

Introduction to GIT

Git is a **Version Control System (VCS)**, a tool which is an extremely smart choice to use even if it sounds too overwhelming. Simply put, **a VCS is a group of files with monitored access**. What does that mean? Let's consider a small example to help you understand.

If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a VCS is a very wise thing to use. It allows you to revert files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover. In addition, you get all this for very little overhead.

Many people's version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if they're clever). This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.

What is Version Control System

Version control refers to the practice of tracking and managing changes to software code, and Version Control Systems refers to software tools that assist software teams in managing changes to source code over time.

In layman's terms, Version Control means that if you rip off the project or accidentally delete files, you can quickly recover them, and the overhead is negligible.

Say, you are adding a new feature to your million-dollar app. You spend sleepless nights writing code for it. And before you can go back in time, you realize that new bugs have crashed the production server.

Now what?

Frantically pressing Ctrl+Z on your keyboard to hopefully return to a previous version of the working code?

That's not going to help always.

So, how do you get out of this Ctrl + z / Ctrl + y loop?

VCS - Version Control System is the answer you are looking for.

Why would you care to add another sword to your software armory?

VCS can not only restore your previous working version, but it can also solve other concerns with code management and application deployment. You cannot avoid learning VCS to manage and make your code production-ready in the current DevOps cultural shift.

What Is Git?

GIT – initially developed by Linus Torvalds is a version control system that is used to manage the source code of a software application.

It facilitates many developers to work collectively at the same time, on the same file without providing any hindrance to the others.

What Is GitHub?

Let us consider a situation for Example. Suppose you are working in a team to develop a software application and two members of a team are working on a specific module. Let us consider them as A and B.

For that module, the developer A will have a copy in his PC and developer B will have a copy on his PC. Now at this point, if developer B changes the code in the module, then developer A wouldn't know the changes made by the developer B and vice versa.

To overcome this problem, we have a platform which is called GitHub. Github is a Web hosting service where the source code of a software application can be uploaded/saved through the git software (installed on your local PC) or you can directly copy the code from your PC and save on the server.

Features of GitHub

Enlisted below are the various features of GitHub.

- **Distributed:** GitHub provides a distributed network, which means it provides a backup of the code. Thus if in case the central server crashes, the coder has its copy in the local repository. It saves each version or each copy of the changed code.
- **Compatible:** Suppose if you are using any other version control system like SVN and you want to switch to GitHub, then you can easily do that without creating new code again.
- **Branching:** Branching is a unique feature provided by GitHub. The developer needs to pick a section of the code from the remote branch, in that the branch developer can make changes, merge or delete codes within a few seconds.
- **Secure:** GitHub uses security features of SHA1(encrypted hexadecimal code) for any changes /commit made by the developers. Thus, it maintains the confidentiality of the developed project.

Basic Terminologies

Let us become familiar with the basic terminologies associated with GitHub.

1. **Pull Request:** If you have made changes in code/script, to show the other collaborators you send a pull request.
2. **Repository:** You can simply, treat it as a storage area of your workplace that contains all your documentation files and the history of changes.
3. **Fork:** It is a copy of other's repository in your account in which you can make changes and it won't affect the original code.
4. **Commit:** Whatever the changes you make in your files will come under commit. Every change is saved under a particular name or ID which is also called "revision".

5. **Branching:** When you extract a portion /section of code from the main or remote track of your software, then it is called 'branch' and the process is known as Branching.

