Introduction and Context

GitHub extends beyond source control into full-fledged collaboration, allowing teams to plan work, document projects, engage communities, and enforce permissions-all in one platform. By the end of this tutorial, you'll know how to track issues with labels, organize work on a Kanban board, plan sprints via Milestones, author project documentation in a Wiki, host community Q&A in Discussions, and configure team roles for secure access.

Case Study Overview

Problem Statement
WeatherNow, a startup delivering hyperlocal forecasts, faces fragmented coordination: bugs slip through, feature requests get lost, documentation lags, and community questions scatter across emails. To scale, the team must centralize planning and communication on GitHub. Success means every bug or feature request is an Issue with labels, tasks flow smoothly across columns on a Project board, sprints conclude on schedule via Milestones, comprehensive API docs live in a Wiki, users find answers in Discussions, and only the right people can push or merge code.

Learning Objectives

- Create and manage GitHub Issues with labels for bug tracking and feature requests
- Set up a Kanban-style Project board to visualize workflow
- Define and use Milestones for sprint planning
- Build a project Wiki for documentation
- Host and moderate GitHub Discussions for community engagement
- Configure Organization teams, roles, and repository permissions

Concepts Explained with Analogies

GitHub Issues
Analogy: Sticky notes on a wall, each noting a task or problem.
Technical: Issues track bugs, enhancements, and tasks; they can be labeled, assigned, and linked to Milestones

Labels
Analogy: Colored tags on sticky notes indicating priority or type.
Technical: Labels categorize and prioritize Issues (e.g., “bug,” “enhancement,” “high priority”)

Project Boards
Analogy: A digital Kanban board where cards (Issues) move through columns like “To Do,” “In Progress,” and “Done.”
Technical: GitHub Projects offer a board view to manage Issues and pull requests via columns and automation

Milestones
Analogy: A sprint deadline on a project timeline, grouping related sticky notes to be completed together.
Technical: Milestones bundle Issues and pull requests, tracking progress towards a release or sprint goal.

Wikis
Analogy: A shared knowledge book where anyone on the team can add or update chapters.
Technical: Each repository’s Wiki hosts long-form documentation in Markdown or other formats

Discussions
Analogy: A town-hall forum where community members ask questions and share ideas.
Technical: Discussions support threaded Q&A, announcements, and idea boards, separate from Issues and PRs

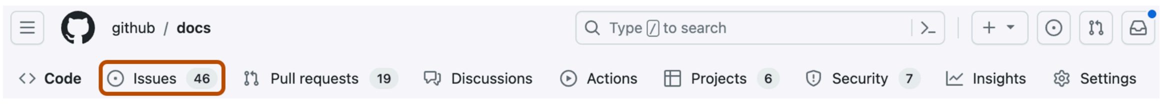
Team Management (Roles & Permissions)
Analogy: Assigning job titles-maintainers, contributors, viewers-to control who can edit what.
Technical: Organization roles (Owner, Member) and repository permissions (Read, Write, Admin) regulate access to code, Issues, and settings

Step-by-Step Guided Walkthrough

Step 1: Log and Label Issues

1. On GitHub, navigate to the main page of the repository.

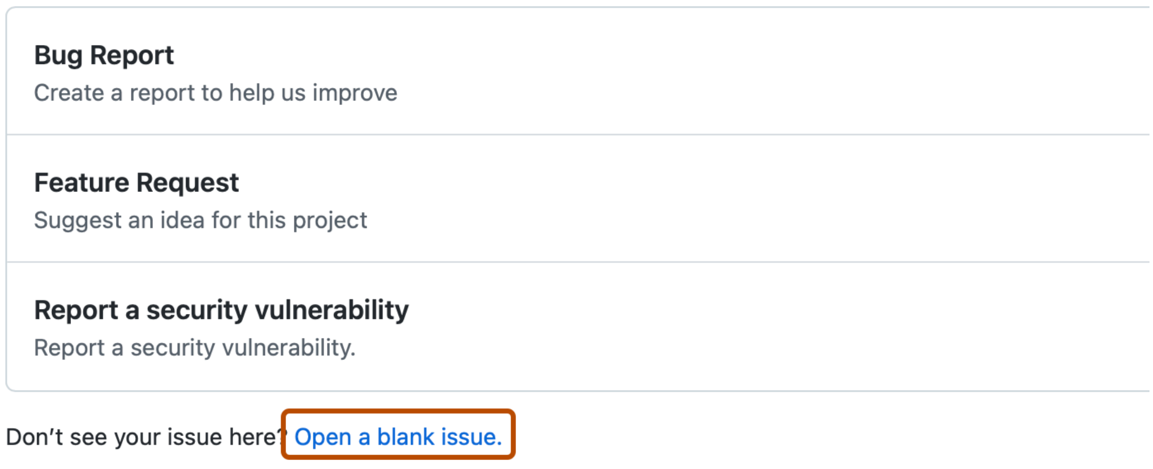
2. Under your repository name, click **Issues**.

