

Using Visual Studio Code with GitHub in a team

Work as a pair, with one student as StudentA and another as StudentX

Steps for StudentA and StudentX are indicated as **A** and **X** respectively

While one student is doing the steps, the other student should see what is being done

If there are an odd number of students in the class, one group can have 3 students

The third student will do steps similar to Student X after all the steps are completed

Overview of steps that will be done

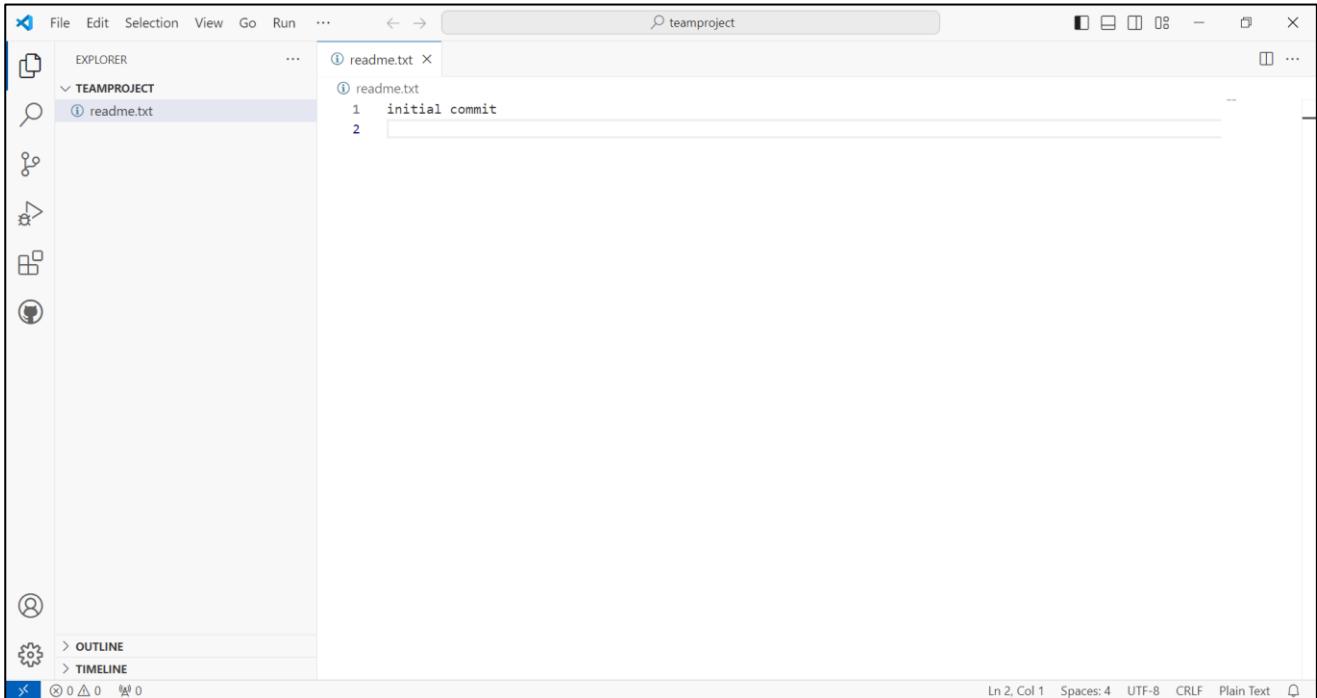
Sequence	Student A	Student X
1	Create a folder for the project	
2	Create a local repository	
3	Stage changes	
4	Commit changes	
5	Publish branch	
6	View repository on GitHub	
7	Invite collaborators	
8		Accept invite
9		Get the project web URL
10		Clone the repository
11		Create a branch
12		Publish the branch
13		Make changes, stage changes, commit changes, sync changes
14		Checkout main
15		Merge branch
16	Create a branch	
17	Publish the branch	
18	Make changes, stage changes, commit changes, sync changes	
19	Checkout main	
20	Merge branch	
21	Resolve conflicts	
22	Sync changes	

Create a folder for the project

- A 1. Create a folder `teamproject`

Open the folder in Visual Studio Code

Add a text file called `readme.txt` with the text `initial commit`

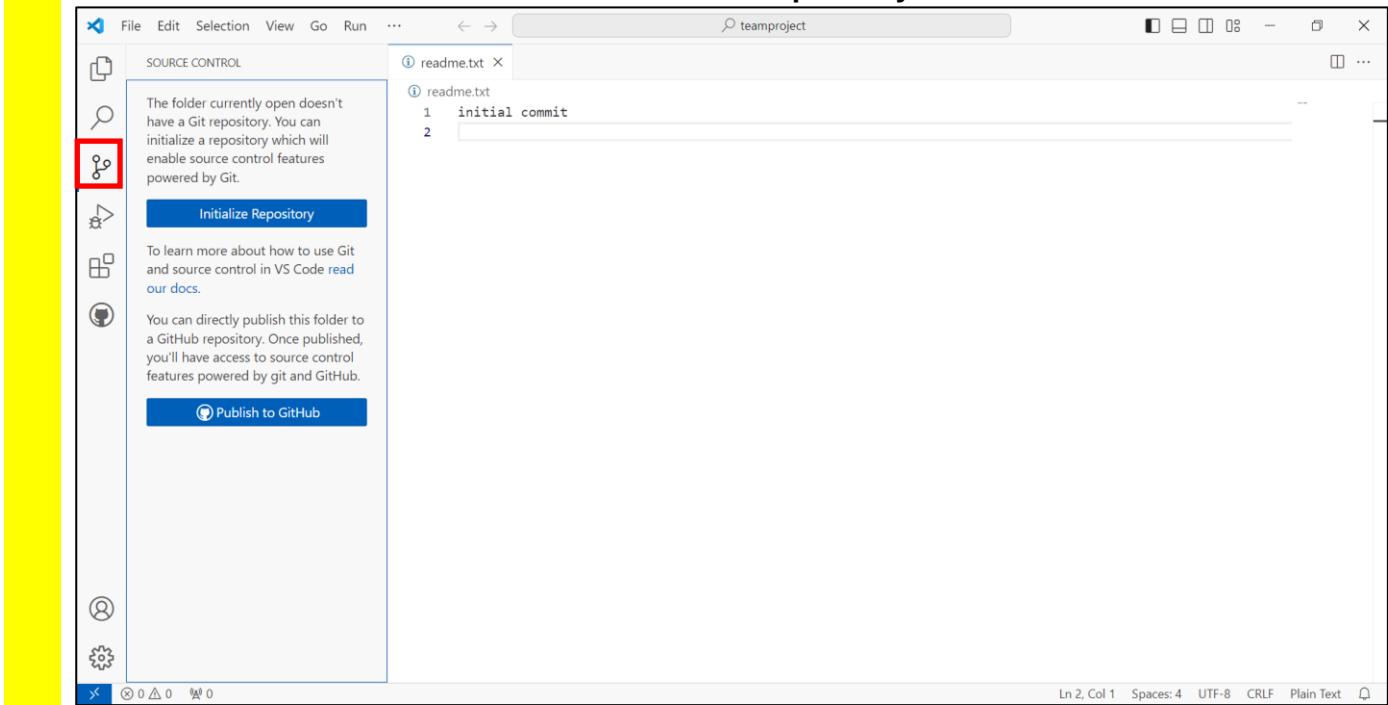


At this point, the project only exists on StudentA's computer

As per normal, you can add folders and source files to your project. For this example, we will work on a single text file, but the same principles apply to other files in your projects.

Create a local repository

A 2. Click the Source Control button and Initialize Repository

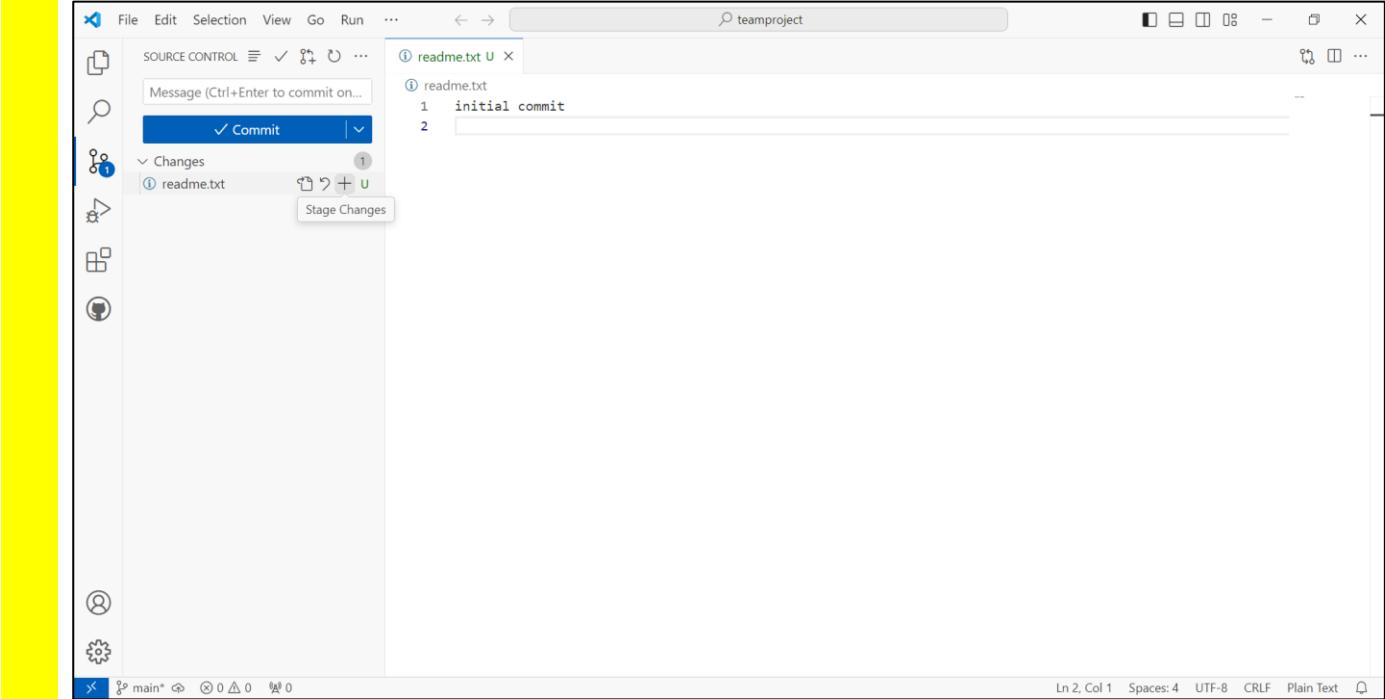


After creating a project, you will need to initialise a **local repository** at the root of your project folder. This will enable version control features for the project.

Stage changes

The bottom left corner of Visual Studio Code shows that you are working on the main branch of the local repository.

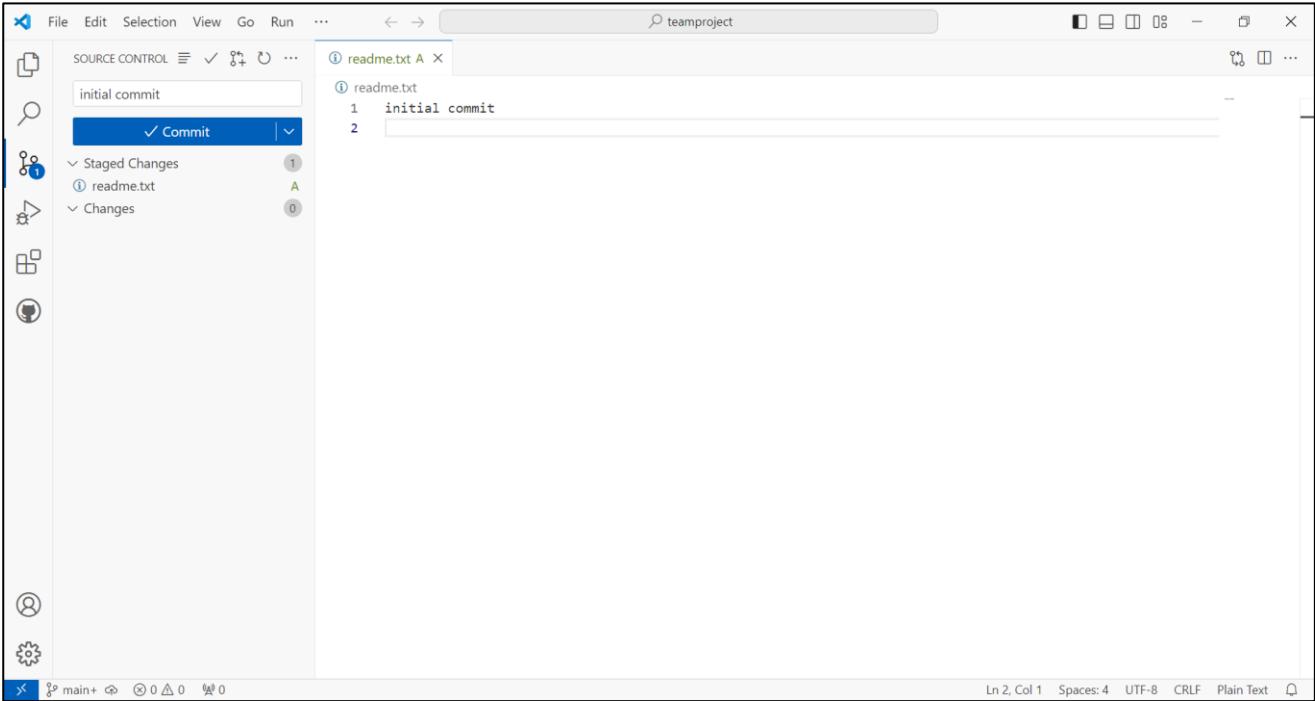
- A 3. Click on the + button next to `readme.txt` to stage changes for that file



After a file has been created, modified or deleted, you will need to stage it. This prepares the file for committing.

Commit changes

- A 4. Set the commit message as **initial commit** and press **Commit**

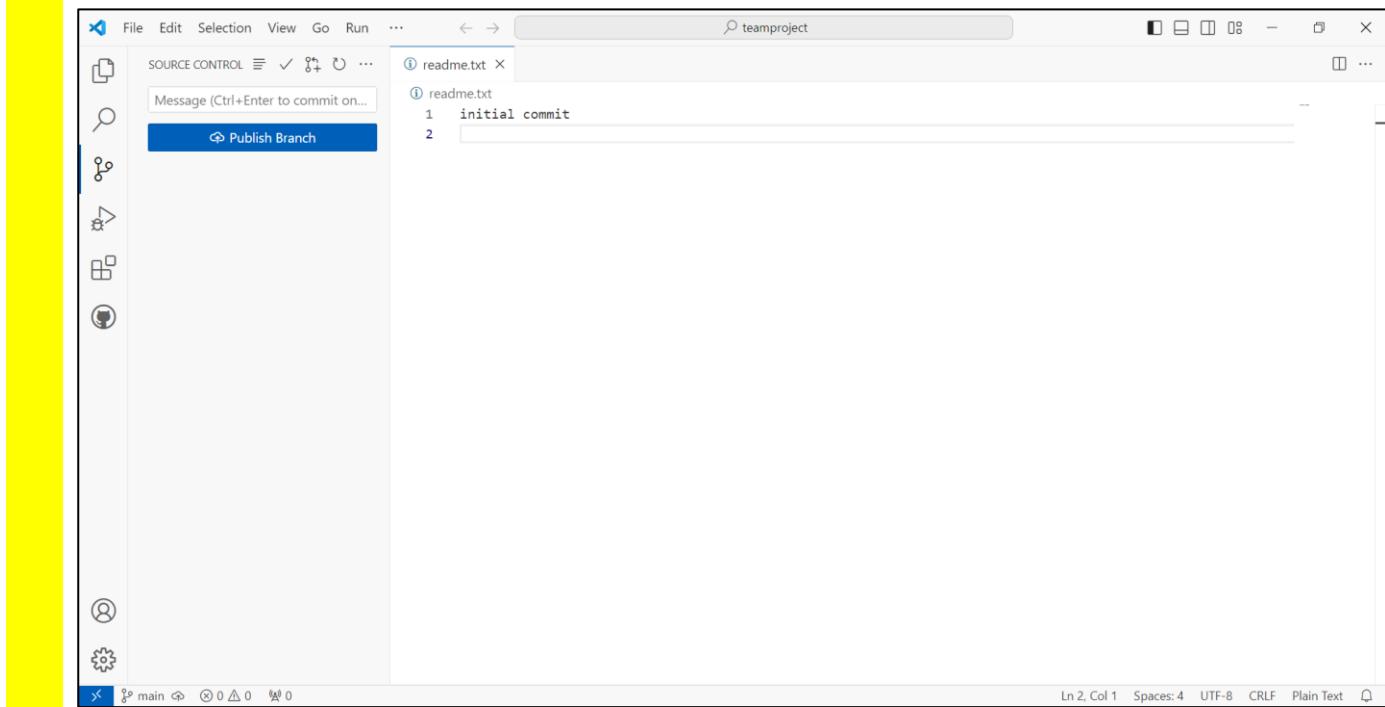


The commit message should be used to provide a meaningful explanation of what was committed.

For example, if you were working on a feature, the commit message should mention the feature that was developed.

Publish branch

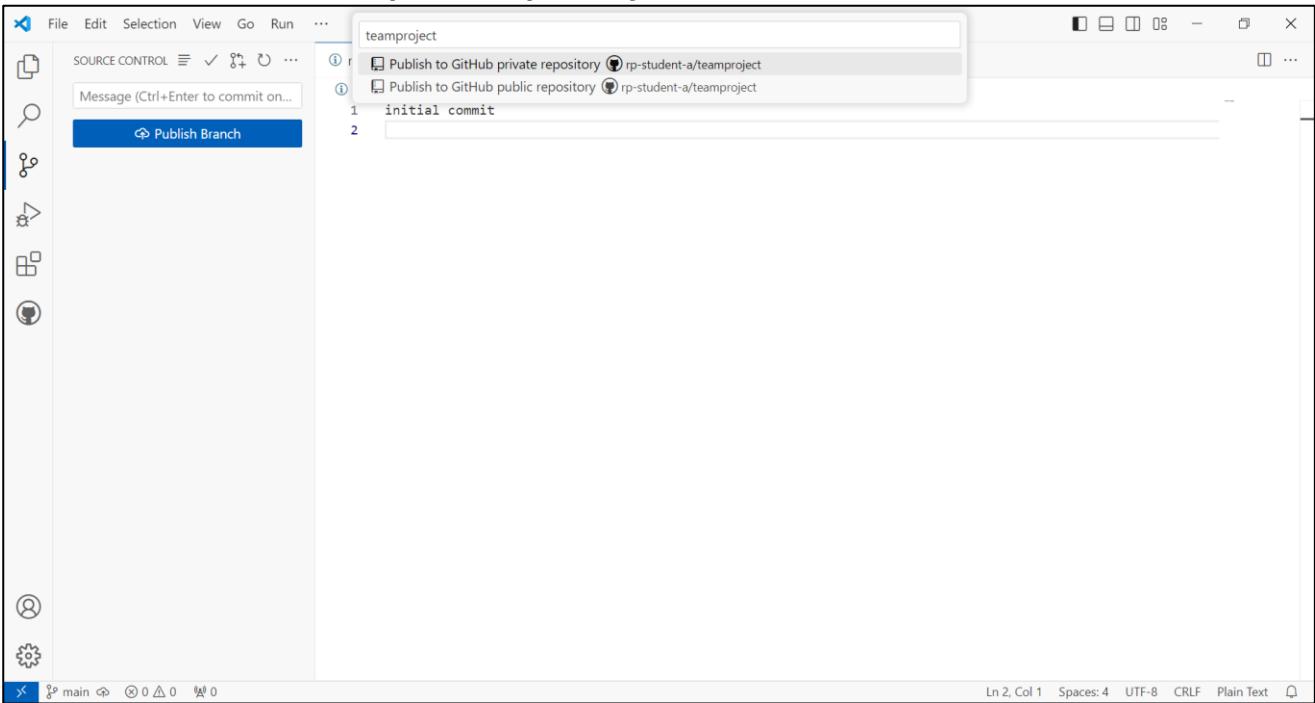
A 5. Click Publish Branch



At this point, the repository only exists on Student A's computer.

Publish branch will publish the project to the main branch of the **remote repository** on GitHub.

A 6. Click Publish to GitHub private repository

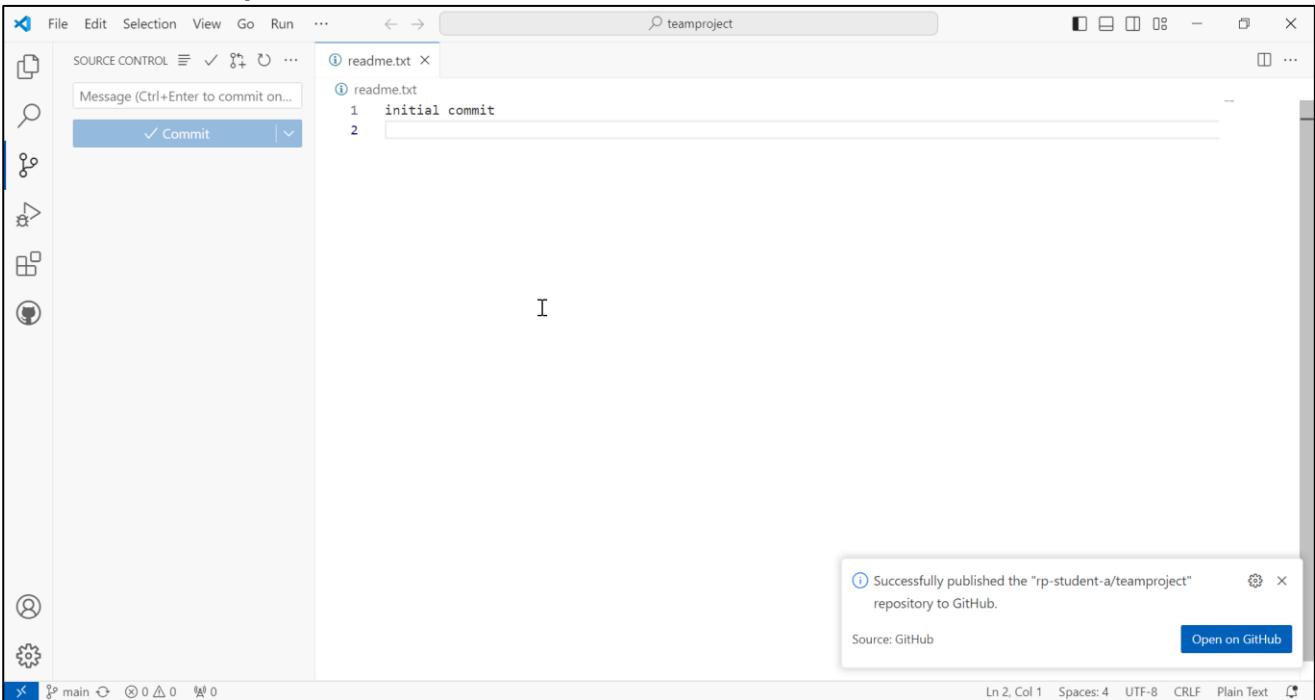


A repository can be **private** or **public**.

Public repositories can be viewed by anyone.

Private repositories can be viewed by collaborators.

A 7. A dialog should appear to indicate that the repository was successfully published to GitHub. Click on the **Open on GitHub** button to launch GitHub in a browser.

