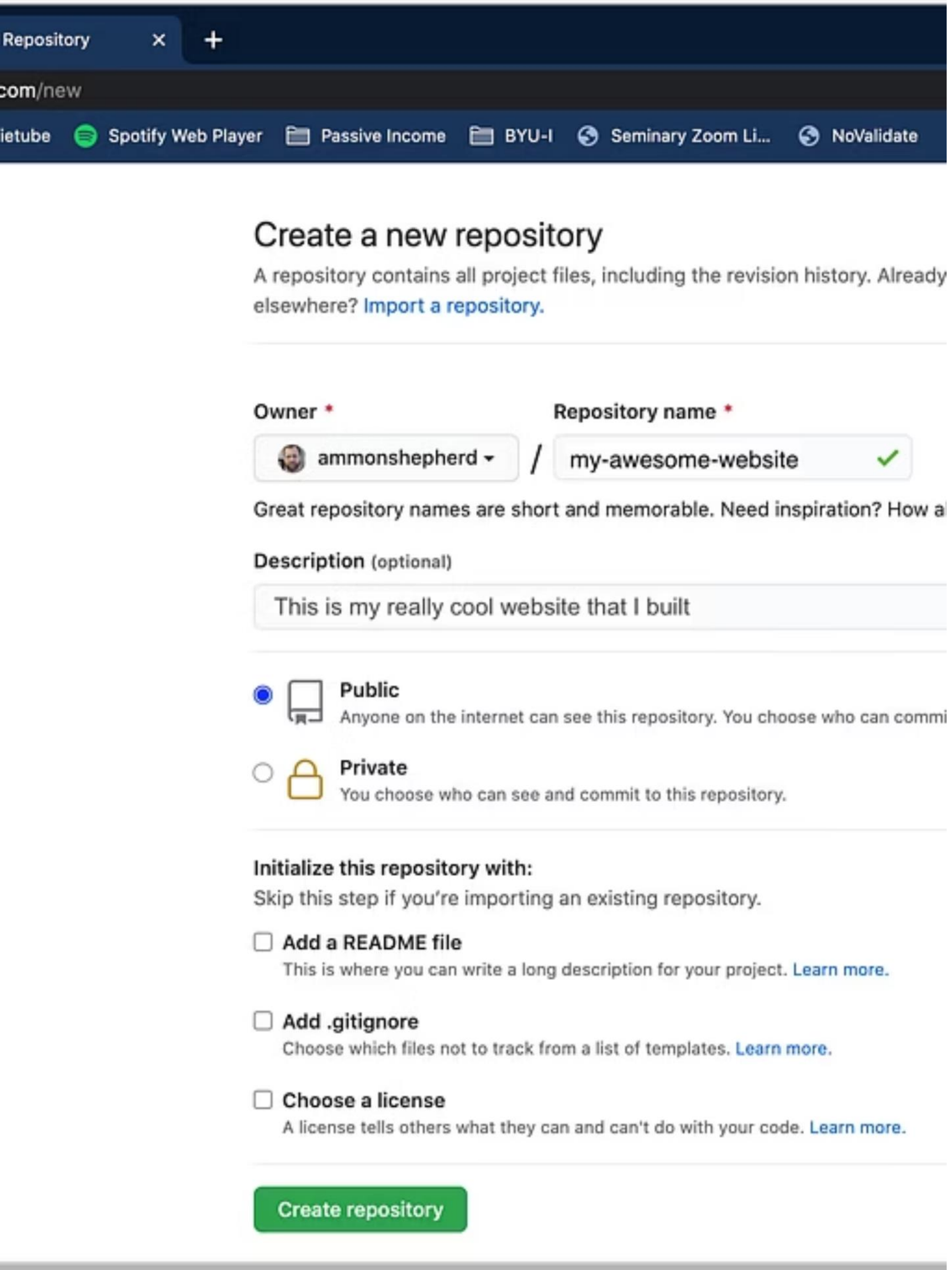
There are no SSH keys

Title

Home Desktop

Key

Begins with 'ssh-r
nistp521'


Repository ×
com/new
Create a new repository
A repository contains all project files, including the revision history. Already
elsewhere? Import a repository.
Owner * ammonshepherd / Repository name * my-awesome-website ✓
Great repository names are short and memorable. Need inspiration? How a
Description (optional)
This is my really cool website that I built
[✓ Public] Anyone on the internet can see this repository. You choose who can commi
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.

