Deployment Workflow

We have the following environment setup:

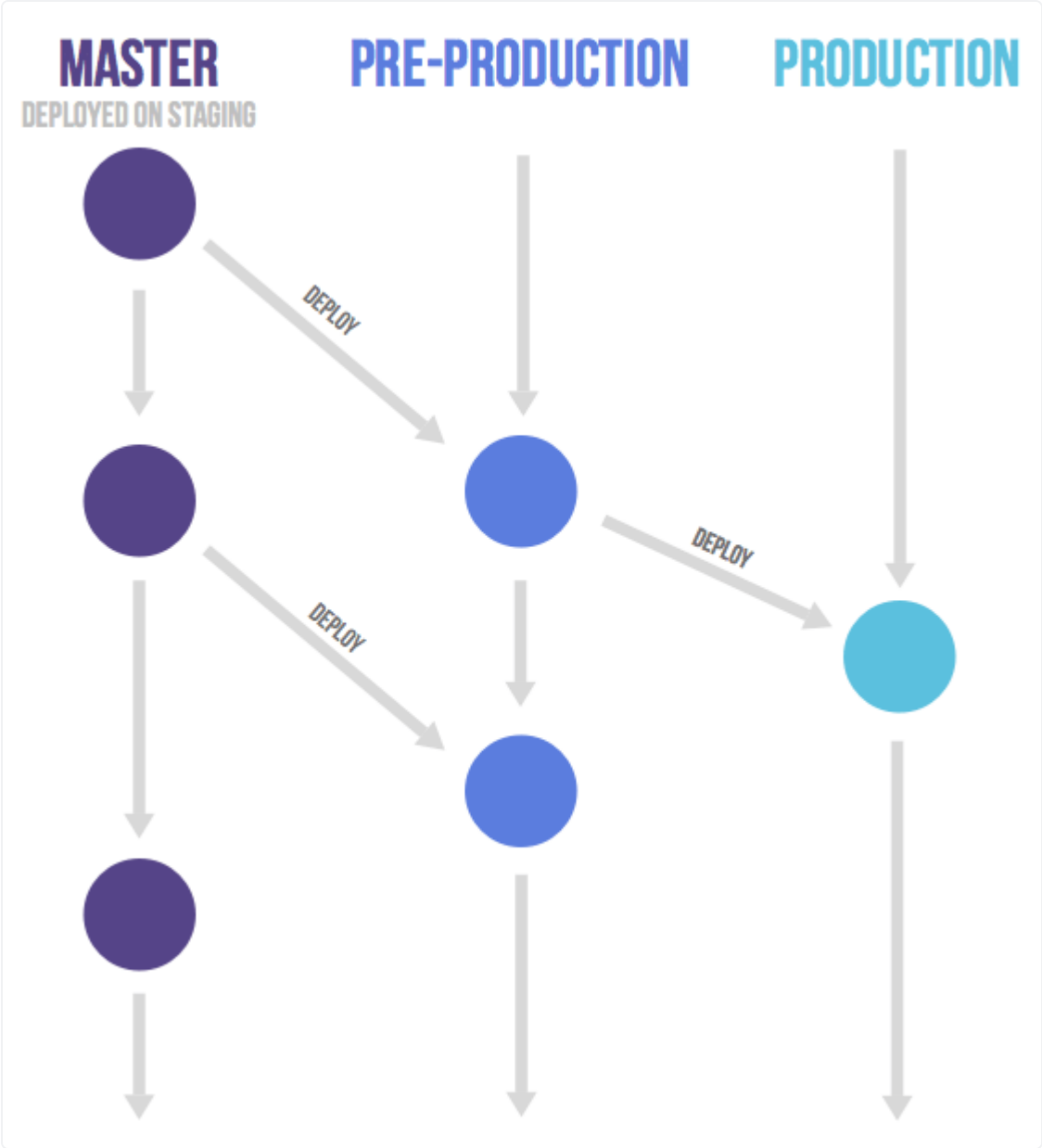
- Development (only internally to test and review every feature)
- Staging (for the client to test and review at the end of each sprint)
- Production (final system for end-users)

