Git Workflow Guidelines skulio

Teoman Köse

February 27, 2025

1 Introduction

This document defines the Git workflow for our development team. Following this workflow will help maintain a structured, conflict-free, and efficient development process.

2 Branching Strategy

We follow a **feature branching model** with the following branch structure:

- **main** – The stable production branch. Only thoroughly tested and reviewed code is merged here.
- **develop** – The integration branch where new features are merged before moving to `main`.
- **ftr-*name_of_feature*** – A branch for each feature.
- **bugfix-*name_of_fix*** – A branch for bug fixes.
- **hotfix-*name_of_critical_fix*** – A branch for urgent production fixes.

Branch Naming Convention:

- `ftr-name`
- `bugfix-name`
- `hotfix-name`

3 Git Workflow

3.1 Creating a Feature Branch

1. Start from the latest `develop` branch:

```
git checkout develop
git pull origin develop
```

2. Create a new feature branch:

```
git checkout -b ftr-name
```

3. Work on the feature, committing changes regularly.

3.2 Merging vs. Rebasing

Rebasing (Preferred) To keep a clean history, always rebase your feature branch before merging:

```
git checkout ftr-name
git fetch origin
git rebase origin/develop
```

If conflicts occur, resolve them, then continue:

```
git rebase --continue
```

Merging (When Necessary) Merging is used when preserving branch history is important:

```
git checkout develop
git merge ftr-name
```

3.3 Pushing Changes

Push your feature branch to remote:

```
git push origin ftr-name
```

3.4 Creating a Pull Request (PR)

1. Push your branch and create a pull request (PR) to `develop`.
2. Ensure the PR description includes:
   - Summary of changes
   - Testing steps
3. Wait for at least one team member to review and approve.

3.5 Merging to `develop`

After approval, merge the PR into `develop` and delete the feature branch:

```
git checkout develop
git pull origin develop
git merge --no-ff ftr-name
git push origin develop
git branch -d ftr-name
git push origin --delete ftr-name
```

3.6 Bugfixes and Hotfixes

Bugfix Branches (Non-Urgent Issues)

1. Create a bugfix branch from `develop`:

   ```
   git checkout -b bugfix-name develop
   ```

2. Fix the issue and create a PR to merge into `develop`.

Hotfix Branches (Urgent Production Fixes)

1. Create a hotfix branch from `main`:

   ```
   git checkout -b hotfix-name main
   ```

2. Fix the issue, then merge intro both `main` and `develop`.

   ```
   git checkout main
   git merge --no-ff hotfix-name
   git push origin main

   git checkout develop
   git merge main
   git push origin develop
   ```

3.7 Releasing to `main`

Once all features for a release are merged into `develop`, create a release branch:

```
git checkout -b release-v1.0 develop
git push origin release-v1.0
```

After testing, merge the release into `main`:

```
git checkout main
git merge --no-ff release-v1.0
git push origin main
```

Then, merge back into `develop`:

```
git checkout develop
git merge main
git push origin develop
```

4 Roles and Responsibilities

4.1 Developers

- Work on `ftr-`, `bugfix-`, or `hotfix-` branches.
- Ensure each feature branch is rebased with `develop` before merging.
- Open Pull Requests (PRs) and wait for review before merging.
- Write clear commit messages describing changes.

4.2 CTO / Lead Developer

- Reviews and approves Pull Requests.
- Ensures the repository stays clean and follows best practices.
- Merges branches into `develop` and `main` when stable.
- Handles hotfixes directly on `main` if critical issues arise.
- Defines Git policies and ensures all team members follow them.

5 Best Practices

- Commit small, meaningful changes with clear commit messages.
- Rebase frequently to keep your branch updated.
- Never push directly to `main` or `develop`.
- Use code reviews to maintain quality.

6 Conclusion

Following this workflow ensures a structured and efficient development process, reduces merge conflicts, and improves code quality. Stick to these guidelines to keep our repository clean and maintainable.