3. Click **New issue**.

4. If your repository uses issue templates, next to the type of issue you'd like to open, click **Get started**.

   If the type of issue you'd like to open isn't included in the available options, click **Open a blank issue**.

   
   Bug Report
Create a report to help us improve
Feature Request
Suggest an idea for this project
Report a security vulnerability
Report a security vulnerability.
Don't see your issue here? Open a blank issue.

5. In the "Title" field, type a title for your issue.

6. In the comment body field, type a description of your issue.

7. If you're a project maintainer, you can assign the issue to someone, add it to a project, associate it with a milestone, set the issue type, or apply a label.

8. When you're finished, click **Submit new issue**.

Checkpoint: How do labels and assignees improve issue triage?

Step 2: Create a Kanban Project Board

1. In the repo, click **Projects** → **New project** → **Board**
2. Name it “Sprint 1” and click **Create project**.
3. Ensure columns: **To do**, **In progress**, **Done**.
4. Click **+ Add cards**, select your open Issues, and drag them into **To do**.

Checkpoint: Why visualize workflow on a board?

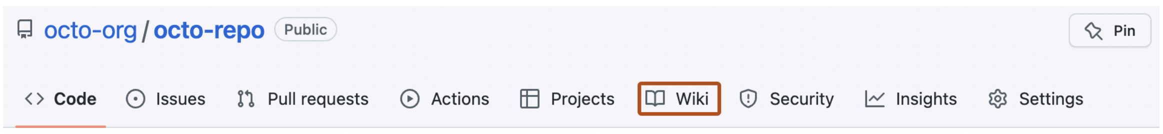
Step 3: Define a Milestone for Sprint Planning

1. Go to **Issues** → **Milestones** → **New milestone**.
2. Title: “Sprint 1”
3. Set due date one week ahead and description “Complete core UI and bug fixes.”
4. Assign existing Issues to this Milestone via each Issue’s sidebar

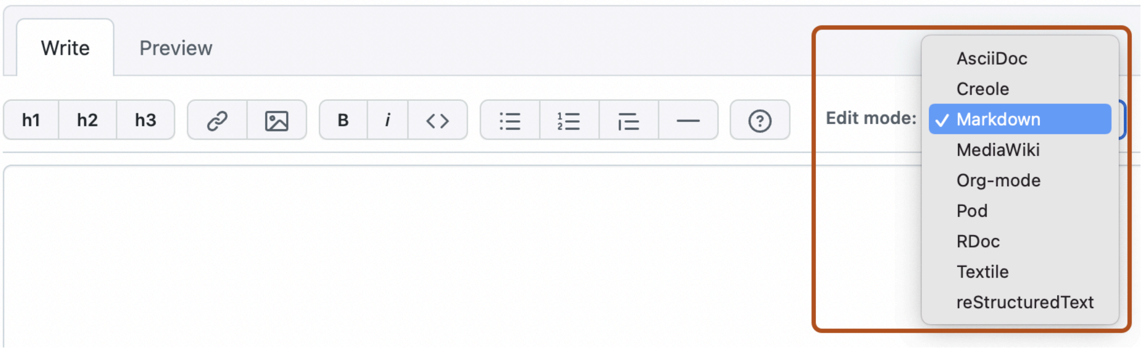
Checkpoint: How do Milestones help track sprint progress?

Step 4: Author Documentation in the Wiki

1. On GitHub, navigate to the main page of the repository.
2. Under your repository name, click **Wiki**.



3. In the upper-right corner of the page, click **New Page**.
4. Optionally, to write in a format other than Markdown, use the “Edit mode” dropdown to choose a different format.



5. Use the text editor to add your page's content.

6. In the "Edit message" field, type a commit message describing the new file you're adding.

7. To commit your changes to the wiki, click **Save Page**.

Step 5: Engage Users with Discussions

1. Enable **Discussions** in the repository settings.

2. In **Discussions**, click **New discussion**, choose category **Q&A**, and ask, "How to integrate WeatherNow API into React?"

3. Encourage community responses and mark the answer that solves the question

Checkpoint: How do Discussions complement Issues?

Step 6: Configure Team Roles and Permissions

1. In your Organization, go to **Teams** → **New team**, name it "Frontend Devs."

2. Add members and give **Write** access to the WeatherNow repo.

3. For senior engineers, assign **Admin** role to manage settings; for interns, grant **Read** only

Checkpoint: Why restrict certain permissions to admins?

Best Practices and Tips

- Use **clear, descriptive Issue titles** and link to code or screenshots.
- Automate board columns by setting **automation rules** (e.g., move card to In progress when a PR is opened).
- Close Issues automatically when PRs referencing them are merged (`Fixes #123`).
- Keep the Wiki **up to date**; treat it as living documentation.
- Encourage community members to ask in **Discussions** before filing Issues to reduce noise.
- Enforce **branch protection rules** requiring reviews and passing CI checks before merges.

Real-World Application and Extension

- Integrate **GitHub Actions** to automate labeling, board updates, and Milestone progress tracking.
- Use **cross-repository Milestones** via API scripting for microservices spanning multiple repos
- Leverage **Discussion Insights** for community sentiment and feature requests.