

Hands on Journey from Git to GitHub

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](#).

Overview

In this exercise, we will be completing multiple assignments to accomplish the following setup.

- CodeTrack— Initial Git Setup (Local Only)
- Tracking and Staging Changes in a CodeTrack Project in Git
- Branching Workflow — Add & Verify a Contact Page in Git
- Setting Up GitHub for CodeTrack
- Collaborating on Mini-Finance with GitHub

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:

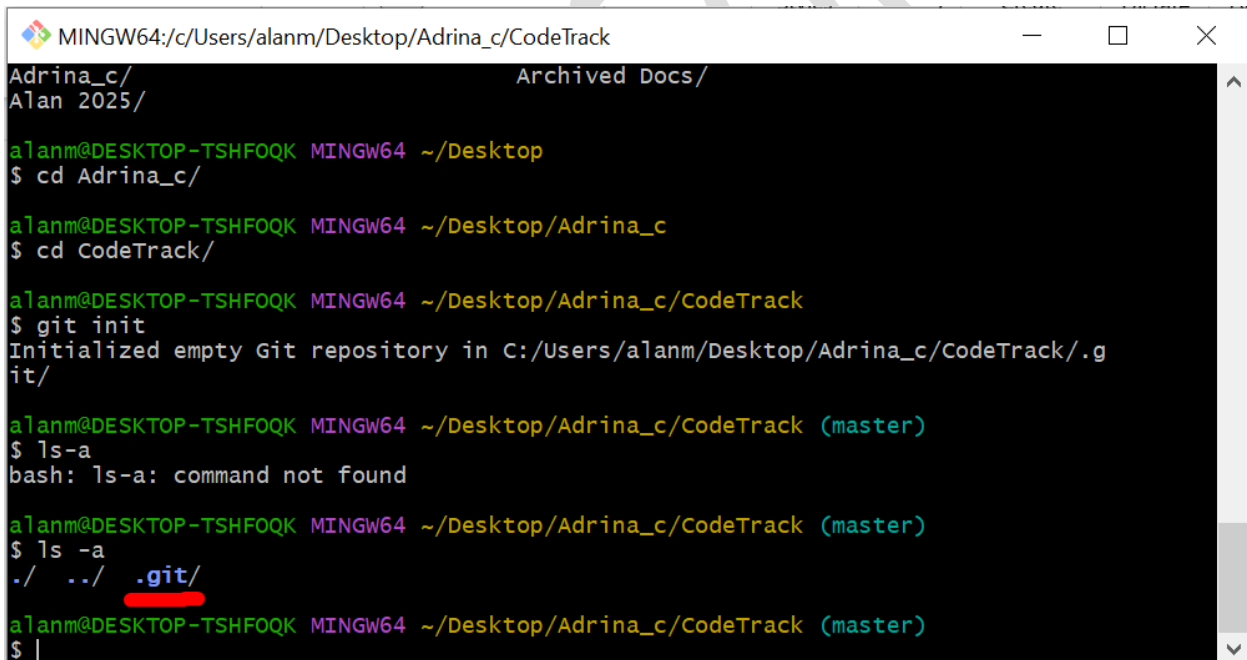
Initialized empty Git repository in .../CodeTrack/.git/

(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/
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/.git/

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"`

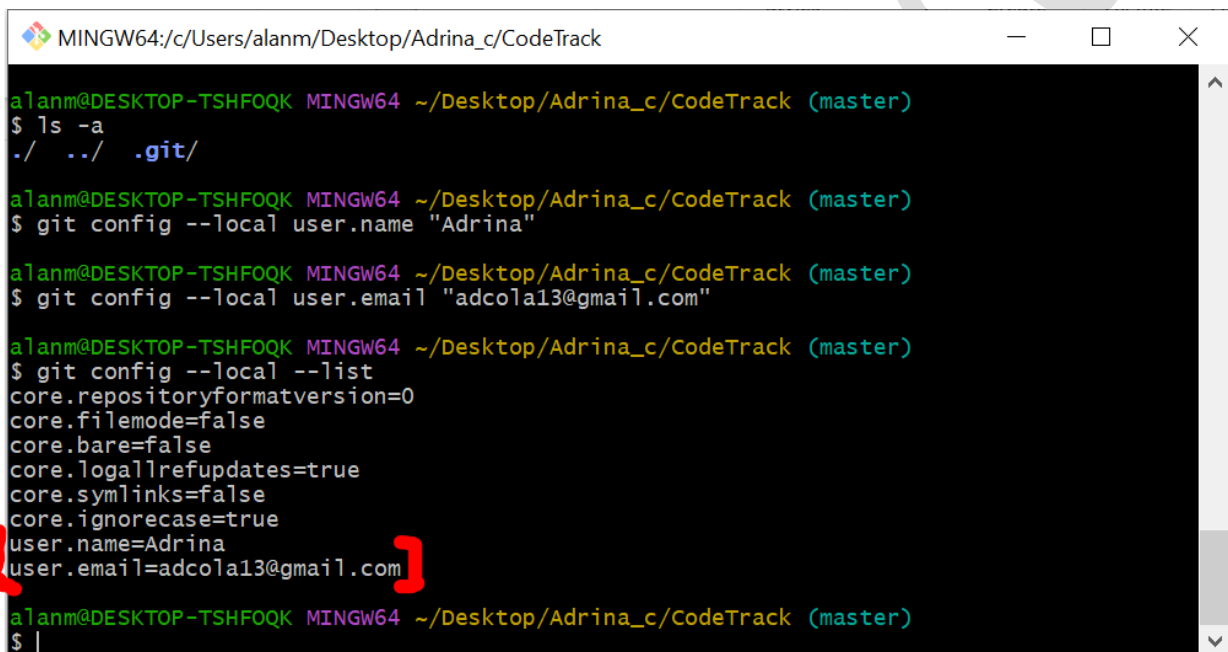
3. `git config --local --list`

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`)



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

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

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git config --local user.name "Adrina"

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git config --local user.email "adcola13@gmail.com"

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git config --local --list
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
user.name=Adrina
user.email=adcola13@gmail.com
```

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`)



```
MINGW64:/c:/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git config --global user.name "Adrina"

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git config --global user.email "adcola13@gmail.com"

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git config --global --list
user.name=Adrina
user.email=adcola13@gmail.com

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

When would you prefer to use **local configuration** instead of **global configuration**?

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`.)

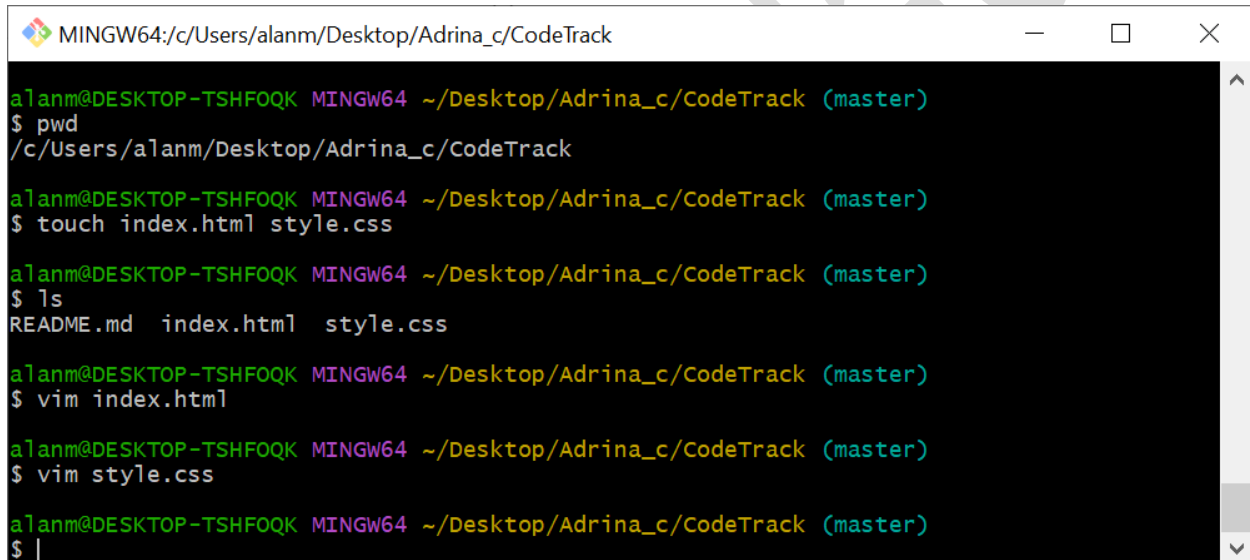
2. **Verify that the files have been created by listing the directory contents:**

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.



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

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).

