

A short tutorial on Git

Servesesh Muralidharan

4 March 2014

This Tutorial

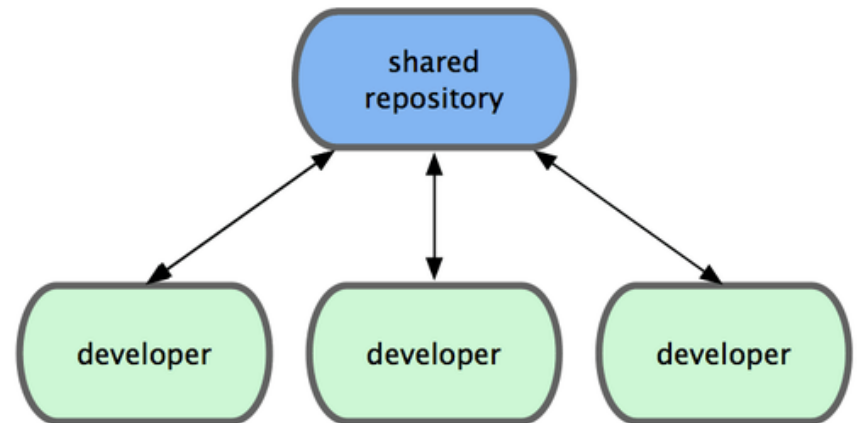
- What is Source Control
- Distributed source control with Git
- Git in Assignment 5
- Using Gitlab
- Using git in Eclipse
- More about Assignment 5

This Tutorial

- What is Source Control
- Distributed source control with Git
- Git in Assignment 5
- Using Gitlab
- Using git in Eclipse
- More about Assignment 5

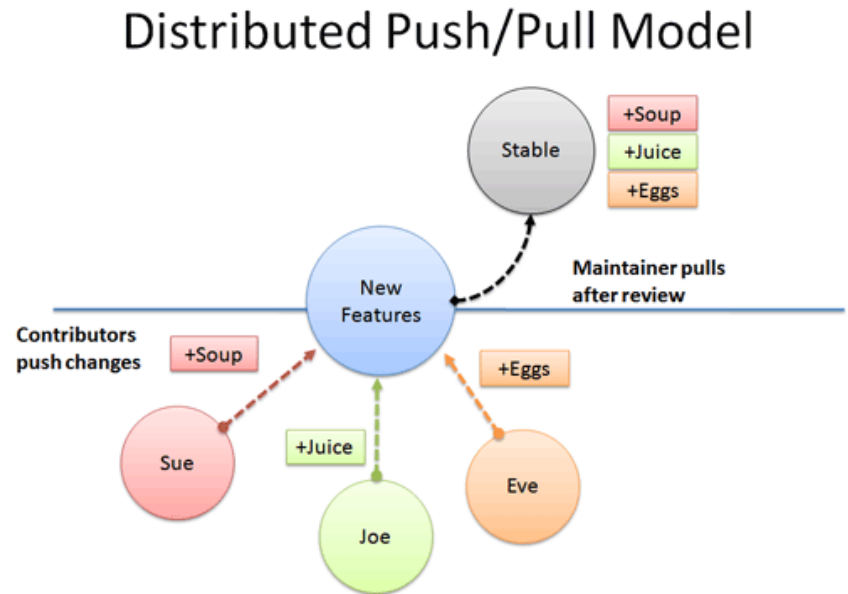
What is Source Control

- Tracks changes to files
- Maintains history of changes
- Maintains snapshots
- Enables collaboration
- Main operations
 - **add**: mark files on the hard drive to be tracked in the repository
 - **commit**: commit to the repository the changes in files on the hard drive
 - ...



Distributed Source Control

- Allows multiple repositories
 - each one is a copy of the **main repository** (usually)
- All repositories can be synchronized
 - **clone**: creates a local copy of the **main repo**
 - **push** the changes from the local repository to the **main repo**
 - **pull** the changes from the **main repo** to the local one



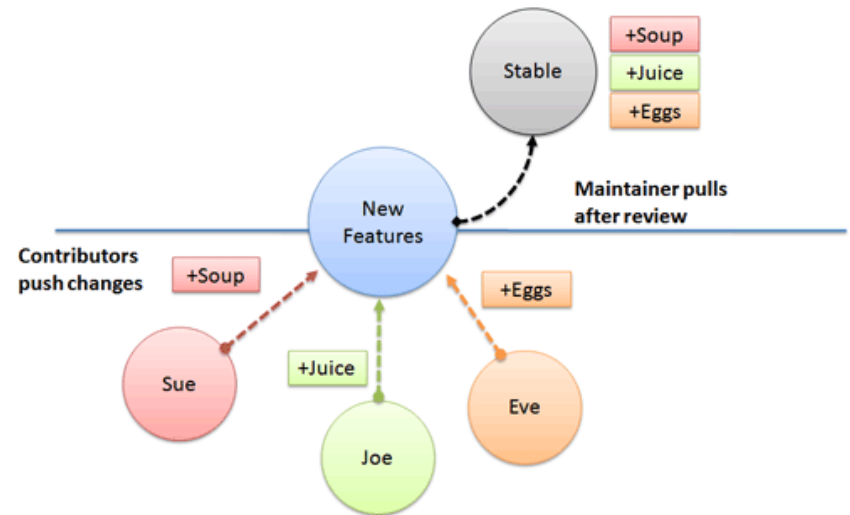
Distributed Source Control with Git

Git a popular program to use distributed source control

Operations:

- add
- commit
- push
- pull
- ...

Distributed Push/Pull Model



This Tutorial

- What is Source Control
- Distributed source control with Git
- **Git in Assignment 5**
- Using Gitlab
- Using git in Eclipse
- More about Assignment 5

Git in Assignment 5

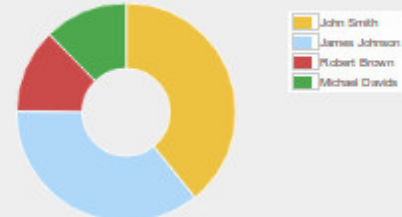
- Teams of 2
 - by Friday, 7 March: **form teams**
 - if you haven't formed team by then you will be assigned a partner
- Each team will use one git repository for the assignment (details to follow)
- Marking:
 - Correctness (common for both team members)
 - Performance (common for both team members)
 - **contribution to the git repository** (for each student)



The output has been generated by [gitinspector](https://code.google.com/p/gitinspector/): the statistical analysis tool for git repositories.

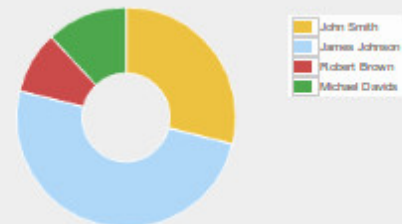
The following historical commit information, by author, was found in the repository.

Author	Commits	Insertions	Deletions	% of changes
John Smith	288	7721	4617	39.19
James Johnson	135	8910	2422	35.99
Robert Brown	71	2564	1352	12.44
Michael Davids	134	2943	954	12.38



Below are the number of rows from each author that have survived and are still intact in the current revision.

Author	Rows	% in comments
John Smith	3533	22.02
James Johnson	6113	52.15
Robert Brown	1123	21.19
Michael Davids	1464	20.15



The following history timeline has been gathered from the repository.

Author	2012W37	2012W38	2012W39	2012W40	2012W41	2012W42	2012W43
John Smith							.
James Johnson							
Robert Brown				.			.
Michael Davids							.
Modified Rows:	1522	3832	7553	6143	5833	5123	1477

The extensions below were found in the repository history (extensions used during statistical analysis are marked).

xml java pdf txt css

Git in Assignment 5

- Make sure you commit code only from your user name
- Code changes, amount of code contributed, type of changes, etc., would be used to calculate individual member's marks
- Not necessary for you to use advanced git features
- You will not loose or gain points for using or not using features of git

This Tutorial

- What is Source Control
- Distributed source control with Git
- Git in Assignment 5
- Using Gitlab
- Using git in Eclipse
- More about Assignment 5

Using Gitlab with eclipse

1. First time (one user does this)

- Create a remote repo in gitlab
- Create a local repo
- Add files to local repo
- Commit to local
- Push (local → global)

2. Every other time (same user, on the same computer)

- Pull global to local
- Change files on disk
- Add more files to local repo
- Commit (local)
- Push (local → global)

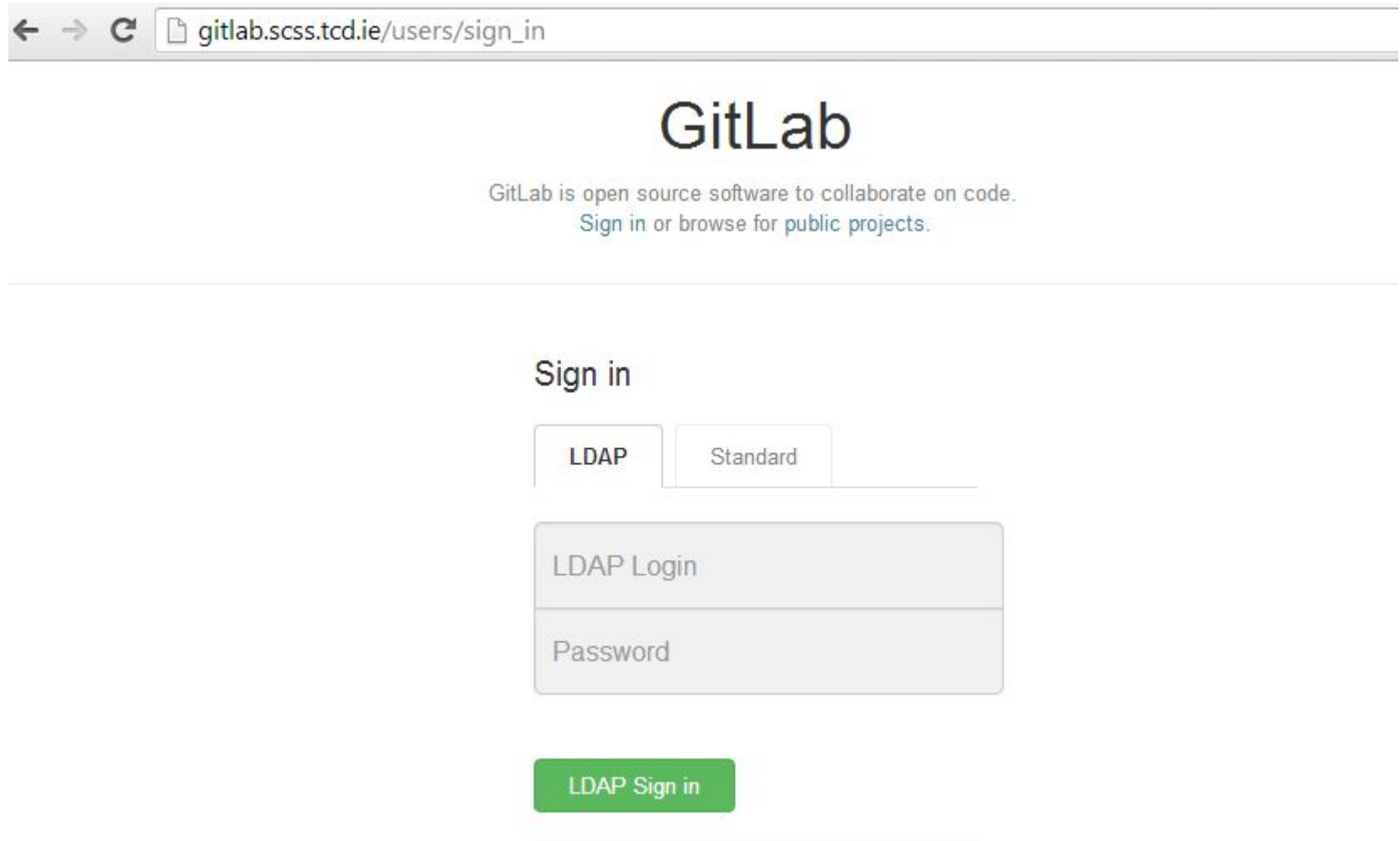
3. Every other time (every user, on different computer)

- Clone the remote repository
- Change / add files
- Commit
- Push

First time (one user does this)

- Create a remote repo in git lab
- Create a local repo
- Add files to local repo
- Commit to local
- Push (local → globa)

1. Goto www.gitlab.scss.tcd.ie
2. Log in using scss username / password



The screenshot shows a web browser window with the address bar displaying `gitlab.scss.tcd.ie/users/sign_in`. The page features the GitLab logo and a description: "GitLab is open source software to collaborate on code. Sign in or browse for public projects." Below this, there is a "Sign in" section with two tabs: "LDAP" (selected) and "Standard". Under the "LDAP" tab, there are two input fields: "LDAP Login" and "Password". At the bottom of this section is a green button labeled "LDAP Sign in".

← → ↻ `gitlab.scss.tcd.ie/users/sign_in`

GitLab

GitLab is open source software to collaborate on code.
[Sign in](#) or browse for [public projects](#).

Sign in

LDAP

Standard

LDAP Login


Password








LDAP Sign in

3. In the dash board click on the '+' symbol




4. Specify a project name
5. Select visibility level to be 'Private'
6. Click on 'Create Project'

 New Project




Project name


[Customize repository name?](#)


 [Import existing repository?](#)

Description
(optional)

Visibility Level
(?)

☒  **Private**
Project access must be granted explicitly for each user.

☐  **Internal**
The project can be cloned by any logged in user.


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

Create project

Need a group for several dependent projects? [Create a group](#)

7. Click on 'http' button

8. Make a note of the url highlighted in yellow



Profile

Search

Issues 0 Merge Requests 0 Wiki Settings

Servesh Muralidharan / Test

SSH HTTP <http://gitlab.scsss.tcd.ie/muralis/test.git>

[- Edit](#)

Git global setup:

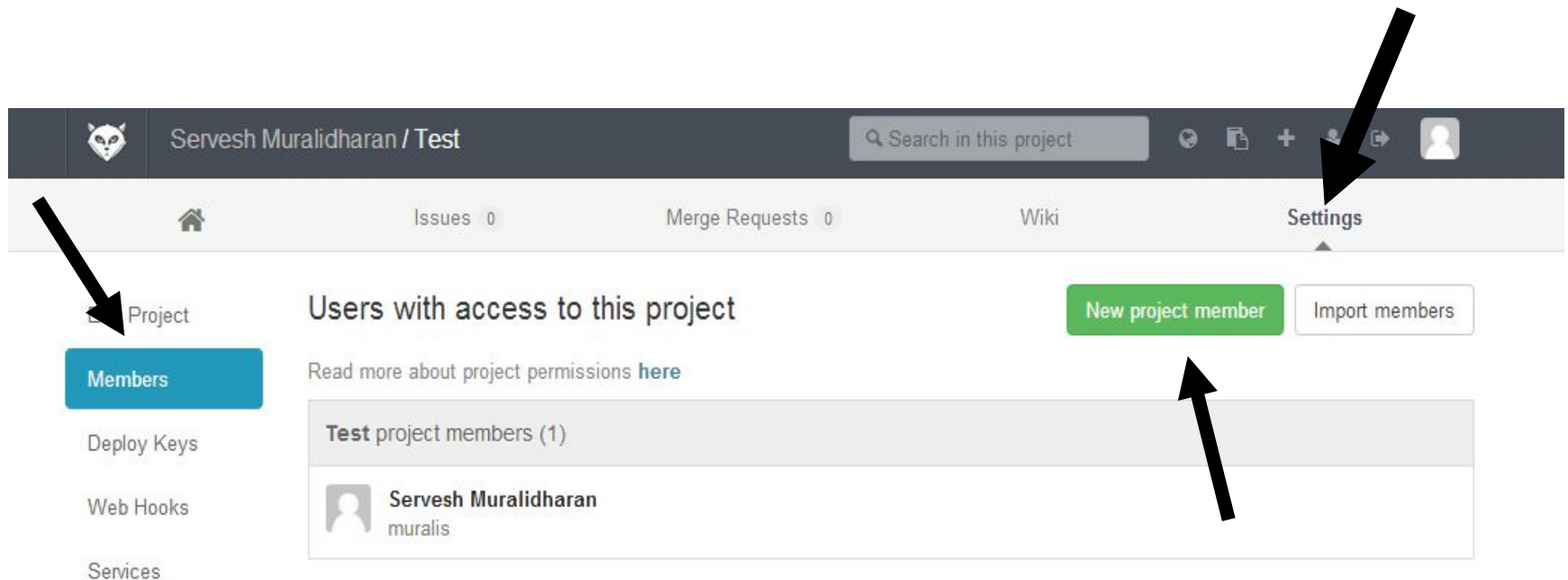
```
git config --global user.name "Servesh Muralidharan"
git config --global user.email "muralis@scsss.tcd.ie"
```

Create Repository

```
mkdir test
cd test
git init
touch README
```

9. Select 'settings' tab on the top, followed 'members' on the left

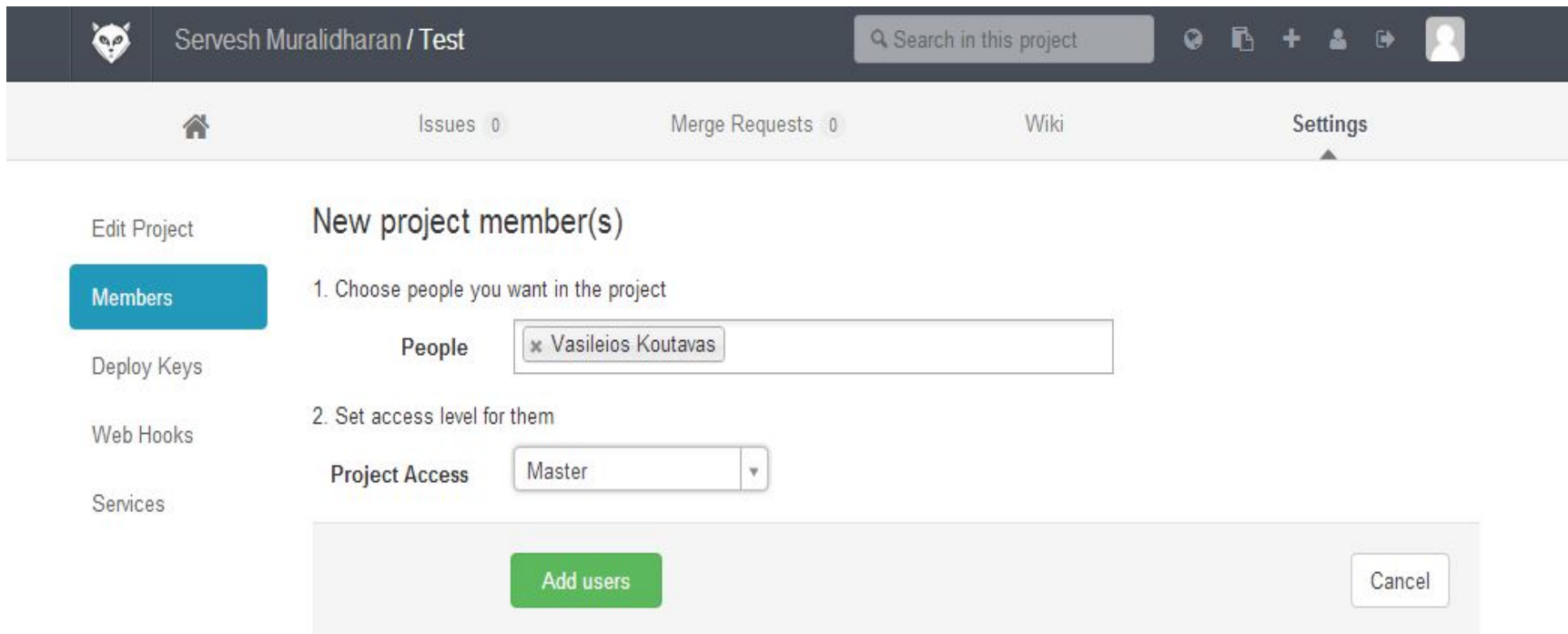
10. Select New project member



11. Search for your team mates name in the 'people field'

12. All users with scss account should be visible

13. Select Project Access level to what you desire, ex: Team Leader – Master and the rest as Developer.



The screenshot shows the GitHub 'New project member(s)' interface. At the top, there is a dark header bar with the GitHub logo, the user 'Servesh Muralidharan / Test', a search bar labeled 'Search in this project', and several icons. Below this is a light gray navigation bar with links for Home, Issues (0), Merge Requests (0), Wiki, and Settings. On the left side, there is a sidebar with links for 'Edit Project', 'Members' (highlighted in blue), 'Deploy Keys', 'Web Hooks', and 'Services'. The main content area is titled 'New project member(s)' and contains two steps: 1. 'Choose people you want in the project' with a 'People' input field containing 'x Vasileios Koutavas', and 2. 'Set access level for them' with a 'Project Access' dropdown menu set to 'Master'. At the bottom, there are two buttons: 'Add users' (green) and 'Cancel' (white).

