Detailed Guide for Developers: Working with the Repositories

Contents

USING TERMINAL TO WORK ON A REPOSITORY

Cloning the Repository

To start working with the repository, clone it to your local machine using terminal:

git clone https://github.com/[username]/[repository-name].git

cd [local-repository-directory]

Setting Up the Development Environment

Ensure you have Git Bash installed. Verify the installation with:

git –version

or use any terminal you are comfortable with.

Branching Strategy

Adopt a consistent branching strategy to manage changes:

Master/Main Branch: Contains the stable and released code.

Feature Branches: Use these for developing new features or fixing bugs. Use a relatable title

- Example: Feature branch “ContentFix” to fix any content mistakes.

Branching and Merging

Create a new branch: git branch [branch-name]. Use your task ID when creating a branch

Switch to a branch: git checkout [branch-name]

To create a feature branch and switch to it immediately use the following command:

git checkout -b feature/[branch-name]

Merge a branch: git merge [branch-name]

Delete a branch: git branch -d [branch-name]

Resolving Conflicts

When merging branches, conflicts might occur if the same lines in the same files have been changed in both branches. Here are the steps to resolve these conflicts.

Identify Conflicts:

git status

This command will show the files that have conflicts. Conflicted files will be marked with “*both modified.*”

Resolve Conflicts

Manually edit the conflicted files. Open the conflicted files in your code editor. Conflicts will be marked with:

```
<<<<<<< HEAD
Your changes in the current branch.
=======
Changes in the branch being merged.
>>>>>>> branch-being-merged
```

Add Resolved Files

After resolving conflicts, add the resolved files to the staging area

```
git add [file-name]
```

Making Changes/New Updates

- Follow the repository's coding style guidelines.
- Write clear, concise, and descriptive commit messages.
    - **feat:** (new feature for the user, not a new feature for build script)
    - **fix:** (bug fix for the user, not a fix to a build script)
    - **docs:** (changes to the documentation)
    - **style:** (formatting, missing semi colons, etc; no production code change)
    - **refactor:** (refactoring production code, eg. renaming a variable)
    - **test:** (adding missing tests, refactoring tests; no production code change)
    - **chore:** (updating grunt tasks etc; no production code change)
- Document your code where necessary.

Complete the Merge

Once all conflicts are resolved and added to the staging area, complete the merge:

```
git commit -m [commit message]
```

- Example message: **docs: Updated the readme file (ticket number)**
    - replace (ticket number) with the number given through click up.

Replace [commit message] with a meaningful description of the changes.

All major changes should be committed to the main repository.

Syncing with Remote Repository

Push changes - sends your committed changes from your local branch to the remote repository.

```
git push [local-repository] [branch-name]
```

Pull changes - updates your local branch with changes from the remote repository.

```
git pull [local-repository] [branch-name]
```

Add remote repository - Associates your local repository with a remote repository

git remote add [local-repository] ssh://git@github.com/[username]/[repository-name].git

Set remote URL - Updates the URL of the remote repository

git remote set-url [local-repository] ssh://git@github.com/[username]/[repository-name].git

Create Pull Request:

- **Navigate to Pull Requests:** Go to the "Pull Requests" tab in your repository and click on "New pull request". Final Pull request should be submitted to the Development Branch
- **Base and Compare Branches:** Choose the base branch (where you want to merge changes) and the compare branch (your newly created branch).
- **Review Changes:** Review the changes and provide a description of what you have done.
- **Create Pull Request:** Click "Create pull request" to submit it for review.

Maintaining Repository Hygiene

Regularly sync with the main branch

Run these set of commands to ensure your local branch is up to date with the upstream repository

git fetch upstream

git checkout main

git merge upstream/main

Clean up local branches that have been merged

Use this command to delete a local branch that has been merged into the current branch. Cleaning up merged branches helps keep your local repository tidy and reduces clutter.

git branch -d [branch-name]

Remove stale remote-tracking branches

To clean up references to remote branches that no longer exist:

git fetch -p

Tagging

Help with better version control.

Create a Tag

The following command creates a tag with the version and a message.

git tag -a [version] -m [message]

Push Tags

Pushes all local tags to the remote repository making them accessible to others who clone the repository.

git push [repository-name] --tags

USING GITHUB WEB INTERFACE TO WORK ON A REPOSITORY

Accessing the Repository

Navigate to GitHub: Go to https://www.github.com and log in to your account.

Access the Repository: Once logged in, navigate to the repository you have read/write privileges for. You can find repositories you have access to under your profile or by using the search bar.

Understanding Repository Structure

Code: This is where the source code of the project resides. Files are organized in directories.

Issues: Used for tracking tasks, bugs, and feature requests.