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

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
```

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).

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

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(-)

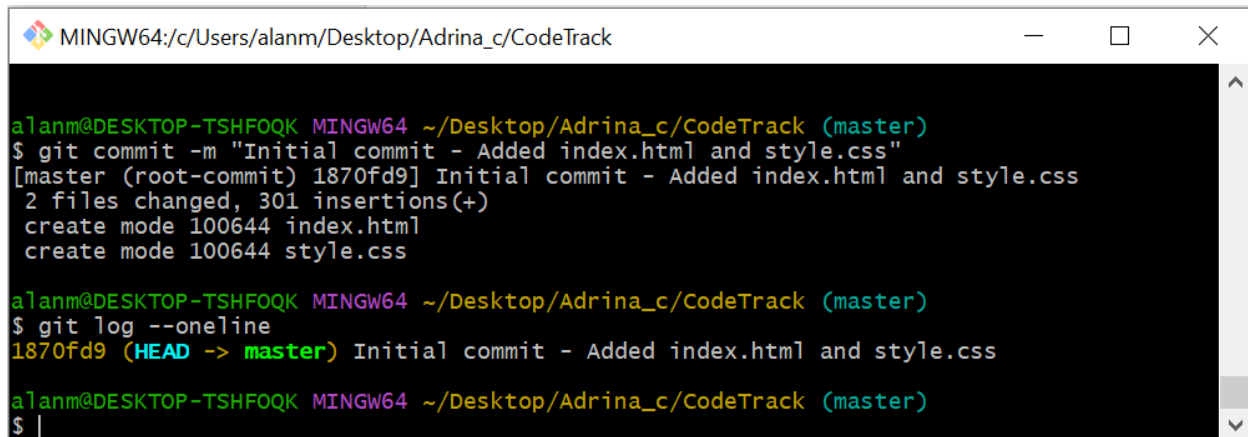
2. Verify the commit history:

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.



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

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git commit -m "Initial commit - Added index.html and style.css"
[master (root-commit) 1870fd9] Initial commit - Added index.html and style.css
2 files changed, 301 insertions(+)
create mode 100644 index.html
create mode 100644 style.css

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git log --oneline
1870fd9 (HEAD -> master) Initial commit - Added index.html and style.css

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

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:

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:

Your Group Number:

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.).

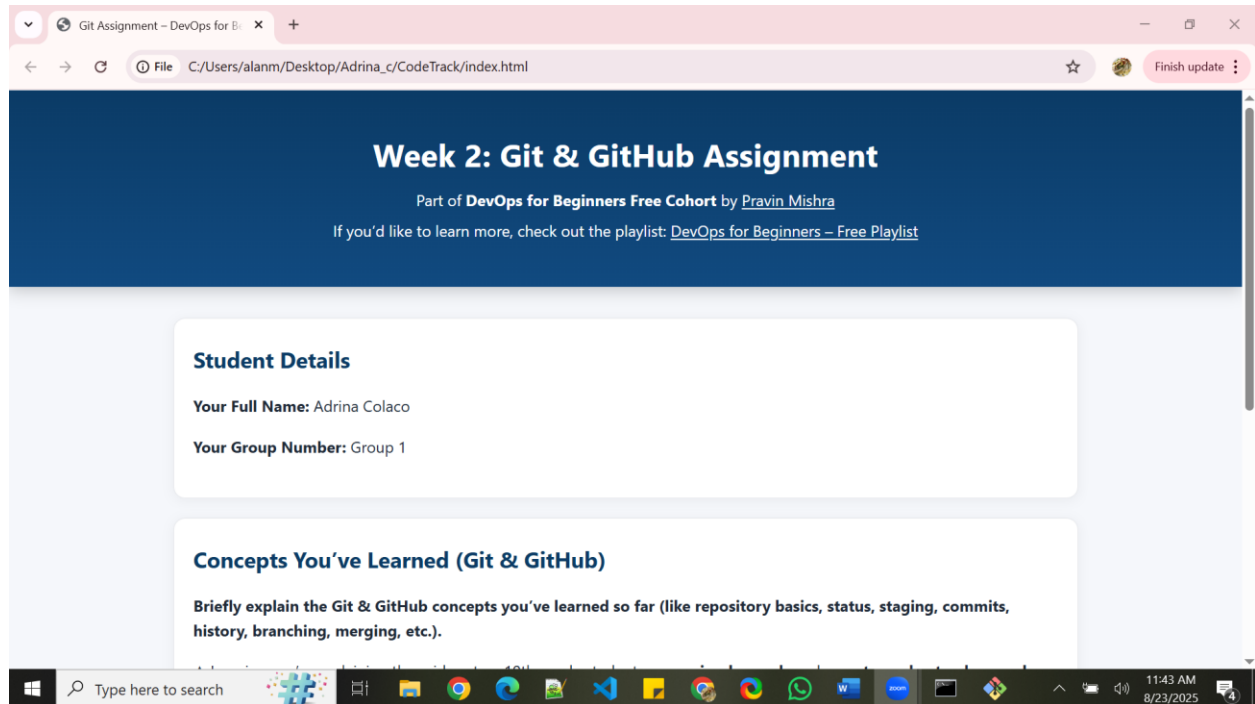
- ✦ Imagine you're explaining these ideas to a 10th-grade student, so use **simple words** and **easy-to-understand examples or analogies**.
- ✦ Also, think like a **DevOps/Cloud engineer** – sometimes applications fail, and while you may not fix the app code itself, you need to **look into code snippets, logs, or commit history** to understand what went wrong.
- ✦ For example: a wrong database connection string in code will show up in logs, and using Git history you can check *who changed it and when*.

Your Task:

- Explain Git concepts in your own words (as if teaching someone new).
- Connect at least one Git concept with a real-world debugging scenario. (e.g., "If an application suddenly stops connecting to the database, I can use `git log` to see if someone recently changed the configuration file.")
- Write 8–10 lines in simple language, add screenshots if possible.

© 2025 DevOps for Beginners - Pravin Mishra

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.

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ 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:   index.html

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

no changes added to commit (use "git add" and/or "git commit -a")

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 status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html

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)
$ git commit -m "23-Aug-2025: Updated student details in index.html"
[master 2500e9c] 23-Aug-2025: Updated student details in index.html
1 file changed, 2 insertions(+), 2 deletions(-)

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git log --oneline
2500e9c (HEAD -> master) 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)
$ |
```

To see all details of the commits use below command

git log

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git log
commit 2500e9c87480bfb7356ad22cfc27b7ef6aef1f1c (HEAD -> master)
Author: Adrina <adcola13@gmail.com>
Date: Sat Aug 23 11:53:42 2025 +0530

    23-Aug-2025: Updated student details in index.html

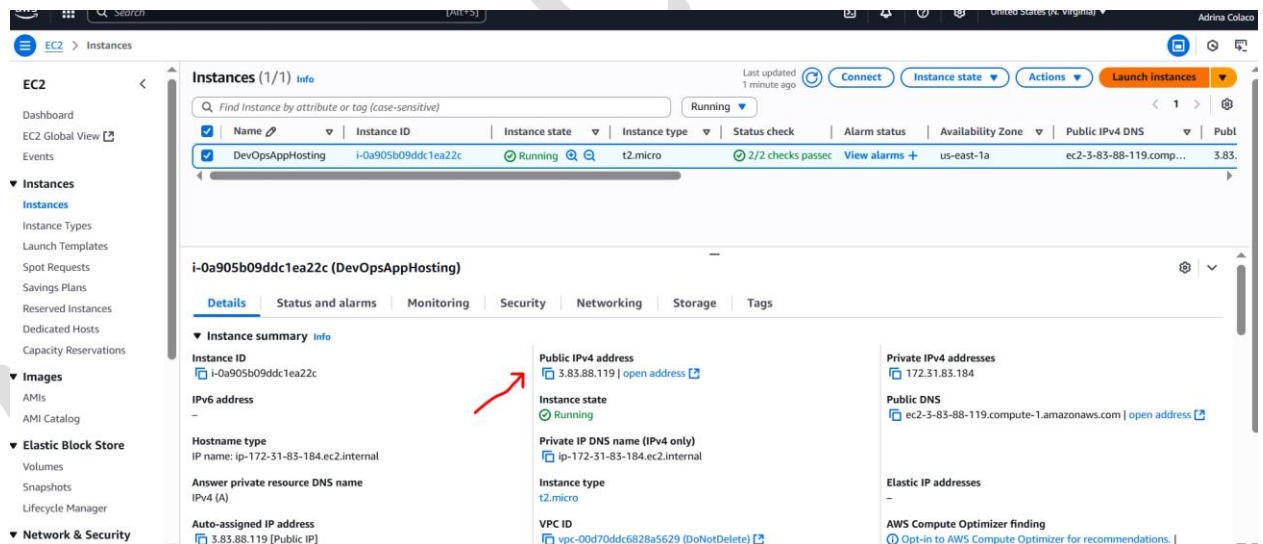
commit 1870fd98da2b78aec7f8f2a6be13264c0e06a0ba
Author: Adrina <adcola13@gmail.com>
Date: Sat Aug 23 11:27:48 2025 +0530

    Initial commit - Added index.html and style.css

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

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

1. Launch an EC2 Instance



EC2 > Security Groups > sg-07b436114694bb715 - launch-wizard-10 > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID
sgr-079e1a2d5ab0648cd

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	Custom	
HTTP	TCP	80	Anywhere...	
HTTPS	TCP	443	Anywhere...	

