Version Control System (VCS)

TEKsystems Global Services

WORKFORCE DEVELOPMENT

LOGISTICS

Class Hours:

- Instructor will set class start and end times.
- There will be regular breaks in class.

Telecommunication:

- Turn off or set electronic devices to silent (not vibrate)
- Reading or attending to devices can be distracting to other students
- Try to delay until breaks or after class

Miscellaneous:

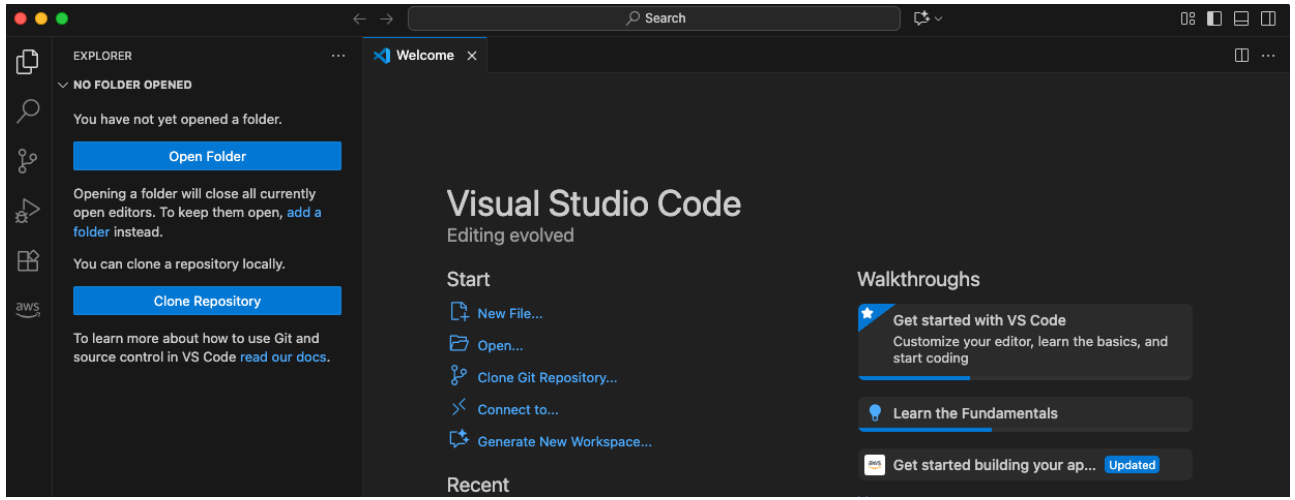
- Courseware
- Bathroom
- Fire drills

VCS

Day 2 Learning Outcomes:

- Use Visual Studio Code to clone, manage, and update GitHub repositories
- Create and merge Pull Requests to update the main branch
- Tag code and package releases using semantic versioning
- Add collaborators and configure repository permissions and settings

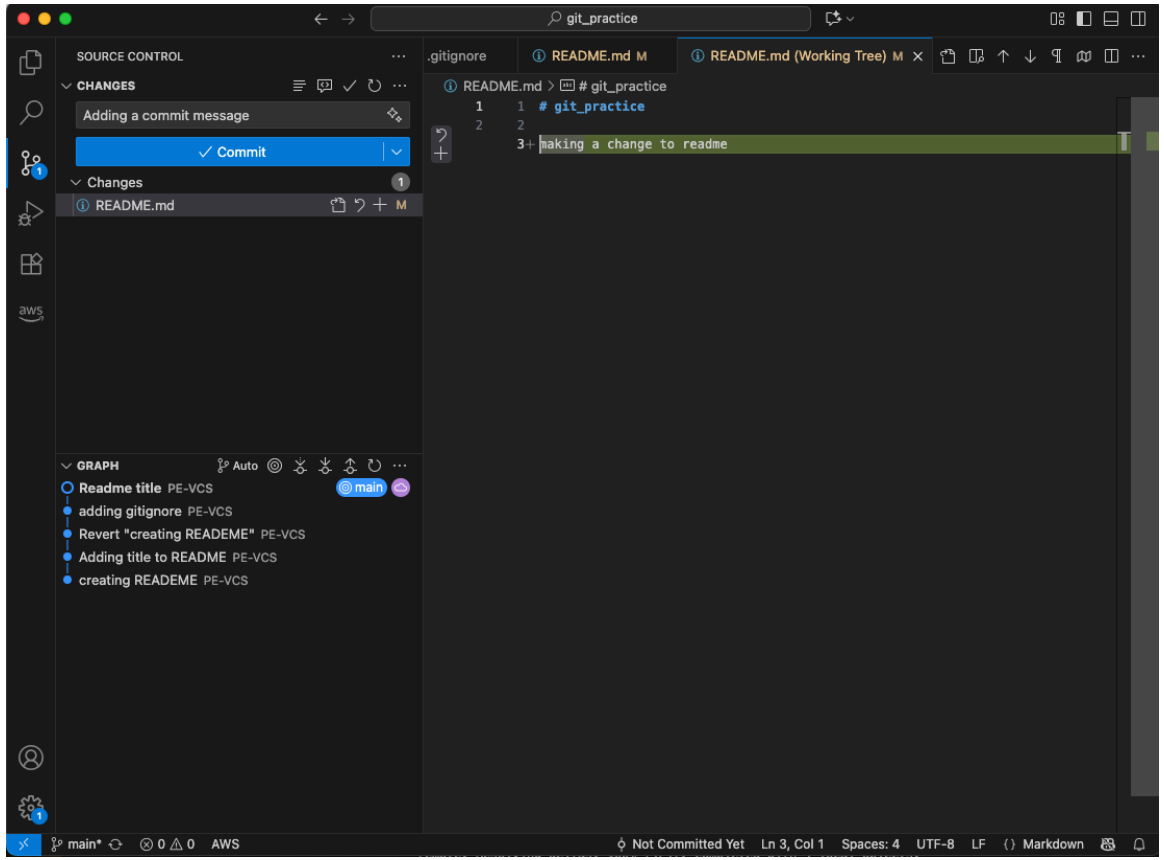
VISUAL STUDIO CODE



lightweight yet powerful source code editor developed by Microsoft. Supports multiple languages and frameworks with rich extensions and customization.