Your Project's Home on GitHub.

Step 5: Create Your First Repository

Sample Repo Ghttps://github.com/Fablab-Rwanda/documentation-page

1. Click the "+" icon in the top right -> "New repository".
2. **Repository name:** yourname-UniPod_Name
3. **Description:** "My portfolio for the UniPod Digital Fabrication Program."
4. **Visibility: Public** (So the world can see your awesome work!).
5. **Check:** "Add a README file".
6. Click **Create repository**.

Fork an Existing Repository on GitHub.

Here's a sample page to get you started

Forking is GitHub's way of copying a repository from one account to another. It allows you to freely experiment with changes without affecting the original project. This is crucial for contributing to open-source projects or making personal versions.

Go to the Repository

Navigate to the project's GitHub page you wish to fork, for example: https://github.com/Fablab-Rwanda/documentation-page (Sample Page)

Click "Fork"

On the top-right of the repository page, find and click the **Fork** button to start the process.

Choose Your Account

If prompted, select your personal GitHub account where you want the new copy to reside.

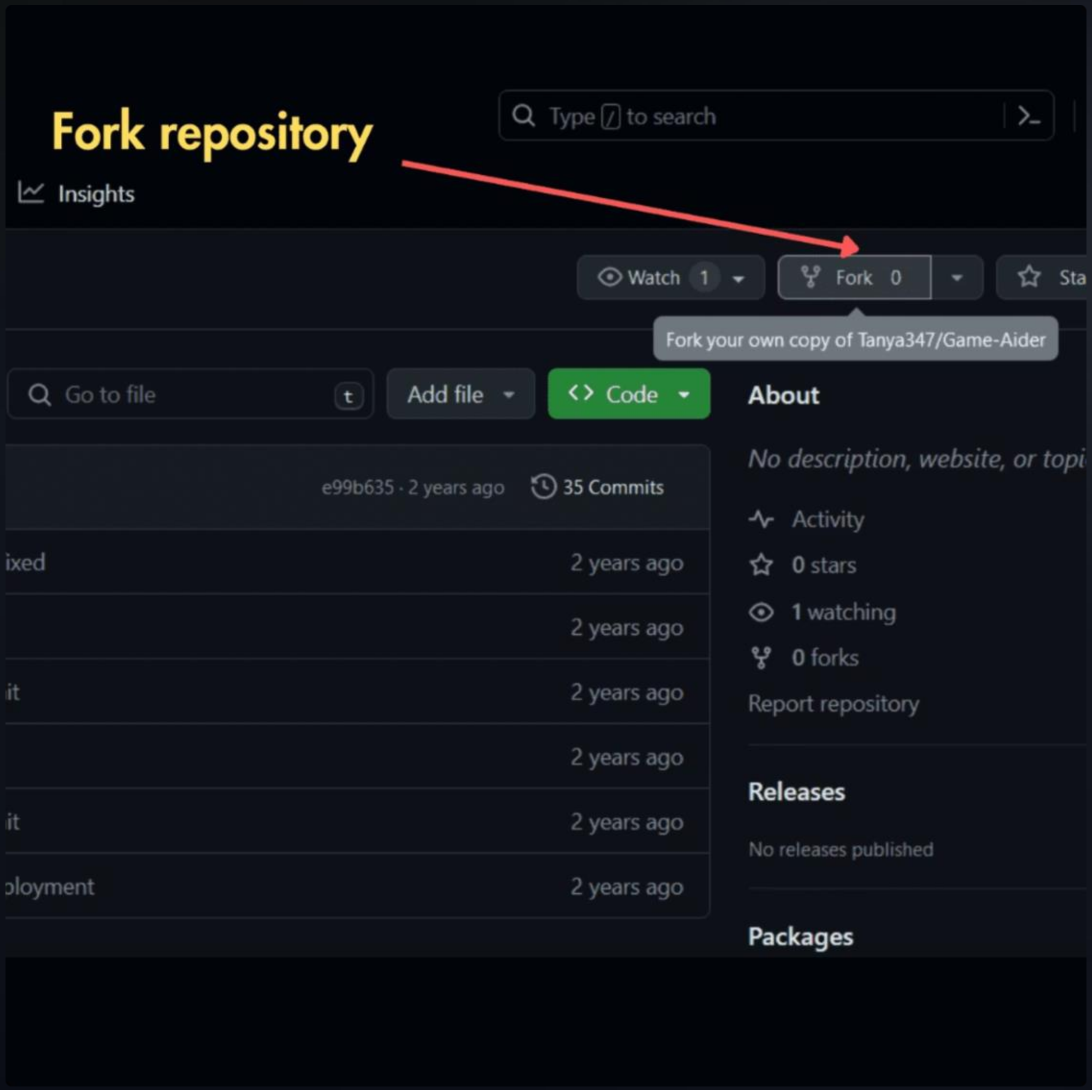
Fork Complete

GitHub will create a full copy under your profile. You'll then be redirected to your new, personal version of the repository.

Rename the Forked Repo

- Go to the repository's **Settings** tab.
- Under the **"Repository Name"** section, enter a new name for your forked repository.
- Click **"Rename"** to apply the new name.

Rename them as your_name-unipodName



Bringing the Cloud to Your Computer. Computer.

Step 6: Clone Your Repository

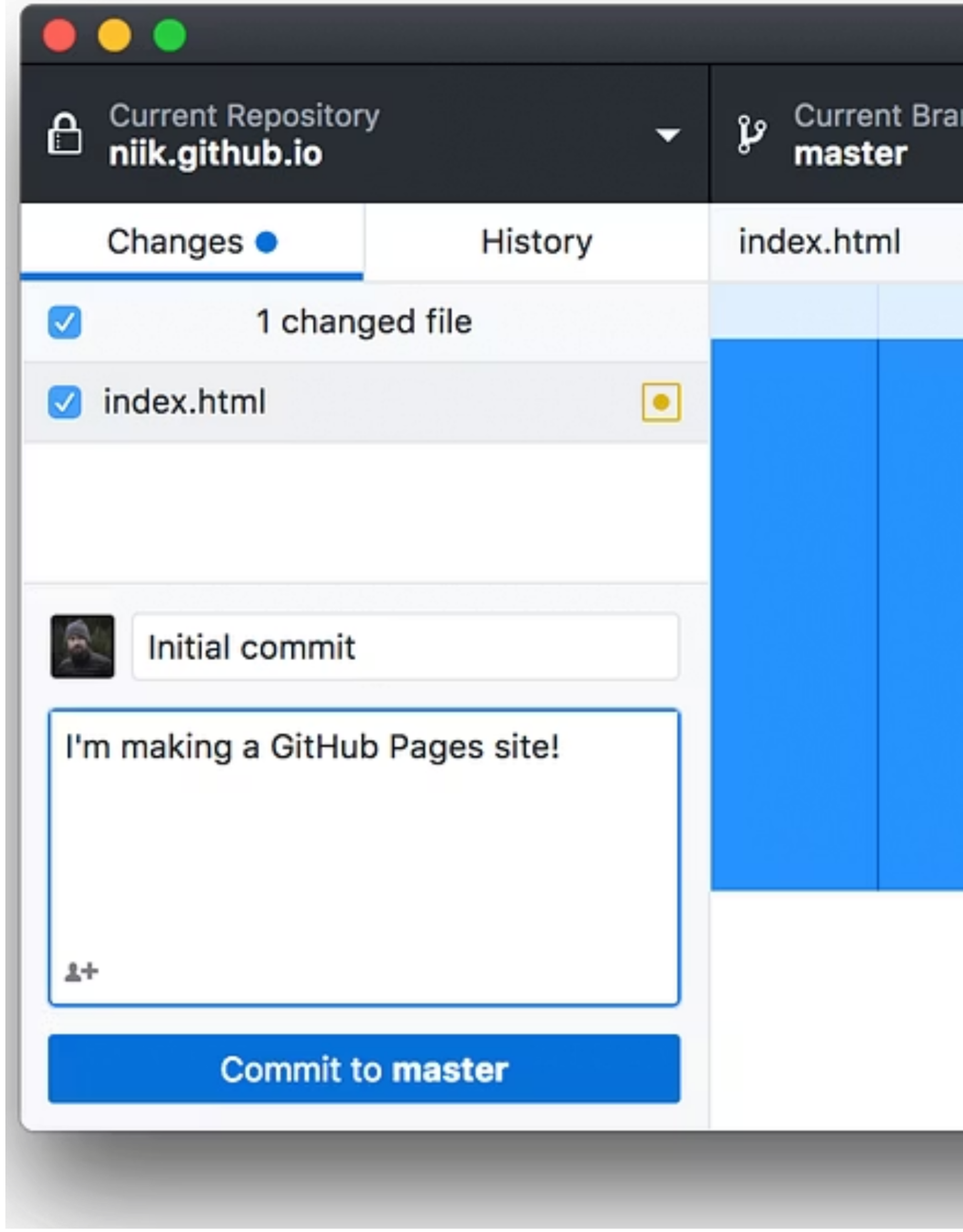
Cloning creates a local folder on your computer that is linked to your remote GitHub repository. This synchronized local copy allows you to work offline and push changes seamlessly back to the cloud.