Add rule

Cancel

Preview changes

Save rules

- Open a terminal on your local machine.
- Use SSH to connect:

```

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ ls
23_8      DevopsAppHosting.pem  contact.html  mini_finance/  pubkey.pub
23_8.pub  README.md             index.html   pubkey         style.css

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ chmod 400 "DevopsAppHosting.pem"

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ ssh -i "DevopsAppHosting.pem" ec2-user@ec2-3-83-88-119.compute-1.amazonaws.com
The authenticity of host 'ec2-3-83-88-119.compute-1.amazonaws.com (3.83.88.119)' can't b
e established.
ED25519 key fingerprint is SHA256:g7UoQAk62Grd9bLc7knKAPEkUotjFJXJJYh4M1b02Bc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-83-88-119.compute-1.amazonaws.com' (ED25519) to the li
st of known hosts.

#_
~\####_ Amazon Linux 2023
~~\#####\
~~\###|
~~\#/
~~V~'-> https://aws.amazon.com/linux/amazon-linux-2023
~~~~
~~.-.-
~/m/'-
[ec2-user@ip-172-31-83-184 ~]$ |
```

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
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

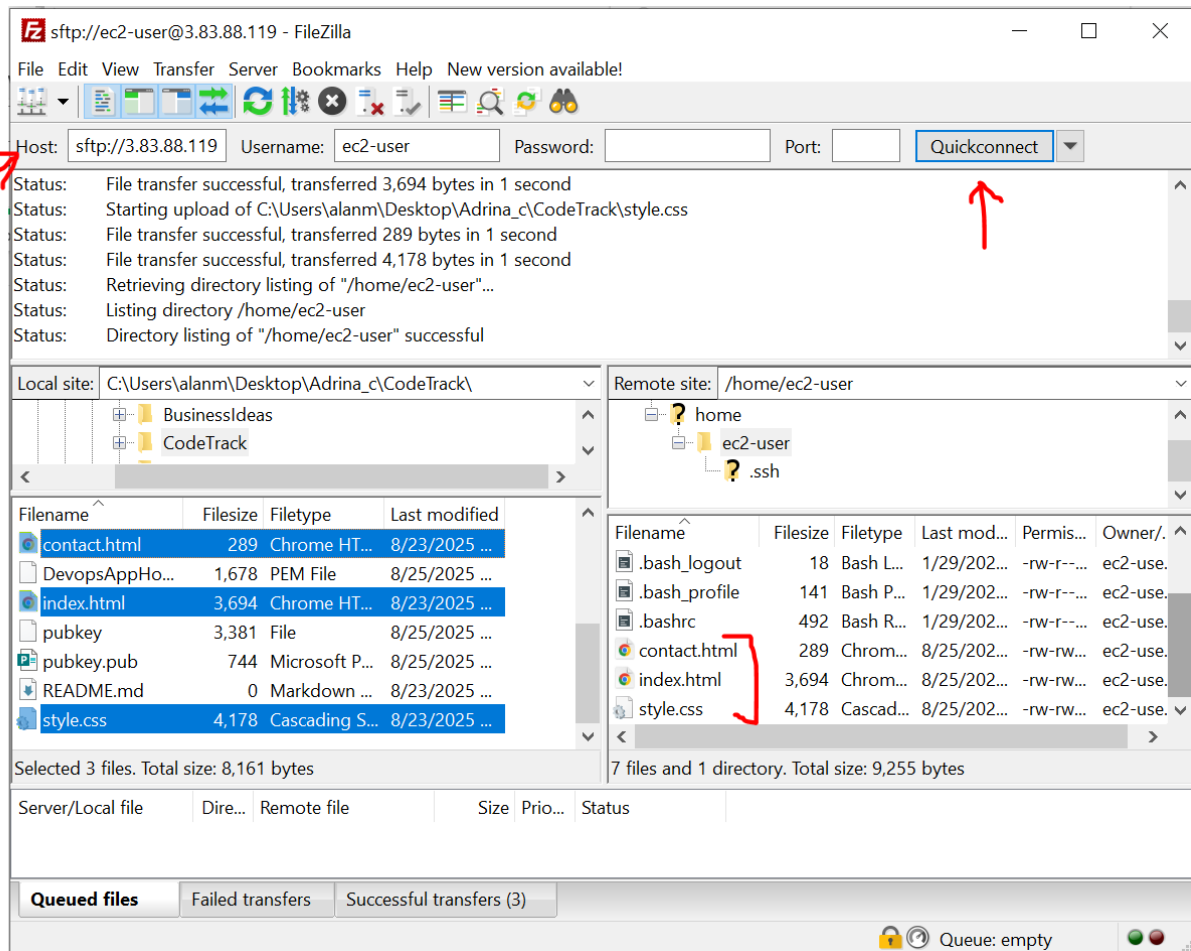
Total download size: 1.1 M
Installed size: 3.7 M
Downloading Packages:
(1/7): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm
(2/7): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm
(4/7): nginx-1.28.0-1.amzn2023.0.2.x86_64.rpm
(5/7): nginx-core-1.28.0-1.amzn2023.0.2.x86_64.rpm
(6/7): nginx-filesystem-1.28.0-1.amzn2023.0.2.noarch.rpm
(7/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch 1/1
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 1/7
Installing : libunwind-1.4.0-5.amzn2023.0.2.x86_64 2/7
Installing : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 3/7
Installing : nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64 4/7
Installing : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 5/7
Installing : nginx-1:1.28.0-1.amzn2023.0.2.x86_64 6/7
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
libunwind-1.4.0-5.amzn2023.0.2.x86_64
nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
nginx-1:1.28.0-1.amzn2023.0.2.x86_64
nginx-filesystem-1:1.28.0-1.amzn2023.0.2.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:
 - Use FileZilla to move the project located on the local machine to EC2(MobaXterm can also be used)
 - Go to Edit → Settings → Under Connection → SFTP → add the PEM key that downloaded while creating your EC2 instance.
 - 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 NGINC web directory.

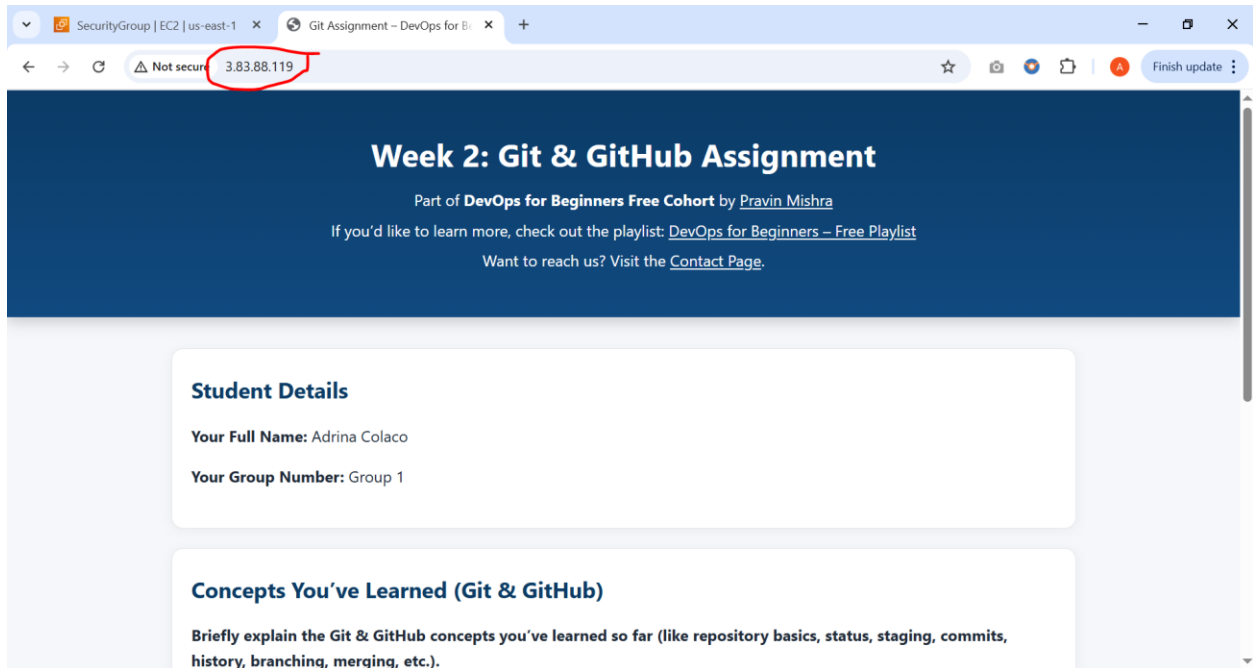
1. `sudo cp /home/ec2-user/index.html /usr/share/nginx/html/index.html`
2. `sudo cp /home/ec2-user/contact.html /usr/share/nginx/html/contact.html`
3. `sudo cp /home/ec2-user/style.css /usr/share/nginx/html/style.css`

```
[ec2-user@ip-172-31-83-184 html]$ sudo cp /home/ec2-user/index.html /usr/share/nginx/html/index.html
[ec2-user@ip-172-31-83-184 html]$ sudo cp /home/ec2-user/contact.html /usr/share/nginx/html/contact.html
[ec2-user@ip-172-31-83-184 html]$ sudo cp /home/ec2-user/style.css /usr/share/nginx/html/style.css
[ec2-user@ip-172-31-83-184 html]$ ls
404.html  50x.html  contact.html  icons  index.html  nginx-logo.png  poweredby.png  style.css
[ec2-user@ip-172-31-83-184 html]$
```

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.



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>
```



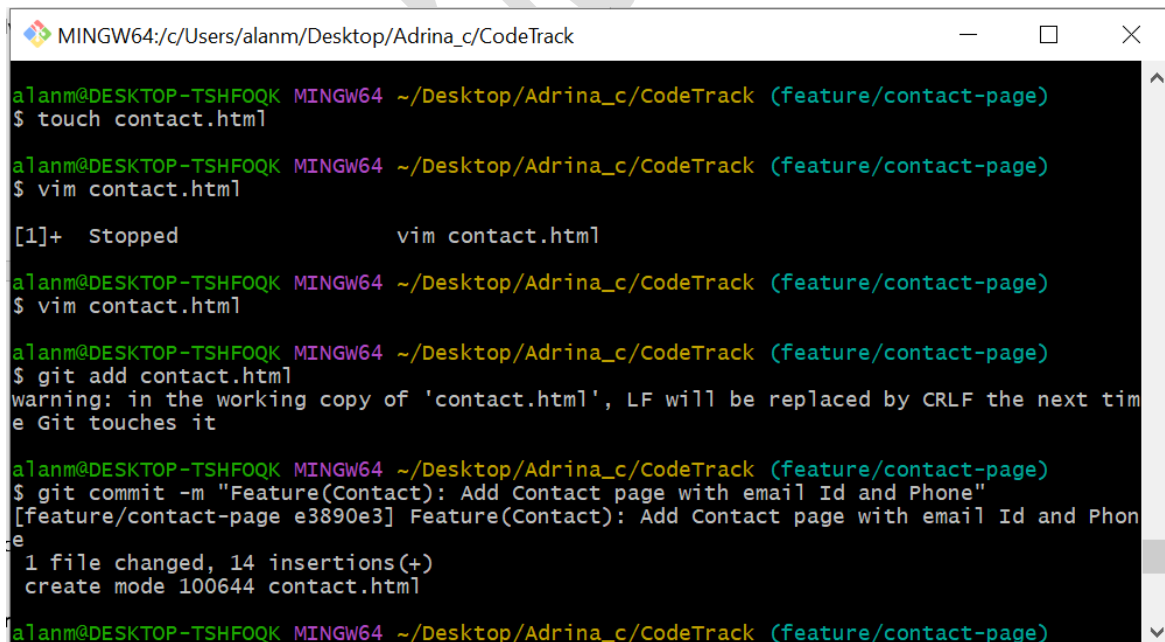
A screenshot of a code editor window titled "MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack". The editor displays the following HTML code:

```
<!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>
```

