Intro to Git and Github

Hyper Island - FED27

Per Enström – 2025-09-05

Git

Branches

- All commits in a git repository belongs to a branch
- By default a branch called `main` (or previously `master`) is created when initializing a repository
- We can look at all commits of a repository a bit like a tree, with branches of code diverging off, and unlike a tree merge together with the trunk again
- Branches are used heavily in development
  - Usually we have a `main` branch, which is always released to our url on the web
  - From that we branch off different branches where we can work on new features
  - On these branches we follow the commit small and often tactic
  - When we're done with our feature, we merge it back into `main` and release it to our users

Git

Branches

- Switching between branches are called Checking out
- To create a new branch and check it out we use the command `git checkout -b my-branch-name`
- Creating new branches are always made from the branch you currently have checked out, nothing is preventing us from branching from a branch
- In the terminal you will see what branch you're currently on, in my terminal it says `git: (main)` when I'm on the `main` branch

Git

Merging

- To merge our branch back into another branch we use the command `git merge`
- We merge a target branch, into the branch we have checked out
  - To merge a branch called `my-branch-name` into `main`, we first `git checkout main`, and then `git merge my-branch-name`
- There are a few merge *strategies* that git automatically choses between
  - The two common ones are *fast-forward* and *merge commit*
  - *fast-forward* takes all commits on our `my-branch-name` and adds them one by one to the `main` branch
  - *merge commit* takes all our commits and mashes them together, and creates a new single commit on `main` with all of our changes
- Merging does not delete a branch, we can continue to commit to either branch afterwards