- Integrated Development Environment (IDE) features without the bloat
- Extensions Marketplace – Add plugins for Docker, Python, Terraform, YAML, Markdown, etc.
- Version Control Integration – Built-in support for Git and GitHub (commit, branch, merge directly in VS Code)
- AI Assistance – GitHub Copilot suggests code, comments, and entire functions using AI
- Cross-Platform – Runs on Windows, macOS, and Linux

VISUAL STUDIO CODE



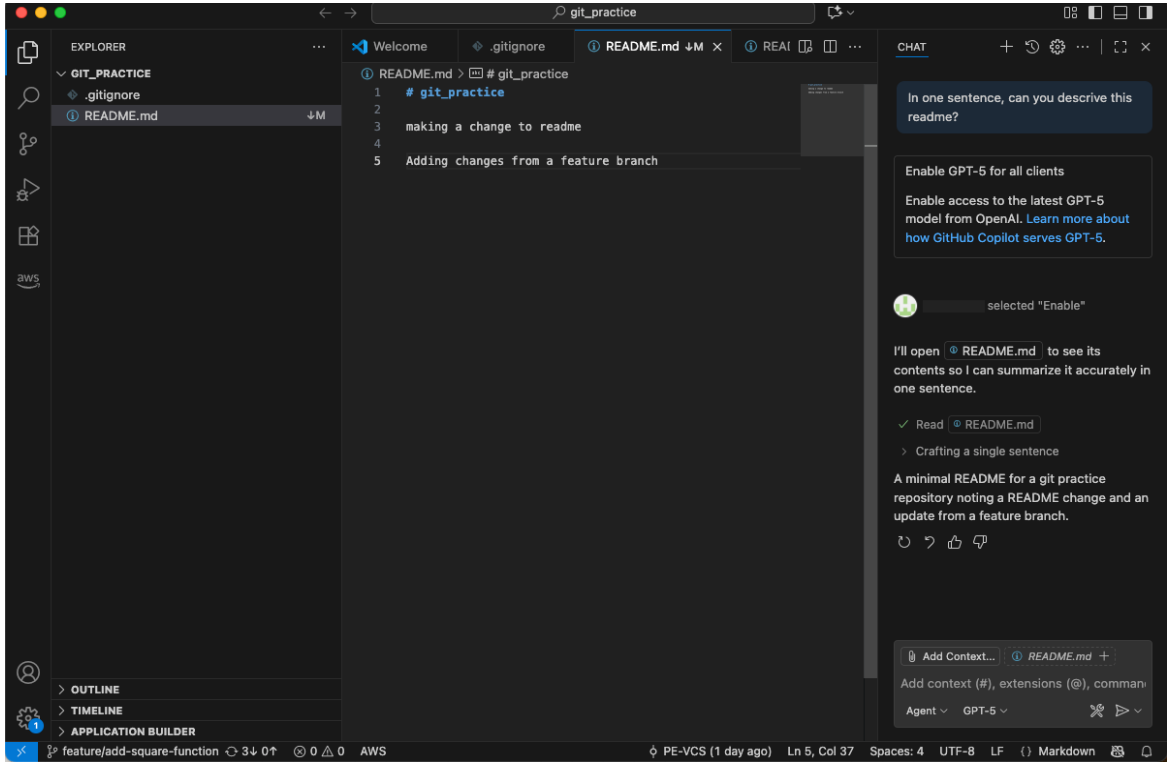
Integrated Git Support:

VS Code detects repositories automatically and displays changes in the Source Control panel.

You can stage, commit, and push changes directly from the editor.

The diff view highlights modifications between your working copy and the last commit.

GITHUB COPILOT



GitHub in VS Code makes it easy to manage repositories, commit changes, and push updates without ever leaving the editor. Its extensive library of extensions supports nearly every major language and tool, from Python and Docker to Kubernetes and Terraform, making it a universal workspace for developers.

With **GitHub Copilot**, VS Code becomes even more powerful. Copilot uses AI to suggest lines of code, complete functions, and even write tests based on context. This tight integration of **AI and version control** allows developers to code faster, learn new patterns, and maintain cleaner, more consistent projects — all within one tool.

LAB 2.1 – VISUAL STUDIO CODE

LAB INTRODUCTION

In this lab, you'll practice what you already know about Git, but this time through Visual Studio Code's built-in tools.

You will:

- Use the Source Control view to track file changes
- Explore diffs to see what has been modified
- Stage and commit changes from inside the editor
- Push updates to GitHub using the integrated interface

POP QUIZ: GIT CLI

Which statement about VCS in Visual Studio code is not true?

A. Visual Studio Code supports git
B. Visual Studio Code can run all the local git commands
C. Visual Studio Code resolves all conflicts on its own
D. Visual Studio Code can push and pull from remote GitHub repositories

POP QUIZ: GIT CLI

Which statement about VCS in Visual Studio code is not true?

A. Visual Studio Code supports git
B. Visual Studio Code can run all the local git commands
C. Visual Studio Code resolves all conflicts on its own
D. Visual Studio Code can push and pull from remote GitHub repositories

VCS

Push, Pull and Merge

2

BRANCHING

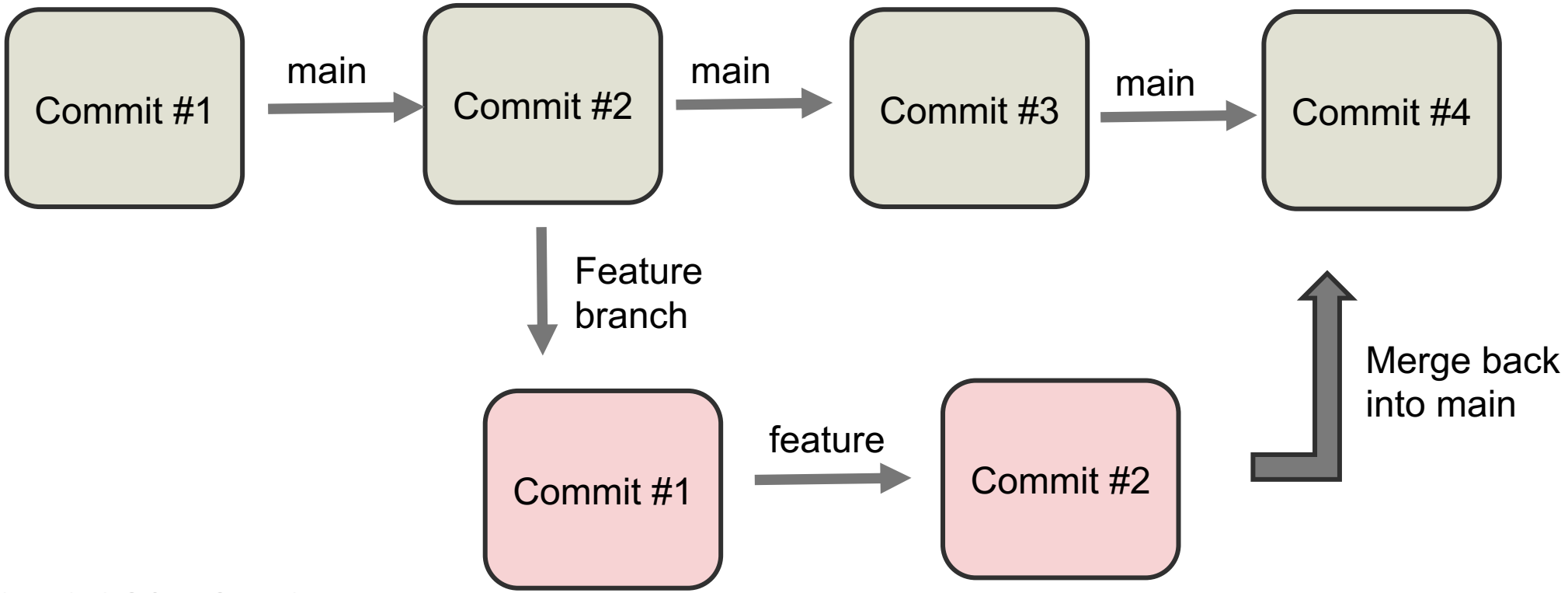
A branch is a separate line of development within a repository. It lets you work on new features, fixes, or experiments without affecting the main codebase.

