

SCC.111 Software Development – Lecture 32: Collaborative Workflows

Adrian Friday, Hansi Hettiarachchi and Nigel Davies

Introduction

- In the last few lectures, we looked at:
 - Component organisation in Java using **Layout Managers**
 - A real examples of **composition** in action
 - Asynchronous programming
- Today we're going to investigate how OO design also helps to promote efficient and safe collaborative software development...
- **We'll also see how version control tools can help to facilitate this**

Version Control Refresher...

In term 1 we saw how a version control tool called git helps to provide resilience

- Git repositories (repos) are typically stored on a server somewhere
- Repos are then **cloned** onto other computers as needed (e.g. your laptop)
- All changes made within a repo are automatically tracked.
- When we have made some useful change (fix a bug, add a new feature...), we **commit**.
- As frequently as we like we **push** those commits back to the server
- As frequently as we like we **pull** changes into our local repo from the server
- All versions of your code are stored, so no work is ever truly lost

Today we're going to see that in action.

- We are going to collaboratively create some software using the GameArena classes from this week's lab. **Watch and learn.** 😊

Creating a Repository

The simplest way is to do this interactively via your favourite provider...

- GitHub and GitLab are some of the most popular
- We will use GitLab today 😊
- Sign up if you haven't already done so.



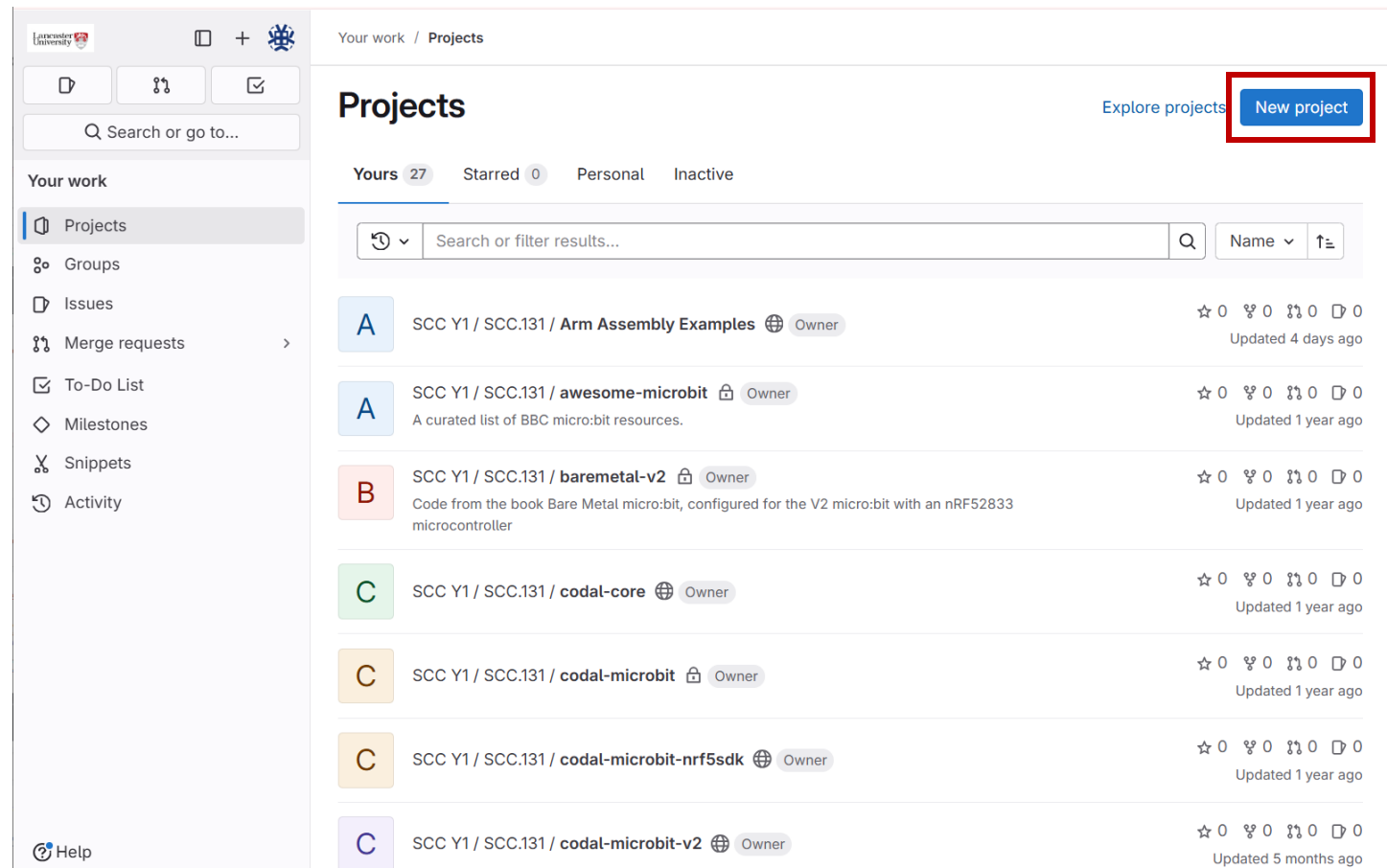
<https://scc-source.lancs.ac.uk/>

GitLab @ SCC



Lancaster University Login

Creating a Repository










Your work / Projects

Projects

Explore projects [New project](#)

