torsdag 25 mars 2021

Git

Working in teams

- HEAD is poiting to the specific branch that we are currently on
  - master - local master branch
  - master/origin - remote master branch
- What to put in gitignore.
  - Dependencies
    - Share gradle file but not the dependencies for android eg
    - Pip for python eg
  - Secrets
    - Things that have to do with security, keys from amazon and google and so on
- Important lesson:
  - If you don’t know what is the best next move ask your peers
- The Branches
  - Many lines of development
    - When different members are working on different features
    - Instead of waiting until your peers are ready
    - Branch of from the master branch on your own branch to work on your specific feature
  - Create a branch
    - git branch my-feature
  - Switch between branches
    - git checkout my-feature or git checkout master
  - Merging and rebasing
    - git merge <branch>
      - You want to include your feature on <branch> into your main line of development that you are standing in right now
    - Two types of merges:

- Fast forward merge
    - When there have been no more commits on the master branch after starting to work on your new branch
    - Merge will only set your master branch pointer to point at the latest commit of your new branch
- Three-way merge
    - When master branch have evolved during the time you worked on your new branch
    - Now a whole new commit will be made and the history of the two branches will be shared when the merge is made (see images from pdf)
    - This kind of merge can lead to conflicts if changes have been made to the same file in the two different branches
    - Git will give you a warning, the conflict have to be resolved before merge can be executed
    - Git will insert symbols in your code to view two different versions, you have two remove these symbols, and pick what piece of code you want to keep
- Strategy to avoid conflicts:
    - Conflicts only occur when you edit the same file on different branches
    - Strategy:
        - git checkout feature
        - git rebase master
            - Will put the base of your branch onto the latest commit of the master branch (see slides for detailed images)
        - Golden rule:
            - Never use rebase on public branches!
            - Use it on your local branch
- Tutorial
    - git branch landing-page
    - git log

    - You can see the new branch landing page, but HEAD is still pointing to master
  - Git checkout landing-page
    - Now HEAD is pointing to landing-page
  - When using git checkout landing-page our IDE will show the files and changes that are included in the landing-page branch
  - git branch -vv to see what branch you are working on right now
  - When a conflict appears
    - Resolve the conflict in IDE
    - Add files to staging area
    - Run git diff —staged to view changes made
    - Commit
    - Run git log
    - We will see now in the log that the merge has been executed
    - This is how we handle a Three way merge
      - This will be showed at the log as Merge: <id branch> <id master>
    - Now we want to push to origin master
    - We can now see on the master branch in GitHub that the commit history of landing-page is going to be part of the master branches history
    - So the merging is made between your local master branch and your local landing-page branch and then pushed up to the remote master branch
- Tutorial 2
  - git checkout -b second-feature
  - git add .
  - Git commit -m ”Add another class”
  - Git checkout master
  - Git add .
  - Git commit -m ”Add file to master branch”

    - We have made commits on both branches, and want to avoid a conflict
    - Use rebase!
    - Git checkout second-feature
    - Git rebase master
    - We will now get a message: First, rewinding head to replay your work on top of it…
    - The rebase have then occurred
    - Rebase often to get smaller conflicts in the future
- Fork and pull requests
    - How do we use git in real life in a project?
    - How you use git alone is very different then you do in a team
    - Fork a repository
        - Create a copy of someone else repository linking back to the source
        - This is done on GitHub
        - A copy will then be copied to my GitHub acount
    - Forks a common in open source projects but not that common in regular work place
    - Pull request
        - Start a discussion with the team about a feature and notify them about any progress
        - You ask someone to pull down your code and check so that it is good enough to push to the original project
        - For eg, my project manager asks me to create a new feature. Do all the changes on a personal branch on my local machine. Then I decide that it is ready to ask project lead to add the code to master. Then you push your branch to GitHub and open a pull request.
            - It allows other people in the team to make comments, some my say it is good or not good enough in some way
            - If it is not good enough you change the code and add a new commit to the branch, push again and then the pull request will contain the modified code

        - Pull requests in words:
            - 1. A developer creates a feature dedicated in the local repo
            - 2. The developer pushes the branch to his public GitHub repo
            - 3. The developer files a pull request via GitHub
            - 4. The rest of the team reviews the code, discusses it, and alters it
            - 5. The project maintainer merges the feature into the official repo and closes the pull request
- Work Flow: Gitflow
    - Branch convensions
        - master: should always represent the current state on the live server
            - The master branch is holy, do not update, do not push directly, do not rewrite history on the master branch!!
            - Master branch is supposed to reflect the live version of the product
            - It's from the master branch we are going to deploy the product
        - develop
            - Branch from master in the beginning of the project
            - This is where code will be merged in
            - This is where we develop the project
            - When we make releases we merge the development branch with the master branch
        - feature/<name>
            - Originates from develop branch
            - Developers working on a specific feature
        - release/<version>
            - Originates from develop
            - Used to prepare the code for release while development on the develop branch continues
            - Release branch is used on larger projects, take this with a grain of salt
        - hotfix/<name>

        - The branch that we do not want to see
        - Used for severe bugfixes that prevent from running properly
        - Originates from: master
        - Branch from master, fix the problem merge into master and into develop
        - Used when the fix is very, very urgent
    - Big Picture
        - Maintainer creates main repository locally and then remotely
            - Adds config files like README and gitignore.
        - Developers clone the remote repository
        - Developers work on their own feature branches on their local machine
        - Developer push his branch to remote repo when finished and open a pull request
    - Important! It is the developer's responsibility to ensure their feature branch is rebased on top of the current development/master (depending on workflow) - Especially before opening a PR
        - Meaning, that if you branch of, then people add code to the master and you want to make a pull request. Before making the pull request, always rebase your branch on top of the current version of master/develop **before** making the pull request to avoid conflicts!
    - Other developers or maintainer may approve PR and merge the branch into develop/master
    - Now, the developers must pull latest changes into their local branch
    - Often, beta versions are deployed from the development branch
- Commands to look at:
    - git diff
    - git diff —staged
    - git log —oneline —decorate —graph —all
- In GitHub
    - Issues is nice to use. You can eg add a new issue to explain a new feature that would be needed in the project

- In Projects, you can add issues in an Trello like board