This correlates with our branches:

- development (= development environment)
- master (= staging environment)
- production (= production environment)

A quick overview of our deployment workflow:

As a basis for our deployment strategy and version controlling, we use Gitlab Flow :



This is what our process basically looks like:

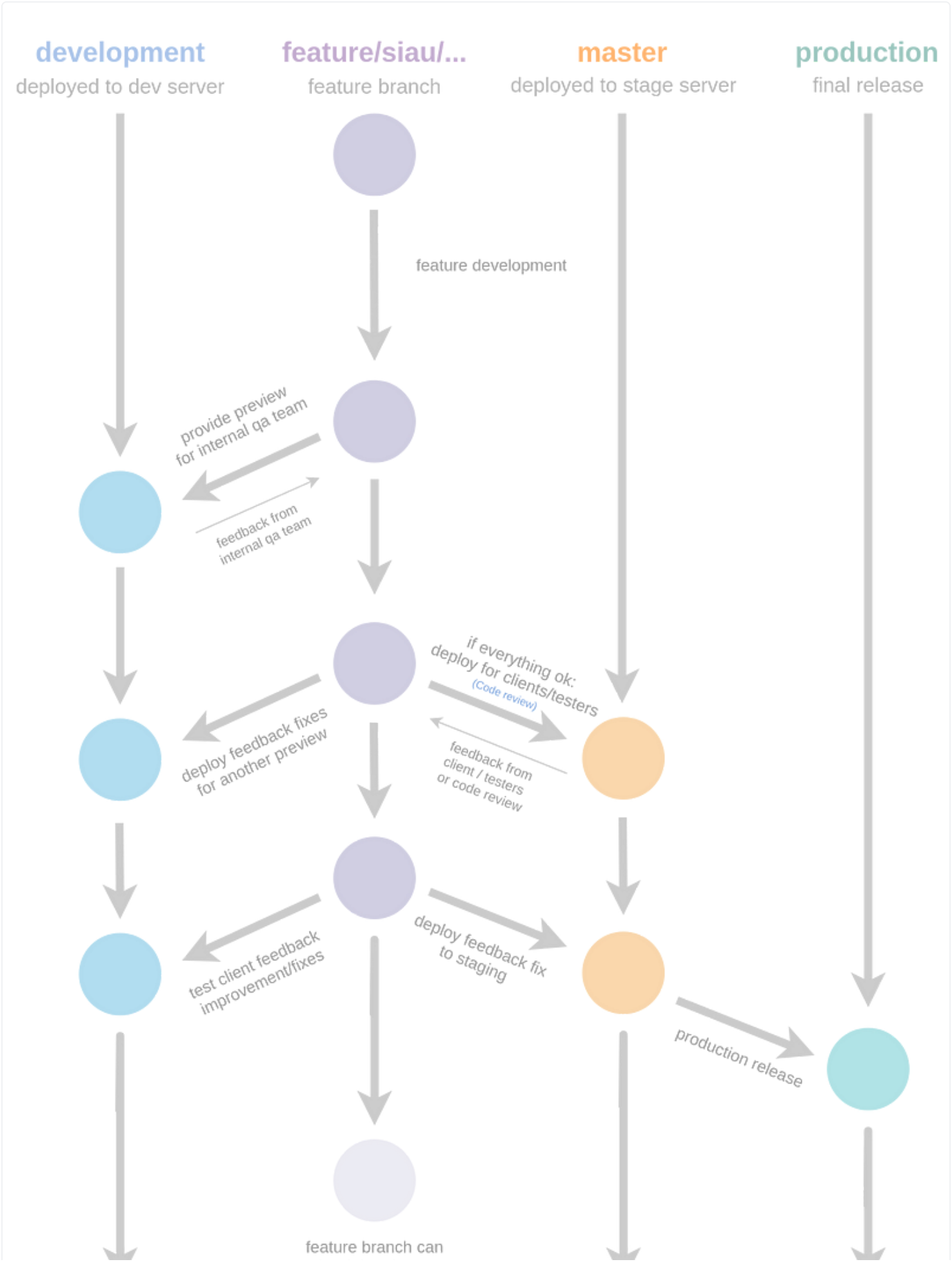
development
deployed to dev server

feature/siau/...
feature branch

master
deployed to stage server

production
final release

Here it is again in non-animated form, to inspect in detail:


development
deployed to dev server
feature/siau/...
feature branch
master
deployed to stage server
production
final release
feature development
provide preview
for internal qa team
feedback from
internal qa team
if everything ok:
deploy for clients/testers
(Code review)
deploy feedback fixes
for another preview
feedback from
client / testers
or code review
test client feedback
improvement/fixes
deploy feedback fix
to staging
production release
feature branch can

Step 1 - Development # Use feature branches.

Every developer pushes their code features/bugfixes/... on dedicated feature branches that follow the naming convention

```
{TYPE}/{INITIALS}/{FEATURE_NAME}
```

{TYPE} = either bugfix/feature/hotfix

{INITALS} = 2–4 long name initials (to know who is responsible for branch)

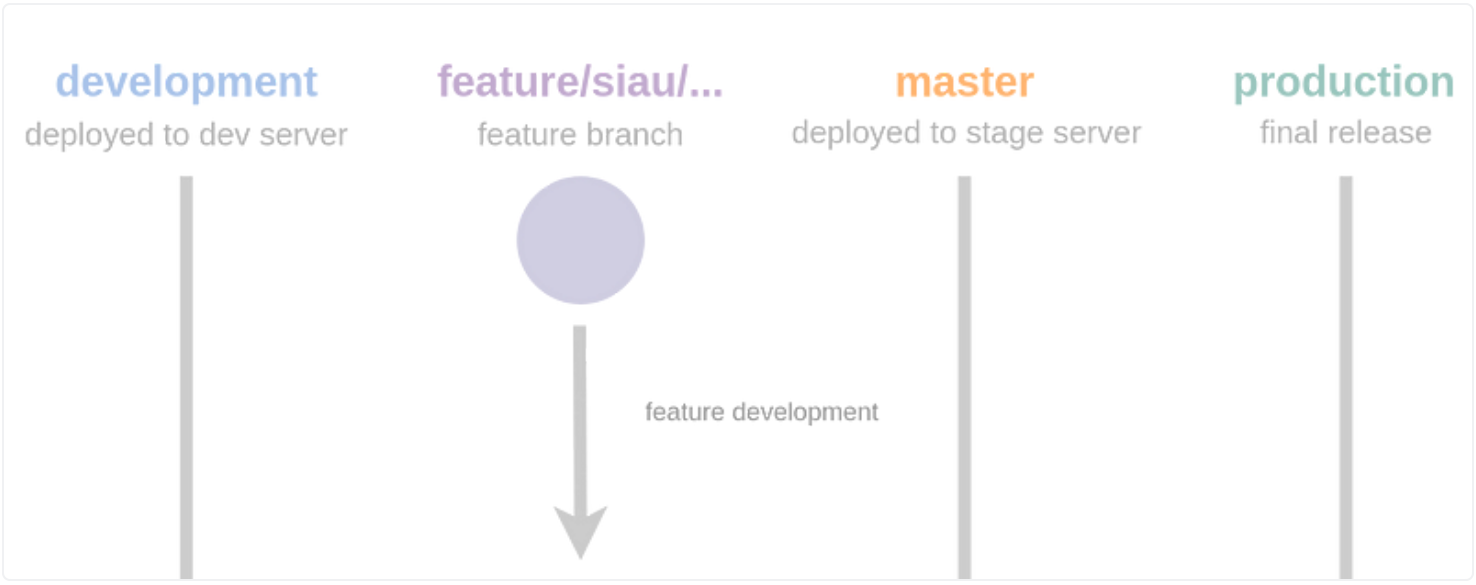
{FEATURE_NAME} = task or feature name

Here is an example of a branch name for my new feature of “adding push notifications”.

```
feature/siau/adding-push-notifications
```

The code will be edited there, and everyone should try to push as often as possible.