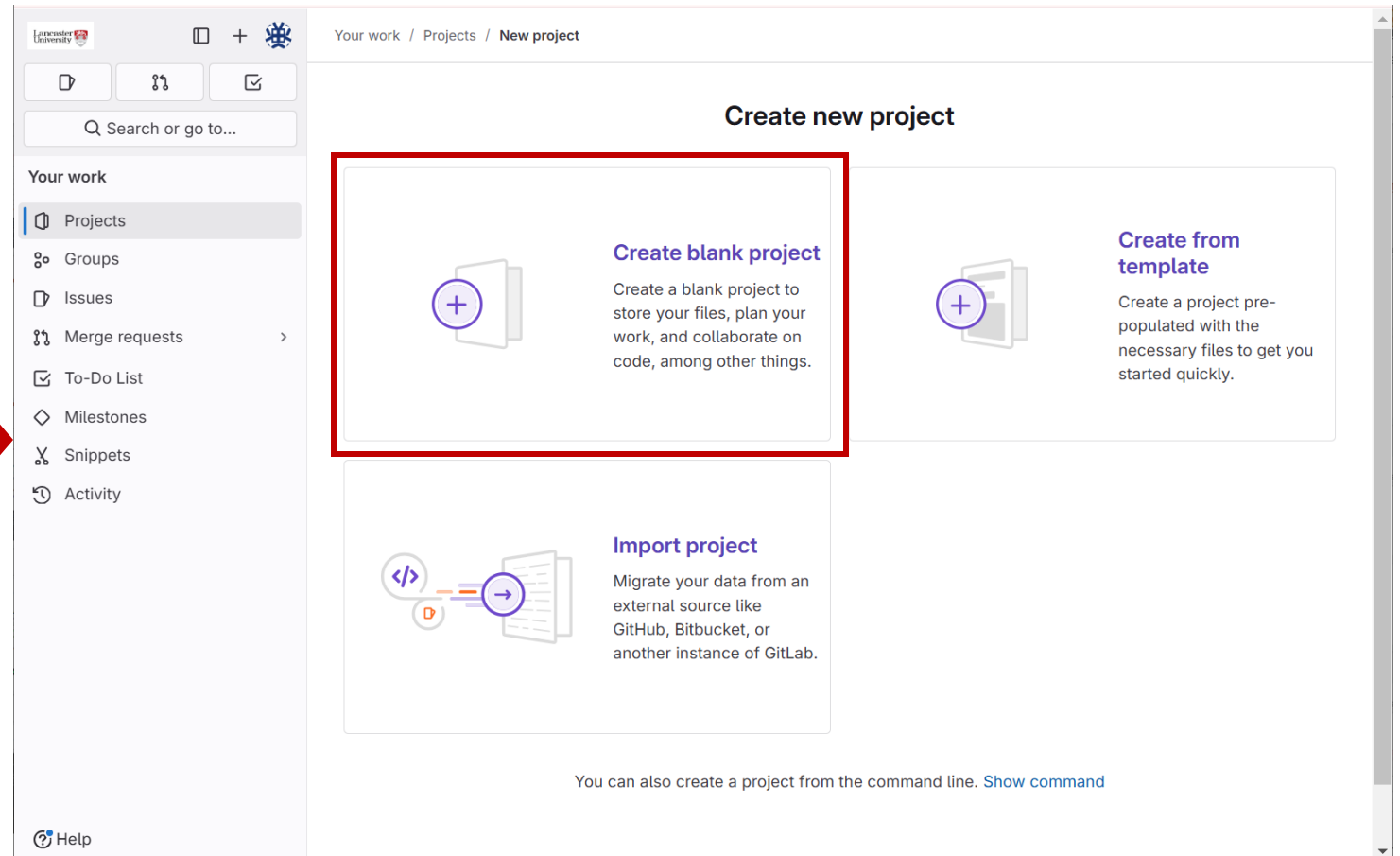
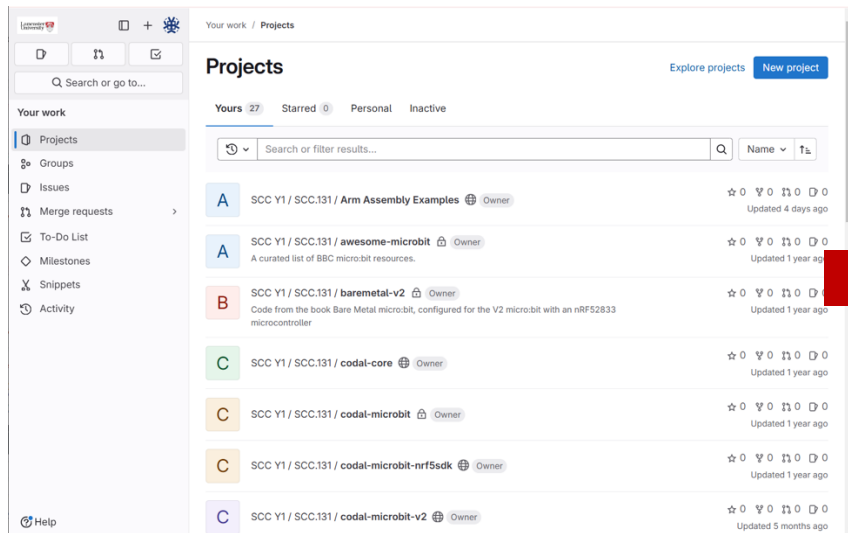
Yours 27 **Starred** 0 **Personal** **Inactive**

Search or filter results... **Name**

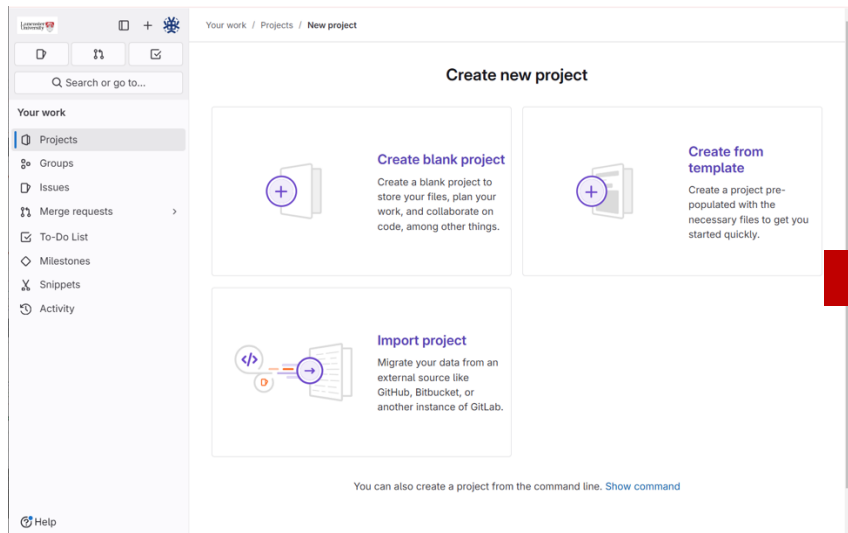
A	SCC Y1 / SCC.131 / Arm Assembly Examples  Owner	☆ 0 🍴 0 📄 0 📌 0 Updated 4 days ago
A	SCC Y1 / SCC.131 / awesome-microbit  Owner A curated list of BBC micro:bit resources.	☆ 0 🍴 0 📄 0 📌 0 Updated 1 year ago
B	SCC Y1 / SCC.131 / baremetal-v2  Owner Code from the book Bare Metal micro:bit, configured for the V2 micro:bit with an nRF52833 microcontroller	☆ 0 🍴 0 📄 0 📌 0 Updated 1 year ago
C	SCC Y1 / SCC.131 / codal-core  Owner	☆ 0 🍴 0 📄 0 📌 0 Updated 1 year ago
C	SCC Y1 / SCC.131 / codal-microbit  Owner	☆ 0 🍴 0 📄 0 📌 0 Updated 1 year ago
C	SCC Y1 / SCC.131 / codal-microbit-nrf5sdk  Owner	☆ 0 🍴 0 📄 0 📌 0 Updated 1 year ago
C	SCC Y1 / SCC.131 / codal-microbit-v2  Owner	☆ 0 🍴 0 📄 0 📌 0 Updated 5 months ago

[Help](#)

Creating a Repository



Creating a Repository



Your work / Projects / New project / Create blank project

Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL **Project slug**

<https://scc-source.lancs.ac.uk/> /

Visibility Level ?