The status bar at the bottom indicates "contact.html [dos] (13:30 23/08/2025) 14,0-1 All" and "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"



A screenshot of a terminal window titled "MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack". The terminal shows the following commands and output:

```
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 time 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 Phone
1 file changed, 14 insertions(+)
create mode 100644 contact.html

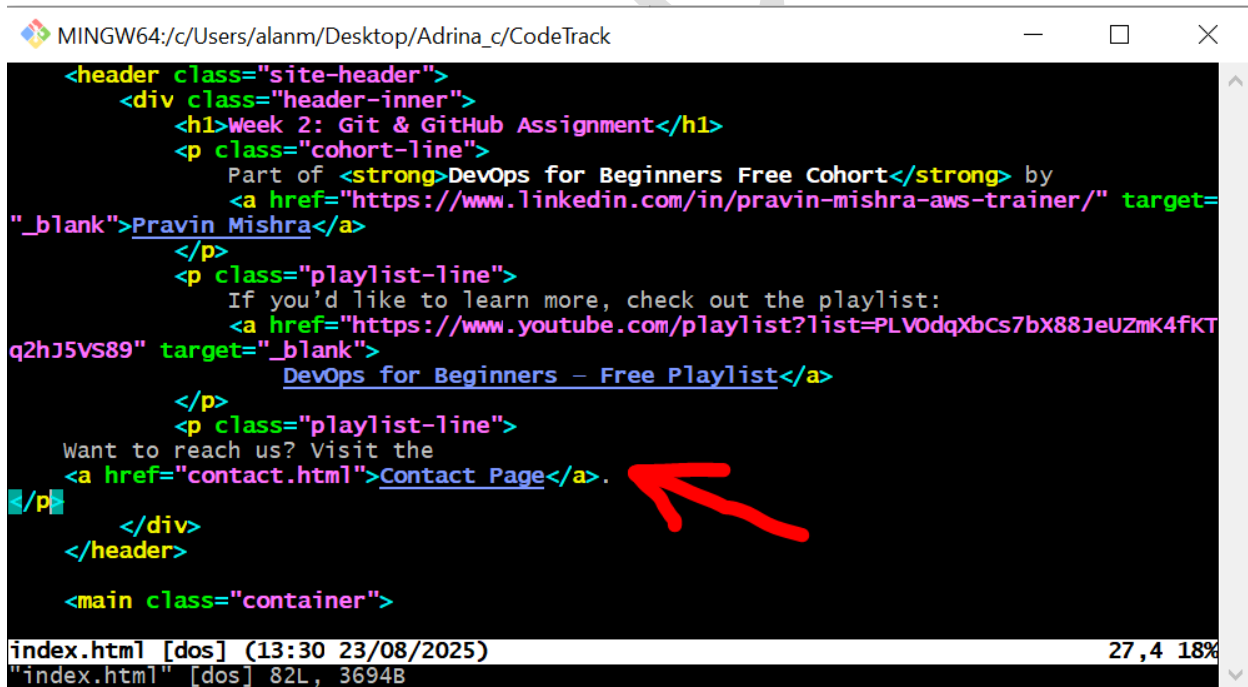
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
```

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>
```



The screenshot shows a code editor window titled "MINGW64:/c:/Users/alanm/Desktop/Adrina_c/CodeTrack". The code is for "index.html" and includes a header section with a site header and a main container. The code is as follows:

```
<header class="site-header">
  <div class="header-inner">
    <h1>Week 2: Git & GitHub Assignment</h1>
    <p class="cohort-line">
      Part of <strong>DevOps for Beginners Free Cohort</strong> by
      <a href="https://www.linkedin.com/in/pravin-mishra-aws-trainer/" target=
"_blank">Pravin Mishra</a>
    </p>
    <p class="playlist-line">
      If you'd like to learn more, check out the playlist:
      <a href="https://www.youtube.com/playlist?list=PLVOdqXbCs7bX88JeUZmK4fKT
q2hJ5VS89" target="_blank">
        DevOps for Beginners – Free Playlist</a>
    </p>
    <p class="playlist-line">
      Want to reach us? Visit the
      <a href="contact.html">Contact Page</a>.
    </p>
  </div>
</header>

<main class="container">
```

A red arrow points to the newly added paragraph in the code, which is the one with the class "playlist-line" and the text "Want to reach us? Visit the Contact Page".

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)
$ 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=PLV0dqXbCs7bX88JeUZmK4fKT
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**. ☒

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git merge feature/contact-page
Updating 2500e9c..ba67ae5
Fast-forward
 contact.html | 14 ++++++
 index.html   |  4 ++++
 2 files changed, 18 insertions(+)
 create mode 100644 contact.html

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

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

Task 6 — Inspect history (nice graph)

1. git log --oneline --graph --decorate --all

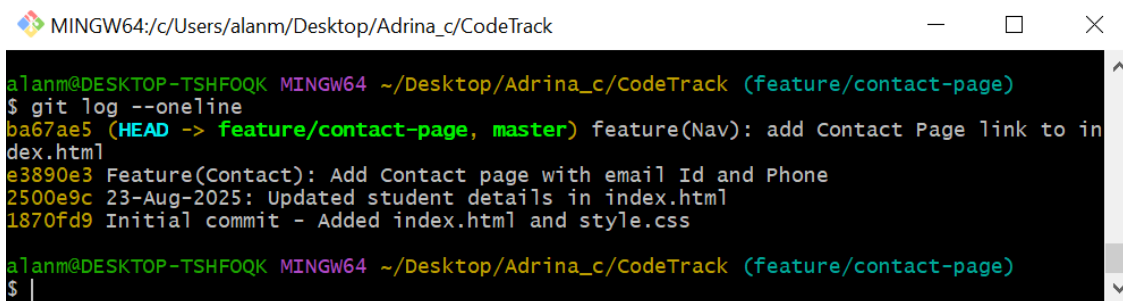
(Optional cleanup)

1. `git branch -d feature/contact-page`

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

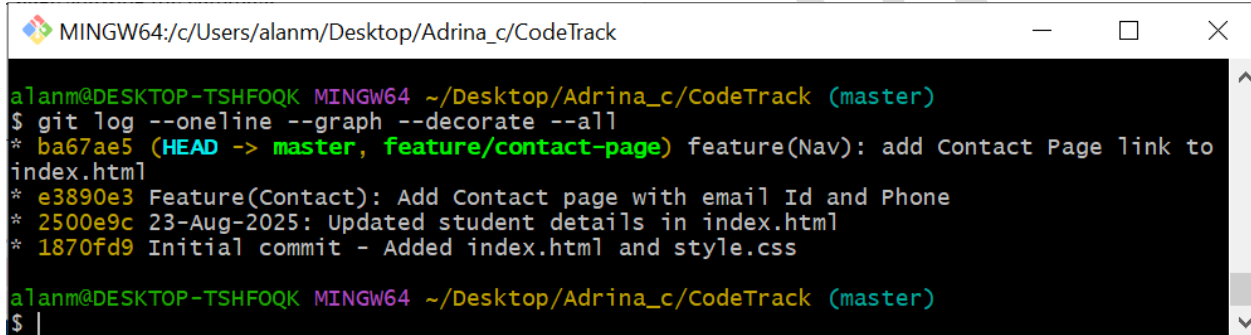
feat(contact): add contact page...

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



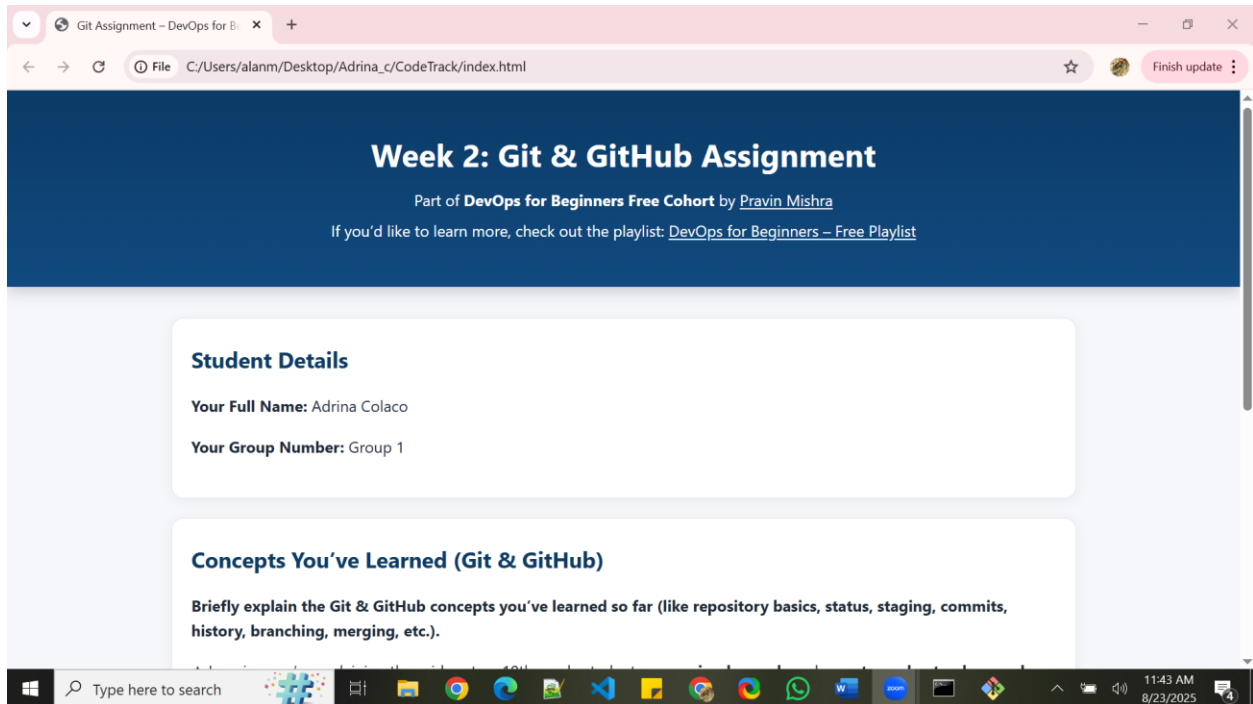
```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (feature/contact-page)
$ git log --oneline
ba67ae5 (HEAD -> feature/contact-page, master) feat(nav): add Contact Page link to index.html
e3890e3 Feature(Contact): Add Contact page with email 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 (feature/contact-page)
$
```

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

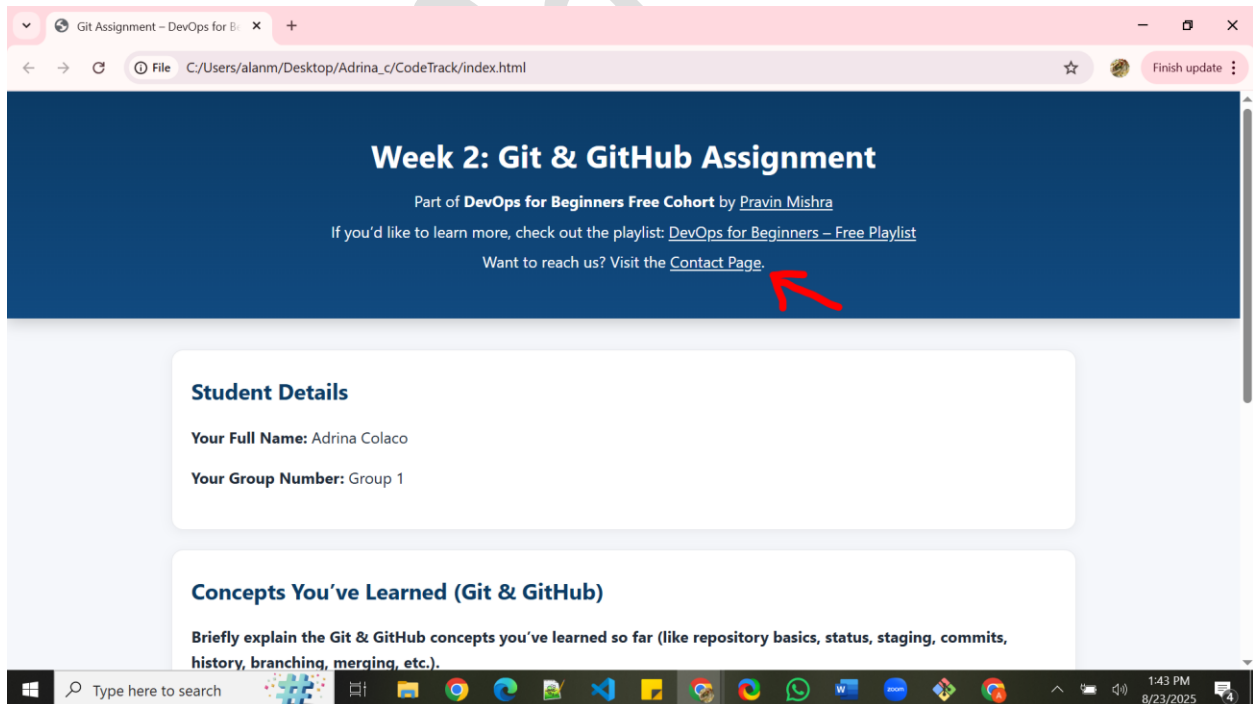