- Develop features independently and merge them when ready
- Enable collaboration without disrupting others' work
- Test or prototype safely before pushing to production

BRANCHING

Typical Workflow:

- `main` (or `master`) — stable production-ready code
- `feature/*` — new functionality under development
- `fix/*` — bug fixes or patches

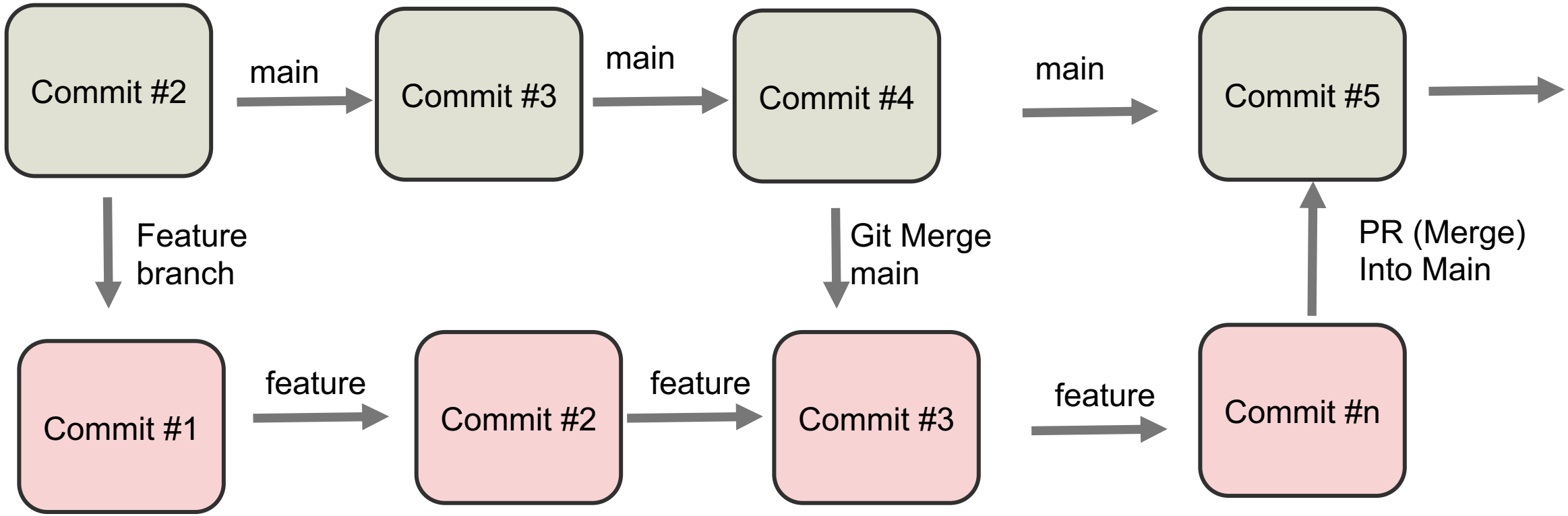


BRANCHING AND MERGING

Typical Workflow:

What if there are changes to main while you are working in your feature?

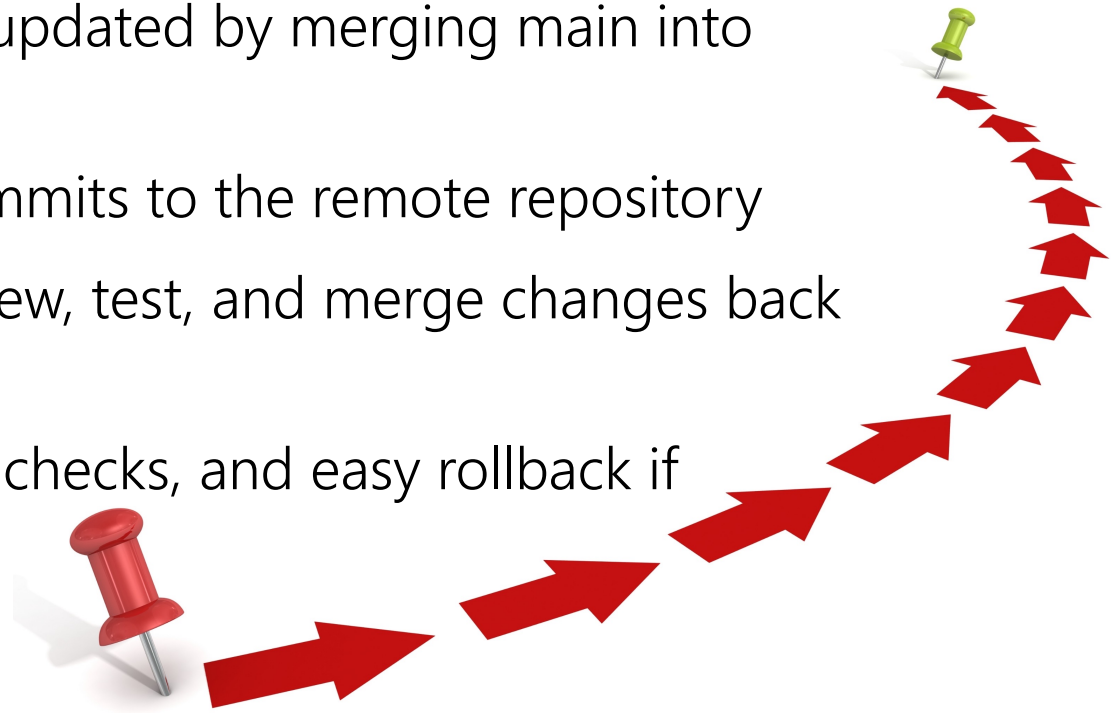
Merge from Main Branch into Feature Branch Often (Pull Often)



