

DevOps



Caltech

Center for Technology &
Management Education

Post Graduate Program in DevOps



Git Plugin with IDE

Learning Objectives

By the end of this lesson, you will be able to:

- 🕒 Work with Git workflow on Eclipse IDE
- 🕒 Add Eclipse project to GitHub repository
- 🕒 Use Git in IntelliJ IDE



EGit and Eclipse IDE

What Is EGit?

EGit is an Eclipse plug-in which allows you to use the distributed version control system “Git” directly within the Eclipse IDE.



NOTE

EGit is based on the JGit library. JGit is a library which implements the Git functionality in Java.

Assisted Practice

Installation of EGit support into Eclipse IDE

Problem Statement: If Git tooling is not available, install it using the Eclipse installation manager.

Steps to Perform:

1. Go to: <http://download.eclipse.org/egit/updates> to get the plugin repository location
2. Go to **Help** and navigate to **Install New Software**
3. To add repository location, click **Add**. Enter repository name as **EGit Plugin**. Location will be the URL copied from the above step. Now click **Ok** to add repository location
4. Wait for a while and it will display the list of available products to be installed. Expand **Eclipse Git Team Provider** and select **Eclipse Git Team Provider**. Now click **Next**.

Working with Git on Eclipse IDE

Create a Git Repository using Eclipse

Create a Git Repository

Create a ".gitignore" file

Create an Eclipse project

Share Project

Use Git Staging

Commit Changes

Review Commit History

Open the Git Repositories view by navigating to the **Window>Show>View>Other>Git>Git Repositories** menu entry. From the toolbar, select the **Create a new Git Repository** and add it to this view entry.



Create a Git Repository using Eclipse

Create a Git Repository

Create a ".gitignore" file

Create an Eclipse project

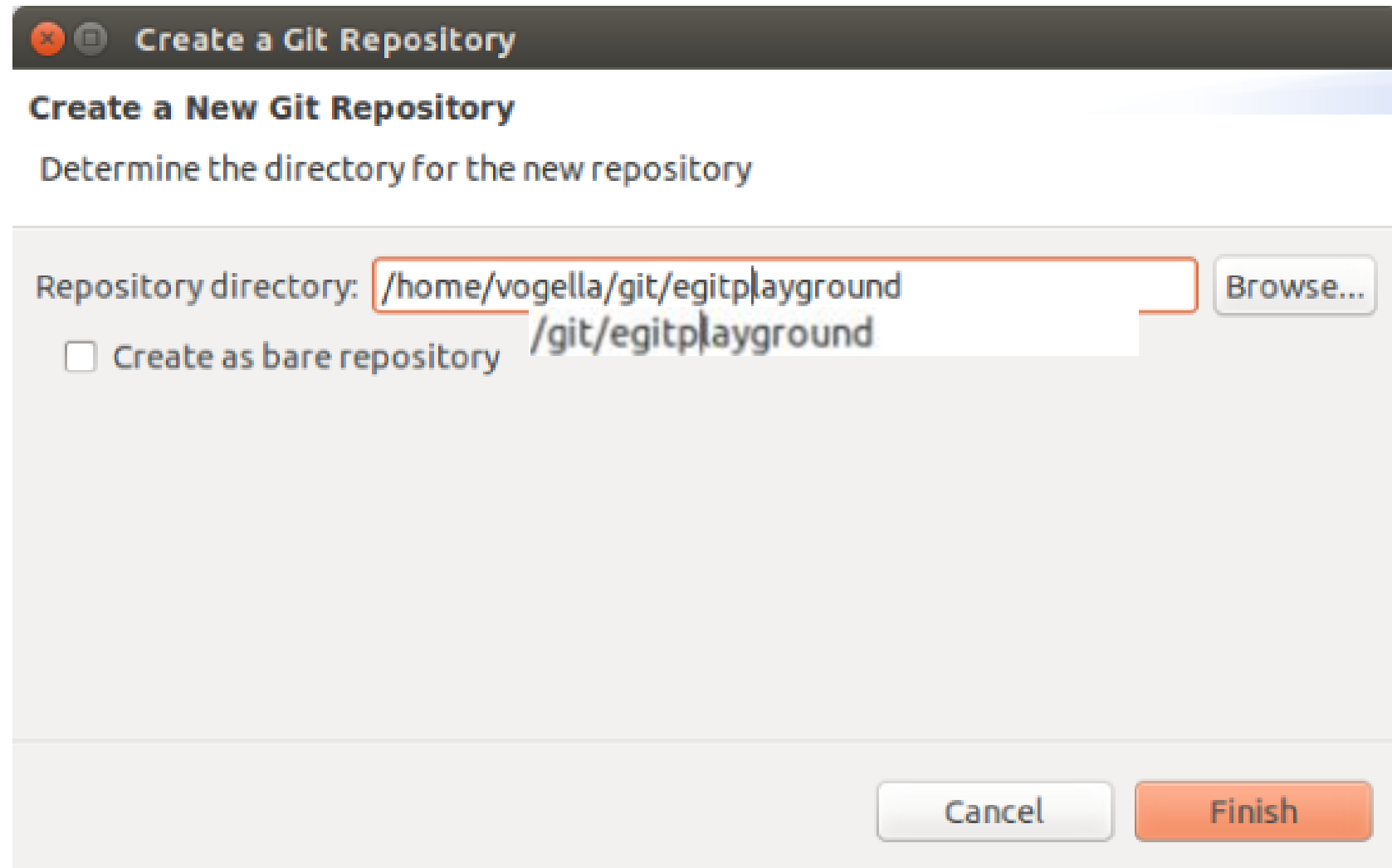
Share Project

Use Git Staging

Commit Changes

Review Commit History

A dialog box appears which allows you to specify the directory for the new Git repository. Select a new directory outside of your workspace.



Press the Finish button. Now the Git repository is created and a reference to it is added to the **Git Repositories** view.

Create a .gitignore file

Create a Git Repository

Create a ".gitignore" file

Create an Eclipse project

Share Project

Use Git Staging

Commit Changes

Review Commit History

All files and directories which apply to the pattern described in a top-level ".gitignore" file are ignored by Git.

NOTE

Eclipse Git does not allow to create a file directly in the top-level folder of your repository. You must do this outside of the Eclipse IDE using the command line.

Create a Git Repository

Create a .gitignore file

Create an Eclipse project

Share Project

Use Git Staging

Commit Changes

Review Commit History

Create an Eclipse Project

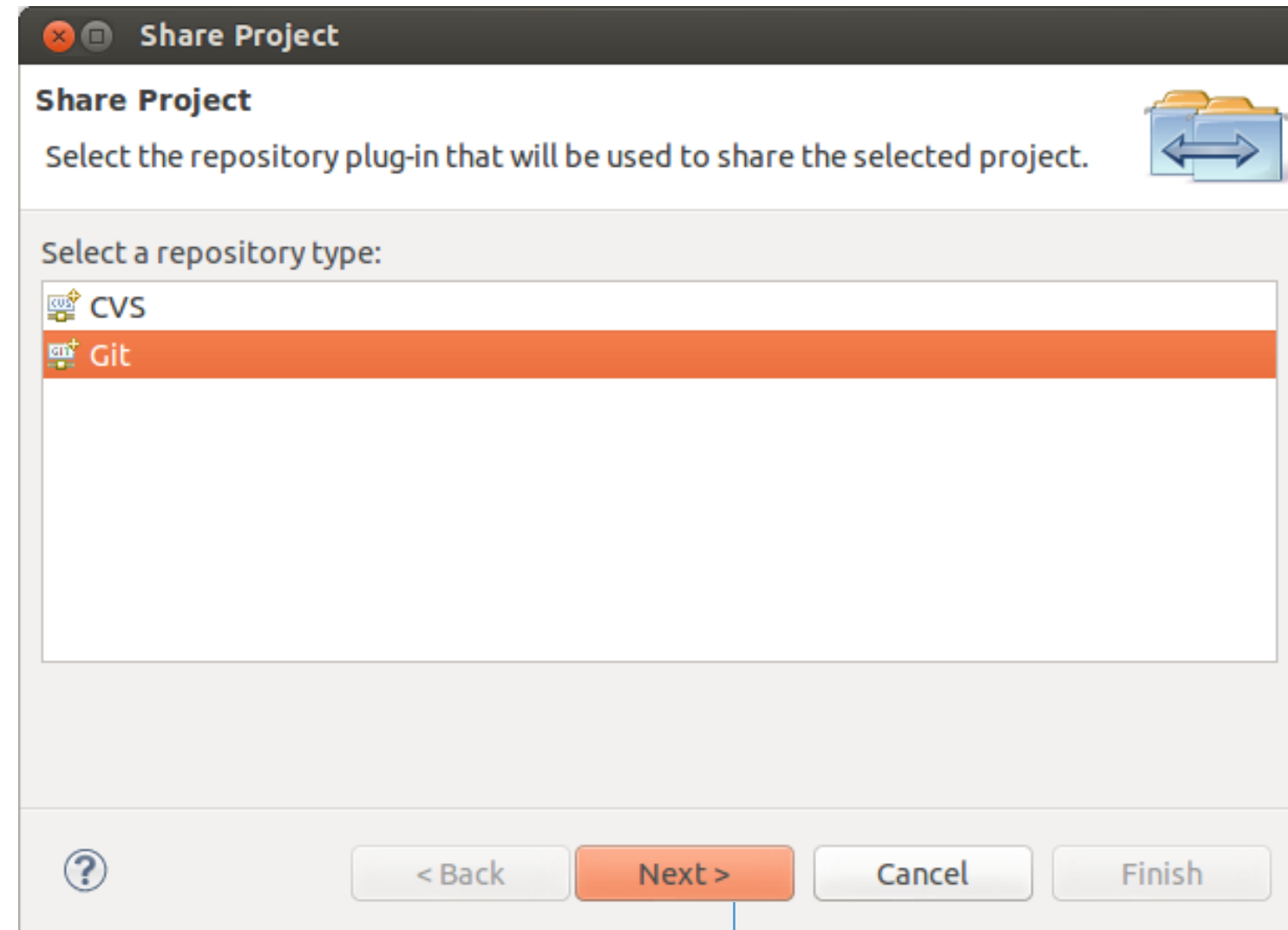
Create a new Java project called by navigating to the **File>New>Other>Java Project** menu entry

```
package com.sample.git.first;

public class GitTest {
    public static void main(String[] args) {
        System.out.println("Git is fun");
    }
}
```

Share Project

Share your new project under version control with Git by right clicking on your project and selecting **Team>Share>Git**.



Click **Next** and select your existing Git repository from the drop-down list.

Create a Git Repository

Create a .gitignore file

Create an Eclipse project

Share Project

Use Git Staging

Commit Changes

Review Commit History

Create a Git Repository

Create a .gitignore file

Create an Eclipse project

Share Project

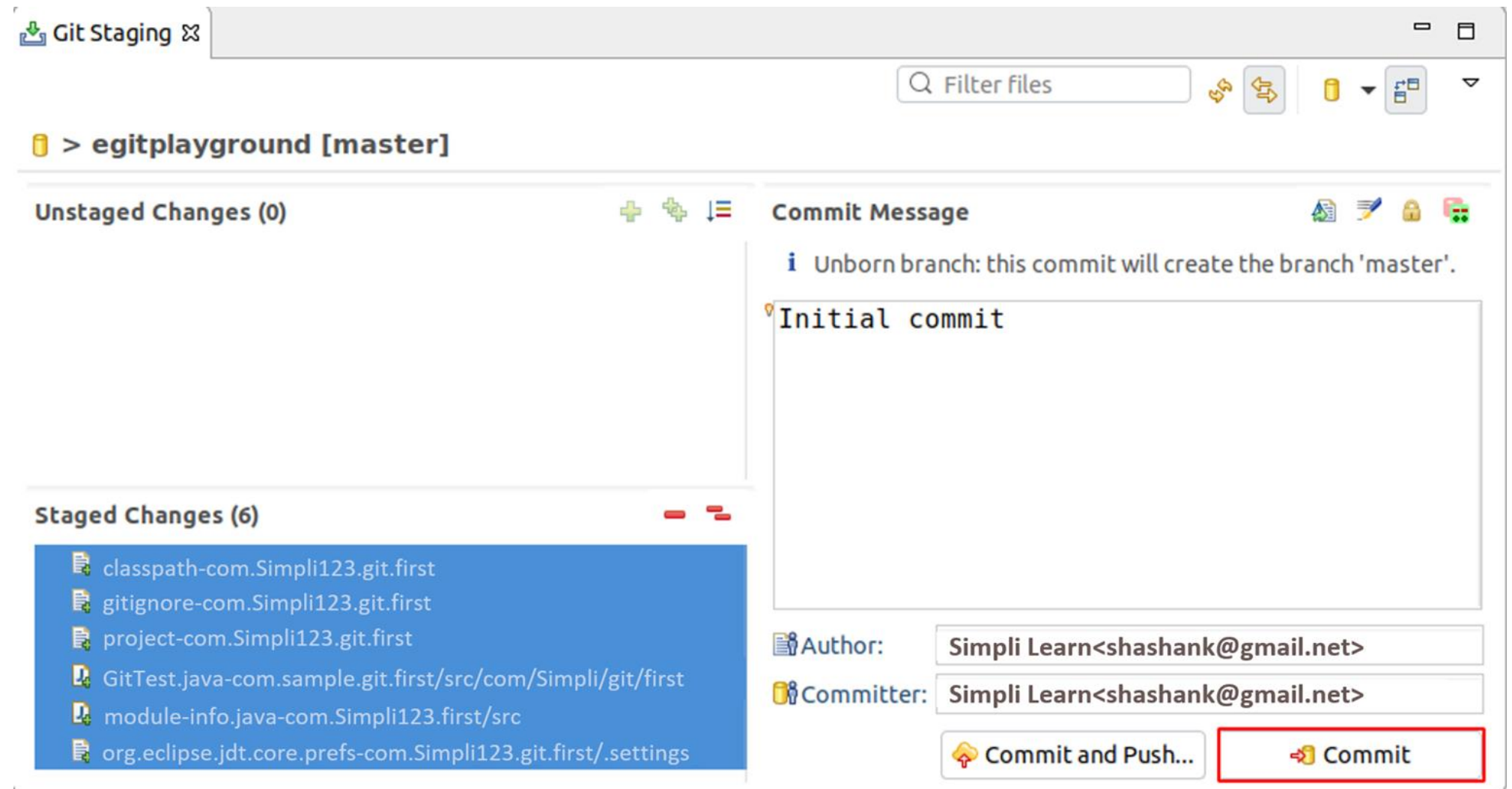
Use Git Staging

Commit Changes

Review Commit History

Use Git Staging Area

Open the Git Staging view, using **Window>Show>View>Other>Git>Git Staging**. In this view drag all files into the **Staged Changes** area, write a meaningful commit message, and press the commit button.