```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git log --oneline --graph --decorate --all
* ba67ae5 (HEAD -> master, feature/contact-page) feat(nav): add Contact Page link to index.html
* e3890e3 Feature(Contact): Add Contact 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)
$
```

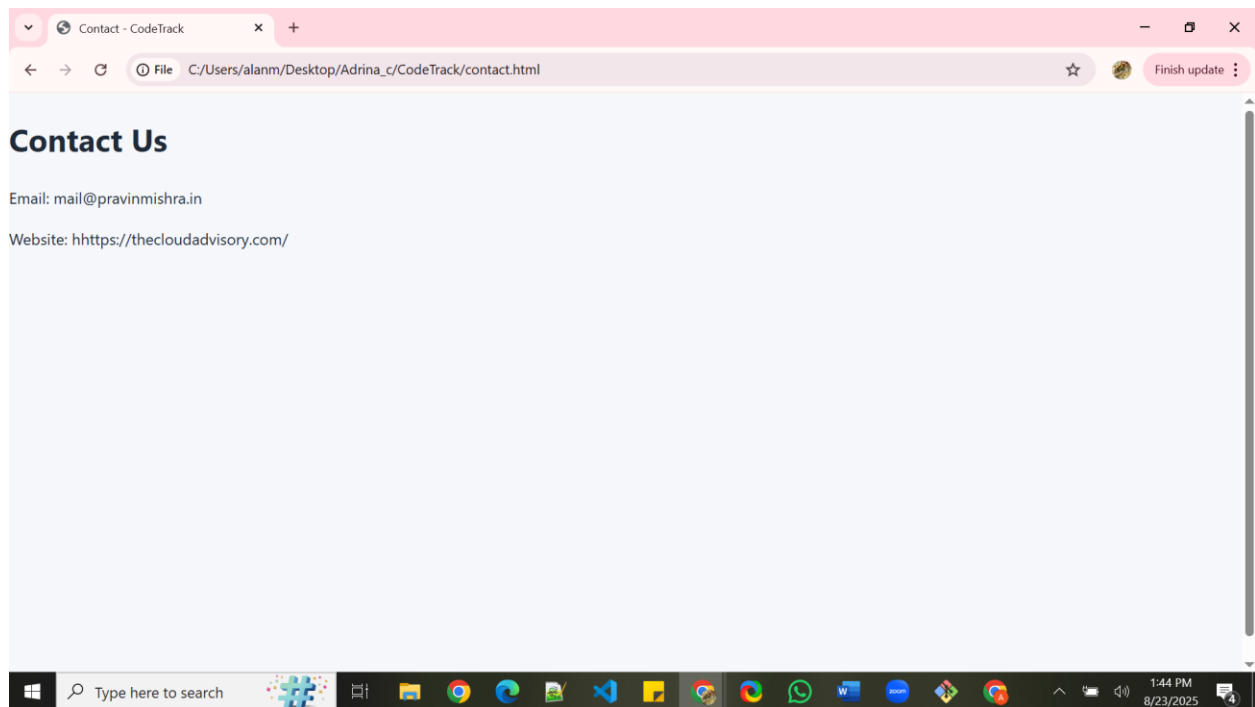
Screenshot E: Browser screenshot of index.html **before merge** (no Contact Page link visible).



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).



Explain why the link wasn't visible before the merge and why it appears after.

The changed code from the Branch was not yet present in the Main(master) branch before Git Merge.

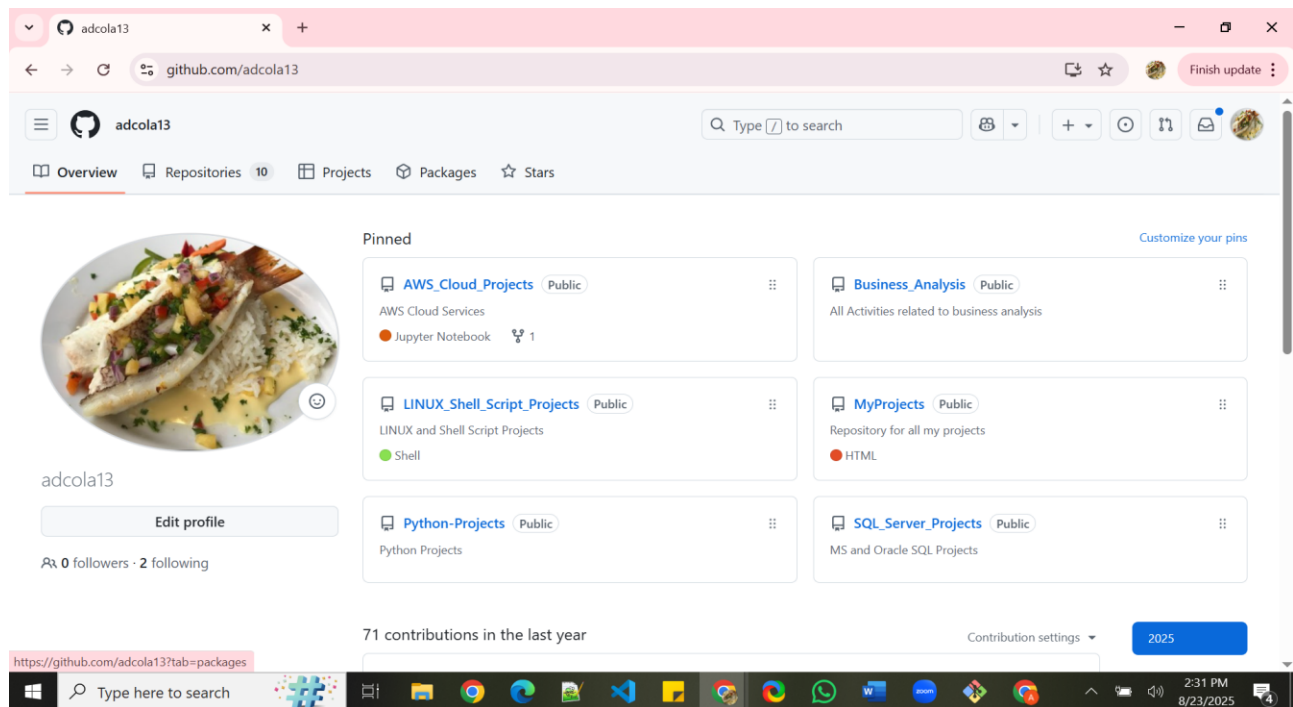
Once Git merge command was run, the code linking to Contact.html file moved to the index.html file present in main(master) branch as well.

Assignment 8 - Setting Up GitHub for CodeTrack

Task 1 — Create a GitHub Account

1. Go to [GitHub](https://github.com) 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.



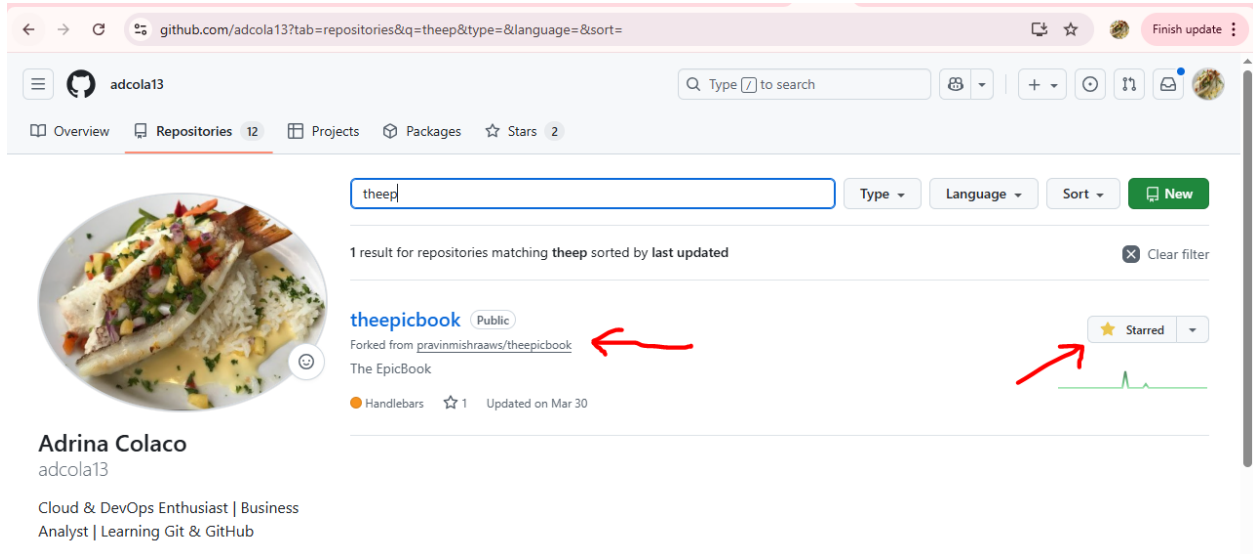
Task 2 — Explore GitHub Features

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.


▼

adcola13

×

+

← → ↻ 🔍 github.com/adcola13



Adrina Colaco

adcola13

Cloud & DevOps Enthusiast | Business Analyst | Learning Git & GitHub

Edit profile

👤 0 followers · 2 following

✉ adcola13@gmail.com

🌐 in/adrina-colaco

🔗 https://medium.com/@adcola13

Pinned

🖨️ **AWS_Cloud_Projects** Publ

AWS Cloud Services

🟠 Jupyter Notebook 🔗 1

🖨️ **LINUX_Shell_Script_Project**

LINUX and Shell Script Projects

🟢 Shell

🖨️ **Python-Projects** Public

Python Projects

72 contributions in the last ye

	Aug	Sep	Oct	Nov
Mon				
Wed				
Fri				

Why is it important to have a professional GitHub profile as a developer?

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 and 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.

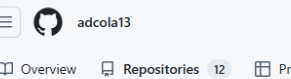
Task 1 — Fork & Authenticate

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

← → ↺ 🌐 github.com/adcola13?tab=repositories&q=mini&type=&language=&sort=

☰ 🐙 adcola13 🔍 Type [Z] to search 🗑️ + + 🔄 📧 🌱

📖 Overview 📁 **Repositories 12** 📁 Projects 📦 Packages ⭐ Stars 2



Adrina Colaco
adcola13

Cloud & DevOps Enthusiast | Business Analyst | Learning Git & GitHub

mini

Type Language Sort **New**

1 result for repositories matching **mini** sorted by **last updated** ✕ Clear filter

mini_finance Public

Forked from [pravinmishraaws/mini_finance](#)

Project of Git & GitHub for Absolute Beginners Udemy Course

🔴 HTML Updated 6 hours ago

☆ Star

- **SSH** (recommended):

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`).

1. Copy the SSH key:
2. `cat ~/.ssh/id_rsa.pub`