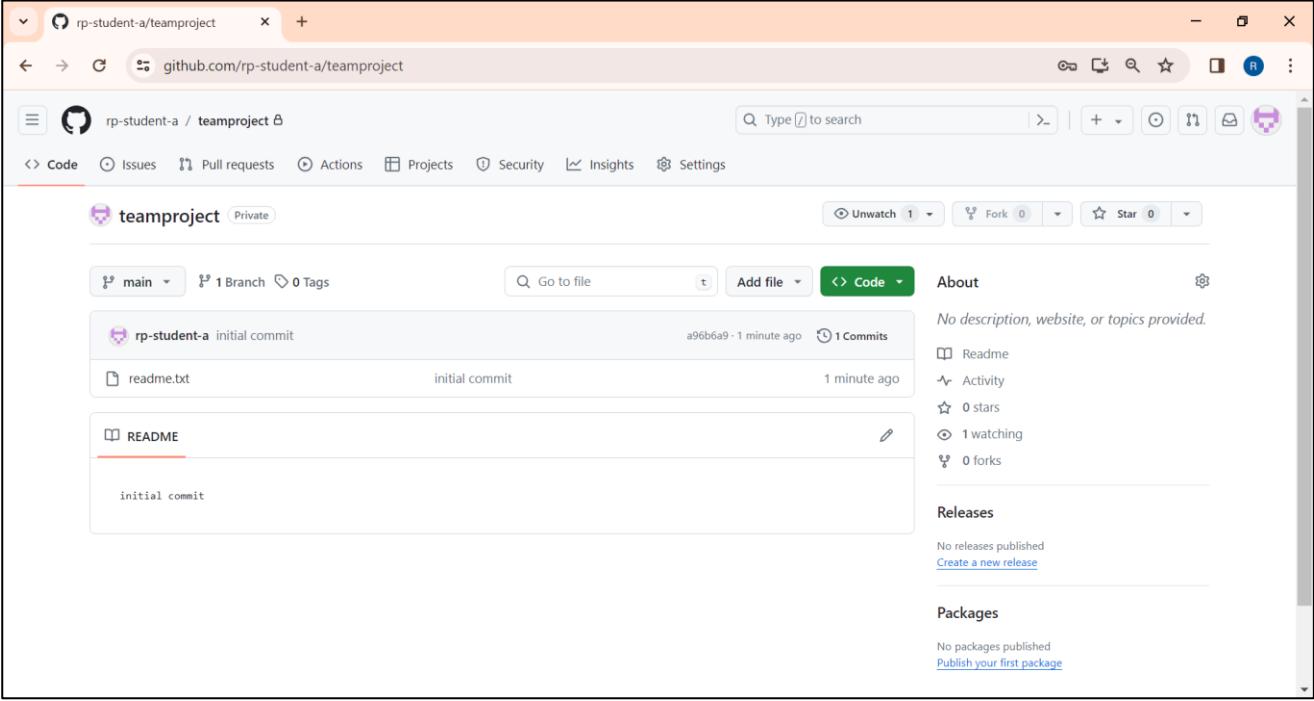


View repository on GitHub

- A 8. The repository has been created on GitHub

Click on **Settings**

For the next step, we will be adding collaborators, which can be done from the Settings page



The screenshot shows a GitHub repository named 'teamproject'. The repository is private and contains one branch ('main') and one commit ('rp-student-a initial commit'). The commit was made by 'rp-student-a' and pushed 1 minute ago. The commit message is 'initial commit'. On the right side of the page, there is an 'About' section which states 'No description, website, or topics provided.' It also lists 'Readme', 'Activity', '0 stars', '1 watching', and '0 forks'. Below that is a 'Releases' section with a link to 'Create a new release'. At the bottom is a 'Packages' section with a link to 'Publish your first package'.

The remote repository can act as a **backup** for your project.

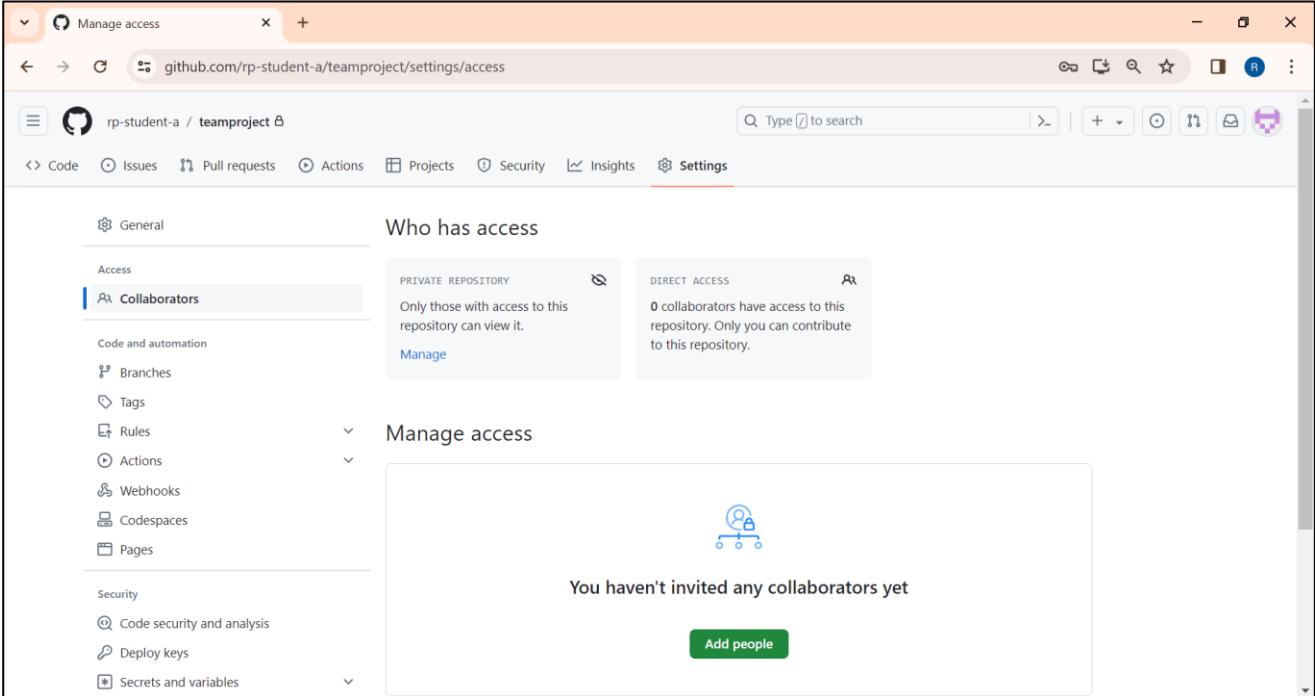
You can **retrieve** the project files that you have published by cloning the repository.

You should publish your projects (e.g. project assignments, final year project) to a **private repository**.

Do not publish to a public repository or else anyone will have access to your projects. It is **your responsibility** to ensure that your projects are not accessed by unauthorised people.

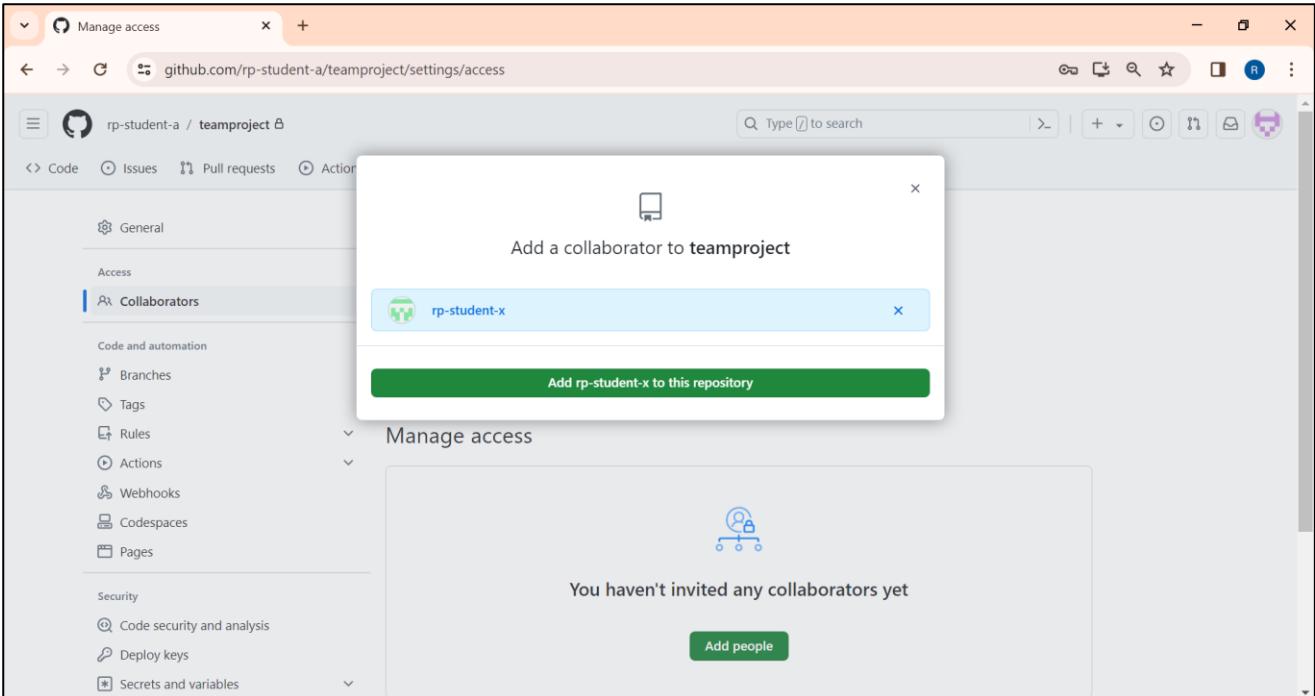
Invite collaborators

- A 9. On the **Settings** page, click **Collaborators** from the left panel
 Click on **Add people**



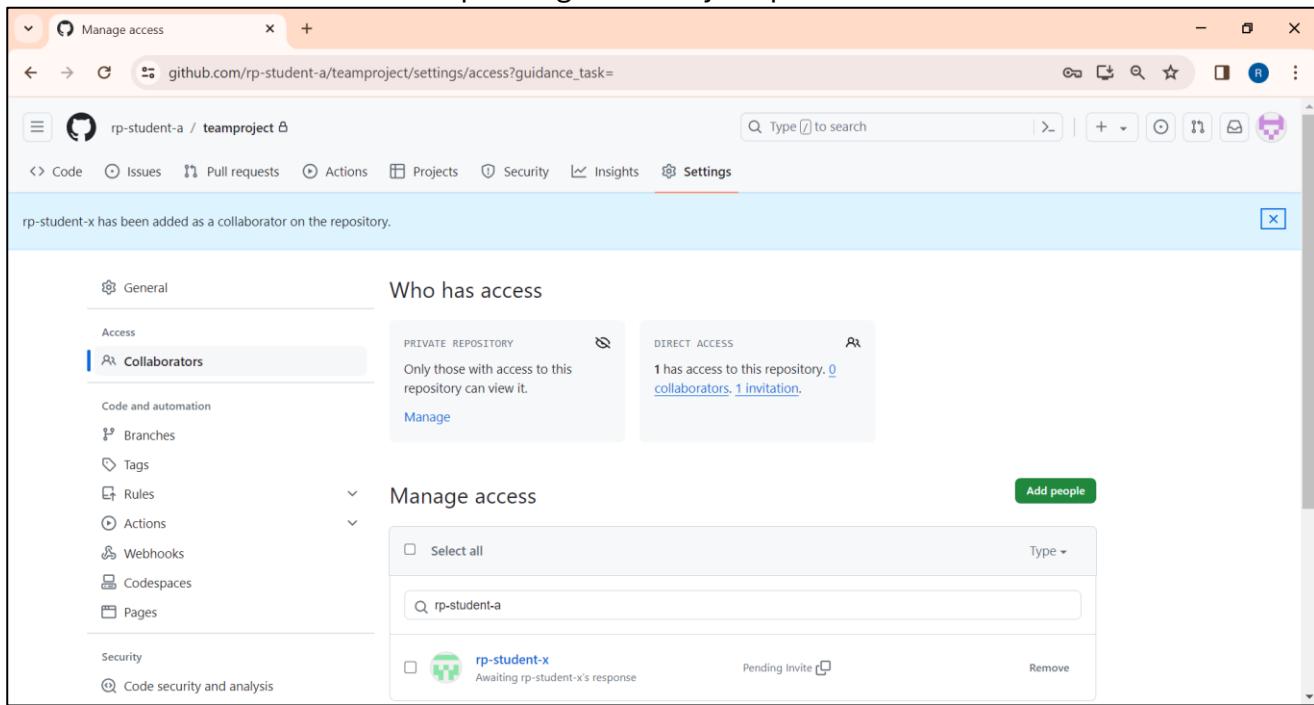
The screenshot shows the 'Manage access' settings page for a private repository. The 'Collaborators' tab is selected in the sidebar. Two sections are displayed: 'PRIVATE REPOSITORY' and 'DIRECT ACCESS'. The 'PRIVATE REPOSITORY' section states 'Only those with access to this repository can view it.' and has a 'Manage' link. The 'DIRECT ACCESS' section states '0 collaborators have access to this repository. Only you can contribute to this repository.' A large green 'Add people' button is located at the bottom of the 'Manage access' section.

- A 10. Enter the GitHub id for your partner and click on the **Add to this repository** button



The screenshot shows the 'Manage access' settings page with a modal dialog open. The dialog is titled 'Add a collaborator to teamproject' and contains a search input with 'rp-student-x' entered. Below the input is a green button labeled 'Add rp-student-x to this repository'. The background page shows the 'Collaborators' section of the settings page, which includes a sidebar with various repository management options like 'Code and automation', 'Branches', 'Tags', etc., and a main area with a message 'You haven't invited any collaborators yet' and a 'Add people' button.

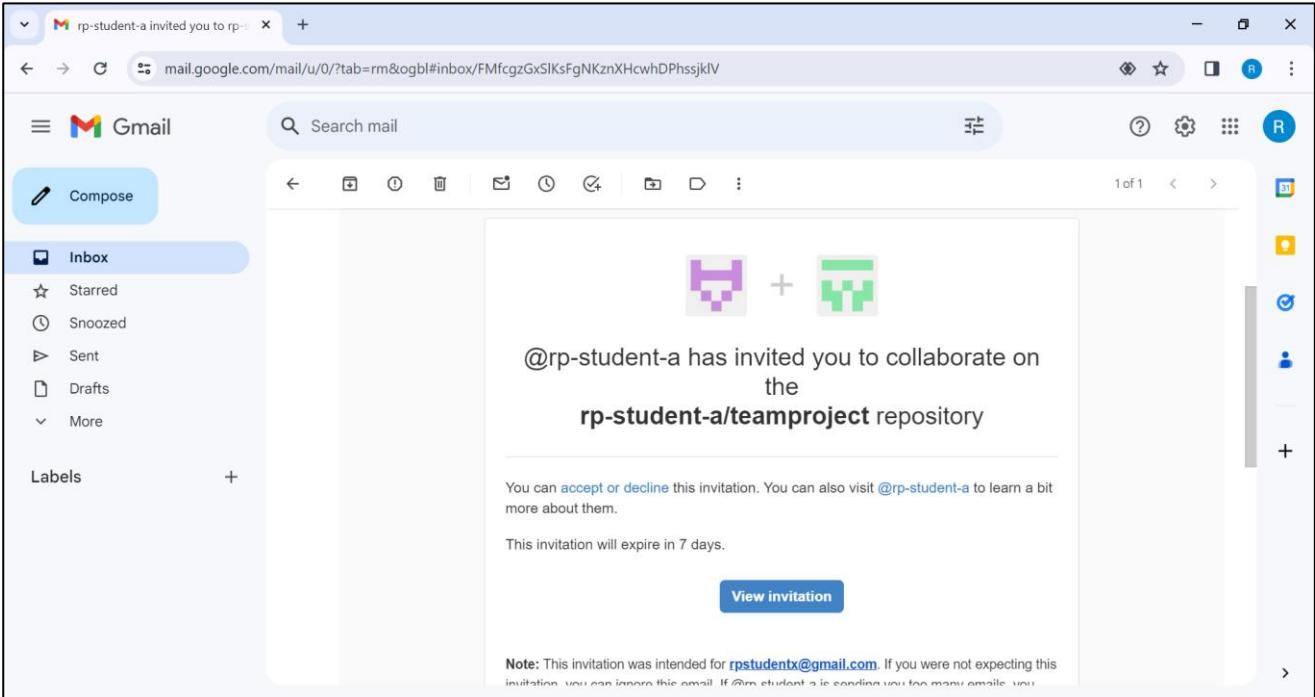
A 11. You should see that there is a pending invite for your partner



The screenshot shows the GitHub repository settings page for 'rp-student-a / teamproject'. The 'Settings' tab is selected. In the 'Who has access' section, it shows 'rp-student-x' has been added as a collaborator, with a status of 'Awaiting rp-student-x's response' and a 'Pending Invite' button. The left sidebar shows the 'Collaborators' section is active.

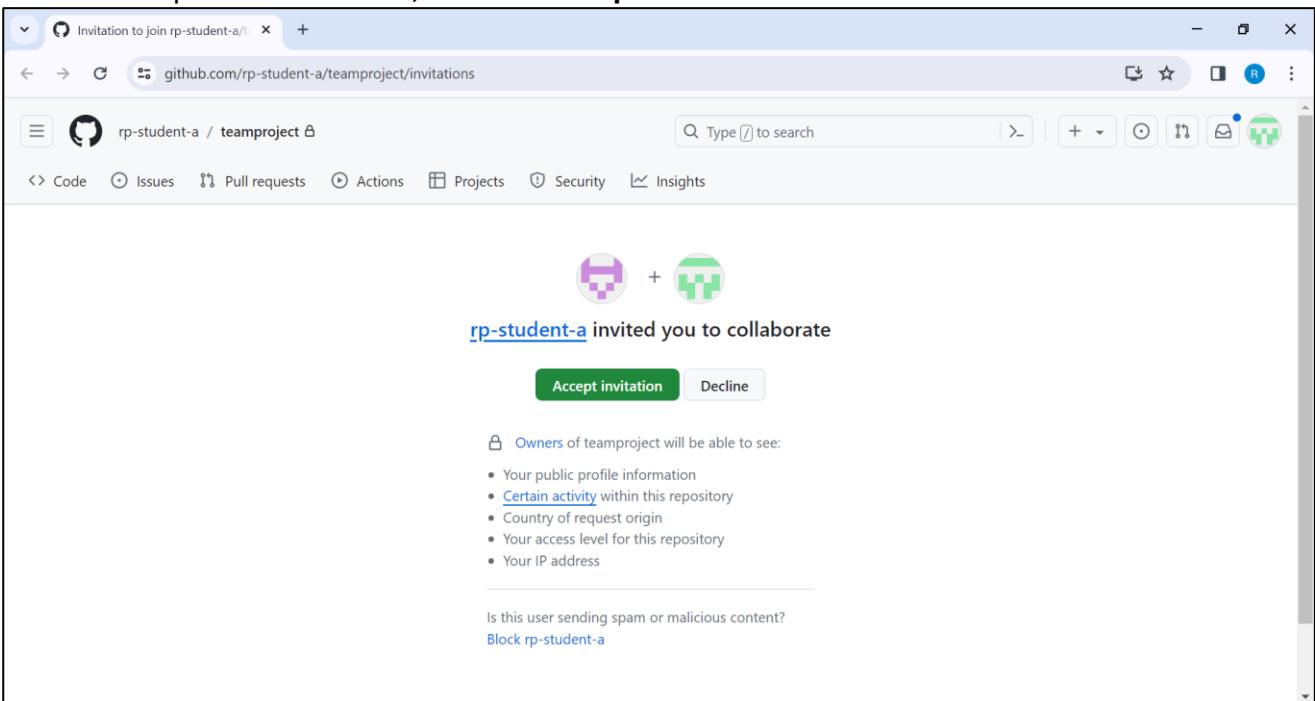
Accept invite

- X 12. Open the email for StudentX to see the invitation, click on **View invitation**



The screenshot shows a Gmail inbox with one unread email. The email is from 'rp-student-a' with the subject 'rp-student-a invited you to rp-student-a/teamproject'. The message body contains a note: 'You can accept or decline this invitation. You can also visit @rp-student-a to learn a bit more about them.' Below this is a note: 'This invitation will expire in 7 days.' At the bottom is a blue 'View invitation' button. The Gmail interface includes a search bar, a toolbar with various icons, and a sidebar with options like 'Compose', 'Inbox', 'Starred', 'Snoozed', etc.

- X 13. GitHub opens in a browser, click on **Accept invitation**



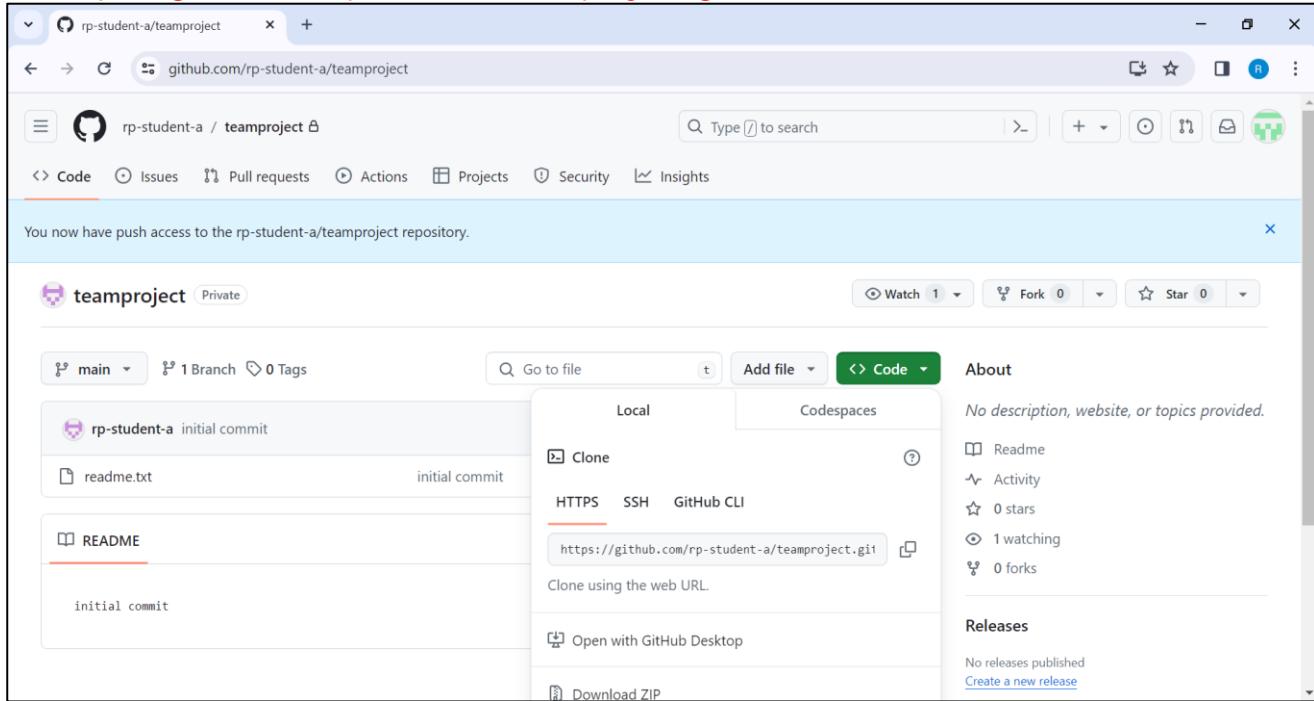
The screenshot shows a GitHub invitation page. At the top, it says 'rp-student-a invited you to collaborate' with a link to 'rp-student-a / teamproject'. Below this are two buttons: 'Accept invitation' (green) and 'Decline' (grey). A note below the buttons states: 'Owners of teamproject will be able to see:' followed by a bulleted list: 'Your public profile information', 'Certain activity within this repository', 'Country of request origin', 'Your access level for this repository', and 'Your IP address'. At the bottom, there is a question 'Is this user sending spam or malicious content?' with a 'Block rp-student-a' link.

