School of Computing & Communications | Lancaster University

SCC.111 Software Development – Lecture 33: Version Control Workflows and CI

Adrian Friday, Hansi Hettiarachchi and Nigel Davies

Introduction

- In the last lecture, we looked at:
  - How **version control** can promote collaboration
  - Practical guidance on using **Git** version control
  - Examples of a typical Git Workflow

- Today we're going to further explore **open-source** version control concepts:

  - Merge conflicts and resolution
  - Version branching
  - Repository forks
  - Merge requests
  - Issue tracking
  - CI/CD: Continuous Integration/Continuous Deployment

Version Control Workflow

Lancaster University

We have seen how to sync local changes to a hosted repo using git workflow

- Clone: **git clone** https://scc-source.lancs.ac.uk/username/reponame
- We add the useful changes we want to "upload" to the repo: **git add *.java**
- We commit the changes**: git commit –m "commit message"**
- We **push** those commit back to the server: **git push origin main**
- We **pull** changes made by others into our local repo from the server: **git pull**

“Issues”

Lancaster University

We can also use issues to track known bugs and feature requests

- Free text description of a known problem
- Anyone with access to a repo can raise **an issue**
- Anyone with access can comment on **an issue**
- Provide a helpful way for users to **report bugs** to you

Issues are often used to synchronize workflow between multiple developers

- Issues can be **assigned** to developers… a great way to organize!
- Issues can be **closed** when they **are complete**.
- They often also refer to individual commits where the change was made
- Very powerful audit history of your project.

Merge conflicts

If someone else has updated file on the server as you have edited locally…

- This is called **a merge conflict**
- Git will need *some help* from you to resolve this.

- Any files in conflict will be updated with **BOTH** copies of lines of code that differ
- Fix this conflict by ether choosing the versions of the lines you want
- Or by editing the code until it is correct.

- Then add these changed files that were in conflict and commit.

```
git add
git commit
```

Repository Branches

Lancaster University

Branches help to support longer running changes.

- Every repository has a main branch – the definitive 'true' version of your code.
- Any changes you make aim to ultimately update this branch.
- You can directly work on and commit to the the main branch (as we've seen).

- Sometimes this is detrimental when working on larger, longer running changes.
- Members of teams must either:
  - push partial changes to the main branch that others are using
  - keep their updates locally on their laptop and not push it for a long time.

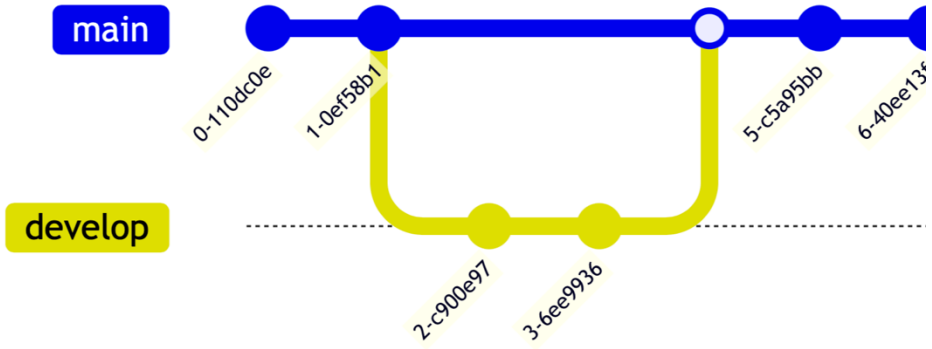
- **Neither of these options are good for stability and safety of the codebase!**

Repository Branches…

Developers are free to create a branch of a repo at any point

- A branch is like a named "sub-copy" of the repo that diverges at a given commit
- This branch can then have new commits pushed to it independently of the main branch, ensuring work is always backed up.
- At a later point in time, when the code development in complete, the branch can be merged back into main.

- **When adding a new feature, developers normally create a branch to hold their changes.**



Working with Branches

Lancaster University

List branches in a repo:

```
git branch
```

Create a new branch: (some-feature can be any name you like)

```
git branch some-feature
```

Switch to a different branch:

```
git checkout some-feature
```

Push a branch to the server:

```
git push origin some-feature
```

Merge two branches:

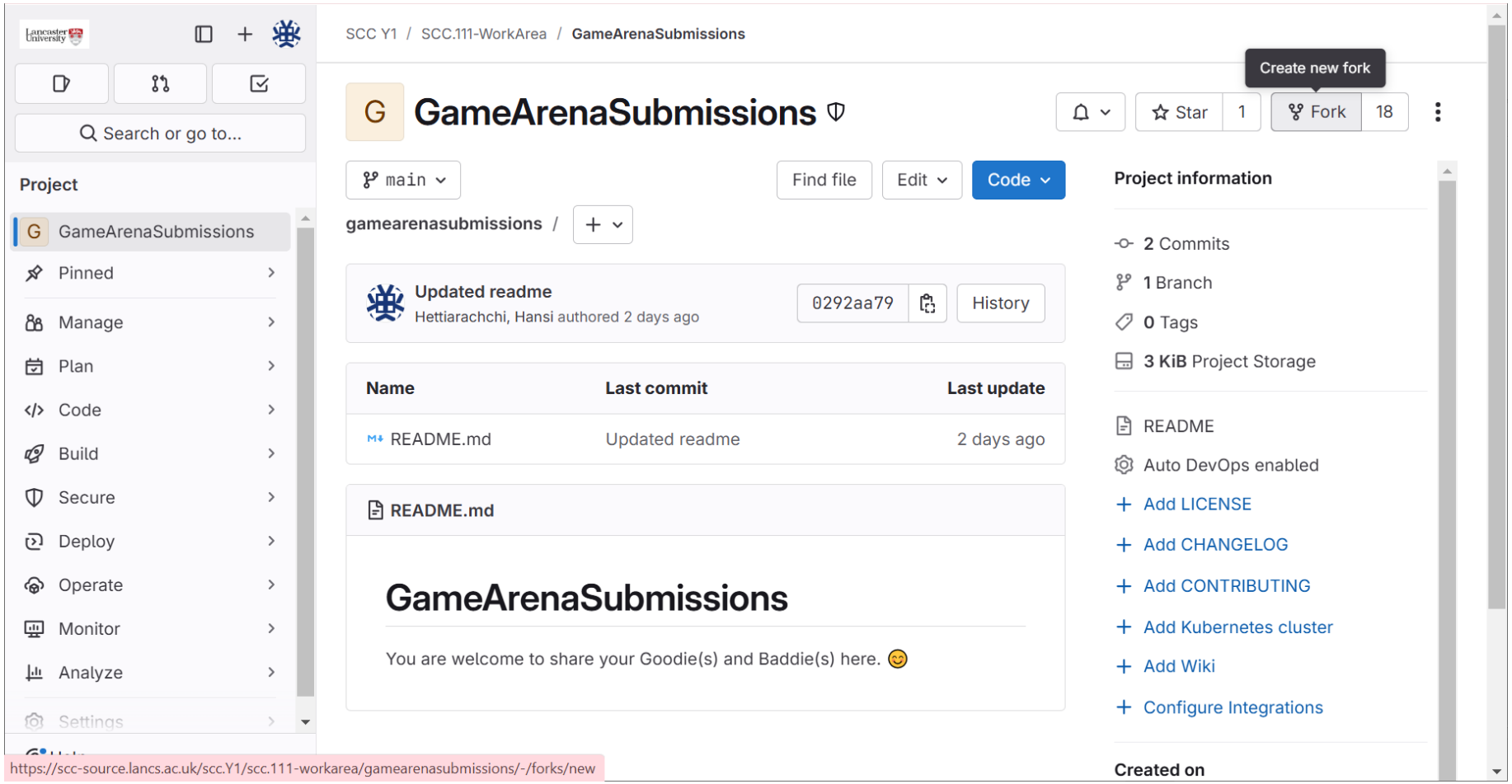
```
git checkout main
git merge some-feature
```