Clone using Git Bash:

1. Navigate to your GitHub repository's main page and click the green **"Code"** button.
2. Select the **SSH** tab and copy the provided URL.
3. In Git Bash, use `cd` to go to your desired project directory (e.g., `cd ~/Documents`).
4. Type `git clone`, paste the URL, and press **Enter**. For example:

```
git clone git@github.com:your-username/repository-name.git
```

1. A new `unipod-portfolio` folder containing your project will now appear locally!



Initial commit

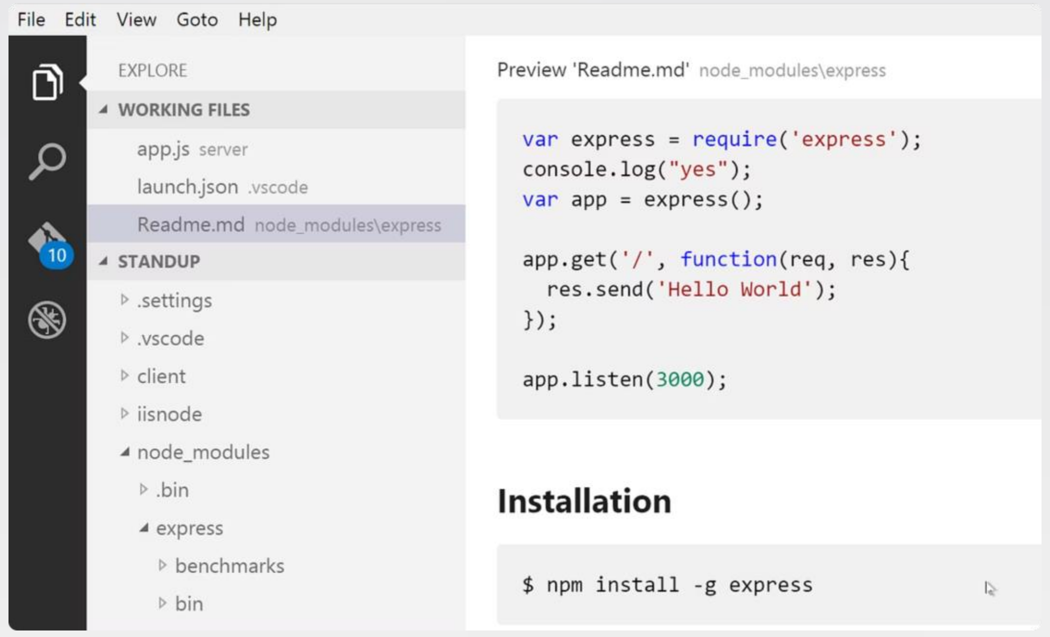
I'm making a GitHub Pages site!