Get the project web URL

- X 14. Click on the **Code** button, then click on the copy button to copy the project web URL

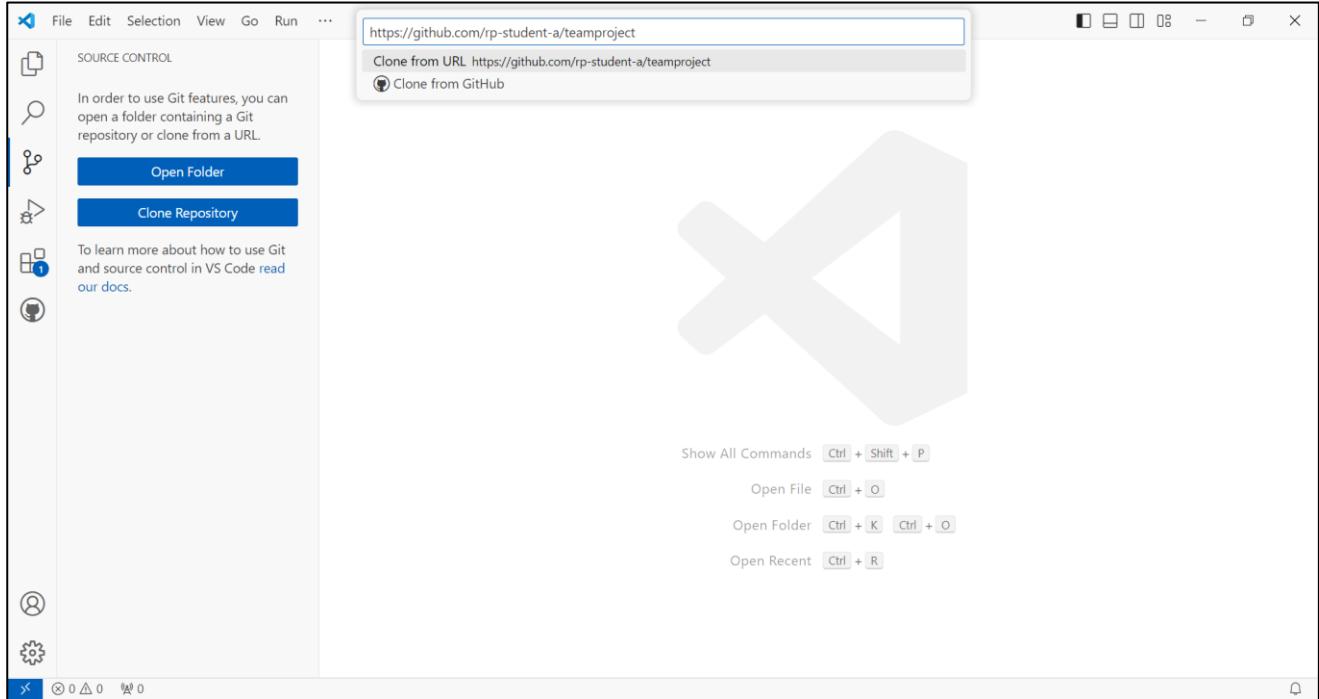
The project web URL in the example below is

<https://github.com/rp-student-a/teamproject.git>

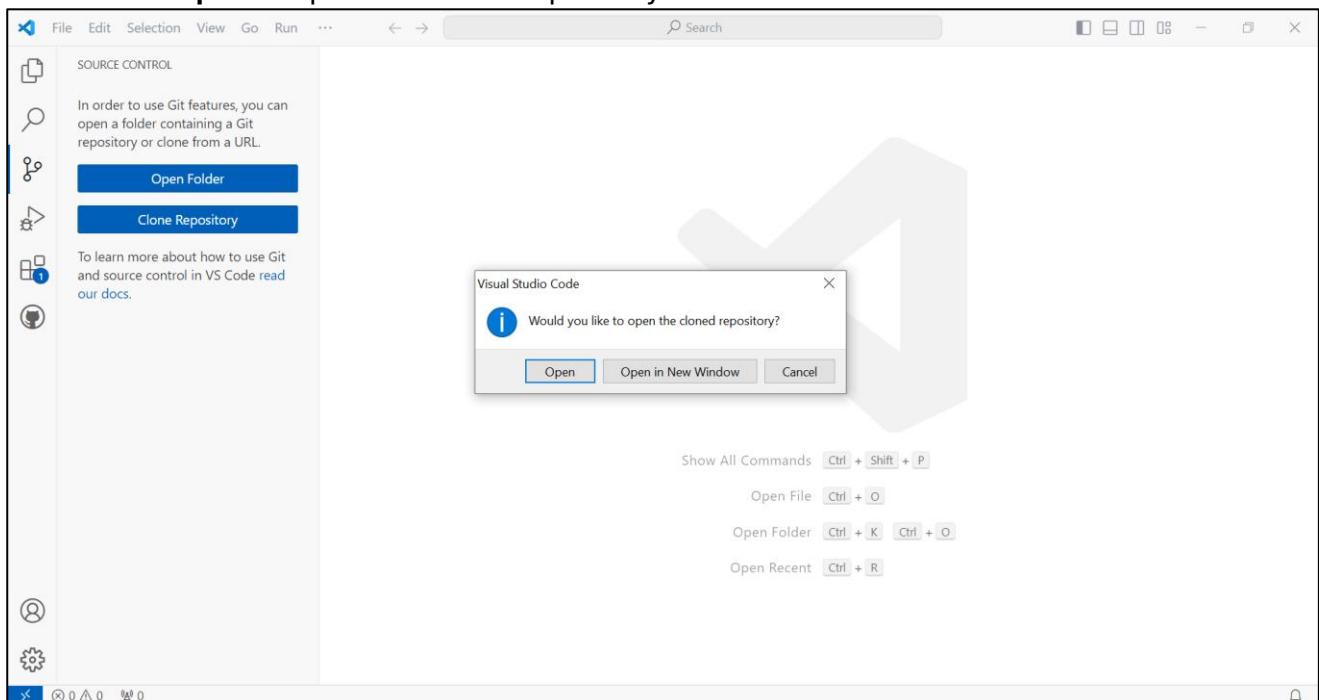


Clone repository

- X 15. Click on the source control button, and **Clone Repository**
 Enter the project web URL and click **Clone from URL**
 Select a folder to clone the project to



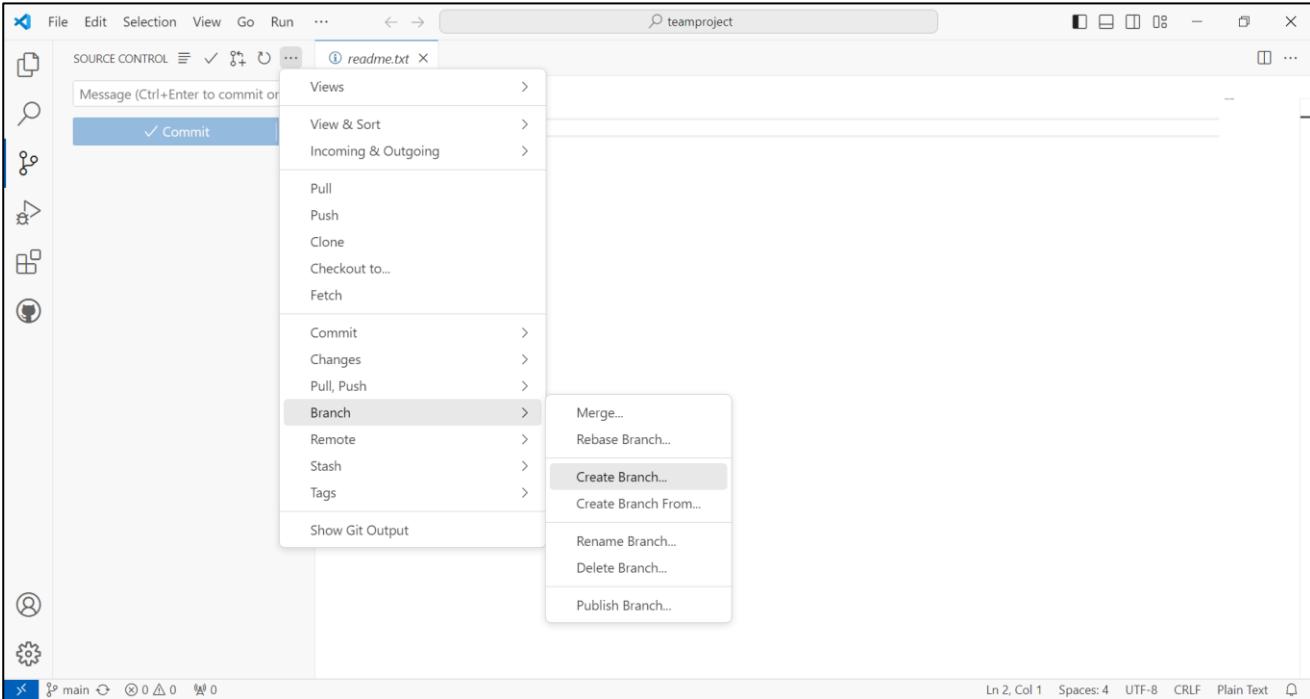
- X 16. Click on **Open** to open the cloned repository in Visual Studio Code



At this point, the project also exists on StudentX's computer

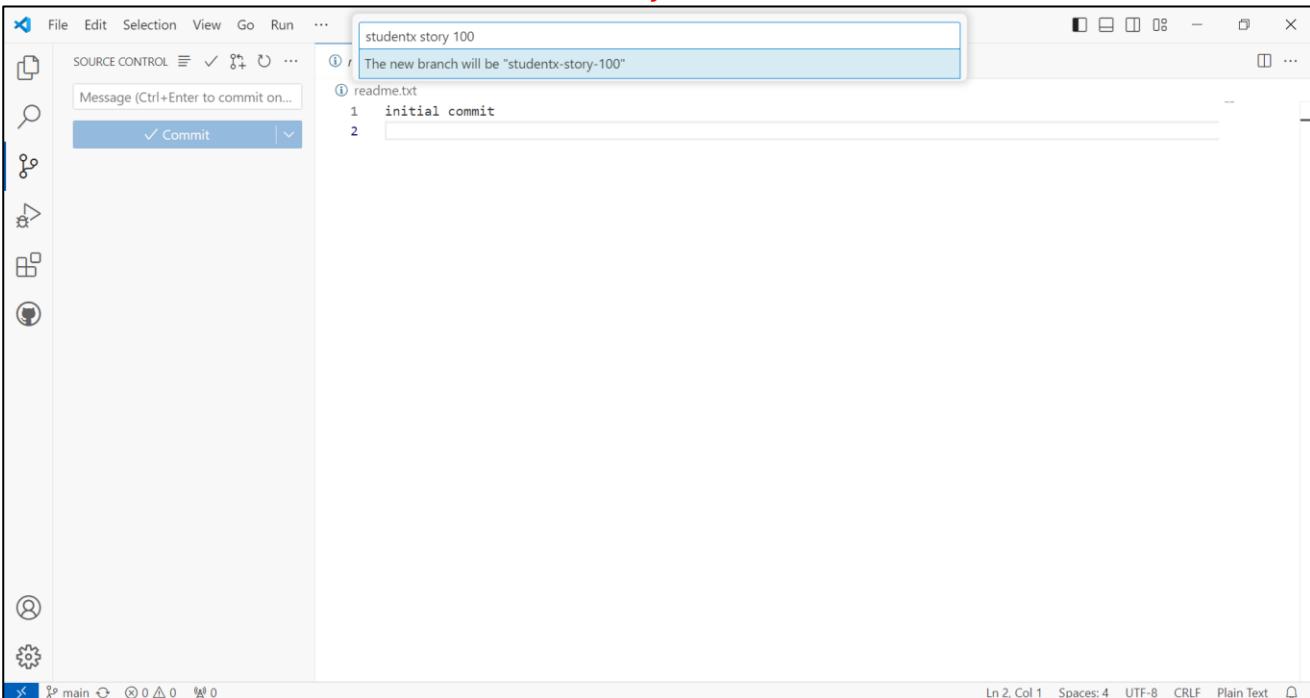
Create branch

- X 17. Click on the ellipsis, select **Branch, Create Branch...**



Before you make any changes to the files, create a branch with a meaningful name, for example, the name of the feature or story you are working on.

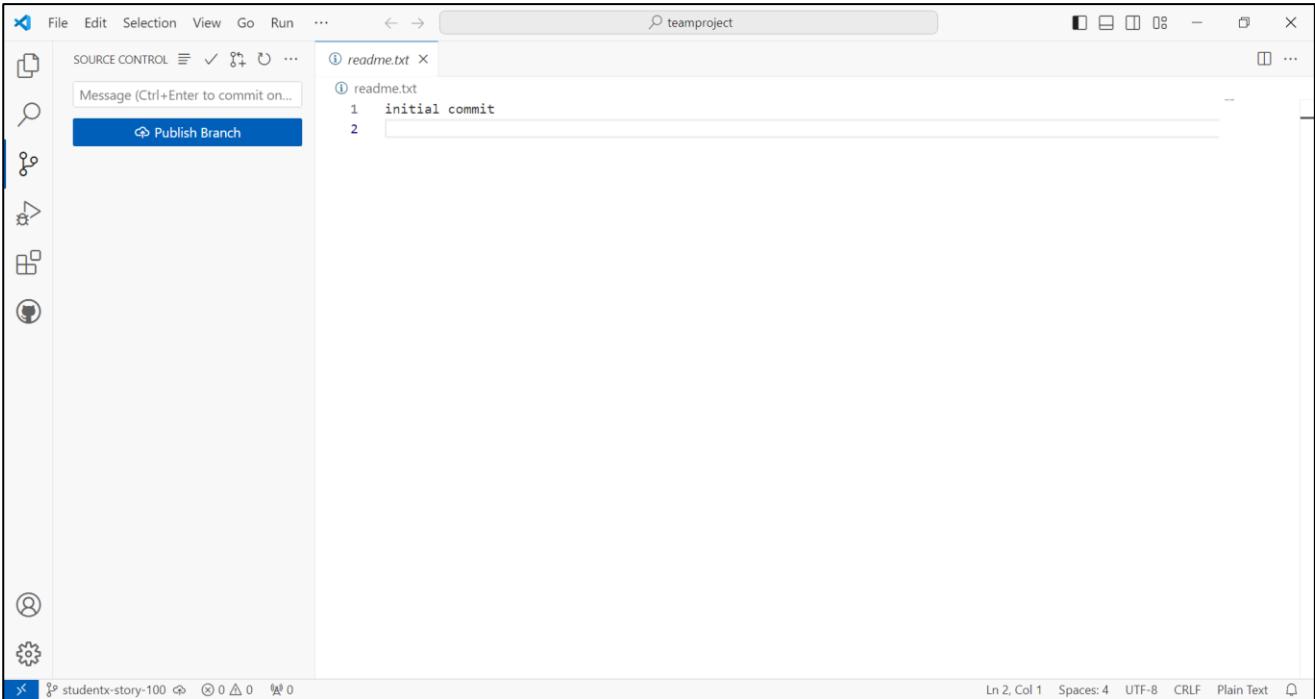
- X 18. Enter the branch name as **studentx_story_100**
The new branch will be called **studentx-story-100**



Publish branch

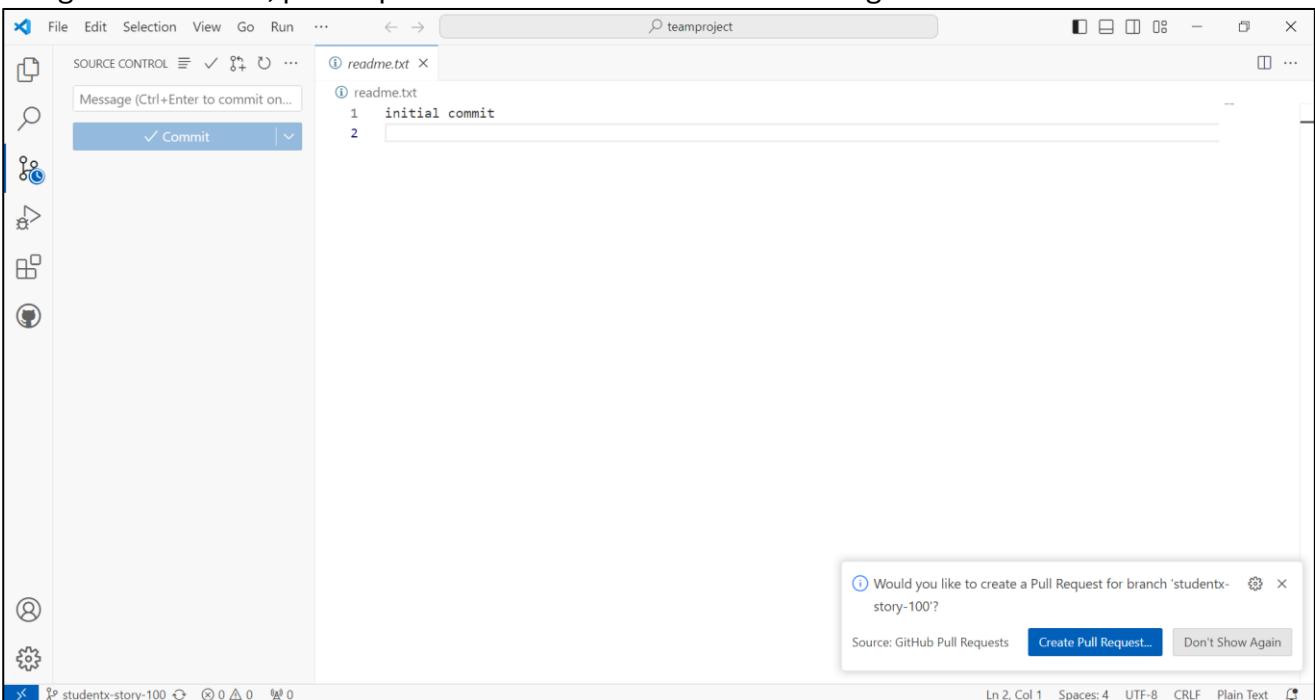
The bottom left corner in Visual Studio Code shows you are on branch **studentx-story-100**
 This branch only exists in the local repository on StudentX's computer
 We will need to publish it to the remote repository on GitHub

X 19. Click on Publish Branch



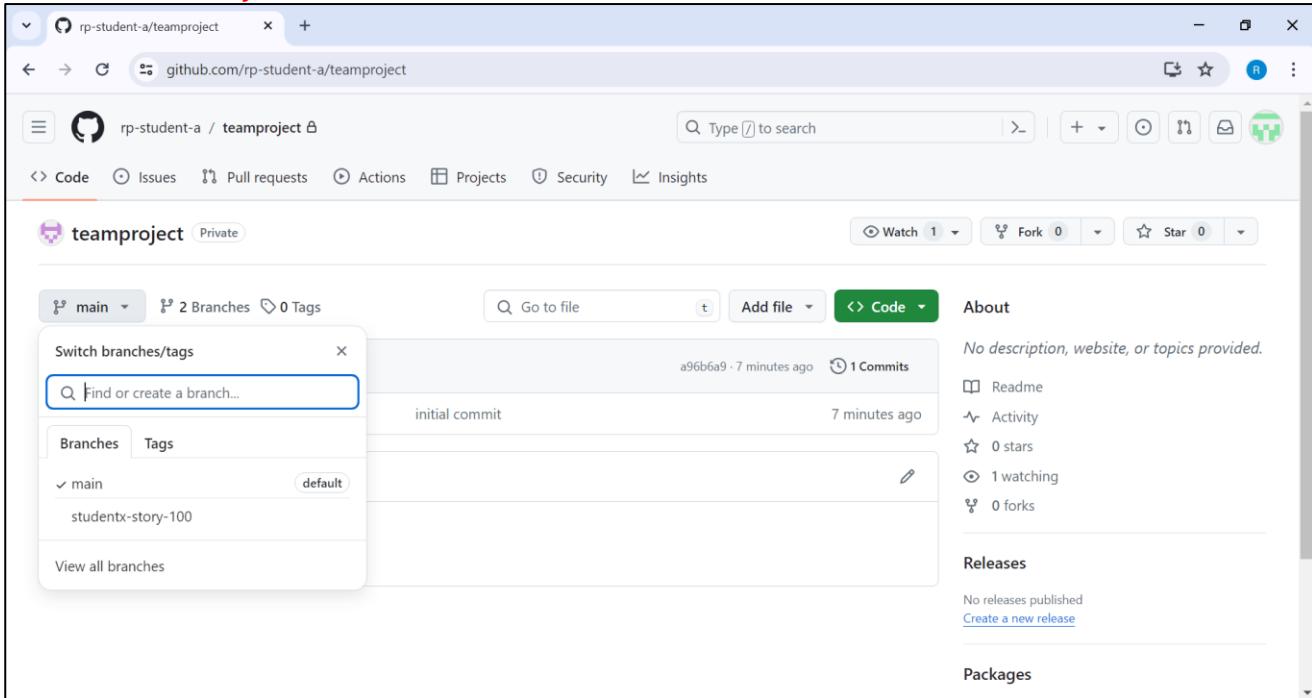
X 20. You should see a dialog to Create Pull Request...

Ignore it for now, pull requests will not be covered at this stage



X 21. View the repository in GitHub in a browser

Click on the dropdown next to **main**, you should see that there are 2 branches - **main** and **studentx-story-100**



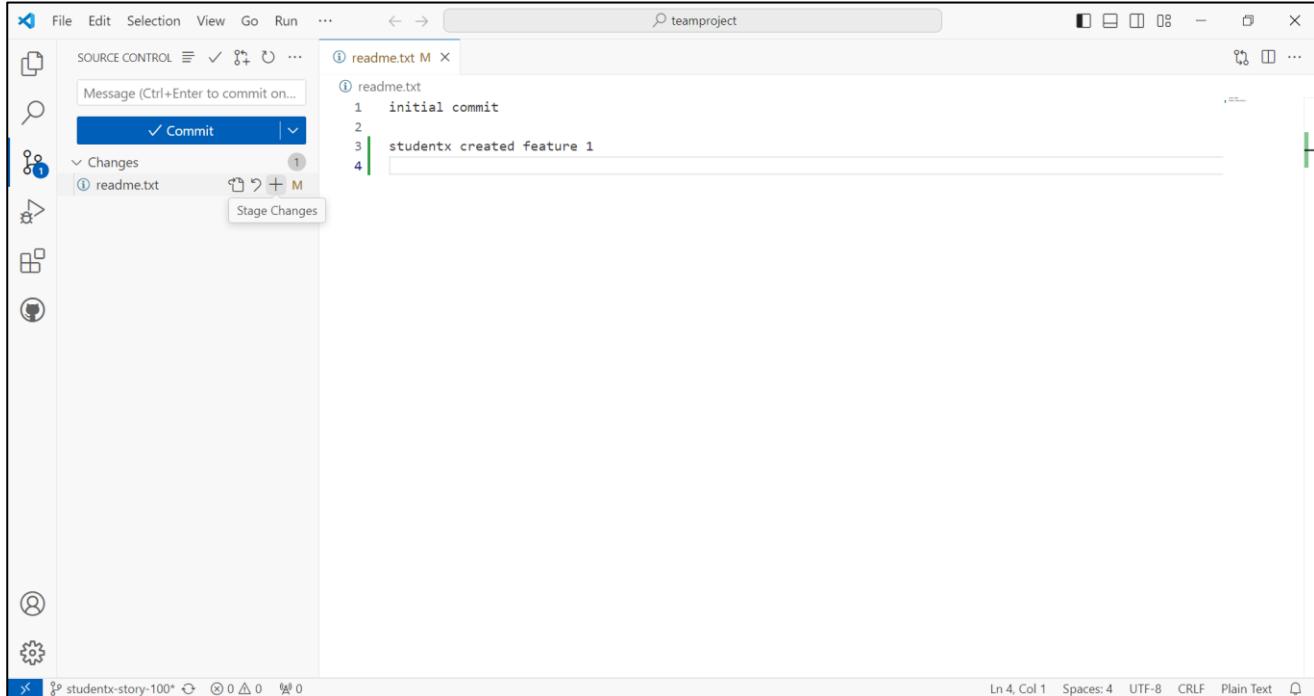
The screenshot shows a GitHub repository page for 'rp-student-a/teamproject'. The repository name is 'teamproject' and it is private. There are two branches: 'main' and 'studentx-story-100'. The 'main' branch is selected. The repository has 1 commit, an initial commit made 7 minutes ago. The 'About' section indicates no description, website, or topics provided. It shows 1 watching and 0 forks. The 'Packages' section is empty.