Servesh Muralidharan / Test

Search in this project

Issues 0 Merge Requests 0 Wiki Settings

Edit Project

Members

Deploy Keys

Web Hooks

Services

New project member(s)

1. Choose people you want in the project

People

2. Set access level for them

Project Access

Add users Cancel

14. After adding all of your team mates, add the following members of CS2012 staff as Reporters.

(1) Vasileios Koutavas

(2) Servesh Muralidharan




(3) Aravind Vasudevan

(4) Shixiong Xu

(5) Yu Xu

Important : To be able to access and mark your project

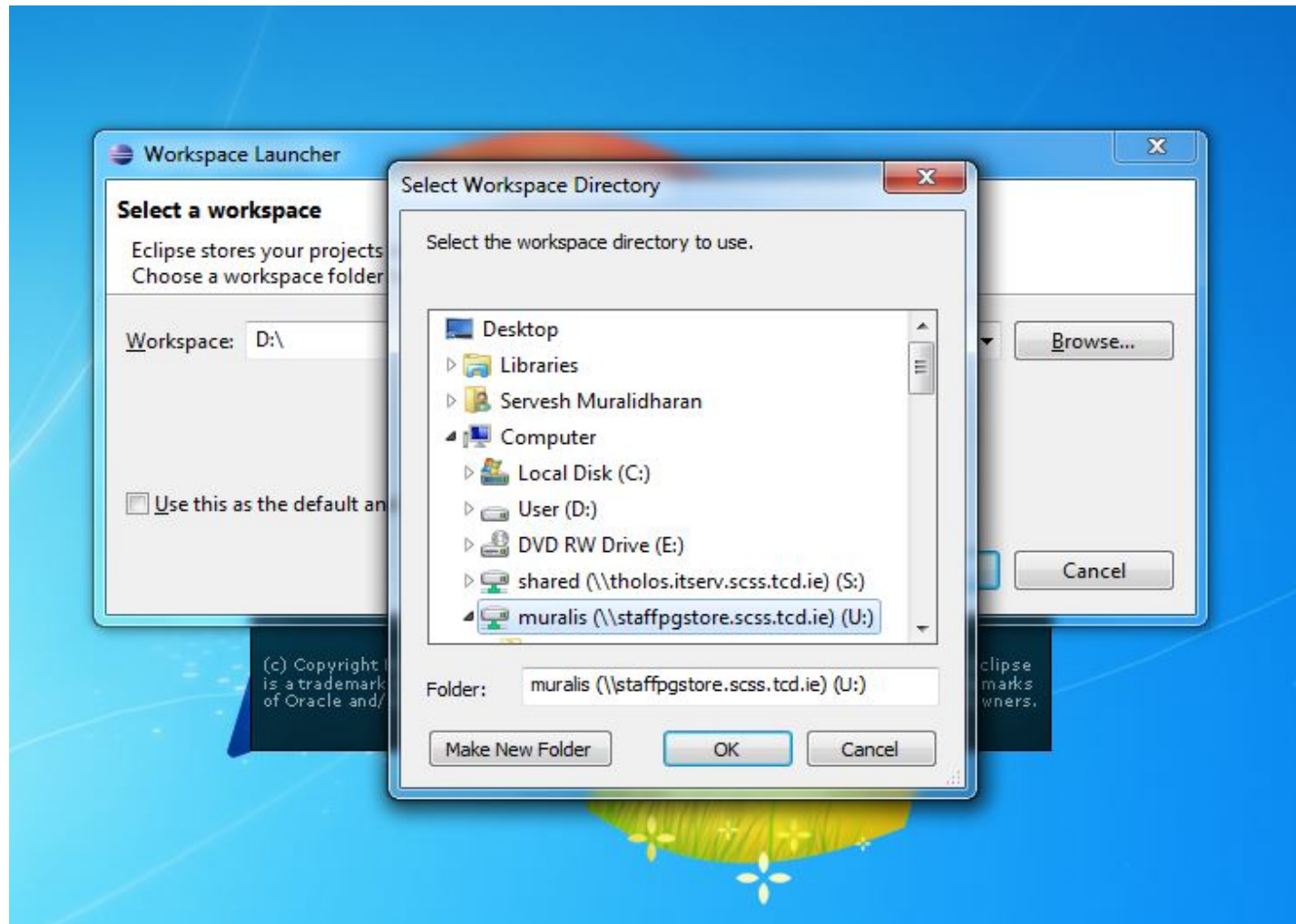
The screenshot shows the GitHub interface for a project named 'Test' by user 'Servesh Muralidharan'. The top navigation bar includes a search bar and icons for repository management. Below the bar, a secondary navigation bar shows 'Issues 0', 'Merge Requests 0', 'Wiki', and 'Settings'. The left sidebar contains links for 'Edit Project', 'Members' (highlighted), 'Deploy Keys', 'Web Hooks', and 'Services'. The main content area is titled 'Users with access to this project' and includes buttons for 'New project member' and 'Import members'. A link 'Read more about project permissions here' is also present. Below this, a section titled 'Test project members (2)' lists two users: 'Servesh Muralidharan' (username: muralis) and 'Vasileios Koutavas' (username: vasileios.koutavas). The user 'Vasileios Koutavas' is shown with a 'Master' role and a red minus button to remove them.

Test project members (2)	
	Servesh Muralidharan muralis
	Vasileios Koutavas vasileios.koutavas Master 

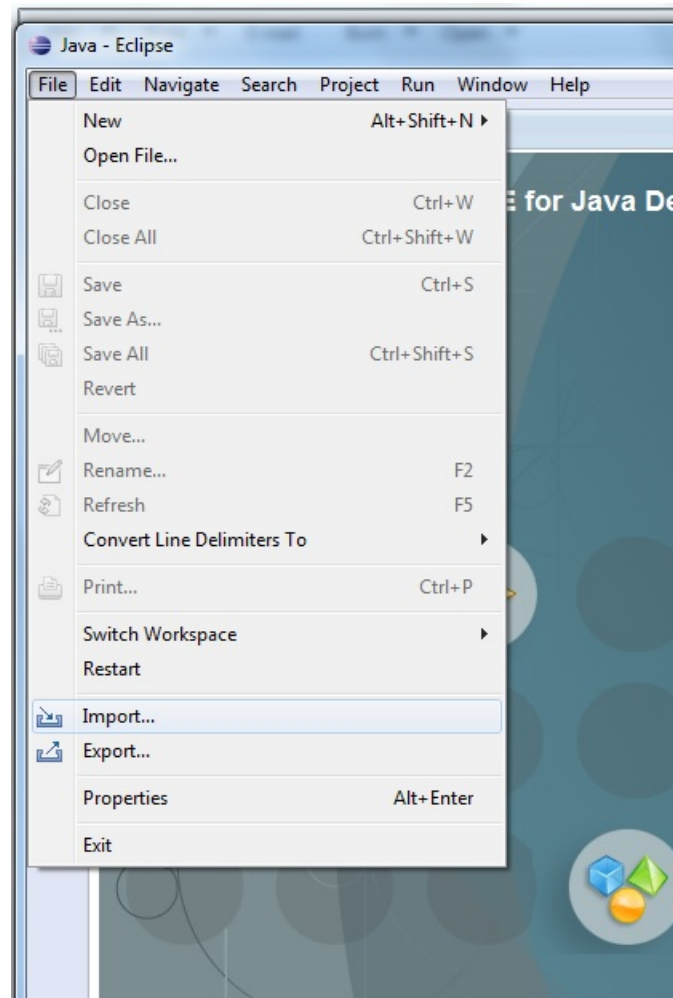
First time (one user does this)

- Create a remote repo in git lab
- Create a local repo
- Add files to local repo
- Commit to local
- Push (local → globa)

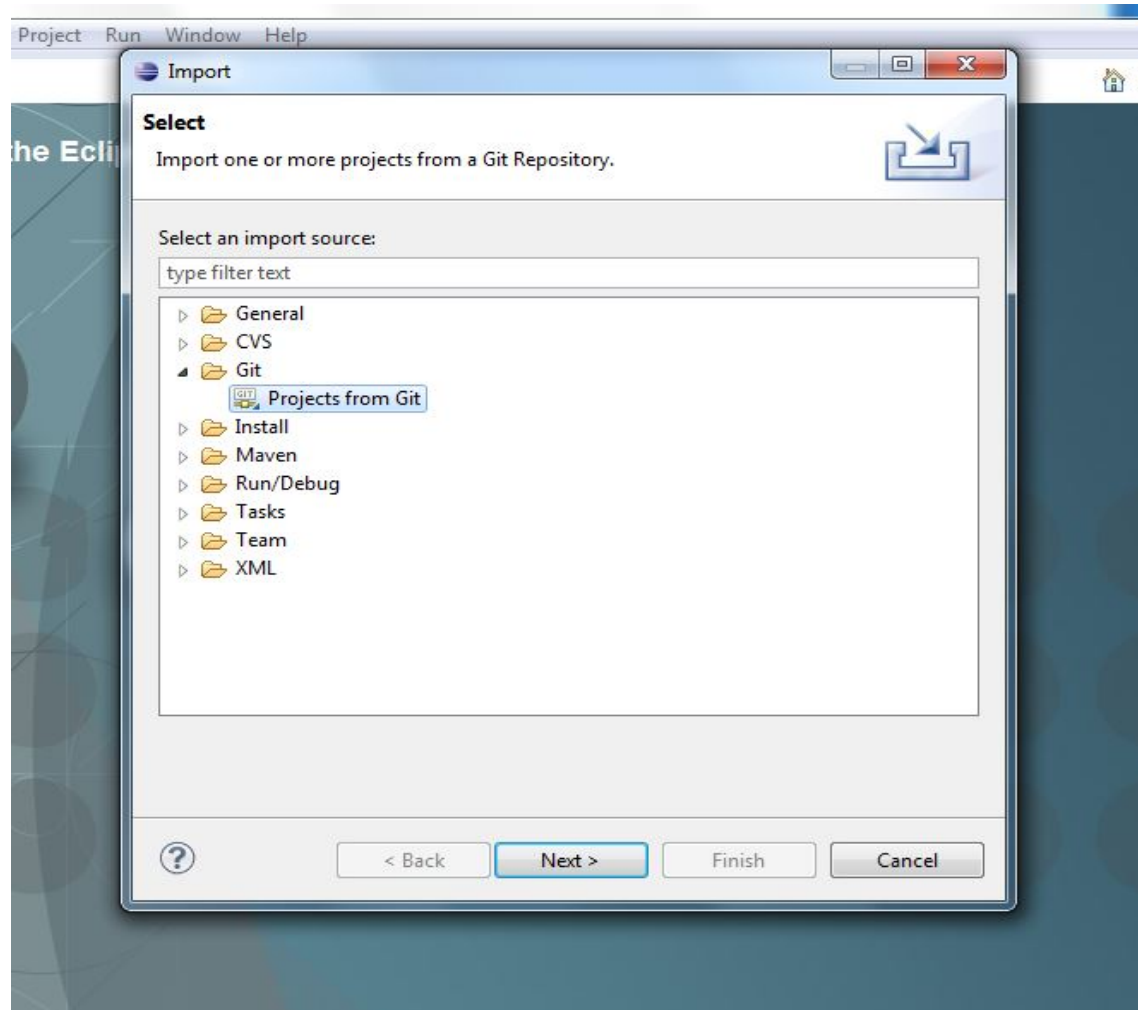
1. You can choose your workspace directory in 'U' Drive, if you don't want to keep setting up eclipse on every system or alternatively you can use your personal machine



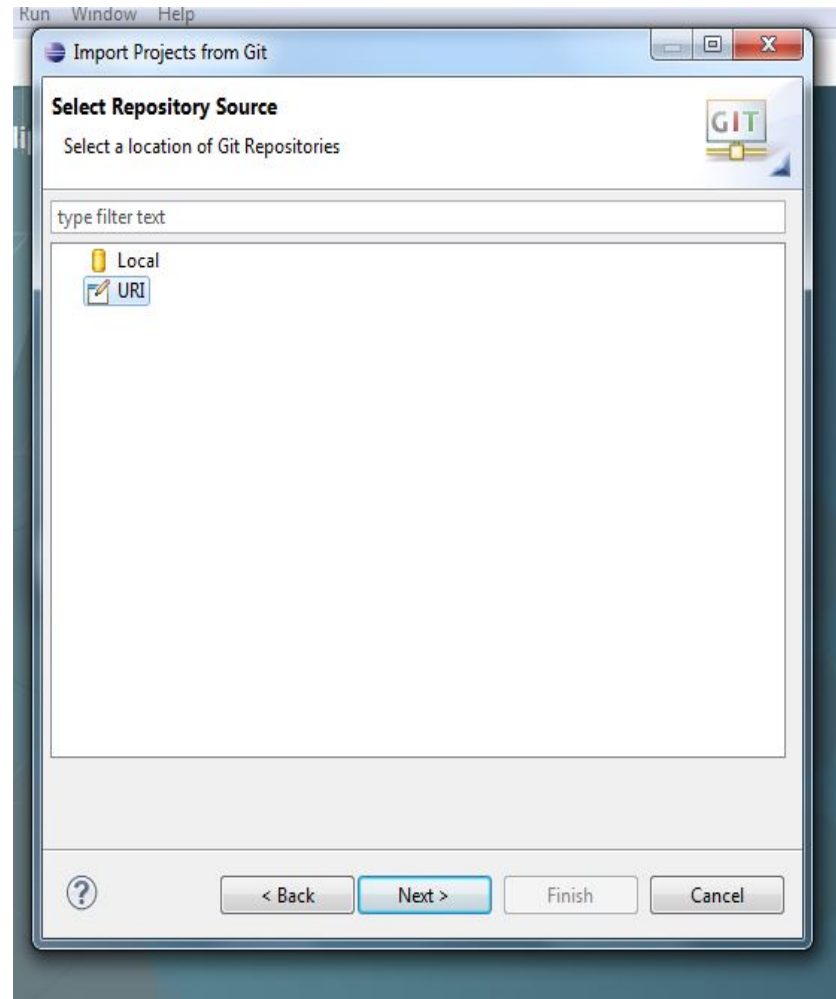
2. Choose import wizard from file menu.



