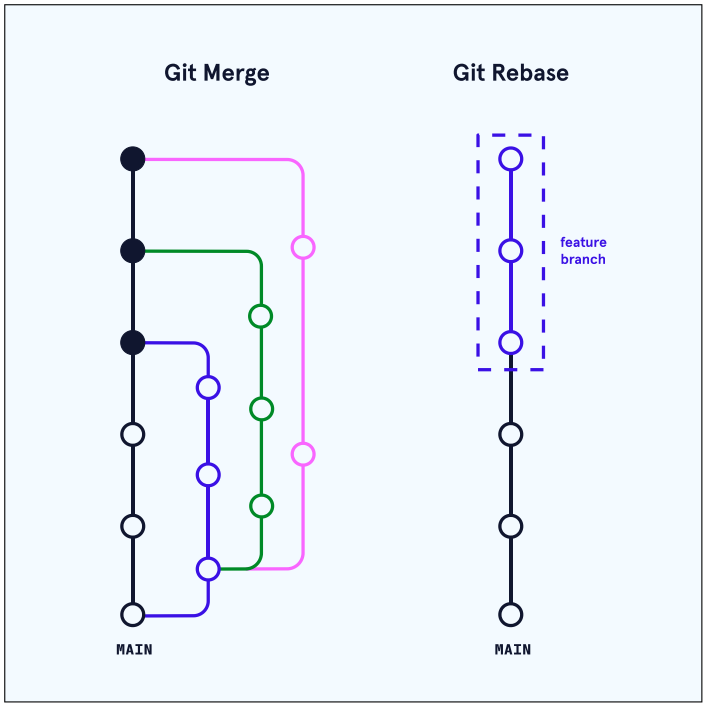
Best Practices for GitHub Repositories

Git Rebase

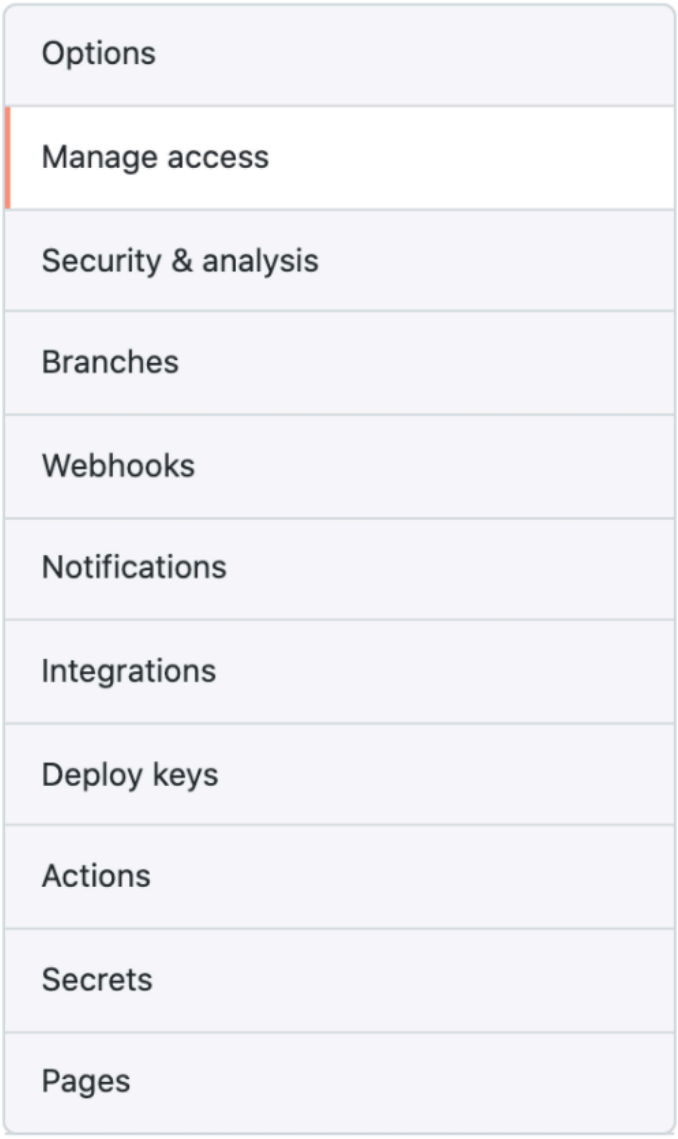
Rebasing is moving the base of a branch onto a different position. It is used to simplify the Git history of a repository. The advantage of using Git rebase over Git merge is that we can create a linear history of changes and reduce merge commits, but rebase should be done carefully.