☒ **Private**
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ **Internal**
The project can be accessed by any logged in user except external users.

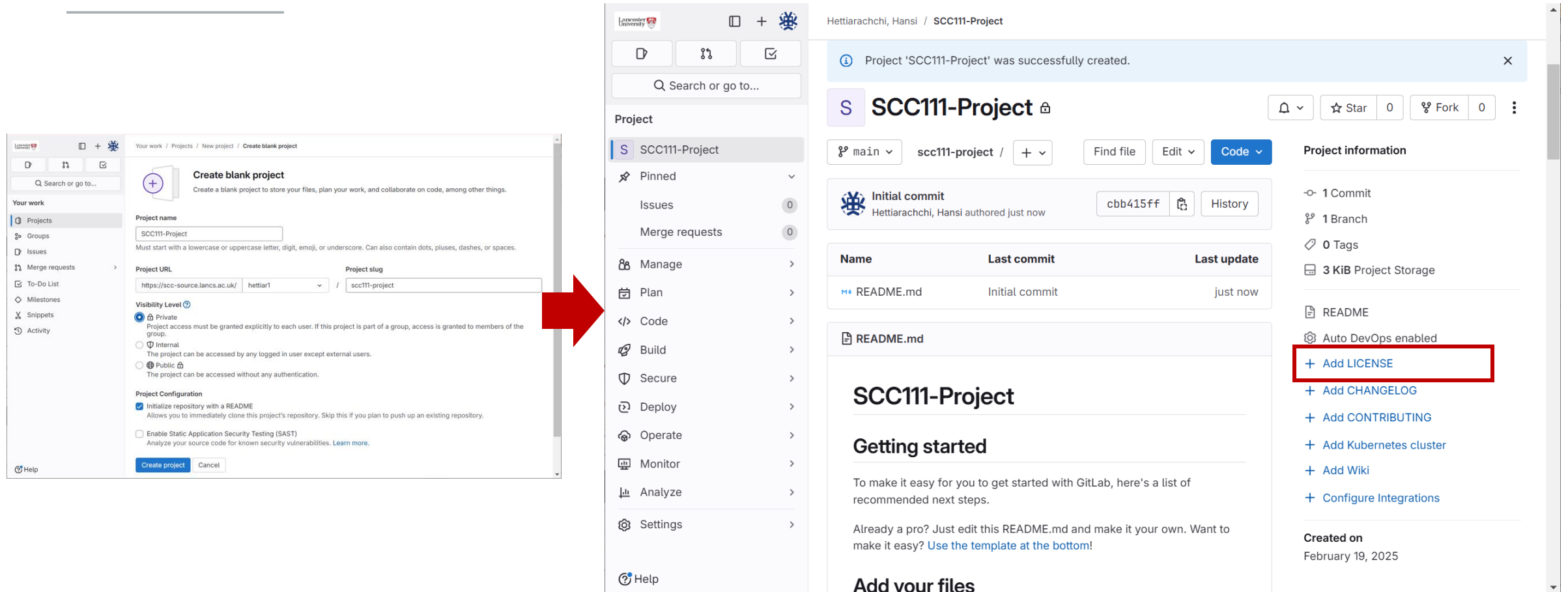
☐ **Public**
The project can be accessed without any authentication.

Project Configuration

☒ **Initialize repository with a README**
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ **Enable Static Application Security Testing (SAST)**
Analyze your source code for known security vulnerabilities. [Learn more.](#)