Commit Changes

Change the System.out.println message in your GitTest class. Stage the file and commit the changes.

```
package com.sample.git.first;

public class GitTest {
    public static void main(String[] args) {
        System.out.println("Git is cool");
    }
}
```

Create a Git Repository

Create a .gitignore file

Create an Eclipse project

Share Project

Use Git Staging

Commit Changes

Review Commit History

Create a Git Repository

Create a .gitignore file

Create an Eclipse project

Share Project

Use Git Staging

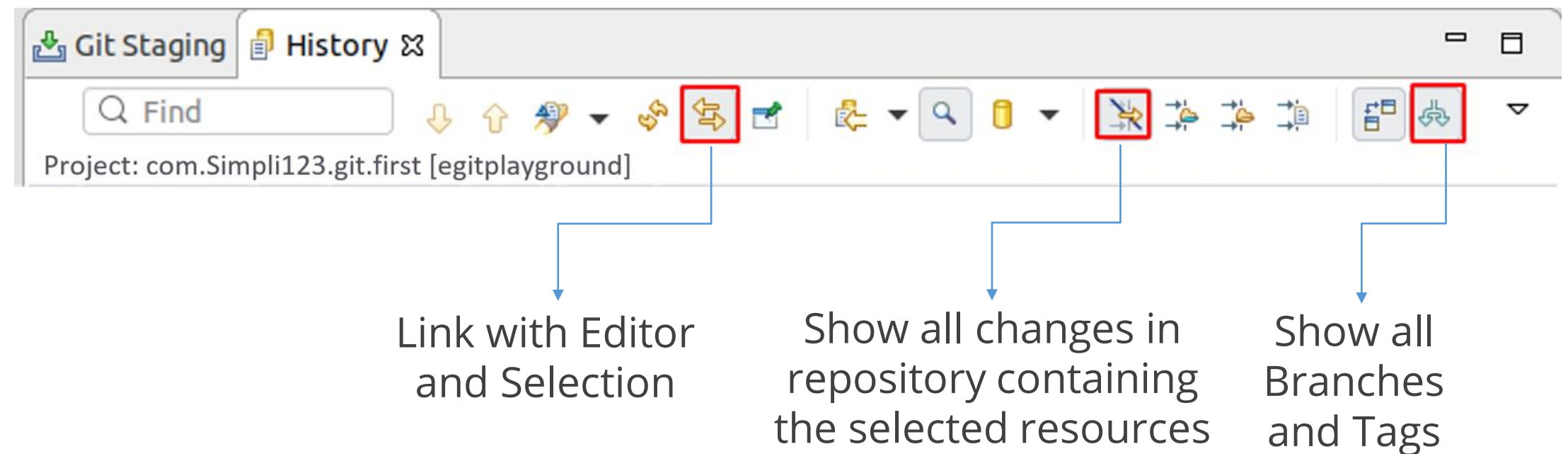
Commit Changes

Review Commit History

Review Commit History

Open the **History view** by navigating to the **Window>Show>View>Other>Team>History** menu. Use it to review which files were included in your individual commits.

In the **History** view click the toggle buttons as shown in the screenshot.

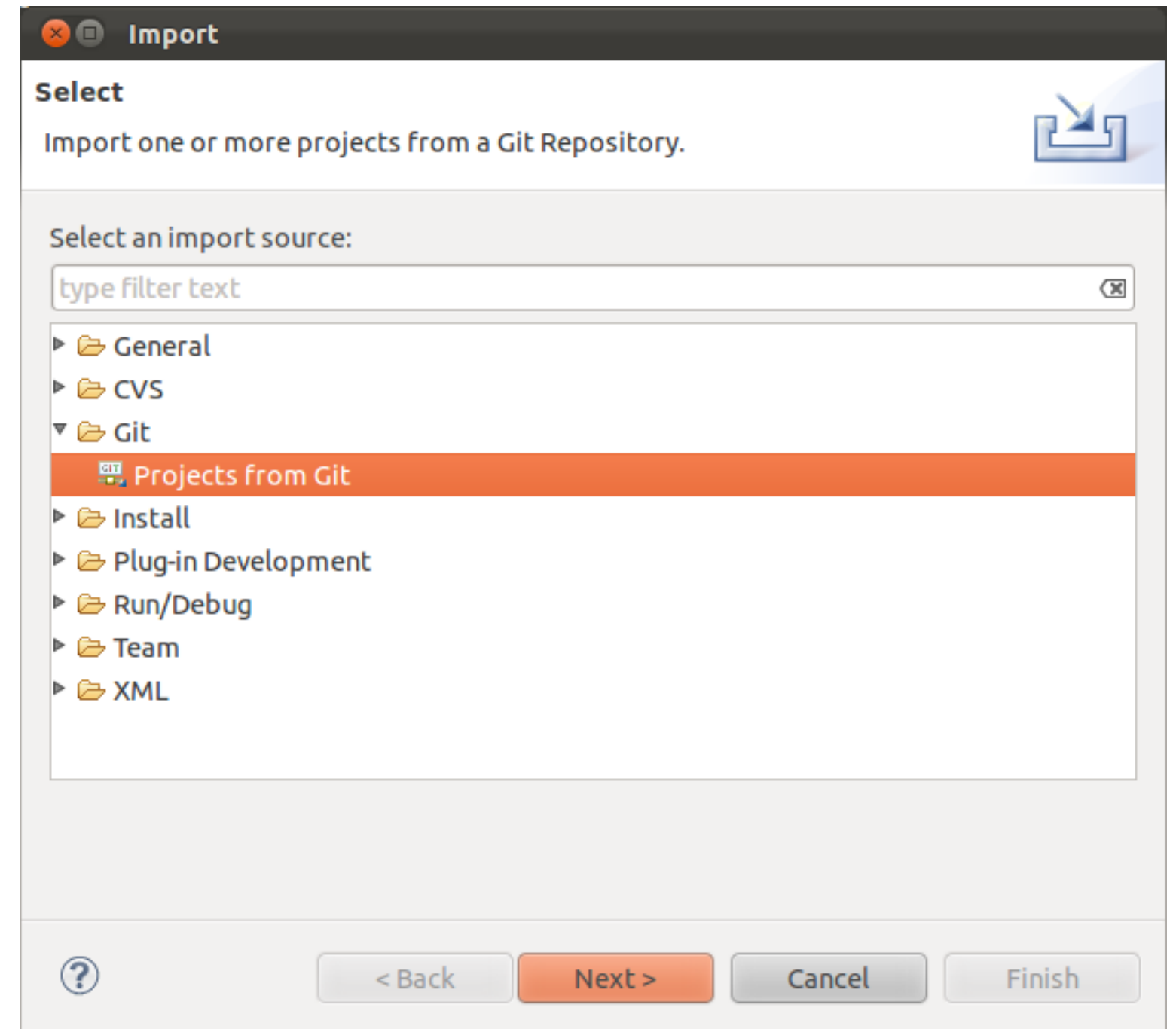


Clone a Repository in Eclipse

Cloning an Existing Repository

To clone a Git repository and import the existing projects from this repository into your workspace, follow these steps:

1. Select **File**
2. Navigate to **Import**
3. Open **Git** folder and Click **Projects from Git**
4. Select **Clone URL** in the next dialog box



Cloning an Existing Repository

Enter the URL to your Git repository which you want to clone.

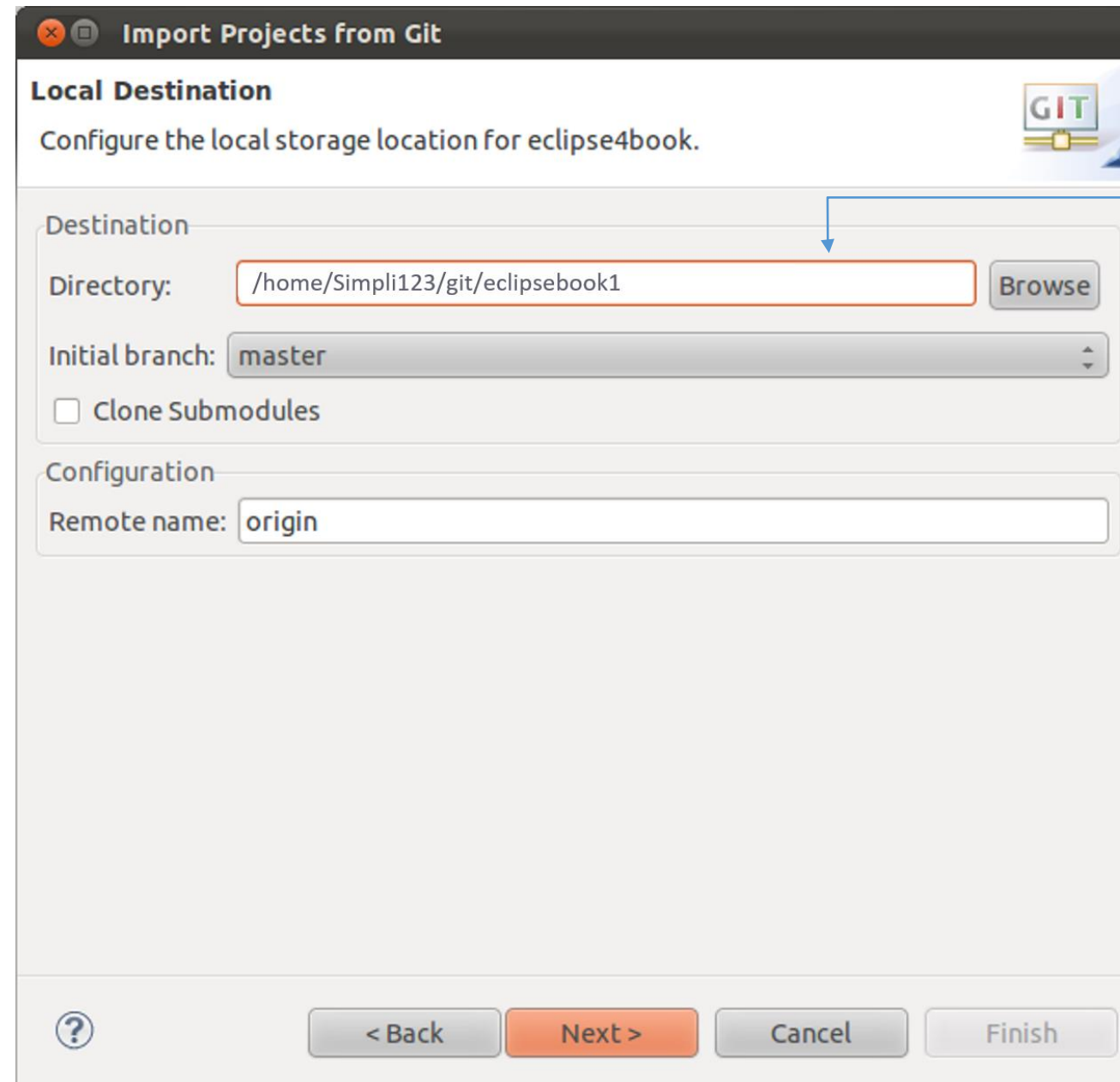
The screenshot shows a dialog box titled "Import Projects from Git" with a subtitle "Source Git Repository". The dialog is divided into three main sections: "Location", "Connection", and "Authentication". In the "Location" section, there are three input fields: "URI:" (with a blue arrow pointing to it from the text "Enter the URL of your repository"), "Host:" (containing "github.com"), and "Repository path:". In the "Connection" section, there is a "Protocol:" dropdown menu (set to "git") and a "Port:" input field. In the "Authentication" section, there are "User:" and "Password:" input fields, and a "Store in Secure Store" checkbox. At the bottom of the dialog, there are four buttons: a help button (question mark icon), "< Back", "Next >" (highlighted in orange with a blue arrow pointing to it from the text "After pressing Next, the system will allow you to import the existing branches. Select at least the master branch."), and "Finish".

Enter the URL of your repository

After pressing **Next**, the system will allow you to import the existing branches. Select at least the master branch.

Cloning an Existing Repository

Specify where the repository should be copied to and which local branch should be created initially.



Import Projects from Git

Local Destination
Configure the local storage location for eclipse4book.

Destination

Directory:

Initial branch:

☐ Clone Submodules

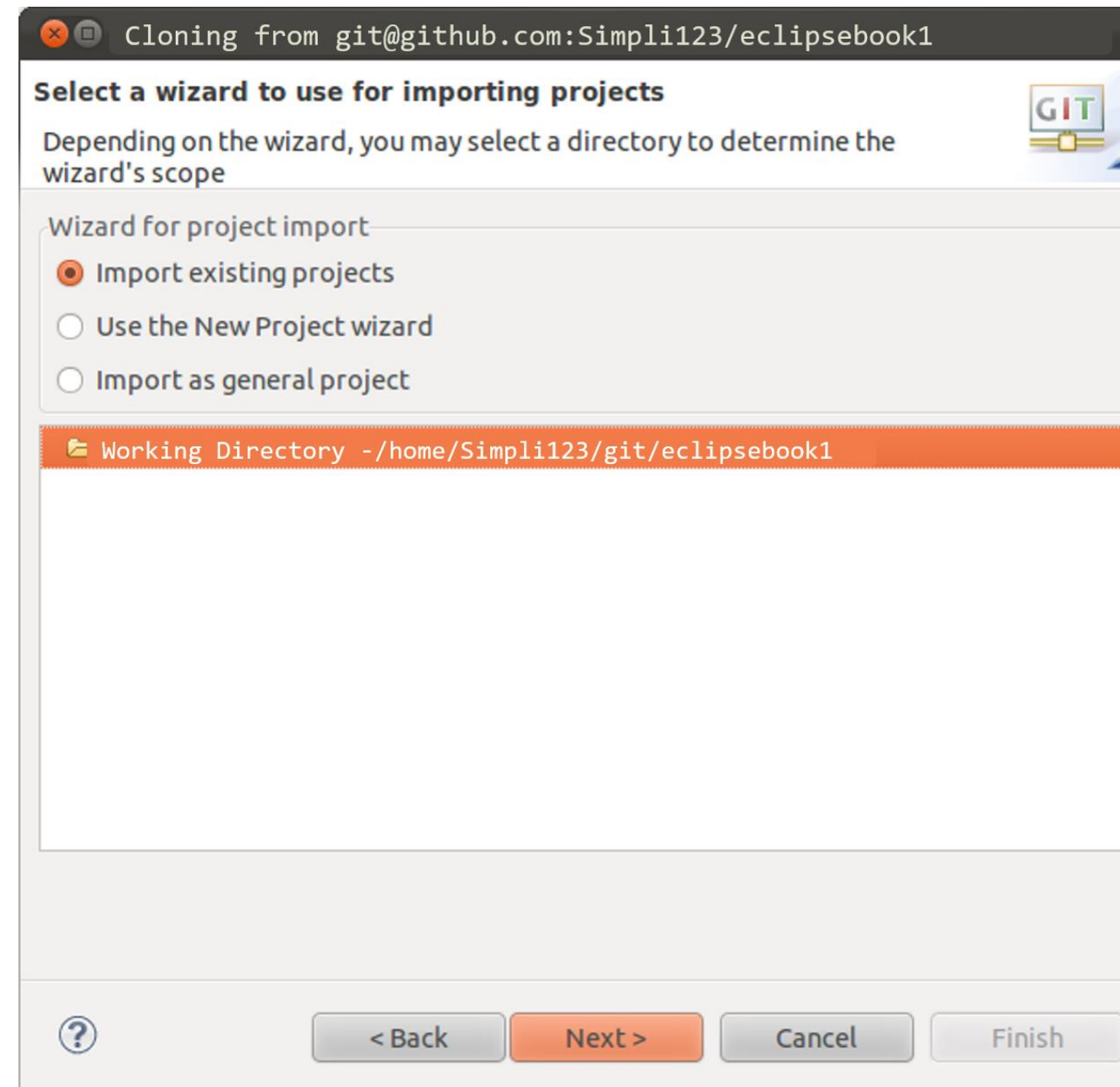
Configuration

Remote name:

Specify the destination directory

Cloning an Existing Repository

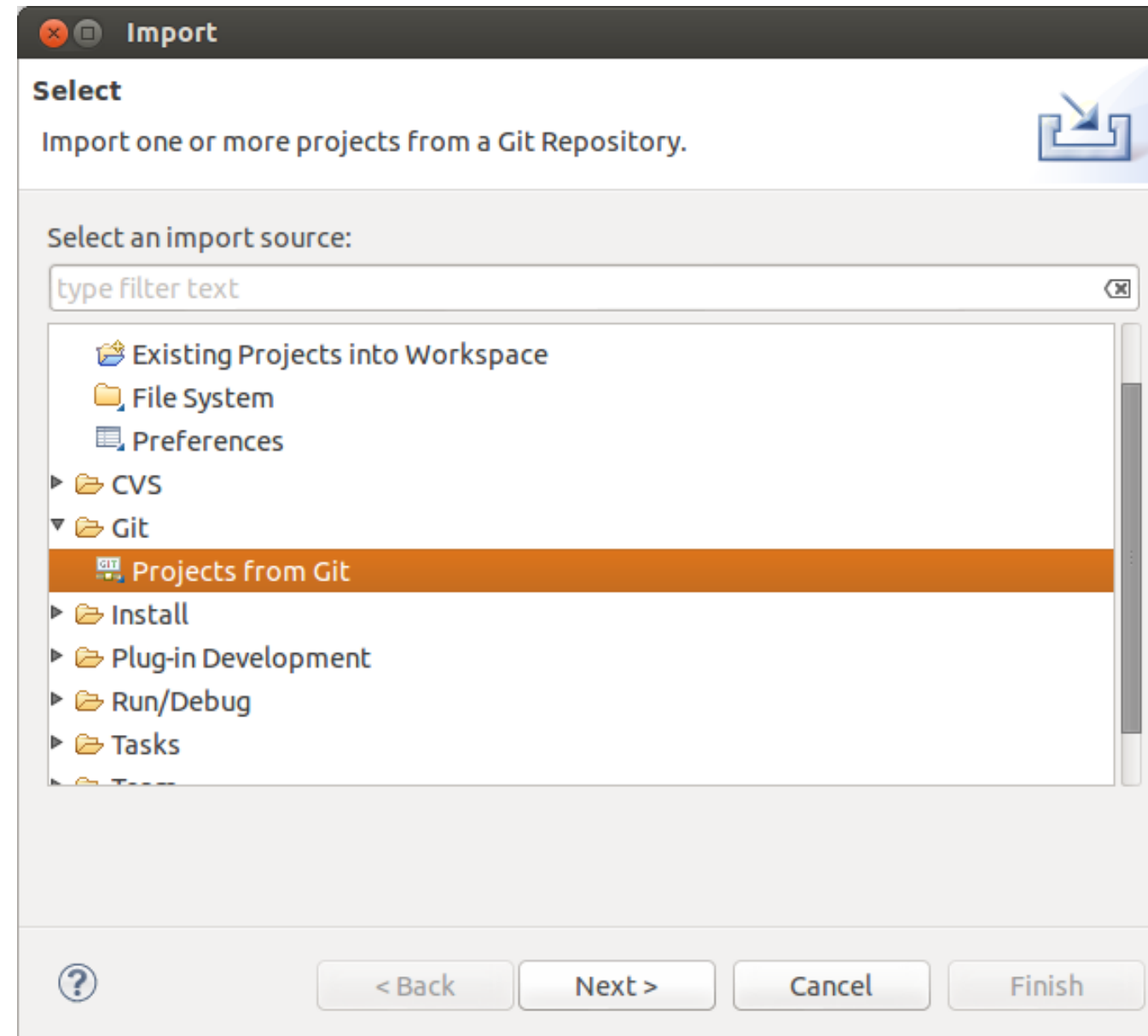
After the Git repository is cloned, import the existing projects.



Import Projects from a Repository

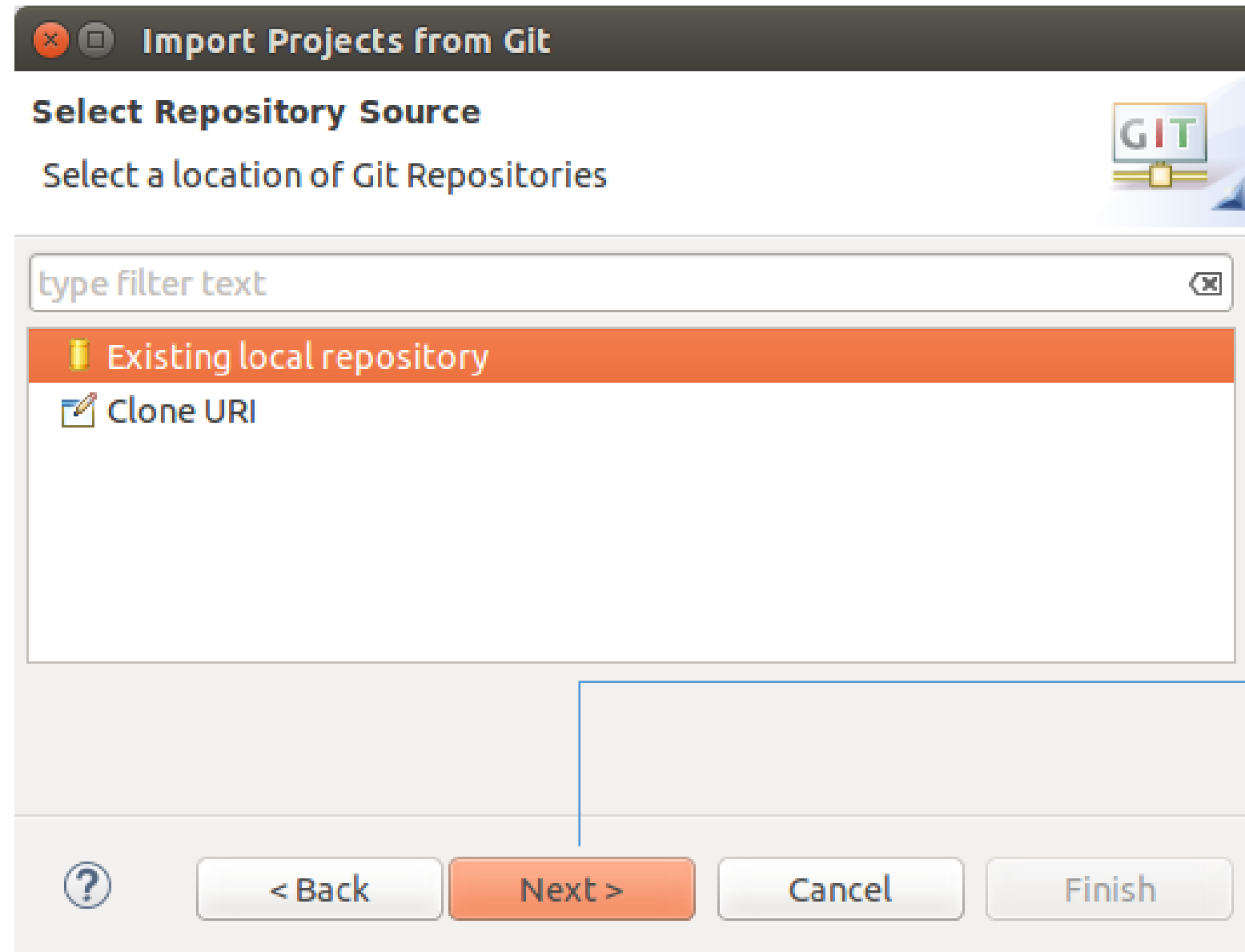
Import Projects from a Git Repository

Import the projects into your workspace by navigating to the **File>Import>Git>Project** from Git menu entry.



Import Projects from a Git Repository

Select **Local** to import from a local repository or **Clone URL** to clone the repository.

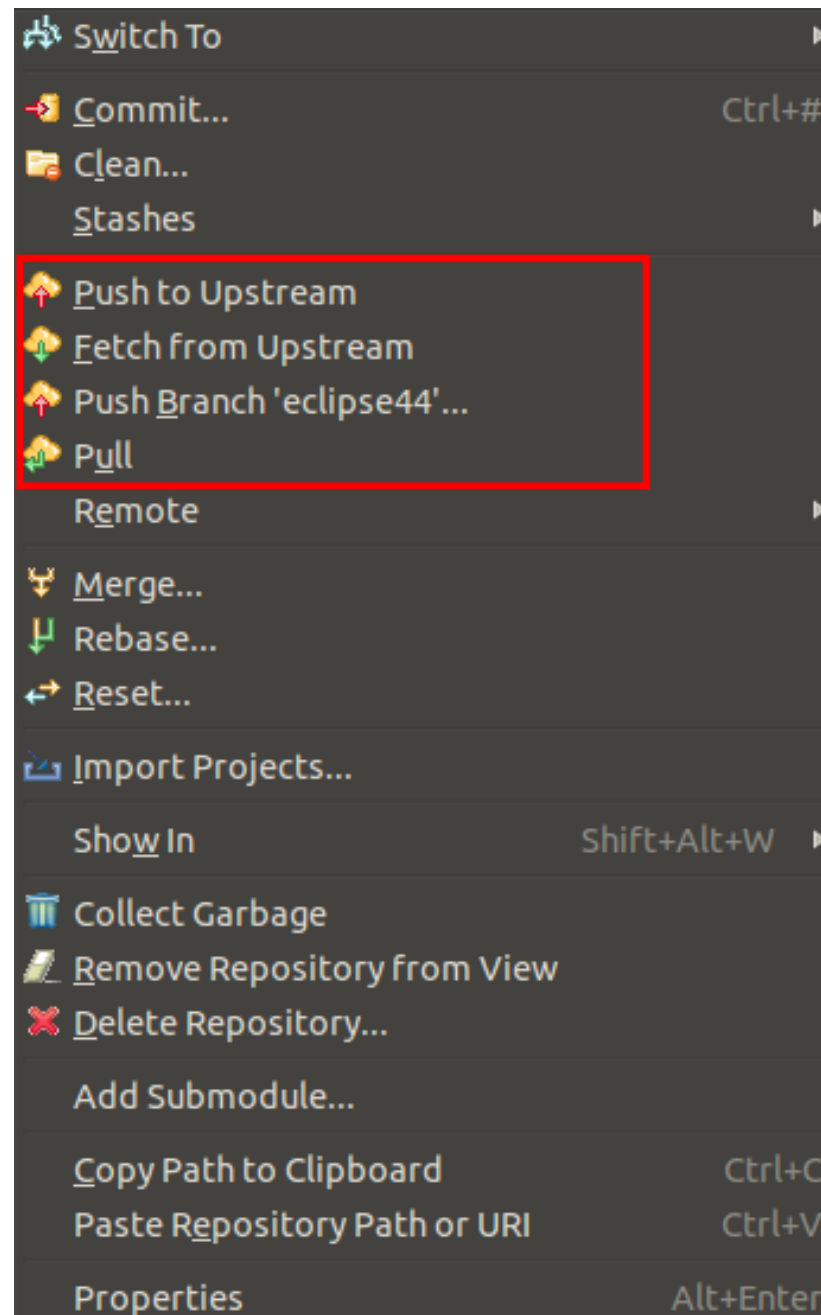


Click **Next** and select the repository to import the project stored in it.

Git Operations in Eclipse

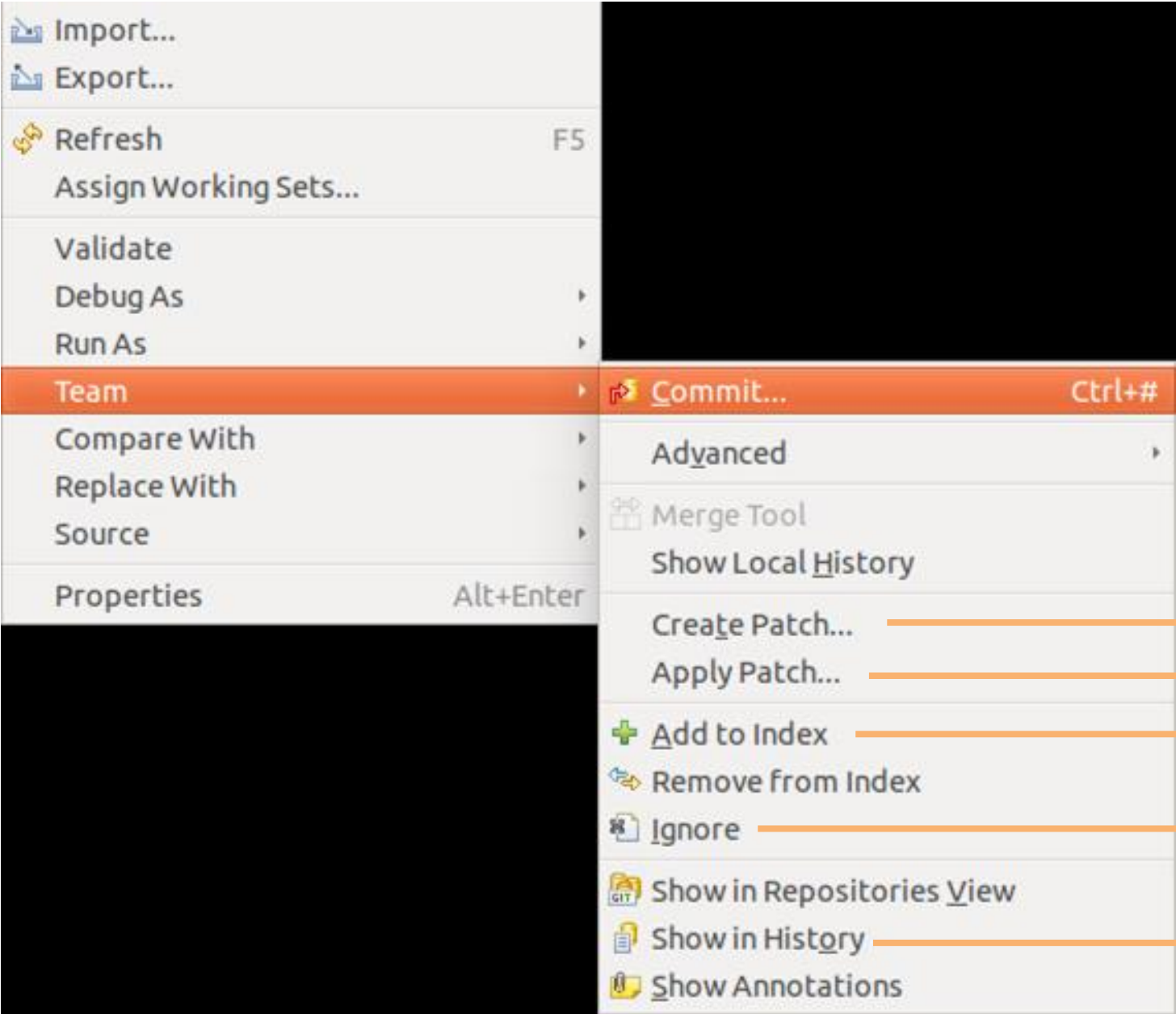
Pull, Push, and Fetch

Use the **Git Repositories** view to pull, push, and fetch to remote repositories. Right-click on your repository and select the required operation.



Basic Team Operations

After placing a project under version control, you can start using team operations on your project.



