COURSEWARE

Branching

Contents

Overview

Branching in Git helps us to define workflows that make sure the code that is being delivered is in the best state possible, minimising risks for any errors or crashes.

With Version Control Systems, like Git, we can separate the codebase on to many different branches.

This feature can be utilised for isolating the development and testing of new features from working code that is running on a production environment.

Git Branching Workflow Example

This is an example of a workflow using Git with some notes below which explain in more detail the processes.

The two main branches that exist are the `develop` and `main` branches.

New Application Features

Conventionally, new features are to be created in feature branches from the develop branch, and, once the feature has been developed, the code can be reviewed by a peer in a Pull Request and deployed to a test environment for Integration or User Acceptance Testing.

If all testing and reviews pass, the Pull Request can be approved and merged into the `develop` branch.

Releases

When there is a release approaching, a release candidate branch can be made.

On this new branch, further testing of all the new features working together can take place, and more candidates can be made on this branch to amend any issues.

Once testing has passed and the release has been signed off by the individual accountable for releases, the release candidate branch can be merged into master for the release to be deployed to a production environment.

Once merged into the master branch, the code can be tagged or marked as a release on the Git service that you are using.

Changes must also be merged back into the develop branch, so that any changes that were made to the release candidate will also be included in future releases.

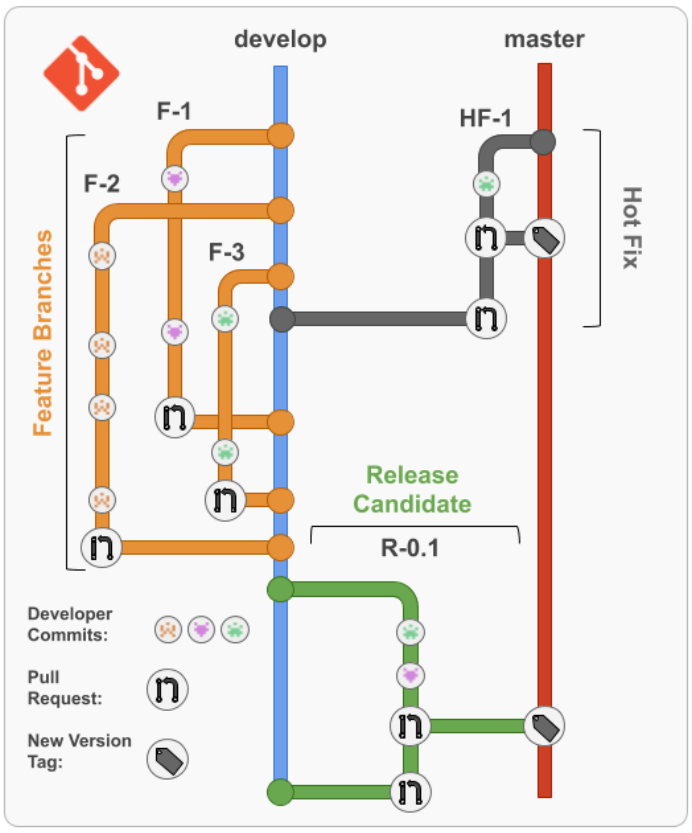
Hotfixes

Preventing hotfixes is one of the main reasons for designing a workflow such as the one below.

Hotfixes should be prevented where possible, but, as this is the wonderful world of IT we work in, they could still happen at some point.

A hotfix can be conducted by creating a hotfix branch from the master branch and applying the changes on that branch; before merging back into the master branch all the changes should, of course, be tested and reviewed to avoid even more hotfixes!

Once merged into the master branch, the changes must also be merged back into the develop branch; this will keep any important changes the hotfix made, and the code in the master branch should be tagged.



Creating and Deleting Branches

You can see all the current branches in your repository using:

```
git branch
```

To create a new branch, the command is:

```
# git branch [NEW_BRANCH_NAME]
git branch develop
```

This will create a new branch from whichever branch you are currently on (so this new branch will branch from `develop` if you run the command whilst on that branch).

If you want to work on a new branch straight away, you can create a branch and `checkout` to it at the same time:

```
# git checkout -b [NEW_BRANCH_NAME]
git checkout -b develop
```

When you have finished working on your branch, and your code has been merged to `main`, it is good practice to delete the branch:

```
# git branch -d [BRANCH_NAME]
git branch -d feature-123
```

You will also need to do this on your Git Service, which you're likely using (such as GitHub). This is usually done when closing a Pull Request.

If you aren't closing a Pull Request and find yourself needing to close a branch, you can use the following command to delete a branch on your remote repository (again, likely on GitHub):

```
# git push --delete origin [BRANCH_NAME]
git push --delete origin feature-123
```

Tutorial

1. Run the command `git clone https://github.com/QACTrainers/git-branching.git`
2. Run the command `cd ./git-branching`
3. See which branches are currently configured for that repository
4. Create a new branch called `develop`
5. Checkout the `develop` branch
6. From `develop`, checkout to a new branch called `issue-1`
7. Delete the `issue-1` and `develop` branches

Exercises

There are no exercises for this module.