GITHUB WORKFLOW

- **Fetch / Pull** – Get the latest updates from the remote repository
- **Create a Branch** – Start a new branch from main for your feature or fix
- **Work on the Branch** – Make and commit your changes locally
- **Merge from Main Often** – Stay updated by merging main into your branch regularly
- **Push to Branch** – Send your commits to the remote repository
- **Open a Pull Request (PR)** – Review, test, and merge changes back into main

PRs allow peer review, automated checks, and easy rollback if needed

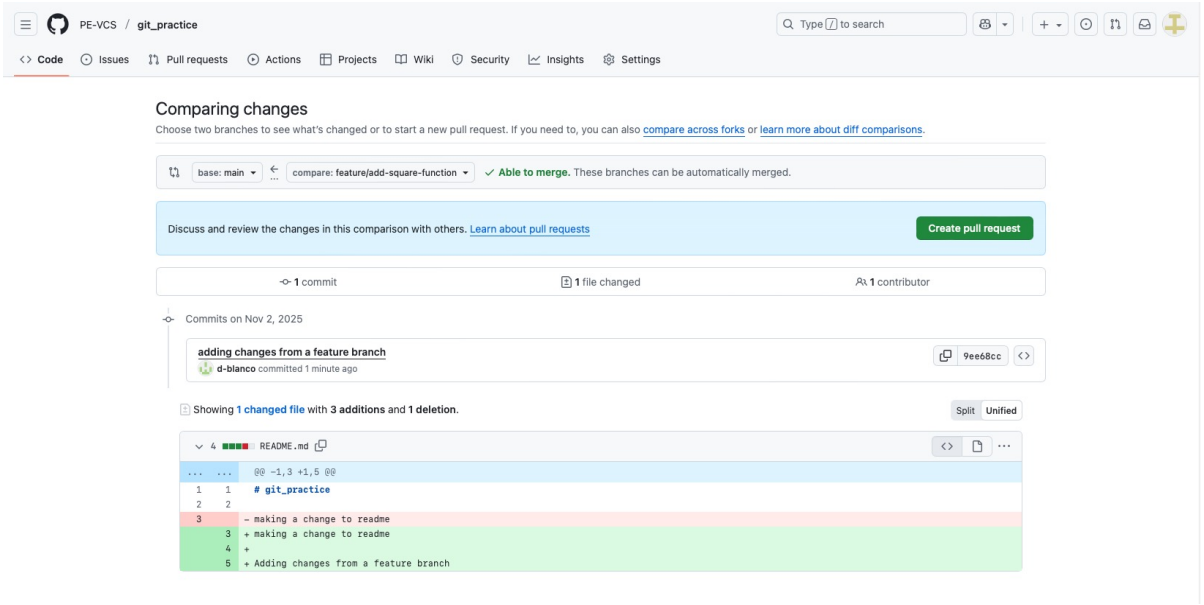


GIT PULL REQUESTS

A Pull Request is a formal way to propose changes from one branch into another — most commonly from a feature branch into main. It lets developers share their work, review differences, and discuss improvements before merging.

Beyond code review, Pull Requests serve as checkpoints in the development process. They can trigger automated tests, security scans, and deployment previews, helping maintain code quality and stability.

GIT PULL REQUESTS



If a Pull Request "**Can't be merged**", it means your feature branch is missing updates from the main branch.

To fix it:

- Run git fetch and merge main into your branch locally
- Resolve any merge conflicts that appear
- Commit and push the updated branch back to GitHub
- The existing PR will update automatically — you don't need to re-create it

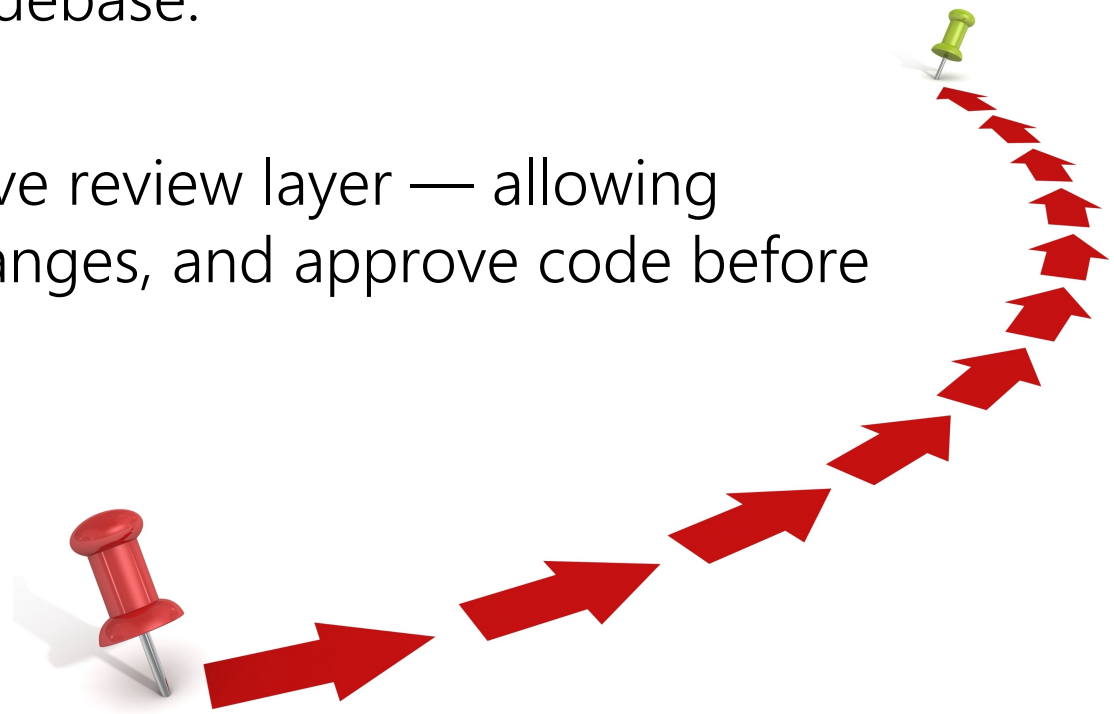
Once your branch is in sync with main, GitHub will show "Able to merge", allowing the PR to be completed.

REFLECTION POINTS

Pull and merge often to keep your branch up to date and prevent conflicts.

Branches provide freedom to experiment and develop safely without disrupting the main codebase.