To open the commit dialog to create a new commit

To create a patch

To apply a patch

To add the selected resources to the Git index

To add a file to the .gitignore file

To display the history of the selected resources

Amending a Commit

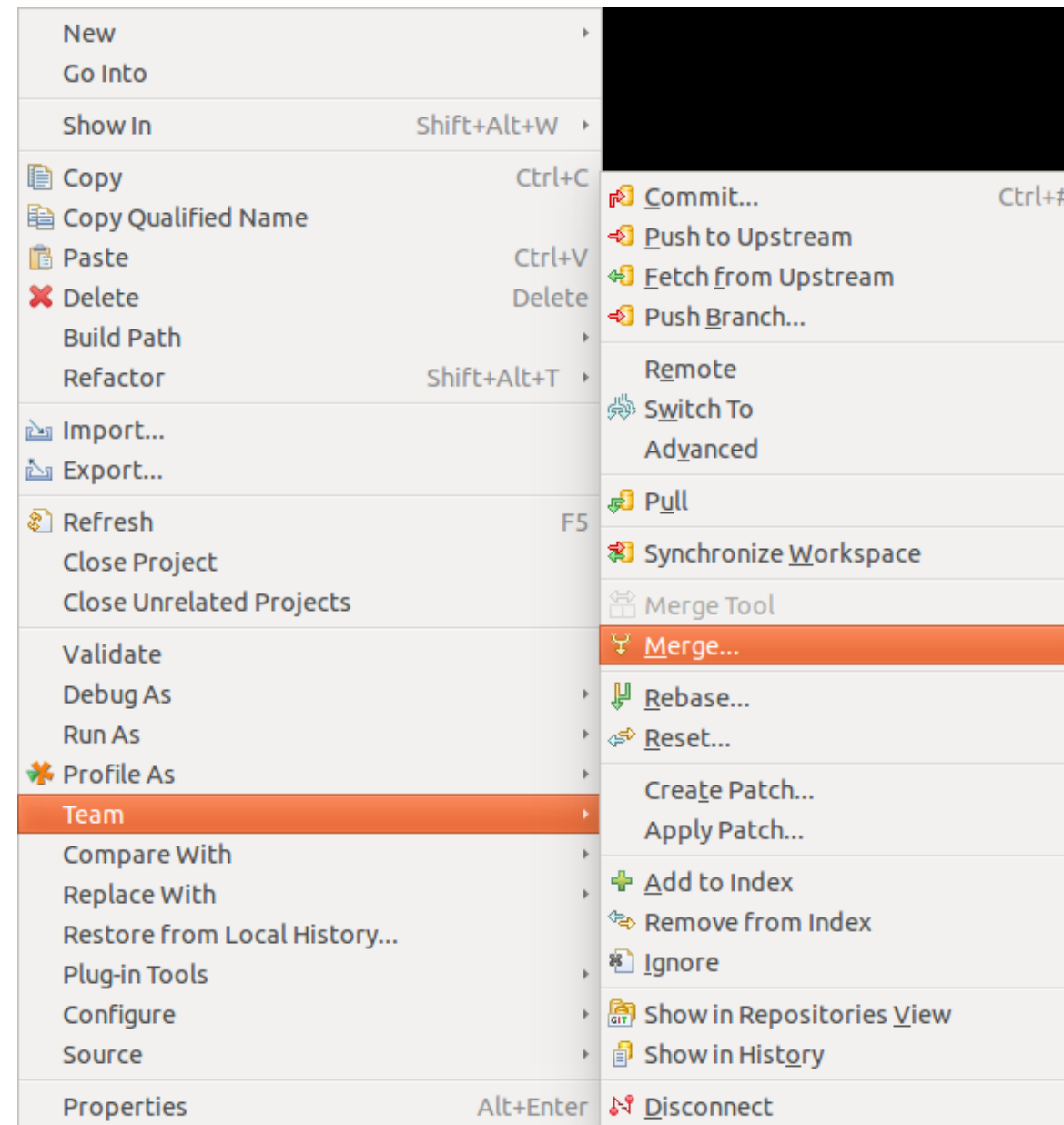
Git amend allows to alter the most recent commit. For example, you can change the commit message or add another modification.

The screenshot shows the 'Git Staging' interface for a repository named 'egitplayground'. The interface has a top bar with 'Git Staging' and 'History' tabs, a 'Filter files' search bar, and window controls. Below the top bar, there are two main sections: 'Unstaged Changes (0)' and 'Staged Changes (0)', both of which are currently empty. To the right of these sections is the 'Commit Message' field, which contains the text 'Now a better commit message.....'. Below the commit message field are two input fields for 'Author:' and 'Committer:', both containing the text 'Simpli Learn<Shashank@gmail.com>'. At the bottom right of the interface are two buttons: 'Commit and Push' and 'Commit'. A red box highlights the 'Commit' button, and a blue arrow points from the text 'Click here to perform the Git amend' to this button.

Click here to perform the Git amend

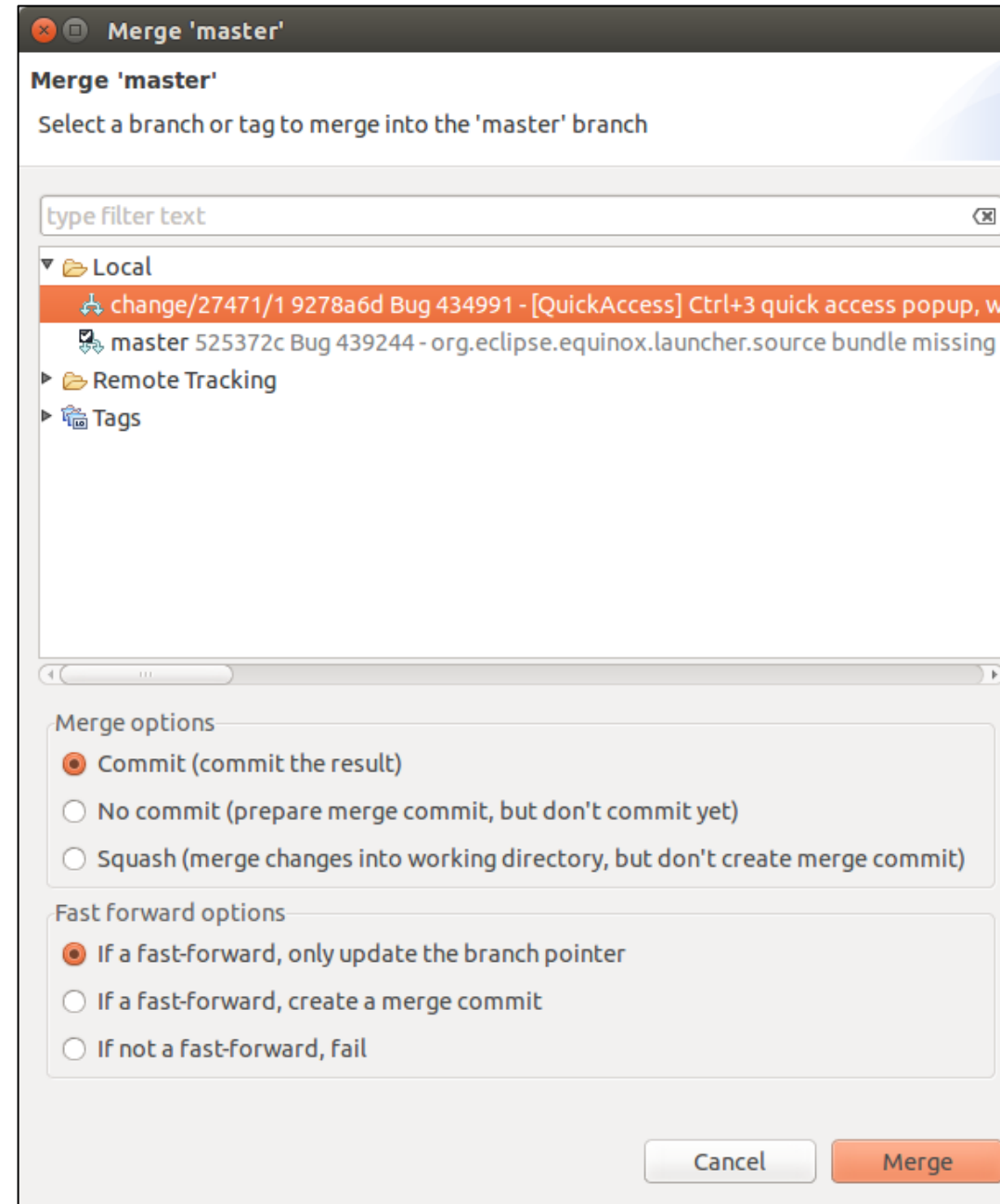
Starting a Merge Operation in Eclipse

Choose the branch into which you want to merge the changes into, select your project, and click on **Team>Merge** to start the merge dialog.



Starting a Merge Operation in Eclipse

Select a branch or tag to merge into the master branch.



Assisted Practice

Merging Files Using EGit

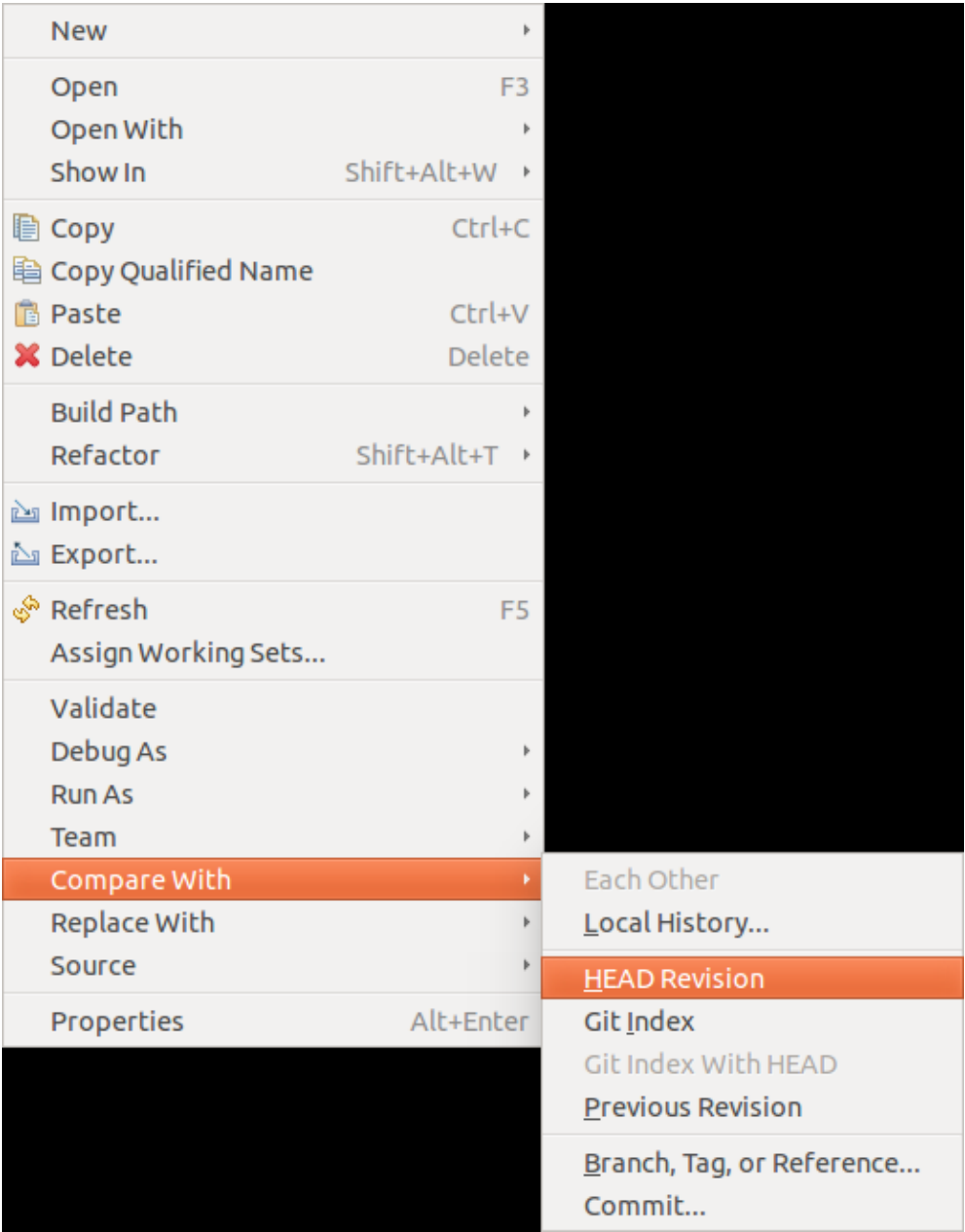
Problem Statement: During a Git operation, two changes are conflicting, you have to solve these conflicts manually before the final merge.

Steps to Perform:

1. Cloning a repository in your Eclipse Project
2. Creating and committing a file in the Eclipse Project
3. Pushing the file to the remote repository
4. Making Changes in the file from remote repository
5. Pulling and merging from the remote repository

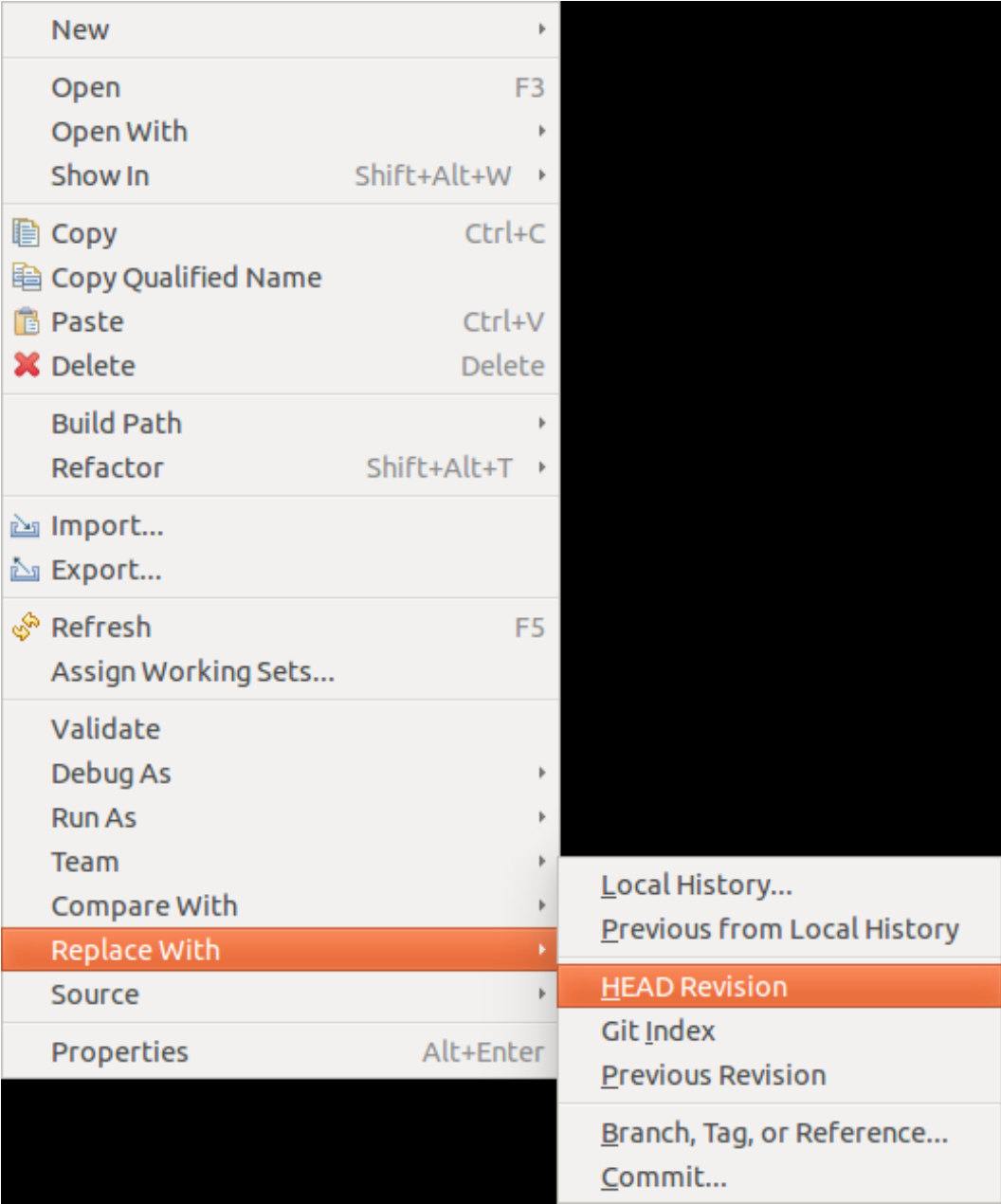
Comparing Files Based on Git History

Use **Team>Compare With** to open the menu, and select what you want to compare with.