Installation

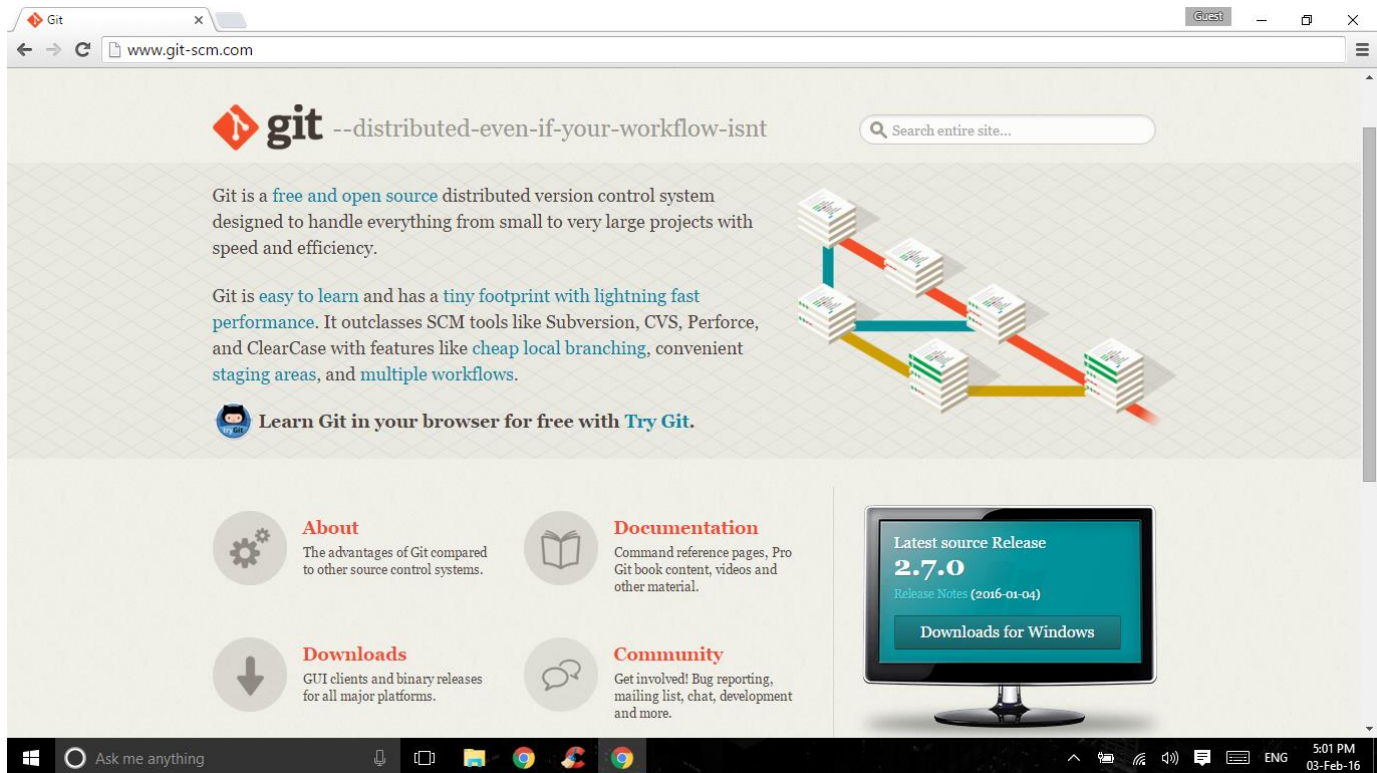
Now that we know what a **Version Control System** is, let's make sure you already have an installed version of **Git** on your computer so it's easy to follow along.

Apple

If you have a Mac, Good News! It comes on your system by default. Go right ahead to the [next chapter](#).

Windows

Head over to [this link](#). You will notice that the website recognizes your OS and prompts you to download the latest version.



Go ahead and install from the package. When you open the installer, just click **Next** and install Git with the **default settings** since they are the recommended settings. You may change the **installation directory** if you need to.

Welcome to the Git Setup Wizard

This will install Git version 2.7.0.2 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

[Next >](#)[Cancel](#)

Information

Please read the following important information before continuing.