Pull Requests add a collaborative review layer — allowing peers to comment, suggest changes, and approve code before it’s merged into main.



LAB 2.2 – BRANCHING / PR

LAB INTRODUCTION

In this lab, you will learn how to use branches to work independently, merge changes safely, and create Pull Requestsfor collaboration.

You'll practice updating your branch, resolving merge conflicts, and submitting code for review before it's added to the main project.

POP QUIZ: BRANCHES

What commands on the CLI can create a branch named feature1?

A. git branch feature1
B. git checkout feature1
C. git checkout -b feature1
D. git switch feature1

5 minutes

POP QUIZ: BRANCHES

What commands on the CLI can create a branch named feature1?

A. git branch feature1
B. git checkout feature1
C. git checkout -b feature1
D. git switch feature1

5 minutes

POP QUIZ: BRANCHES

What command is used to update your current branch from another one?

A. git pull
B. git checkout
C. git branch
D. git merge

POP QUIZ: BRANCHES

What command is used to update your current branch from another one?

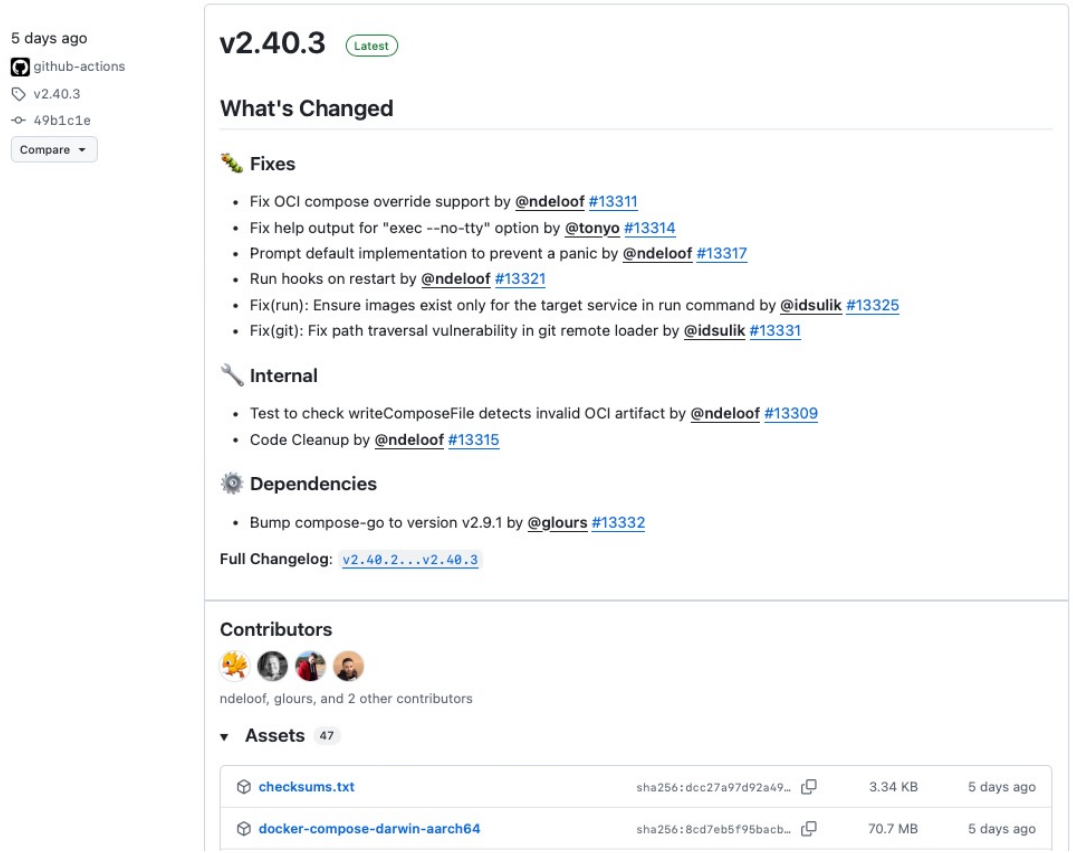
A. git pull
B. git checkout
C. git branch
D. git merge

VCS

Tagging and Releases

2

TAGGING AND RELEASING



Tags mark specific points in a repository's history — often used to identify versions of your project, such as v1.0.0 or v2.1.3.
They make it easy to reference stable releases, roll back to previous versions, or track what changed between releases.

A **release** builds on a tag by packaging your code, notes, and assets for distribution. In GitHub, releases are created from tags and can include release notes or binaries for others to download.

TAGGING

Tags are labels attached to specific commits that represent key points in a project's history — such as a release, milestone, or deployment.

They're useful for tracking progress, rolling back to previous versions, and identifying stable builds.

- Mark important versions (v1.0.0, v1.1.0, etc.)
- Simplify debugging and rollback
- Help teams communicate which code is in production

TAGGING SEM-VER

Semantic versioning follows the format:

MAJOR.MINOR.PATCH

- **MAJOR** → Breaking changes (e.g., 1.0.0 → 2.0.0)
- **MINOR** → New features, backward compatible (1.1.0)
- **PATCH** → Bug fixes, small improvements (1.1.1)

Example Workflow:

- You finish a new feature — tag v1.2.0
- Fix a bug later — tag v1.2.1
- Major redesign — tag v2.0.0

Following SemVer helps teams and tools automatically understand the impact of each release.

RELEASES

Once you've created and pushed a tag, you can turn it into an official release on GitHub.

A release packages your code, notes, and version history so others can easily download or deploy it.

In your repository, go to Releases → "Draft a new release"
Select an existing tag (or create one during the process)
Add release notes, attach optional build files, and click Publish Release

GitHub releases make your tagged versions easy to share

LAB 2.3 – TAGGING / RELEASING

LAB INTRODUCTION

In this lab, you'll practice creating Git tags and GitHub releases to mark important points in your project's history.

You'll create and push tags, add release notes, and understand how teams use version numbers to track and share updates.

POP QUIZ: TAGS

Which of the following best describes a Git tag?

A. A temporary branch used for testing
B. A label pointing to a specific commit in history
C. A merge request between branches
D. A type of Git configuration file

POP QUIZ: TAGS

Which of the following best describes a Git tag?

A. A temporary branch used for testing
B. A label pointing to a specific commit in history
C. A merge request between branches
D. A type of Git configuration file

POP QUIZ: TAGS

According to semantic versioning, which change should increase the MAJOR version number?

A. Adding a backward-compatible feature
B. Fixing a small bug
C. Introducing changes that break existing functionality
D. Updating documentation only

POP QUIZ: TAGS

According to semantic versioning, which change should increase the MAJOR version number?