GitHub Repository Settings

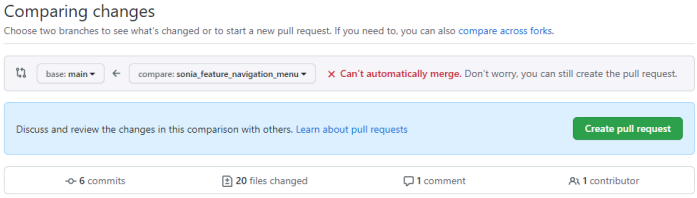
The settings page of a GitHub repository allows us to customize and change many options that can take our repository to the next level.

- Options: enable features like Wikis and Discussions, change basic repository information including name and visibility, and delete the repository.
- Security & Analysis: enable/disable a variety of security features.
- Branches: set default branch and create branch protection rules.
- Webhooks: use webhooks to get notifications when certain events happen.
- Notifications: receive email notifications when push events are triggered.
- Integrations: applications integrated with our repository will appear here.
- Deploy Keys: use SSH keys to grant servers access to a repository for deployment.
- Actions: change permissions for GitHub Actions.
- Secrets: encrypted environment variables that can be used in Actions.
- Pages: host simple web pages straight from the repository.



Pull Requests on GitHub

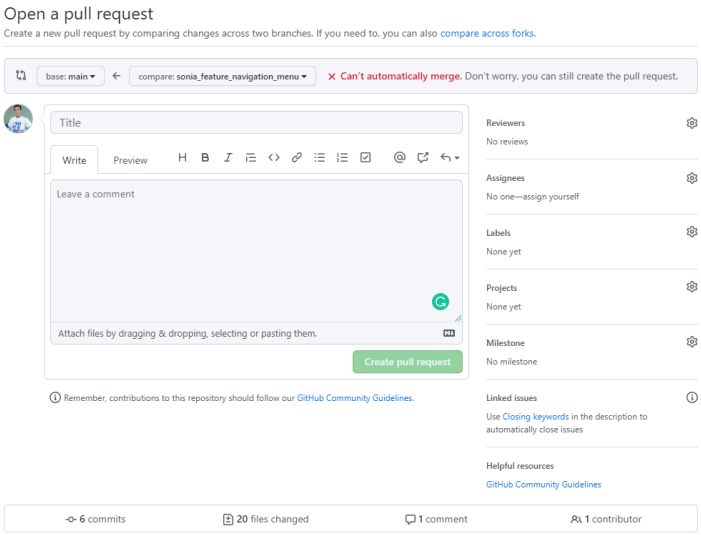
A pull request is a feature of GitHub and other source code management tools that allow a repository's collaborators to review and give feedback on proposed code changes before they are accepted and merged to another branch, usually the main branch. Each pull request creates a discussion forum that can be used by reviewers and authors alike to highlight or add comments to a single line of code or chunk of code, add videos or images, etc.
Going through the pull request process can increase group knowledge, improve product quality, and develop professional skills through group critique.



Writing a Good Pull Request

Creating a good pull request can increase the chances of having your changes being accepted by the collaborators. Follow the steps below to properly structure your pull request, which can make it easier for reviewers to understand your proposed changes and receive their feedback quicker:

- Concisely explain the purpose of the pull request in the title
- Put the thought process behind code changes and the options you have considered in the description
    - Including screenshots, GIFs, and videos can help others visualize your changes
- Use and follow the pull request templates and guidelines, if provided
- Make commit messages clear
- Use comments both in your code and in the discussion
- Keep pull requests small



Using .gitignore in a GitHub Repository

When pushing our project to GitHub, there are often files we do not want to be shared with others. A **.gitignore** file is used to specify which files or folders we want Git to ignore when staging. Follow the steps below to include this file in your GitHub project, which can result in cleaner staging areas and prevent sensitive information from accidentally being pushed.

1. Create a text file named **.gitignore**
2. Write names of files or folders to ignore

   - Each line corresponds to a file, directory, or pattern
   - Be sure to include `/` at the end of directory names
   - Use globbing patterns to handle special cases

1. Ignored files won't be staged but be sure to stage and commit the **.gitignore**!

```
# An example Java gitignore file

# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar
```

Print Share