```

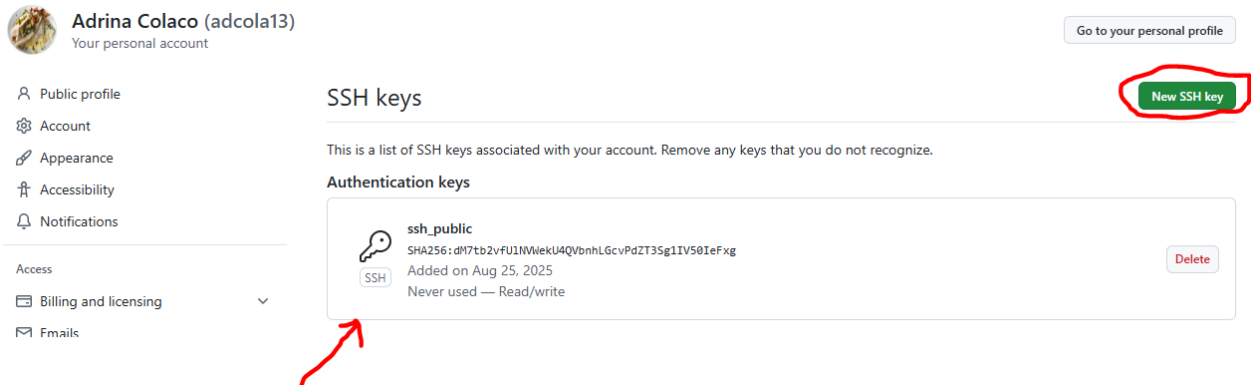
alann@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ ssh-keygen -t rsa -b 4096 -C"adcoai13@gmail.com"
generating public/private rsa key pair.
Enter file in which to save the key (/c:/Users/alann/.ssh/id_rsa):
Enter passphrase for /c:/Users/alann/.ssh/id_rsa (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c:/Users/alann/.ssh/id_rsa
Your public key has been saved in /c:/Users/alann/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dM7tb2vFU1NVvekU4QVbhhLGVcPdZT35glIV501eFvg adcoai13@gmail.com
The key's randomart is:
-----[RSA 4096]-----
      .
     o.B
    .o.XoD
   . . .X+X
  . + . .++
   S o . .
    . . .
   . . .
  . . .
   oo+
  -----[SHA256]-----

alann@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ cat -v ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQNGktyVAtasUMBfhTeTQLFzVavXEDKh510PD76HchddHk4Vb5Aio5g7eL3LR8LmvuULVfn802Jh2721EAatPMGzmfKockbChk1kQdYecfWgFi+FjXccxq83rIrU5KgcN+KtI/7Zhn3R2eH2ec14jCqZdoT-OSFKFBESraMrwbbqgDz0ebwHTS5IYjAtopHcJ3xmqwGvoCdc+JwMGSd1qNn/p/59UkX35Z-x0AbFjUXU7DvmyyZP4b7Hyswidu-bG1VsyFF04vF9dKD3sP05/CWPjQBNQ613He9Uj/mn2F/y/cv31uLCPNvYgHkq8LrBdu55xV6-7DhVxvUqgb0B0+Pi/AgRHnkrGXPXSsmXBM+peQg+q/yf+82Bdoag5ng09UKRfHEH2Uj125F0J2jYm64QvE8F5Z0dqW57kFue+CRE3B37amWg7Y/eonkvriT06v33j+DxfY10VWpU2vBn4m8Adm0Tk+hwk=M220x6EBn67xtChvY1lqptYLk6Ehx1oRfyd+4cyRc1tZf/gqRCf5oyEa19wQm4YfhkBVuWgs/rjZtPih5dRJo18iAPhDfOm6wk+mbo2mwY5Xbb5No/N9D11Xaj/FASG6LoJbDwJRv0tpMY1Th9p37w82kCEn6q1UQO== adcoai13@gmail.com

alann@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ !

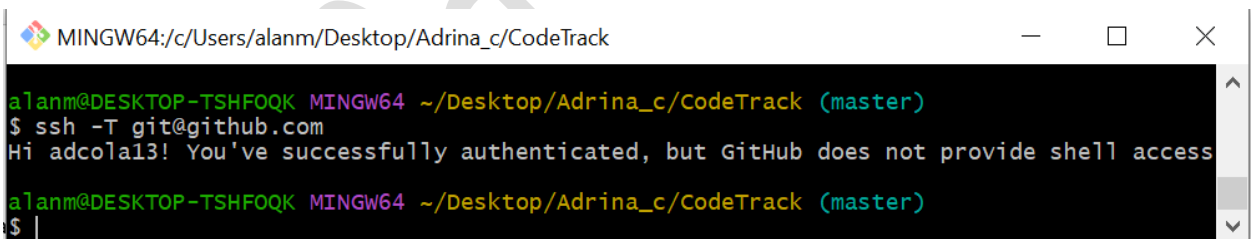
```

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:

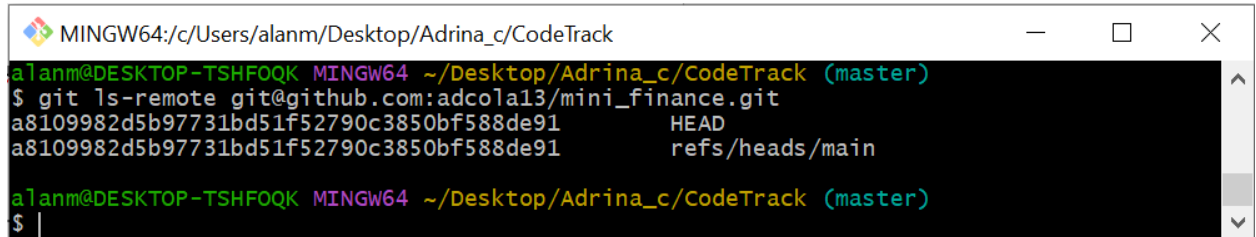
5. `git clone git@github.com:yourusername/repository.git`
6. `git config --global url."git@github.com:".insteadOf "https://github.com/"`

• OR HTTPS:

1. `git config --global credential.helper cache`

Test with:

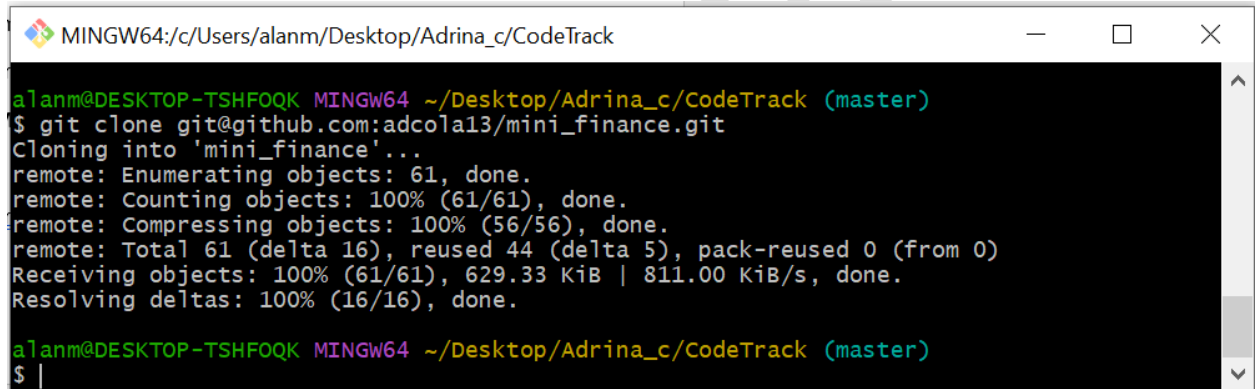
2. `git ls-remote git@github.com:yourusername/mini_finance.git`
3. **Expected outcome:** You have forked the repo and your terminal is authenticated for Git ops.



```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$ git ls-remote git@github.com:adcola13/mini_finance.git
a8109982d5b97731bd51f52790c3850bf588de91      HEAD
a8109982d5b97731bd51f52790c3850bf588de91      refs/heads/main
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack (master)
$
```

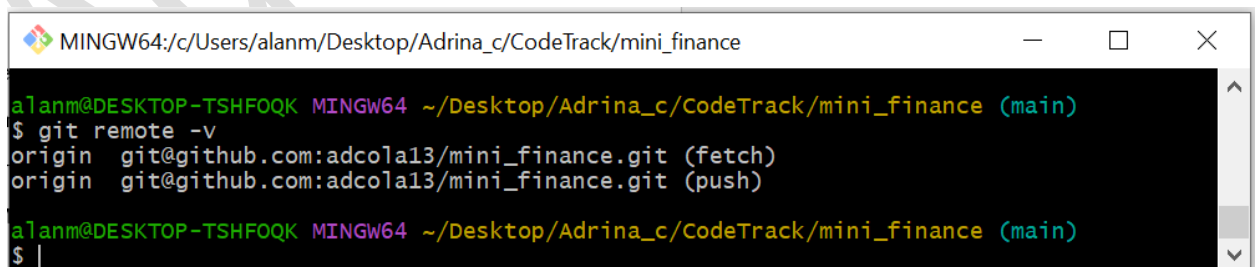
Task 2 — Clone Your Fork Locally

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.

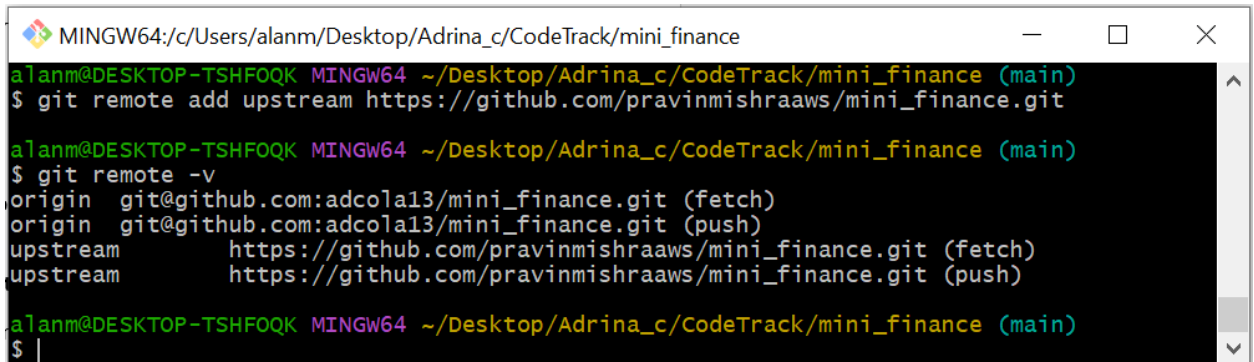


```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance
alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)
$ git remote -v
origin git@github.com:adcola13/mini_finance.git (fetch)
origin git@github.com:adcola13/mini_finance.git (push)
alanm@DESKTOP-TSHFOQK MINGW64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

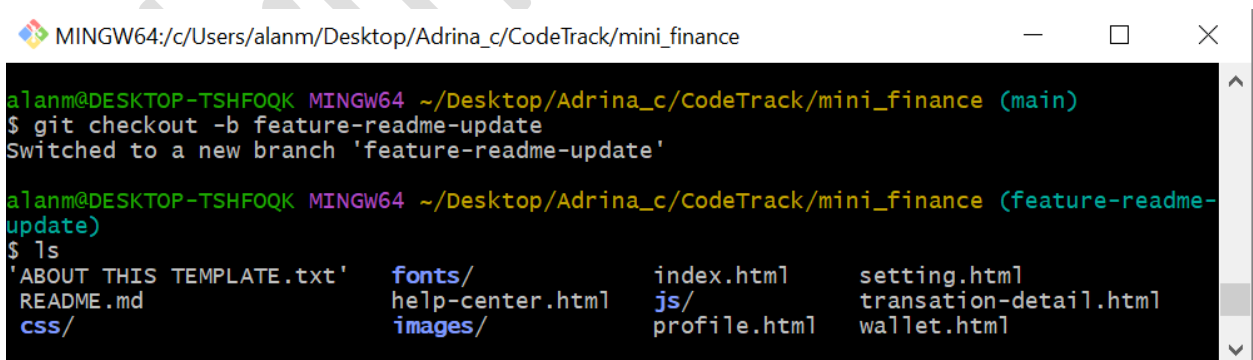
A terminal window titled 'MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance'. The prompt is 'alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)'. The user enters '\$ git remote add upstream https://github.com/pravinmishraaws/mini_finance.git'. The prompt changes to 'alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)'. The user enters '\$ git remote -v'. The output shows: '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)', and 'upstream https://github.com/pravinmishraaws/mini_finance.git (push)'. The prompt returns to '\$ |'.

What is the purpose of the upstream remote in your workflow?

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.

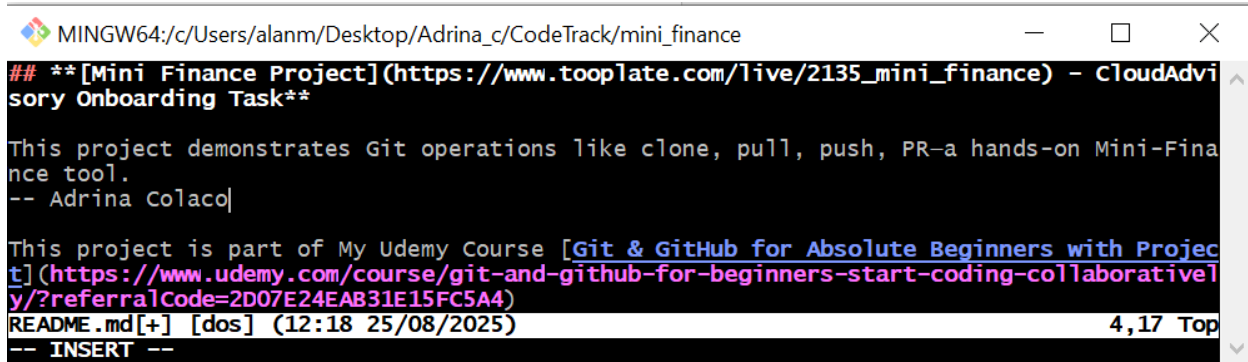
Task 3 — Create a Feature Branch & Make a Change

2. Create a new branch:
 1. `git checkout -b feature-readme-update`

A terminal window titled 'MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance'. The prompt is 'alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (main)'. The user enters '\$ git checkout -b feature-readme-update'. The output is 'Switched to a new branch 'feature-readme-update''. The prompt changes to 'alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-update)'. The user enters '\$ ls'. The output is a directory listing: 'ABOUT THIS TEMPLATE.txt', 'README.md', 'css/', 'fonts/', 'help-center.html', 'images/', 'index.html', 'js/', 'profile.html', 'setting.html', 'transation-detail.html', and '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.”