The project from the **main** branch should only contain the stable and fully tested code

The project from the **studentx-story-100** branch contains the code that StudentX is working on for the feature defined by story 100. Before the feature is completed, this branch may contain code that is only partially complete.

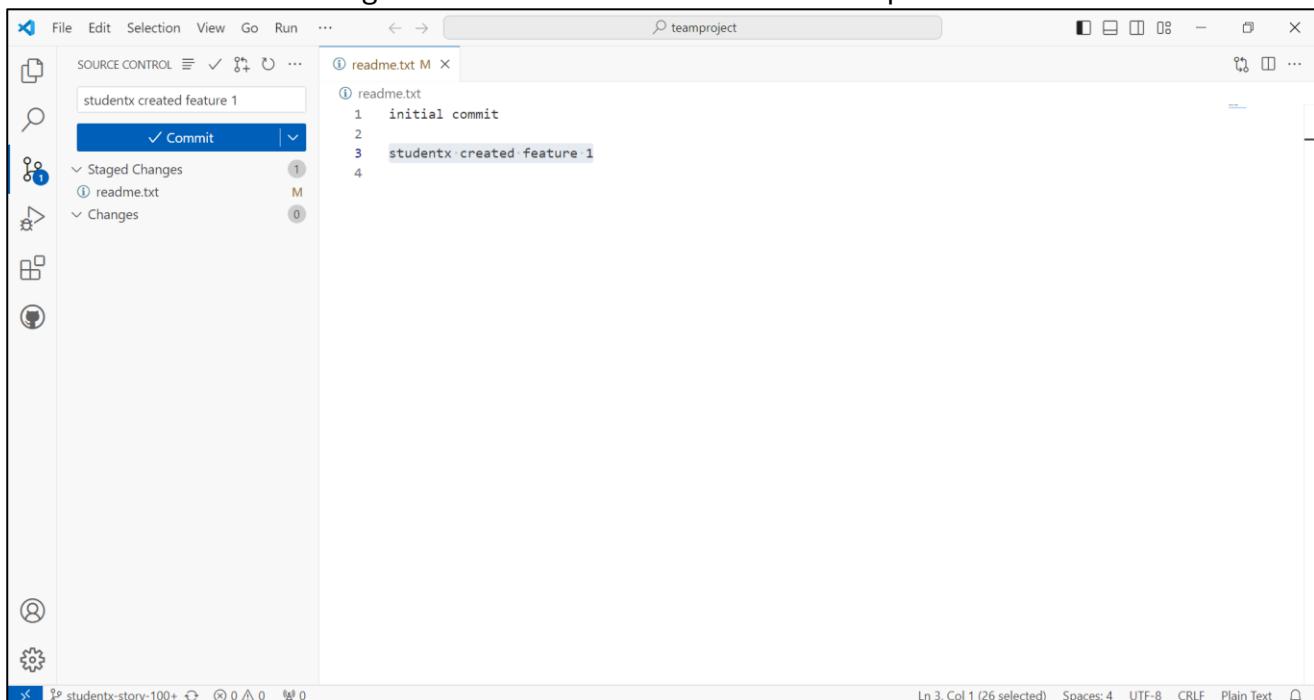
Make changes, stage changes, commit changes, sync changes

- X 22. Modify `readme.txt` by adding the text `studentx created feature 1` on line 3 and add a new line at line 4
 Click on the + button next to `readme.txt` to stage changes for that file



```
① readme.txt M x
① readme.txt
1 initial commit
2
3 studentx created feature 1
4
```

- X 23. Set the commit message as `studentx created feature 1` and press **Commit**

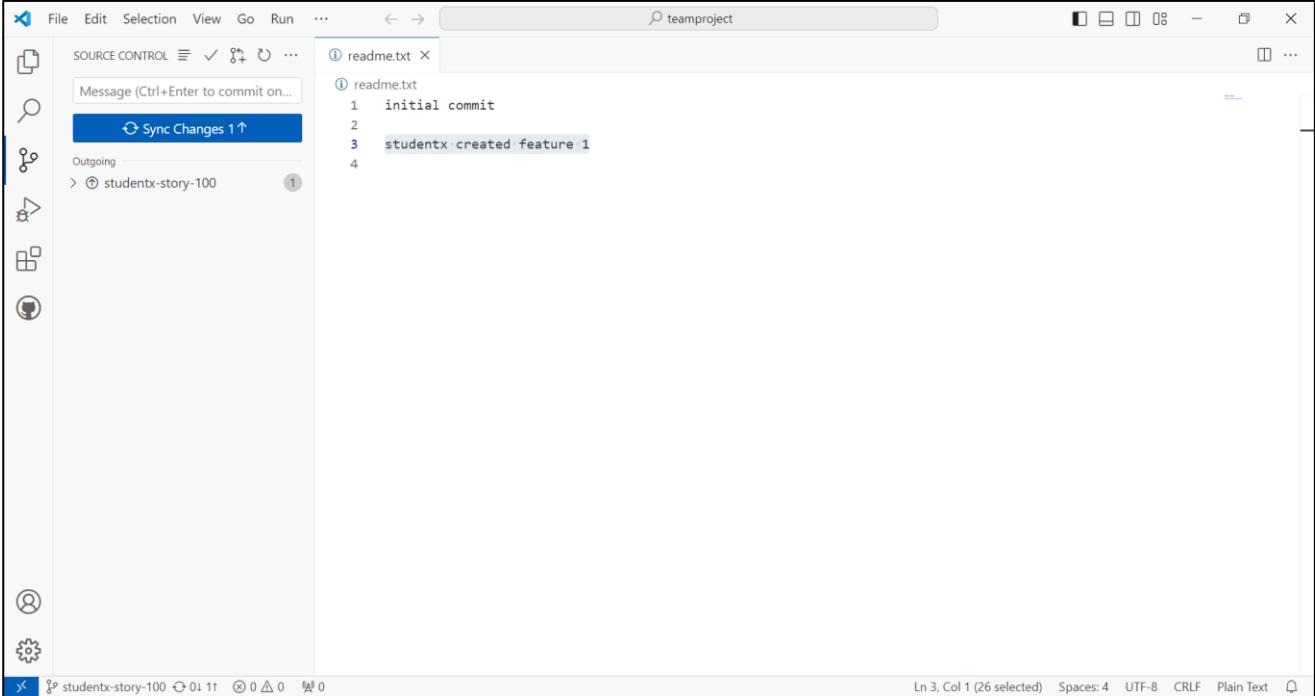


```
① readme.txt M x
① readme.txt
1 initial commit
2
3 studentx created feature 1
4
```

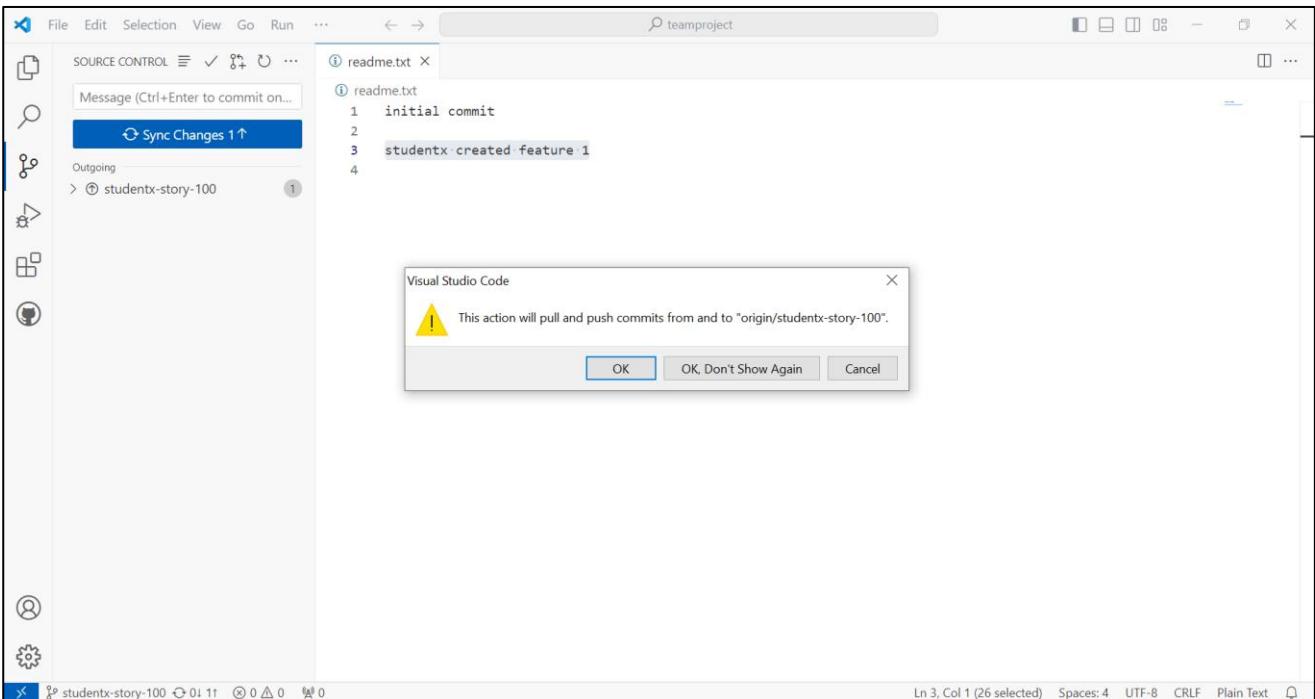
The changes are committed to the local repository

We need to sync the changes in the local repository to the remote repository

X 24. Click on Sync Changes



X 25. The dialog informs you that sync changes will pull and push commits from and to **origin/studentx-story-100** Click OK



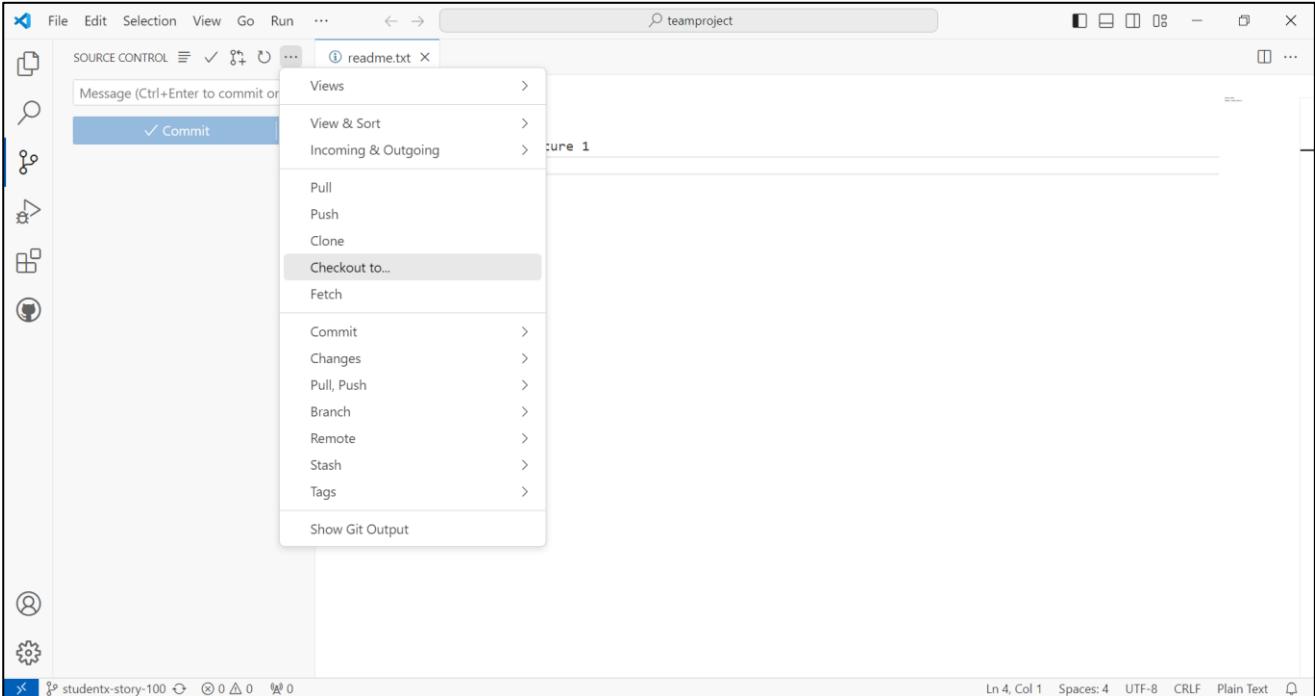
pull will pull the changes from the remote repository to the local repository
push will push the changes from the local repository to the remote repository

Checkout main

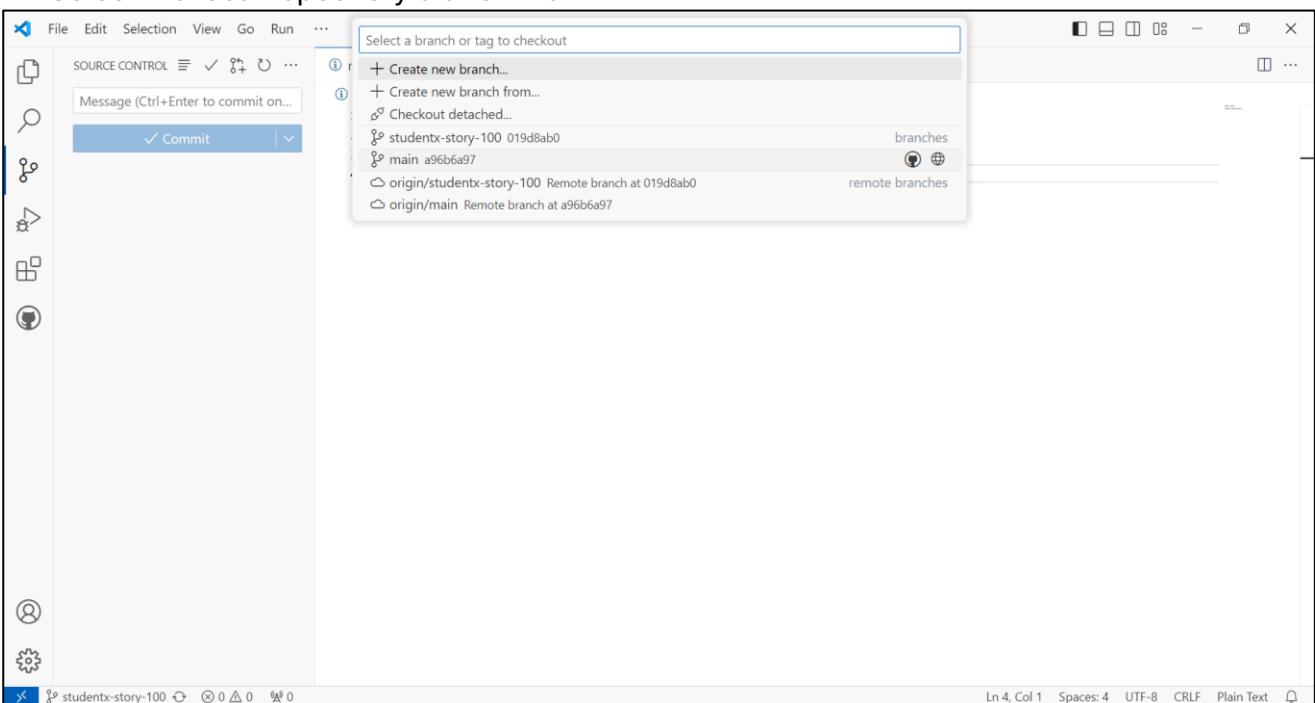
Once StudentX has completed the feature, it is time to merge the changes to the main branch
 There are several steps to this process

- 1) checkout to the local repository branch main
- 2) merge from the story branch to the main branch

X 26. Click the ellipsis, select **Checkout to..**



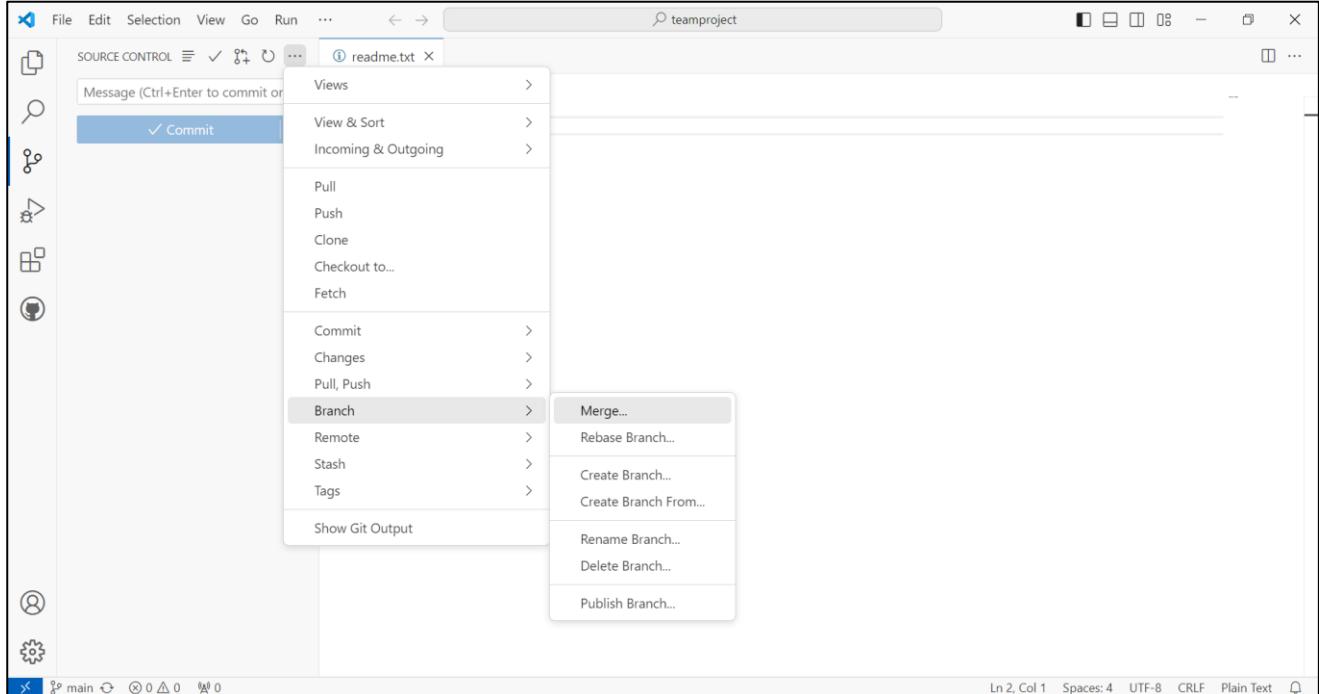
X 27. Select the local repository branch **main**



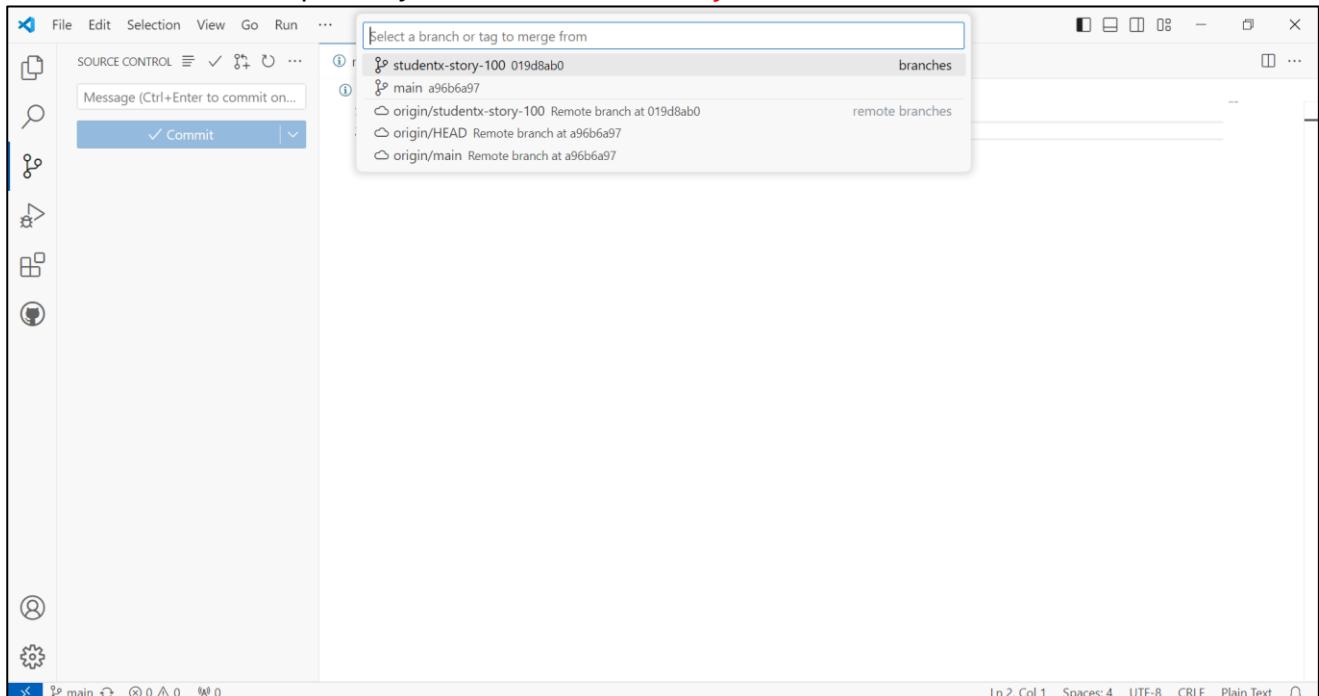
Merge branch

The bottom left corner in Visual Studio Code shows you are on the local repository branch **main**. We now need to merge from the local repository branch **studentx-story-100** to the local repository branch **main**.