Replacing Files Based on Git History

The **Team>Replace With** menu entry allows you to replace the current selection with the version contained in the selected commit or the Git Index.

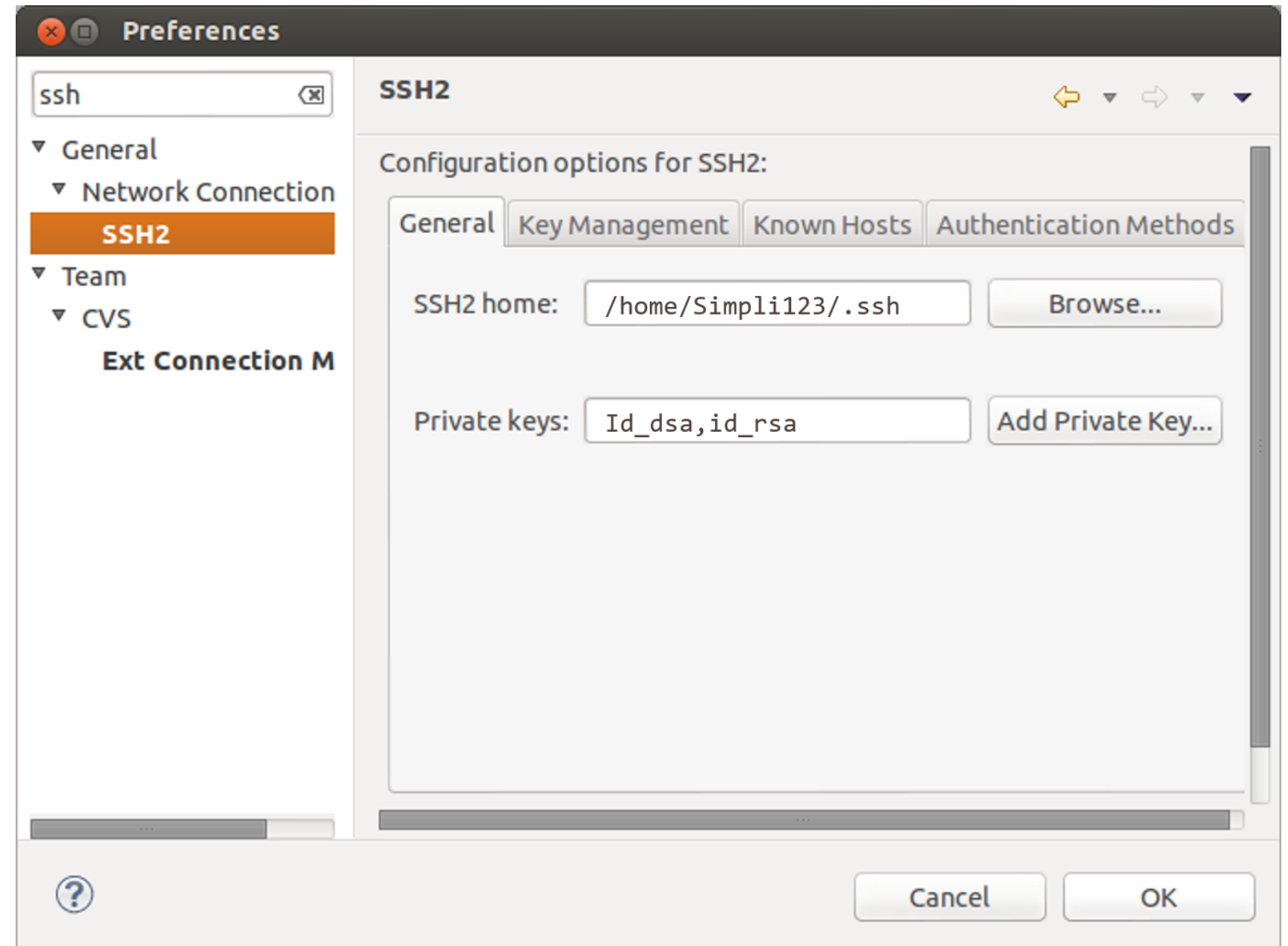


Eclipse Git with GitHub

Cloning a Project

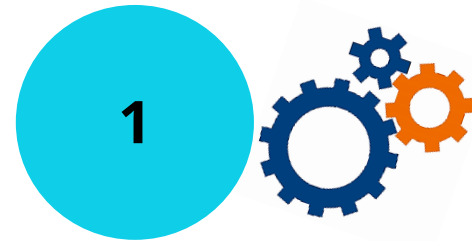
Copy the URL from GitHub and navigate to **File>Import>Git>Projects from Git**. Eclipse fills out most of the fields based on the URL in the clipboard. Enter your username and password to push the changes to GitHub.

Alternatively, you can also use an SSH key. You can configure Eclipse to know your SSH by navigating to **Window>Preferences>General>Network Connection>SSH2** preference setting.

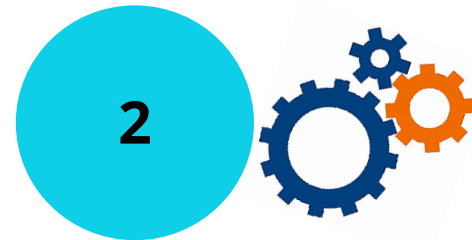


Pushing Changes

After you made changes and committed them to your local repository.
This requires write access to the GitHub repository.



Create the Java project in Eclipse.



Right-click on the project and select **Team>Push** to push your changes to your GitHub repository.

Git with IntelliJ

Installing IntelliJ IDE

Install the Toolbox App

1. Visit <https://www.jetbrains.com/idea/download/#section=linux> and download the community edition.
2. Extract the tarball to a directory that supports file execution.
3. Execute the idea.sh script from the extracted directory to run IntelliJ IDEA.
4. On the Welcome screen, click Configure | Create Desktop Entry, to create a desktop entry.

Create a Project

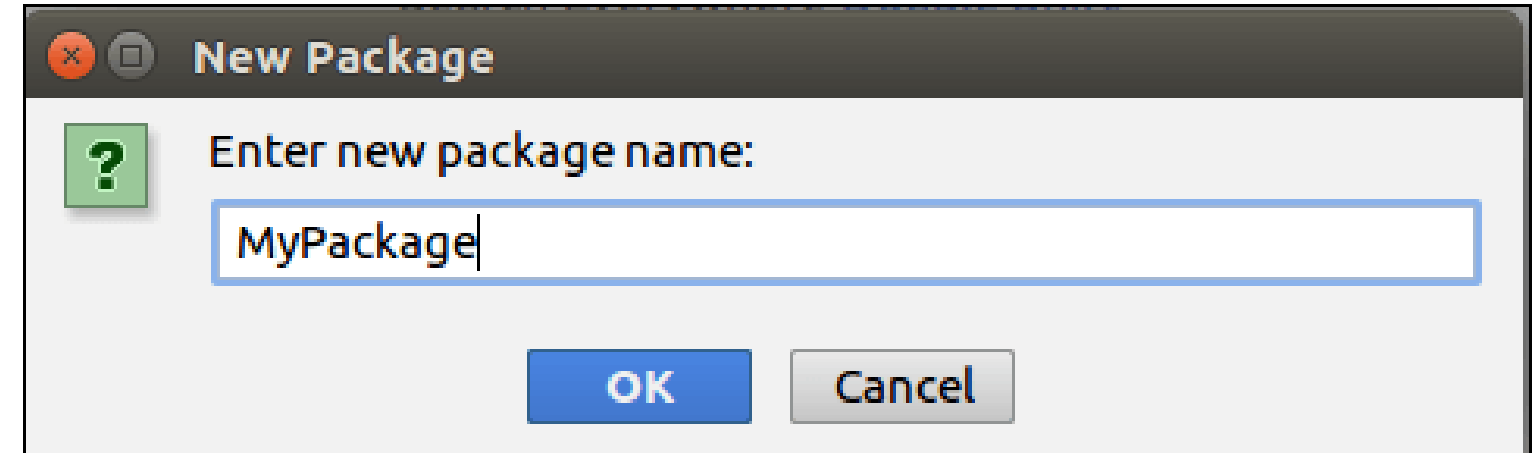
1. Start IntelliJ IDE
2. Click **Create New Project**
3. Select the **Java Project** and **Project SDK**, and click **Next**
4. Type the **Project name** and select **Project location**. Click **Finish**.



Create a Package and a Java Class

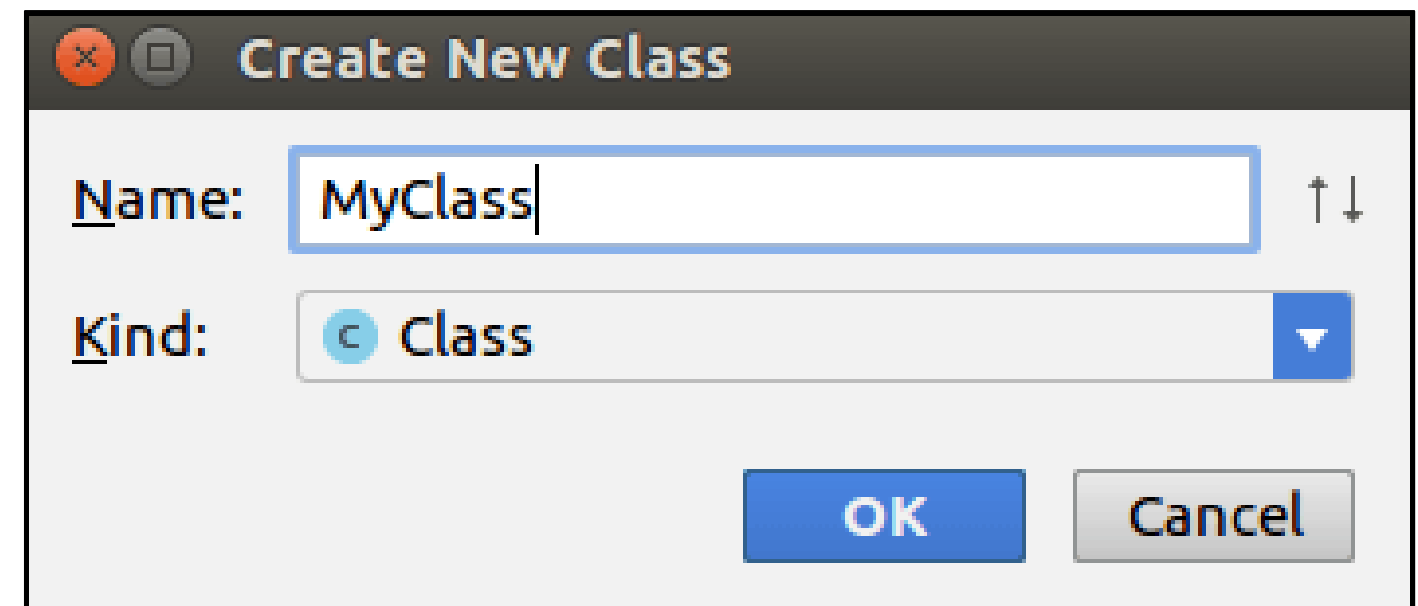
Create a Package:

1. Go to **Project Structure**.
2. Right-click and navigate to **src>New> Package**.
3. Enter the package name, and click **OK**.



Create a Java Class:

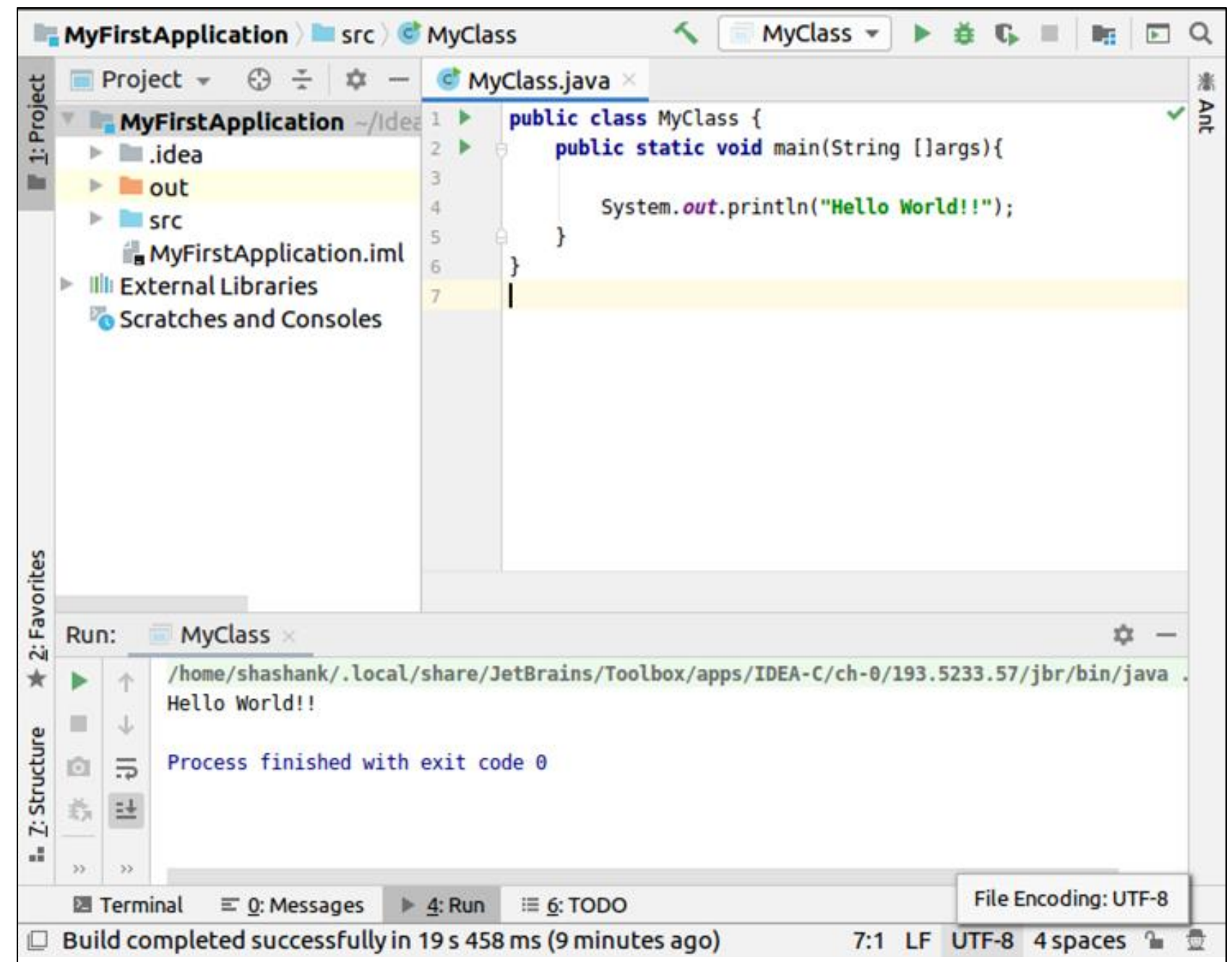
1. Go to **Project Structure**.
2. Right-click and navigate to **src>New> Java Class**.
3. Create a new class name, and click **OK**.