Fork a Repository

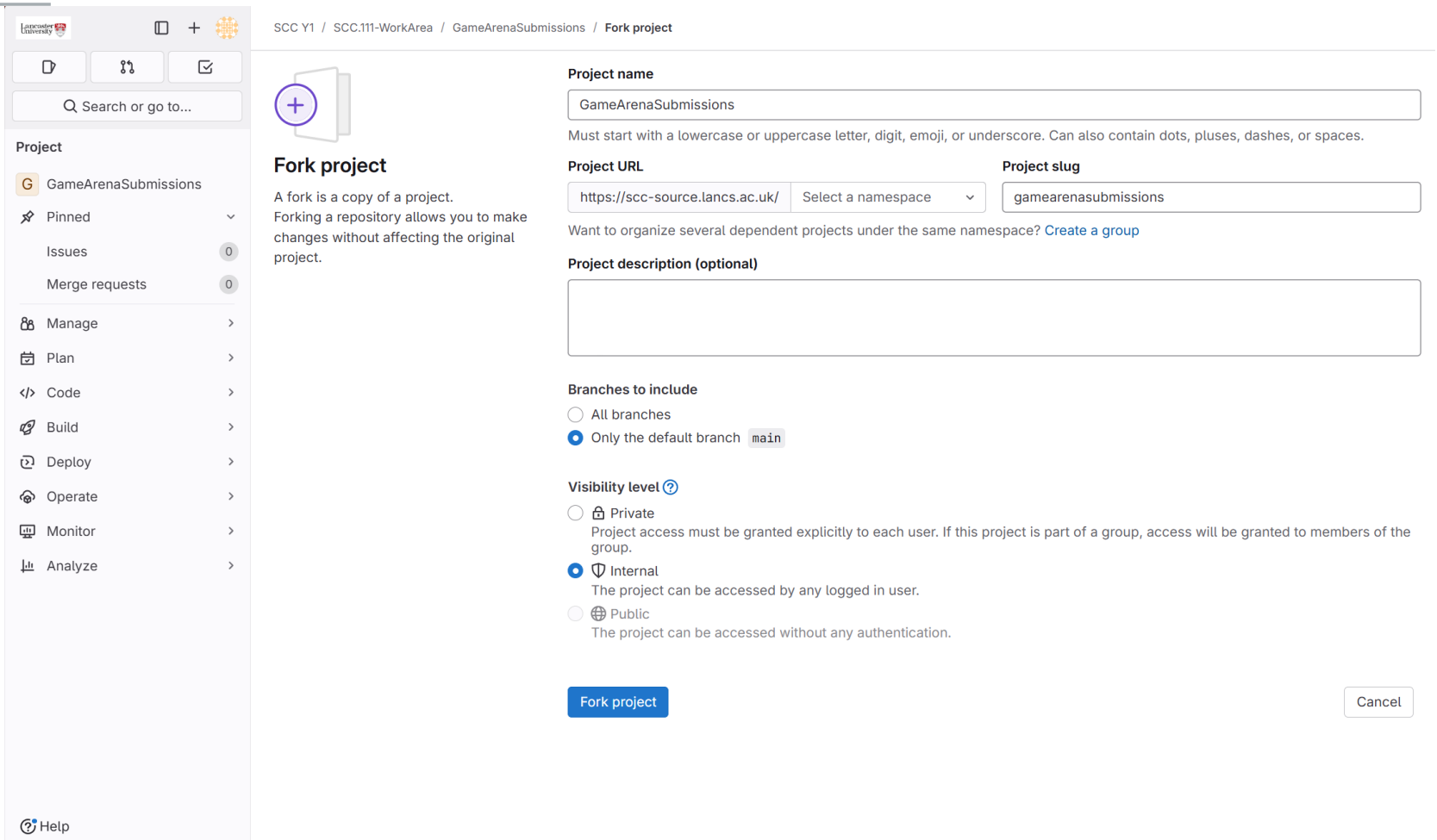
A fork is a deep copy of a repository

- Allows developers to experiment with changes without affecting the original project.
- The fork is created on the developer's own account/organization…
- …**not** the account/organization hosting the original repo.
- This fork can then host new branches, commits etc like any other repo.