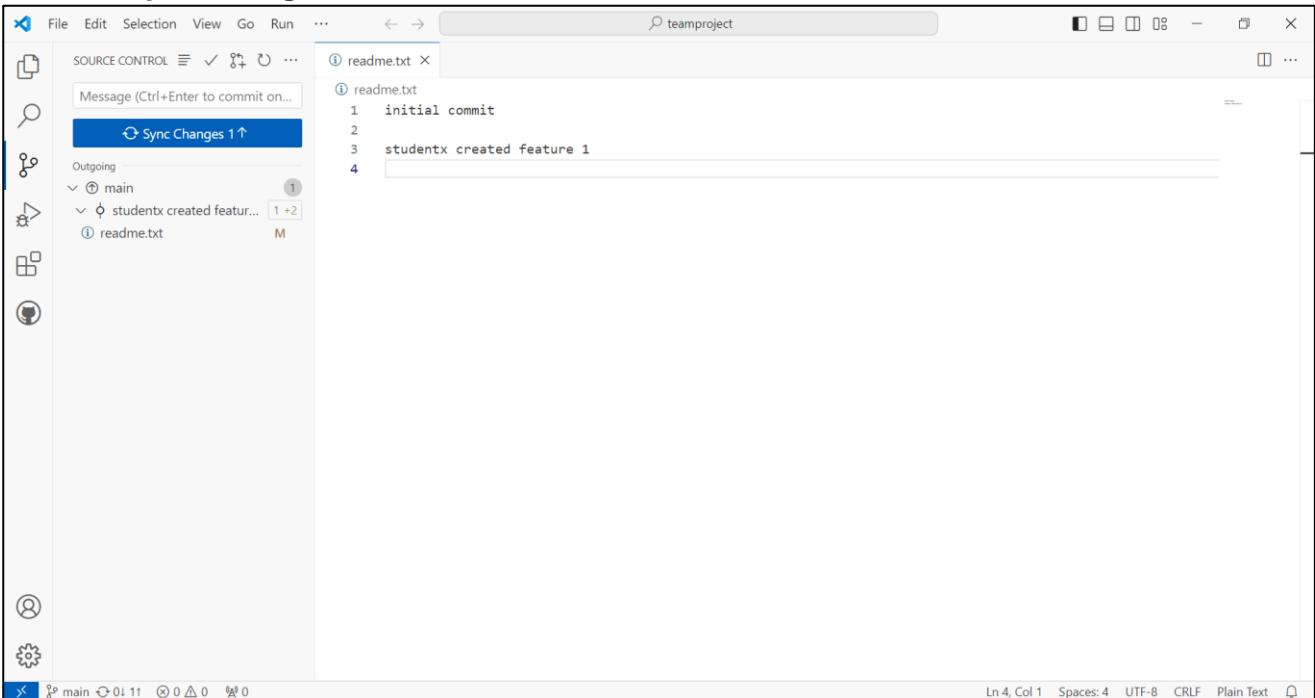
- X 28. Click on the ellipsis, select **Branch, Merge..**



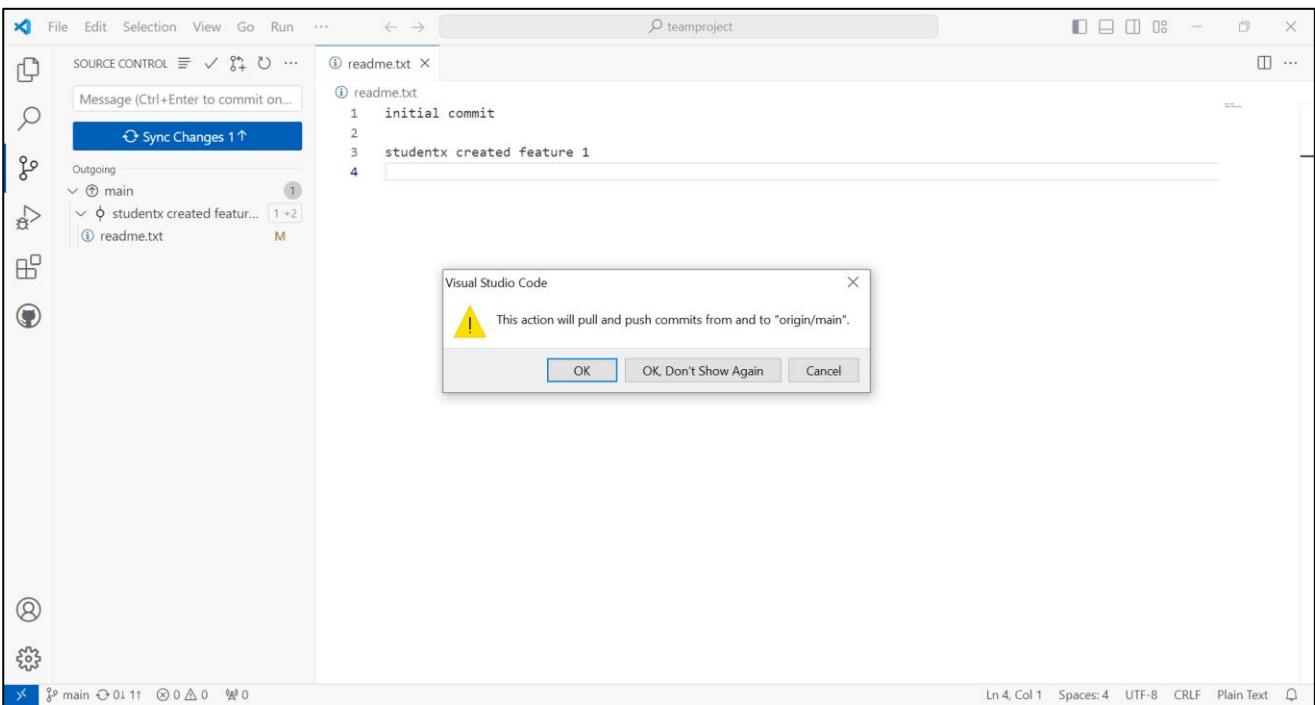
- X 29. Select the local repository branch **studentx-story-100**



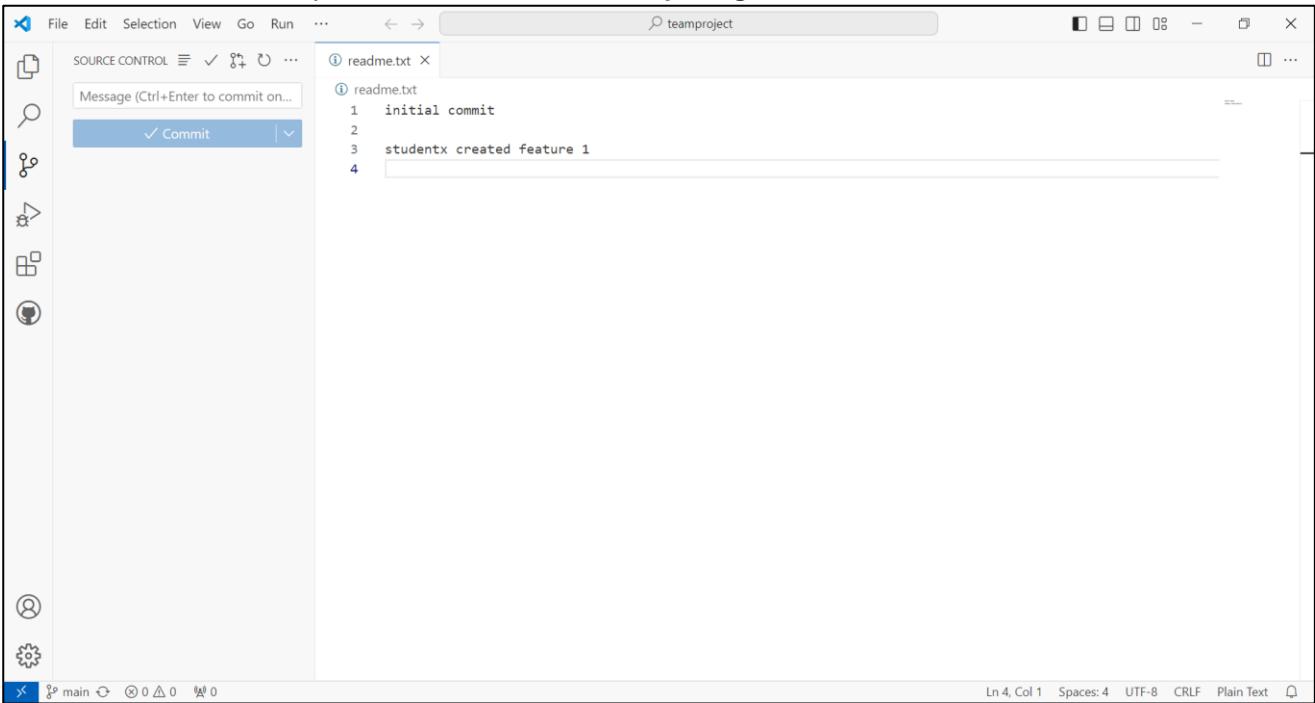
X 30. Click Sync Changes



X 31. The dialog informs you that sync changes will pull and push commits from and to origin/main Click OK



X 32. The source control panel should not show anything of note



The screenshot shows the Visual Studio Code interface with the Source Control panel open. The panel displays a single commit message for a file named 'readme.txt'. The commit history is as follows:

```
① readme.txt ×  
① readme.txt  
1 initial commit  
2  
3 studentx created feature 1  
4
```

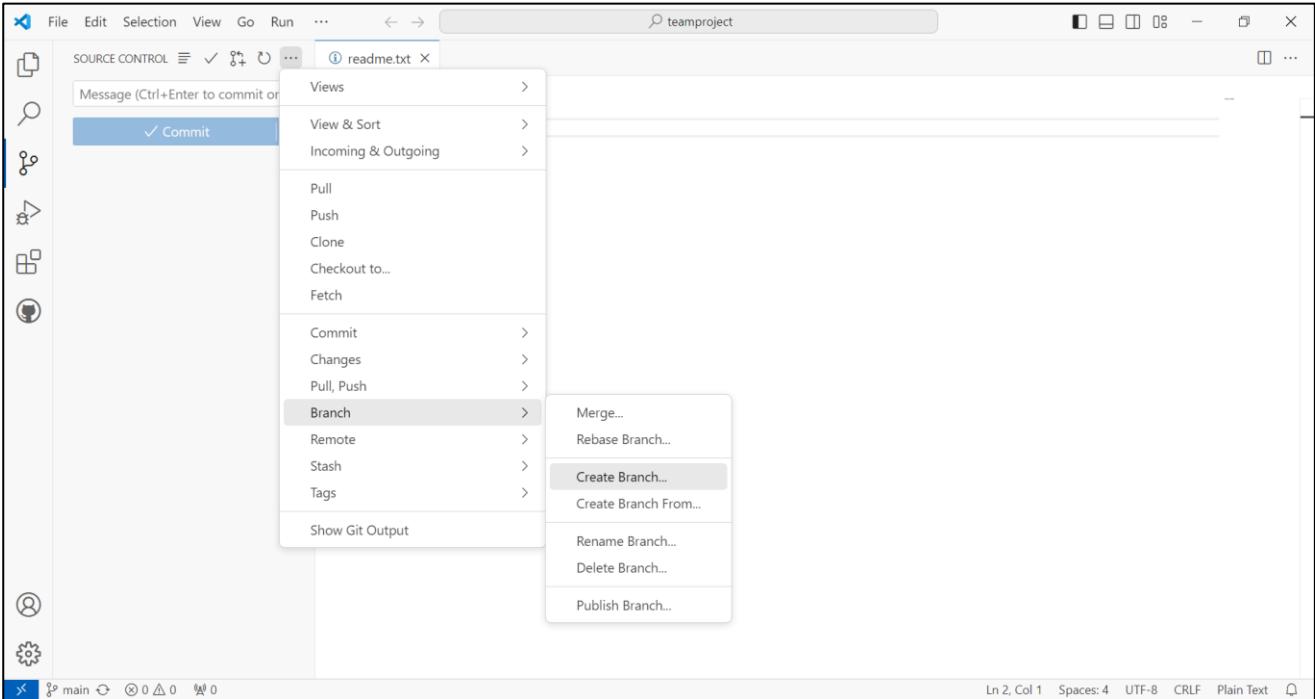
The commit message is: "initial commit". The commit author is "studentx" and the commit subject is "created feature 1". The commit count is 4. The status bar at the bottom indicates the current branch is 'main'.

At this point, StudentX's local repository branch **main** is in sync with the remote repository branch **main**

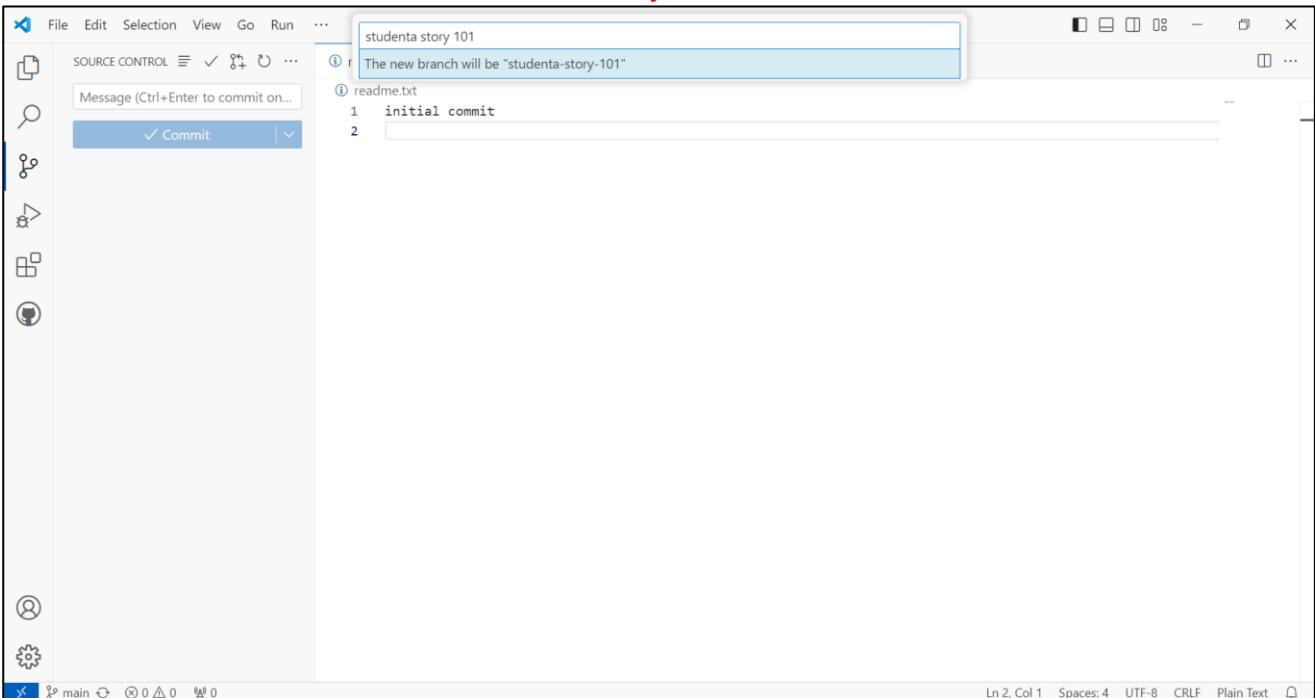
Create branch

The next few steps to create branch, publish branch, make changes, stage changes, commit changes, sync changes are the same as what StudentX did

- A 33. Click on the ellipsis, select **Branch**, **Create Branch...**

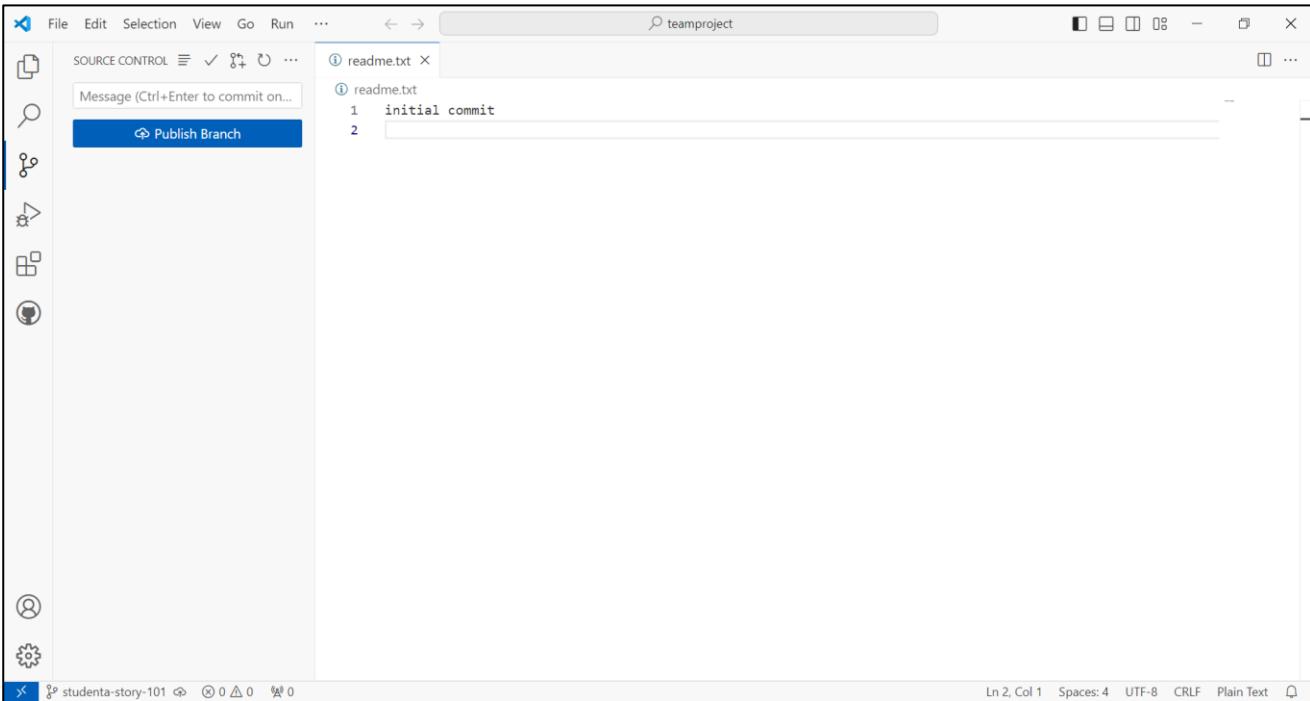


- A 34. Enter the branch name as **studenta_story_101**
 The new branch will be called **studenta-story-101**

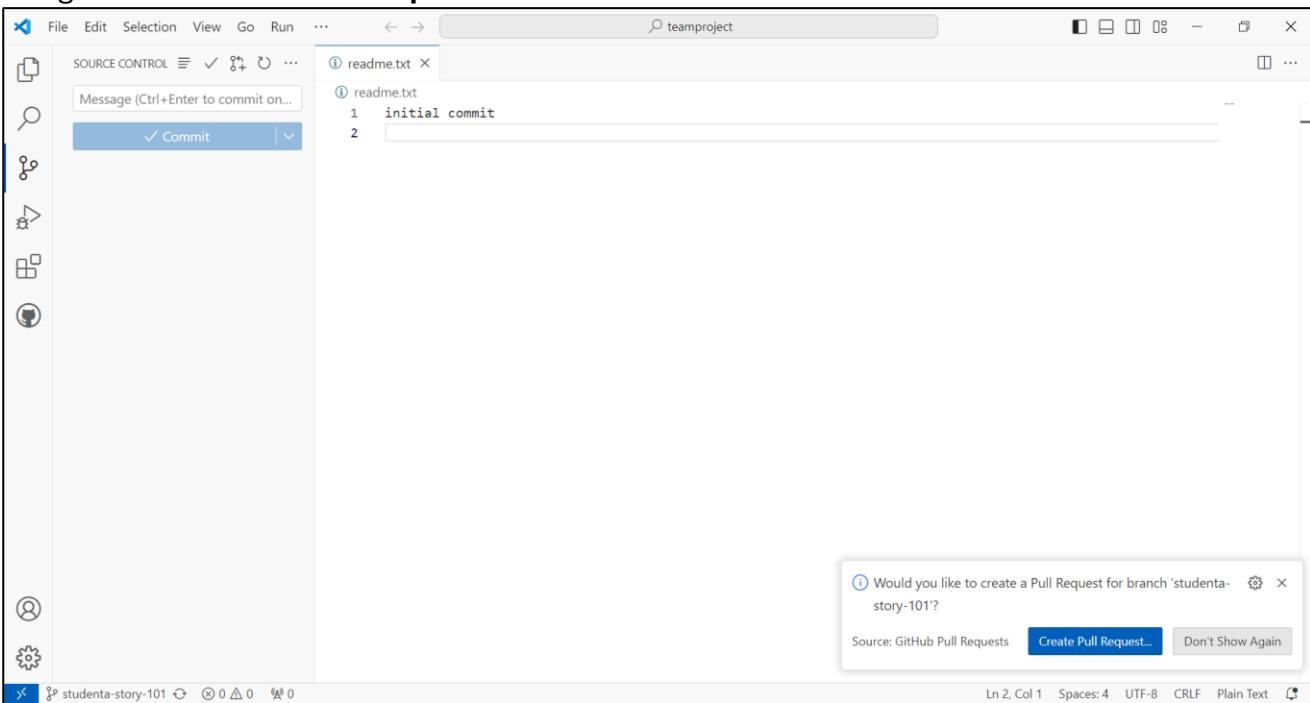


Publish branch

A 35. Click Publish Branch

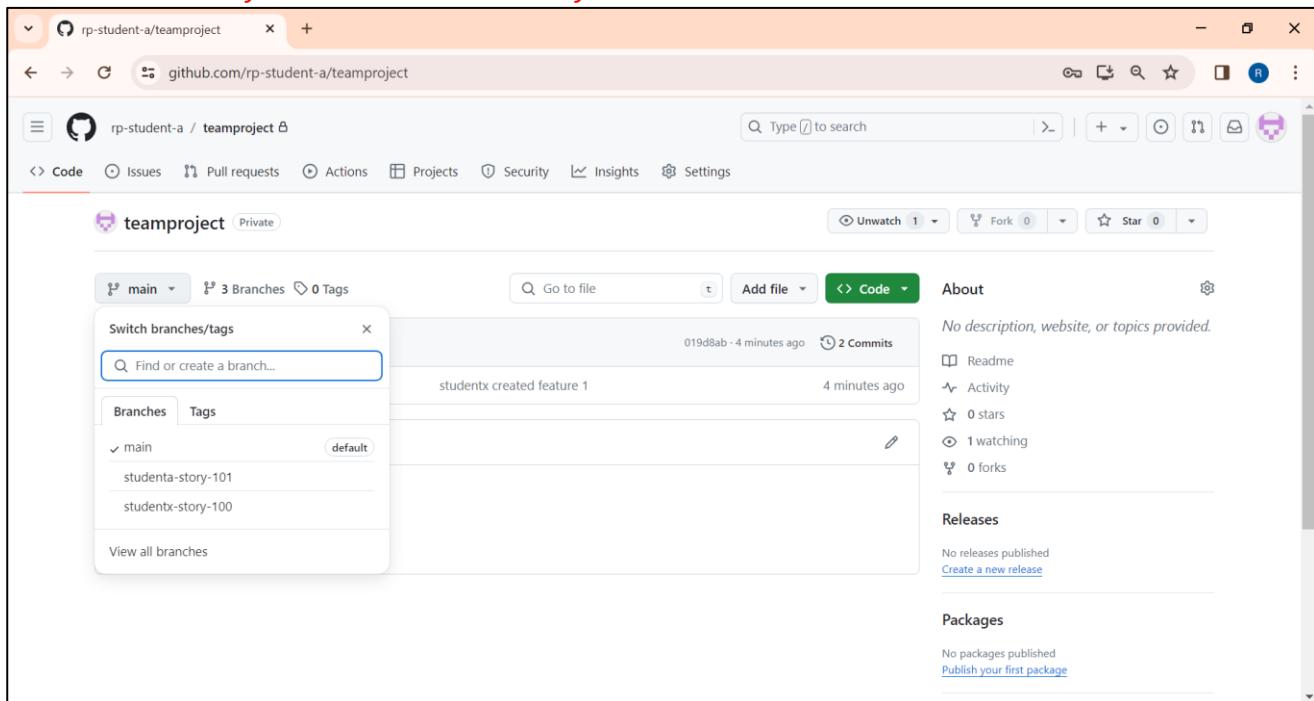


A 36. Ignore the Create Pull Request... button



A 37. View the repository in GitHub in a browser

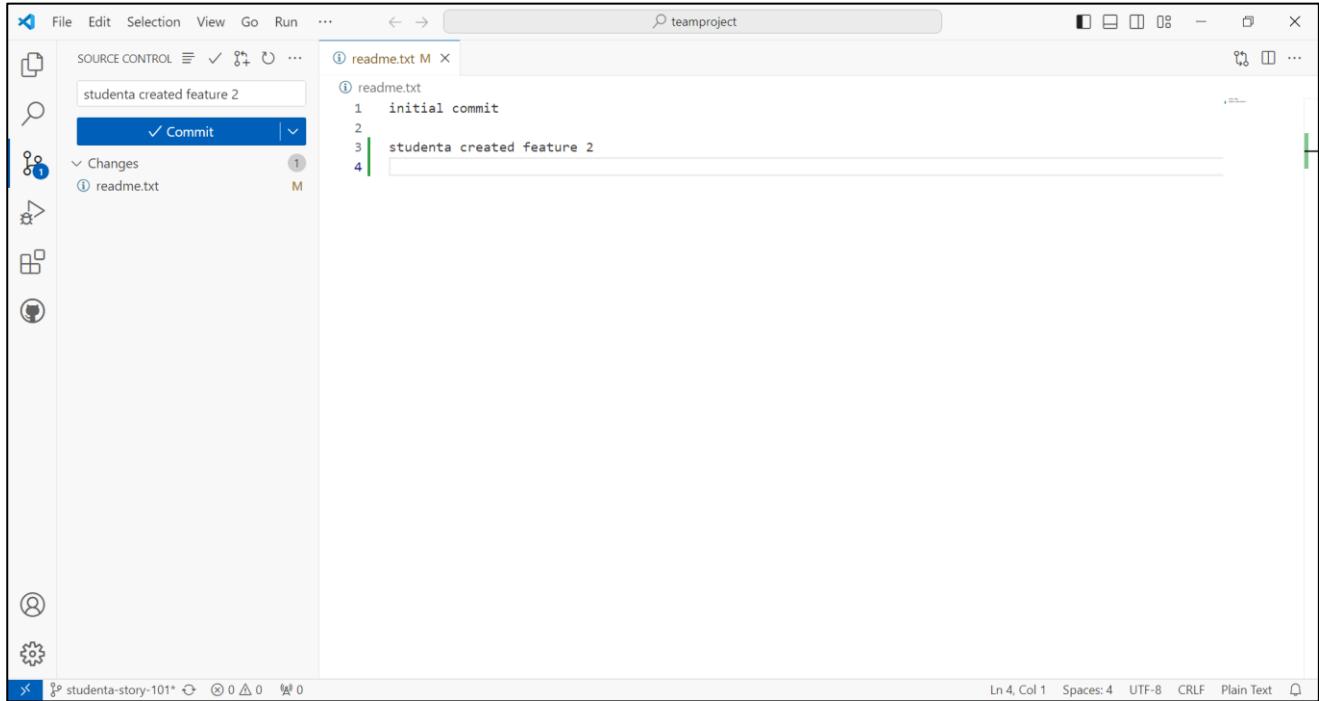
Click on the dropdown next to **main**, you should see that there are 3 branches - **main**, **studentx-story-100** and **studenta-story-101**