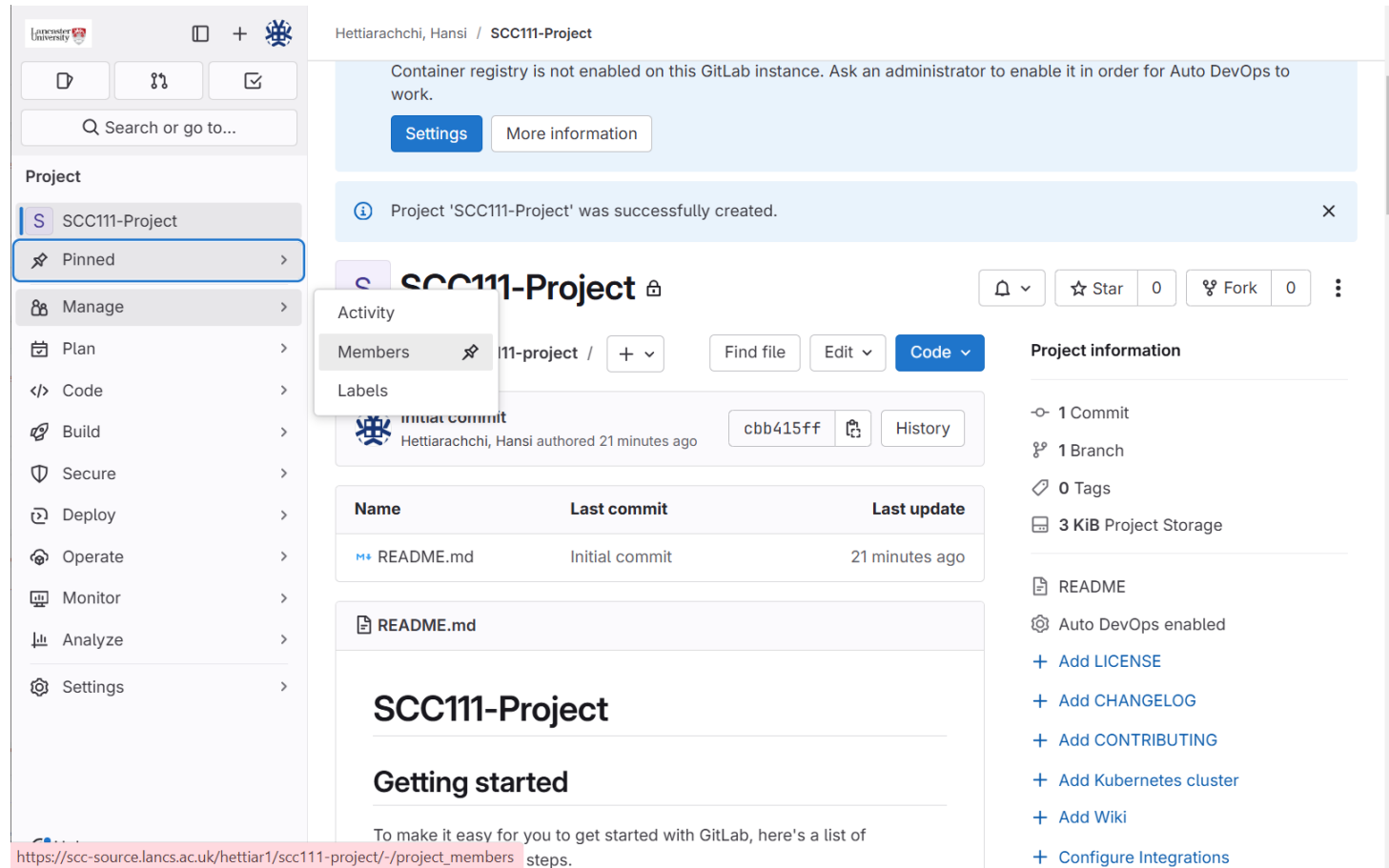
Creating a Repository



The screenshot illustrates the process of creating a repository in GitLab. It shows three main components:

- Create blank project dialog:** A form for creating a new project. The 'Project name' is 'SCC111-Project'. The 'Project URL' is 'https://scc-source.lancs.ac.uk/'. The 'Project slug' is 'scc111-project'. The 'Visibility Level' is set to 'Private'. The 'Project Configuration' section includes options for 'Initialize repository with a README' (checked), 'Enable Static Application Security Testing (SAST)' (unchecked), and 'Auto DevOps enabled' (checked).
- SCC111-Project sidebar:** A sidebar for the 'SCC111-Project' showing various project management options: Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, and Settings.
- SCC111-Project details page:** The main page for the 'SCC111-Project'. It shows the 'Initial commit' by 'Hettiarachchi, Hansi' with the commit hash 'cbb415ff'. The 'Project information' section lists '1 Commit', '1 Branch', '0 Tags', and '3 KiB Project Storage'. The 'Add LICENSE' button is highlighted with a red box.

Adding Contributors/Members



Hettiarachchi, Hansi / SCC111-Project

Container registry is not enabled on this GitLab instance. Ask an administrator to enable it in order for Auto DevOps to work.

Settings More information

Project 'SCC111-Project' was successfully created.

SCC111-Project

Activity Members Labels

Initial commit
Hettiarachchi, Hansi authored 21 minutes ago cbb415ff History

Name	Last commit	Last update
README.md	Initial commit	21 minutes ago

README.md

SCC111-Project

Getting started

To make it easy for you to get started with GitLab, here's a list of steps.

Project information

- 1 Commit
- 1 Branch
- 0 Tags
- 3 KiB Project Storage
- README
- Auto DevOps enabled
- [Add LICENSE](#)
- [Add CHANGELOG](#)
- [Add CONTRIBUTING](#)
- [Add Kubernetes cluster](#)
- [Add Wiki](#)
- [Configure Integrations](#)

https://scc-source.lancs.ac.uk/hettiar1/scc111-project/-/project_members

Adding Contributors/Members

Project

SCC111-Project

Pinned

Manage

Plan

Code

Build

Secure

Deploy

Operate

Monitor

Analyze

Settings

Container registry is not enabled on this GitLab instance. Ask an administrator to enable it in order for Auto DevOps to work.

Project 'SCC111-Project' was successfully created.

SCC111-Project

Activity

Members

Labels

Find file

Edit

Code

Project information

1 Commit

1 Branch

0 Tags

3 KiB Project Storage

README

Auto DevOps enabled

Add LICENSE

Add CHANGELOG

Add CONTRIBUTING

Add Kubernetes cluster

Add Wiki

Configure Integrations

README.md

Getting started

To make it easy for you to get started with GitLab, here's a list of steps.

Project

SCC111-Project

Pinned

Manage

Activity

Members

Labels

Plan

Code

Build

Secure

Deploy

Operate

Monitor

Analyze

Settings

Help

Hettiarachchi, Hansi / SCC111-Project / Members

Project members

Import from a project

Invite a group

Invite members

You can invite a new member to SCC111-Project or invite another group.

Members 1

Filter members


Account

Source

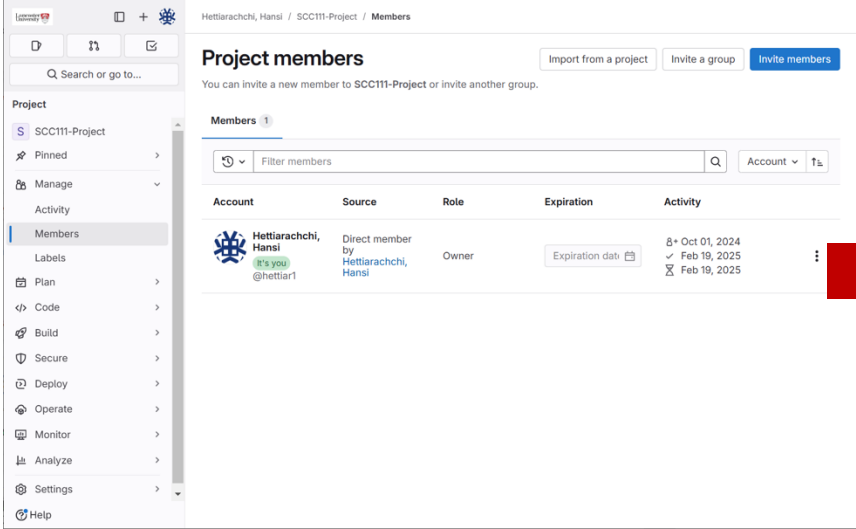
Role

Expiration

Activity

 Hettiarachchi, Hansi It's you @hettiar1	Direct member by Hettiarachchi, Hansi	Owner	Expiration date	8+ Oct 01, 2024 ✓ Feb 19, 2025 ✗ Feb 19, 2025	
-----------------------------------------------------------------------------------------------------------------------------------	---------------------------------------	-------	-----------------	-----------------------------------------------------	--



