Components of the GitHub flow

In this section, we're reviewing the following components of the GitHub flow:

- Branches
- Commits
- Pull Requests
- The GitHub Flow

What are branches?

In the last section, we created a new file and a new branch in your repositories.

Branches are an essential part of the GitHub experience because they're where we can make changes without affecting the entire project we're working on.

Key Features of Branches:

- Your branch is a safe place to experiment with new features or fixes.
- If you make a mistake, you can revert your changes or push more changes to fix the mistake.
- Your changes won't update on the default branch until you merge your branch.

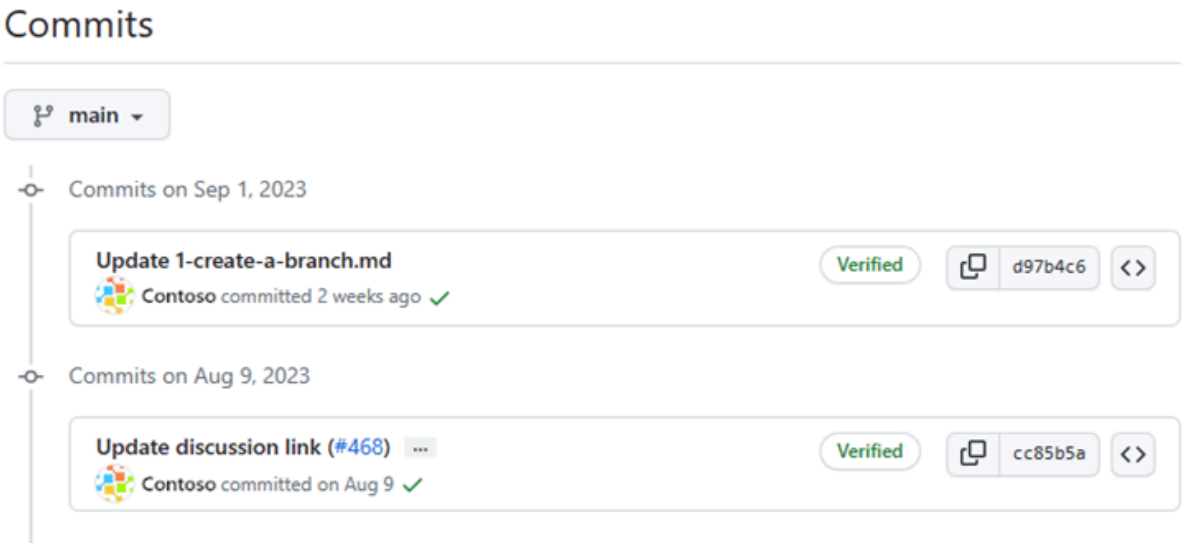
Note: You can create a new branch and check it out by using git in a terminal. The command would be: `git checkout -b newBranchName`

What are commits?

In the previous unit, you added a new file into the repository by pushing a commit. Let's briefly review what commits are.

Key Features of Commits:

- A commit is a change to one or more files on a branch.
- Every time a commit is created, it's assigned a unique ID and tracked along with the time and contributor.
- Commits provide a clear audit trail for anyone reviewing the history of a file or linked item, such as an issue or pull request.



File States in a Git Repository

Within a Git repository, a file can exist in several valid states as it goes through the version control process. The primary states for a file in a Git repository are **Untracked** and **Tracked**.

Untracked: An initial state of a file when it isn't yet part of the Git repository. Git is unaware of its existence.

Tracked: A tracked file is one that Git is actively monitoring. It can be in one of the following substates:

- **Unmodified**: The file is tracked, but it hasn't been modified since the last commit.
- **Modified**: The file has been changed since the last commit, but these changes aren't yet staged for the next commit.
- **Staged**: The file has been modified, and the changes have been added to the staging area (also known as the index). These changes are ready to be committed.
- **Committed**: The file is in the repository's database. It represents the latest committed version of the file.

