Version control and deployments

Branches

main (or develop)

This is the main development branch, it should contain the latest runable code.

feature

A feature branch is used when working on a ”feature” (new functionality or updates to current functionality). Almost all active development is done in feature branches. The feature branch must be merged from `main` or (in some cases) another `feature` branch. If the functionality that are being worked on depends on code that doesn't exists in `main`, it is possible to branch the new `feature` branch off another `feature` branch (but is should in most cases be avoided).

A feature branch should always be kept as small and short lived as possible. Split large tasks into several small features so they can be merged back into `main` as soon as possible.

When the feature is done the `feature` branch should be merged back into `main`.

release

A release branch is used to create a release “package” to deploy.

hotfix

A hotfix branch is used to handle urgent fixes in an environment (test/preprod/prod). The hotfix branch must be merged from a `release` branch. E.g. if a fix needs to be made in production the hotfix branch should be merged from `release/prod`. When the fix is done the hotfix branch should be merged back to `release/prod` as well as the `main` branch.

Hotfixes should be avoided if possible, always ask if the fix is so urgent that it can't wait until next planned release.

bugfix

A branch used to fix bugs. Should be merged from `main` and back to `main` when the work is done. This is for all bugfixes that doesn't need to be deployed directly, but can wait for the next planned release.

Environments

Which environments that are available might differ between projects, but the most common setup is this:

- test
- preprod
- prod

In this setup the version control will have a `release` branch for each environment:

- release/test
- release/preprod
- release/prod

Deployments

When working with deployments it's good to work with planned releases as much as possible. Where there are predefined release cycles. This way the whole team always know when a release is comming up and adjust the work for that.

An example release cycle might look like this:

- Each week on tuesdays a test release is made
- Every two week a preprod release is made
- Every month a production release is made

Then the teams knows that every monday a merge to `release/test` has to be made. It is also important that code that shouldn't be part of the test release doesn't get merged to `main` during this time. When the deployment to test is done it's free to start merging new features to `main`. If there is need to adjust the test release this can be made with a `hotfix` (or, if no new code has been merged into `main` it can be fixed with a `bugfix` branch, then merged to `main` and then to `release/test`).

Example of deployment work flows

https://embed.diagrams.net/?embed=1&proto=json&spin=1&saveAndExit=1&noSaveBtn=1&noExitBtn=0

Roles and responsibilities

Project manager / customer

Decides how the deployment cycle should look like (with what frequency and which days should deployments be made). The project manager is responsible for making sure the team knows the deploy schedule.

Team lead

Each team (backend and frontend) should have a team lead. It is the team leads responsibility to make sure that the work is merged into the `release` branch at the right time. It is also the team leads responsibility to make sure the deployment is done. Also the team lead must (as soon as possible) inform the project manager (and customer) if there will be some delays in either meging the code to the `release` branch or getting it deployed. It is not the team leads that must do the merge/deploy, anyone in the team can do this, but it's the team lead that must make sure it is being done.

The team lead is reponsible for setting code stops (no merges to `main` allowed) during the time of a release. No merges (except bugfixes) to `main` should be allowed during the time of a release. Exactly how long code stop depends on the project (but perhaps the day before release, the day of the release and the day after a release). It is often good to not allow new code into `main` when all features of a sprint is done.

But the code stop only means no new code can be merged into `main`, the developers can continue working in `feature` branches.

Developer

Each developer has the responsibility to branch off the correct branches (`main` for feature/bugfix branches and a `release` branch for hotfixes). Each developer is responsible for making sure that the code they merge into `main` is runable. Each developer is responsible for making sure that the code they write is solving the task in the way that is required. Each developer must make sure to not merge new features into `main` during the days of a release.