Git Workflow for Collaborative Development

Creating a new project

One team member (the Git Auth) makes a GitHub repository:

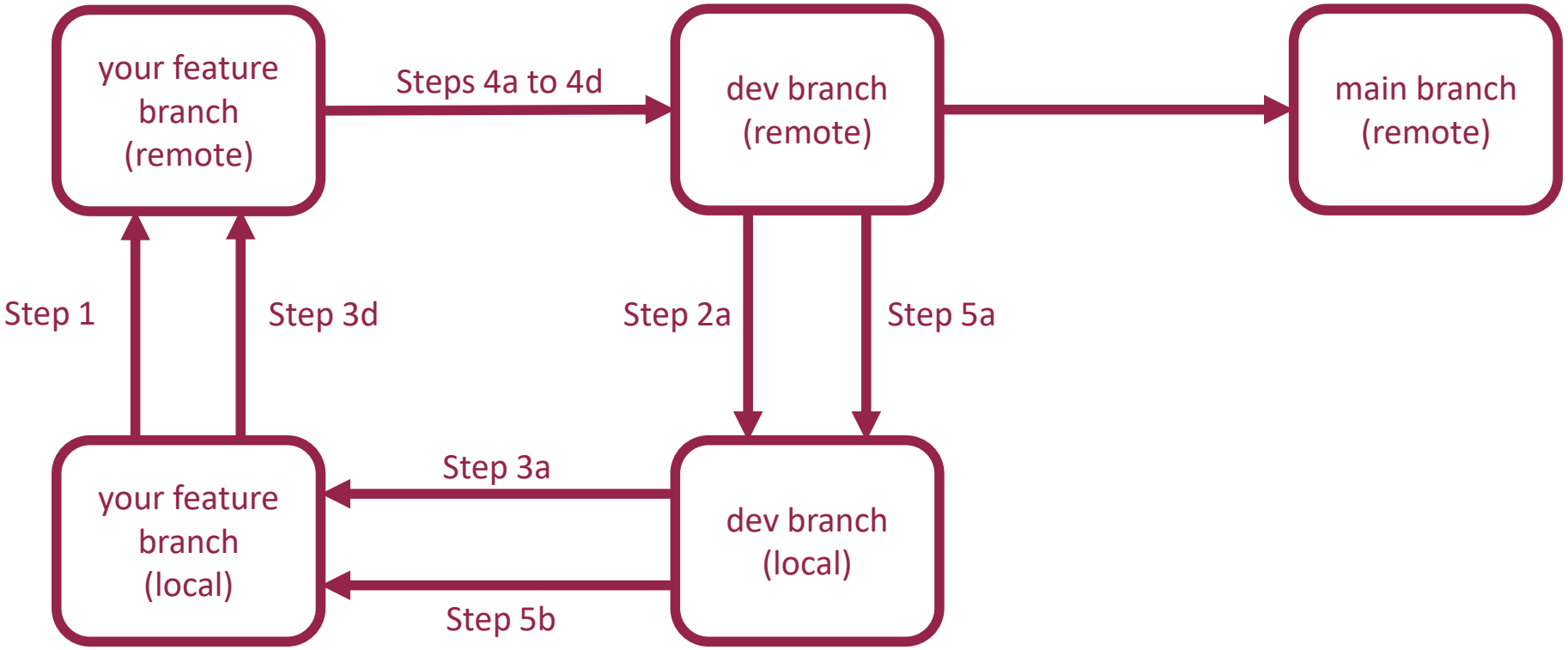
- Go to your GitHub account and select the Repositories page
- Click the New button to make a new repository
- Add a README and a .gitignore file with the appropriate template
- Go to the repository on your GitHub account and select the Settings tab
- Manage access – add your collaborators to the project
- Branches:
  - You should be on the main branch
  - Add a branch protection rule with the same name (main)
  - Select “Require a pull request before merging”, “Require approvals” and 2 approvals before merging and save changes
  - Create a dev branch based on the main branch
    - You will normally merge your code changes into the dev branch and push this code to main at intervals when you have a stable build (eg after each sprint)
  - Update the default branch to dev
  - Add a branch protection rule for the dev branch, normally like the one for main

Working on the project

- One pair of developers can set up the initial code for the project and folder structure and push it to dev and main before adding the protection rules
- Team members can now clone the project locally
- Do NOT edit the code in the main branch or dev branch, instead:
  - Pull the dev branch to your local machine
  - Create a feature branch locally; give it your own name, and/or the name of the feature you are working on (eg woody_roundup)
  - git checkout –b woody_roundup
  - Develop your code on this branch

Seeing remote branches

- If you can’t see someone else’s remote branch:
  - Open a command window in the top directory of your solution
  - git status – should show you which branch you are on and if there is anything to commit
  - git fetch origin – fetches references to any new remote branches
  - You still need to pull down a local copy of a branch if you want to use it.



See next page for step descriptions

Step 1

Commit the changes in your local branch regularly, and push them to remote

Step 2

When you are ready to merge to dev, first update your feature branch to include the latest changes from dev:
a) Locally, switch to dev branch and sync with the server (git pull)
b) If there are any merge conflicts, resolve them and commit locally

Step 3

Merge the latest changes with your feature branch:
a) Locally switch to your feature branch and merge from dev
b) If there are any merge conflicts, resolve them and commit locally
c) Check everything still builds and tests pass; commit any changes
d) Push your branch to remote

Step 4

Pull request:
a) Create a pull request
b) Fill in the code reviewer(s)
c) The reviewer may add comments and may approve the request
d) If approved and no changes, complete the pull request
e) If changes required, make them locally, make sure the code still compiles and runs, commit and push to remote; complete pull request once approved
f) If you are finished with this feature branch, delete it

Step 5

Update your local dev branch:
a) Switch to dev and sync with the server (git pull)
b) Switch back to your feature branch and merge from local dev, or create a new branch for your next piece of work, based on local dev