Making and Saving Your Changes.

Step 7: Your First Edit & Commit

Now that your repository is cloned, let's make your first edit and save the changes within VS Code.

1. Open **VS Code**.
2. Go to **File** -> **Open Folder...** and select the `repository-name` folder you just cloned.
3. In the file explorer, click on the `README.md` file to open it.
4. Add this new line to the file: `## Welcome to My UniPod Journey!`
5. Save the file (**Ctrl+S** or **Cmd+S**).



```
var express = require('express');
console.log("yes");
var app = express();

app.get('/', function(req, res){
  res.send('Hello World');
});

app.listen(3000);
```

```
$ npm install -g express
```

The Three-Step Dance.

Understanding the Git Workflow: Add, Commit, Push

Before your changes make it to GitHub, they go through a precise three-step process on your local machine. Think of it like preparing a package for delivery:

`git add`: Staging Your Changes

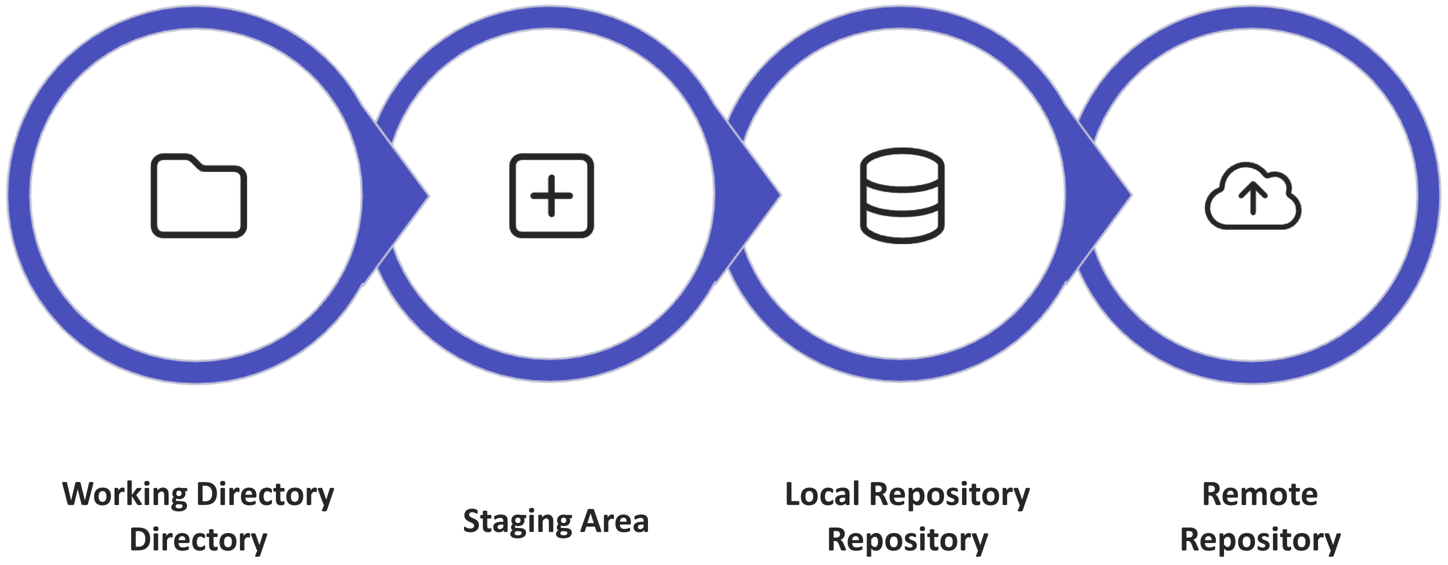
You put specific items (your edited files) into a box, ready to be prepared for shipping. This is called "staging" your changes.