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

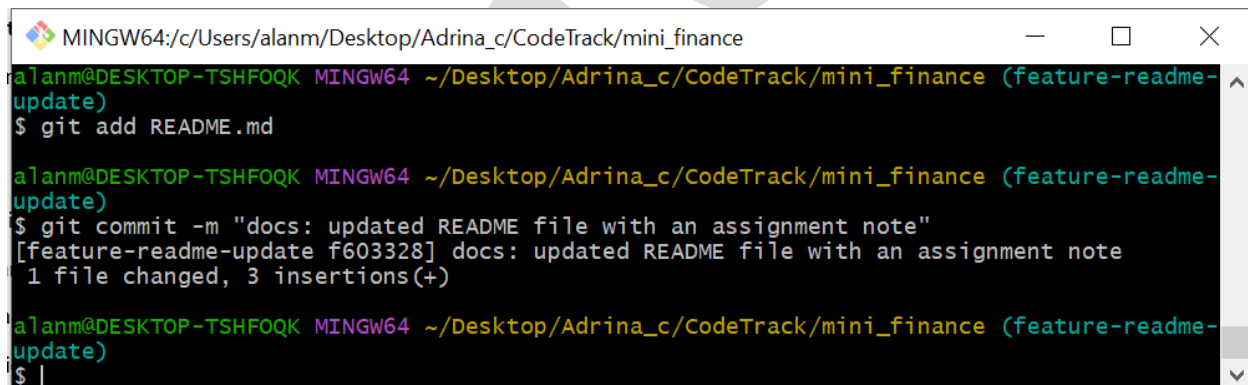
This project demonstrates Git operations like clone, pull, push, PR—a hands-on Mini-Finance tool.
-- Adrina Colaco

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-collaborative/?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`)



```
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

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

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

alanm@DESKTOP-TSHFOQK MINGW64 ~/Desktop/Adrina_c/CodeTrack/mini_finance (feature-readme-update)
$ git rebase main
Current branch feature-readme-update is up to date.

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

Why did you rebase or merge from upstream/main before pushing?

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:

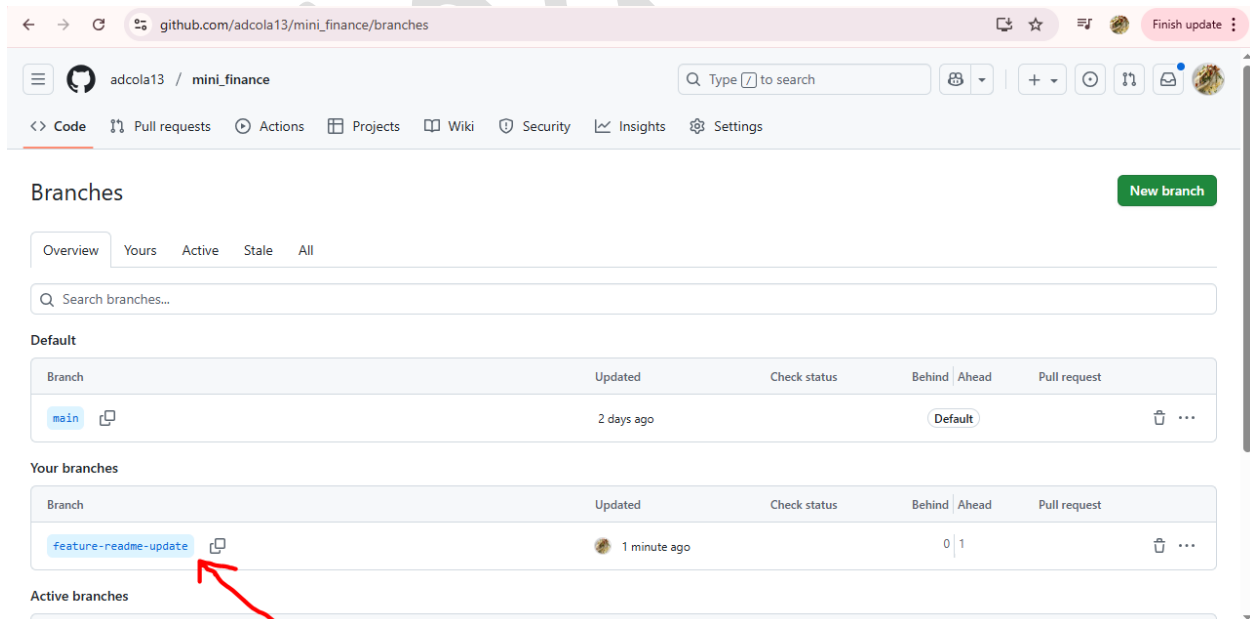
1. `git push -u origin feature-readme-update`

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

```
MINGW64:/c/Users/alanm/Desktop/Adrina_c/CodeTrack/mini_finance
$ git push -u origin feature-readme-update
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 415 bytes | 207.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature-readme-update' on GitHub by visiting:
remote:   https://github.com/adcola13/mini_finance/pull/new/feature-readme-update
remote:
To github.com:adcola13/mini_finance.git
 * [new branch]      feature-readme-update -> feature-readme-update
branch 'feature-readme-update' set up to track 'origin/feature-readme-update'.

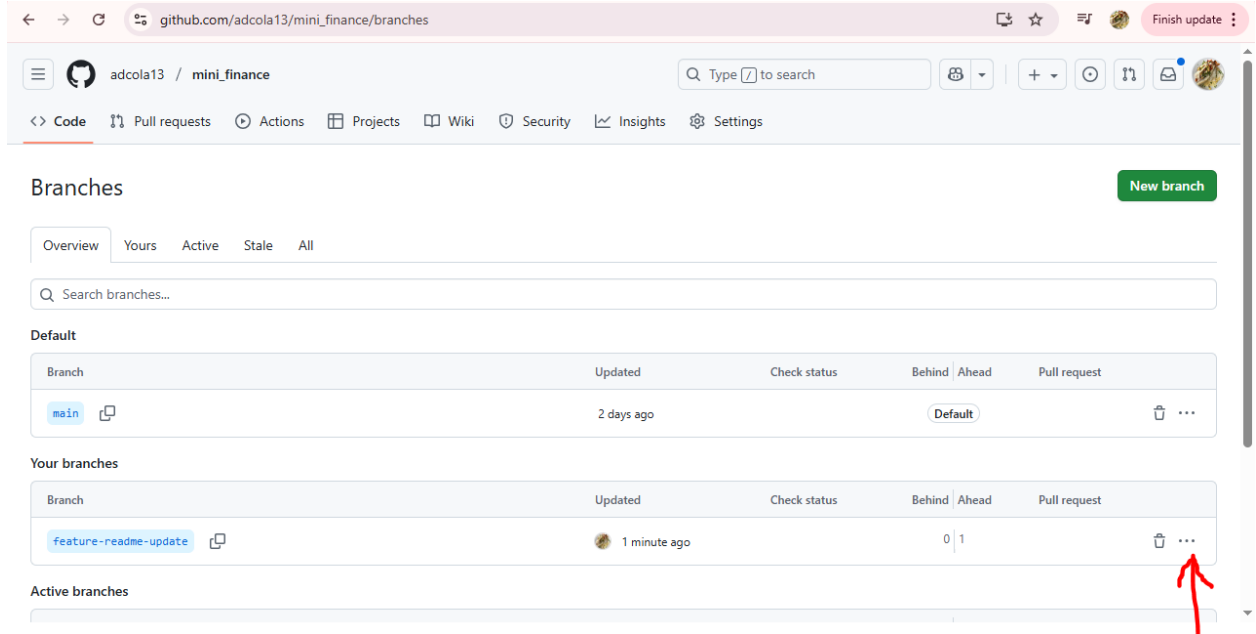
$
```

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**.

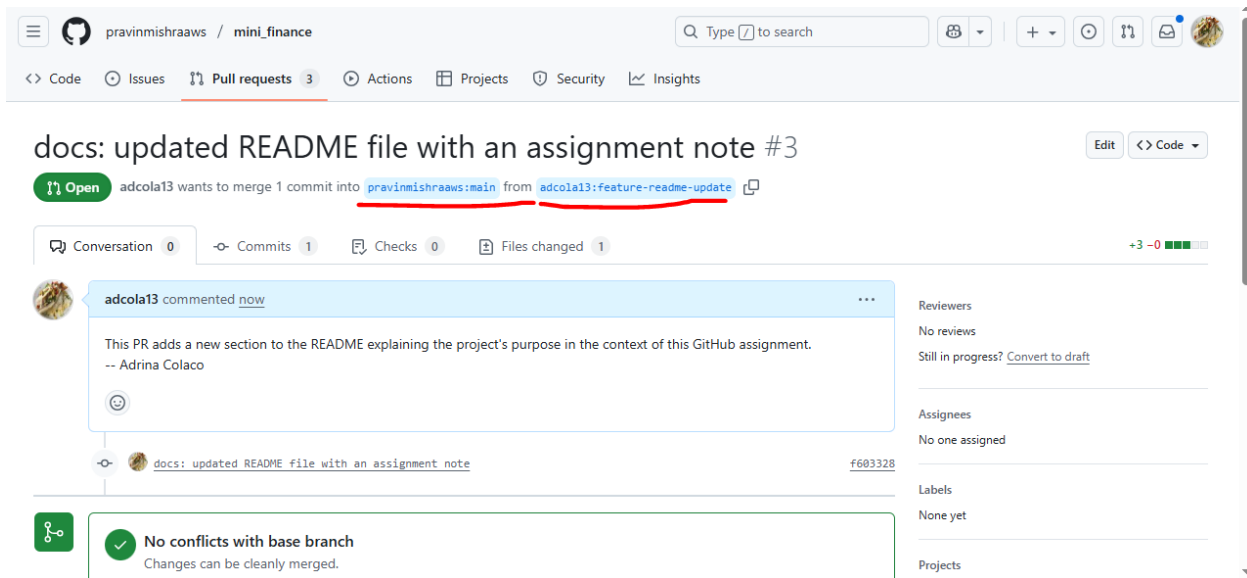


3. Make sure it's targeting pravinmishraaws/mini_finance:main from your feature-readme-update branch.
 1. Title: "docs: update README with assignment note"
 2. In the body, add a short description:
"This PR adds a new section to the README explaining the project's purpose in the context of this GitHub assignment."

4. Submit the **Pull Request**.

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