Run a Java Application

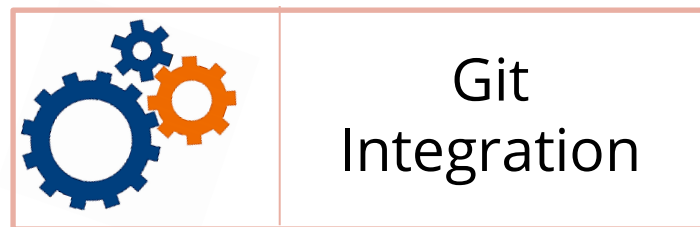
Steps to run a Java application:

1. Write the code in the editor window.
 2. Select the class name and click **Run**.
- The output will appear on the **console**.



Git Integration

Git integration can be performed by the following methods:



01

Sign up on GitHub and create a new repository.

02

Open IntelliJ IDE. Go to **VCS>Enable Version Control Integration.**

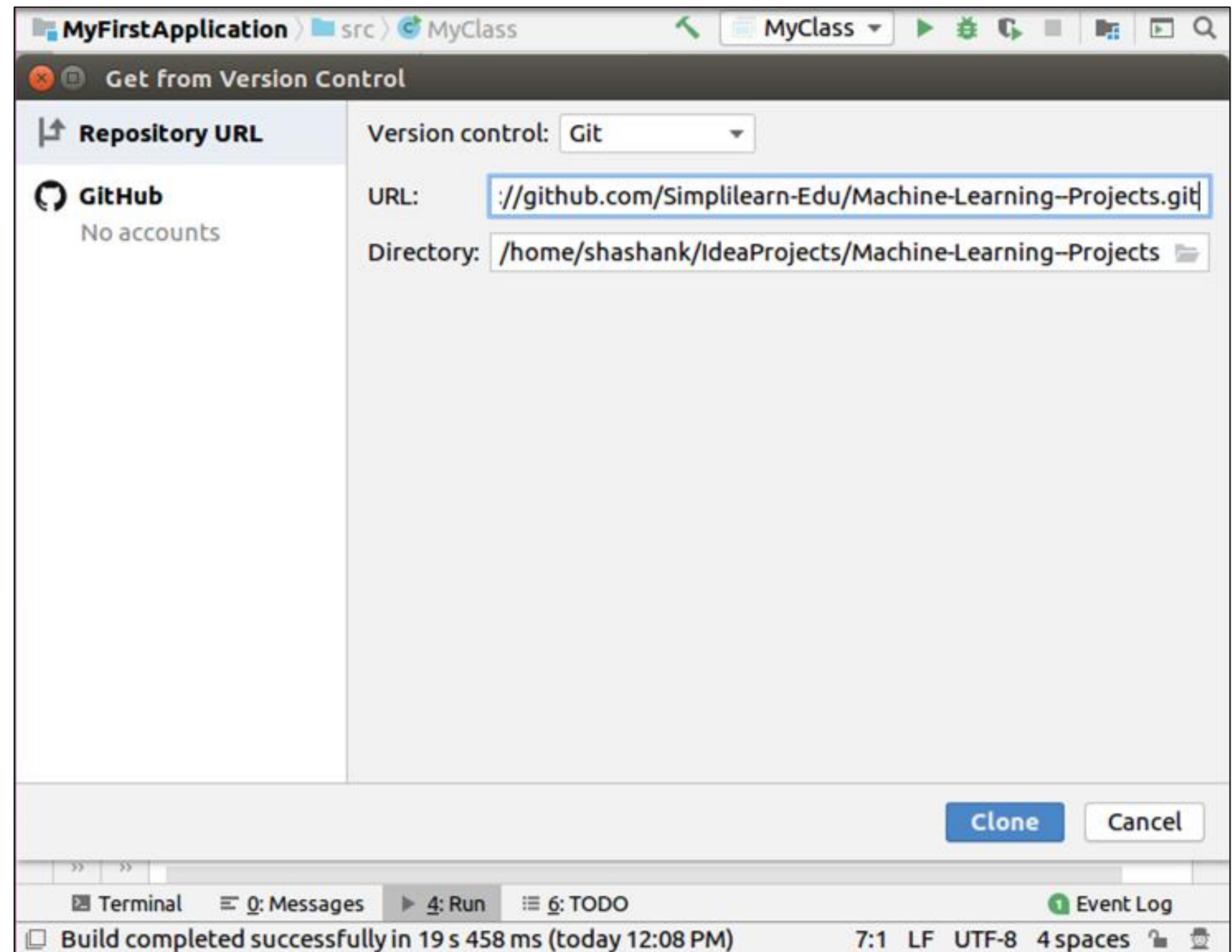
03

Select version control system to associate with the project root and select Git in drop down menu.

Git Clone

Steps to clone a Git repository:

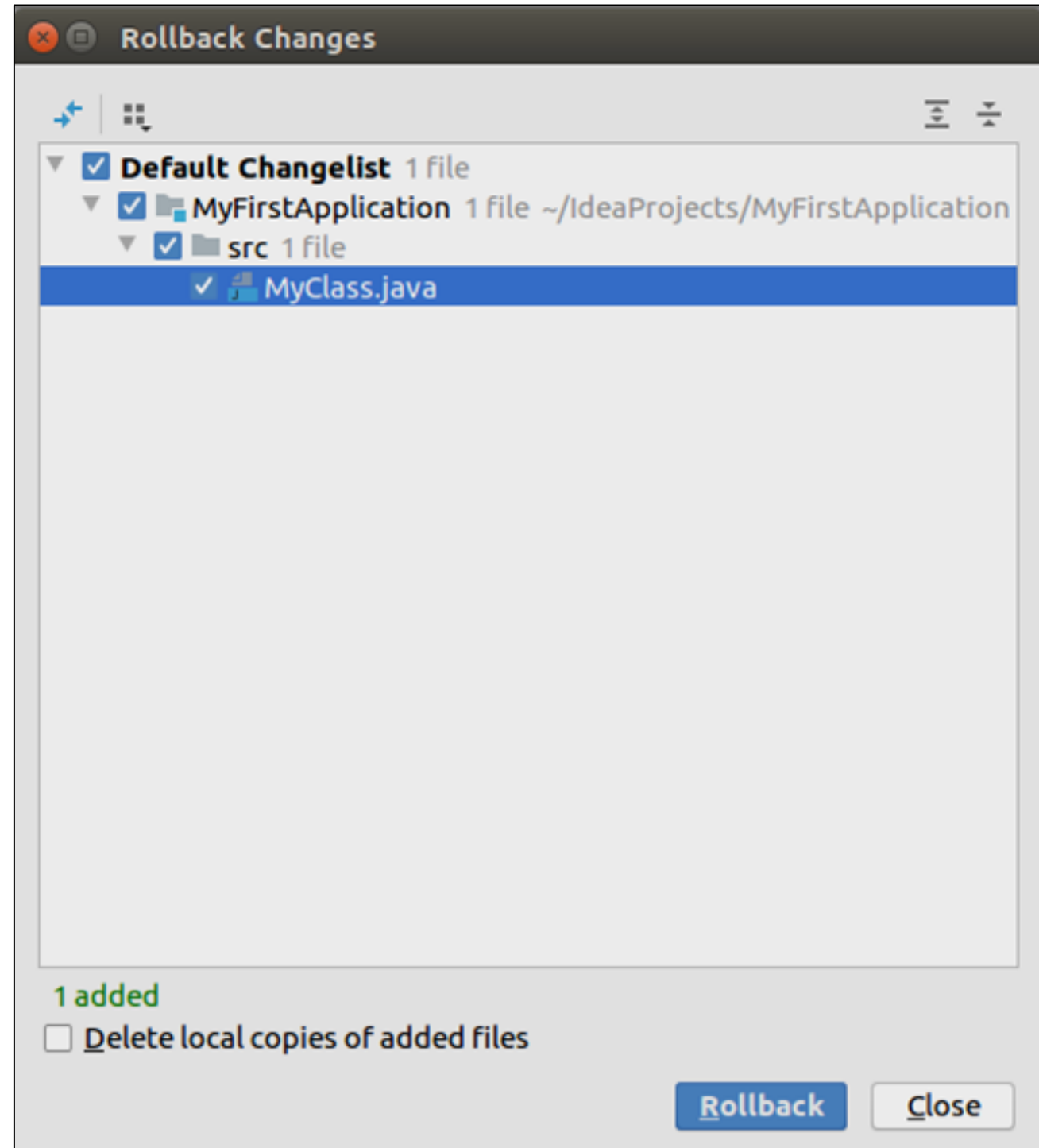
1. Go to **File>New>Project from Version Control>Git**.
2. Enter the repository URL, directory name, and click **Clone**.



Rollback Local Changes

Steps to rollback local changes:

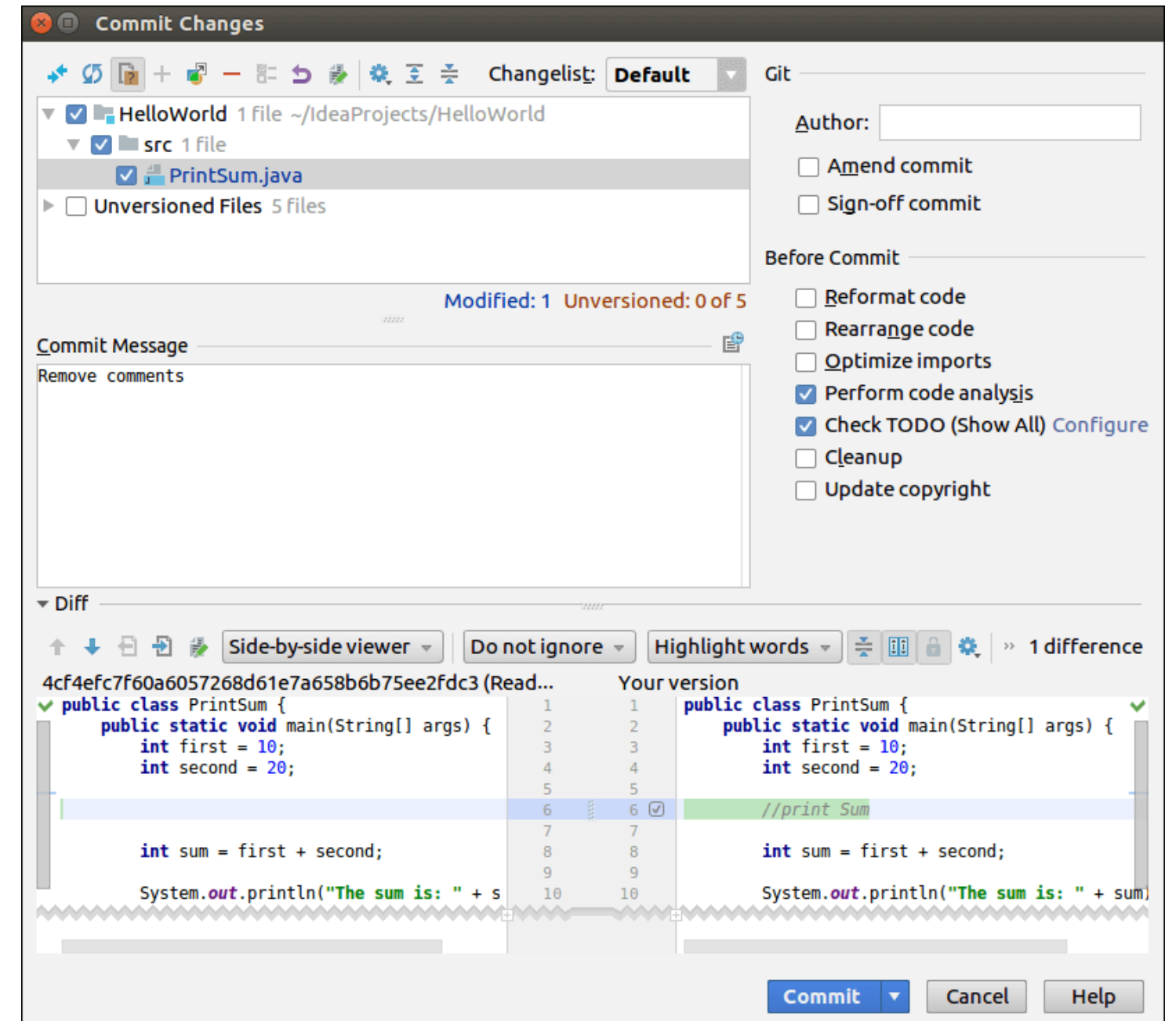
1. Go to **Build>VCS>Git>Revert**.
2. Click **Rollback** in the dialog box.



Commit Changes Locally

Steps to commit changes locally:

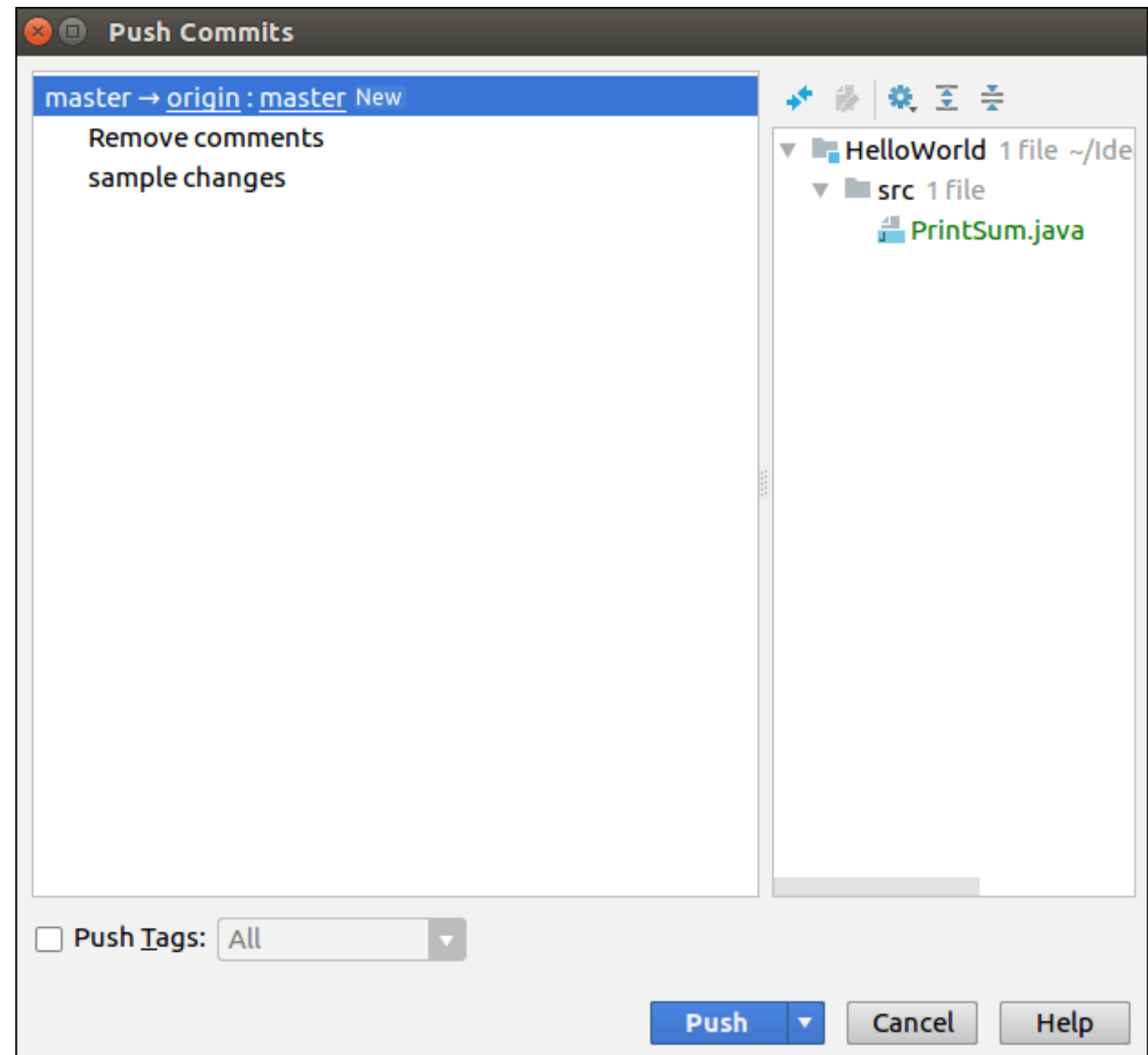
1. Select the modified file under Git version control.
2. Go to **VCS->commit changes** or **VCS>Git>commit file**.
3. Select the check box of the files to be committed in the dialog box.
4. Enter the **commit message** and click **Commit**.



