Hands-On Guide: Taking Your Project from Git to GitHub

Project Setup & Collaboration Workflow Overview

This structured workflow guides you through setting up and collaborating on your **CodeTrack** project—a comprehensive journey from local preparation to remote collaboration on GitHub.

We will be completing multiple assignments to accomplish the following setup.

- CodeTrack— Initial Git Setup (Local Only)

  - This enables version control in your project directory, creating a .git folder and setting up the default branch (typically main) for managing your project's history.

- Tracking and Staging Changes in a CodeTrack Project in Git

  - This two-step process ensures that only reviewed changes move from the working directory into the repository history.

- Branching Workflow — Add & Verify a Contact Page in Git

  - Work within this branch without affecting the primary codebase, then commit and merge back once verified.

- Setting Up GitHub for CodeTrack

  - This enables remote collaboration, version centralization, and code sharing.

- Collaborating on Mini-Finance with GitHub

  - This mirrors GitHub Flow, fostering streamlined, traceable teamwork

Prerequisites

- Git installed (Git Bash on Windows, Terminal on macOS/Linux).

  - Install the latest Git from the url - https://git-scm.com/downloads

- Basic terminal navigation.

- Account in GitHub(Preferable)

Instructions

Assignment 5- CodeTrack— Initial Git Setup (Local Only)

Task 1 — Create a Local Project Directory

We will simulate the creation of a new **CodeTrack** project for new development.

- Navigate to your preferred working location (Documents/Desktop/projects).

Windows (Command Prompt or PowerShell or Git Bash):

1. mkdir CodeTrack

2. cd CodeTrack

macOS/Linux (Terminal):

1. mkdir CodeTrack && cd CodeTrack
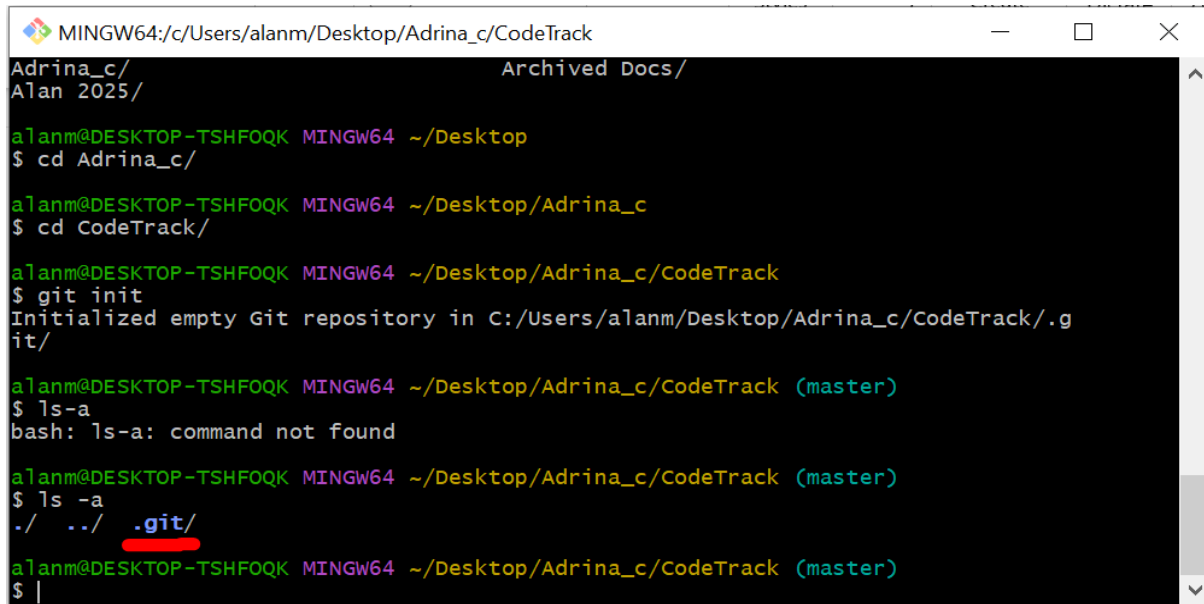
Initialize Git:

1. git init

Expected:

Initialized empty Git repository in …/CodeTrack/.git/

(Optional check)

1. ls -a   # or: dir /a (Windows)

You should see a hidden .git folder.

Screenshot A: Initialized empty Git repository(git init) || the .git folder listed with ls -a


```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack                    —    □    ×

Adrina_c/                        Archived Docs/
Alan 2025/

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop
$ cd Adrina_c/

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c
$ cd CodeTrack/

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack
$ git init
Initialized empty Git repository in C:/Users/alanm/Desktop/Adrina_c/CodeTrack/.g
it/

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ ls-a
bash: ls-a: command not found

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ ls -a
./   ../   .git/

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ |
```

Task 2 — Configure Git Locally for CodeTrack

Configure identity **only for this repo** (recommended for team/enterprise scenarios when identities differ by project).
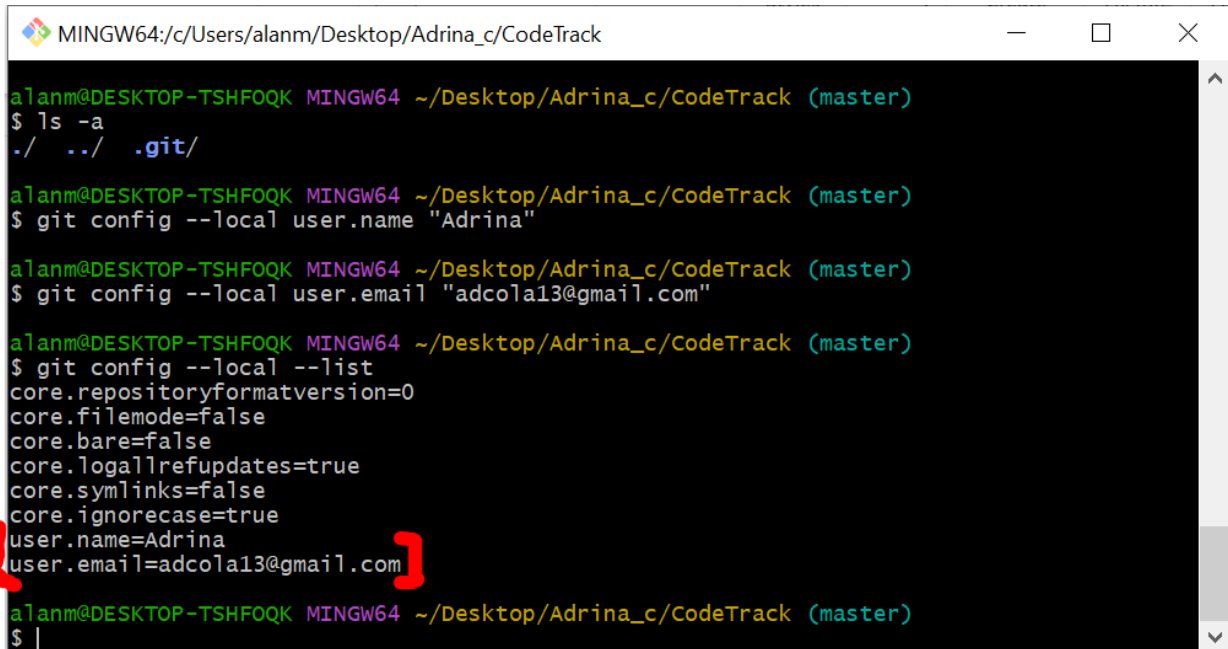
1. git config --local user.name "Your Name"

2. git config --local user.email "your.email@example.com"

3. git config --local --list

Expected: output includes:

1. user.name=Your Name

2. user.email=your.email@example.com

Tip: If you'll push to GitHub and want to keep your email private, use GitHub's noreply email (e.g., 12345678+username@users.noreply.github.com).

Screenshot B: Set up your local Git identity(git config --local –list)

Task 3 — Configure Git Globally (Optional, Recommended)

Sets your default identity for **all repos** on this machine.

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

2. git config --global user.email "your.email@example.com"

3. git config --global --list

Expected: output lists your global user.name and user.email.

Screenshot C: Set up your global Git identity(git config --global –list)

Global configuration is used to standardize your identity across all personal projects.

Local configuration is used to override your identity per project (e.g., work vs personal).

Assignment 6- Tracking and Staging Changes in a CodeTrack Project in Git

Task 0 – Navigate to Project Folder

Since you have already set up the Project Folder in the last assignment, navigate it:

- **On Windows (Command Prompt):**

    1. cd path\to\CodeTrack

- **On macOS/Linux (Terminal):**

    1. cd ~/path/to/CodeTrack

If you're unsure of your current directory, use the command below to check:

1. pwd

Task 1: Create and Modify Files

1. **Create two new files inside the CodeTrack directory:**

    1. touch index.html style.css

(For Windows users who don't have touch, use echo > index.html and echo > style.css.)

2. **Verify that the files have been created by listing the directory contents:**

    1. ls

Expected Output: You should see **index.html** and **style.css** in the file list.

3. **Modify index.html and style.css using a text editor.**

    - Go to GitHub and open the repository named "**Week-2---Git-GitHub-Assignment**".

- Copy everything from the index.html file and the style.css file, then paste them into the same files in your own project.

Task 2: Track the Files Using Git

1. **Check the repository status:**

   1. git status

Expected Output:
Git will list both index.html and style.css as **untracked** files.

Screenshot A: Output of git status before adding files (showing untracked).

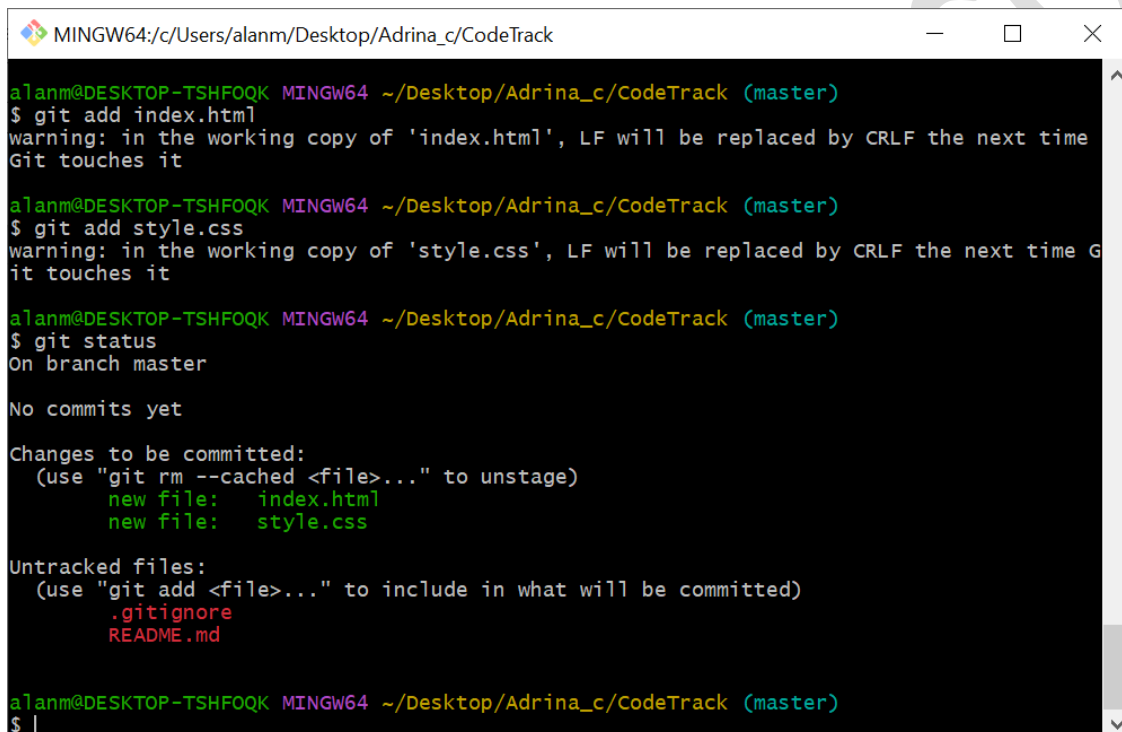2. **Stage the files for tracking:**

   1. git add .