The screenshot shows a GitHub repository page for 'rp-student-a/teamproject'. The 'Code' tab is active. On the left, a modal window titled 'Switch branches/tags' is open, showing a dropdown menu with 'main' selected and '3 Branches' listed. Below the dropdown are tabs for 'Branches' and 'Tags', with 'Branches' currently selected. Under 'Branches', three branches are listed: 'main' (selected), 'studenta-story-101', and 'studentx-story-100'. A 'View all branches' link is also present. The main content area displays a single commit by 'studentx' with the message 'created feature 1', timestamped 4 minutes ago. The commit has 2 commits. To the right of the commit, there's an 'About' section with a note 'No description, website, or topics provided.', and sections for 'Releases' (no releases published) and 'Packages' (no packages published). The top navigation bar includes links for 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'.

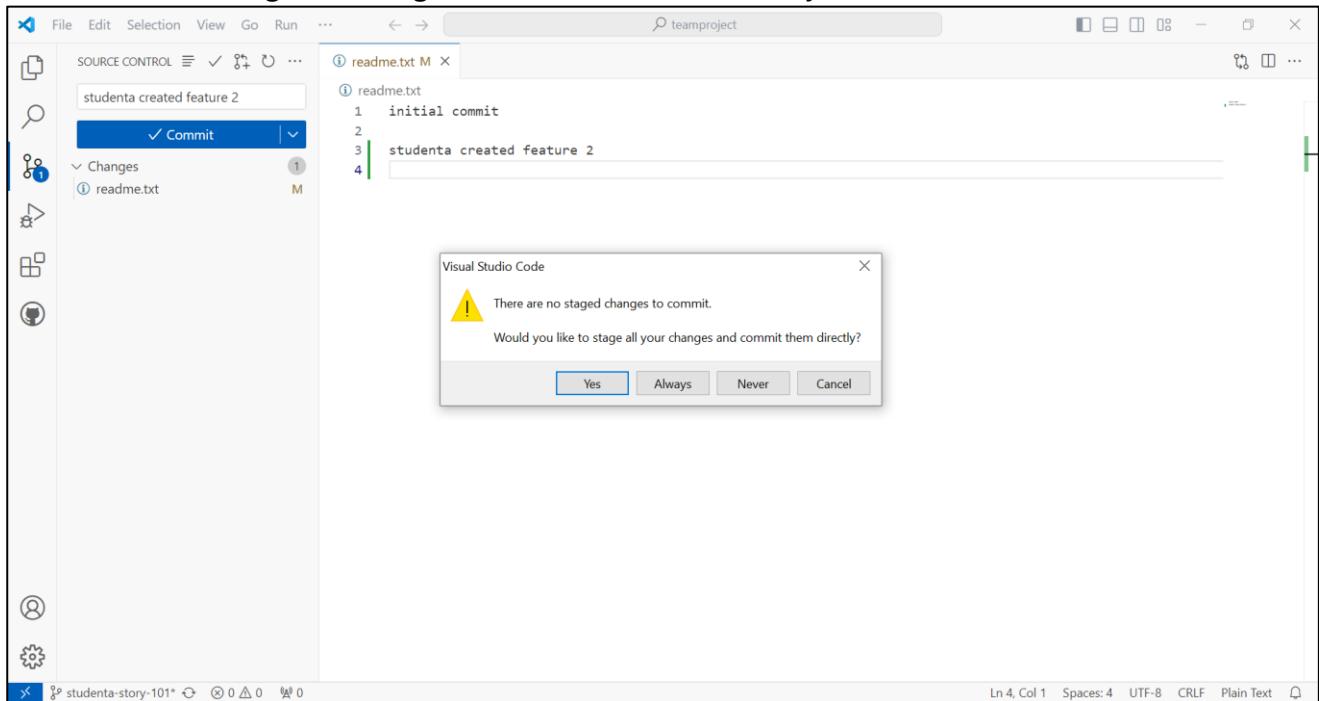
Make changes, stage changes, commit changes, sync changes

- A 38. Modify `readme.txt` by adding the text `studenta created feature 2` on line 3 and add a new line at line 4
Set the commit message as `studenta created feature 2`
Click **Commit**

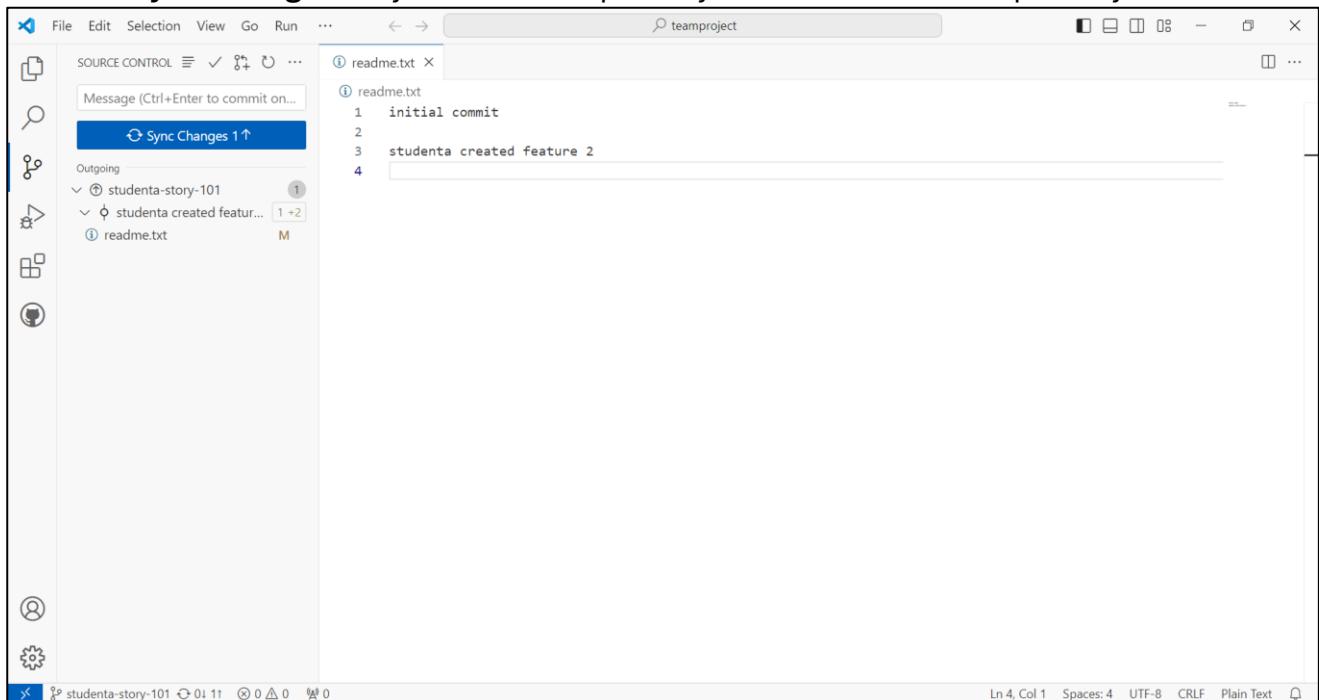


If you try to commit before staging changes, Visual Studio Code will ask if you want to stage **all** changes. Only do so if you are certain that you want to stage all changes

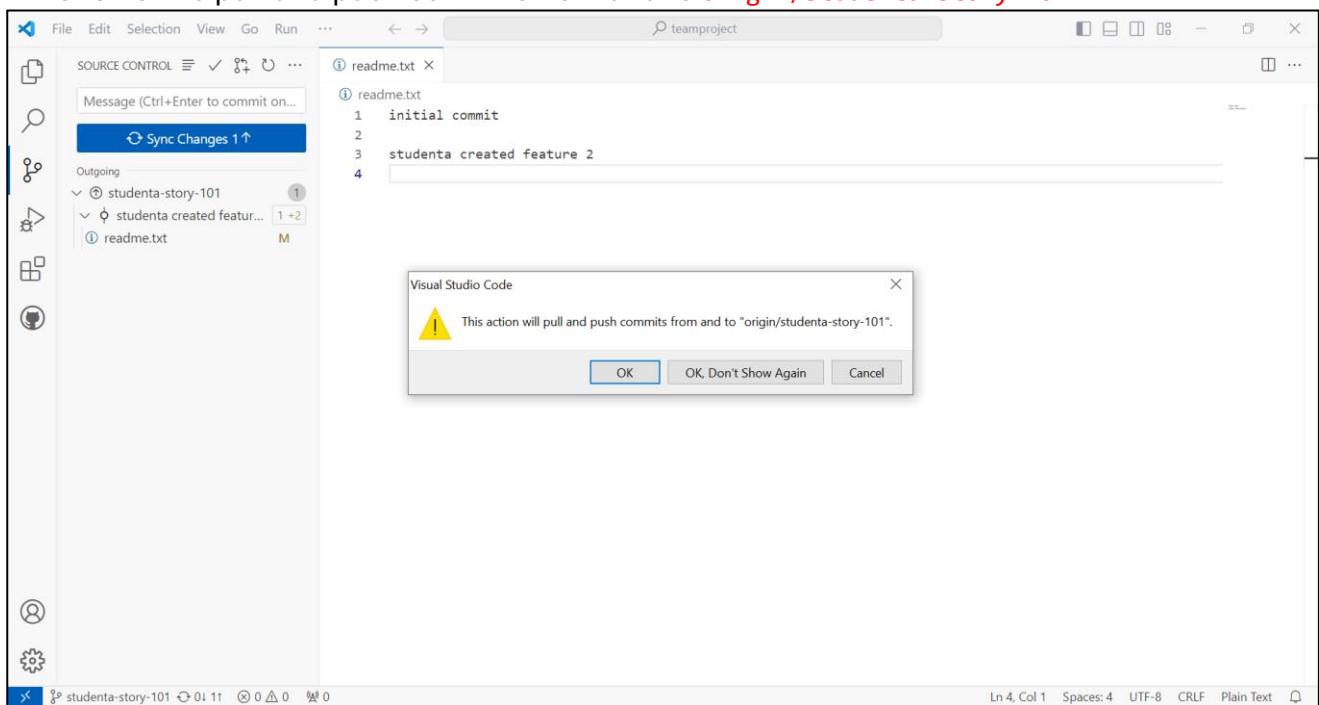
- A 39. Click **Yes** to stage all changes and commit them directly



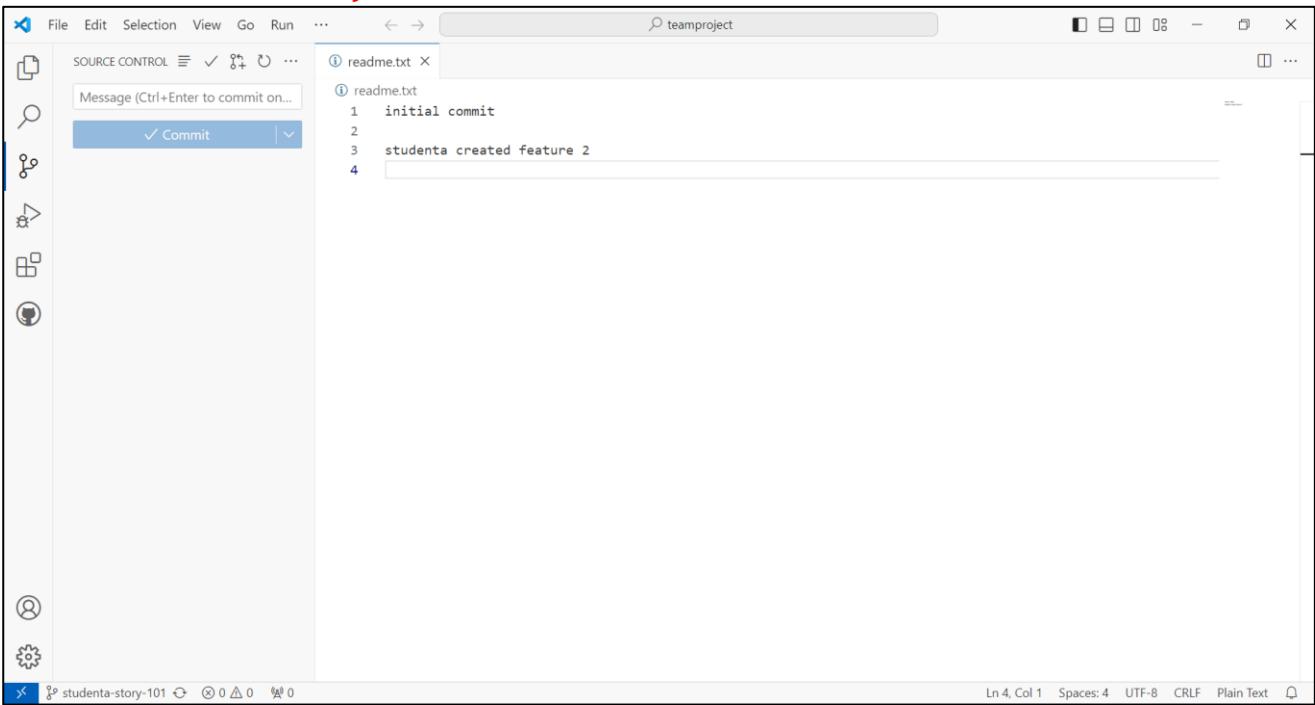
A 40. Click **Sync Changes** to sync the local repository branch and remote repository branch



A 41. Click **OK** to pull and push commits from and to [origin/studenta-story-101](#)



- A 42. The remote repository branch **studenta-story-101** is now in sync with the local repository branch **studenta-story-101**



The screenshot shows the Visual Studio Code interface with the Source Control tab selected. A commit message is open for the file 'readme.txt'. The message contains the following text:

```
initial commit
studenta created feature 2
```

The commit message field at the bottom of the Source Control panel shows the message 'studenta created feature 2'.

Note that at this point, the local repository branch **main** is out of sync with the remote repository branch **main**

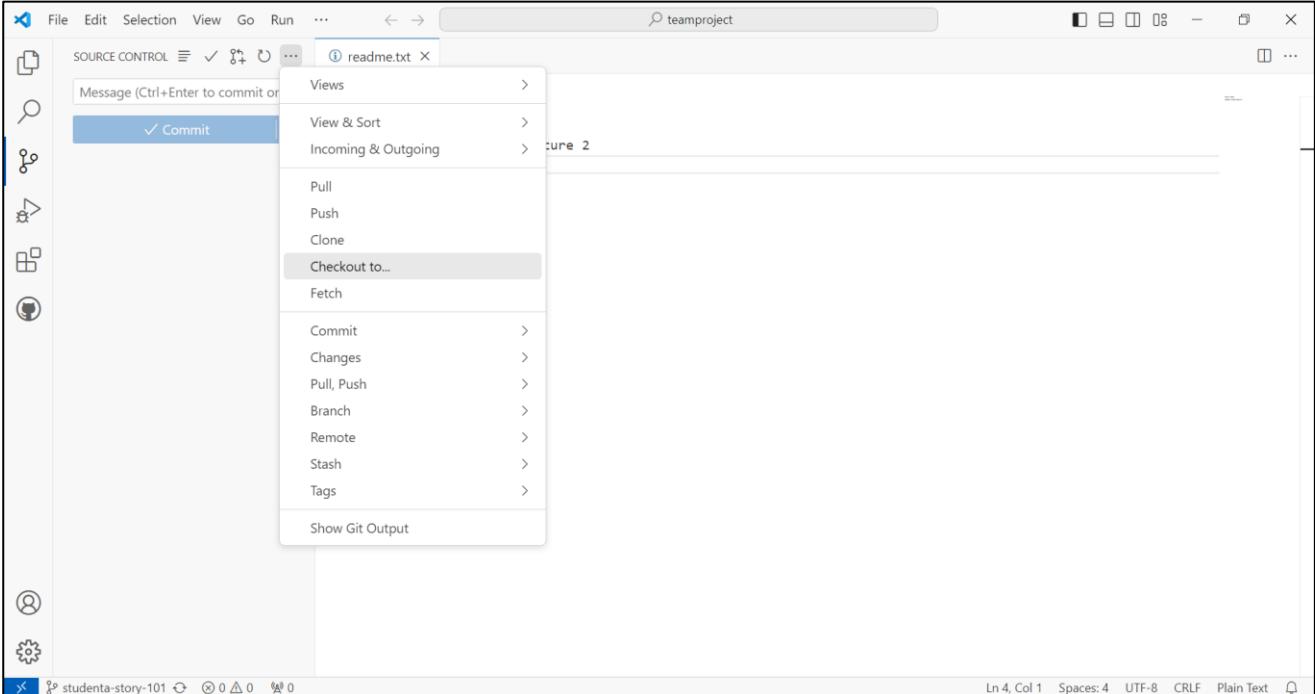
In the local repository branch **main**, line 3 says **studenta created feature 2**

In the remote repository branch **main**, line 3 says **studentx created feature 1**

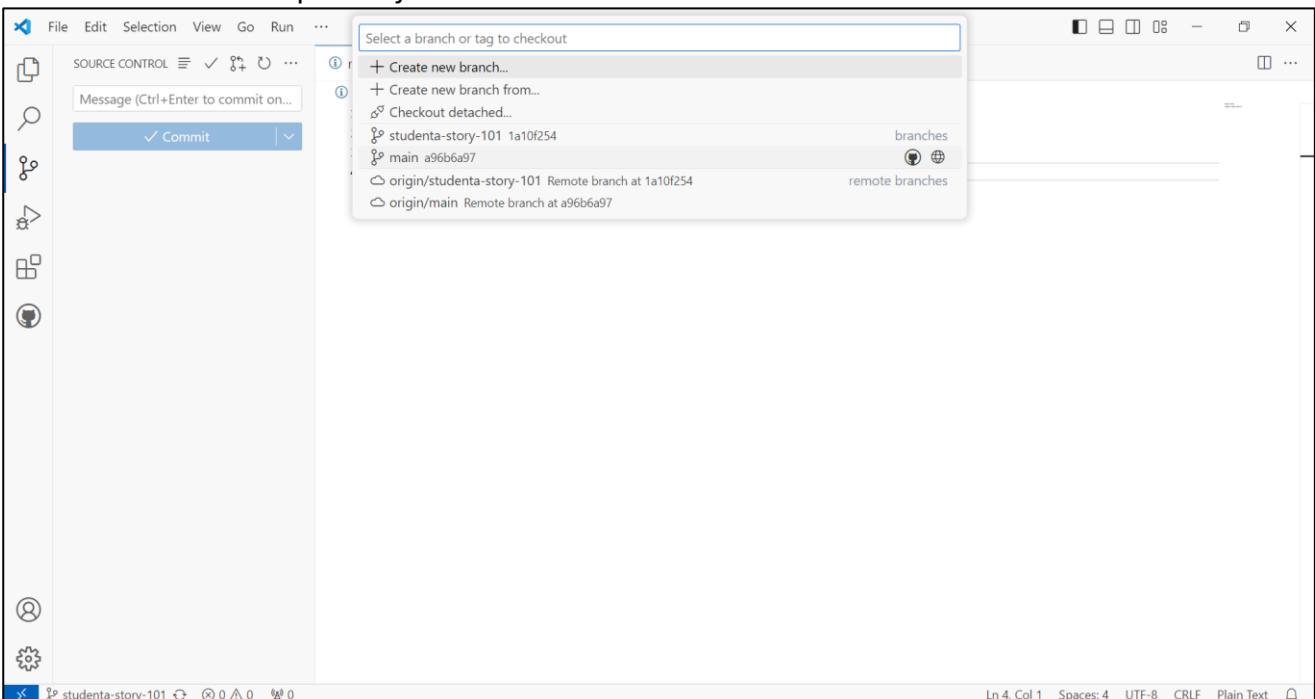
Checkout main

The next few steps to checkout main and merge branch are the same as what StudentX did
 However, there will be a conflict we need to resolve when doing the merge branch