The branch does not need to be in a fully working state at all times and is usually only used by the feature author.



Step 2 - Preview # Merge into development branch

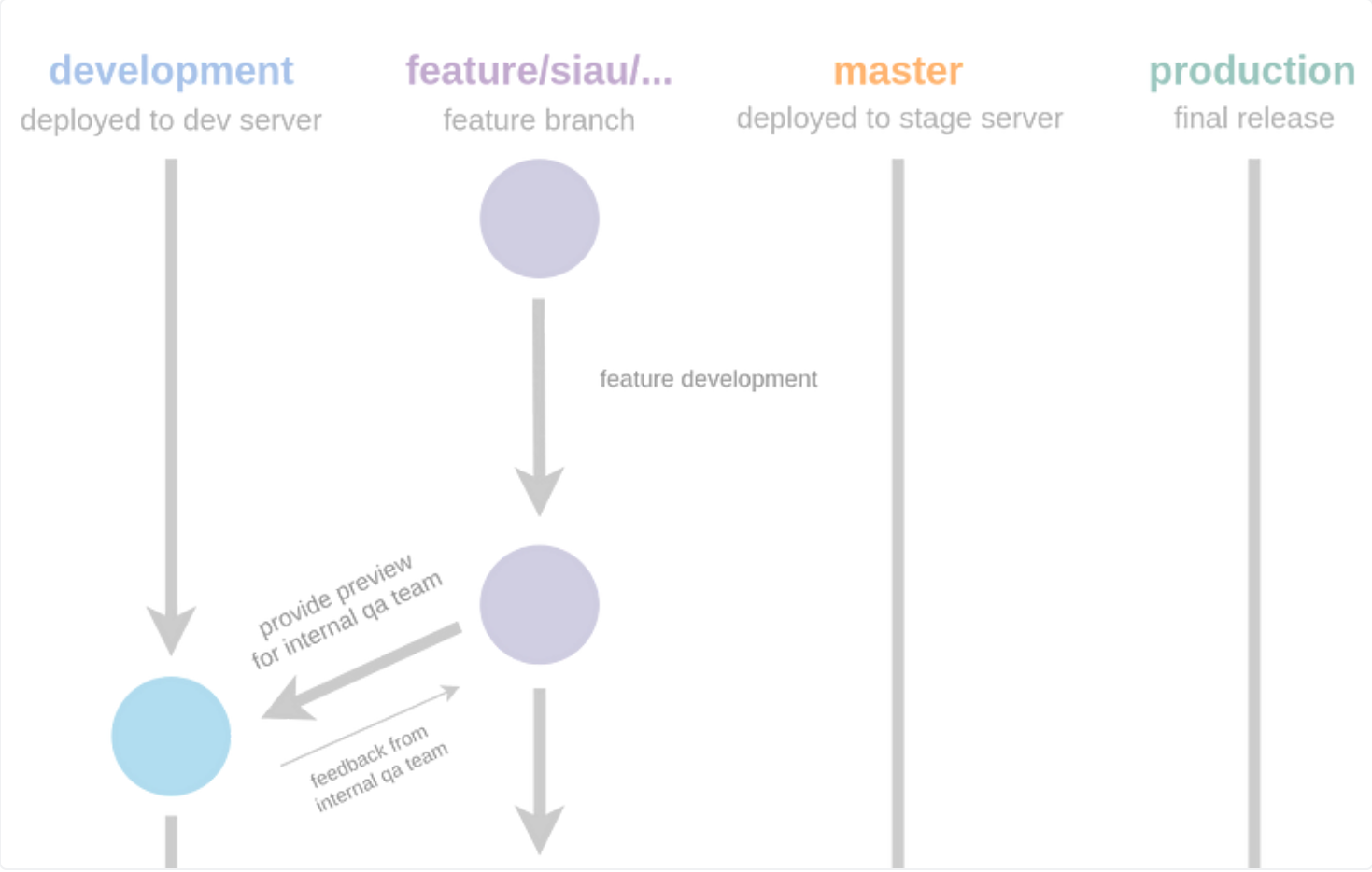
When the feature (at least partially) works and needs to be tested (or when another dev needs to implement an interwoven feature with this one), the developer can merge their changes directly into development without waiting for approval or code review.