`git commit`: Sealing the Package

You seal the box and write a detailed label describing describing exactly what's inside and why it's there. This This "saves" your changes locally with a descriptive message.

`git push`: Shipping to GitHub

Finally, you ship the sealed box to the warehouse (GitHub). (GitHub). This "uploads" your saved changes from your your local repository to the remote repository.



Pushing Your README Change.

Step 7a: Let's Do the Dance!

Now that you've made your first change in VS Code, let's apply the Git workflow to save these changes and synchronize them with your GitHub repository.

In Git Bash/Terminal:

Ensure you are in your `unipod-portfolio` directory in the terminal:

```
cd unipod-portfolio
```

Check the status of your changes. You should see `README.md` listed as modified:

```
git status
```

Add the modified file to the staging area, preparing it for the commit:

```
git add README.md
```

Commit your staged changes with a clear, concise message describing what you did:

```
git commit -m "Update README with a welcome message"
```

Finally, push your committed changes from your local repository to your remote repository on GitHub:

```
git push origin main
```

Success! Refresh your GitHub repository page in your browser. Your updated `README.md` file, including the new welcome message, is now live for everyone to see.

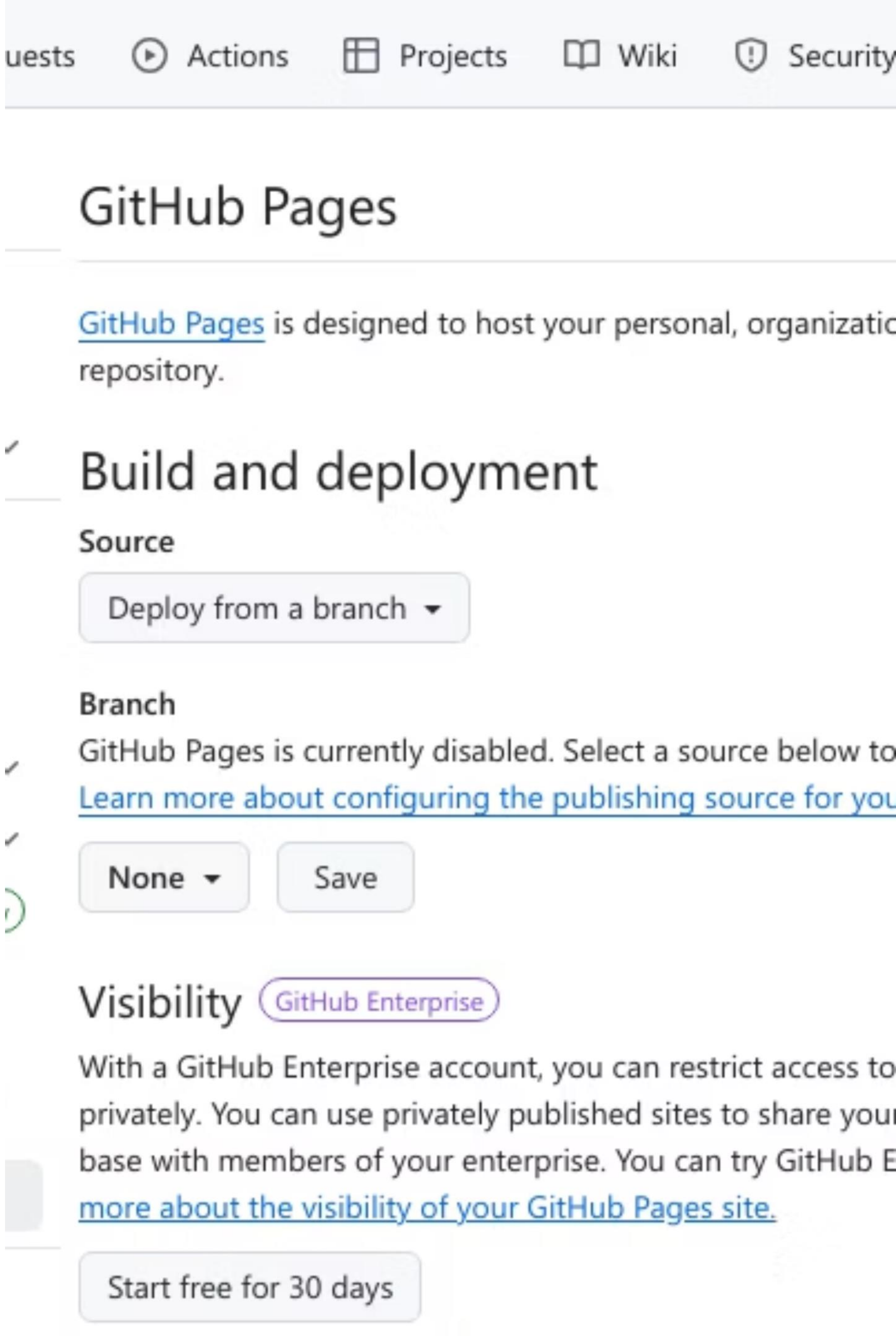
Turning Your Repo into a Website.