- A 43. Click the ellipsis, select **Checkout to..**

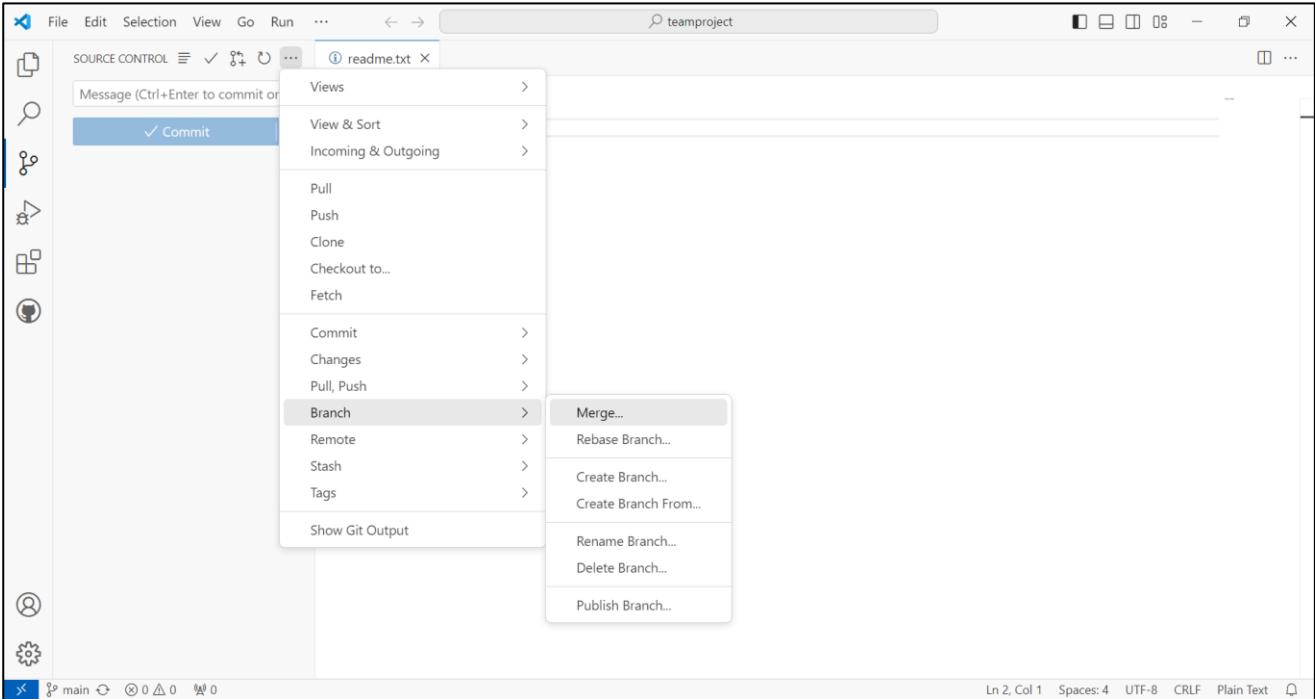


- A 44. Select the local repository branch **main**

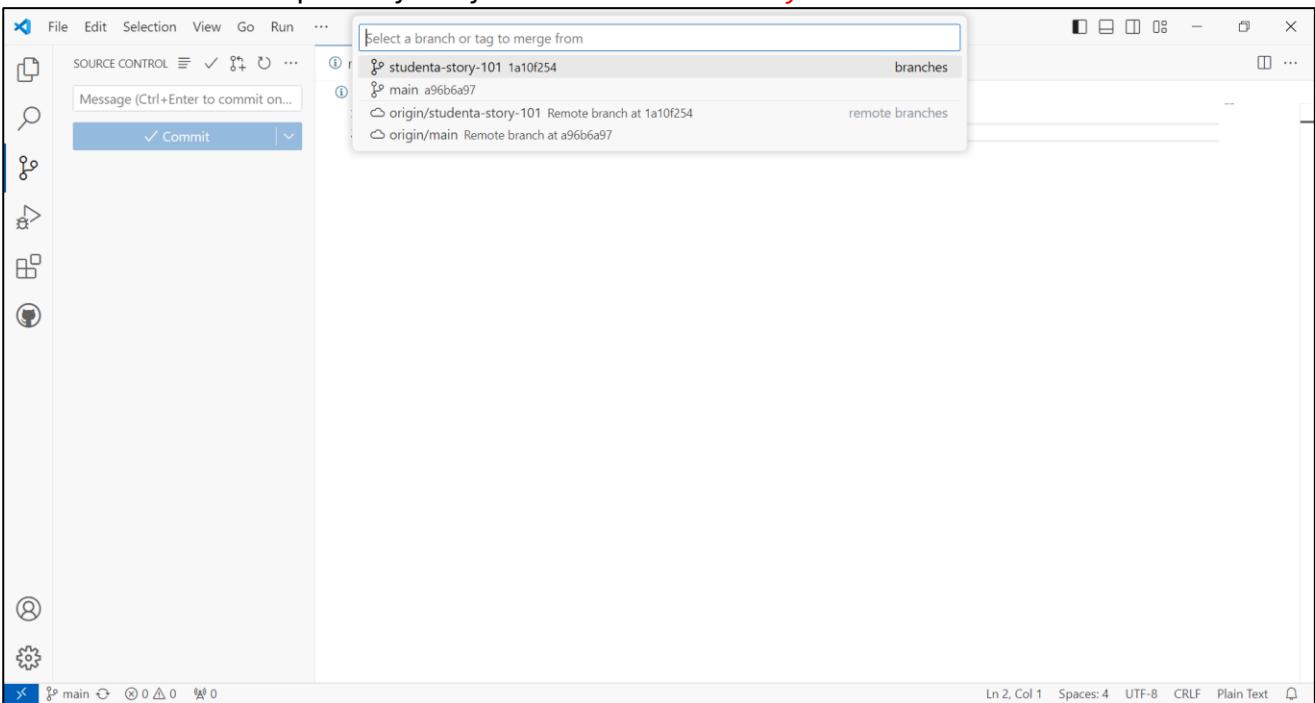


Merge branch

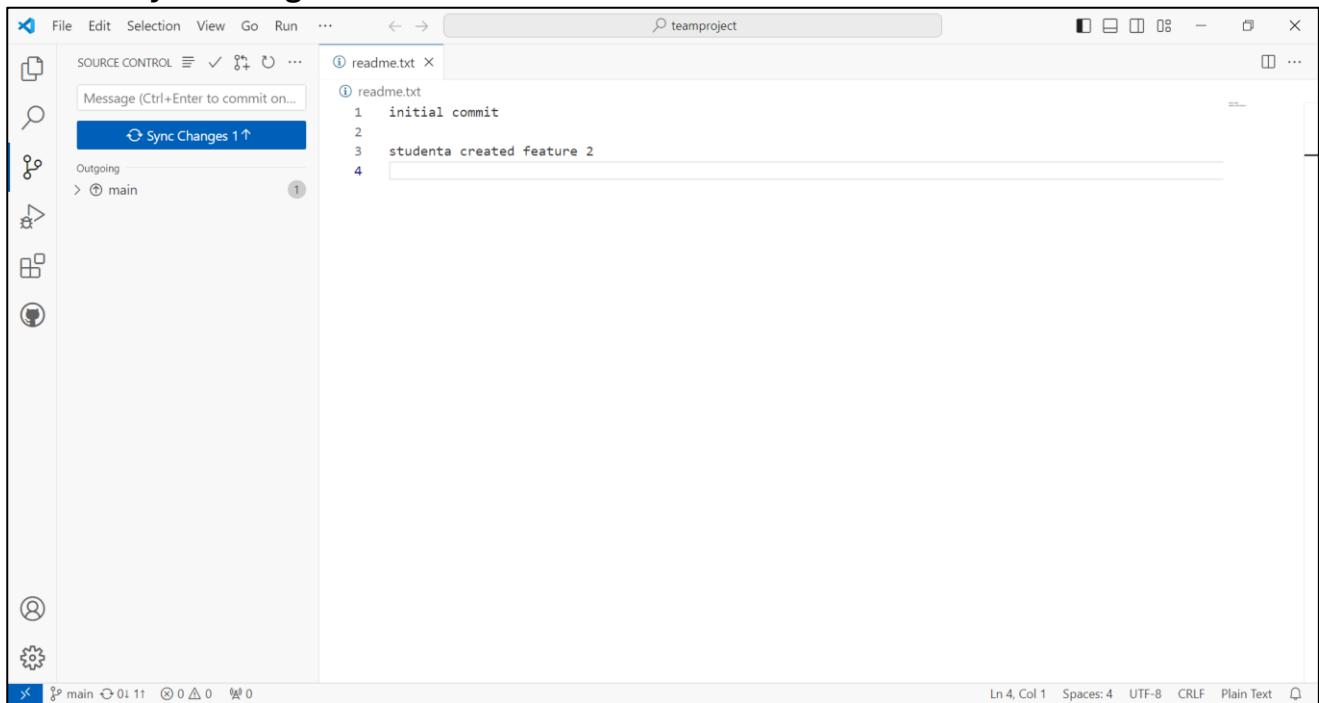
A 45. Click on the ellipsis, select **Branch, Merge..**



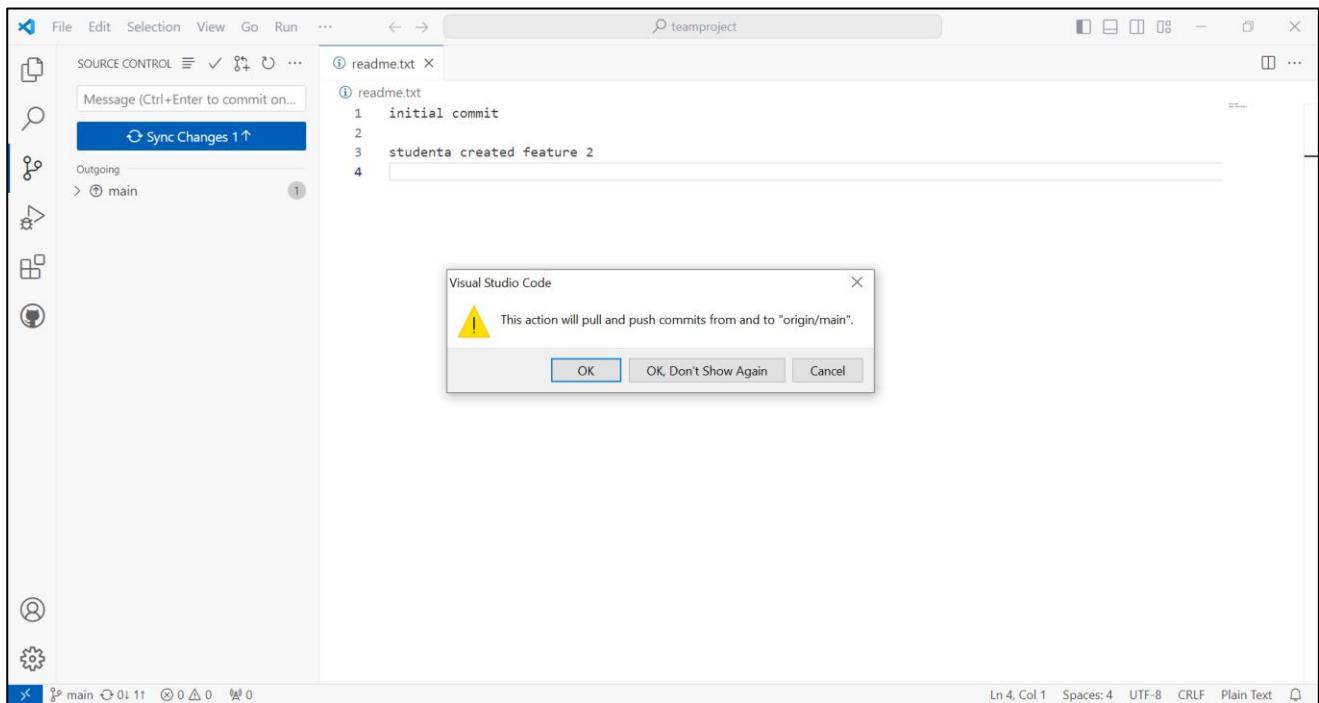
A 46. Select the local repository story branch **studenta-story-101**



A 47. Click Sync Changes



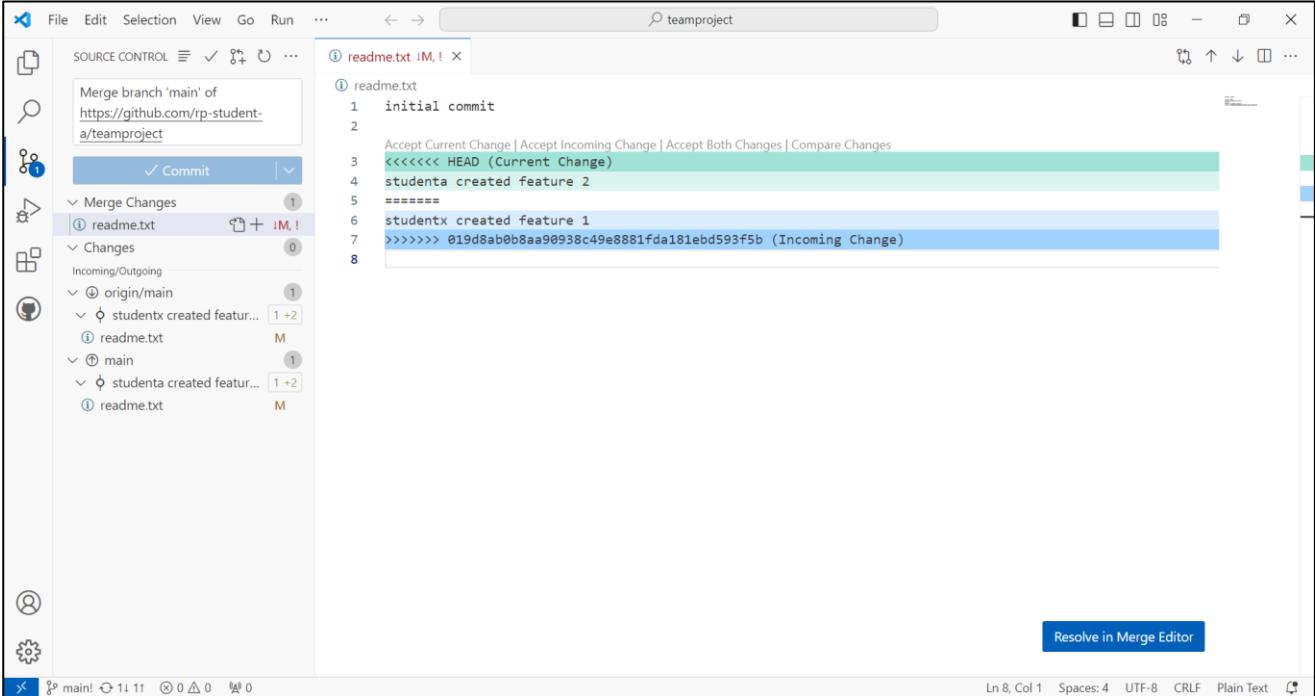
A 48. Click OK



Resolve conflicts

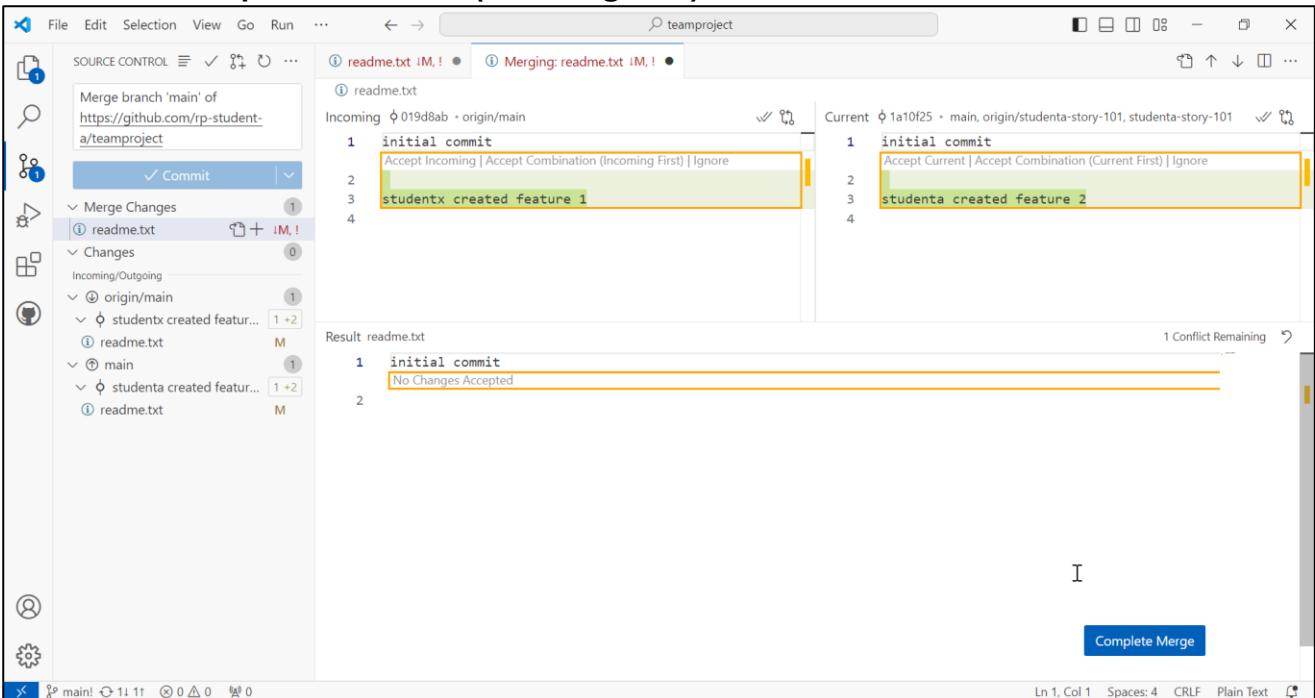
A 49. There is a conflict when trying to merge `readme.txt`

Click **Resolve in Merge Editor**



```
readme.txt IM! x
① readme.txt
1 initial commit
2
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3 <<<< HEAD (Current Change)
4 studenta created feature 2
5 =====
6 studentx created feature 1
7 >>>> 019d8ab0b8aa90938c49e8881fda181ebd593f5b (Incoming Change)
8
```

A 50. Click on **Accept Combination (Incoming First)**



```
readme.txt IM! x Merging: readme.txt IM! !
① readme.txt
Incoming ⚡ 019d8ab · origin/main
1 initial commit
Accept Incoming | Accept Combination (Incoming First) | Ignore
2
3 studentx created feature 1
4
Current ⚡ 1a10f25 · main, origin/studenta-story-101, studenta-story-101
1 initial commit
Accept Current | Accept Combination (Current First) | Ignore
2
3 studenta created feature 2
4
```

Result readme.txt

```
1 initial commit
No Changes Accepted
2
```

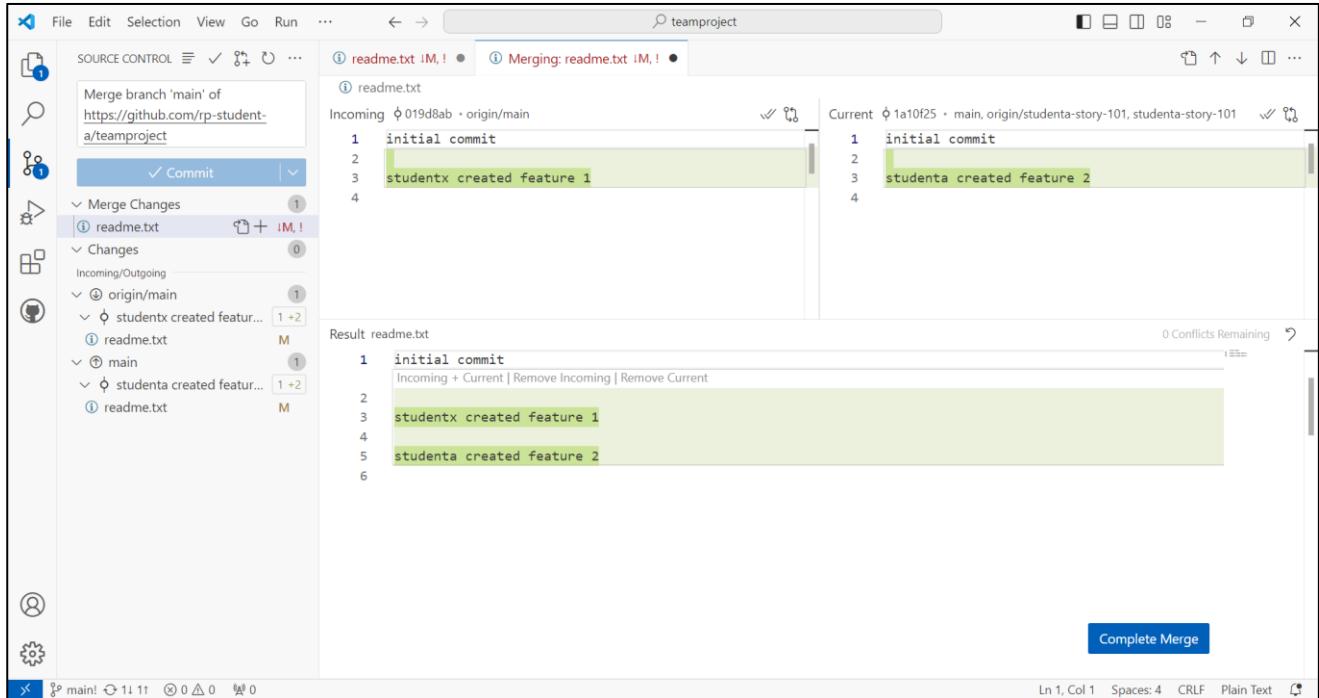
1 Conflict Remaining

Complete Merge

There are several options for how to resolve the merge - accept incoming, accept current, accept combination, ignore, etc, or you can manually edit the contents of the file in the bottom panel

When merging files with conflicts, talk to the person who did the previous merge for that file to understand the changes and what the final merged version should contain

A 51. Check that the resultant `readme.txt` looks as shown below then click **Complete Merge**

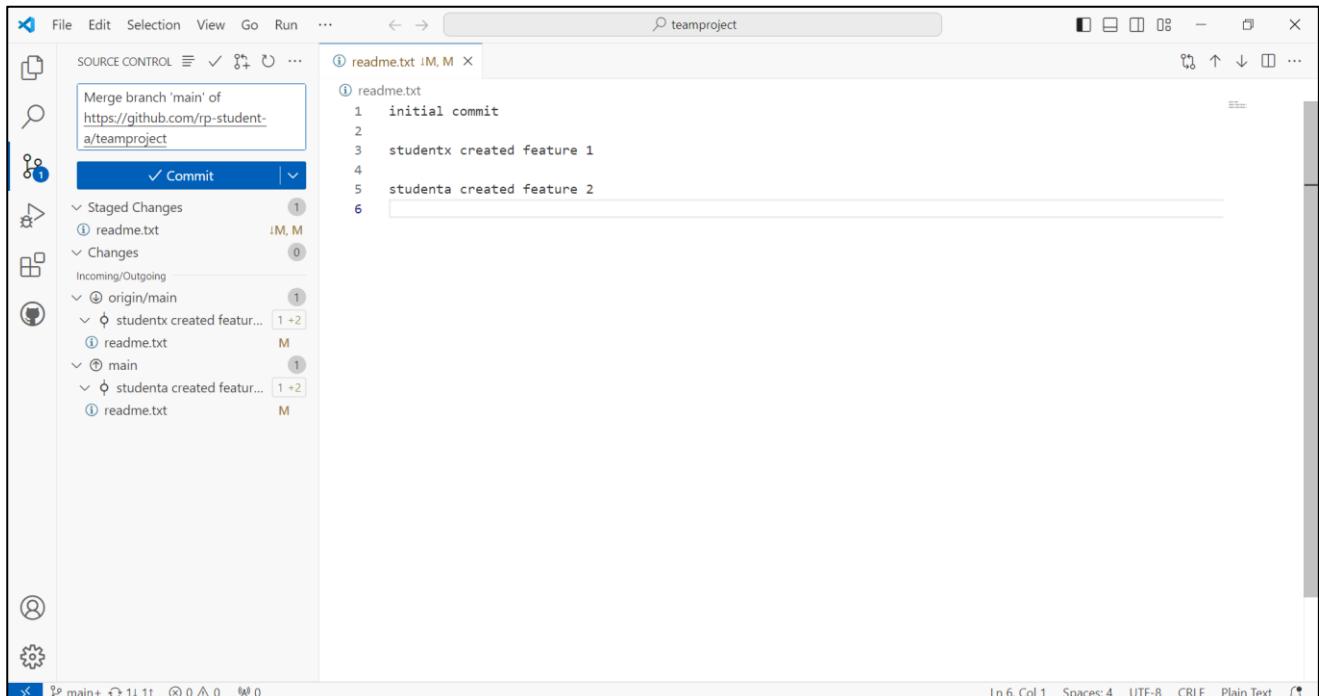


```

1 initial commit
2
3 studentx created feature 1
4
5 studenta created feature 2
6