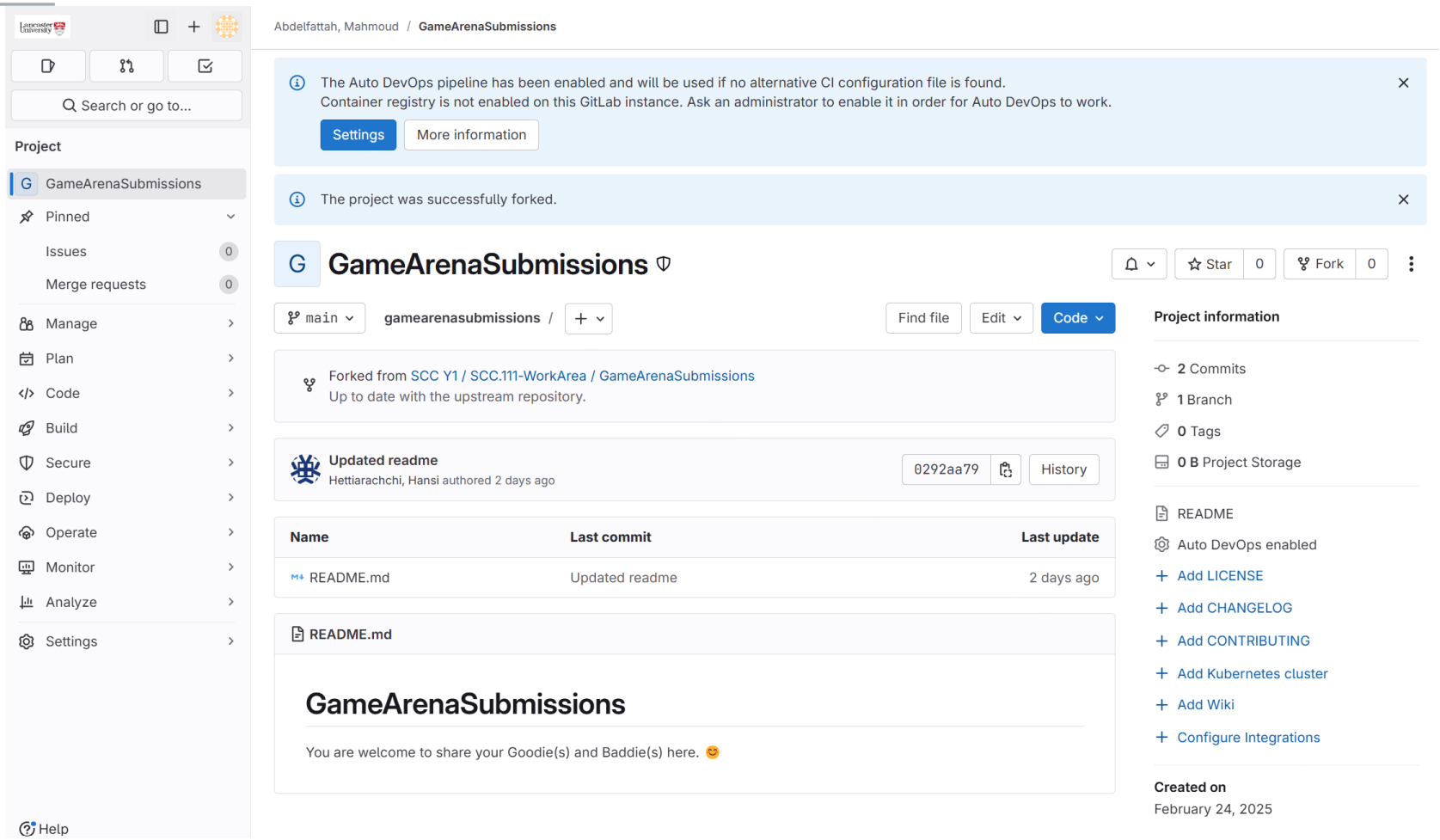
Fork Example



Fork Example….



Fork Example….



Merge Requests

Lancaster University

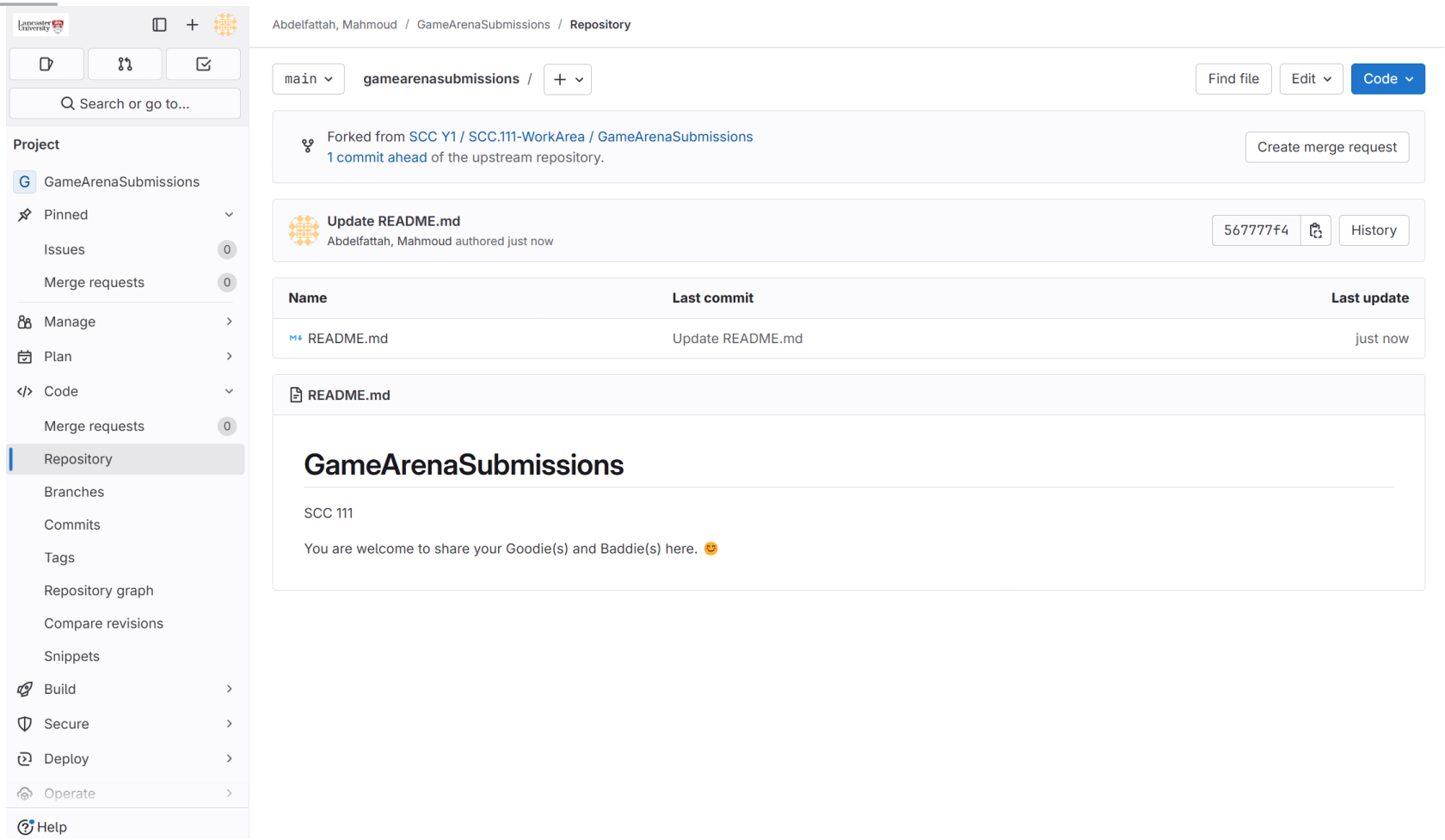
Sometimes we want to accept code from people we don't know!