Adding Contributors/Members

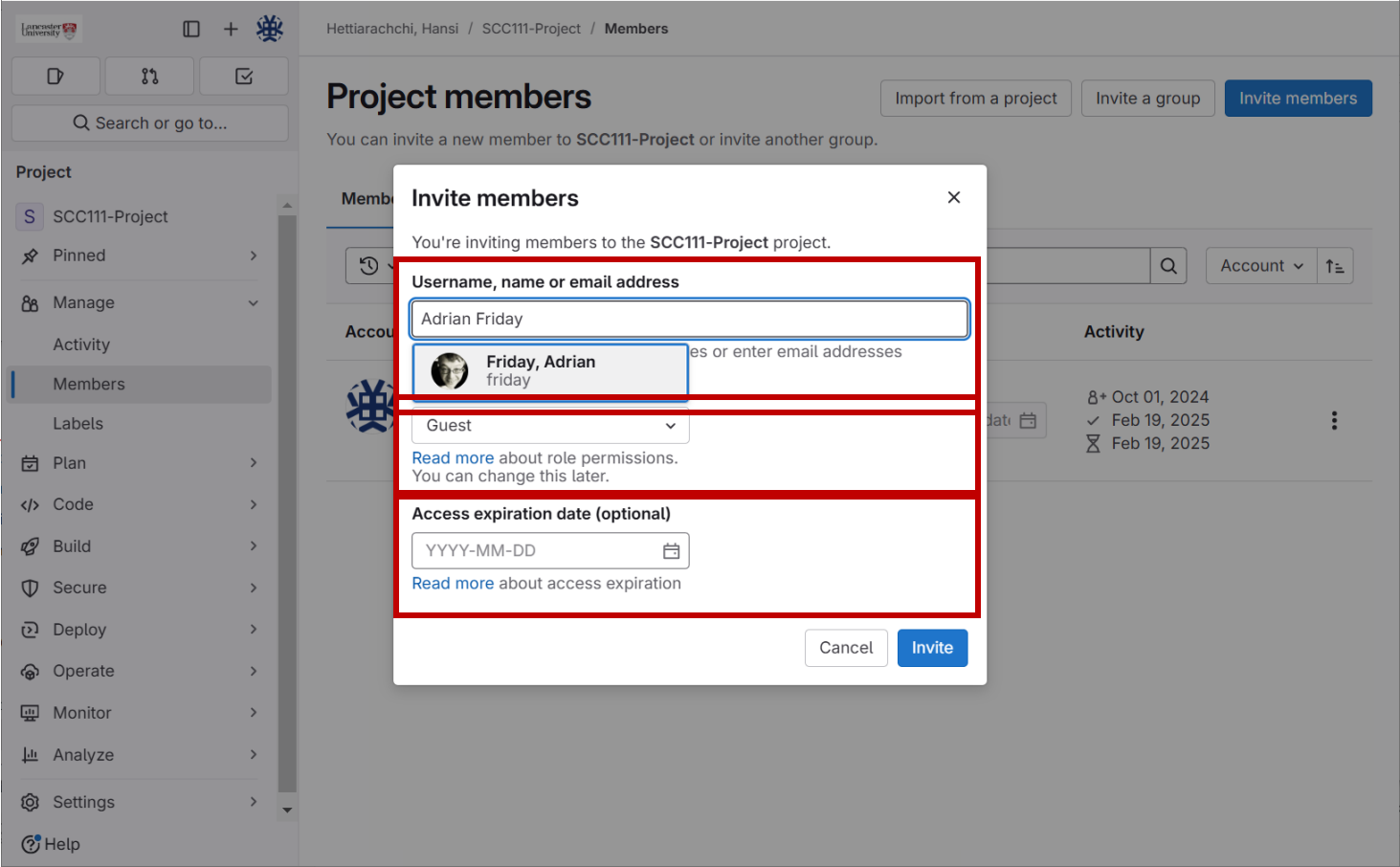


Project members

You can invite a new member to SCC111-Project or invite another group.

Members 1

Account	Source	Role	Expiration	Activity
 Hettiarachchi, Hansi @hettiar1	Direct member by Hettiarachchi, Hansi	Owner	Expiration date 	8+ Oct 01, 2024 ✓ Feb 19, 2025 ✗ Feb 19, 2025



Project members

You can invite a new member to SCC111-Project or invite another group.


Import from a project Invite a group **Invite members**

Invite members

You're inviting members to the SCC111-Project project.

Username, name or email address


Adrian Friday

 Friday, Adrian
friday

Guest

[Read more](#) about role permissions. You can change this later.

Access expiration date (optional)

YYYY-MM-DD 

[Read more](#) about access expiration

Cancel **Invite**

Accessing a repository for the first time

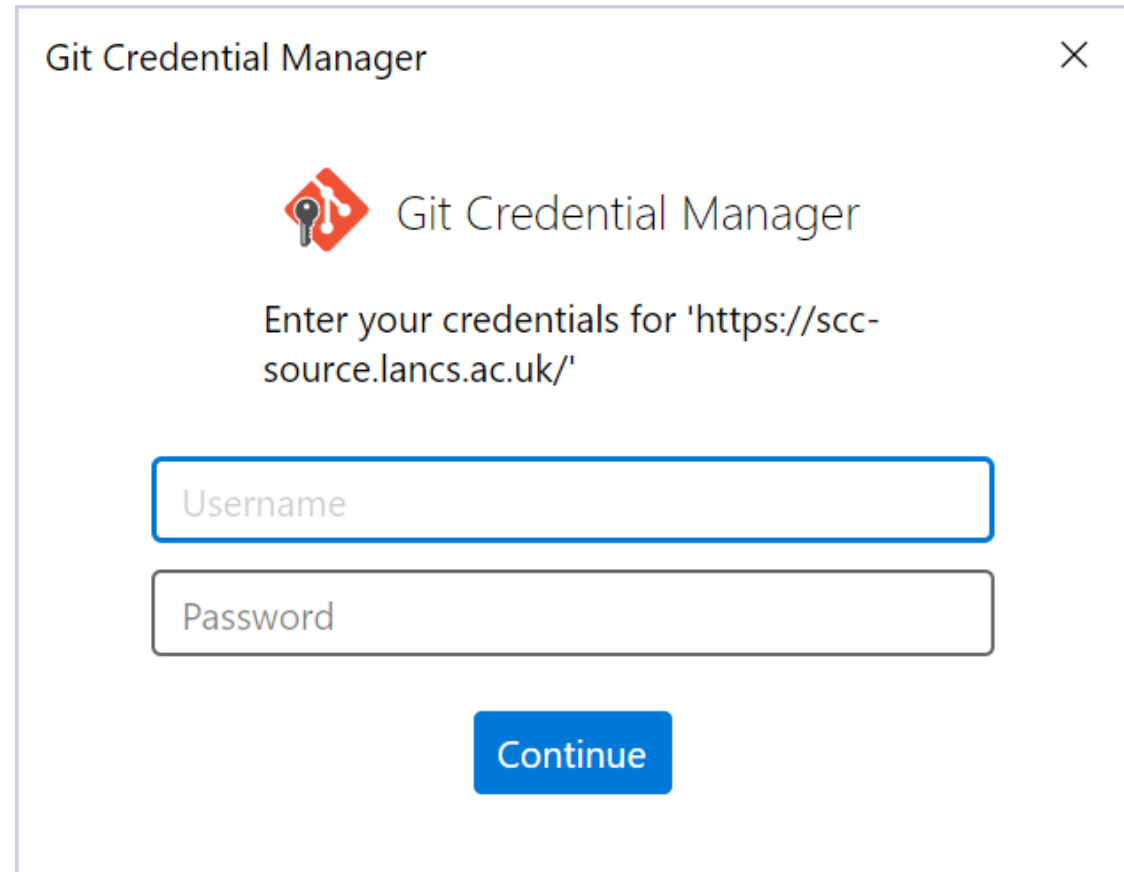
Once there is a remote repo, we often want a local copy to work on...