```

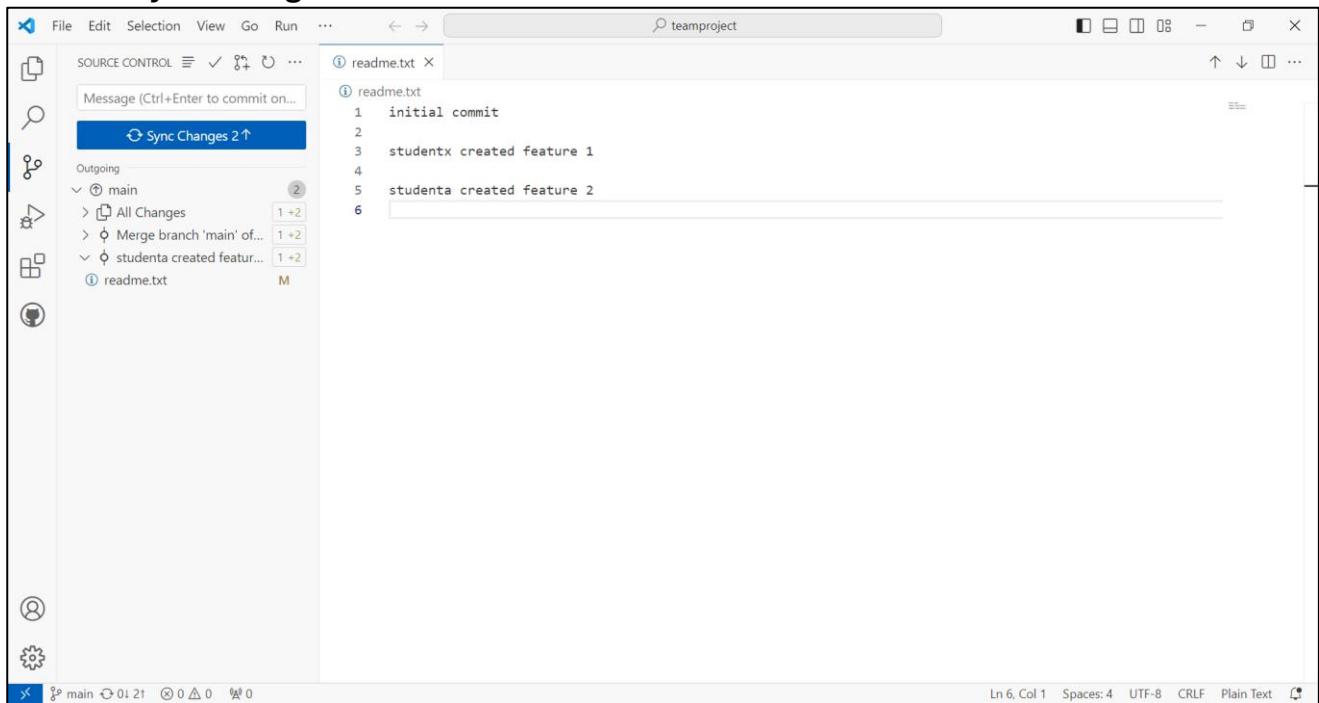
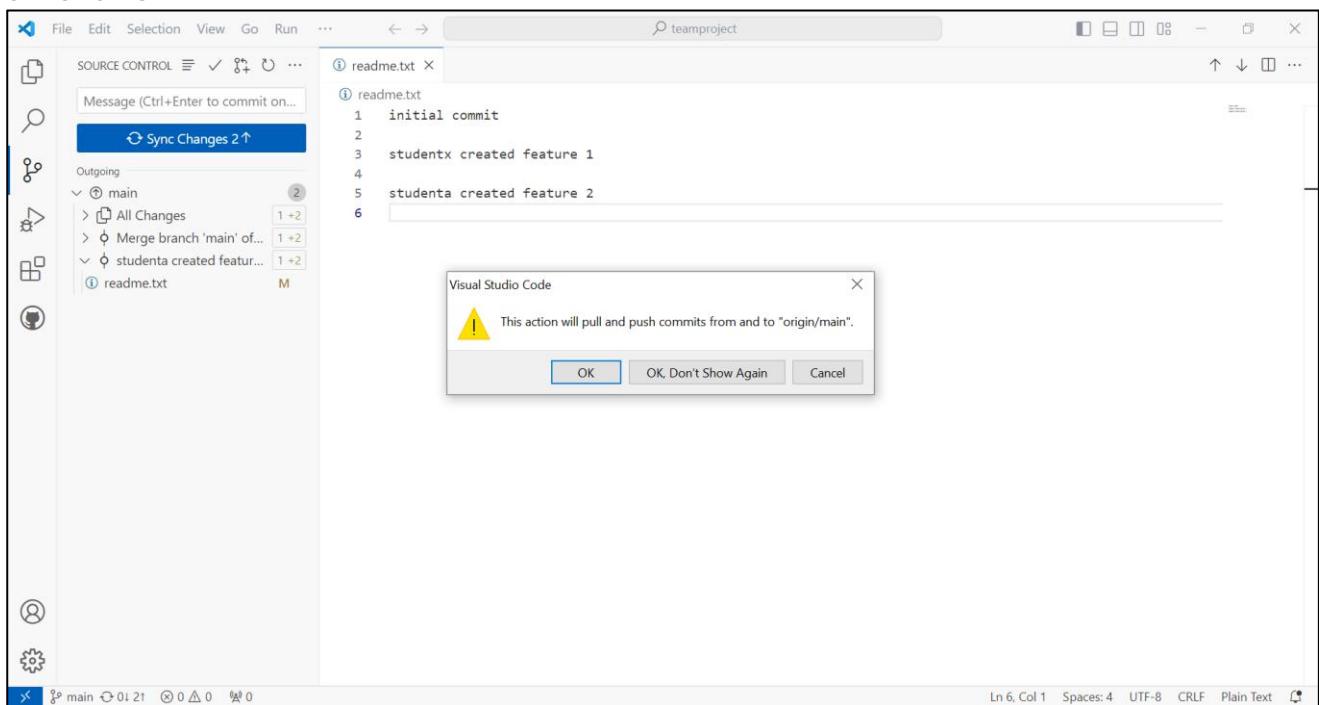
A 52. Click Commit



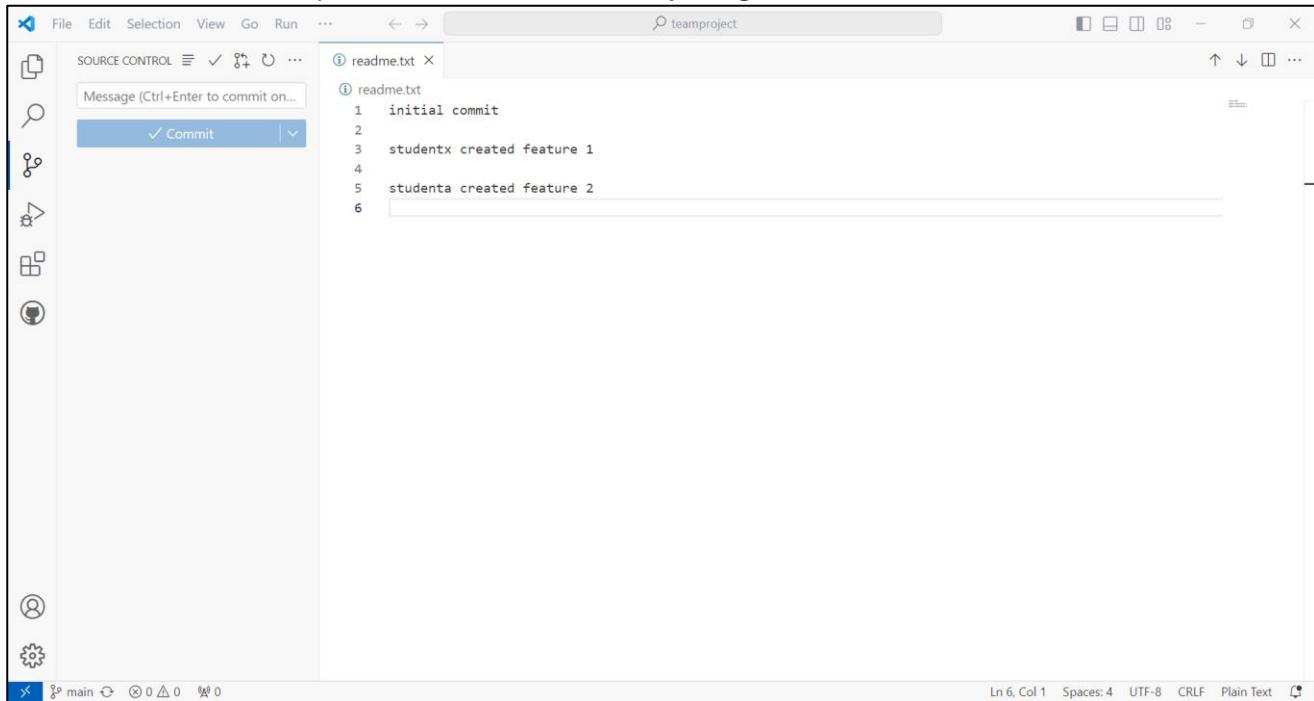
```

1 initial commit
2
3 studentx created feature 1
4
5 studenta created feature 2
6

```

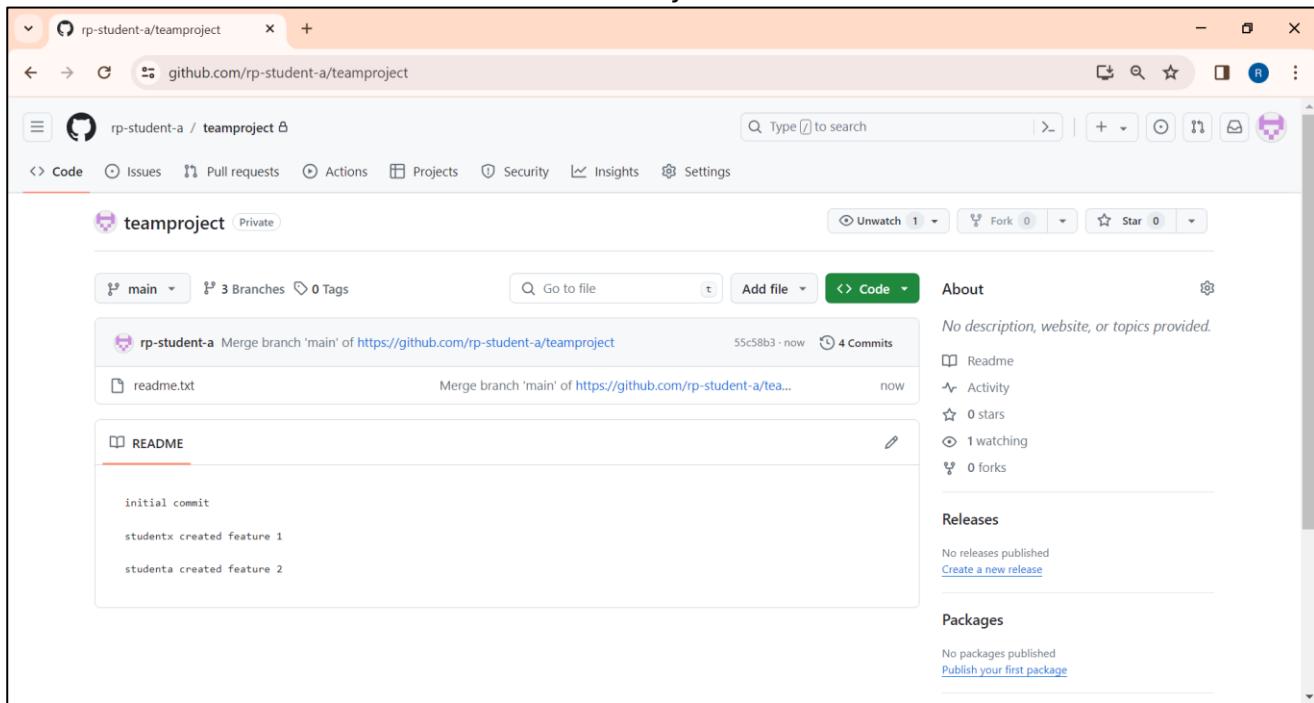
A 53. Click Sync Changes**A 54. Click OK**

A 55. The source control panel should not show anything of note



The screenshot shows the Visual Studio Code interface with the Source Control panel open. The commit history for 'readme.txt' is displayed, showing two commits from 'studentx' and 'studenta'. A message input field is present, and a 'Commit' button is highlighted.

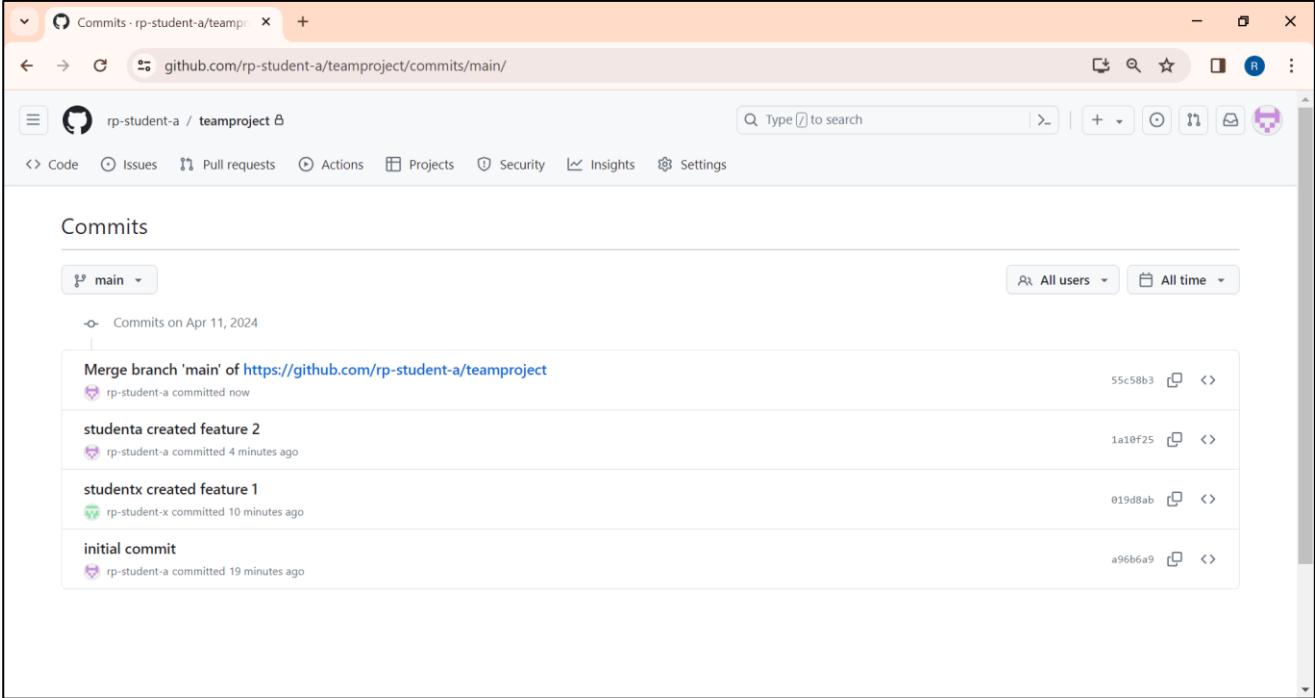
A 56. Check the project in GitHub in a web browser
Click on **4 Commits** to see the commit history



The screenshot shows a GitHub repository page for 'rp-student-a/teamproject'. The 'Code' tab is selected. A commit history is shown with 4 commits. The first commit is from 'rp-student-a' and merges the 'main' branch. The commit message is 'Merge branch 'main' of https://github.com/rp-student-a/teamproject'. The commit date is 55c58b3 - now. The commit count is 4 Commits. The commit message for the first commit is 'initial commit'.

A 57. Check the project in GitHub in a web browser

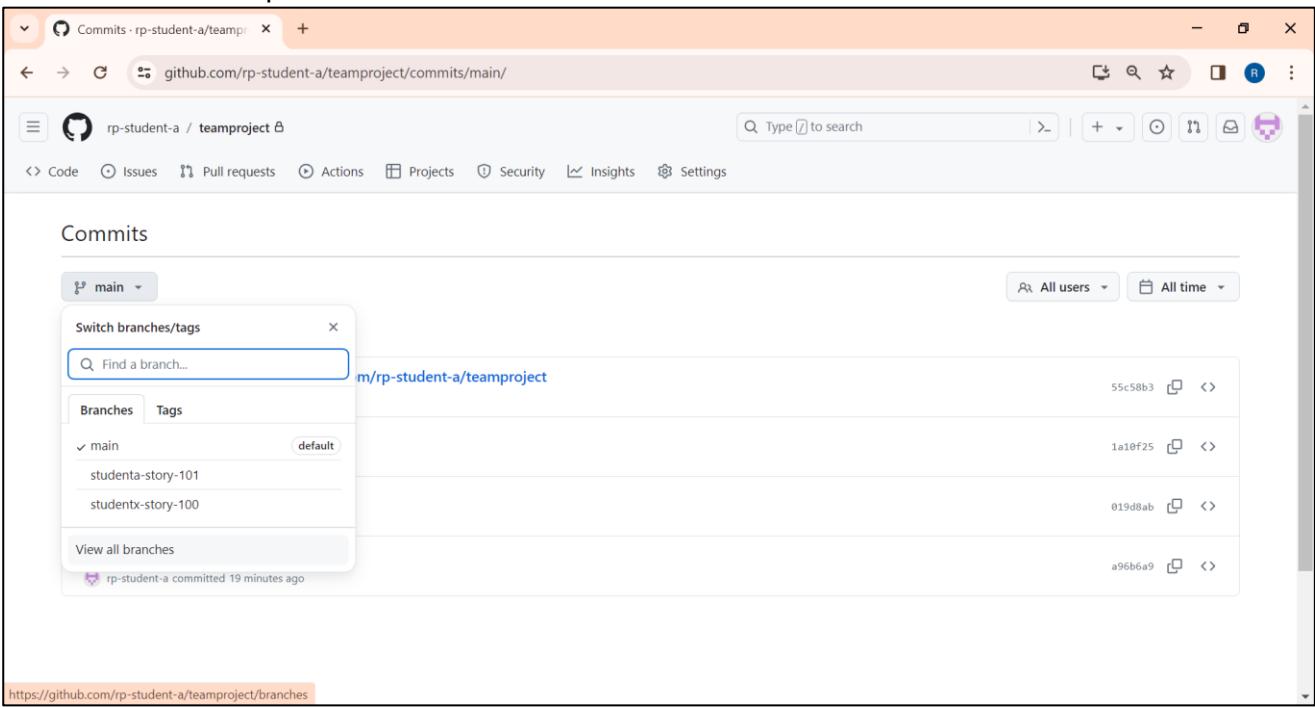
The commit messages are shown here and describe the reason for the commit, hence the need for meaningful commit message



The screenshot shows the GitHub 'Commits' page for the repository 'rp-student-a/teamproject'. The main commit dropdown is set to 'main'. It displays four commits:

- Merge branch 'main' of <https://github.com/rp-student-a/teamproject> (rp-student-a committed now)
- studenta created feature 2 (rp-student-a committed 4 minutes ago)
- studentx created feature 1 (rp-student-x committed 10 minutes ago)
- initial commit (rp-student-a committed 19 minutes ago)

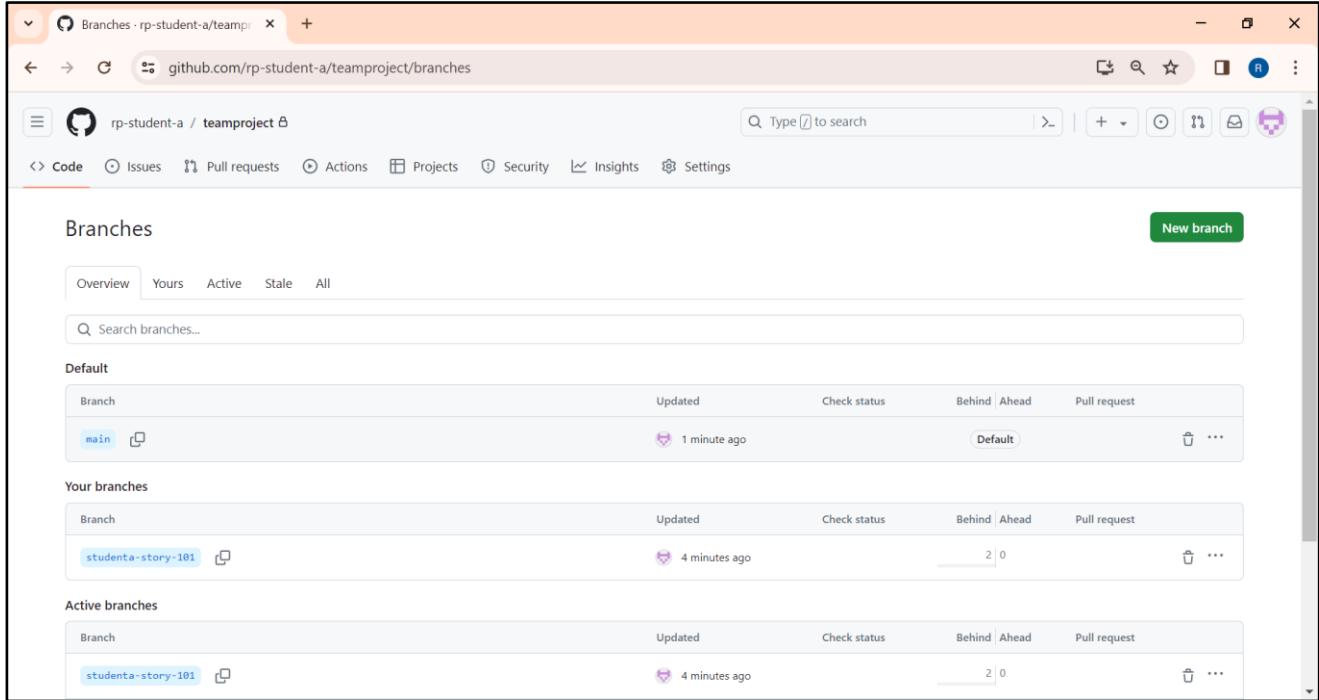
A 58. Click on the dropdown next to main to see all the branches. Click **View all branches**



The screenshot shows the GitHub 'Commits' page for the repository 'rp-student-a/teamproject'. The main commit dropdown is set to 'main'. A modal window titled 'Switch branches/tags' is open, showing a list of branches and tags. The 'View all branches' option at the bottom is highlighted.

The commit list on the right shows the same four commits as in the previous screenshot.

- A 59. On this page, you can see the branches for this repository. A best practice once the code from a branch has been committed to main is to delete the branch. The commit history for that branch would still exists in the main branch.



Default
Branch
main

Your branches
Branch
student-a-story-101

Active branches
Branch
student-a-story-101

End of exercise