This will trigger an automated deployment to the dev server.

The dev server is only accessible internally and will be used for UX/UI reviews.

This is done to make sure there is no waiting time for reviewers to look at the code, to test on the implementation, and also to make it possible for our product testers, designers, UX people, frontend devs to check the live result before a code reviewer even spends time to check the code.

If something was found to improve or fix in this development version, feedback will be passed back to the developer in the form of a comment, and the dev will fix it in the feature branch and repeat this process until everyone is happy with the result.



Step 3 - Staging # Merge into master branch

If the feature is finished and all the internal UI and product reviewers are happy, we create a pull/merge request into our master branch.

This pull request, however, is not merged automatically. It has to be code reviewed and accepted by at least one other developer before it is fully merged and deployed to the staging server.

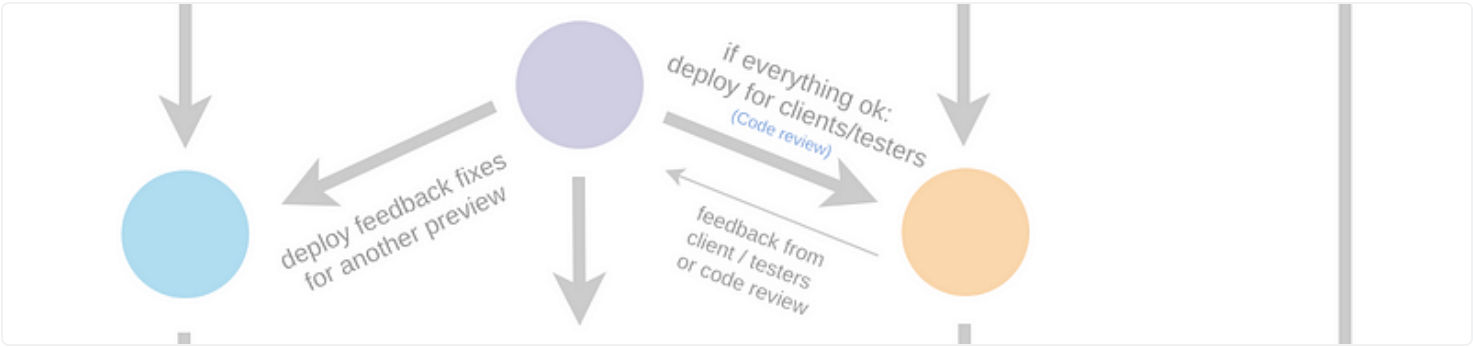
Therefore, the master branch is always a representation of clean, valid, and checked UI and code.

Code Review:

If potential code improvements are found during code review, and changes are necessary, feedback is provided. The dev needs to incorporate them in the feature branch, push them to dev-branch first to see if everything works, and then push them again to the master branch for another code review.

If that turns out positive, the reviewer will merge it into master and trigger a deployment.

The reviewer will delete the feature branch on merge in.



Client/Product owner improvement suggestions:

If new feature requests or improvements are planned, a new ticket will be generated, and we start again at step 1.

Step 4 - Release # Merge into the production branch

At the end of every sprint, we merge the staging branch into our production branch and thereby deploy all of our new features that have been tested on master/staging to our production server.