3. Select 'Git' followed by 'Projects from Git'



4. Choose the 'URI' option



5. Provide the link you saved from the gitlab
6. Provide your scss username and password.

Import Projects from Git

Source Git Repository
Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

Store in Secure Store ☒

Profile

Search

Issues 0 Merge Requests 0 Wiki Settings

Serves Muralidharan / Test

[- Edit](#)

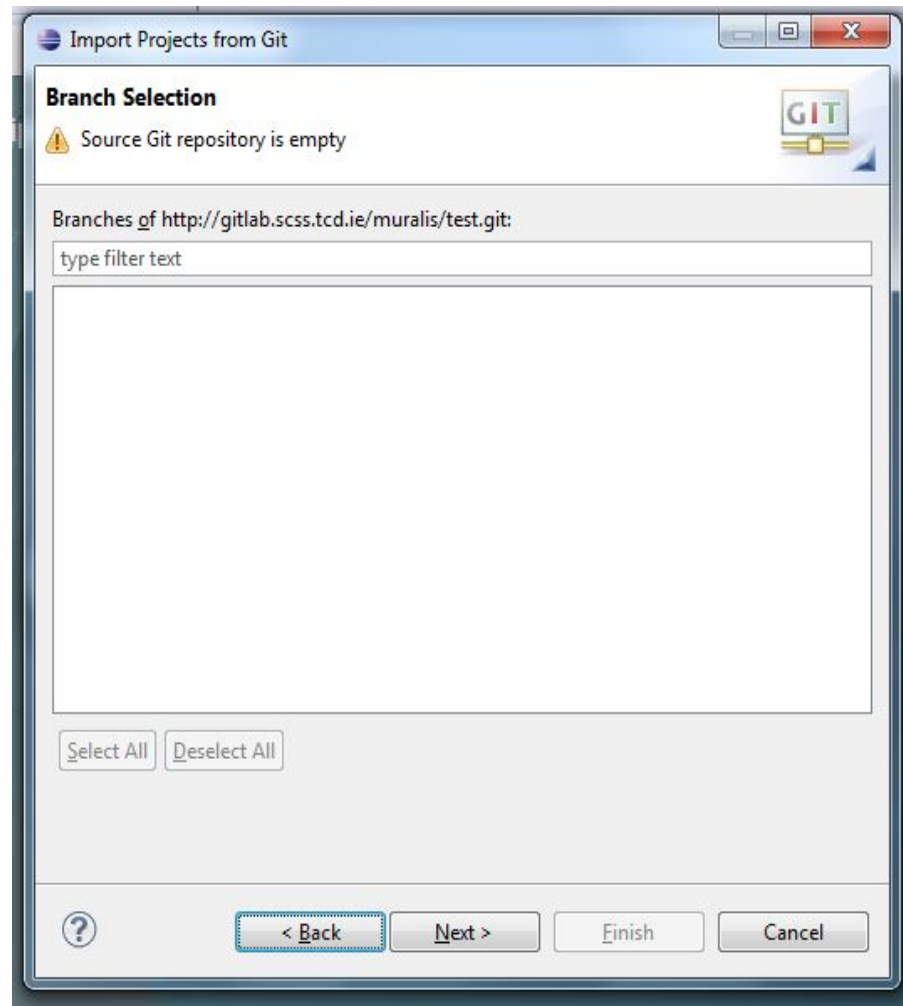
Git global setup:

```
git config --global user.name "Serves Muralidharan"
git config --global user.email "muralis@scss.tcd.ie"
```

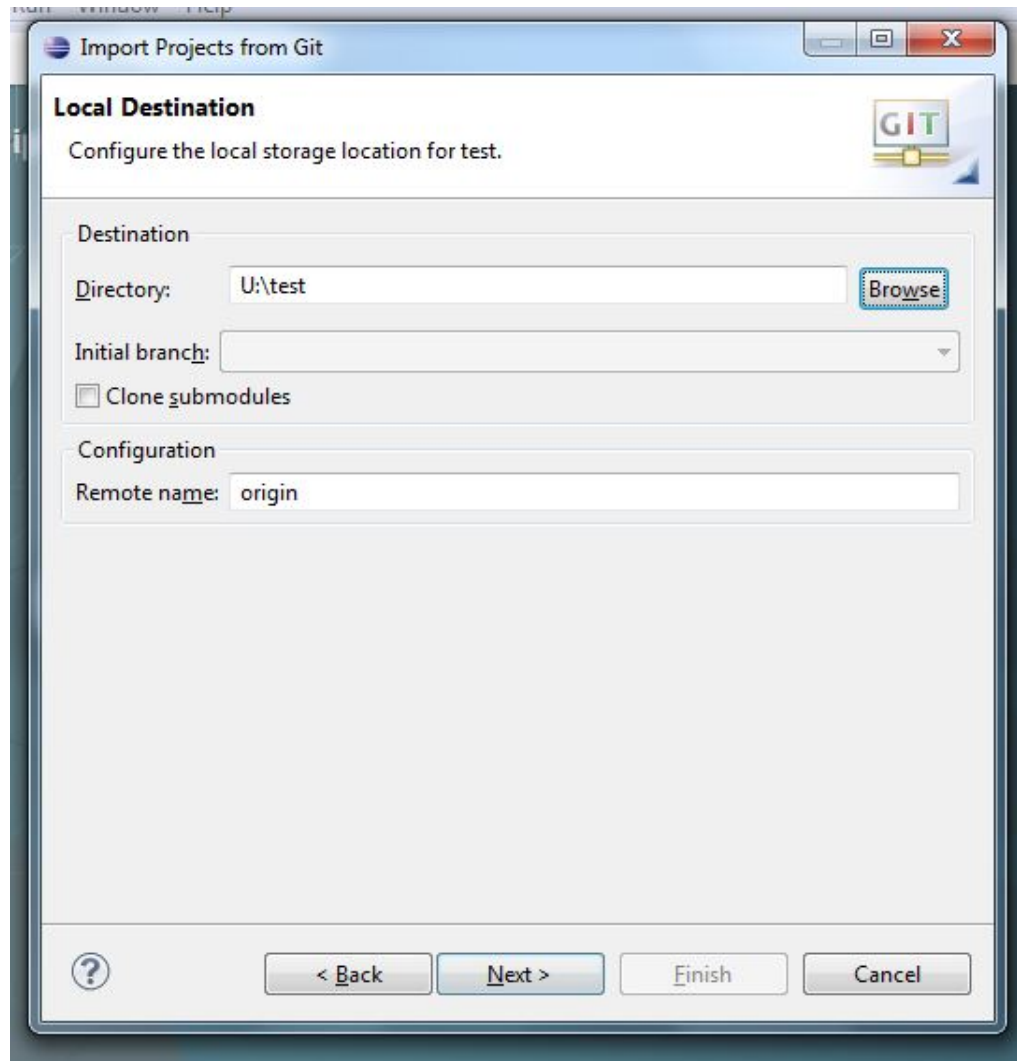
Create Repository

```
mkdir test
cd test
git init
touch README
```

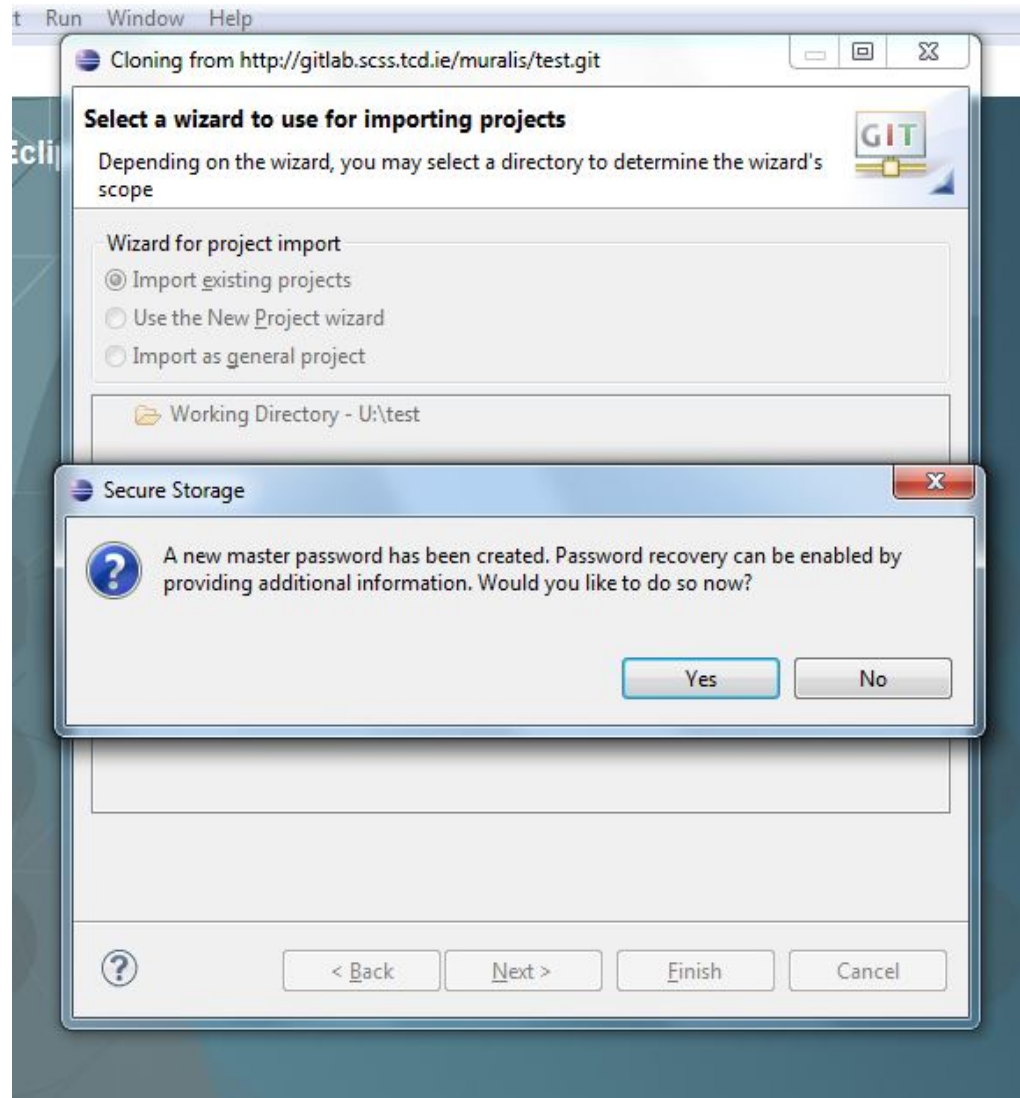
7. Since this is an empty repository eclipse will warn you about it, select next.



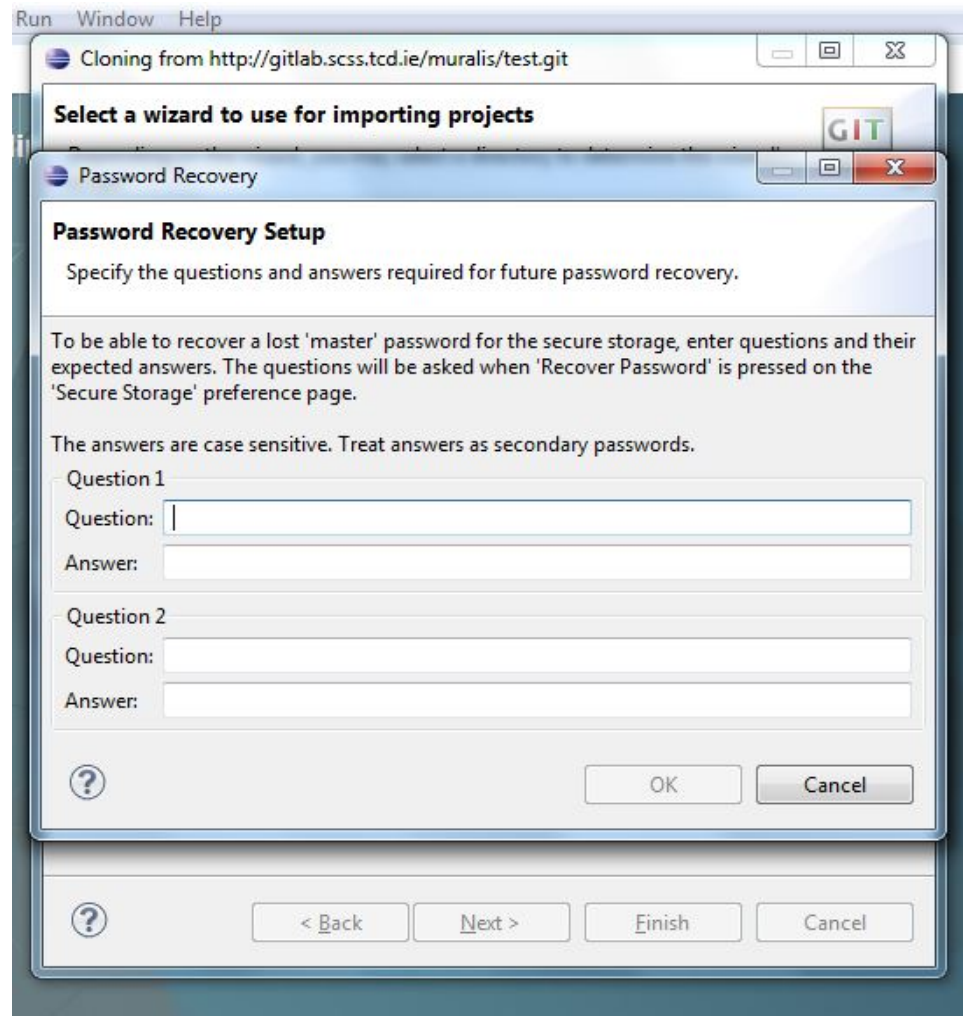
8. Select a directory where you want to create the local repository



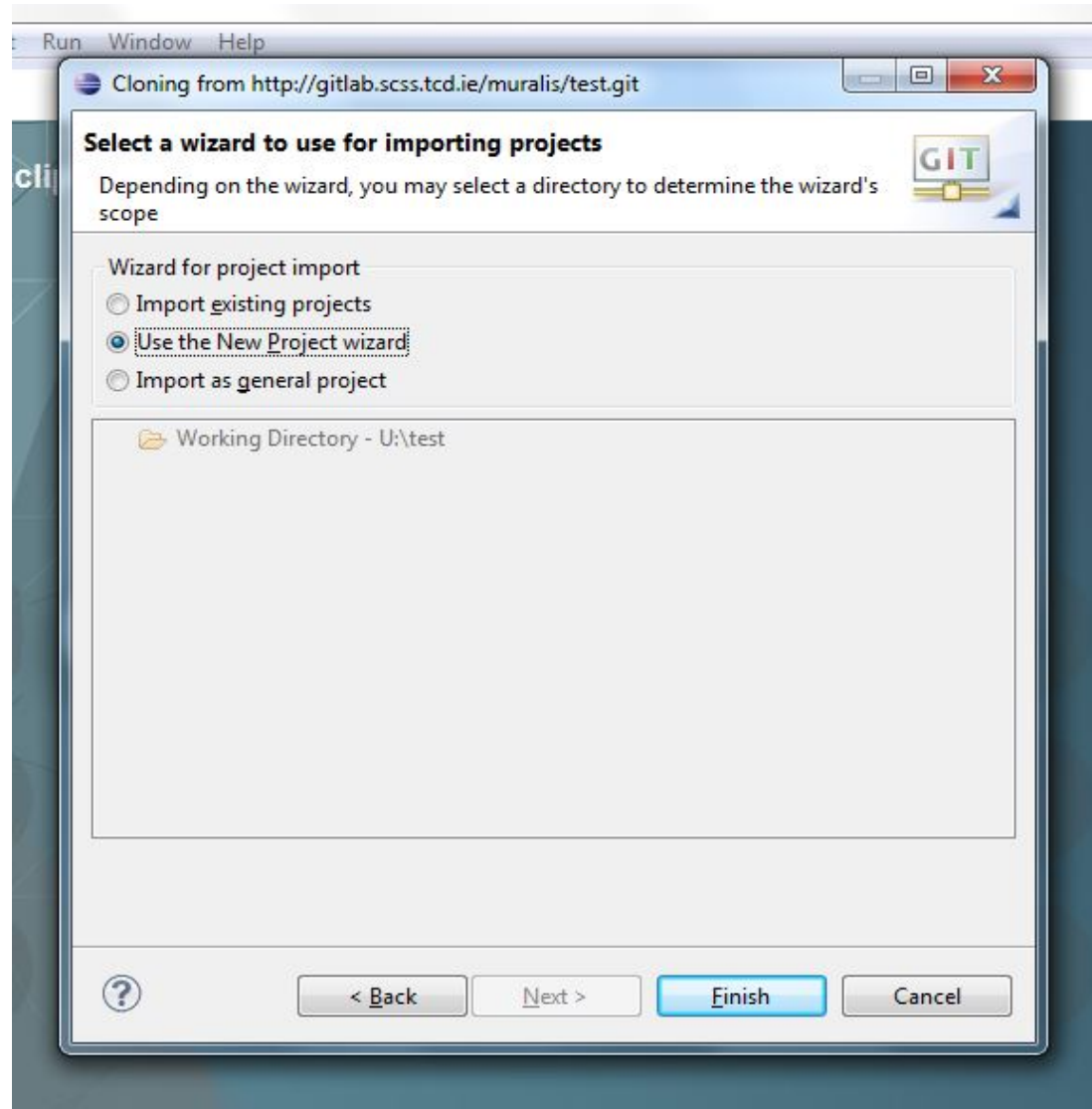
9. The master password warning is not required to get rid of it click on 'yes' and select cancel on the next step



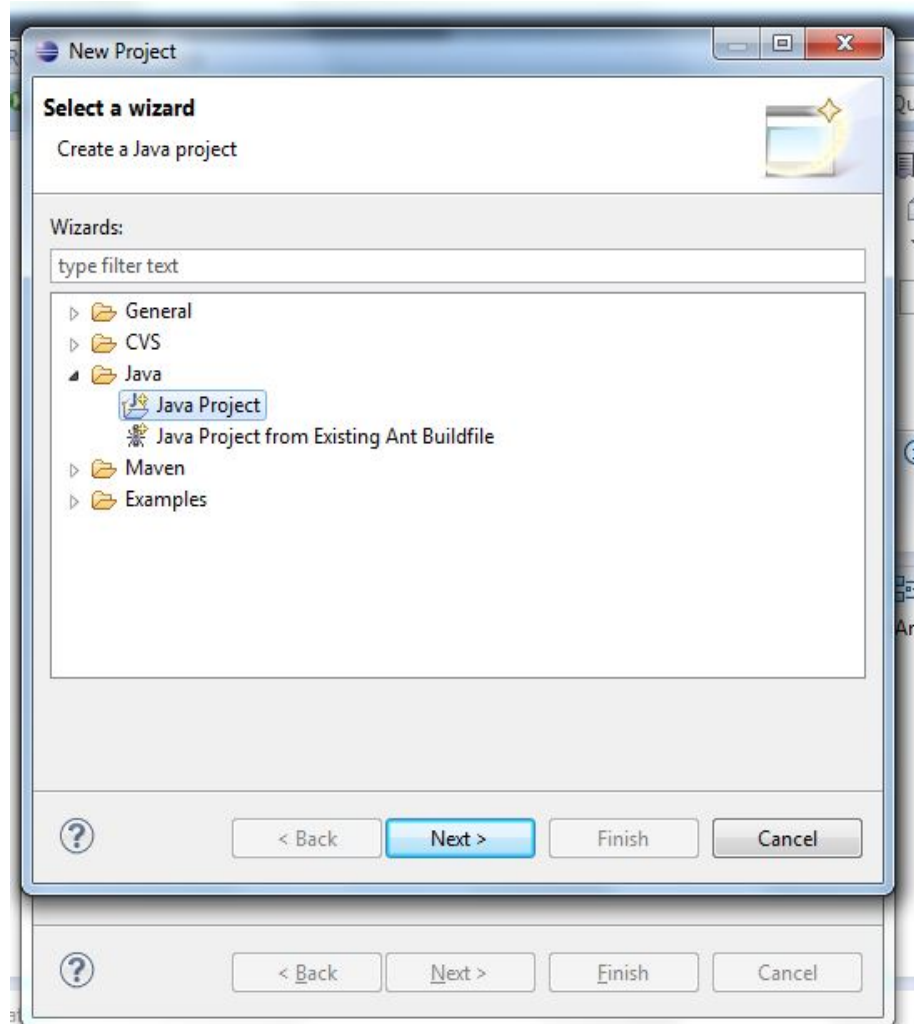
10. Click cancel to ignore master password creation



