Comprehensive Git Contribution Workflow Guide

Contents

Introduction

Git is an essential tool for modern software development, enabling collaboration across teams and individuals worldwide. Whether you're contributing to open-source projects, working within a team, or managing your personal projects, mastering Git workflows is critical for efficient and organized development.

This guide provides a comprehensive walkthrough of the essential Git contribution workflow, tailored for scenarios where you do not have direct write permissions to a repository. It covers every step, from forking a repository to creating pull requests and keeping your fork in sync with the original repository. By following this workflow, you will:

- Learn how to fork and clone repositories.
- Manage and update your local branches effectively.
- Collaborate with others through pull requests and address feedback seamlessly.
- Keep your fork synchronized with the latest changes in the original repository.

This guide is designed to be beginner-friendly while also serving as a reference for experienced users. It emphasizes best practices and provides commands for all the essential operations, ensuring you can contribute to projects confidently and efficiently.

1. Fork the Repository

- Go to the original repository on GitHub.
- Click the **Fork** button (top-right corner).
- This creates a copy of the repository under your GitHub account.

2. Clone Your Fork

Clone your fork to your local machine:

```
git clone https://github.com/<your-username>/<repo-name>.git
```

This creates a local copy of your forked repository.

3. Add the Original Repository as Upstream

Navigate to the cloned repository:

```
cd <repo-name>
```

Add the original repository as the upstream remote:

```
git remote add upstream https://github.com/<original-owner>/<repo-name>.git
```

Verify remotes:

```
git remote -v
```

4. Keep Your Fork in Sync with Upstream

To sync your fork with the upstream repository:

1. Fetch updates from upstream:

```
git fetch upstream
```

2. Merge updates into your local branch (usually `main`):

```
git checkout main
git merge upstream/main
```

3. Push updates to your fork:

```
git push origin main
```

5. Create a New Branch for Your Work

Never work directly on the `main` branch. Create a new branch:

```
git checkout -b <branch-name>
```

To switch branches, e.g. to the new created branch or even the main branch, use:

```
git switch <branch-name>
```

6. Make Changes and Commit

After making changes:

- Stage changes:

```
git add <file-name>
```

- Commit changes:

```
git commit -m "Brief description of changes"
```

7. Push Your Branch to Your Fork

Push your branch to GitHub:

```
git push origin <branch-name>
```

8. Create a Pull Request

- Go to your fork on GitHub.
- Switch to your branch and click **New Pull Request**.
- Explain your changes and submit the pull request (PR).

9. Address Changes (if Requested)

If the maintainers request changes:

1. Make the requested changes.
2. Commit and push the changes to the same branch:

```
git add <file-name>
git commit -m "Address review comments"
git push origin <branch-name>
```

The pull request will automatically update with the new changes.

10. Sync Regularly with Upstream

To avoid falling behind, sync your fork regularly:

1. Fetch updates:

```
git fetch upstream
```

2. Merge updates:

```
git checkout main
git merge upstream/main
```

3. Push updates:

```
git push origin main
```

Best Practices

- Create small, focused pull requests (PRs).
- Write clear and concise commit messages.
- Always pull the latest changes before starting new work.
- Follow the project's contribution guidelines.
- Review and test your changes locally before pushing them.