git/GitHub for Developers

February 4, 2024

Engineers for Exploration, UC San Diego

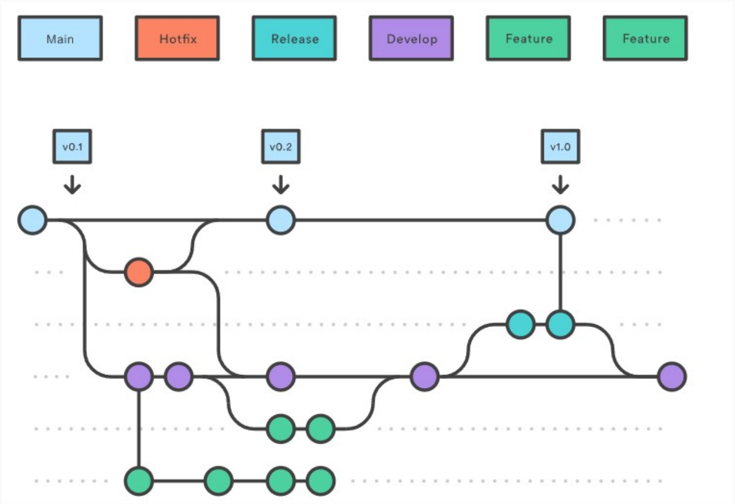
Introduction

- **Git** is a distributed version control system. **GitHub** is a platform for hosting Git repositories and facilitating collaboration.
- git/GitHub enables efficient project management and team collaboration with its version control management and branches.
- **Basic Commands:**
  - *git init*: Initialize a new Git repository.
  - *git clone*: Copy an existing repository.
  - *git add*: Stage changes for commit.
  - *git commit*: Save staged changes along with a commit message.
  - *git push*: Upload local repository content to a remote repository.
  - *git pull*: Fetch and integrate changes from a remote repository to your current branch.
  - *git branch*: List, create, or delete branches.
  - *git checkout*: Switch branches or restore working tree files.
  - *git merge*: Combine changes from different branches into your current branch.

Git Workflows

- **Feature Branch Workflow**:
  - **Description:** Each new feature is developed in its own branch to avoid disrupting the main codebase. Merging is done via pull requests to facilitate code review.
  - **Benefits:** Keeps the main branch stable. Encourages collaboration and review before integration.
  - **Use Case:** Ideal for ongoing projects with multiple team members working on different features simultaneously.
- **Gitflow Workflow**
  - **Description:** A branching model for project release management with separate branches for development, features, releases, hotfixes, and the main branch.
  - **Benefits:** Manages releases systematically, assigns clear roles to each branch, and tracks progress more efficiently.
  - **Use Cases:** Suited for projects with scheduled release cycles and the need for parallel releases.
- **Fork Workflow**

Git Workflows Cont.



Tags

- **Purpose of Tags:**
  - Mark significant points of the project's history.
  - Useful for marking release points (remember semantic versioning).
- **Creating Tags:**
  - Lightweight tags: *git tag tagname*
  - Annotated tags: *git tag -a tagname -m "message"*
- **Listing and Deleting Tags:**
  - List all tags: *git tag*
  - Delete a tag: *git tag -d tagname*

Releases

- **Creating Releases in GitHub:**
  - Navigate to your repository's releases section.
  - Draft a new release and choose the git tag that marks the version.
  - Add release notes to describe the changes or improvements.
- **Benefits of GitHub Releases:**
  - Bundle source code, executable files, and other assets in one package.
  - Provide detailed release notes to inform users about the changes or new features.
  - Auto updates (Demo).

Pull Requests (PR) Management

- **Keep PRs Small and Focused:** Encourage contributions that are easy to review and discuss. Smaller changes are easier to understand and less likely to introduce errors.
- **Use a Checklist:** Develop a review checklist to ensure consistency and thoroughness. This can include code style, testing, documentation, and performance considerations.
- **Automated Checks:** Utilize GitHub Actions to run automated tests, linting, and other checks when PRs are opened or updated.

GitHub Actions for Automation

- **Event-Driven:** Trigger workflows on GitHub events like push, pull requests, or issue comments.
- **Workflows and Actions:** Combine multiple actions to create workflows defined in YAML files.
- **Hosted Runners:** Run workflows on GitHub-hosted runners or self-hosted runners.
- **Examples:**
  - **Automating Testing and Deployment**: Automatically run tests on every pull request or push to a specific branch. Deploy your application when a pull request is merged into the main branch.
  - **Custom Workflows for Project Management:** Auto-assign project issues to members based on labels. Auto-label pull requests based on modified file paths.

Securing Your Project

- **Branch Protection Rules:** Configure rules to protect branches, requiring pull request reviews, status checks before merging, and more.
- **Dependabot:** Automatically scans your dependencies for known vulnerabilities and suggests updates or patches.
- **Secret Scanning:** Detects secrets and credentials exposed in your code and provides alerts.
- **Code Scanning:** Automatically scans your code for vulnerabilities when you push code to GitHub.

Extras

- **Interactive Rebase:**
  - **Definition:** Tool for rewriting history, used to edit, delete, or squash commits.
  - **Use Case:** Cleaning up a feature branch before merging it into the main branch.
- **Cherry-picking:**
  - **Definition:** Allows you to pick a commit from one branch and apply it to another.
  - **Use Case:** Applying a bug fix from one branch to another without merging all changes.
- **Stashing Changes:**
  - **Definition:** Temporarily, shelves (or stashes) change so you can work on a different task.
  - **Use Case:** Switching between branches without committing half-done work.