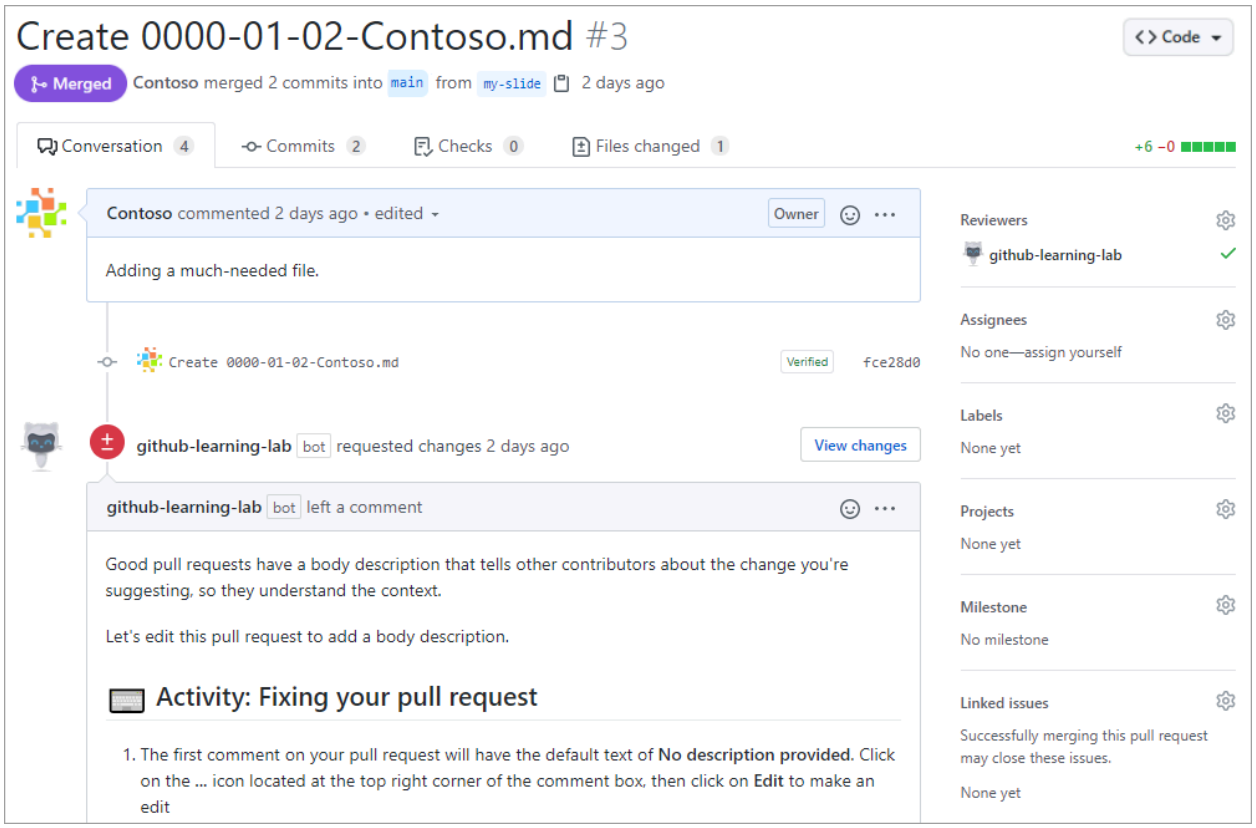
These states and substates are important for collaborating with your team to know where each and every commit is in the process of your project.

What are Pull Requests (PR)?

A pull request is the mechanism used to signal that the commits from one branch are ready to be merged into another branch.

Key Features of Pull Requests:

- The team member submitting the pull request asks one or more reviewers to verify the code and approve the merge.
- These reviewers have the opportunity to comment on changes, add their own, or use the pull request itself for further discussion.
- Once the changes have been approved (if required), the pull request's source branch (the compare branch) is merged into the base branch.

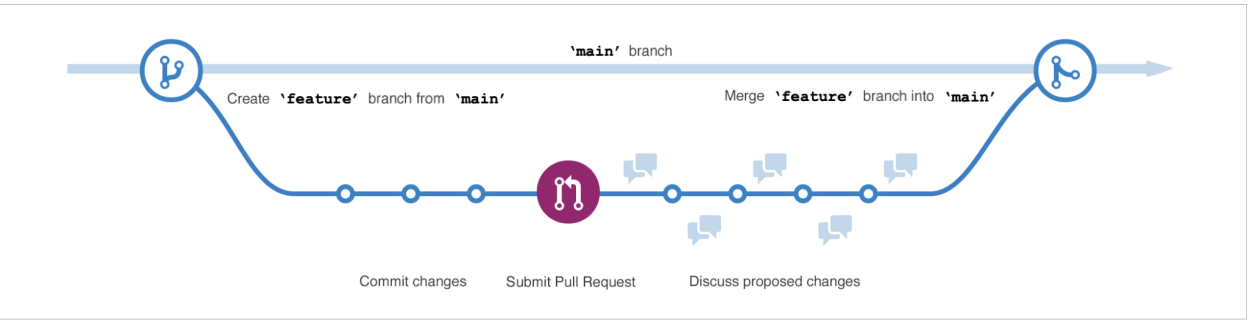


Pull requests are a crucial part of the collaborative workflow in Git, ensuring that changes are reviewed and approved before being integrated into the main codebase.

Let’s review the GitHub flow.

The GitHub flow

The GitHub flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly. You can test new ideas and collaboration with your team by using branching, pull requests and merging.



Here's a step-by-step overview of the GitHub flow:

1. Create a branch:
   - Start by creating a new branch from the main branch (usually `main` or `master`). This branch will serve as a safe space to make changes without affecting the main codebase.
   - Example: `` `git checkout -b feature/new-feature` ``

2. Make changes
   - Work on your new feature or fix in the new branch. Make the necessary changes to the files.
   - Example: Edit files, add new files, etc.

3. Stage and commit changes:
   - Stage the changes you've made by adding them to the staging area.
   - Commit the staged changes with a meaningful commit message.
   - Example:
     ```sh
     git add .
     git commit -m "Add new feature"
     ```

4. Push changes to GitHub
   - Push your branch to the remote repository on GitHub.
   - Example: `git push origin feature/new-feature`

5. Create a Pull Request (PR)
    - On GitHub, create a (PR) from your feature branch to the main branch.
    - Provide a clear title and description for your PR, explaining the changes and why they are necessary.

6. Review and Discuss
    - Team members review the changes in the pull request.
    - They can leave comments, suggest changes, or add their own commits.
    - Example: Reviewers can comment on specific lines of code or request changes.

7. Merge the Pull Request
    - Once the changes are reviewed & approved, merge the PR into the main branch.
    - Example: Click the "Merge Pull Request" button on GitHub.

8. Deploy
    - After merging, the changes can be deployed to production or staging environments as part of your CI/CD pipeline.

9. Close the Branch
    - Optionally, delete the feature branch after merging to keep the repository clean.
    - Example: `` `git branch -d feature/new-feature` ``

By following these steps, the GitHub flow ensures that changes are reviewed and tested before being merged into the main codebase, maintaining the quality and stability of your project.