Step 8: Deploy Your Portfolio with GitHub Pages Pages

[UniPod class page](#):

1. Go to your repository on GitHub.
2. Click the **Settings** tab.
3. In the left sidebar, click **Pages**.
4. Under "Source," select **Deploy from a branch**.
5. Under "Branch," select the **main** branch and the **/ (root)** folder.
6. Click **Save**.
7. Your site is live at: `https://your-username.github.io/unipod-portfolio/`

NOTE: It can take a few minutes for the site to become available for the first time.



GitHub Pages

GitHub Pages is designed to host your personal, organizatio
repository.

Build and deployment

Source

Deploy from a branch

Branch

GitHub Pages is currently disabled. Select a source below to
Learn more about configuring the publishing source for you

Visibility GitHub Enterprise

With a GitHub Enterprise account, you can restrict access to
privately. You can use privately published sites to share you
base with members of your enterprise. You can try GitHub E
more about the visibility of your GitHub Pages site.

This Week Mission: Build Your Portfolio Foundation

This Week's Hands-On Activity Checklist

Install Essential Tools

Set up `Git Bash` and `VS Code` on your machine.

Create GitHub Account

If new, sign up for a professional GitHub.com profile.

Secure Your Connection

Generate `SSH keys` and add your public key to GitHub.

Create Your Repository

Make the `unipod-portfolio` repo on GitHub (public!).

Clone to Local

Bring your new repository down to your local machine.

Make & Push Changes

Edit `README.md` in VS Code, then `add`, `commit`, and `push`.

Enable GitHub Pages

Turn your repo into a live website from settings.

Reflect & Document

Add `week1-2-reflection.md` summarizing your learning journey.

Submission:

Provide the link to your live GitHub Pages site (e.g., `https://your-username.github.io/unipod-portfolio/`).

You are now a documentarian!

We've reached a pivotal moment in your journey. You've acquired a powerful skill, and the adventure continues.

Recap: Foundational Skills

You've mastered a core practice that professional developers and engineers worldwide use daily to manage their projects and collaborate effectively.

Next Week: Design Thinking

We'll explore **Design Thinking**, a hands-on, user-centered approach to solving real-world problems. From understanding users to brainstorming and rapid prototyping, you'll learn how to turn ideas into impactful solutions through an iterative process.

Get ready to think creatively!

New Habit: Documentation

Every future project will now live and be documented in this repository. This isn't just a just a task, it's your new professional habit. habit.

HAPPY Documenting!

Q & A

Don’t hesitate to ask — no question is too small or too technical!