- There is a standard command line tool called git.
- There are also GUI tools, and VS Code integration.
- The command line tool provides more control over your work, so is good to learn.
- **We recommend you learn the command line tool in this course.**
- To take a local copy of a git repo, we use the **git clone** command:

```
git clone https://scc-source.lancs.ac.uk/hettiar1/scc111-project.git
```

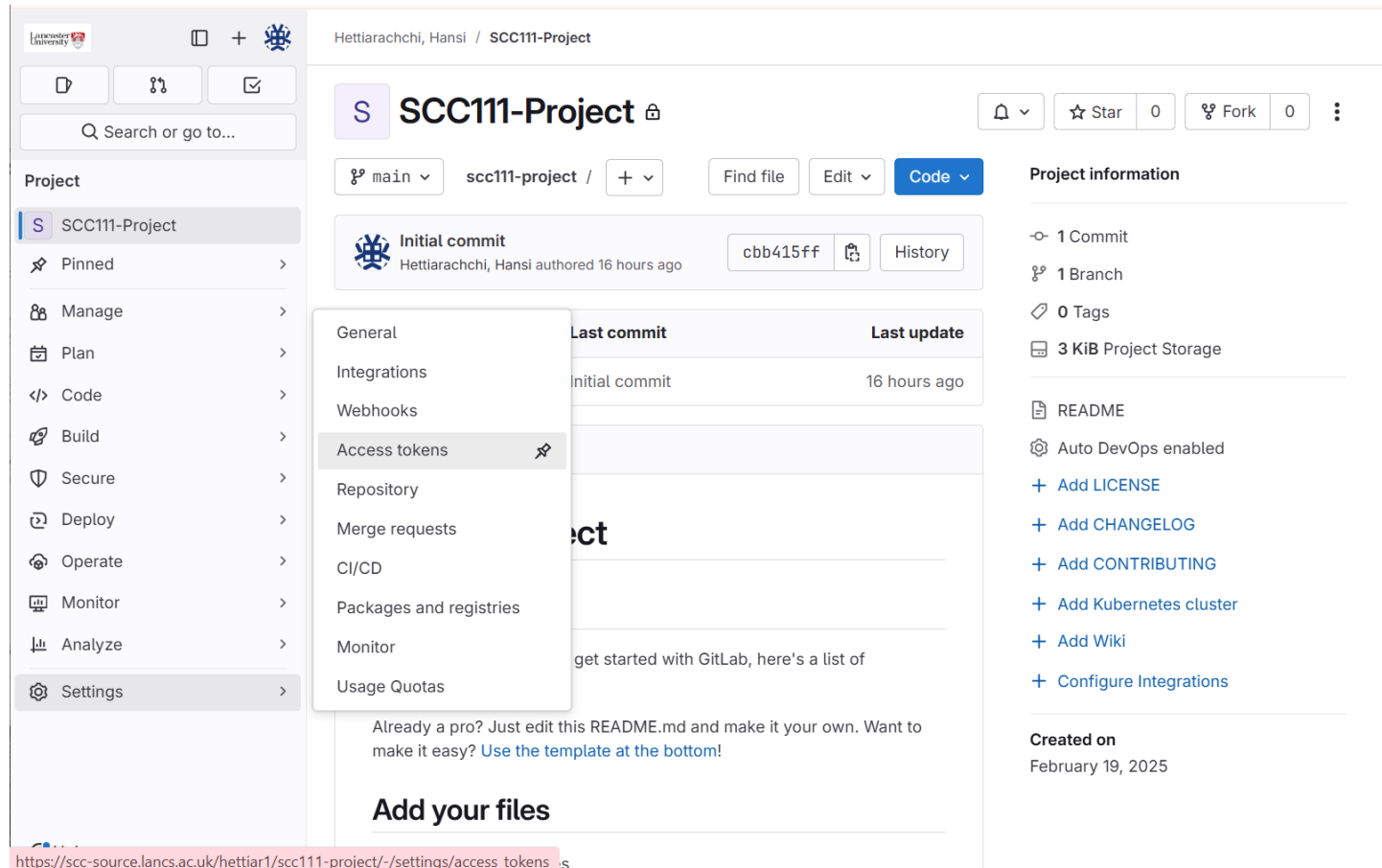
Accessing a private repository

- If your repository is private, you need to provide your credentials.
 - It is recommended providing an access token as the password.
- ❖ Access tokens are similar to passwords, except you can limit access to resources, select a limited role, and provide an expiry date.