or stage them one by one:

2. git add index.html

3. git add style.css

3. **Verify that the files are now staged:**

1. git status

Expected Output:

The files should now be **staged** and marked as **"Changes to be committed."**

Screenshot B: Output of git status after adding files (showing staged).


```
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time
Git touches it

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git add style.css
warning: in the working copy of 'style.css', LF will be replaced by CRLF the next time G
it touches it

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css

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

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ |
```

Task 3: Commit the Changes

1. **Commit the staged files with a meaningful message:**

1. git commit -m "Initial commit - Added index.html and style.css"
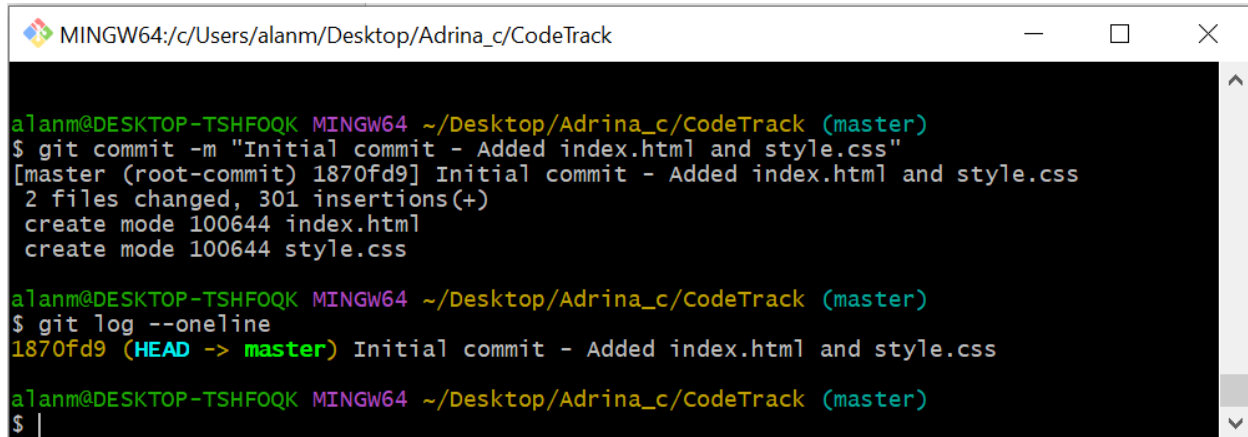
Expected Output:

2. [master (root-commit) 1a2b3c4] Initial commit - Added index.html and style.css

3. 2 files changed, 10 insertions(+), 0 deletions(-)

2. **Verify the commit history:**

1. git log --oneline

Expected Output:

2. 1a2b3c4 Initial commit - Added index.html and style.css

Screenshot C: Output of git log --oneline after the first commit.

Task 4: Modify a File and Commit Again

1. Next, open the index.html file in your favorite browser. (Tip: right-click the file →
   choose **Open With Browser**).

2. Read the instructions written inside index.html and follow them step by step to
   complete the assignment. It must be like below:

3. After finishing, move on to the next steps.
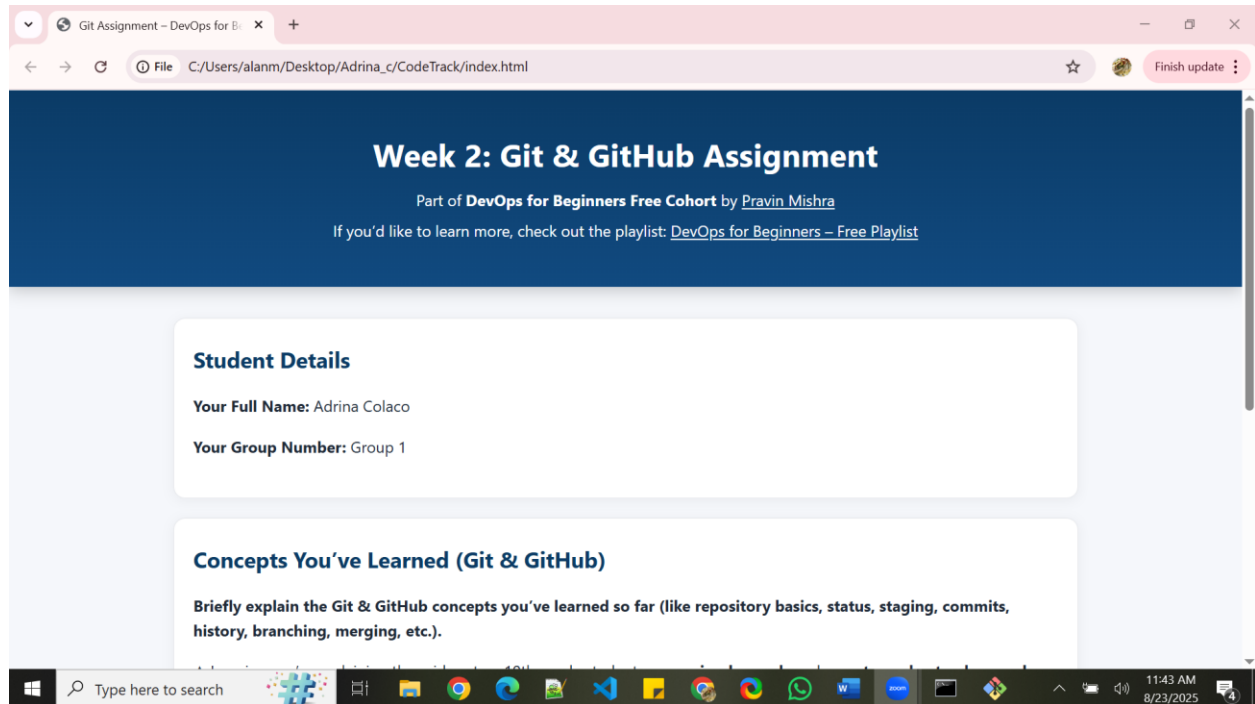
4. **Check the repository status:**

   1. git status

   Git will show index.html as **modified**.

5. **Stage the modified file:**

   1. git add index.html

6. **Commit the changes with a message:**

   1. git commit -m "Meaningful message"

7. **Verify commit history again:**

   1. git log --oneline

Expected Output:

   2. 3d4e5f6 Updated heading in index.html

   3. 1a2b3c4 Initial commit - Added index.html and style.css

Screenshot D: Output of git log --oneline after the second commit.

To see all details of the commits use below command

git log

Task 5: Deploy this application on EC2 Instance (as done in the Linux section)

1. Launch an EC2 Instance
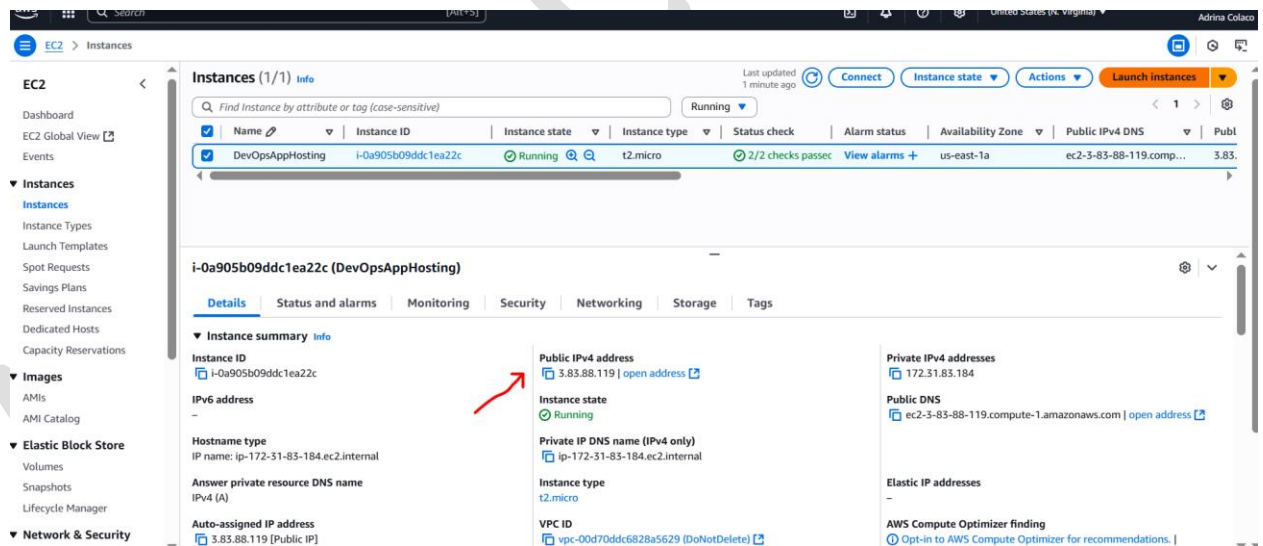
Make sure to add HTTP to the inbound rules of the respective security group while creating the instance.

2. Connect to Your EC2 Instance

- Open a terminal on your local machine.

- Use SSH to connect:

    1. ssh -i your-key.pem ec2-user@<EC2-Public-IP>

3. Update Packages *(as done in Linux section)*

4. Install Nginx *(as done in Linux section)*

5. Start Nginx Service

```
[root@ip-172-31-83-184 ec2-user]# sudo yum update
Amazon Linux 2023 Kernel Livepatch repository                                    171 kB/s |  19 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-83-184 ec2-user]# sudo yum install-y  nginx
No such command: install-y. Please use /usr/bin/yum --help
It could be a YUM plugin command, try: "yum install 'dnf-command(install-y)'"
[root@ip-172-31-83-184 ec2-user]# sudo yum install -y  nginx
Last metadata expiration check: 0:00:52 ago on Mon Aug 25 16:44:30 2025.
Dependencies resolved.
====================================================================================================================
 Package                    Architecture         Version                       Repository            Size
====================================================================================================================
Installing:
 nginx                      x86_64               1:1.28.0-1.amzn2023.0.2        amazonlinux            33 k
Installing dependencies:
 generic-logos-httpd        noarch               18.0.0-12.amzn2023.0.3        amazonlinux            19 k
 gperftools-libs            x86_64               2.9.1-1.amzn2023.0.3          amazonlinux           308 k
 libunwind                  x86_64               1.4.0-5.amzn2023.0.2          amazonlinux            66 k
 nginx-core                 x86_64               1:1.28.0-1.amzn2023.0.2       amazonlinux           686 k
 nginx-filesystem           noarch               1:1.28.0-1.amzn2023.0.2       amazonlinux           9.6 k
 nginx-mimetypes            noarch               2.1.49-3.amzn2023.0.3         amazonlinux            21 k

Transaction Summary
====================================================================================================================
Install  7 Packages

Total download size: 1.1 M
Installed size: 3.7 M
Downloading Packages:
(1/7): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm                     507 kB/s |  19 kB     00:00
(2/7): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm                                 1.6 MB/s |  66 kB     00:00
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm                           6.0 MB/s | 308 kB     00:00
(4/7): nginx-1.28.0-1.amzn2023.0.2.x86_64.rpm                                    1.3 MB/s |  33 kB     00:00
(5/7): nginx-core-1.28.0-1.amzn2023.0.2.x86_64.rpm                                24 MB/s | 686 kB     00:00
(6/7): nginx-filesystem-1.28.0-1.amzn2023.0.2.noarch.rpm                         475 kB/s | 9.6 kB     00:00
(7/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm                          1.1 MB/s |  21 kB     00:00
--------------------------------------------------------------------------------------------------------------------
Total                                                                            9.3 MB/s | 1.1 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                   1/1
  Running scriptlet: nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch                                   1/7
  Installing       : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch                                   1/7
  Installing       : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch                                      2/7
  Installing       : libunwind-1.4.0-5.amzn2023.0.2.x86_64                                             3/7
  Installing       : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64                                       4/7
  Installing       : nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64                                         5/7
  Installing       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                                 6/7
  Installing       : nginx-1:1.28.0-1.amzn2023.0.2.x86_64                                              7/7
  Running scriptlet: nginx-1:1.28.0-1.amzn2023.0.2.x86_64                                              7/7
  Verifying        : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                                 1/7
  Verifying        : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64                                       2/7
  Verifying        : libunwind-1.4.0-5.amzn2023.0.2.x86_64                                             3/7
  Verifying        : nginx-1:1.28.0-1.amzn2023.0.2.x86_64                                              4/7
  Verifying        : nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64                                         5/7
  Verifying        : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch                                   6/7
  Verifying        : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch                                      7/7

Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch            gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  libunwind-1.4.0-5.amzn2023.0.2.x86_64                        nginx-1:1.28.0-1.amzn2023.0.2.x86_64
  nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64                    nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch
  nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
[root@ip-172-31-83-184 ec2-user]# sudo systemctl start nginx
[root@ip-172-31-83-184 ec2-user]# sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[root@ip-172-31-83-184 ec2-user]# |
```

6. Deploy Your Application *(same as Linux section)*

- Copy your project files (index.html, style.css, etc.) into Nginx's web directory:

  1. Use FileZilla to move the project located on the local machine to EC2(MobaXterm can also be used)
  2. Go to Edit → Settings → Under Connection → SFTP → add the PEM key that downloaded while creating your EC2 instance.
  3. Quick Connect to the server using sftp and move the files from local machine to EC2 server.
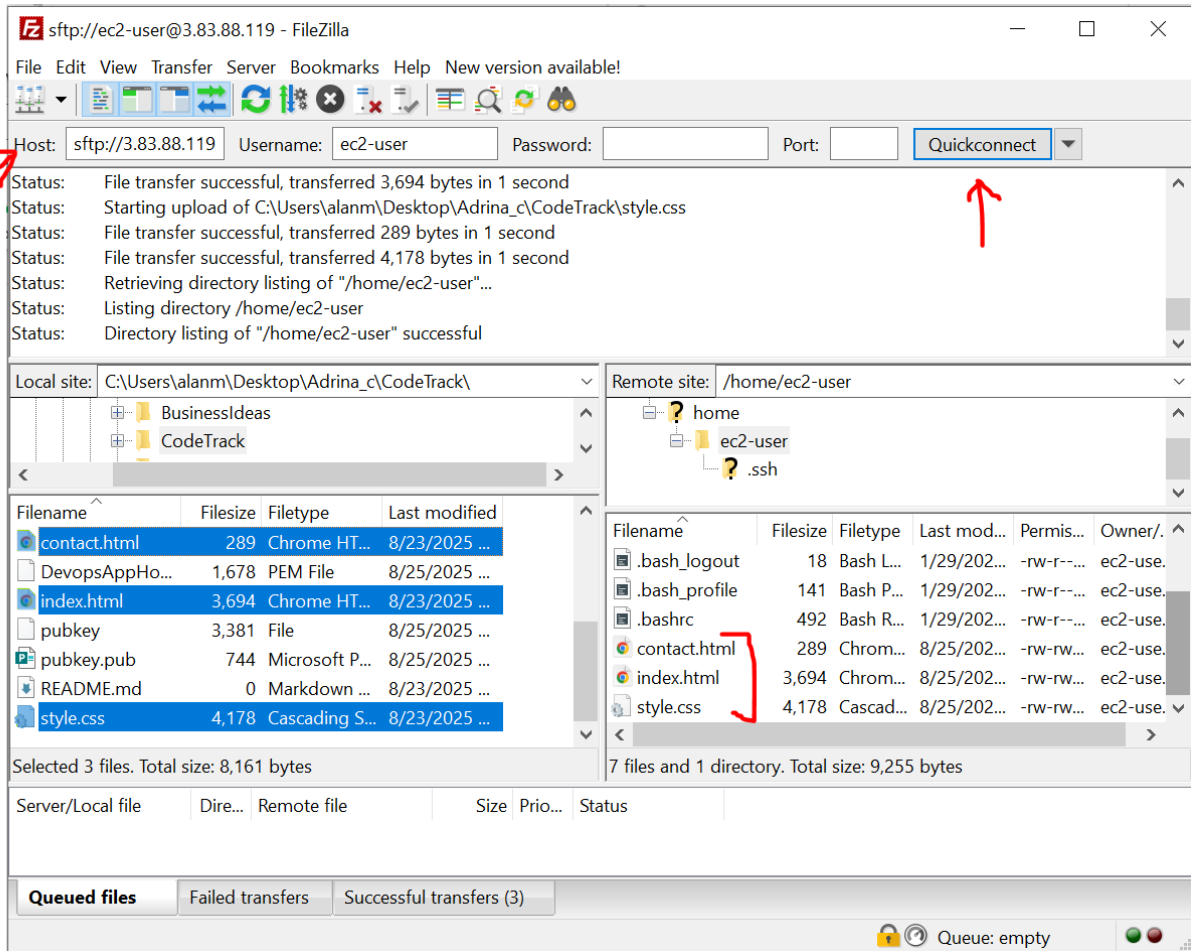
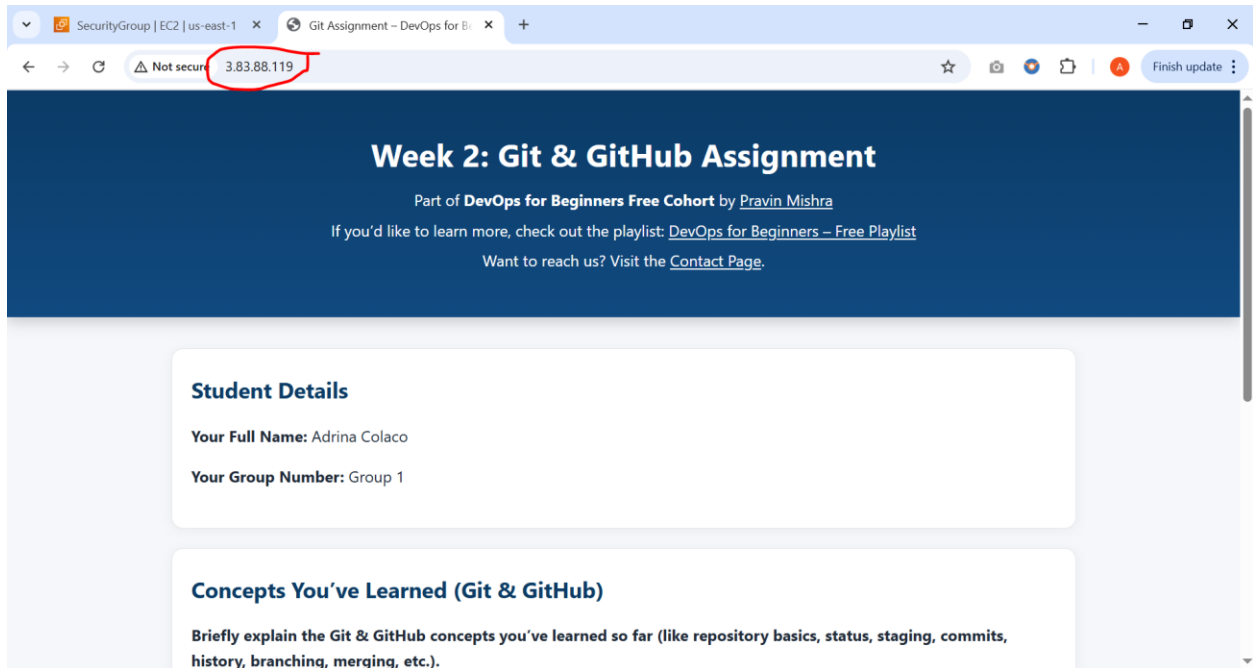4. Copy the required web files to the NGINX web directory.
   1. sudo cp /home/ec2-user/index.html /usr/share/nginx/html/index.html
   2. sudo cp /home/ec2-user/contact.html /usr/share/nginx/html/contact.html
   3. sudo cp /home/ec2-user/style.css /usr/share/nginx/html/style.css

7. Access the Application
   - Open a browser and go to:
     1. http://<EC2-Public-IP>

You should see your application running successfully.

Text File (git_tracking_summary.txt):

- Listing all Git commands used so far with a short explanation for each.

Git_Tracking_Summa
ry.txt

Assignment 7- Branching Workflow — Add & Verify a Contact Page in Git

Task 0 — Start from your existing repo

Navigate your **existing CodeTrack** project from the last assignment:

1. cd path/to/CodeTrack

2. git status

3. git branch

Ensure you're on main (or master).

Task 1 — Create and switch to a feature branch

1. git checkout -b feature/contact-page

2. git branch

Expected: * feature/contact-page.