A. Adding a backward-compatible feature
B. Fixing a small bug
C. Introducing changes that break existing functionality
D. Updating documentation only

VCS

Collaborating

2

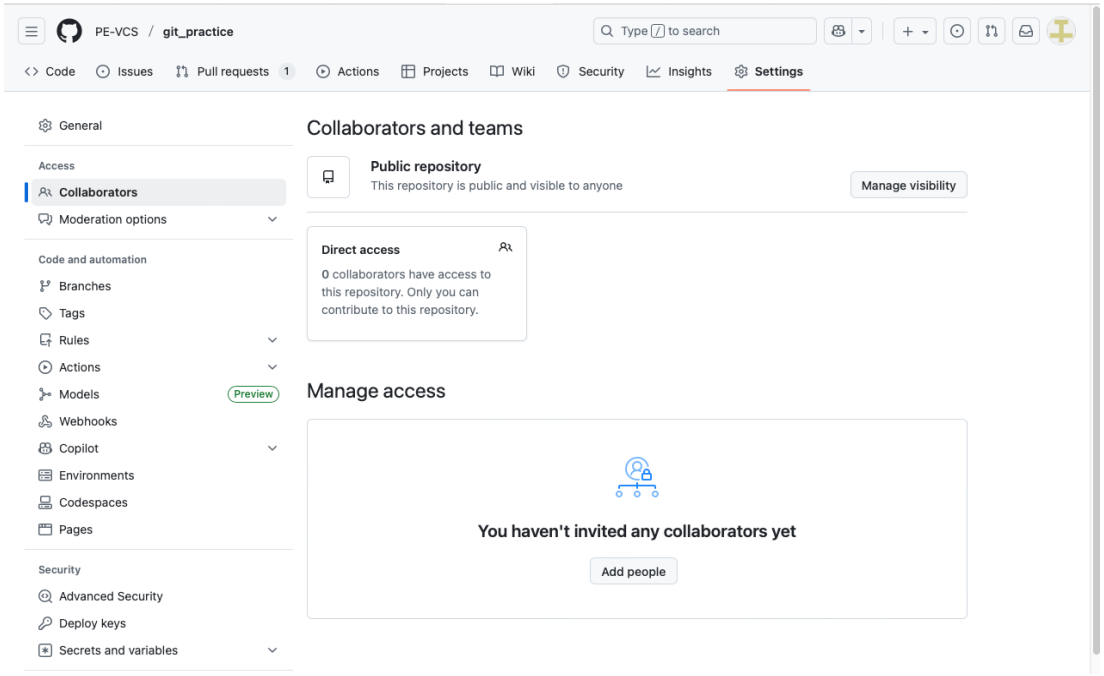
COLLABORATION

Modern software development is a team effort, and GitHub provides powerful tools to help teams work together efficiently.

Collaboration in Git involves sharing repositories, contributing through branches, and reviewing each other's code through Pull Requests.

You'll learn how to add collaborators, manage permissions, and use GitHub's built-in features — like issues, discussions, and code reviews — to keep projects organized and productive.

COLLABORATORS



GitHub allows you to invite collaborators to your repository so others can contribute directly to your project.Each collaborator can be assigned different levels of access depending on their role.

Read: Can view the repository and open issues

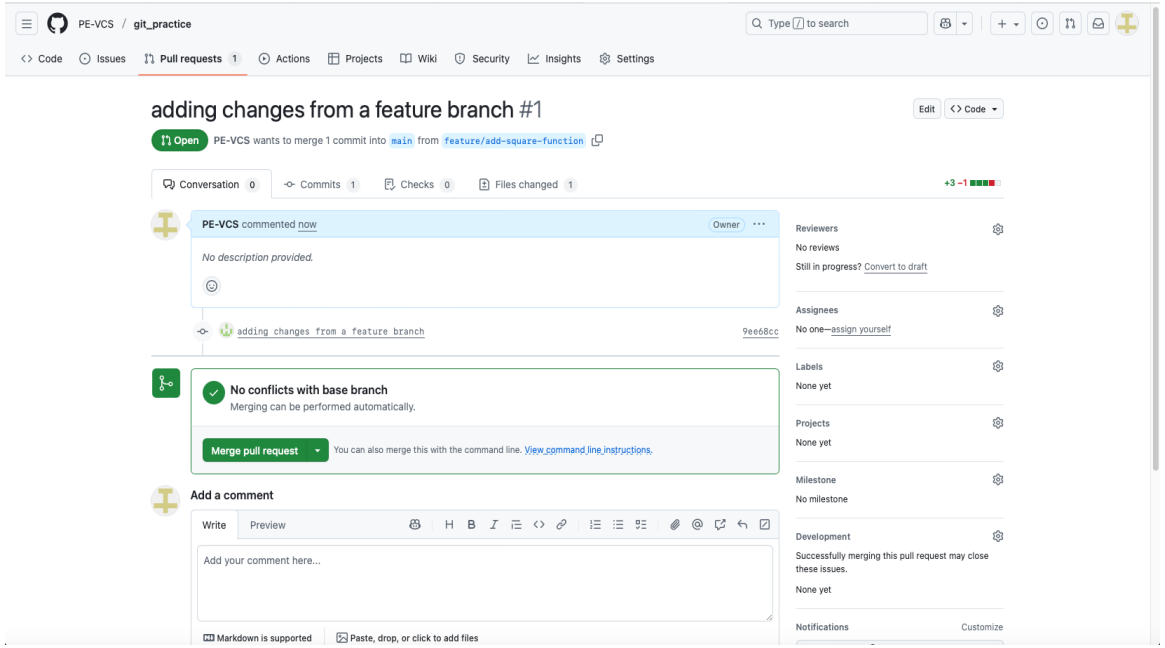
Triage: Can manage issues and pull requests without writing to the repo

Write: Can push commits and manage branches

Maintain: Can manage settings and workflows, but not delete the repo

Admin: Full access — can manage collaborators, settings, and repository visibility

COLLABORATORS



Pull Requests are the main way collaborators propose changes and get them reviewed before merging into main.They help maintain code quality and ensure that all changes are discussed and approved.