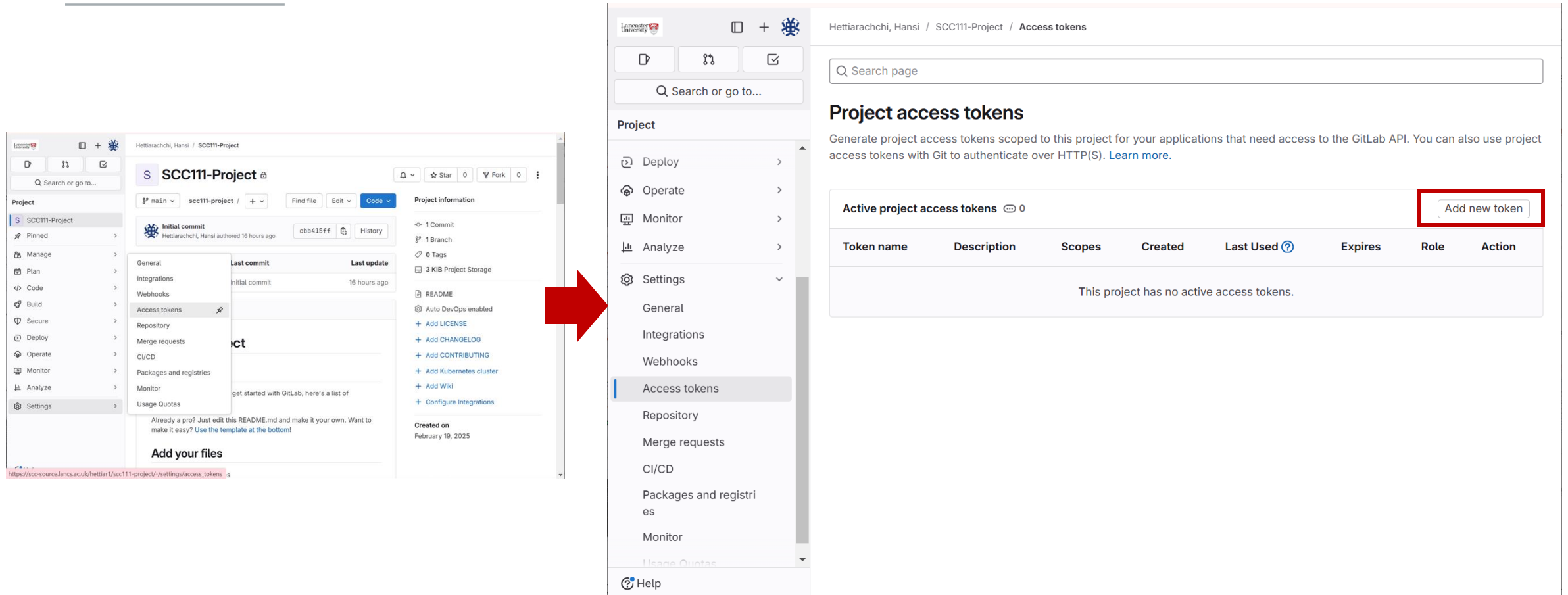
The screenshot shows a 'Git Credential Manager' window. At the top, it says 'Git Credential Manager' with a close button. Below that is the Git logo and the text 'Git Credential Manager'. The main instruction is 'Enter your credentials for 'https://scc-source.lancs.ac.uk/'. There are two input fields: 'Username' and 'Password'. At the bottom right is a blue 'Continue' button.

Creating a project access token



The screenshot shows the GitLab interface for the 'SCC111-Project' repository. The left sidebar contains a 'Project' section with a search bar and a list of project settings: Pinned, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The 'Settings' option is highlighted. A dropdown menu is open from 'Settings', showing options: General, Integrations, Webhooks, Access tokens (highlighted with a gear icon), Repository, Merge requests, CI/CD, Packages and registries, Monitor, and Usage Quotas. The main content area shows the repository details, including the 'Initial commit' by Hettiarachchi, Hansi 16 hours ago. The 'Project information' sidebar on the right lists: 1 Commit, 1 Branch, 0 Tags, 3 KiB Project Storage, README, Auto DevOps enabled, and links to Add LICENSE, Add CHANGELOG, Add CONTRIBUTING, Add Kubernetes cluster, Add Wiki, and Configure Integrations. The 'Created on' date is February 19, 2025. The URL at the bottom is https://scc-source.lancs.ac.uk/hettiar1/scc111-project/-/settings/access_tokens.

Creating a project access token



The screenshot illustrates the steps to create a project access token in GitLab. On the left, the 'Settings' menu is open, and 'Access tokens' is selected. A red arrow points to the 'Project access tokens' page on the right. In this page, the 'Add new token' button is highlighted with a red box.

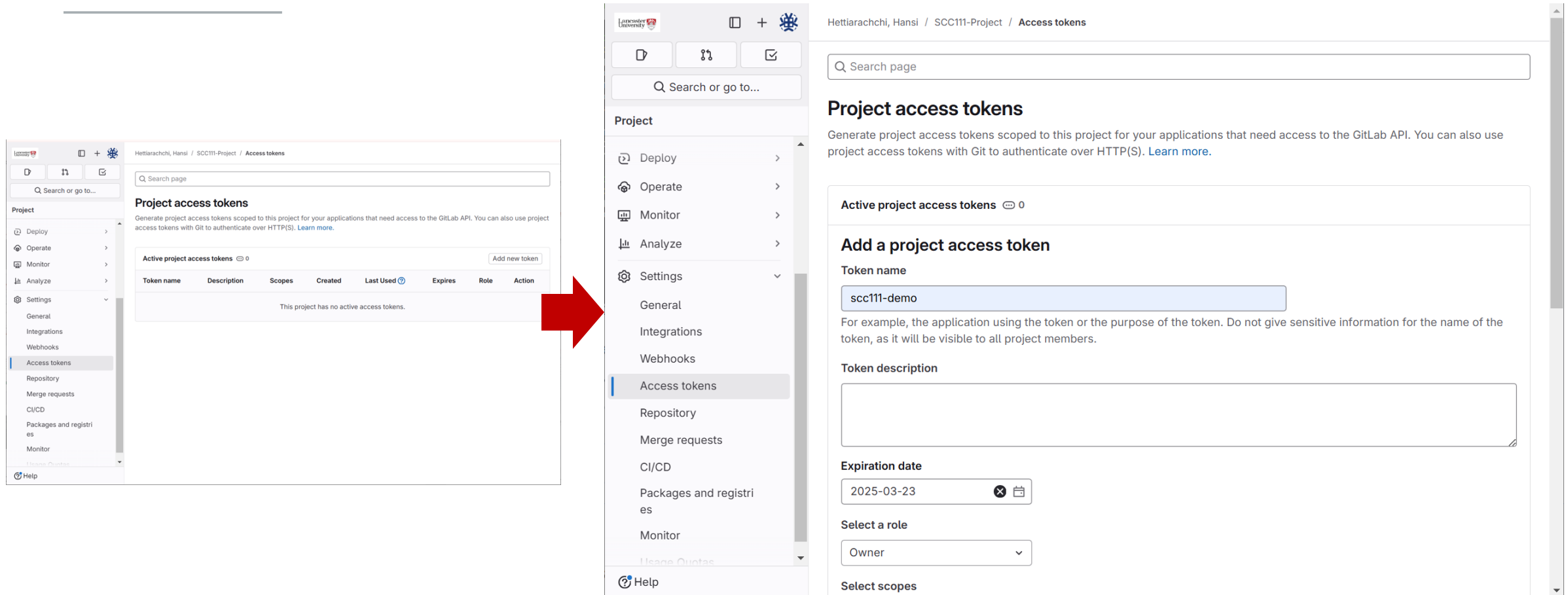
Project access tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API. You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Active project access tokens 0 Add new token

Token name	Description	Scopes	Created	Last Used ?	Expires	Role	Action
This project has no active access tokens.							