Screenshot A: git branch output right after creating feature/contact-page (shows * feature/contact-page).

```
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git checkout -b feature/contact-page
Switched to a new branch 'feature/contact-page'

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ git branch
* feature/contact-page
  master

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ |
```

Task 2 — Add contact.html (in the branch)

Create the file and add content:

1. # macOS/Linux

2. touch contact.html

3. # Windows PowerShell alternative:

4. # ni contact.html

contact.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Contact - CodeTrack</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Contact Us</h1>
```

<p>Email: mail@pravinmishra.in</p>
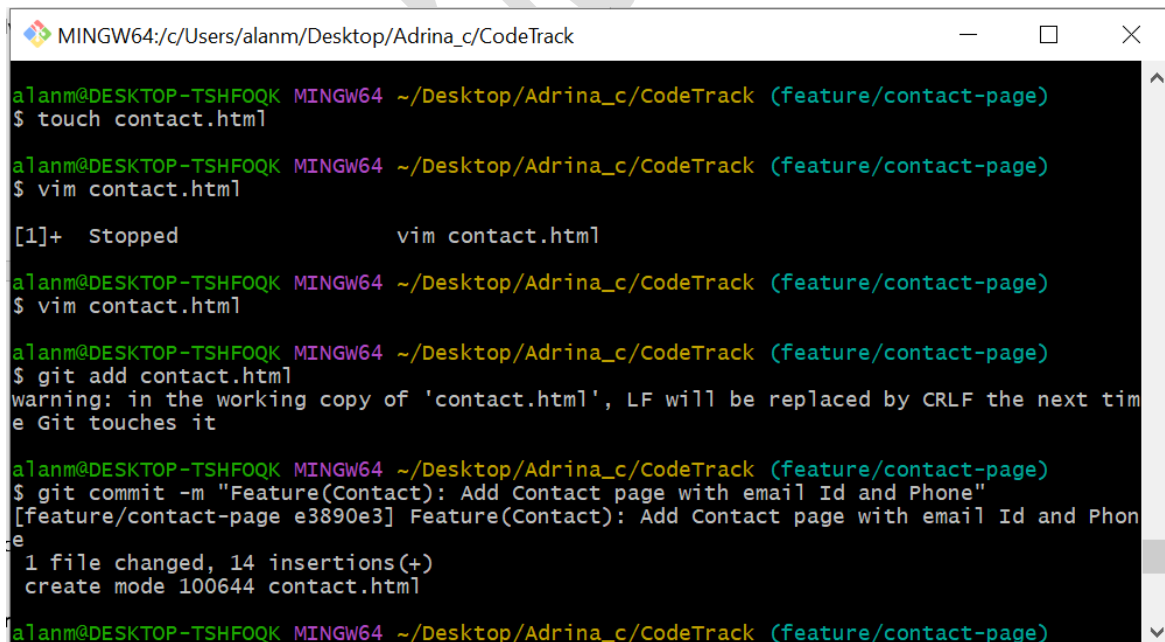
<p>Website: hhttps://thecloudadvisory.com/</p>

</body>

</html>

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Contact - CodeTrack</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Contact Us</h1>
  <p>Email: mail@pravinmishra.in</p>
  <p>Website: hhttps://thecloudadvisory.com/</p>
</body>
</html>
```

`contact.html [dos] (13:30 23/08/2025)`   `14,0-1 All`
`"contact.html" [dos] 14L, 289B`

Stage & commit:

1. git add contact.html
2. git commit -m "feat(contact): add contact page with email and phone"

```
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ touch contact.html

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ vim contact.html

[1]+  Stopped                 vim contact.html

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ vim contact.html

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ git add contact.html
warning: in the working copy of 'contact.html', LF will be replaced by CRLF the next tim
e Git touches it

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ git commit -m "Feature(Contact): Add Contact page with email Id and Phone"
[feature/contact-page e3890e3] Feature(Contact): Add Contact page with email Id and Phon
e
 1 file changed, 14 insertions(+)
 create mode 100644 contact.html

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
```
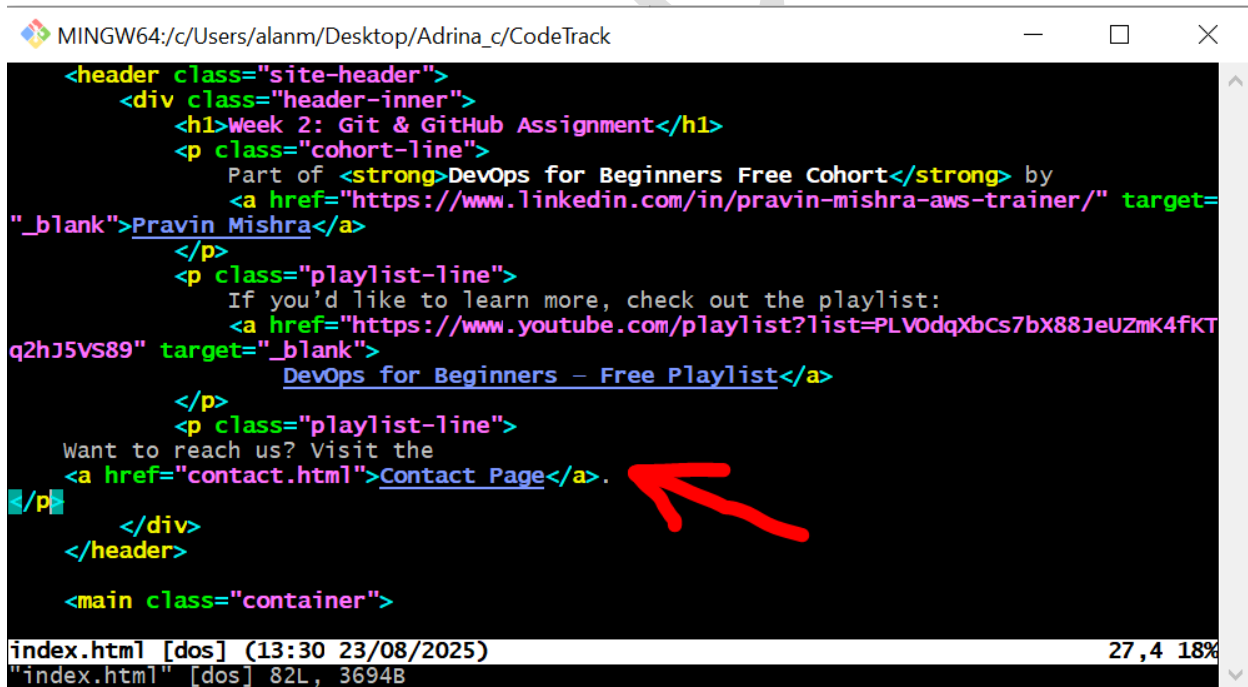
Task 3 — Add a link to the contact page in index.html (still on the branch)

Insert this **new paragraph** directly **below** the existing playlist paragraph (the one with class playlist-line):

```
<p class="playlist-line">
   If you'd like to learn more, check out the playlist:
   <a
href="https://www.youtube.com/playlist?list=PLVOdqXbCs7bX88JeUZmK4fKTq2h
J5VS89" target="_blank">
      DevOps for Beginners – Free Playlist
   </a>
</p>

<!-- ADD THIS NEW PARAGRAPH RIGHT BELOW THE ONE ABOVE -->
<p class="playlist-line">
   Want to reach us? Visit the
   <a href="contact.html">Contact Page</a>.
</p>
```


Save, then:

1. git add index.html

2. git commit -m "feat(nav): add Contact Page link to index.html"

```
  MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack                    —   □   ✕

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ vim index.html

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time
Git touches it

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ git commit -m "feature(Nav): add Contact Page link to index.html"
[feature/contact-page ba67ae5] feature(Nav): add Contact Page link to index.html
 1 file changed, 4 insertions(+)
```

Task 4 — Verify isolation (switch back to main)

1. git checkout main

2. ls

Screenshot B. git branch output after switching back to main.

```
  MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack                    —   □   ✕

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
t$ git checkout master
Switched to branch 'master'

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ ls
README.md   index.html   style.css
```

- contact.html should **not** be in main yet.

- Open index.html in your browser → the **Contact Page** link should **not** exist on main yet.
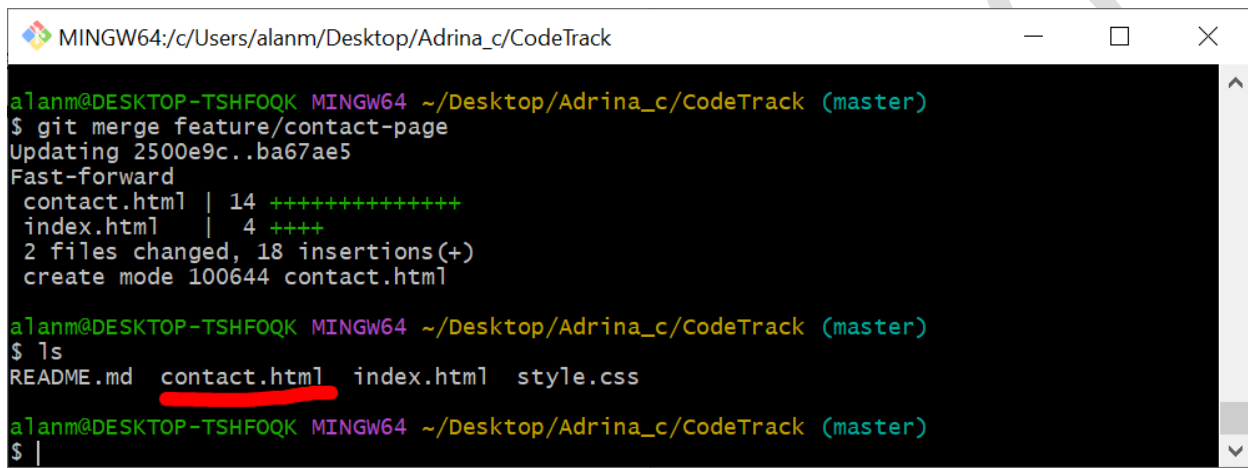
(This proves your changes live only on the branch.)

```
  MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack                    —   □   ✕

                <a href="https://www.youtube.com/playlist?list=PLVOdqXbCs7bX88JeUZmK4fKT
q2hJ5VS89" target="_blank">
                    DevOps for Beginners — Free Playlist</a>
            </p>
        </div>
    </header>

  | <main class="container">

        <!-- Student Details -->
        <section class="card student-info" aria-labelledby="student-details">
            <h2 id="student-details">Student Details</h2>
            <p><b>Your Full Name:</b> <span class="placeholder">Adrina Colaco</span></p>
            <p><b>Your Group Number:</b> <span class="placeholder">Group 1</span></p>
        </section>
index.html [dos] (13:21 23/08/2025)                                    27,4 31%
"index.html" [dos] 78L, 3577B
```

Task 5 — Merge the feature branch into main

1. git merge feature/contact-page

Now verify:

- ls → contact.html is present.

- Open index.html in your browser → you should see the **Contact Page** link.

- Click the link → it should open **contact.html**. ✅

Task 6 — Inspect history (nice graph)

1. git log --oneline --graph --decorate --all
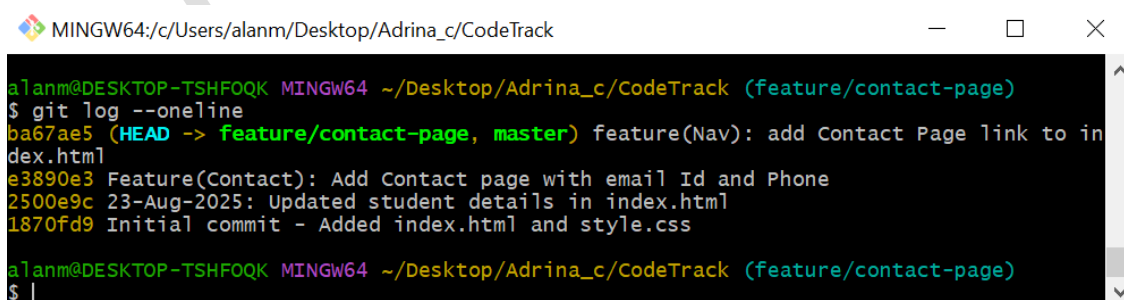
(Optional cleanup)

1. git branch -d feature/contact-page

Screenshot C. git log --oneline while on feature/contact-page showing the commits:

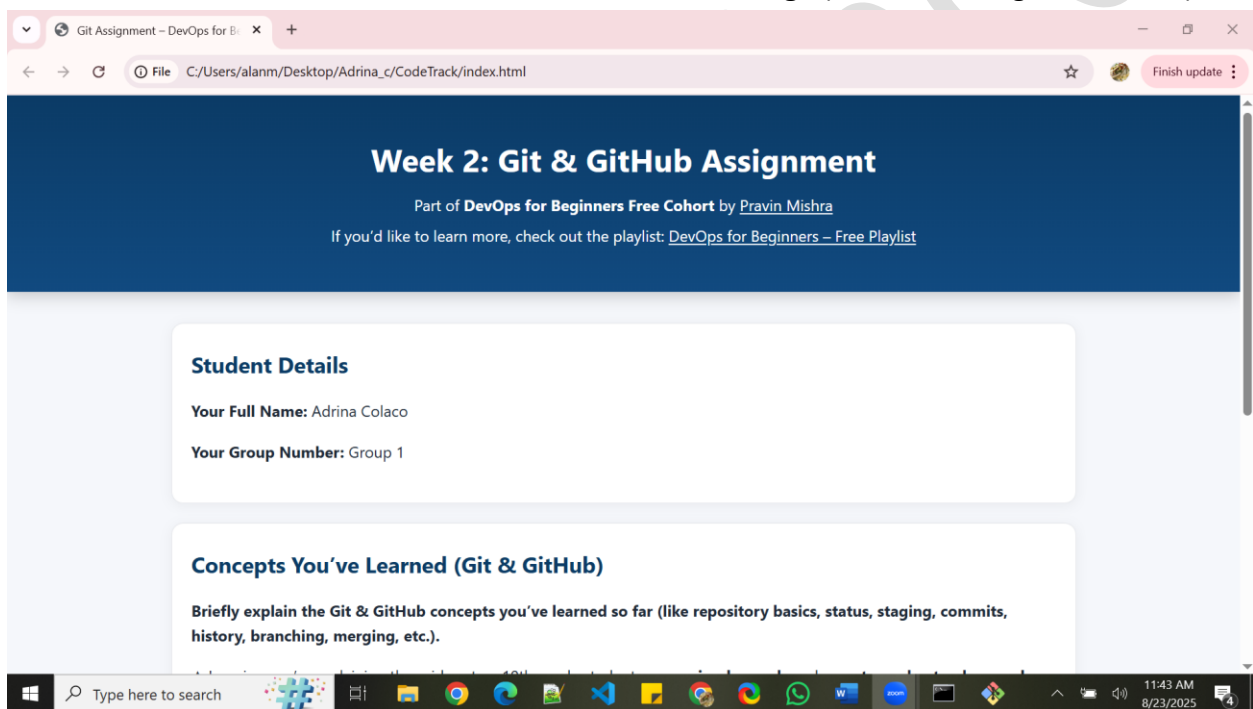feat(contact): add contact page...
feat(nav): add Contact Page link...

Screenshot D. git log --oneline --graph --decorate --all after merging into main.

MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack

```
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git log --oneline --graph --decorate --all
* ba67ae5 (HEAD -> master, feature/contact-page) feature(Nav): add Contact Page link to
index.html
* e3890e3 Feature(Contact): Add Contact page with email Id and Phone
* 2500e9c 23-Aug-2025: Updated student details in index.html
* 1870fd9 Initial commit - Added index.html and style.css

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ |
```

Screenshot E: Browser screenshot of index.html **before merge** (no Contact Page link visible).

Git Assignment – DevOps for B...

File C:/Users/alanm/Desktop/Adrina_c/CodeTrack/index.html

Finish update

Week 2: Git & GitHub Assignment

Part of **DevOps for Beginners Free Cohort** by Pravin Mishra

If you'd like to learn more, check out the playlist: DevOps for Beginners – Free Playlist

Student Details

Your Full Name: Adrina Colaco

Your Group Number: Group 1

Concepts You've Learned (Git & GitHub)

Briefly explain the Git & GitHub concepts you've learned so far (like repository basics, status, staging, commits, history, branching, merging, etc.).

Type here to search

11:43 AM
8/23/2025

Screenshot F: Browser screenshot of index.html **after merge** showing the new **Contact Page** link.
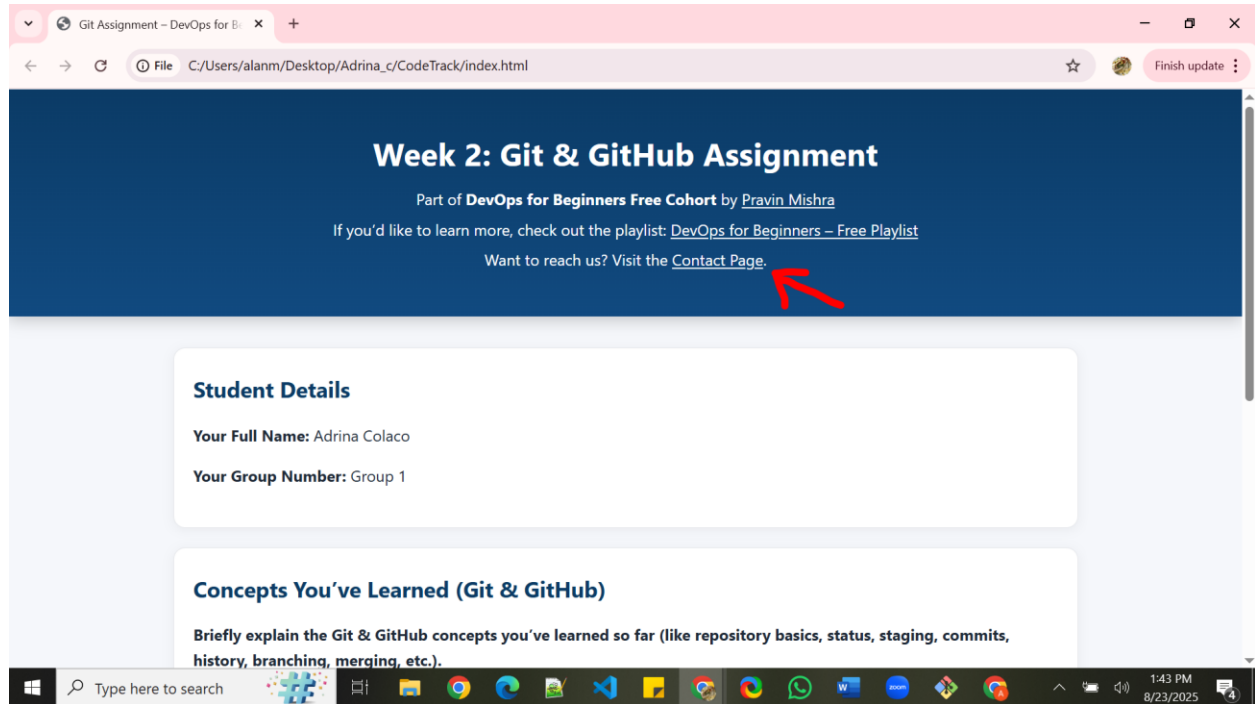
Screenshot G: Browser screenshot of the loaded **contact.html** (after clicking the link).
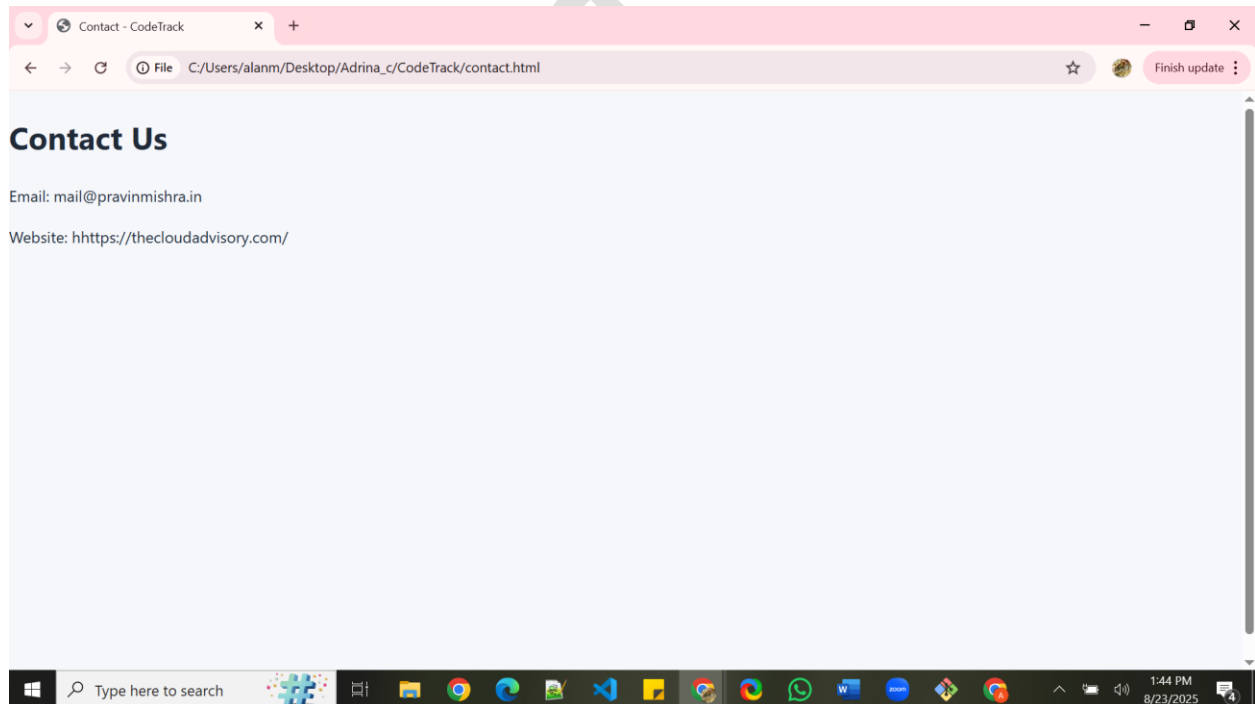
The changed code from the Branch was not yet present in the Main(master) branch before Git Merge.

Once Git merge command was run, the code linking to Contact.html file moved to the index.html file present in main(master) branch as well.

