Git merge workflow

- Gitflow Workflow
- Creating Pull Request

Branche types

- Master or Main branch (main or master)
- production branch (prod)
- development branch (develop)
- qa branch (qa)
- stage branch (stg)
- Release branch (release)
- and feature branch

Case 1

Naming convenction: master, develop, qa, stg, prod, feature/EC-3000, release1.4, feature/ON-680 etc.

1. A develop branch is created from the master or prod branch
2. Feature branches are created from develop branch (This is when developers want to start a new feature)
3. When developers are finished with the feature branch, they merge it into the develop branch, and the deployment kick-off to test the new feature. The release branch will be created when the develop branch deployment is successful.
4. A release branch is created from develop.
5. From the release branch, we can merge the code to **qa, stg, prod and master**.
6. If an issue is detected in any of these environments (**qa, stg, prod and master**), a hotfix branch is created from that specific env.
7. Once the hotfix is complete it is merged to both develop and master
8. The release branch can be deleted and we can restart the cycle

Case 2

Naming convenction: master, develop, dev1, dev2, dev3, qa1, qa2, stg1, stg2, prod1, prod1, feature/EC-3000, release1.4, feature/ON-680 etc.

- These branches are used to deploy the app/service's code to a particular environment. Once a developer is ready to deploy to an environment, they will raise a ***pull request*** from the **develop**. Depending on the environment, team members will review the code and approve the PR. Once the code is merged a deployment will start.
- **NB:** EC-3000 or ON-680 is the ticket number for the new feature to be develop

Staging, Qa, and Production environments

For smaller companies (it's not clear how big yours might be), three environments ***(dev, stage, production)*** are common. Larger companies will often have a **QA** environment between dev and stage.

These normally break down as follows:

- **Dev:** Dev environment is the one created and maintained by the Development team for writing the code. Access to this environment is given to the development team only. Usually, the **QA** team doesn't have access to this environment. This environment is mostly used by the Dev team for their unit testing. Changes made by developers are deployed here so integration and features can be tested.
- **QA/Test:** (Not all companies will have this). Environment for quality assurance. QA environment is where the testing actually takes place. This environment is owned by the ***QA team***. The ***DEV team doesn't have access to this environment***. After design and coding completion, the code is moved to the QA environment for the QA team to conduct test execution.
- **Staging/ Pre-Production:** This environment is normally a mirror of the ***production environment***. The staging area contains the "next" version of the application and is used for final stress testing and **client/manager** approvals before going live.
- **Production:** This is the currently released version of the application. This is the environment that your customers interact with. It needs to be as stable and bug-free as possible.

What is a Master Branch?

The master branch is the branch that reflects what is currently running in a production

What is a Feature Branch?

A feature branch is simply a separate branch in your Git repo used to implement a single feature in your project. When “using feature branches,” you are creating a new branch for each new feature you develop, instead of just checking in all your changes into the master branch (which is typically the name given for the main development branch).

A feature is really anything you want it to be—a bug fix, new functionality, or even just more documentation. Once the feature is complete, the changes are `merged` into a respective environment using a pull request or PR.

Feature branches are short-lived branches that should only exist for the duration of a ***JIRA feature***, story (1-8) days. Once feature development has been completed, the changes committed to the feature branch should be merged to the ***dev branch*** and the feature branch should be deleted.

What is the release branch?

A release branch will be created when all development for a particular deployment is complete and we are ready to start testing for that deployment. This branch will be a snapshot of the develop branch and will be the branch that is promoted through environments unless there is a requirement to use develop

What is Git Merge?

Merge is a command used in Git to move the changes in a branch to another. Usually, the new features are developed in the dev branch and merged into the master branch after finishing the development. All the changes in the dev branch is added to the master branch on the merge. but the dev branch will be unaffected.

What is a pull request or PR?

Creating a pull request

About pull request reviews

How do I get changes approved with a pull request?

After **forking the repository** on the remote hosting service and cloning it to your local machine, you can start working on making changes to the code.

After you’re happy with the changes, you can push the work up to your fork and open a pull request that will signal the main repo. This pull request asks the maintainer(s) to review your work, provide comments, request edits, etc.

If your pull request is approved, the maintainer will merge your changes into the main repo.

merge conflict

Resolving a merge conflict on GitHub

A difference that occurs between merged branches. **Merge conflicts** happen when people make different changes to the same line of the same file, or when one person edits a file and another person deletes the same file. The merge conflict must be resolved before you can merge the branches.

The overall flow of Git Workflow

- Git branching model

1. A develop branch is created from master
2. A release branch is created from develop
3. Feature branches are created from develop
4. When a feature is complete it is merged into the develop branch
5. When the release branch is done it is merged into develop and master
6. If an issue in the master is detected a hotfix branch is created from master
7. Once the hotfix is complete it is merged to both develop and master