11. Since its an empty repository, you need to use the New project wizard

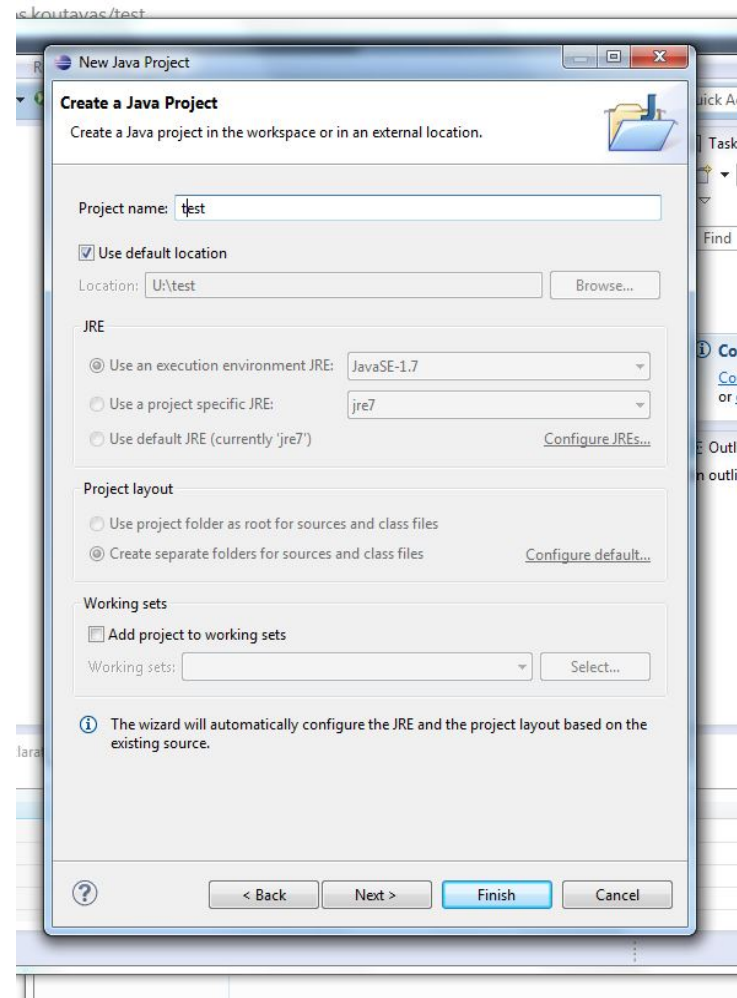


12. Select 'Java Project' as the type

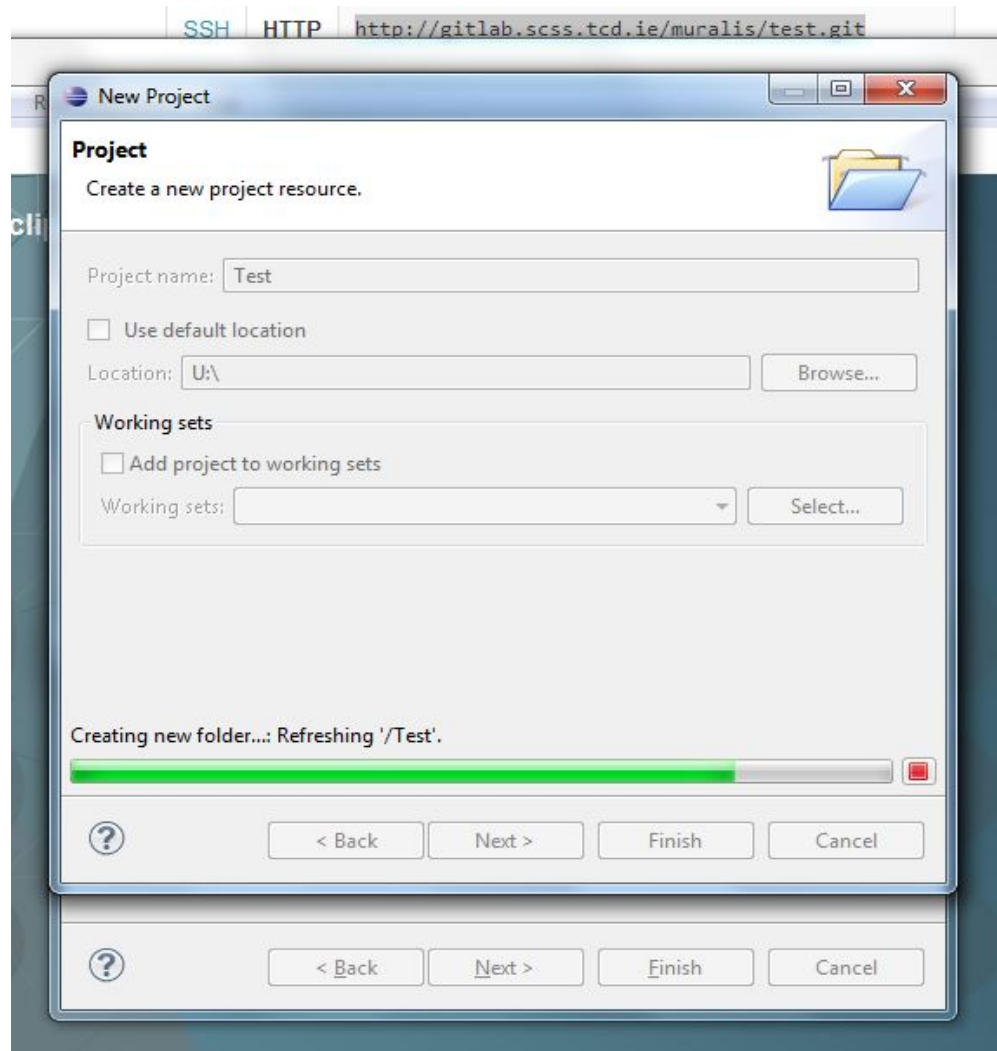


13. Provide project name, note it should be the same name that you used in the gitlab server

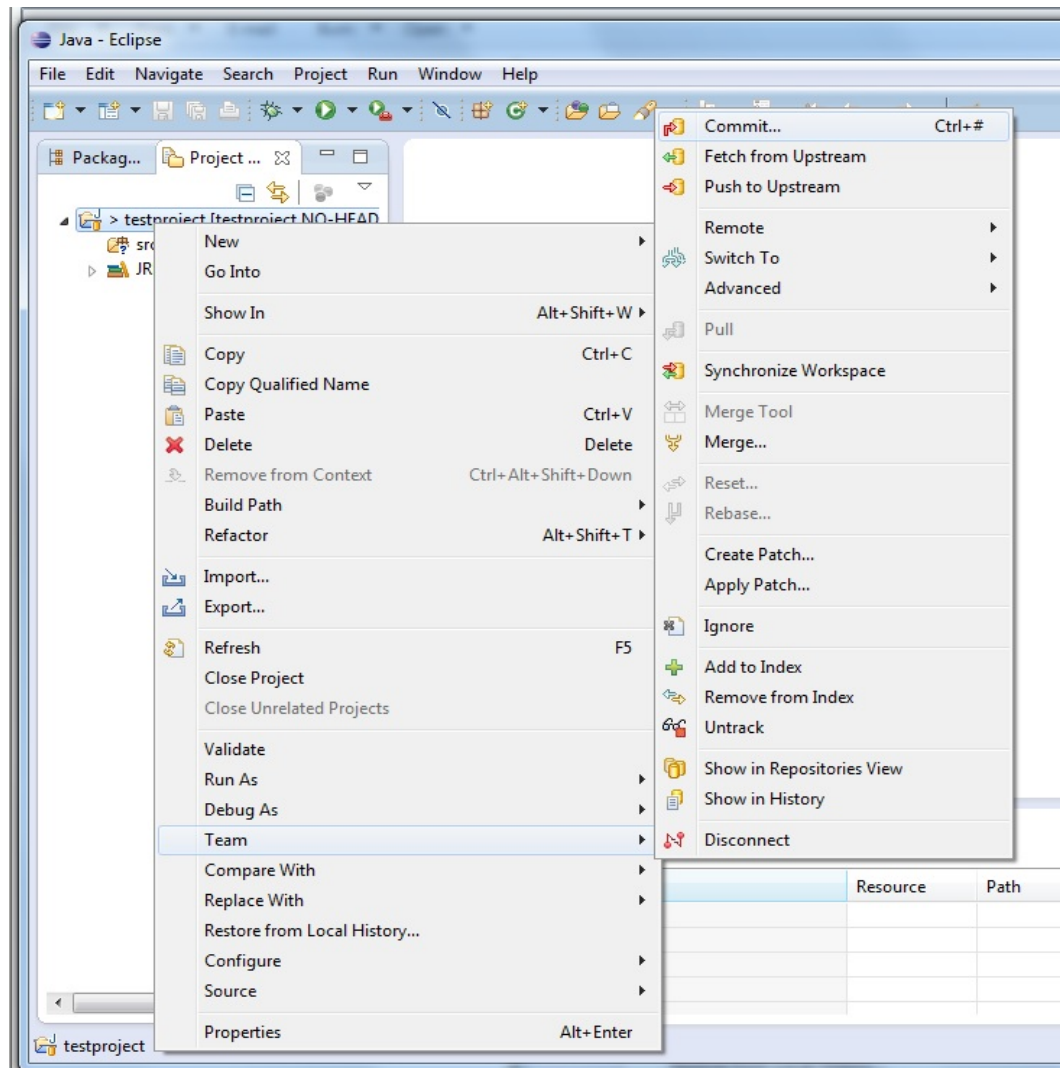
14. The rest of options leave it at default and click 'Finish'



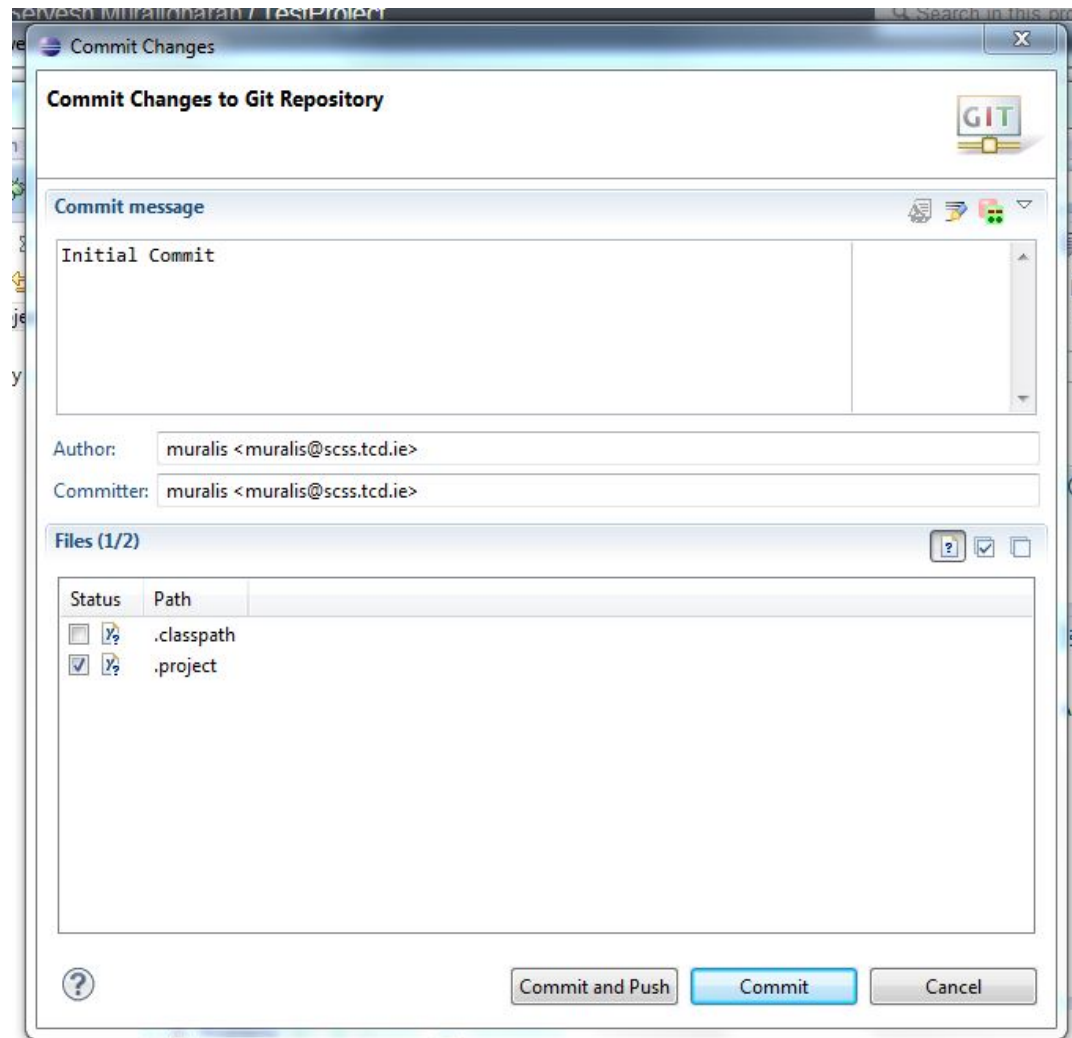
15. Eclipse will take a few sec and create the project



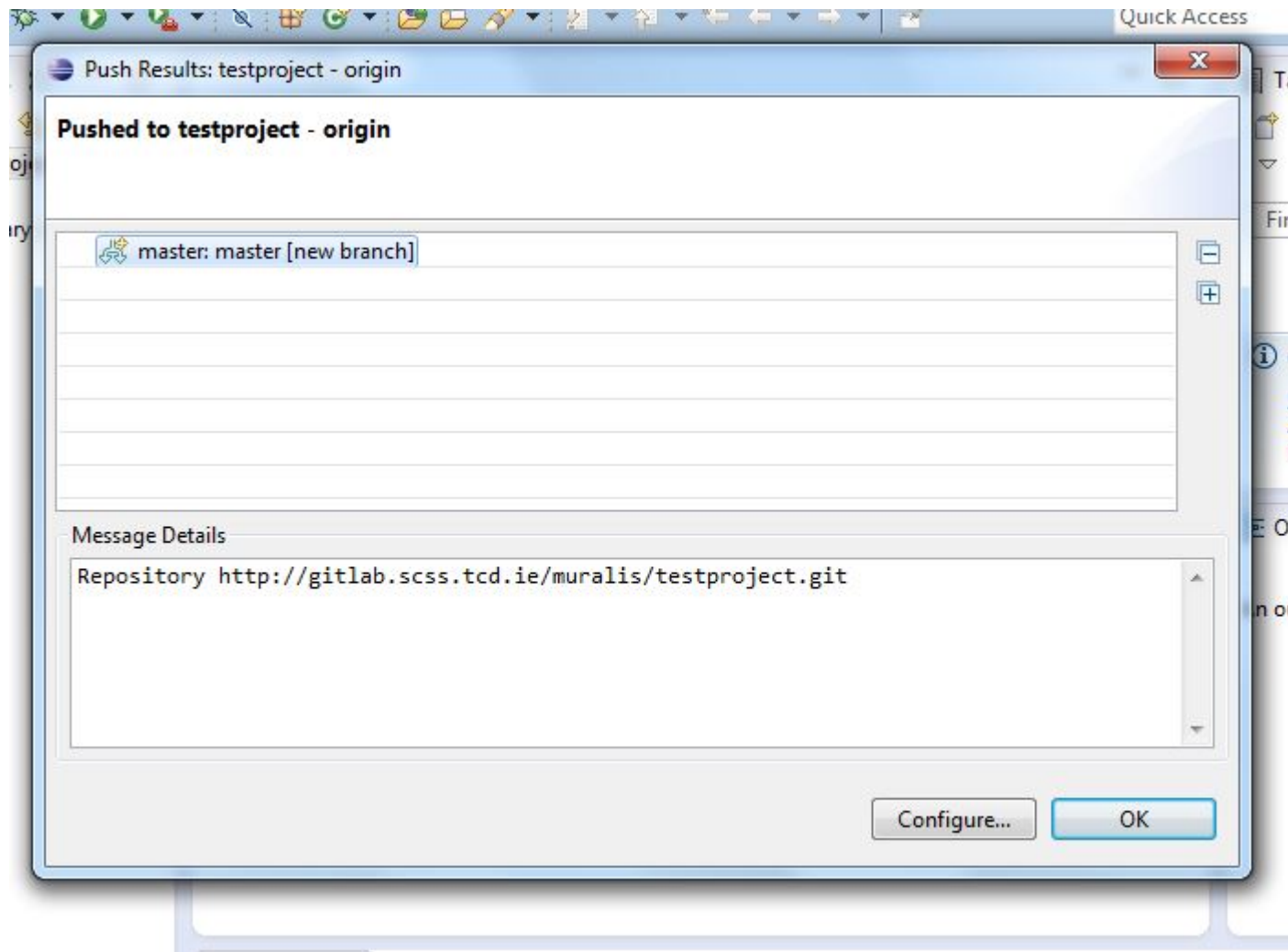
16. Right click on **project name** in project explorer window then select **Team** followed by **Commit**