Assignment 8 - Setting Up GitHub for CodeTrack

Task 1 — Create a GitHub Account

1. Go to GitHub and click **Sign Up**.

2. Enter your **email, password, and username**.

3. Complete verification and click **Create Account**.

4. Once logged in, navigate to your **GitHub Dashboard**.

Expected Outcome: You now have a GitHub account and access to the dashboard.
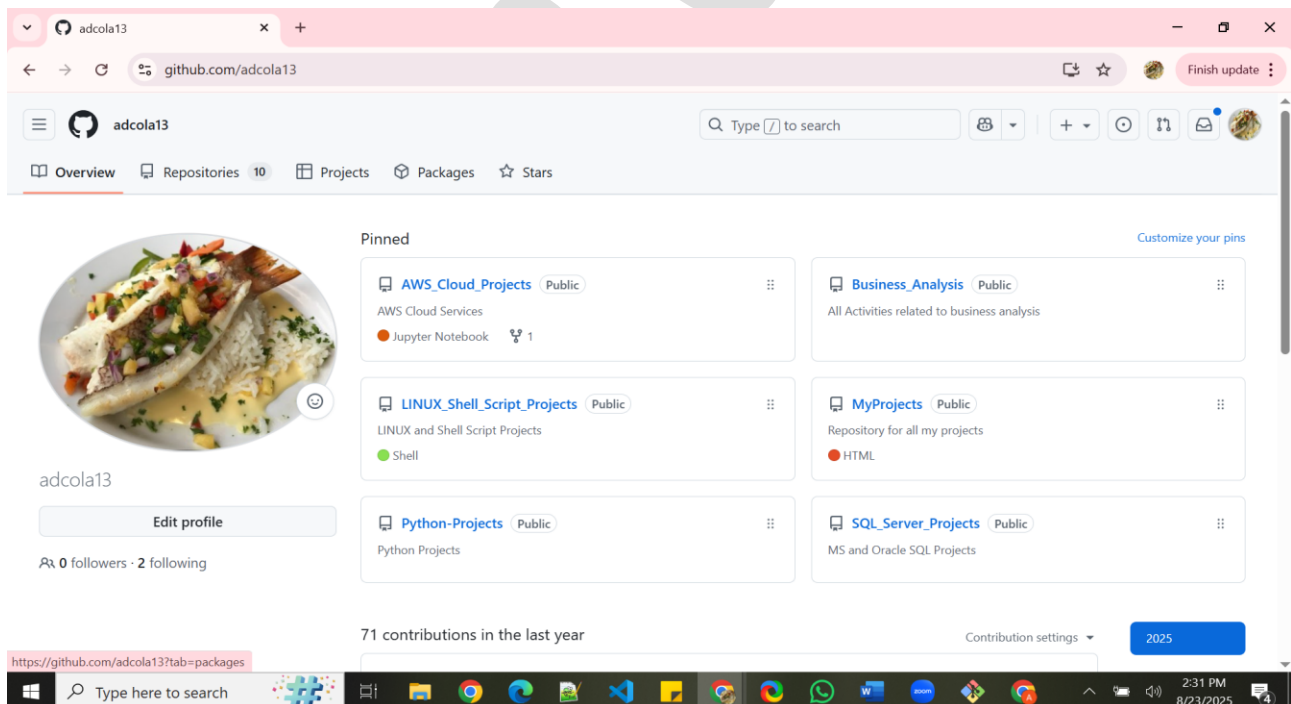
Task 2 — Explore GitHub Features

1. From the top menu, click **Explore**.

2. Browse **Trending Repositories** to see what's popular.

3. Use the search bar to find an open-source project (e.g., type theepicbook).

4. Click the ⭐ **Star** button on a repository that interests you.

5. Click **Fork** on any public repository to create your own copy.

Expected Outcome: You have explored GitHub, starred at least one project, and forked one repository.

Screenshot A: The **repository you starred.**

Screenshot B: The **repository you forked**.
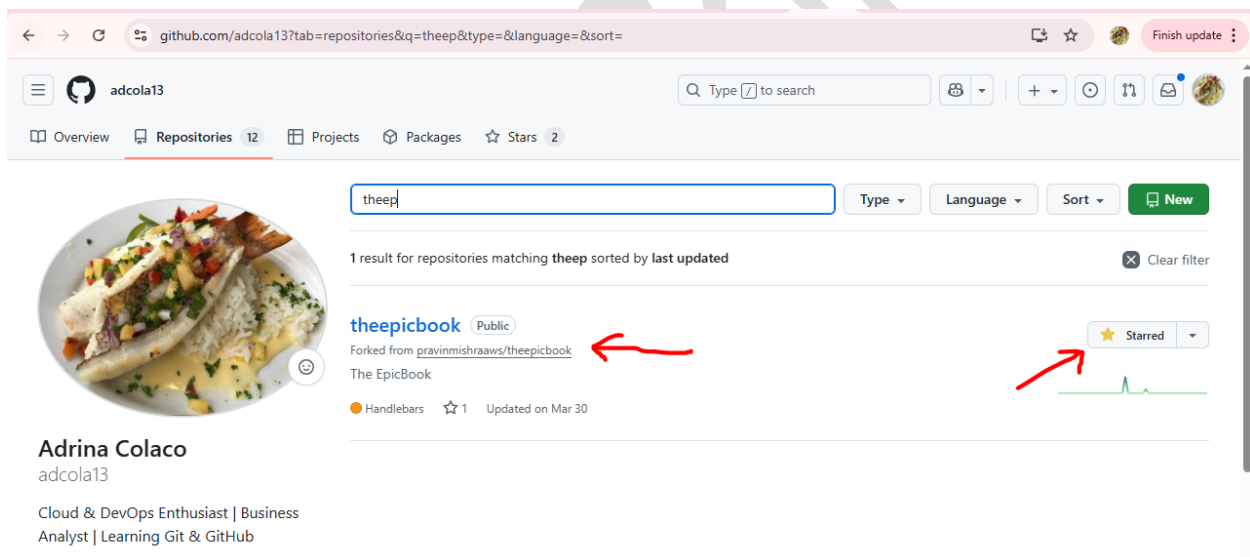
Step 3 — Update Your GitHub Profile

1. Click your profile picture (top-right) → **Your Profile**.

2. Click **Edit Profile** and:

   • Add a short **bio** (e.g., "Cloud & DevOps Enthusiast | Learning Git & GitHub").

   • Optionally add **location**, **company/school**, and **social links**.

   • Upload a **profile picture** (optional but recommended).

3. Save changes.

Expected Outcome: Your GitHub profile looks personalized and professional.

Screenshot C: Your updated **GitHub profile page** showing your new bio.

A polished GitHub profile serves as a **live portfolio**, showcasing your real code, collaboration history, and growth trajectory, making it a powerful tool for credibility, visibility, and hiring potential.

Key Takeaways:

- **Real code → resume claims**: Viewers can assess your actual work & problem-solving style.
- **Reflects growth & commitment**: A history of contributions signals continuous learning.
- **Trusted evidence**: For those with limited formal experience, your GitHub can be the only proof you can code.

Assignment 9- Collaborating on Mini-Finance with GitHub

Task 0 — Access Existing Mini-Finance Code
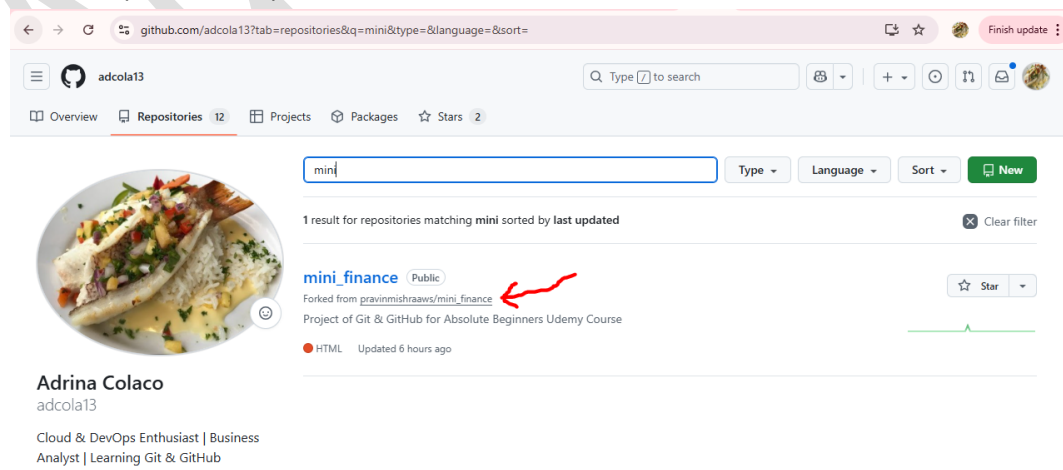
The upstream repository exists at:

1. https://github.com/pravinmishraaws/mini_finance

We'll fork this repository to your GitHub account, clone, and work with it.

Task 1 — Fork & Authenticate

1. Login to GitHub and **Fork** the mini_finance repository into your account.

   Screenshot A: The GitHub page showing your **forked mini_finance repo** under your account (URL visible).
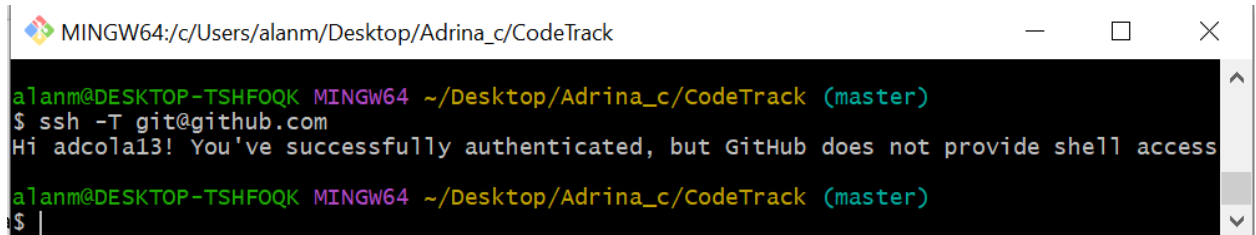
2. In your terminal, configure authentication if not already set:

- **SSH** (recommended):

    1. **Generate an SSH Key (if not already created):**

        1. ssh-keygen -t rsa -b 4096 -C "your-email@example.com"

        2. Press **Enter** to save the key in the default location (~/.ssh/id_rsa).

    2. **Add the SSH Key to GitHub:**

        1. Copy the SSH key:

        2. cat ~/.ssh/id_rsa.pub

        3. Go to **GitHub → Settings → SSH and GPG keys →** Click **New SSH Key**

        4. Paste the key and save

    3. **Test the SSH Connection:**

1. ssh -T git@github.com

2. If successful, you should see a message like:

3. Hi yourusername! You've successfully authenticated, but GitHub does not provide shell access.

4. **Use SSH for Git Commands:**
   When cloning a repository, use the SSH URL instead of HTTPS:

5. git clone git@github.com:yourusername/repository.git

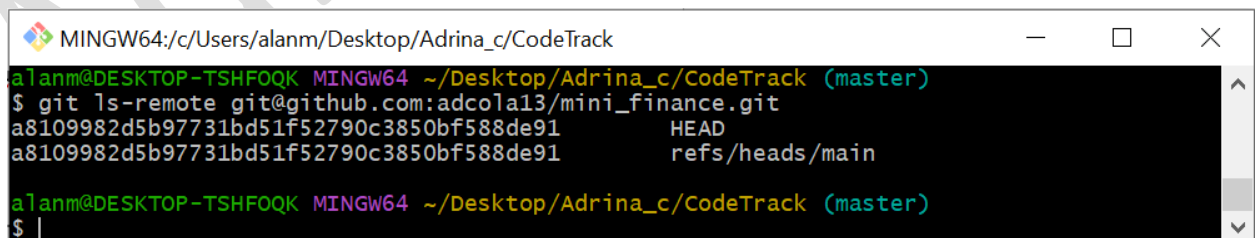6. git config --global url."git@github.com:".insteadOf "https://github.com/"

- **OR HTTPS**:

1. git config --global credential.helper cache

Test with:

2. git ls-remote git@github.com:yourusername/mini_finance.git
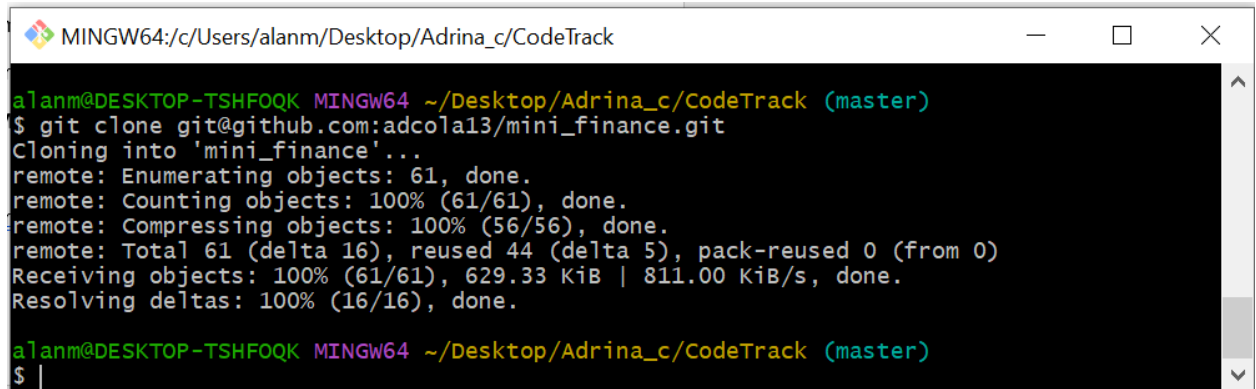
3. **Expected outcome:** You have forked the repo and your terminal is authenticated for Git ops.