- Push changes to a feature branch
- Open a Pull Request to propose merging into main
- Team members review, comment, and approve the changes
- Once approved, the PR can be merged into the main codebase

COLLABORATORS

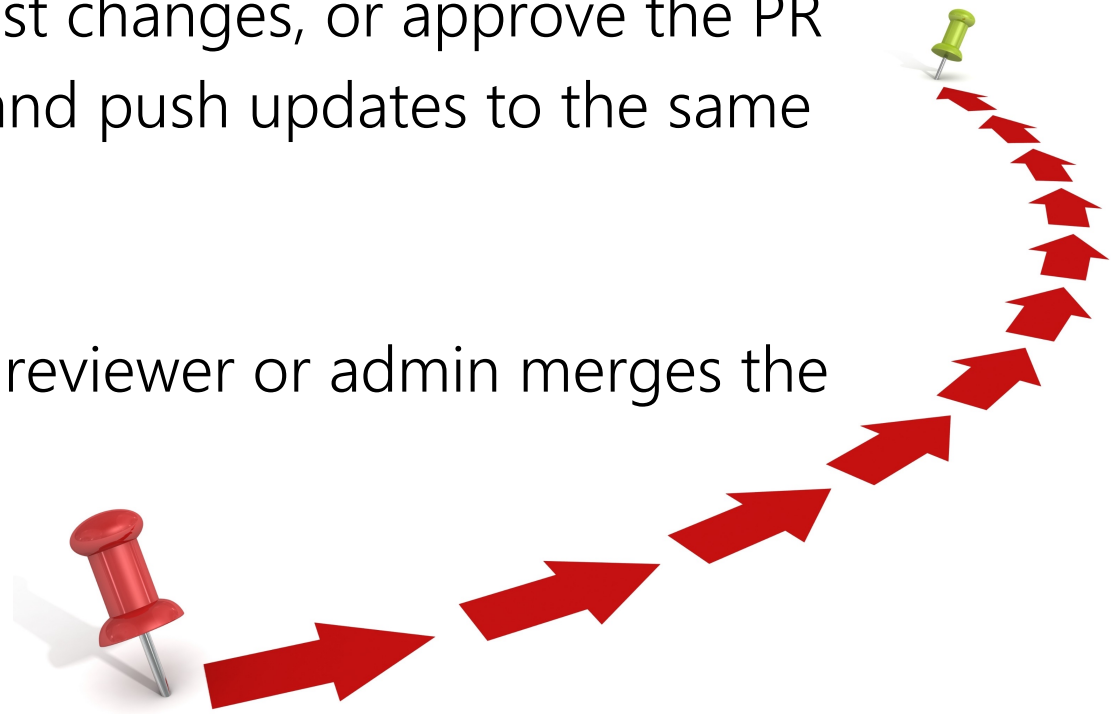
GitHub allows assigning specific reviewers to a Pull Request for focused feedback and accountability.

Assign one or more reviewers when opening or updating a PR

Reviewers can comment, request changes, or approve the PR

The author can then fix issues and push updates to the same branch

Once everything looks good, a reviewer or admin merges the PR



LAB 2.4 – COLLABORATING

LAB INTRODUCTION

In this lab, you'll work with a partner to practice real-world GitHub collaboration — adding collaborators, cloning each other's repositories, creating branches, and submitting Pull Requests for review and approval.

You'll experience the full teamwork workflow of reviewing, merging, and syncing code, just like on a professional development team.

POP QUIZ: GITHUB

What must you do before a teammate can push changes to your GitHub repository?

A. Make the repository public
B. Add them as a collaborator
C. Share your Git credentials
D. Fork the repository

POP QUIZ: GITHUB

What must you do before a teammate can push changes to your GitHub repository?

A. Make the repository public
B. Add them as a collaborator
C. Share your Git credentials
D. Fork the repository

POP QUIZ: GITHUB

What is the command to download a GitHub repository?

A. git pull
B. git clone
C. git get repo
D. git restore

POP QUIZ: GITHUB

What is the main purpose of creating a Pull Request in a collaborative workflow?

A. To back up your code to GitHub
B. To merge changes automatically without review
C. To allow others to review, discuss, and approve your changes before merging
D. To clone the repository into your local machine

POP QUIZ: GITHUB

What is the main purpose of creating a Pull Request in a collaborative workflow?

A. To back up your code to GitHub
B. To merge changes automatically without review
C. To allow others to review, discuss, and approve your changes before merging
D. To clone the repository into your local machine

VCS

Repo Settings

2

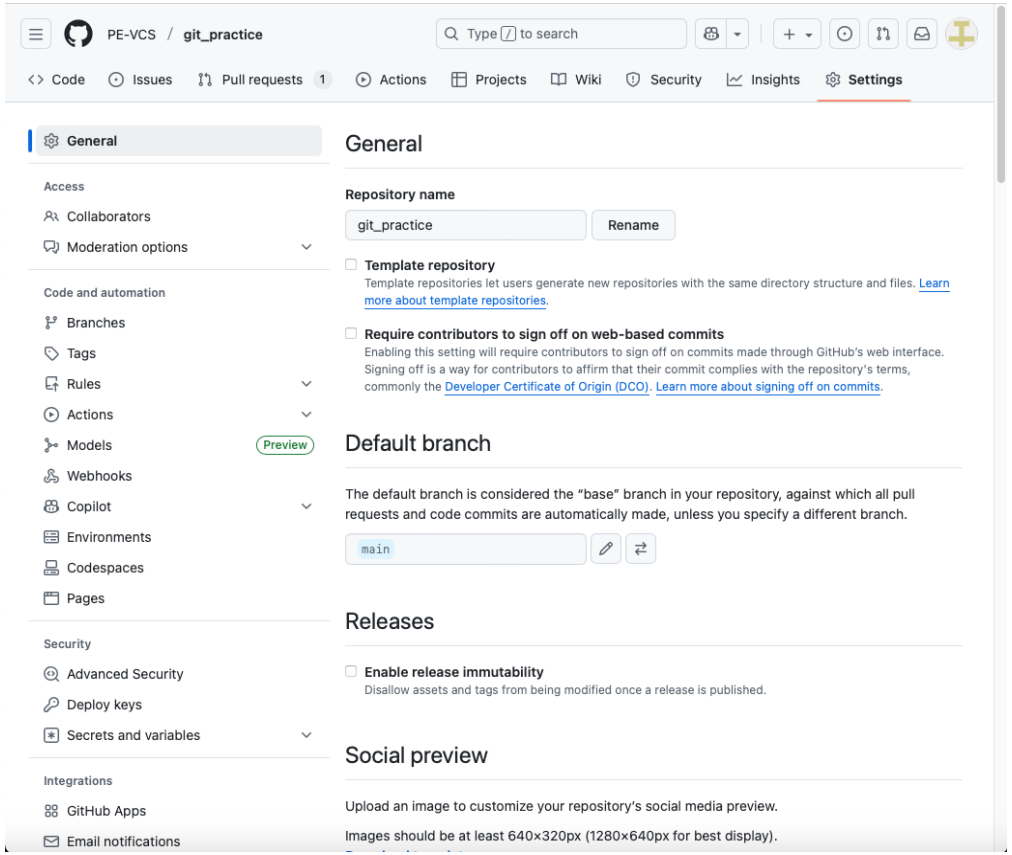
REPOSITORY SETTINGS

Every GitHub repository includes a Settings section that lets you control how your project behaves and who can access it. From here, you can manage repository details such as its name, description, visibility, and features like Issues, Wikis, and Discussions.