17. Type in your commit message
18. Select '**All the files**' in your project
19. Click on **commit and push**



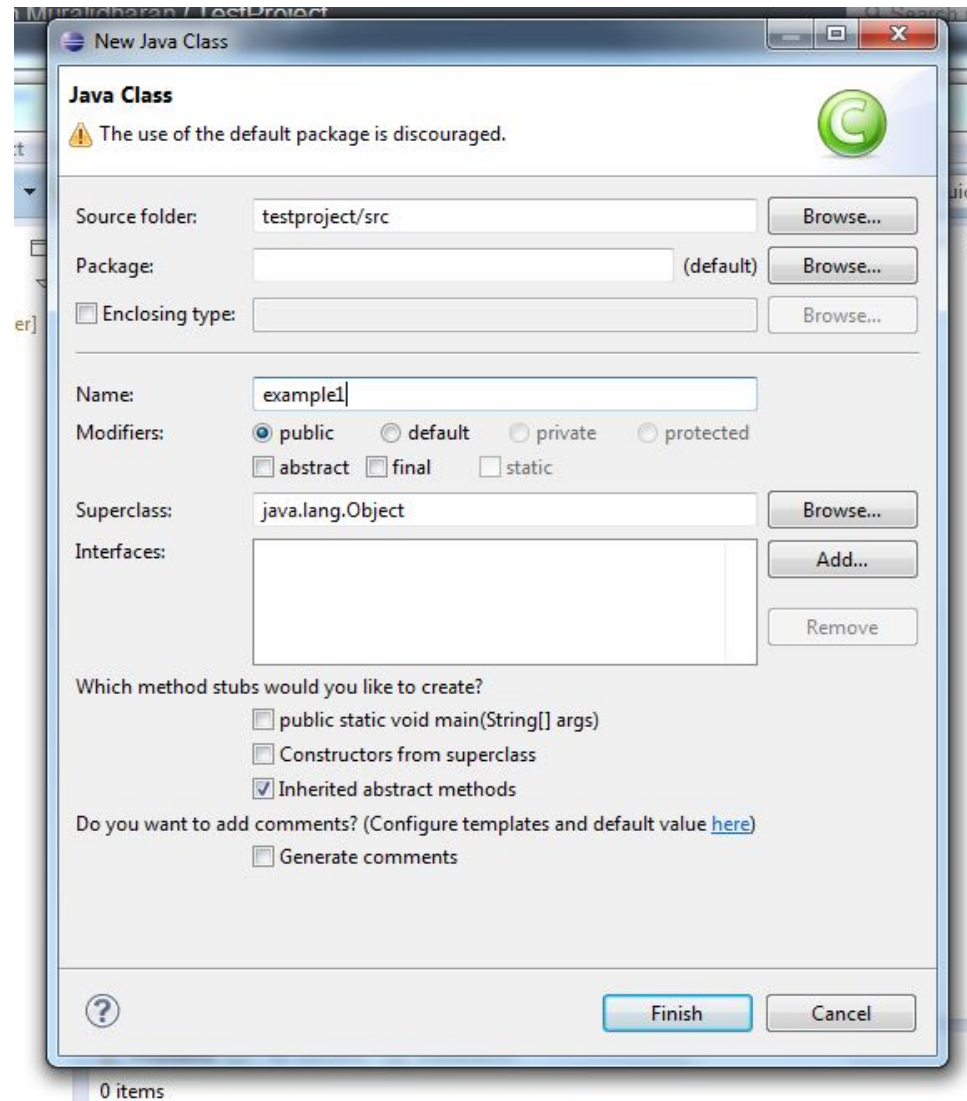
20. Eclipse displays the message showing the successful commit and the creation of master branch



First time (one user does this)

- Create a remote repo in git lab
- Create a local repo
- Add files to local repo
- Commit to local
- Push (local → globa)

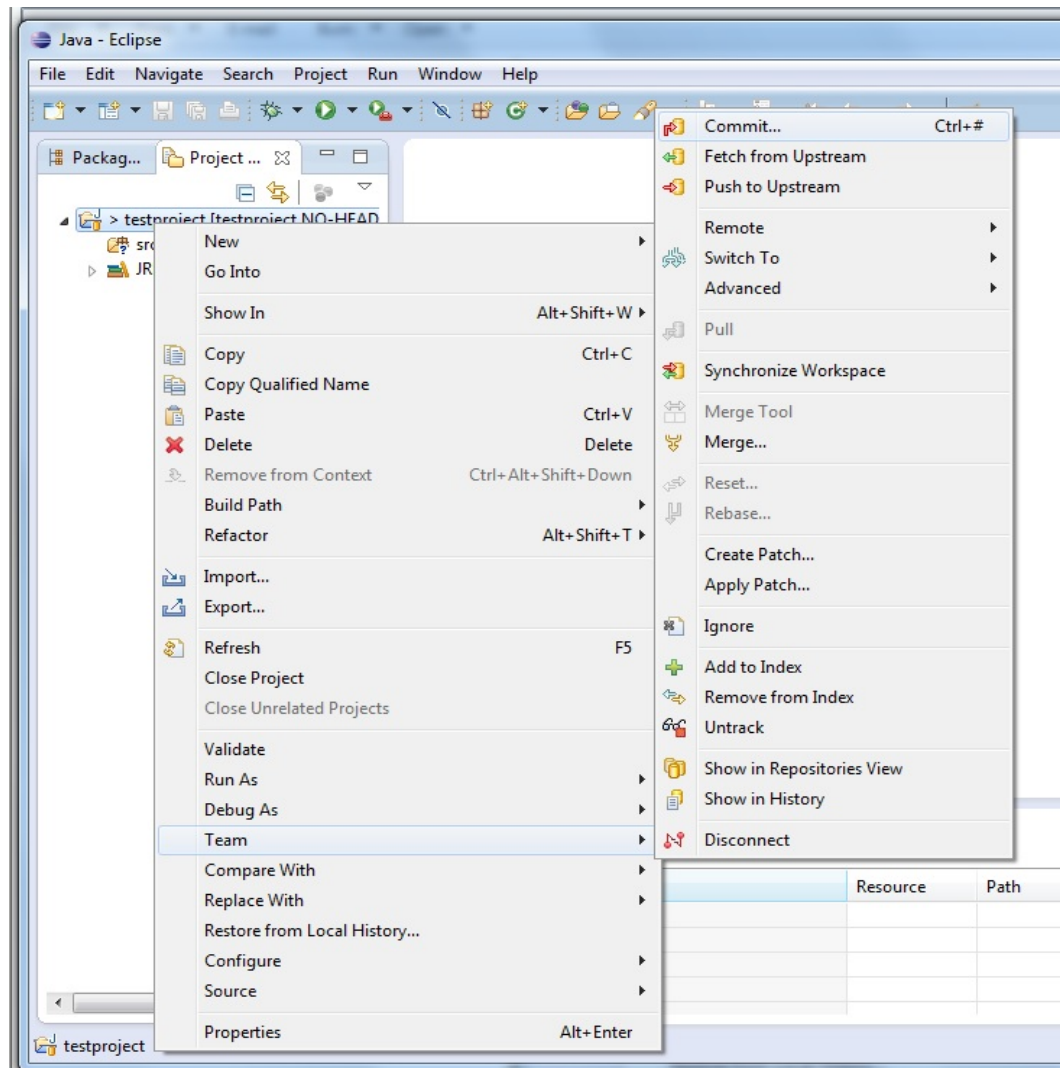
1. Add a file to the java project as usual



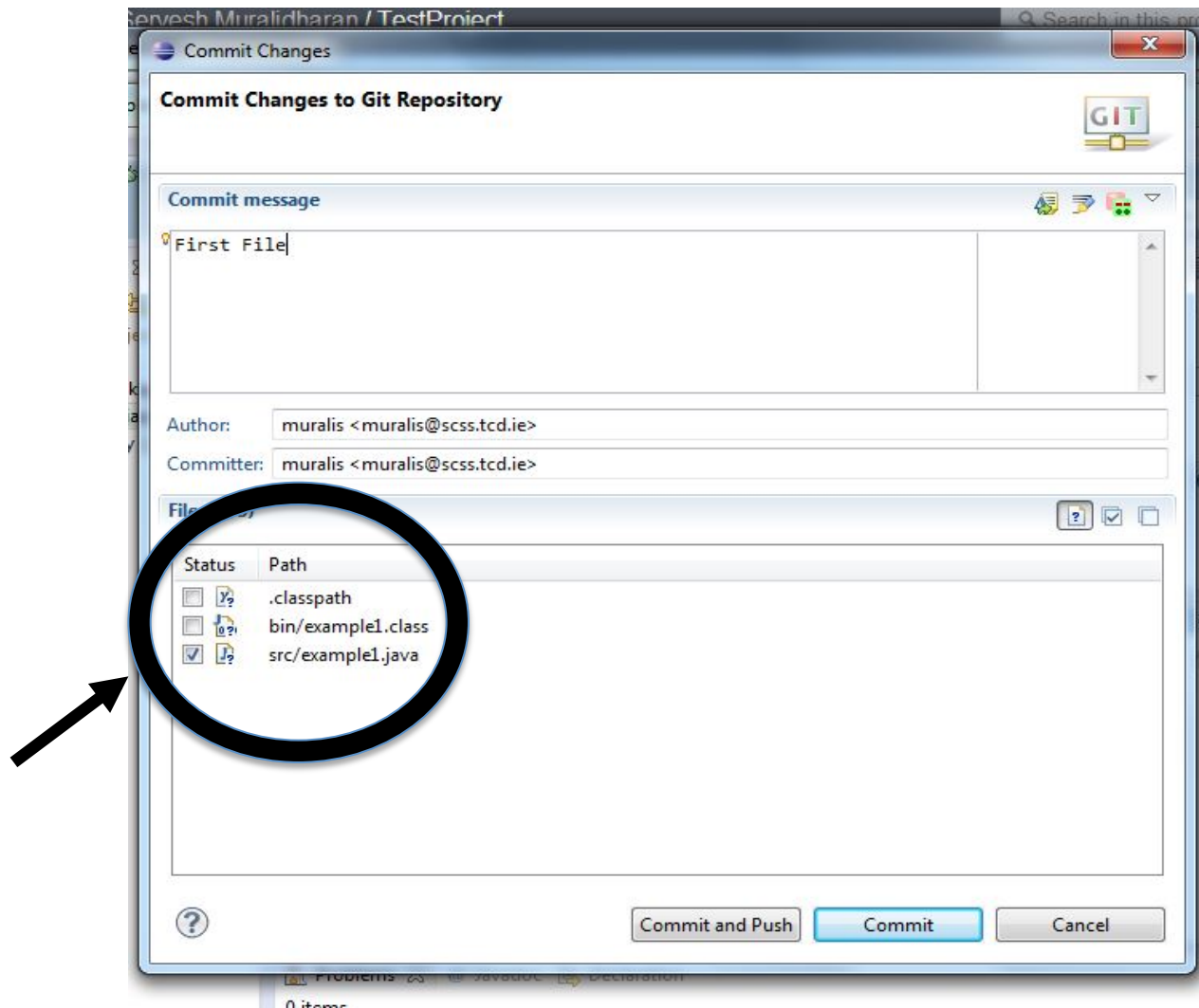
First time (one user does this)

- Create a remote repo in git lab
- Create a local repo
- Add files to local repo
- Commit to local
- Push (local → globa)

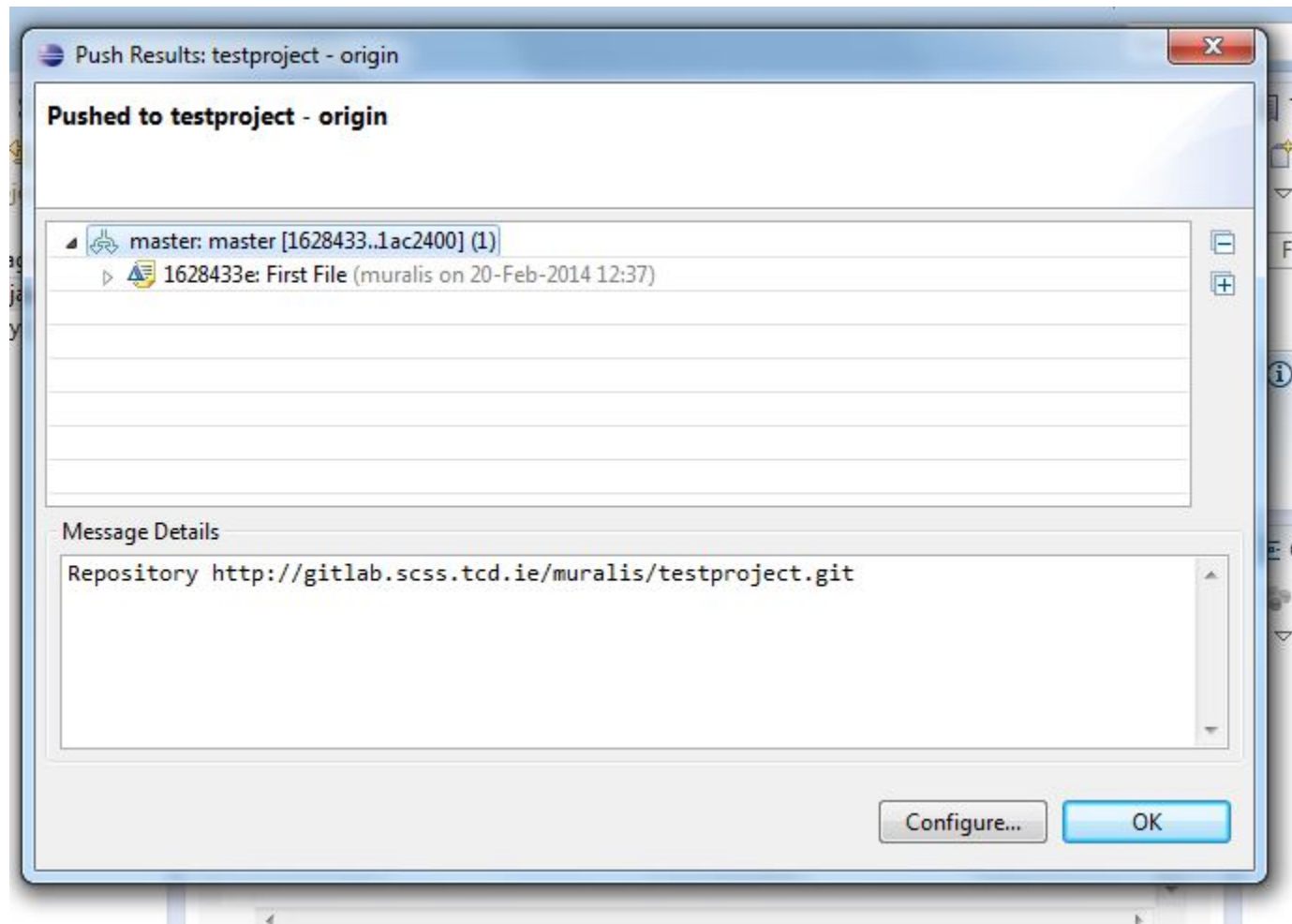
2. Right click on **project name** in project explorer window then select **Team** followed by **Commit**



3. Commit and push the changes, but make sure to **add** the newly created files



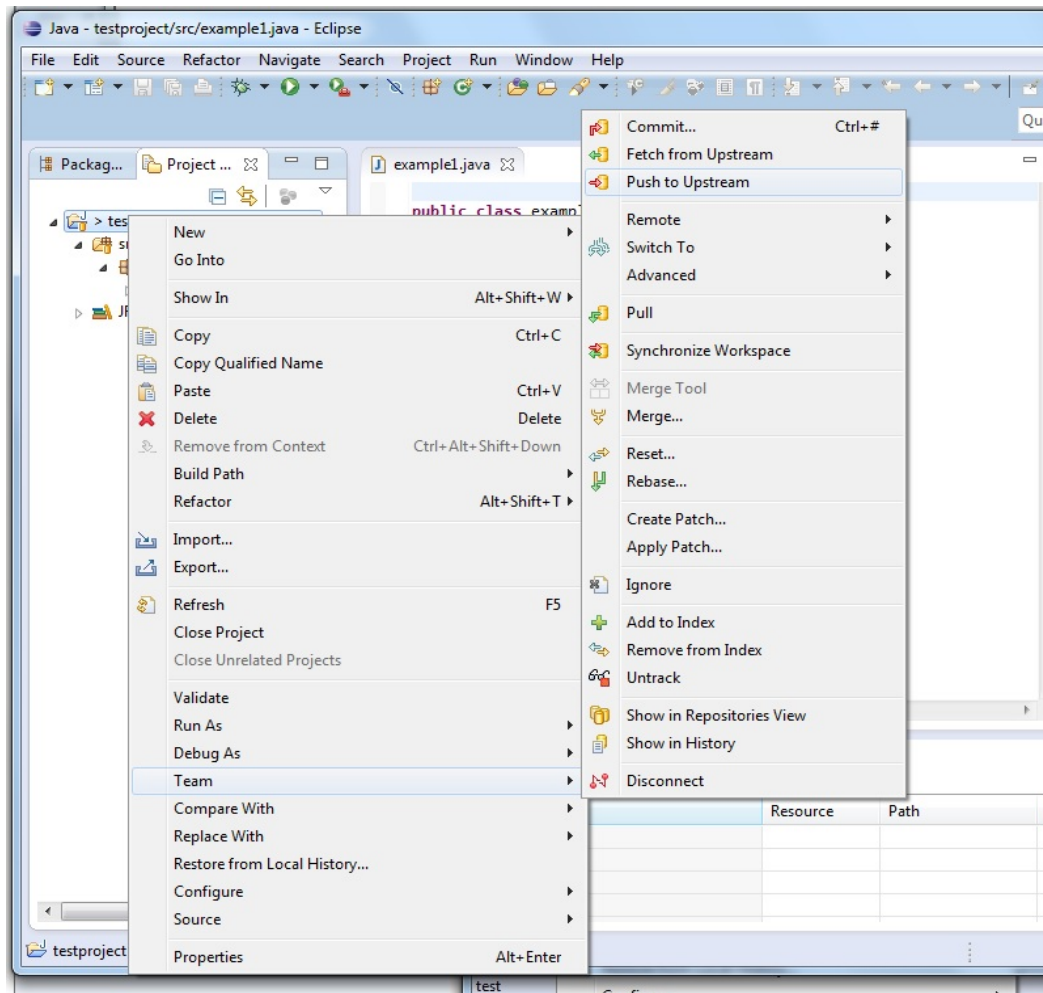
4. Notice a new node being created for the commit of the file



Every other time (same user, on the same computer)

- Pull global to local
- Change files on disk
- Add more files to local repo
- Commit (local)
- Push (local → global)