Push Changes to Remote Repository

Steps to push changes to remote repository:

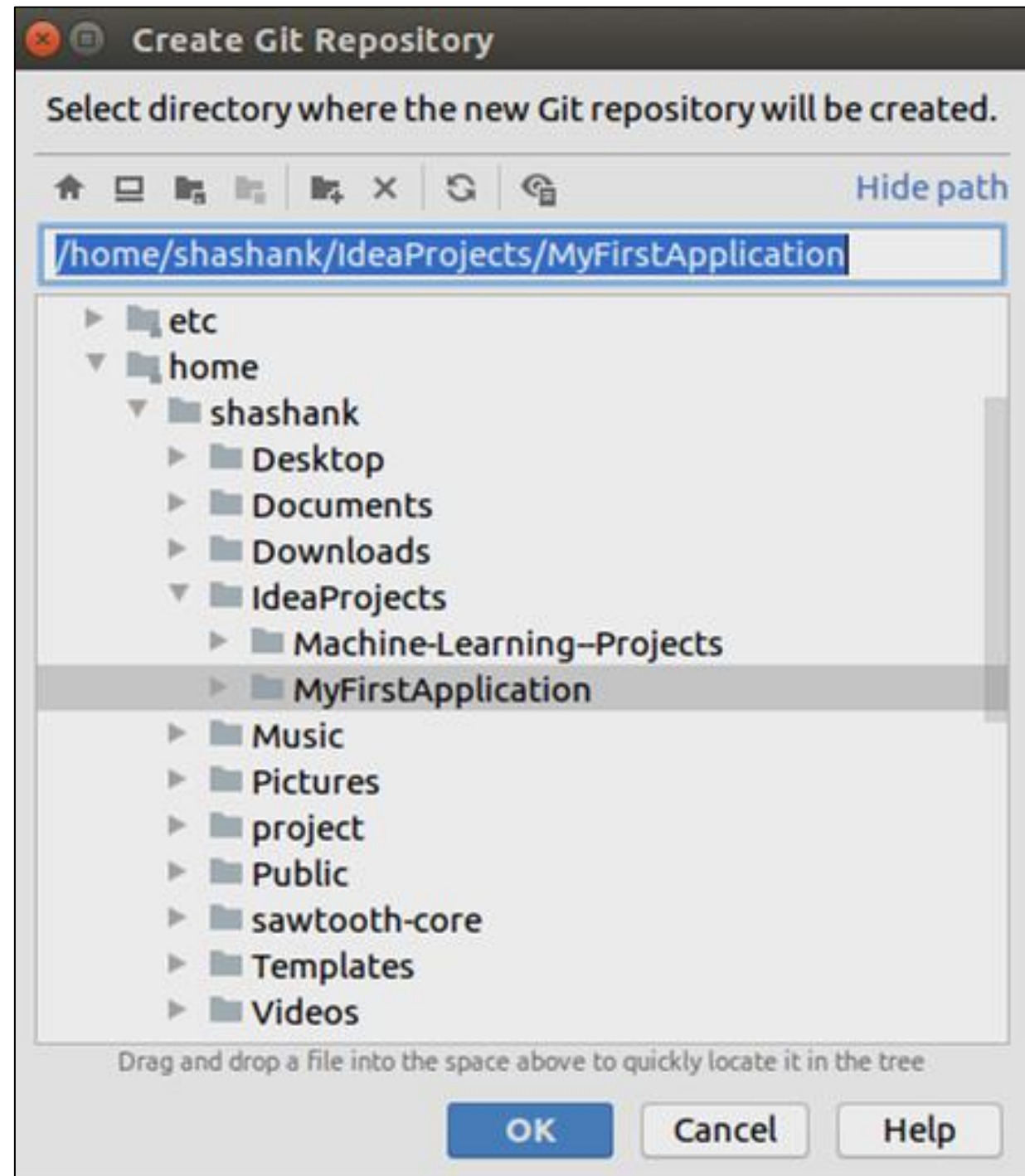
1. Go to **VCS>Git>Push**.
2. Select the commit and click **Push**.



Import Existing Project to Git

Steps to import an existing project to Git:

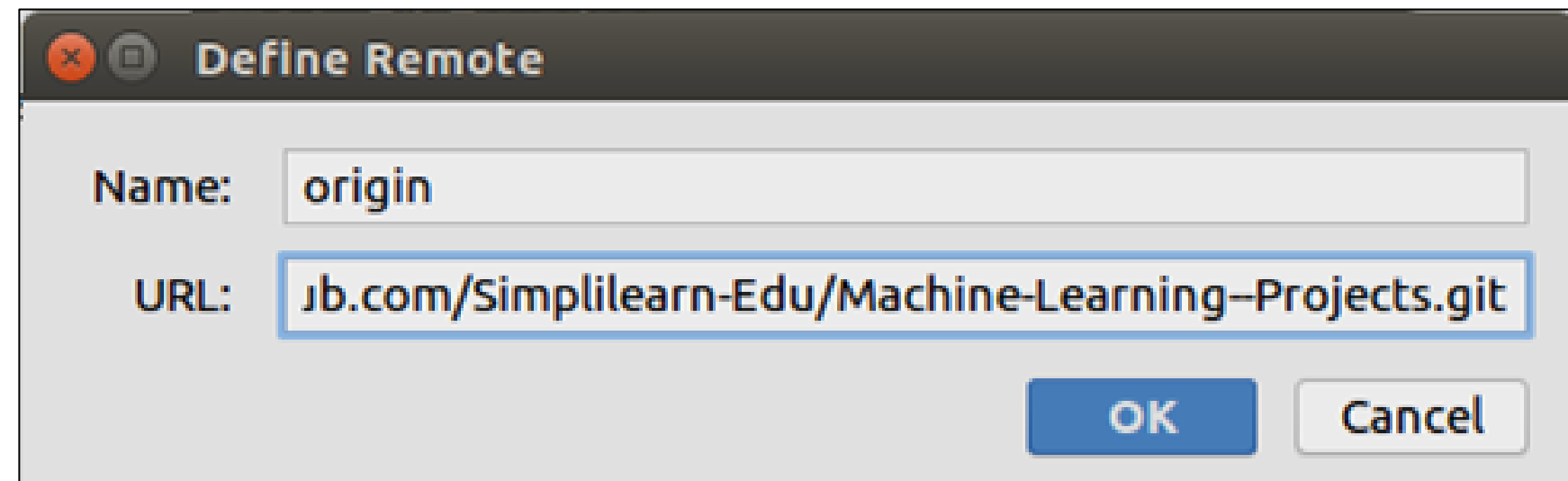
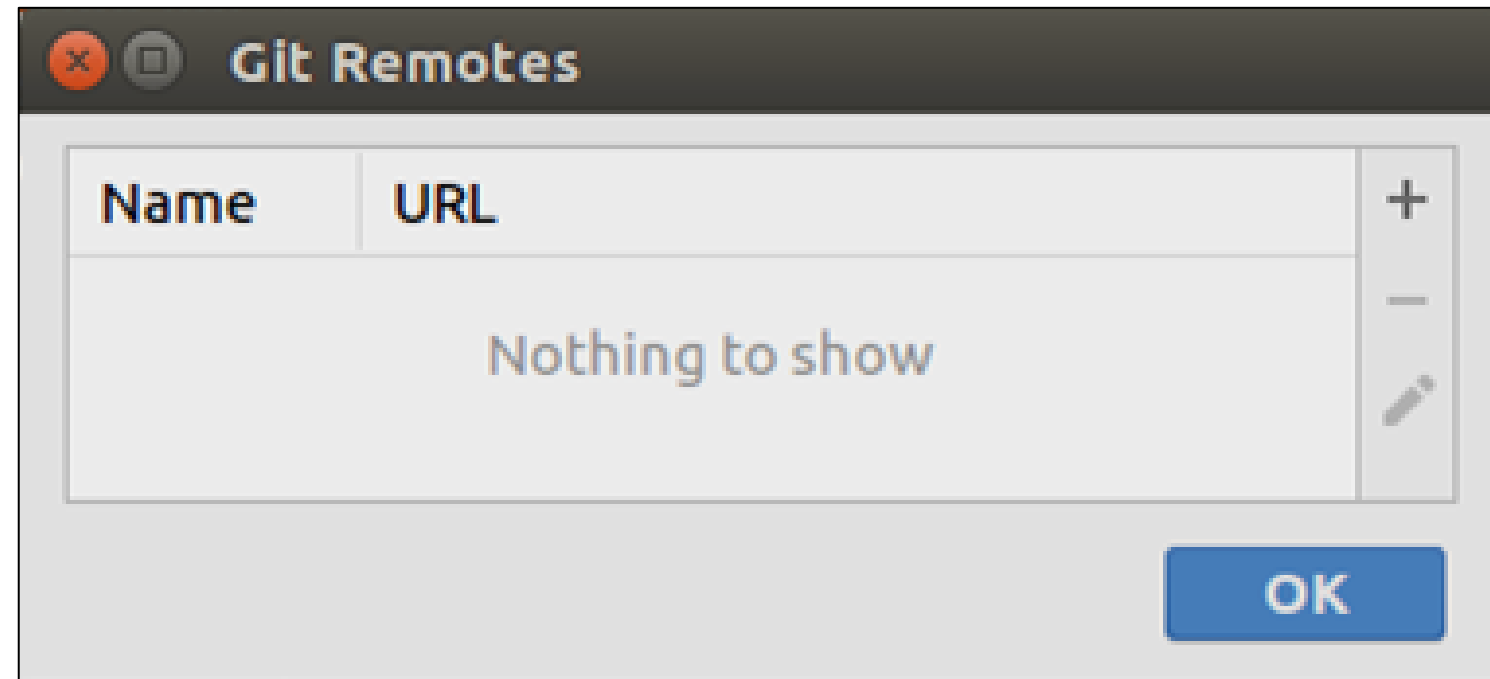
1. Go to **VCS>Import into Version Control>Create Git Repository**.
2. Select the project and click **OK**.



Define Remote

Steps to define remote:

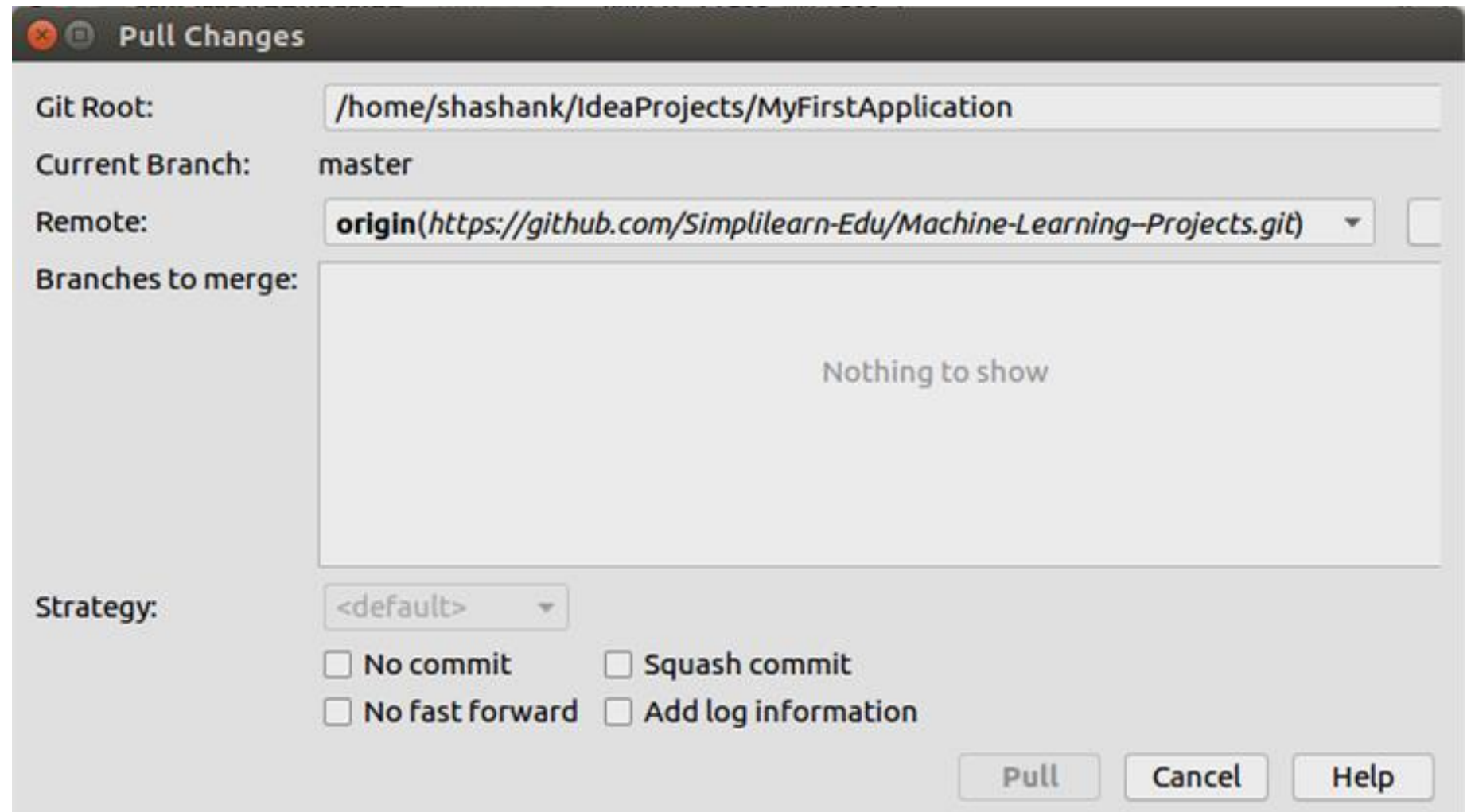
1. Go to **Build>VCS>Git >Remotes**.
2. Click **+** icon to add URL.
3. Paste the URL and click **OK**.