Beyond the basics, Settings also provides tools for automation, security, and customization. You can configure branch rules, enable GitHub Pages for website hosting, manage Actions for CI/CD, and securely store secrets like API keys.

Understanding what each section controls helps you maintain a well-organized, secure, and professional repository.

REPOSITORY SETTINGS

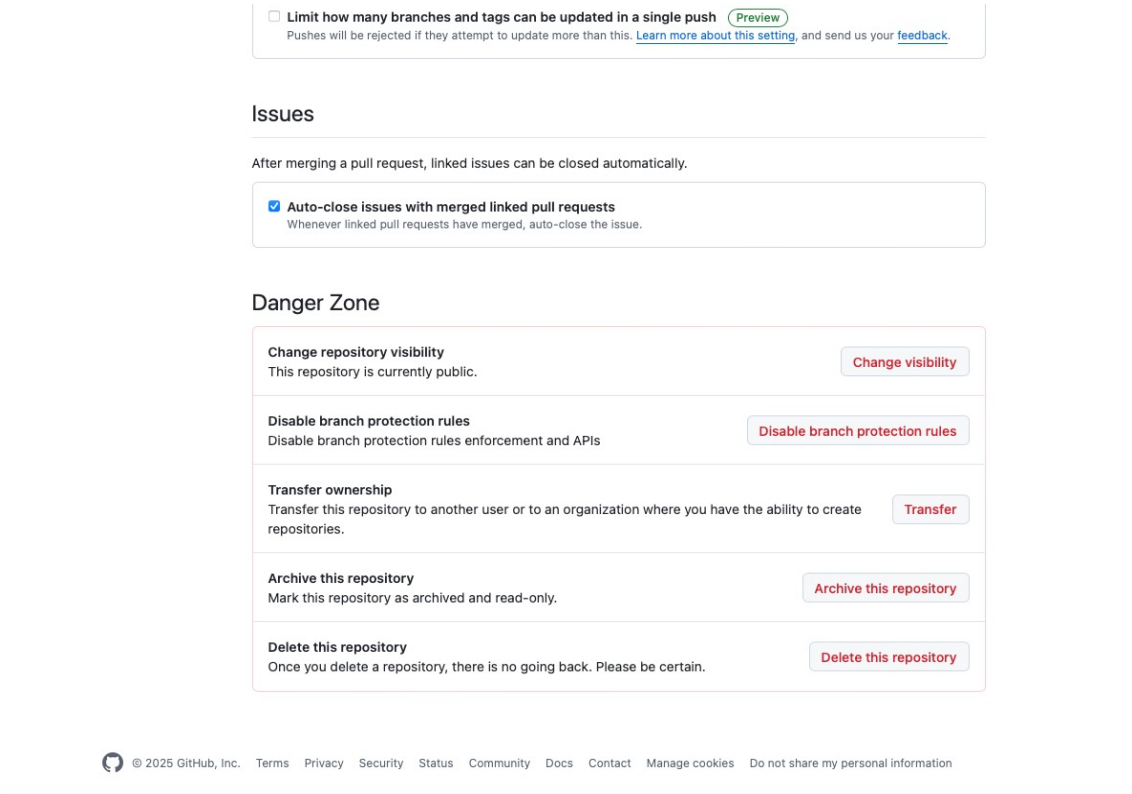


Repository Basics

Have a look at the settings. You can change the repository name, default branch or enable features like issues and wiki pages.

On the left notice different tabs for different pages with settings. For example, Secrets and Variables would be a place for adding

REPOSITORY SETTINGS



🚨 Danger Zone

- Archive → read-only repository
- Transfer ownership → move repo to another account
- Delete repository → permanent removal

⚠️ Avoid clicking these during the lab!

REPOSITORY SETTINGS: PAGES

GitHub Pages allows you to turn your repository into a live website. It's ideal for documentation, personal portfolios, or demo projects.

You simply select a branch or folder (often /docs) and GitHub automatically builds and hosts the site for free.

This feature helps developers showcase work, publish project documentation, or share tutorials — all version-controlled within the same repository.

REPOSITORY SETTINGS: WIKIS

A **GitHub Wiki** provides a separate, editable space for **documentation and guides** related to your project. It's perfect for things like setup instructions, FAQs, internal notes, or developer guides.

Unlike a README file, a wiki can have multiple pages, easy navigation, and collaborative editing — making it a great tool for maintaining living documentation that grows alongside your project.

REPOSITORY SETTINGS: ISSUES

Issues are GitHub's built-in bug and task tracking system. They let teams log bugs, propose new features, and track progress on ongoing work.

Each issue can include labels, milestones, assignees, and comments, turning it into a lightweight collaboration hub for planning and problem-solving within your repository.

LAB 2.4 – REPO SETTINGS

LAB INTRODUCTION

In this lab, you'll explore the different configuration options available in a GitHub repository and learn what each section controls.

You'll practice enabling and using features like Wikis, Issues, and Pages, and even create a test issue to simulate tracking an incident or enhancement request — just like in a real project environment.

POP QUIZ: GITHUB

Where can you rename your GitHub repository and enable features like Issues or Wikis?

A. The Code tab
B. The Collaborators page
C. The Settings tab
D. The Pull Requests tab

POP QUIZ: GITHUB

Where can you rename your GitHub repository and enable features like Issues or Wikis?

A. The Code tab
B. The Collaborators page
C. The Settings tab
D. The Pull Requests tab

POP QUIZ: GITHUB

What is the main purpose of GitHub Pages?

A. To host automated workflows
B. To store encrypted secrets
C. To host a website or documentation directly from your repository
D. To manage repository access and permissions

POP QUIZ: GITHUB

What is the main purpose of GitHub Pages?

A. To host automated workflows
B. To store encrypted secrets
C. To host a website or documentation directly from your repository
D. To manage repository access and permissions

VCS DAY 2 COMPLETE

Congratulations — you've learned how to use Visual Studio Code with Git and GitHub to manage your code through a complete version control workflow. You practiced working with repositories, creating branches, merging changes, and submitting Pull Requests to collaborate effectively.

You also learned how to tag and release your code, explore key repository settings, and enable features like Wikis, Issues, and Pages. These skills form the foundation of real-world development workflows — you now have the tools to manage projects confidently, both individually and as part of a team.