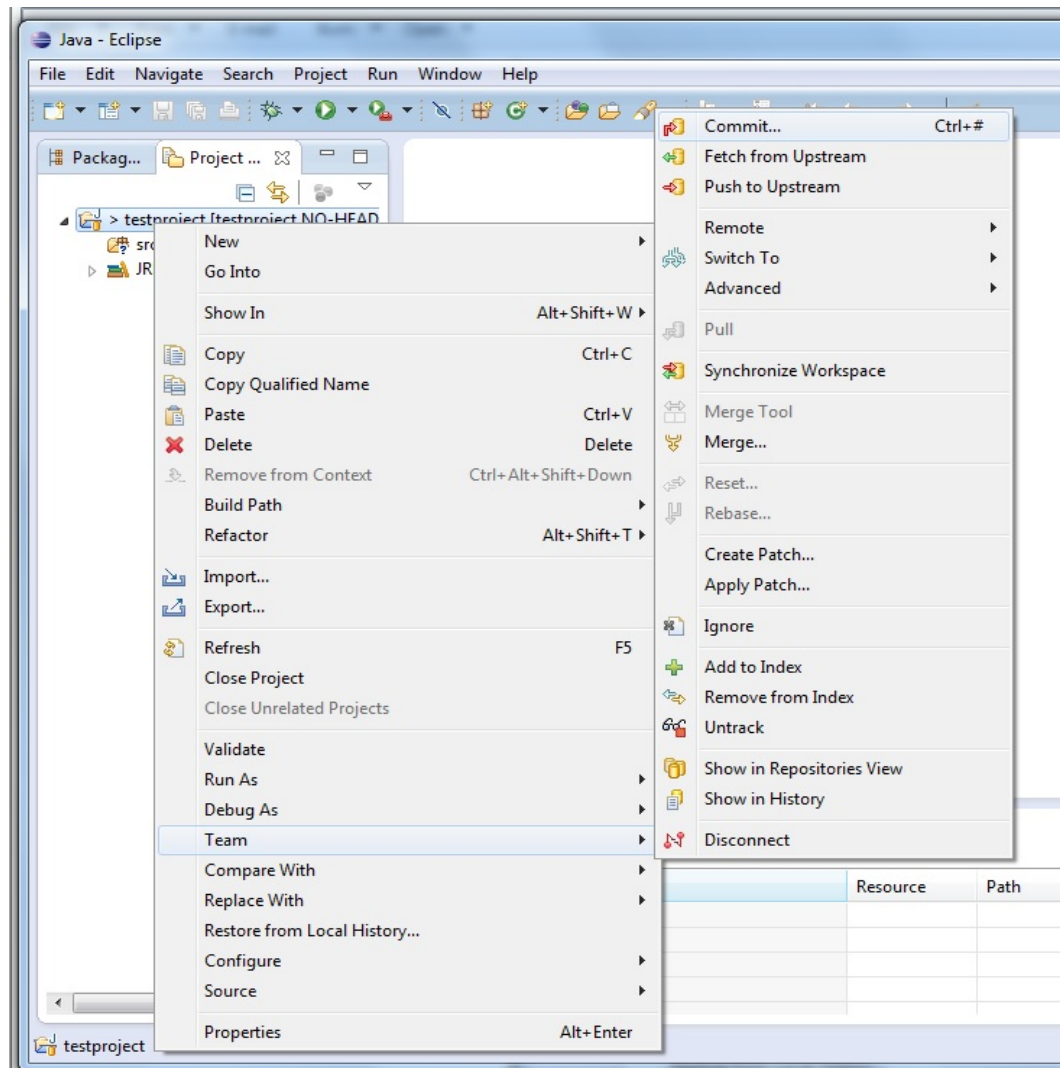
1. To fetch or push from the repository, select the **project name** followed by **Team** and then choose either '**Fetch from upstream**' or '**Pull**' for pull



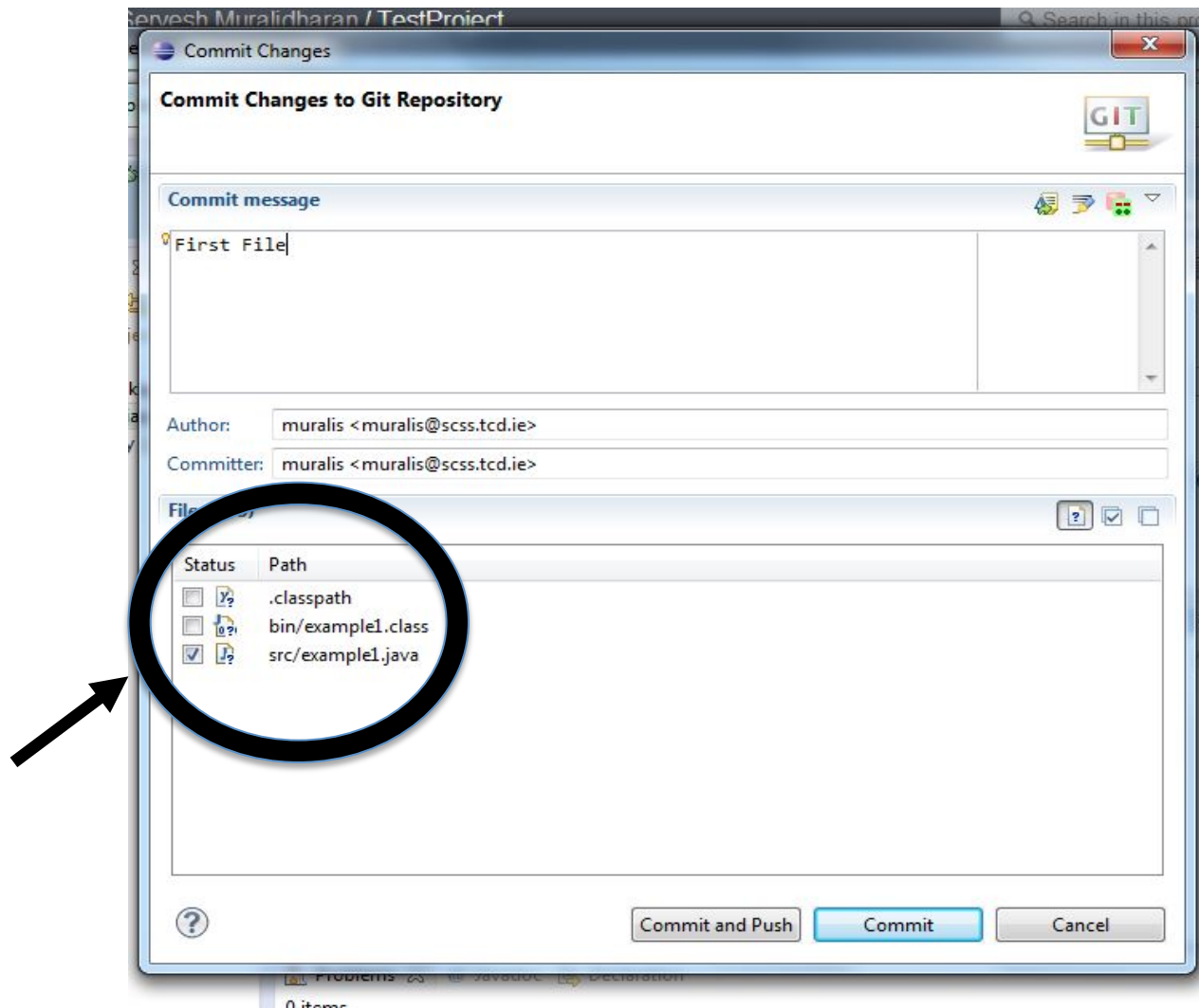
Every other time (same user, on the same computer)

- Pull global to local
- Change files on disk
- Add more files to local repo
- Commit (local)
- Push (local → global)

1. Right click on **project name** in project explorer window then select **Team** followed by **Commit**



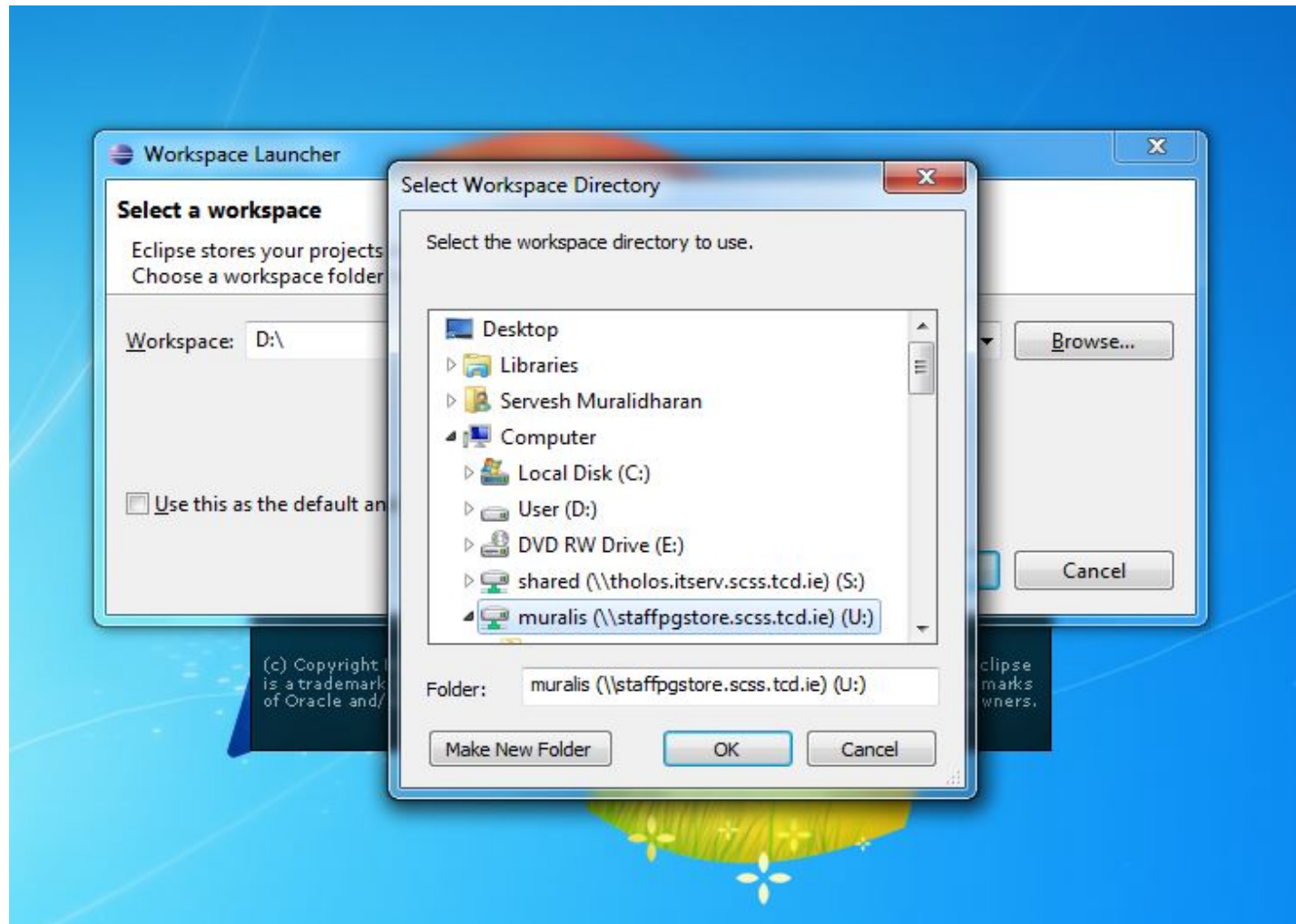
2. Commit and push the changes, but make sure to **add** the newly created files



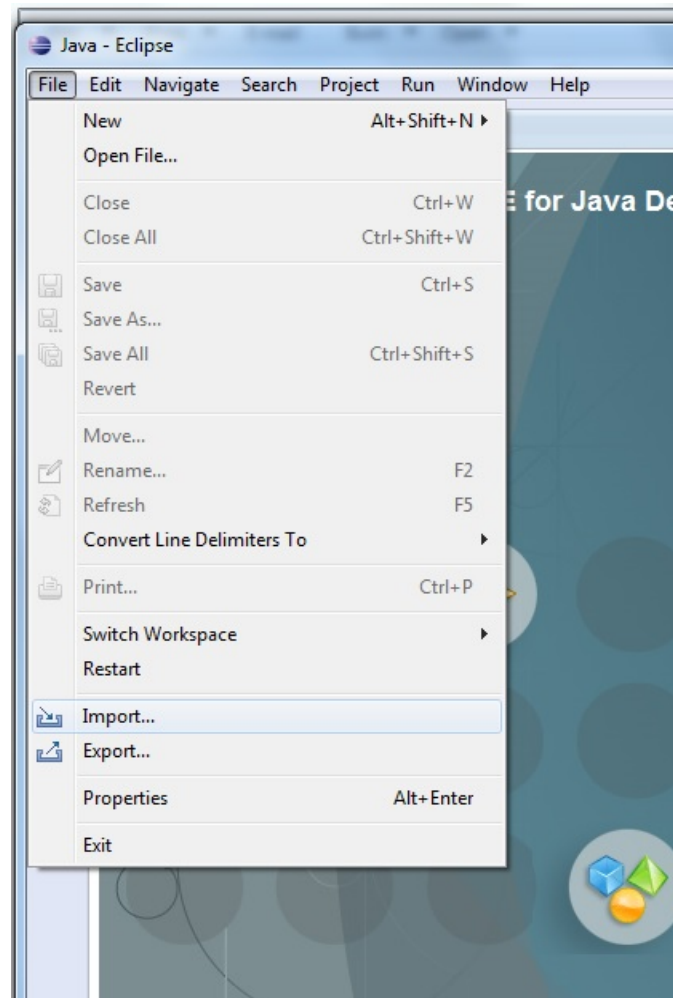
Every other time (every user, on different computer)

- Clone the remote repository
- Change / add files
- Commit
- Push

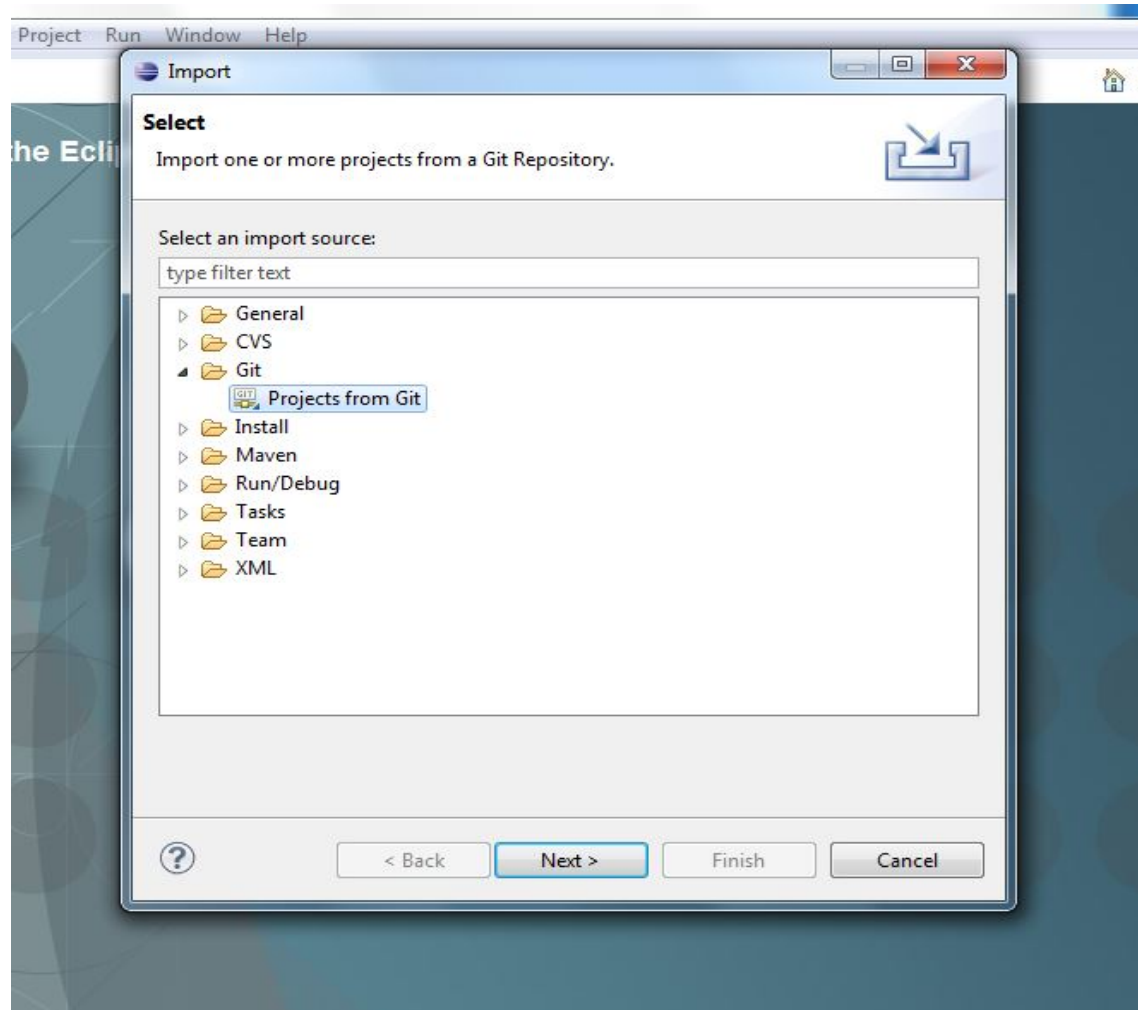
1. You can choose your workspace directory in 'U' Drive, if you don't want to keep setting up eclipse on every system or alternatively you can use your personal machine



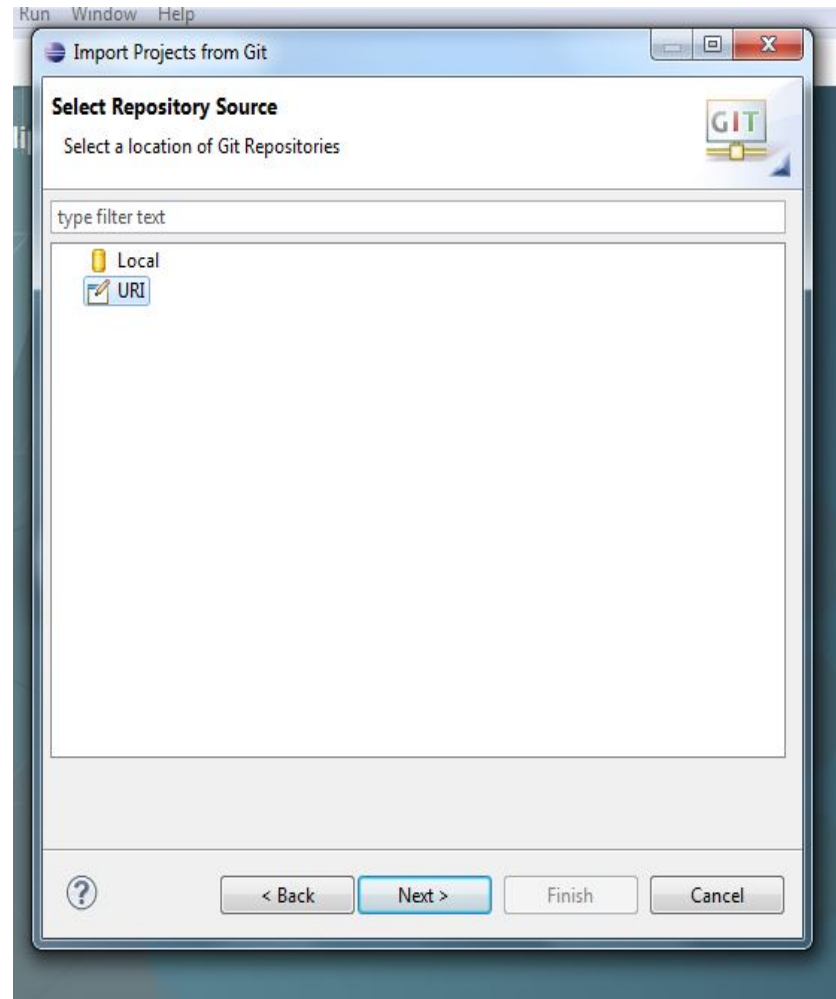
2. Choose import wizard from file menu.