When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

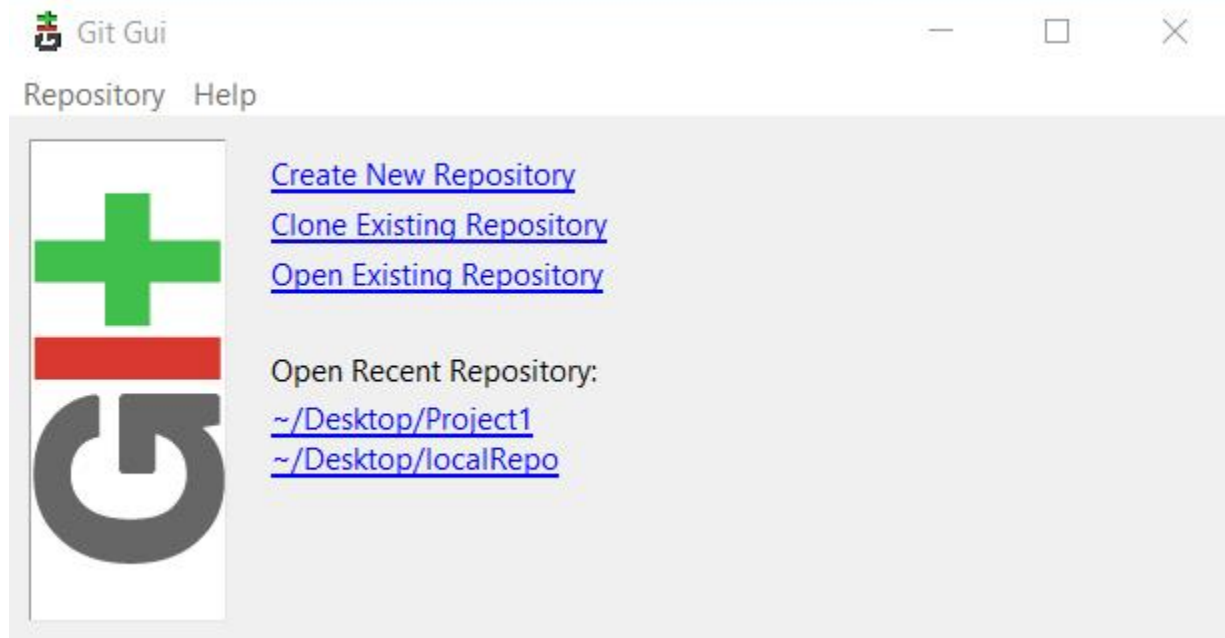
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change

<https://git-for-windows.github.io/>

[< Back](#)[Next >](#)[Cancel](#)

- Go into the **Start Menu** after the installer is complete and look for the **Git folder / icon**. You may see two options, **Git GUI** or **Git Bash**. Some people prefer to use the GUI but we are going to focus on the **command line** in this course. But feel free to check the Git GUI whenever you like.

Getting started with Git GUI



Step 1: Create Remote Repository

Now, we need a Git repository, and we'll create a new remote repository on Github.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner: iammridul503

Repository name *: test_repo_for_GitGui ✓

Great repository names are short and memorable. Need inspiration? How about ~~special~~ system?

Description (optional): Testing Repository for Git Gui

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None Add a license: None ⓘ

Create repository

Step 2: Create a Local Repository

For creating a local repository: in our Git GUI, click on “**Create New Repository**”. Select the location you wish to store your repository in. It is important to note that the selected repository location **MUST NOT** exist.