Creating a project access token



The screenshot shows the GitLab interface for a project named 'SCC111-Project'. The left sidebar contains a 'Project' menu with options: Deploy, Operate, Monitor, Analyze, Settings, General, Integrations, Webhooks, Access tokens (highlighted), Repository, Merge requests, CI/CD, Packages and registries, and Monitor. The main content area is titled 'Project access tokens' and includes a search bar, a description of project access tokens, and a table of active tokens. A red arrow points from the 'Access tokens' link in the sidebar to the 'Add a project access token' form on the right. The form includes fields for 'Token name' (set to 'scc111-demo'), 'Token description', 'Expiration date' (set to '2025-03-23'), 'Select a role' (set to 'Owner'), and 'Select scopes'.

Hettiarachchi, Hansi / SCC111-Project / Access tokens

Search page

Project access tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API. You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Active project access tokens 0

Token name	Description	Scopes	Created	Last Used	Expires	Role	Action
This project has no active access tokens.							

Add new token

Add a project access token

Token name

scc111-demo

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Token description

Expiration date

2025-03-23

Select a role

Owner

Select scopes

Accessing a repository for the first time

```
MINGW64:/c/Users/hettiar1/OneDrive - Lancaster University/Teaching/SCC111/Co...
hettiar1@LU-GM0K1F09 MINGW64 /c/Users/hettiar1/OneDrive - Lancaster University/T
eaching/SCC111/CodeSamples/L32
$ git clone https://scc-source.lancs.ac.uk/hettiar1/scc111-project.git
Cloning into 'scc111-project'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

hettiar1@LU-GM0K1F09 MINGW64 /c/Users/hettiar1/OneDrive - Lancaster University/T
eaching/SCC111/CodeSamples/L32
$ cd scc111-project

hettiar1@LU-GM0K1F09 MINGW64 /c/Users/hettiar1/OneDrive - Lancaster University/T
eaching/SCC111/CodeSamples/L32/scc111-project (main)
$ ls
README.md

hettiar1@LU-GM0K1F09 MINGW64 /c/Users/hettiar1/OneDrive - Lancaster University/T
eaching/SCC111/CodeSamples/L32/scc111-project (main)
$ |
```

What Git does...

Once you have a local repo, your computer is tracking ALL changes you make in that folder.

- New files
- Deleted files
- Changed files

Your local copy will not change by itself – even if someone else updates the remote repo

- You can trust the local files no to change
- You can trust that **ANY** changes you make locally will not irrevocably affect others
- You can trust that any code that has been committed can never be truly destroyed as long as the repo exists.

Adding new files...

If you create a new file in a repo, this will be noticed

- But it will not automatically become part of the repo
- Do this explicitly
- Once added, the files content will be tracked for changes.
- **Therefore, only add files you want to be tracked.**
- Use "git status" to show untracked files, and changes in tracked files.
- Use "git add" to add a newly created file to a repo

```
git status
```

```
git add <filename>
```

Committing changes...

When you reach a good "checkpoint" commit your changes

- E.g. fixed a bug, added a new feature...
- Keep commits clean (one meaningful code change per commit)
- Choose a name for a commit that describes the change
- Commit often and include as much detail as necessary.
- **Review changes before you commit. Did you intend to make all those changes??**

```
git add <filename>
```

```
git commit -m "A short description of what you changed."
```

Sharing changes...

Once you have one or more local commits, you can share back to the server.

- This makes your changes visible to others
- It also means your work is properly backed up...
- Until you push, you are still vulnerable to local loss hardware fail, theft, damage, accidental deletion...

Use git push to propagate your commits back to the server.

- We specify where to push to (normally "origin" – wherever you cloned from) and the branch to push (the "main" branch is fine for now!)

```
git push origin main
```

Receiving changes...

You can receive updates from others via explicitly merging them into your local repo

- Changes only arrive when you want them to.
- Use **git pull** to do this
- Any commit you don't have are then downloaded and merged into your local repo.

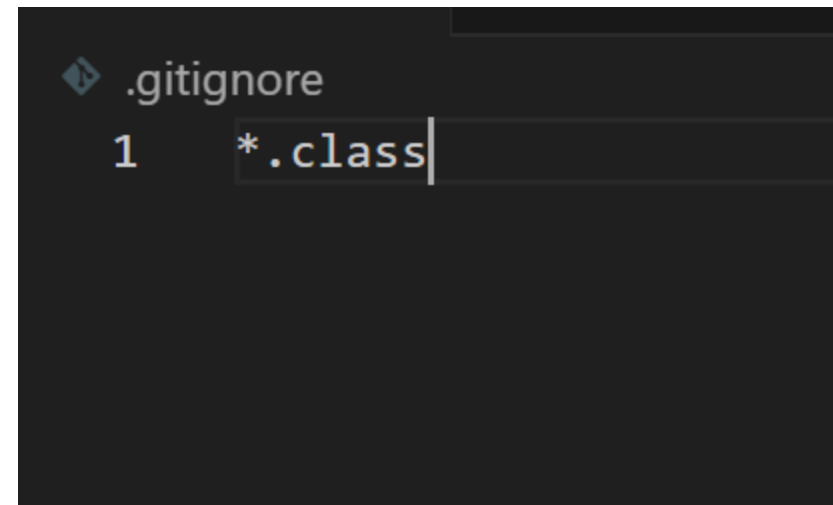
Unless both you and someone else have editing the same file since you last pulled, no user intervention is needed...

If the version on the server is newer than your local version, you will be required to perform a pull operation before you can push.

```
git pull
```

Ignoring changes...

- We might need to keep some files only in our local repository (e.g. .class files).
- You can configure Git to ignore such files.
- Create a **.gitignore** file in your local repository's root directory to tell Git which files and directories to ignore when you make a commit.



```
.gitignore
1 *.class
```

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
```


Summary

Today we learned:

- How version control can promote collaboration
- Practical guidance on using Git version control
- Examples of a typical Git Workflow

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>