3. Select 'Git' followed by 'Projects from Git'



4. Choose the 'URI' option



5. Provide the link you saved from the gitlab
6. Provide your scss username and password.

Import Projects from Git

Source Git Repository
Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

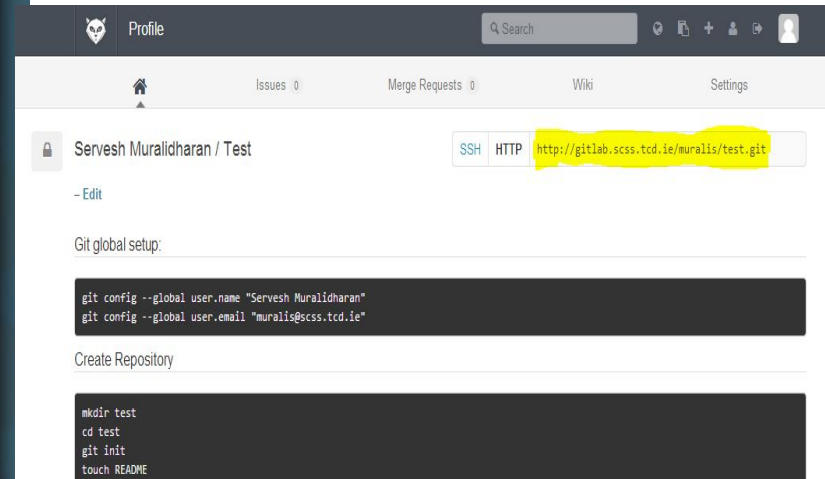
Port:

Authentication

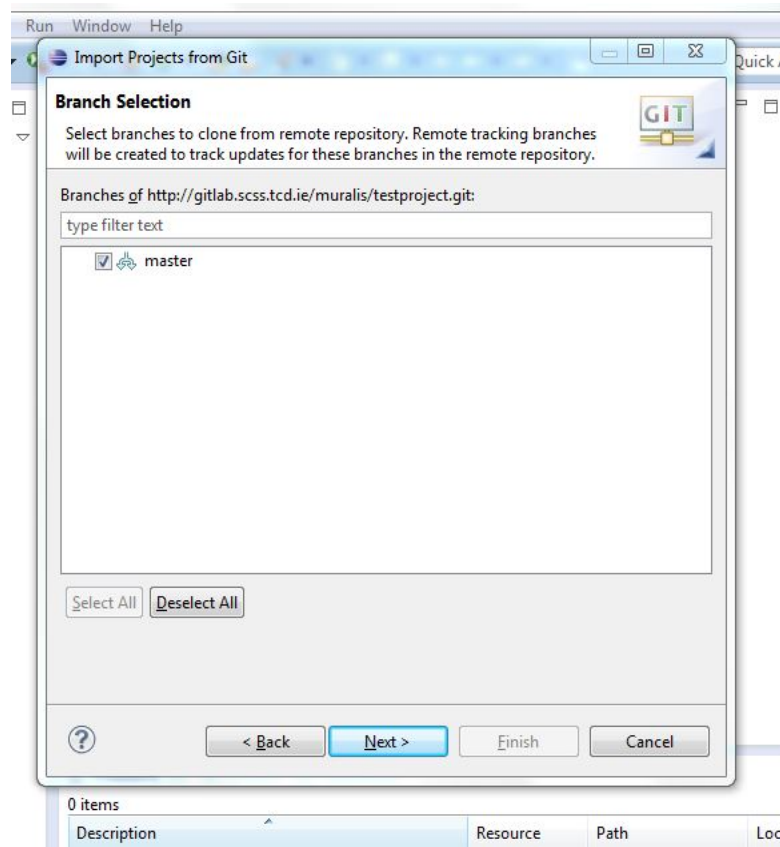
User:

Password:

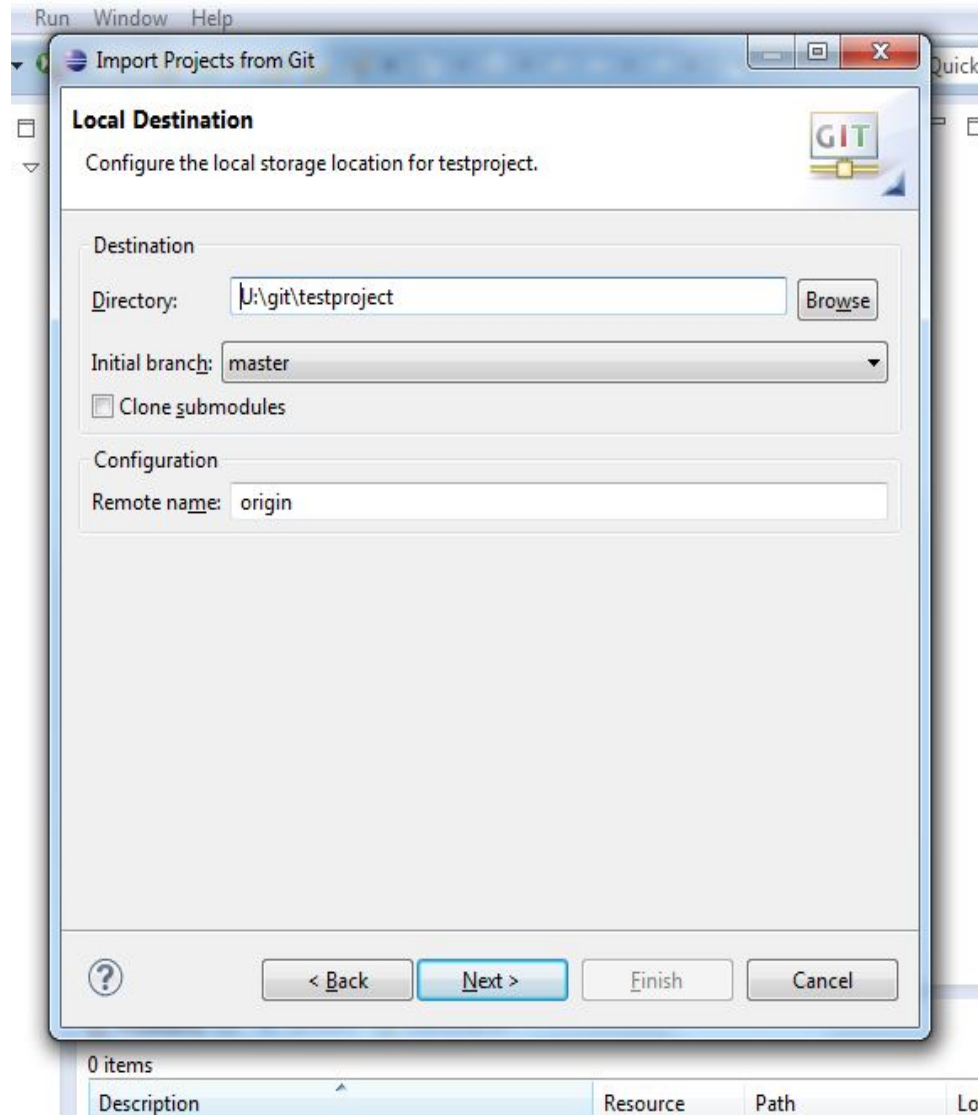
Store in Secure Store ☒



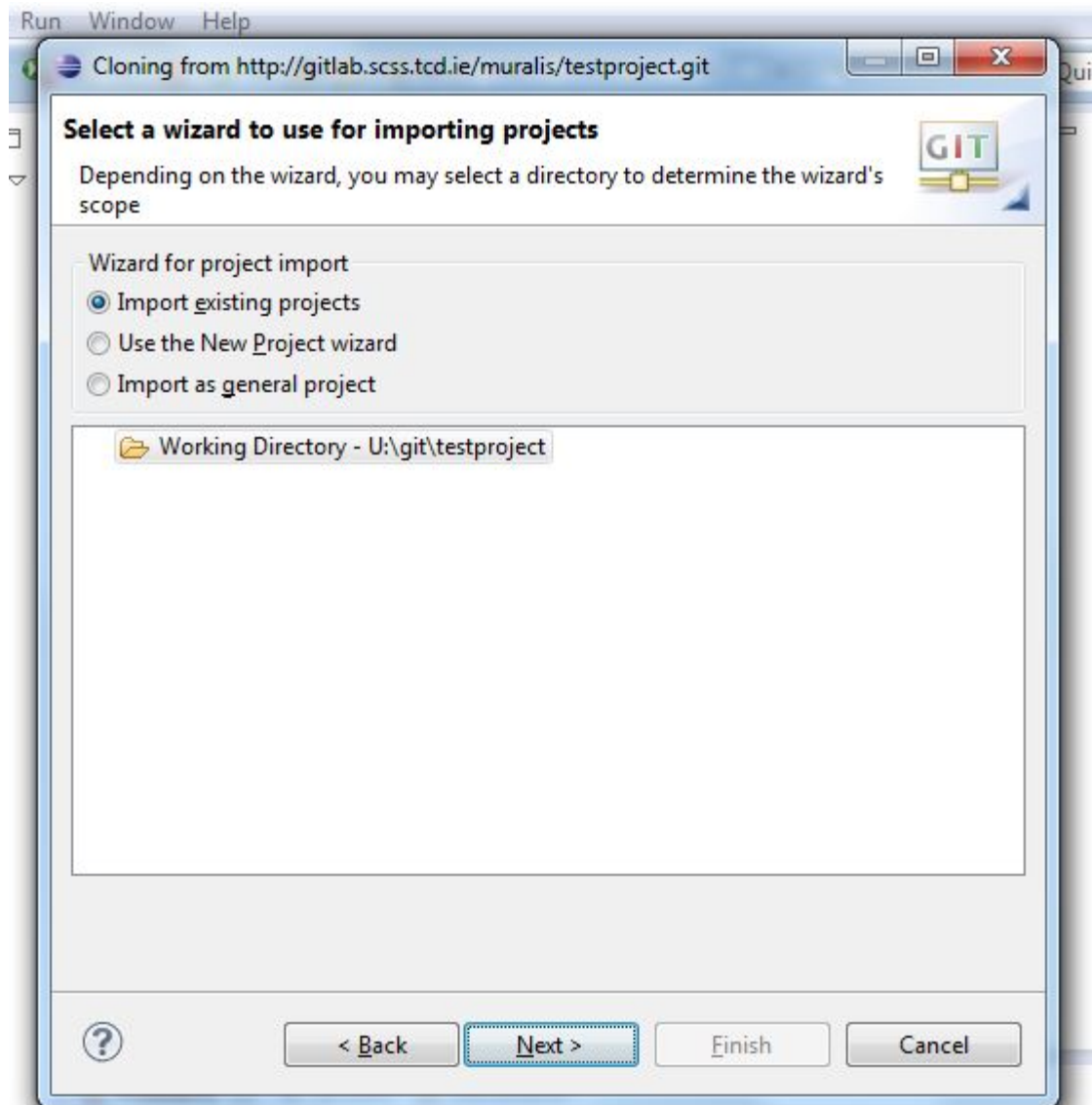
7. When you reach the Branch selection window you should find the already created master branch.
8. If you get permission error or unable to access the repository ensure that you have the relevant access privileges when the team lead creates them
9. Alternatively you can log into your gitlab account and should notice the project you have been added under



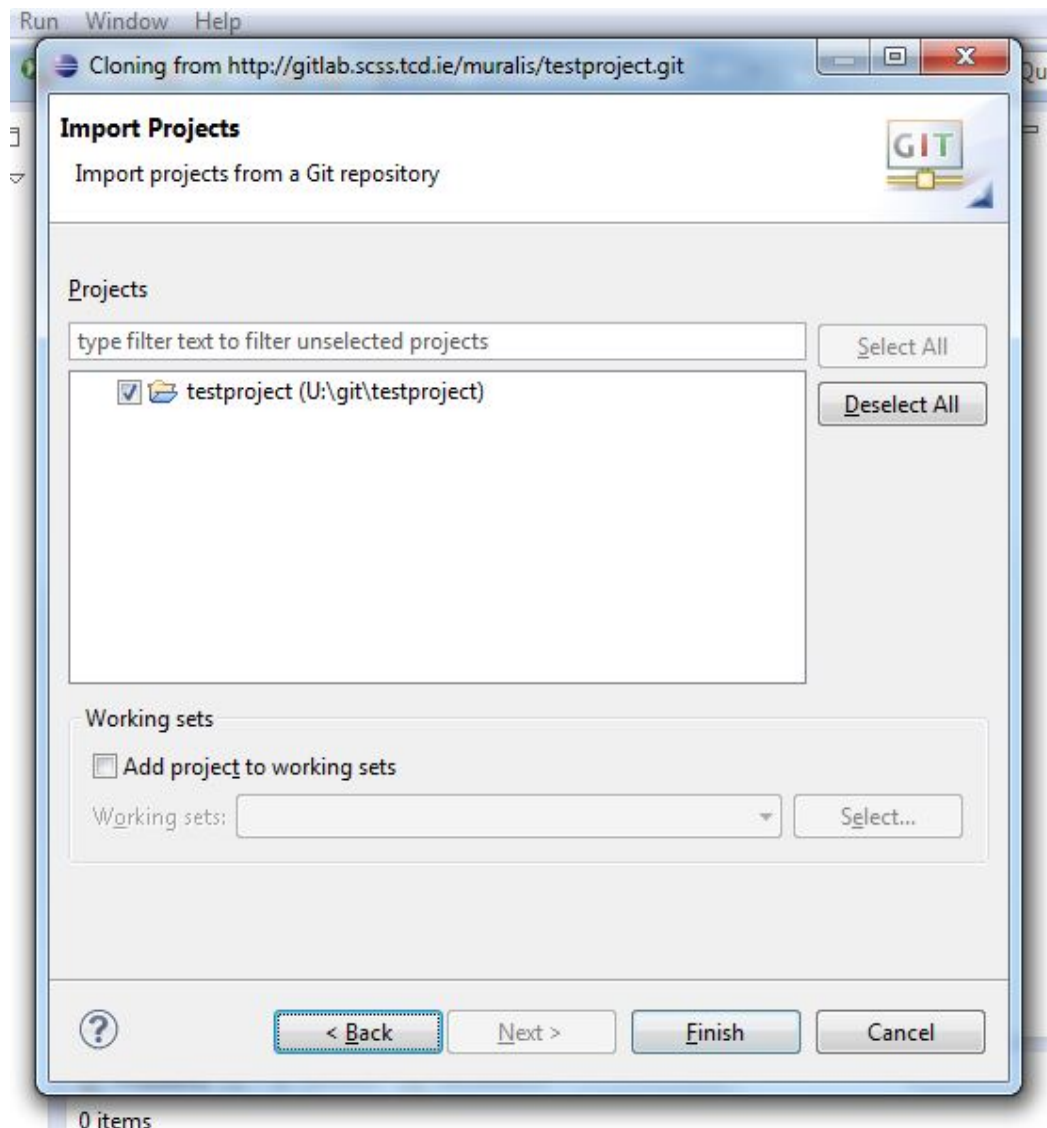
10. Specify the location where the local repository should be created