Git Gui

Repository Help

Create New Repository

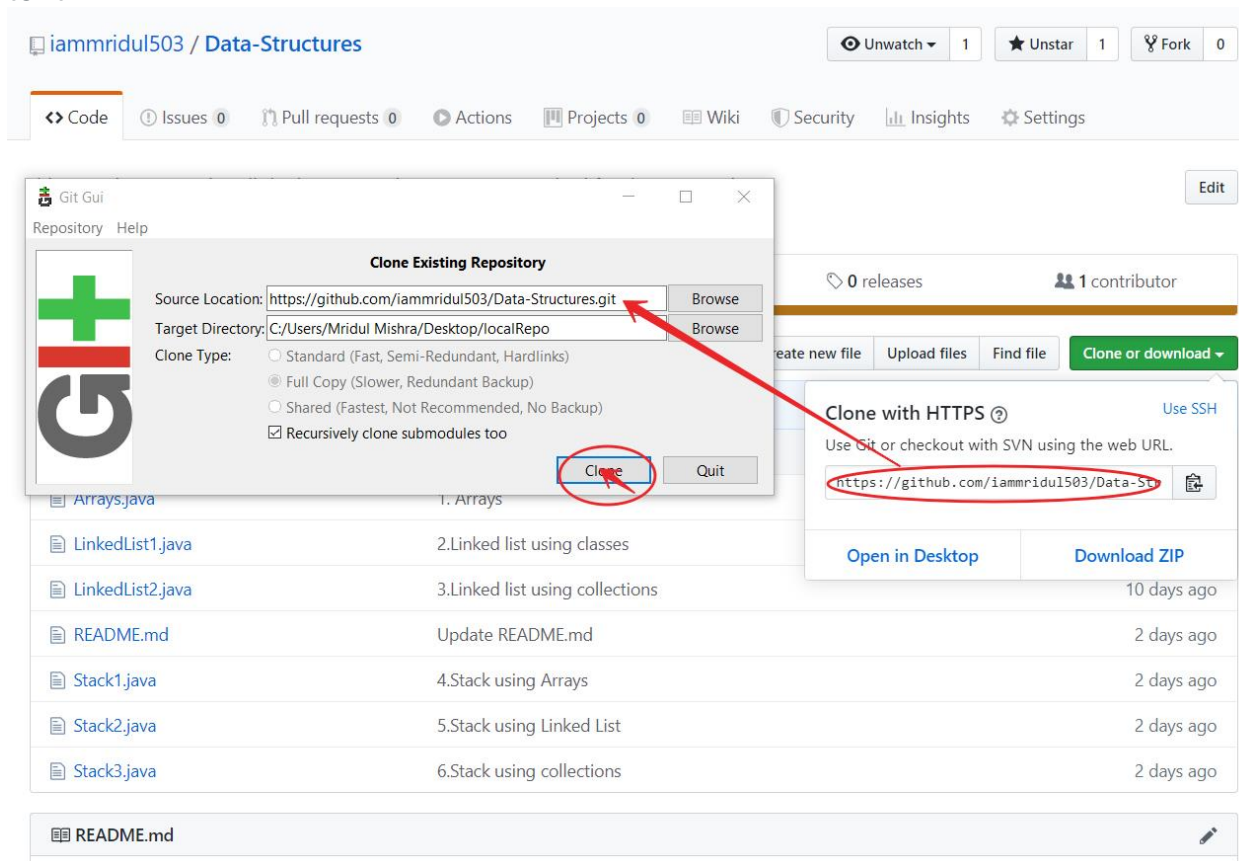
Directory: C:/Users/Mridul Mishra/Desktop/localRepo Browse

Create Quit

In order for this new repository to be initialized, you must first create a file, any file, in your local repo. Then, you must Commit and Push to the remote Git repository location.

Step 3: Clone a Remote Repository to a Local Repository

In order to clone a repository, click on the “**Clone Existing Repository**” link in the Git GUI window. An existing repository is one that is already initialized and/or has commits pushed to it.