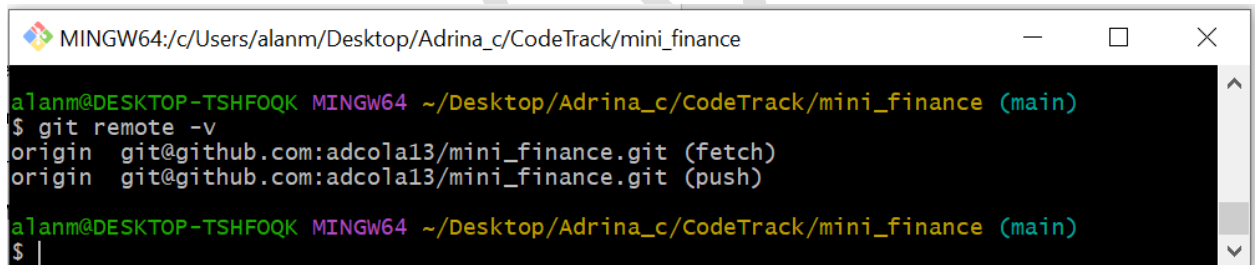**Task 2 — Clone Your Fork Locally**

1. git clone git@github.com:yourusername/mini_finance.git

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack                    —    □    ✕

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git clone git@github.com:adcola13/mini_finance.git
Cloning into 'mini_finance'...
remote: Enumerating objects: 61, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 61 (delta 16), reused 44 (delta 5), pack-reused 0 (from 0)
Receiving objects: 100% (61/61), 629.33 KiB | 811.00 KiB/s, done.
Resolving deltas: 100% (16/16), done.

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$
```

2. cd mini_finance
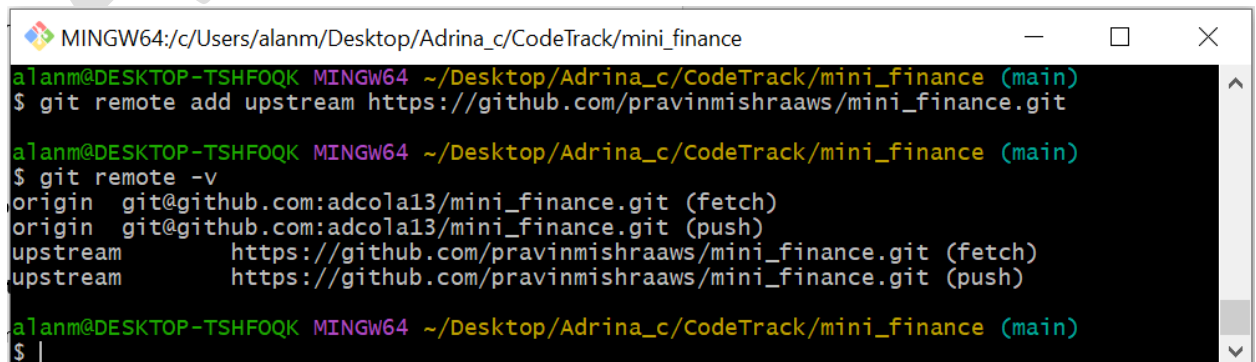
3. git remote -v

• origin should point to your fork.

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance       —    □    ✕

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ git remote -v
origin  git@github.com:adcola13/mini_finance.git (fetch)
origin  git@github.com:adcola13/mini_finance.git (push)

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$
```
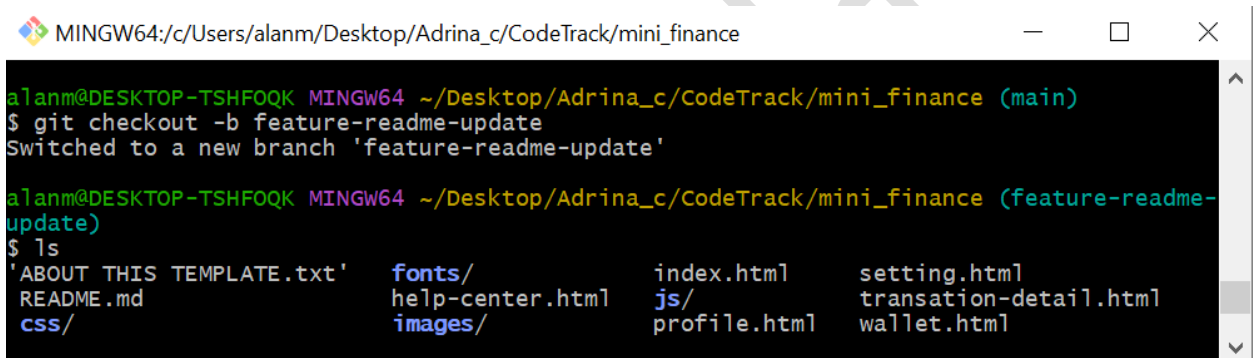
1. Add an upstream remote to the original repo:

    1. git remote add upstream https://github.com/pravinmishraaws/mini_finance.git

Expected outcome: Local clone with origin (your fork) and upstream properly set.

Screenshot B: Terminal output of git remote -v after cloning, showing both origin and upstream.

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance       —    □    ✕
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ git remote add upstream https://github.com/pravinmishraaws/mini_finance.git

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ git remote -v
origin  git@github.com:adcola13/mini_finance.git (fetch)
origin  git@github.com:adcola13/mini_finance.git (push)
upstream        https://github.com/pravinmishraaws/mini_finance.git (fetch)
upstream        https://github.com/pravinmishraaws/mini_finance.git (push)

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$
```

The **upstream remote** serves as an anchor to the original repository, enabling you to **fetch** the latest changes, keep your fork in sync, and make your contributions clean and up-to-date. It's essential for effective collaboration, especially in open-source or team-based workflows.

Task 3 — Create a Feature Branch & Make a Change

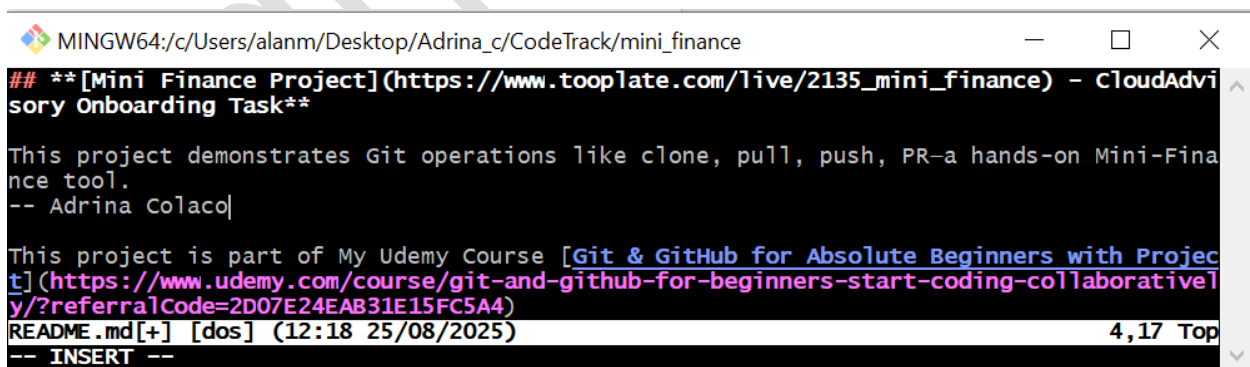2. Create a new branch:

    1. git checkout -b feature-readme-update

MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance — □ ✕

```
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ git checkout -b feature-readme-update
Switched to a new branch 'feature-readme-update'

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-update)
$ ls
'ABOUT THIS TEMPLATE.txt'   fonts/            index.html      setting.html
 README.md                  help-center.html  js/             transation-detail.html
 css/                       images/           profile.html    wallet.html
```

3. Open README.md and add a new section:

You may write: "This project demonstrates Git operations like clone, pull, push, PR—a hands-on Mini-Finance tool."

MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance — □ ✕

```
## **[Mini Finance Project](https://www.tooplate.com/live/2135_mini_finance) - CloudAdvisory Onboarding Task**

This project demonstrates Git operations like clone, pull, push, PR—a hands-on Mini-Finance tool.
-- Adrina Colaco

This project is part of My Udemy Course [Git & GitHub for Absolute Beginners with Project](https://www.udemy.com/course/git-and-github-for-beginners-start-coding-collaboratively/?referralCode=2D07E24EAB31E15FC5A4)
README.md[+] [dos] (12:18 25/08/2025)                                    4,17 Top
-- INSERT --
```

4. Save, then stage and commit:

    1. git add README.md

    2. git commit -m "docs: update README with assignment note"

Screenshot C: Terminal showing your commit on feature-readme-update. (e.g., git log --oneline -n 3)
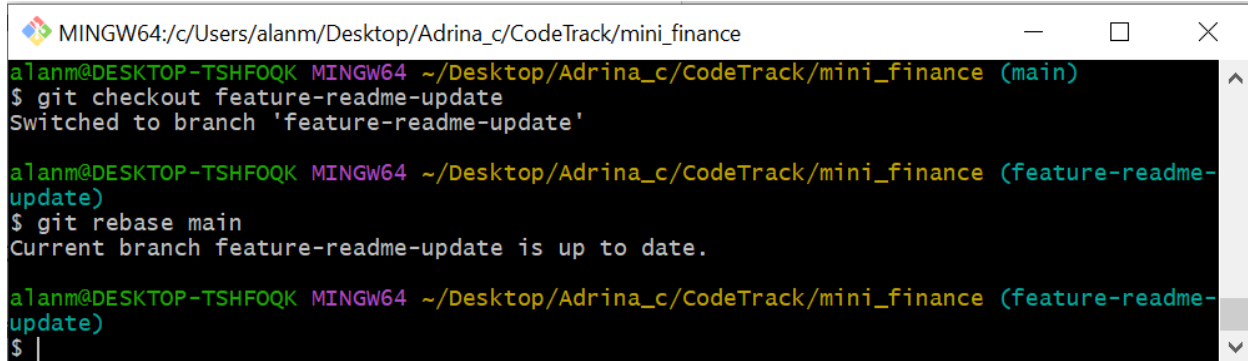


```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance        —    □    ×

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-
update)
$ git add README.md

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-
update)
$ git commit -m "docs: updated README file with an assignment note"
[feature-readme-update f603328] docs: updated README file with an assignment note
 1 file changed, 3 insertions(+)

