Version Control

KentSoft

We will be using GitLab as our Version Control. This is because of the it provides a vast amount of services and not just a Version control system.

As a DevOps lifecycle tool, which is exactly how we will be using it, as it will be at the core of our developers environments as stated in the Developer Guidelines Document. We as a team will be able to track issues, collaborate remotely, and review each others changes and additions.

We will set up branches for each team member, this in order for each developer to have their own development environment. Team members will then request to merge their branches into the master origin, where they will be essentially submitting their work/completed task to the rest of the team.

Team members must ensure that their local master branch stays up to date as possible, and should ensure to keep their working branch up to date with the master as much as possible.

A warning to developers, that are collaborating with Microsoft Office files, via merging. There will be conflicts if files that exist in the master branch are changed on another branch. And these have to be resolved via the terminal.

The best way to work around this is if each team member completes a different document for the documentation, and then they are all merged into one directory at the end.

Sometimes, it is more efficient, if the team is present/has given you authorisation it would be quicker to make quick and necessary edits in the master branch in order for there not to be any merge conflicts. This is because documentation conflicts cannot be resolved in the browser like most program files and it is hard to identify the differences as Microsoft office does not support any version control.

New to Git?

Find the Git Command Cheat sheet here. Make sure you have git bash installed if you are using a Windows PC!