Get Updates

Steps to receive updates from the repository:

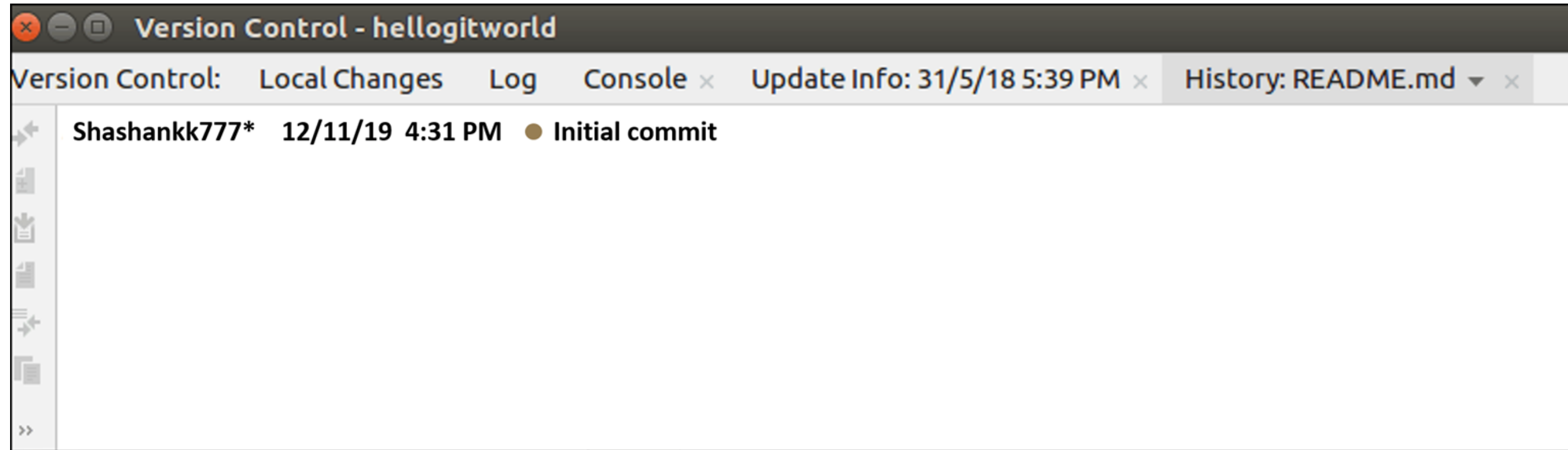
1. Go to **Build>VCS>Git >Pull**.
2. Select the required option.
3. Click **Pull**.



Show History or Log

Steps to show history or log::

- Go to **Build>VCS>Git >History**.

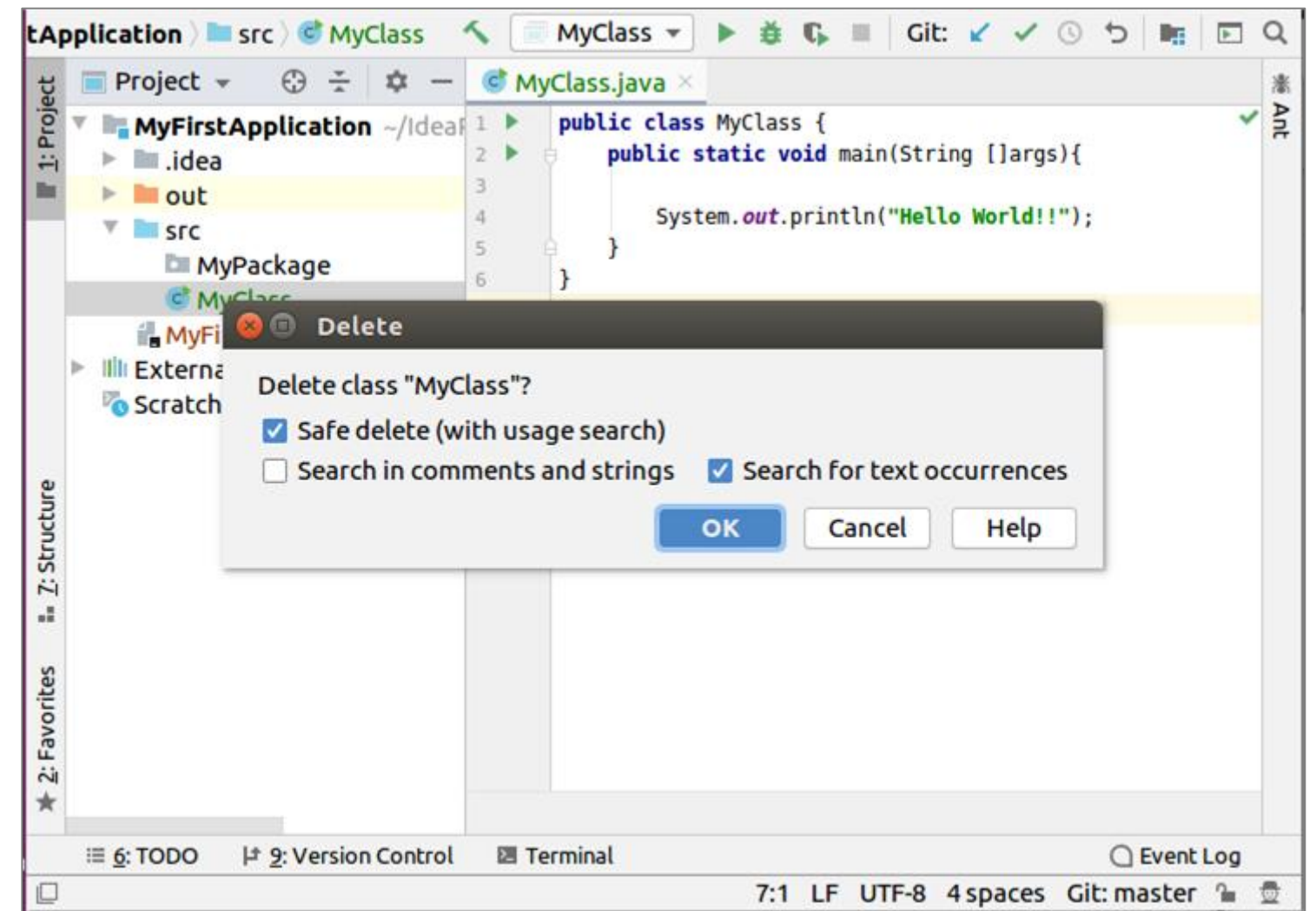


Delete Files from the Repository

A file deleted under version control still exists in the repository until the changes are committed. The deleted file is placed to the active changelist and is highlighted in grey.


Steps to delete files from the repository :

1. Select a file in the **Project** tool window and press Delete, or choose **Delete** from the context menu.
2. In the dialog that opens, choose whether you want to delete this file without searching for usages, or to perform *safe delete* to ensure that you are deleting an unused file by checking the **Safe delete** option.
3. Commit the changes to the repository.



Comparing Nodes and File Versions

Comparing a modified file with its repository version:

Select a file in the Local Changes tab of the Version Control tool window and click  on the toolbar, or press Ctrl+D.

Comparing the current revision of a file with another branch:

1. Select a file in the Project tool window and choose <your_VCS>. Select **Compare With Branch** from the context menu.
2. Choose the branch you want to compare the current file version with from the dialog box.

Comparing Nodes and File Versions

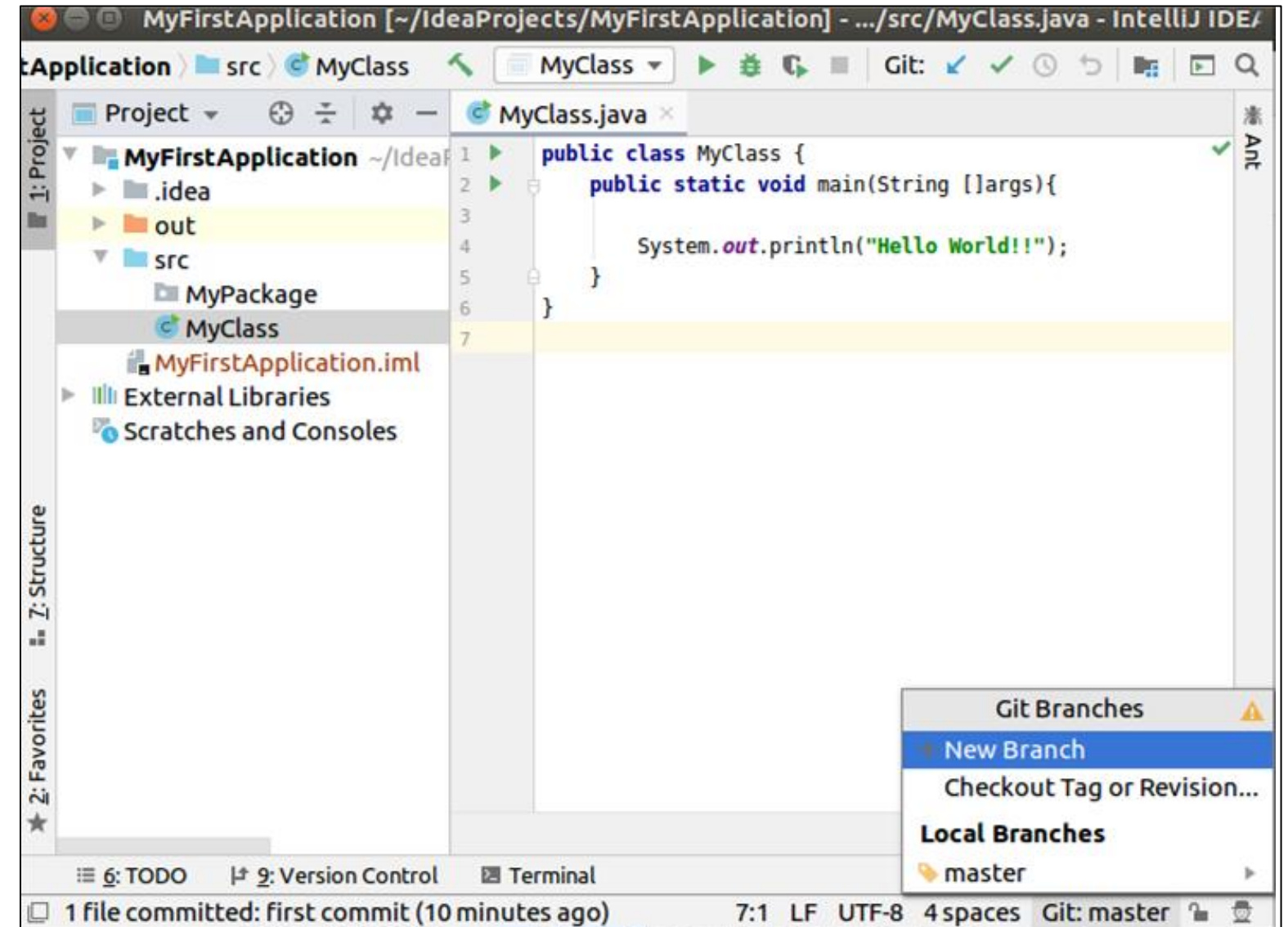
Comparing the current revision of a file with a selected revision in the same branch:

1. Select a file in the Project tool window and choose <your_VCS>. Select **Compare With** from the context menu.
2. Choose a revision you want to compare the current file version with from the dialog box.

Git Branches

Steps to invoke Git branches popup:

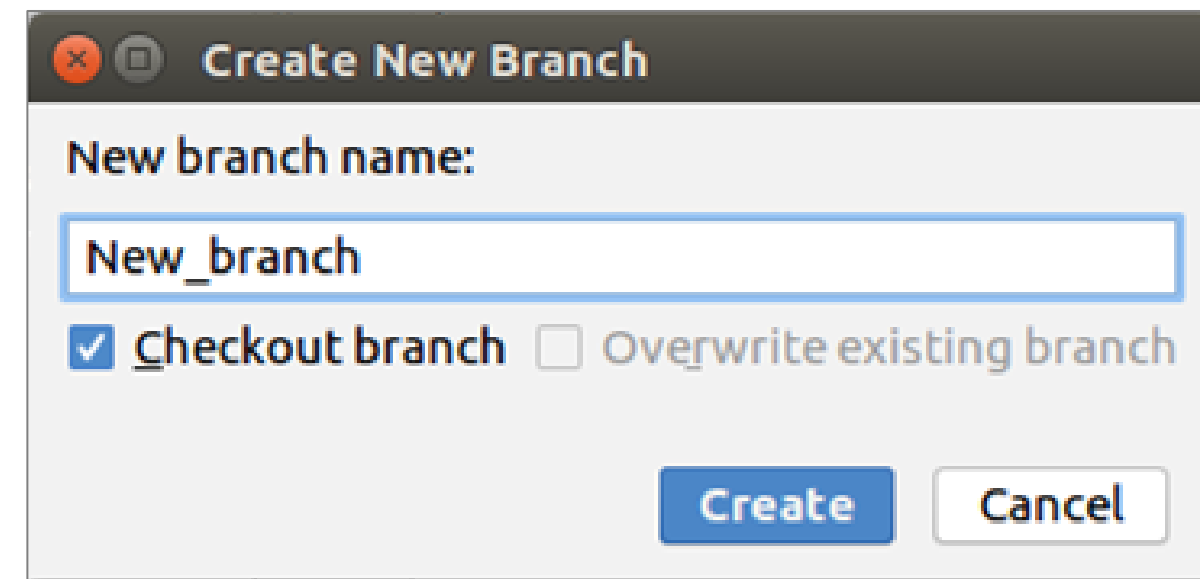
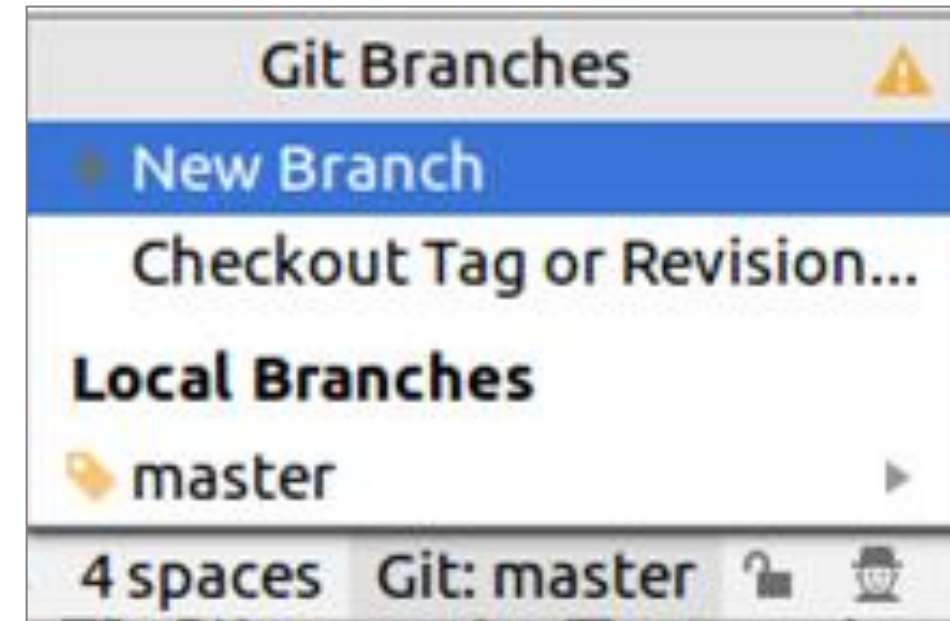
1. Click the Git widget in the Status bar and select **Branches**.
2. Toggle the **Show Only Favorites** and the **Show More** commands at the bottom of the **Git Branches** popup.



Create a New Branch

Steps to create a new branch:

1. In the **Git Branches popup**, choose **New Branch**.
2. Specify the branch name, and make sure the **Checkout branch** option is selected if you want to switch to that branch.



Undo Changes

Steps to revert uncommitted changes:

1. Open the **Version Control** tool window and switch to the **Local Changes** tab.
2. Select one or more files that you want to revert and select **Revert** from the context menu.

Steps to revert a pushed commit:

1. Locate the commit you want to revert in the **Log** view, right-click it and select **Revert** from the context menu.
2. Click **Commit**.

Key Takeaways

- EGit is an Eclipse plugin which allows you to use the distributed version control system “Git” directly within the Eclipse IDE.
- Git pull updates remote tracking branches, merges changes into the working branch of the local repository, and reports conflicts.
- Conflicts must be added to the Git Index before the conflicted files can be committed.



Lesson-End Project

Push an Eclipse Project into GitHub

Problem statement: You have created a Java project in your local directory that needs to be shared with the test engineering team for verification. Push the project into a newly created GitHub repository for sharing the code and tracking changes.

You must use the following:

- Git: To create and work with a local Git repository
- Eclipse IDE: To create a Java project
- GitHub: To create a remote repository