- Especially in open-source projects
- People you don't trust enough to be collaborators on your project may still have valuable contribution to make…
- But such code would need review and control before it is merged
- GitLab provide Merge Requests for this purpose…

Anyone with a fork of a repo can request one of the branches on that fork be merged

- At that point the repo collaborators can review the changes
- Communicate with the person requesting the update
- Accept or reject the request – all through the website

Merge Request Example



Merge Request Example….

Lancaster University

Abdelfattah, Mahmoud / GameArenaSubmissions / Merge requests / New

Search or go to...

Project

GameArenaSubmissions

Pinned

Issues 0

Merge requests 0

Manage

Plan

Code

Build

Secure

Deploy

Operate

Monitor

Analyze

Settings

Help

New merge request

From `abdelfa2/gamearenasubmissions:main` into `scc.Y1/scc.111-workarea/gamearenasubmissions:main` Change branches

Title (required)

Update README.md

Mark as draft
Drafts cannot be merged until marked ready.

Description

Preview

```
Updated readme file and added "SCC 111" and spaces.
```

Switch to rich text editing

Merge options

Squash commits when merge request is accepted.

Contribution

Allow commits from members who can merge to the target branch. About this feature.
Not available for protected branches

Create merge request Cancel

Commits 1 Changes 0

Feb 24, 2025

Update README.md
Abdelfattah, Mahmoud authored just now

567777f4

Continuous Integration (CI)

Lancaster University

Definition:

- **CI** is a software development practice where developers **regularly integrate** their code changes into a central repository.
- Each integration is verified by an **automated build** and **automated tests**

Benefits:

- Detect and fix integration issues **early**.
- Ensure code quality and reliability.

CI Workflow + example

CI workflow steps:

- Define the CI workflow: environment, building and testing variables, etc.

1. Developer **pushes changes** to the version control system (git)
2. **CI server** detects the changes and **triggers** an automated build and test process
3. Results are reported back to the developer

Let's look at a Gitlab example

Continuous Deployment (CD)

CD is the next step in the CI/CD pipeline, where the code that passes CI is automatically **deployed to a beta production environment**.

- Goal: **automate** the deployment **process**, **reducing manual** intervention
- Result: **faster** and more **reliable** software delivery

Summary

Lancaster University

Today we learned about:

- Merge conflicts and resolution
- Version branching
- Repository forks
- Merge requests
- Issue tracking
- CI/CD: Continuous Integration/Continuous Deployment

Some open-source repos you might find interesting:

https://github.com/lancaster-university/infolab-lights/

https://github.com/lancaster-university/microbit-v2-samples

https://github.com/openjdk/jdk

https://github.com/finneyj/GameArena