11. Make sure you choose 'Import existing projects'



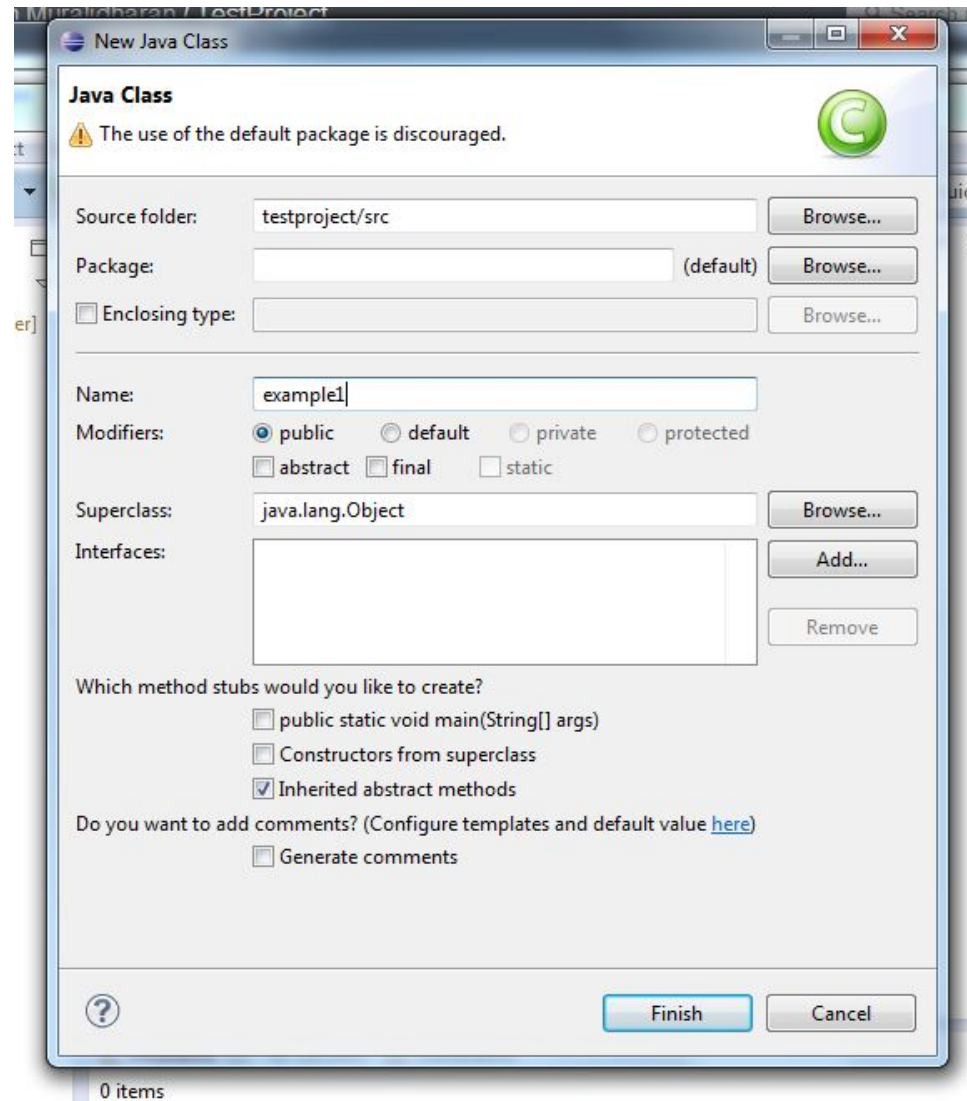
12. Finally click on finish to complete the import of the existing project



Every other time (every user, on different computer)

- Clone the remote repository
- Change / add files
- Commit
- Push

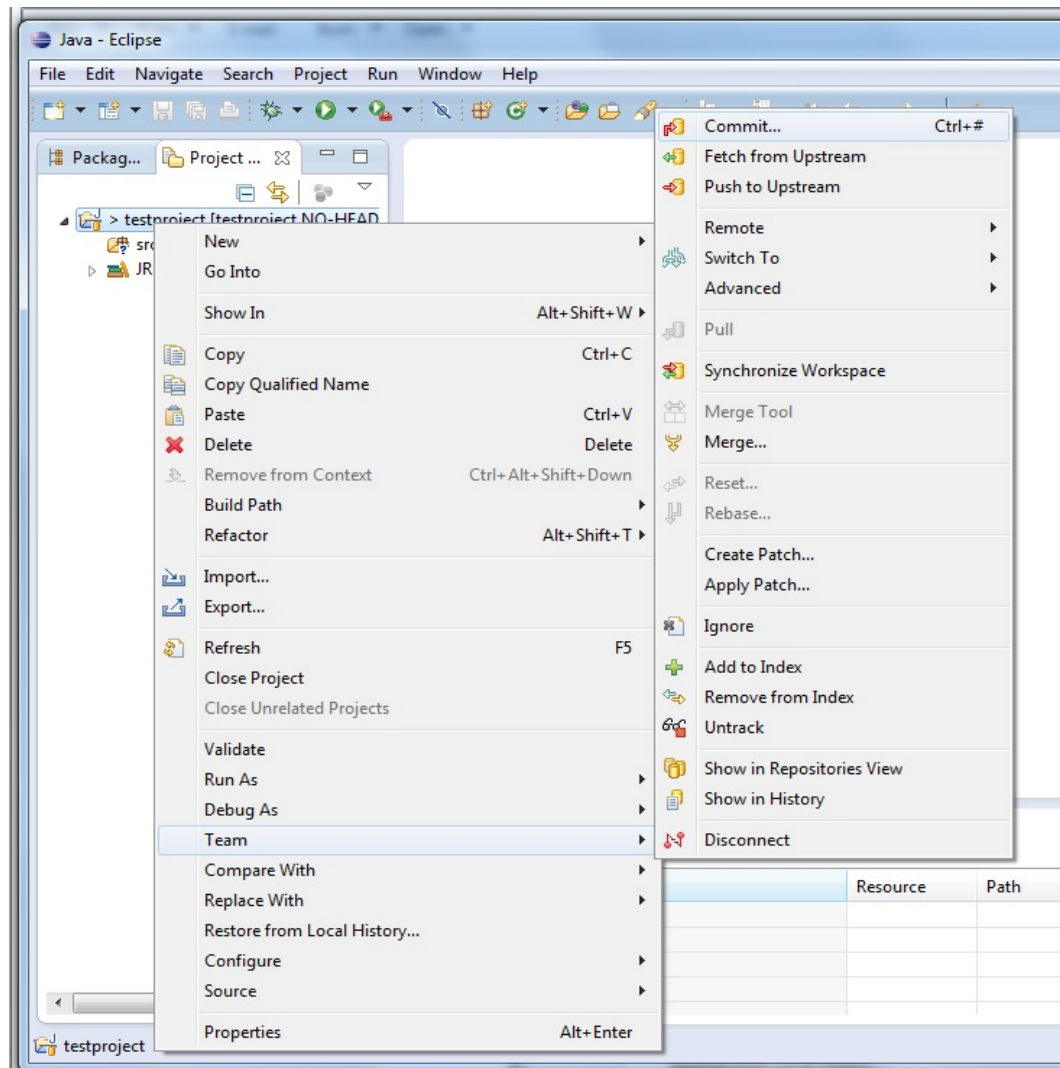
1. Add a file to the java project as usual



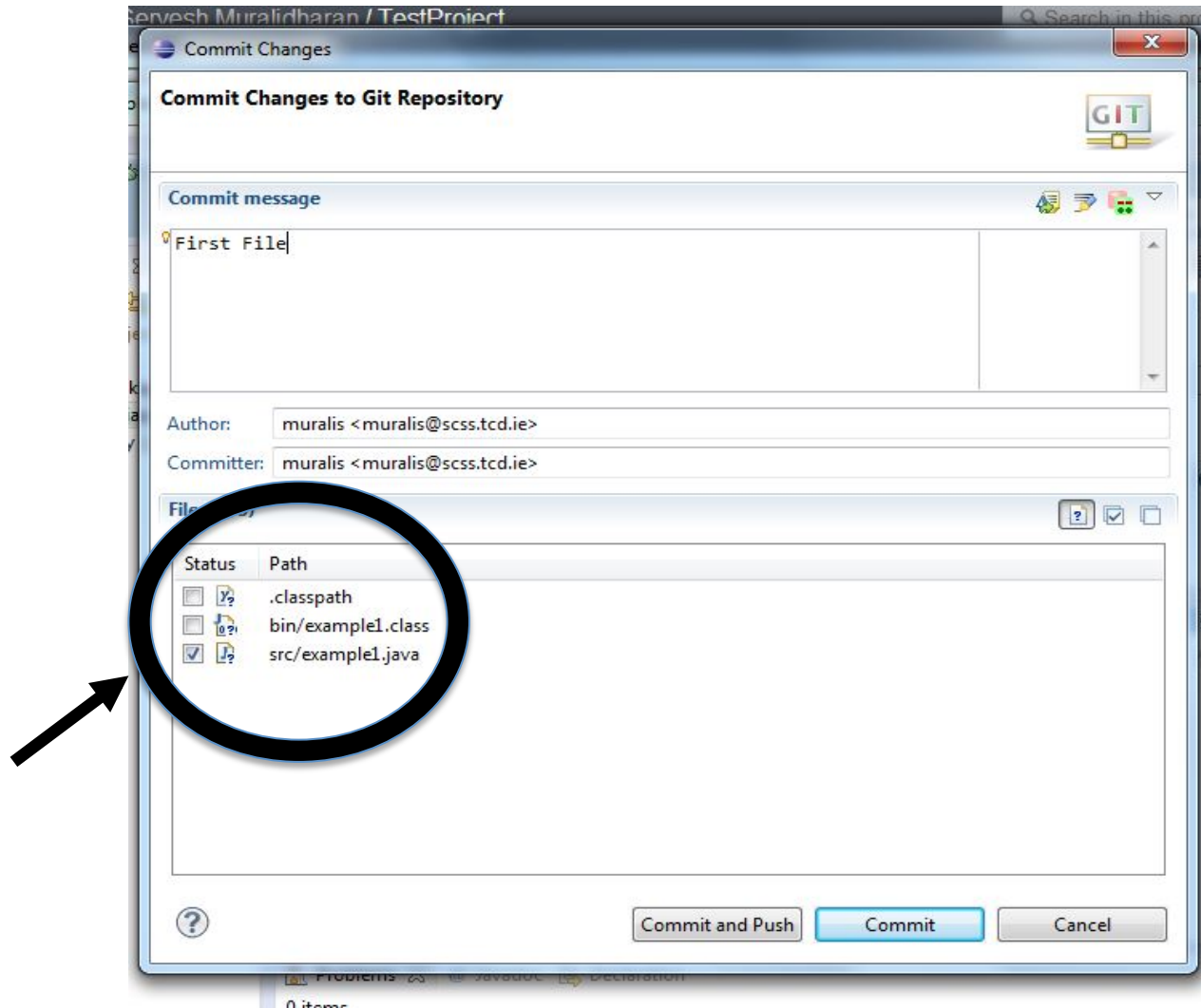
Every other time (every user, on different computer)

- Clone the remote repository
- Change / add files
- Commit
- Push

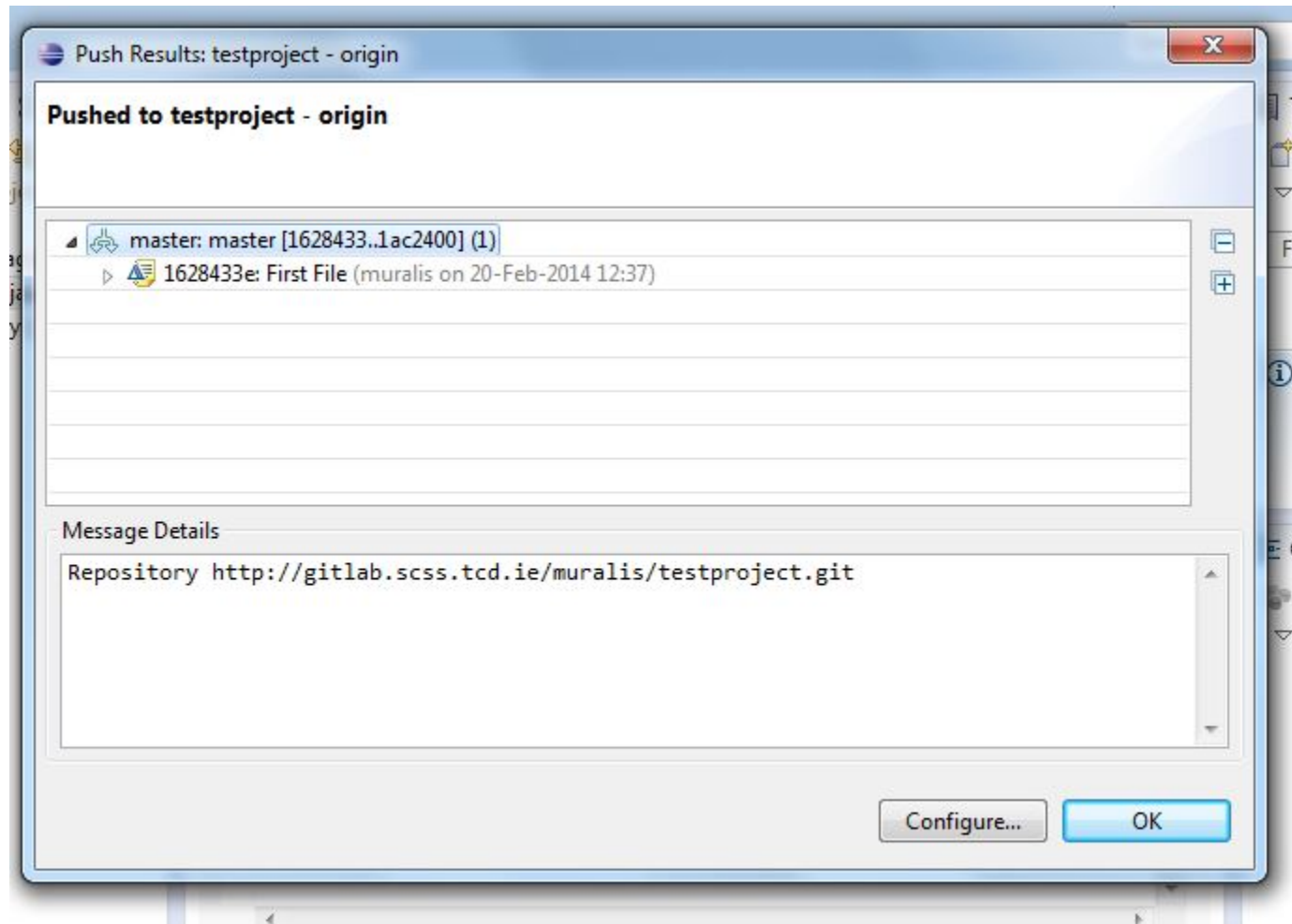
1. Right click on **project name** in project explorer window then select **Team** followed by **Commit**



2. Commit and push the changes, but make sure to **add** the newly created files

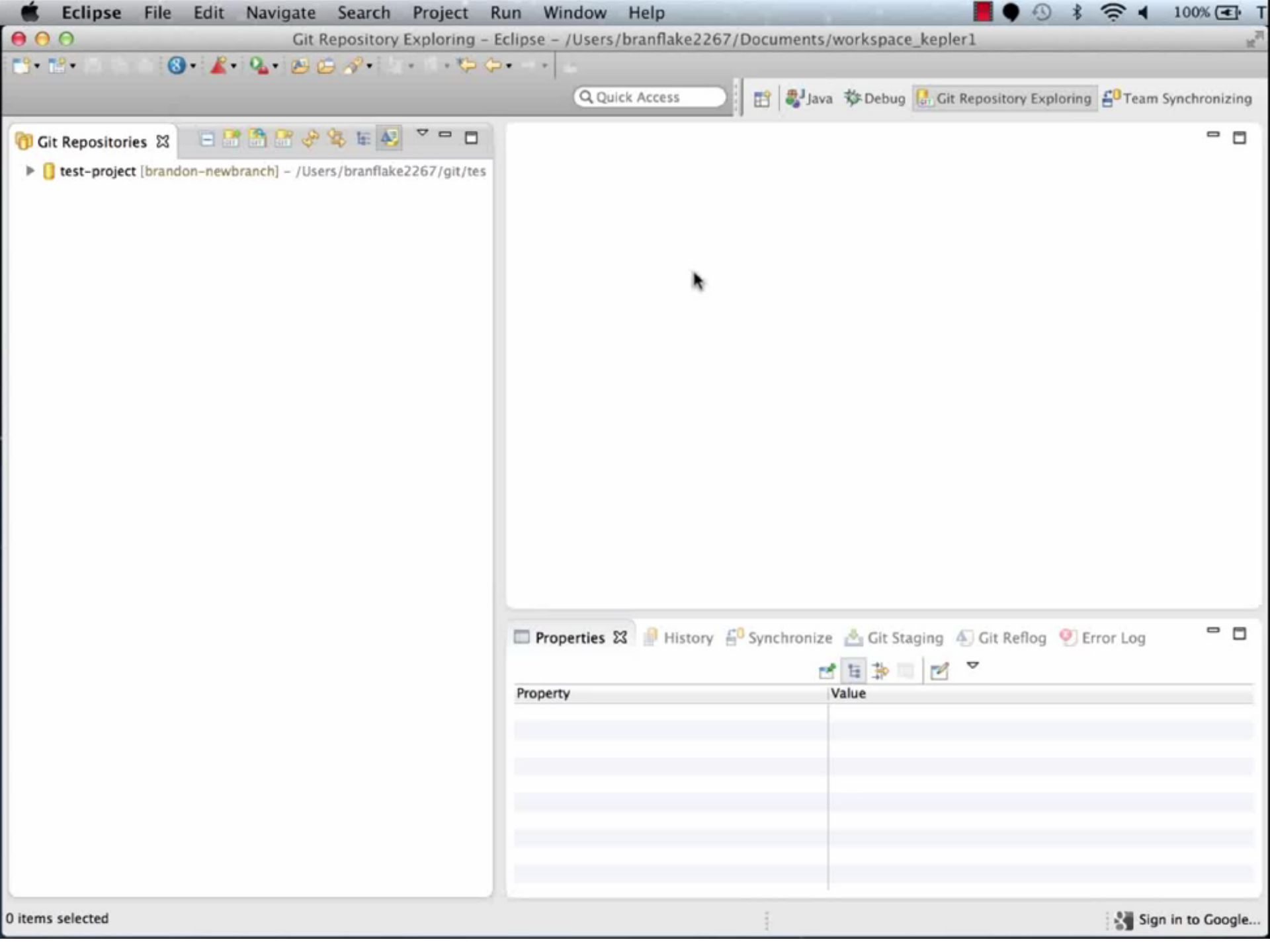


3. Notice a new node being created for the commit of the file



“Conflict resolution”

- Both partners edit the **same lines of code**
→ conflict
 - Git will refuse to push code onto remote repository
 - Prompts for resolution
- Resolution: **merge** conflicts manually
 - In Eclipse: <http://youtu.be/ZK20jVt7XEc>
 - Commit/push again



Avoiding conflicts

- Simple solution
 - Team work
 - Decide on a function for each team member
 - Only one team member modifies a particular function at a time
 - Commit and push code every time you have reasonable set of changes
 - Always pull the code from the server before you modify existing code
- but it's not tragic!

This Tutorial

- What is Source Control
- Distributed source control with Git
- Git in Assignment 5
- Using Gitlab
- Using git in Eclipse
- **More about Assignment 5**

Specifics for Assignment 5

- Deadline for setting up your team members, **Friday, March 7th**.
- Report to us if you don't have a team.
- Deadline for setting up a bare repository and each team member committing and pushing a change is end of day **Tuesday, March 11th**.
- You would need to submit the team names, members and the git URL as a text document to the submission server.