Pull Requests: Where code changes are proposed, reviewed, and discussed before merging.

Actions (if enabled): Automation workflows, like testing or deployment.

Projects (if enabled): Organize tasks and track progress.

Making Changes

Editing Files

- Navigate to the file you want to edit in the repository's code section.
- Click the pencil icon (Edit this file) to start editing directly in your browser.
- Make your changes and describe them briefly in the provided field.
- Choose to create a new branch and commit your changes changes.

Creating a Branch:

Branching: Click on the branch dropdown (usually shows "main" or "master").

New Branch: Type a name for your new branch (e.g., feature/new-feature-name). Use your task ID when creating a branch

Create Branch: Click "Create branch" to make your changes on this branch instead of the main one.

Committing Changes

Commit Message: Provide a clear and concise commit message that explains your changes.

- **feat:** (new feature for the user, not a new feature for build script)

    - **fix:** (bug fix for the user, not a fix to a build script)
    - **docs:** (changes to the documentation)
    - **style:** (formatting, missing semi colons, etc; no production code change)
    - **refactor:** (refactoring production code, eg. renaming a variable)
    - **test:** (adding missing tests, refactoring tests; no production code change)
    - **chore:** (updating grunt tasks etc; no production code change)
- Example message: **docs: Updated the readme file (ticket number)**
    - replace (ticket number) with the number given through click up.
- All major changes should be committed to the main repository.

Commit Directly: Commit changes directly to the main branch if it's allowed, or to your newly created branch.

Creating Pull Requests

- **Navigate to Pull Requests:** Go to the "Pull Requests" tab in your repository and click on "New pull request". Final Pull request should be submitted to the Development Branch
- **Base and Compare Branches:** Choose the base branch (where you want to merge changes) and the compare branch (your newly created branch).
- **Review Changes:** Review the changes and provide a description of what you have done.
- **Create Pull Request:** Click "Create pull request" to submit it for review.

Reviewing Pull Requests

Code Review: Other contributors or maintainers will review your pull request.

Discussion: Use the comment section to discuss the changes if needed.

Approval: Once approved, the pull request can be merged into the base branch by a maintainer

Managing Issues

View Issues: Navigate to the "Issues" tab to see all open issues.

Creating Issues: Click "New issue" to create a new task, bug report, or feature request.

Labels and Assignees: Add labels to categorize issues and assign them to team members

Documentation: Keep the README.md file up to date with relevant information about the project, setup instructions, and how to contribute guidelines.

Industrial language

AC - Acceptance criteria
SaaS - Software as a service
ETA - Estimated time (of arrival)
PR - Pull request
MR - Merge request
Prod - Production (environment)
UAT - User acceptance testing
Dev - Development
Param - Parameters
Config - Configurations
ML - Machine learning
NLP - Natural language processing
UX - User experience
VS - Visual studio
API - Application Programming Interface
UI - User Interface
UX - User Experience
SDK - Software Development Kit
IDE - Integrated Development Environment
SQL - Structured Query Language
HTML - Hypertext Markup Language
CSS - Cascading Style Sheets
JS - JavaScript
JSON - JavaScript Object Notation
XML - Extensible Markup Language
HTTPS - Hypertext Transfer Protocol Secure
URL - Uniform Resource Locator
HTTP - Hypertext Transfer Protocol
CMS - Content Management System
CRM - Customer Relationship Management
ERP - Enterprise Resource Planning
QA - Quality Assurance
CI/CD - Continuous Integration/Continuous Deployment
DevOps - Development and Operations
SaaS - Software as a Service
PaaS - Platform as a Service
IaaS - Infrastructure as a Service
MVP - Minimum Viable Product
APIaaS - API as a Service
JVM - Java Virtual Machine
OOP - Object-Oriented Programming
MVC - Model-View-Controller
REST - Representational State Transfer
SOAP - Simple Object Access Protocol
CRUD - Create, Read, Update, Delete
TCP/IP - Transmission Control Protocol/Internet Protocol
DNS - Domain Name System
URL - Uniform Resource Locator
CDN - Content Delivery Network
SSL/TLS - Secure Sockets Layer/Transport Layer Security
JVM - Java Virtual Machine
JIT - Just-In-Time Compilation
CLI - Command Line Interface
GUI - Graphical User Interface
DRY - Don't Repeat Yourself
TDD - Test-Driven Development
BDD - Behavior-Driven Development
MVP - Model-View-Presenter
MVVM - Model-View-ViewModel
ORM - Object-Relational Mapping
SQL - Structured Query Language
NoSQL - Not Only SQL
JWT - JSON Web Token
CSRF - Cross-Site Request Forgery
CR - Change request
EOD - End of the day
KT - knowledge transfer
JD - job description