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-
update)
$ |
```
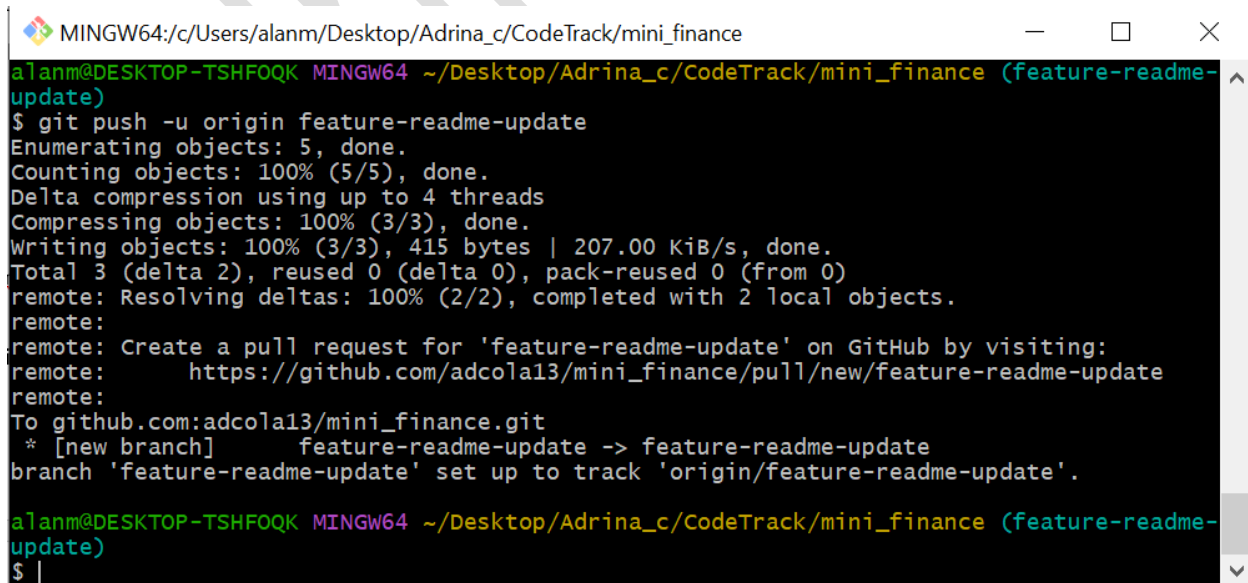
Task 4 — Pull From Upstream & Push to Origin

1. Sync changes from upstream's main:

    1. git fetch upstream


```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance        —    □    ×

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-
update)
$ git fetch upstream
From https://github.com/pravinmishraaws/mini_finance
 * [new branch]      main       -> upstream/main

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-
update)
$ |
```

    2. git checkout main

    3. git merge upstream/main


```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance        —    □    ×

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-
update)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ git merge upstream/main
Already up to date.

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ |
```

2. Switch back to your feature branch:

   1. git checkout feature-readme-update

   2. git rebase main   # optional but recommended

Rebasing or merging from upstream/main (or upstream/master) before pushing is primarily about keeping your work in sync with the latest changes in the main project and ensuring your commits integrate cleanly.

3. Push your branch to your fork:

   1. git push -u origin feature-readme-update

Screenshot D: Confirmation of git push -u origin feature-readme-update.

Expected outcome: Your feature branch is available on GitHub under your fork.
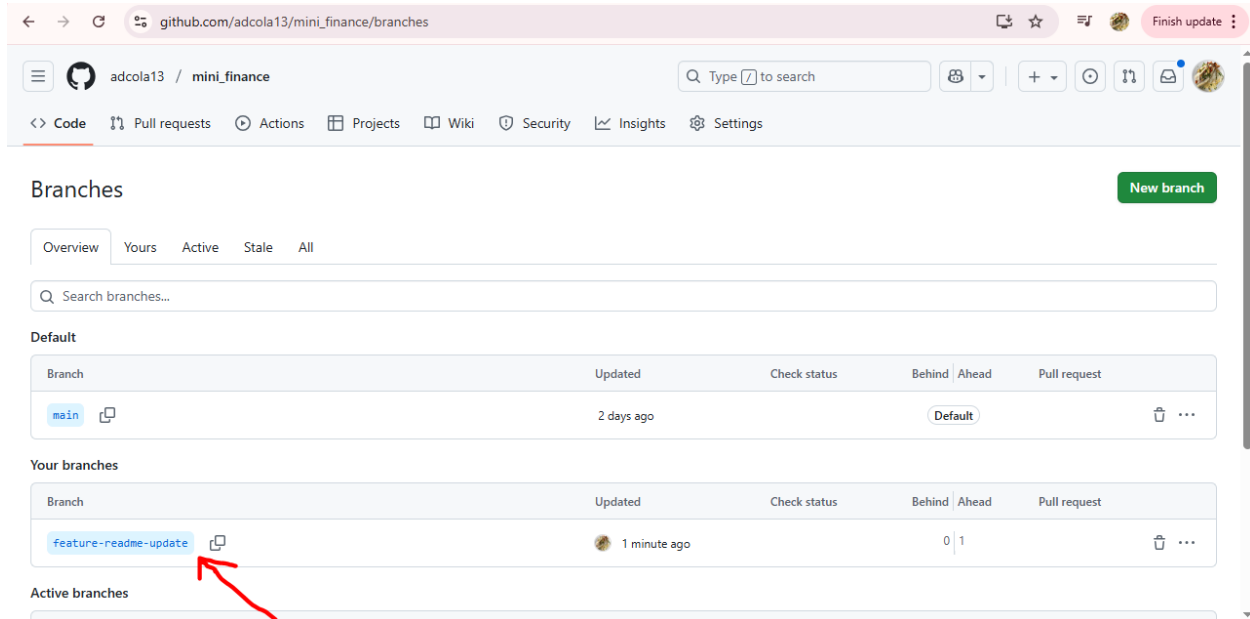
Task 5 — Create a Pull Request

1. Go to your fork on GitHub.

2. Click on 3 dots and then on **New Pull Request**.
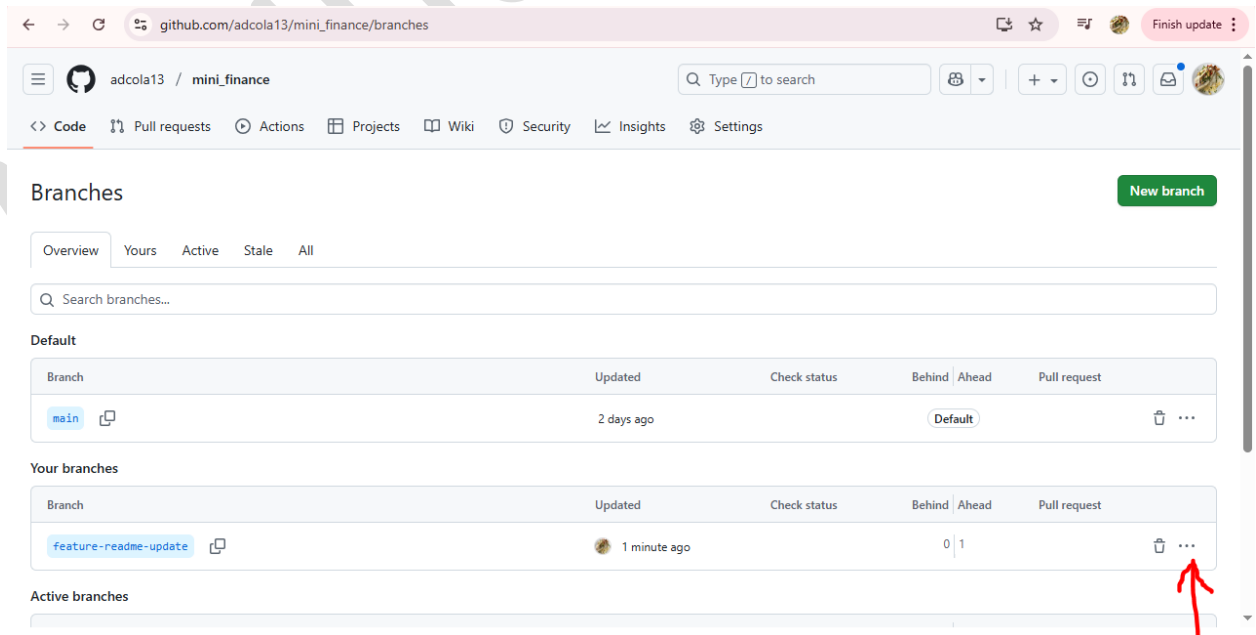
3. Make sure it's targeting pravinmishraaws/mini_finance:main from your feature-readme-update branch.

   1. Title: "docs: update README with assignment note"

   2. In the body, add a short description:

   "This PR adds a new section to the README explaining the project's purpose in the context of this GitHub assignment."

4. Submit the **Pull Request**.

Screenshot E: The GitHub UI showing **open Pull Request** (title and description visible).
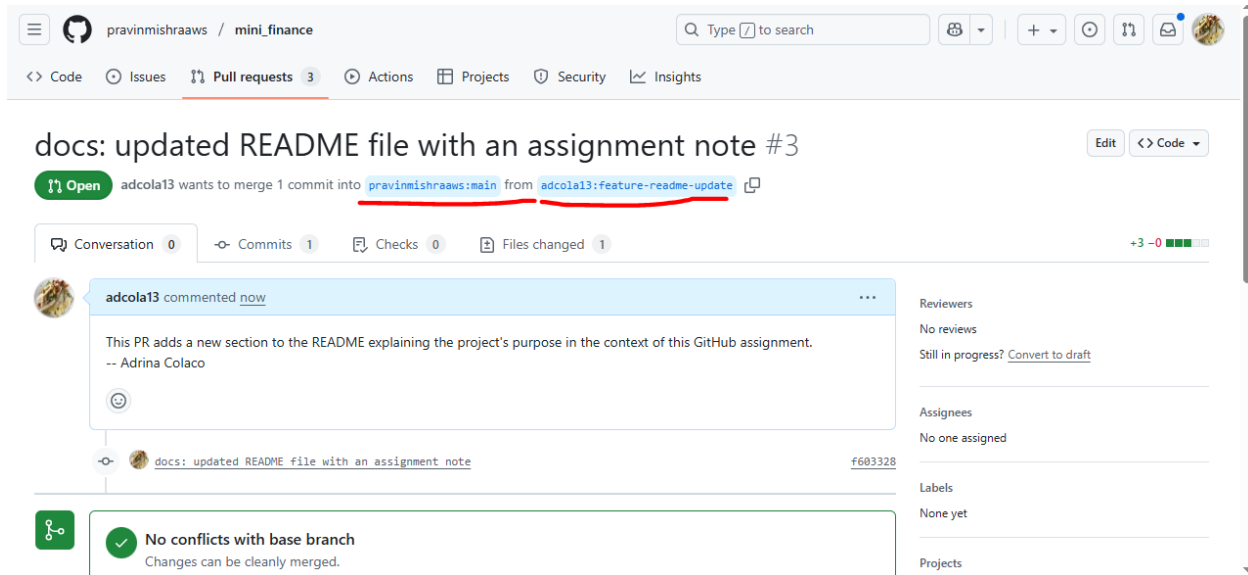
==Why is creating a Pull Request an important step in team collaboration?==

Pull Requests are essential in modern Git workflows because they:

- Ensure high code quality via structured review
- Enable clear communication and knowledge sharing
- Provide transparent documentation
- Support automated checks and mitigate risks
- Streamline collaboration at scale—especially in distributed environments

The End

P.S. This document is part of the **FREE DevOps for Beginners Cohort** run by [Pravin Mishra](#). You can start your DevOps journey for free from his [YouTube Playlist](#).