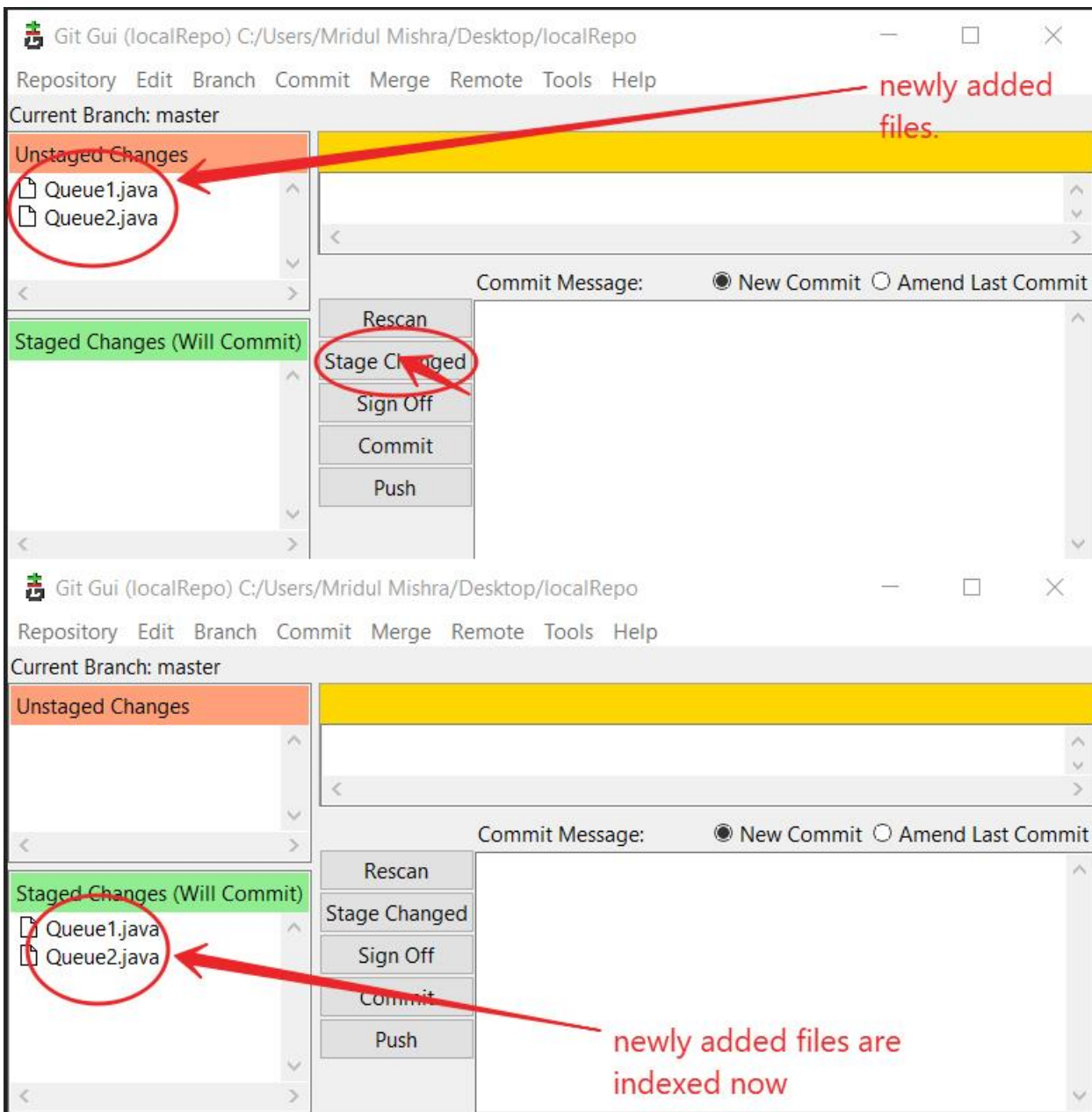
Note: In the Source Location field, fill in the Git remote repository location. The target directory works as same in the case of creating a local repository. Git will attempt to create it, and it will fail if it already exists.

Working with the GUI Client

The Git GUI makes it easier to perform Git-related tasks, such as staging changes, commits, and pushes.

Staged Changes

When we move files to a Git directory, you will see all the files in the “Unstaged Changes” window. This basically means that new files have been added, removed, updated, etc. When we click the “Stage Changed” button, it will attempt to add all the new files to the Git index.



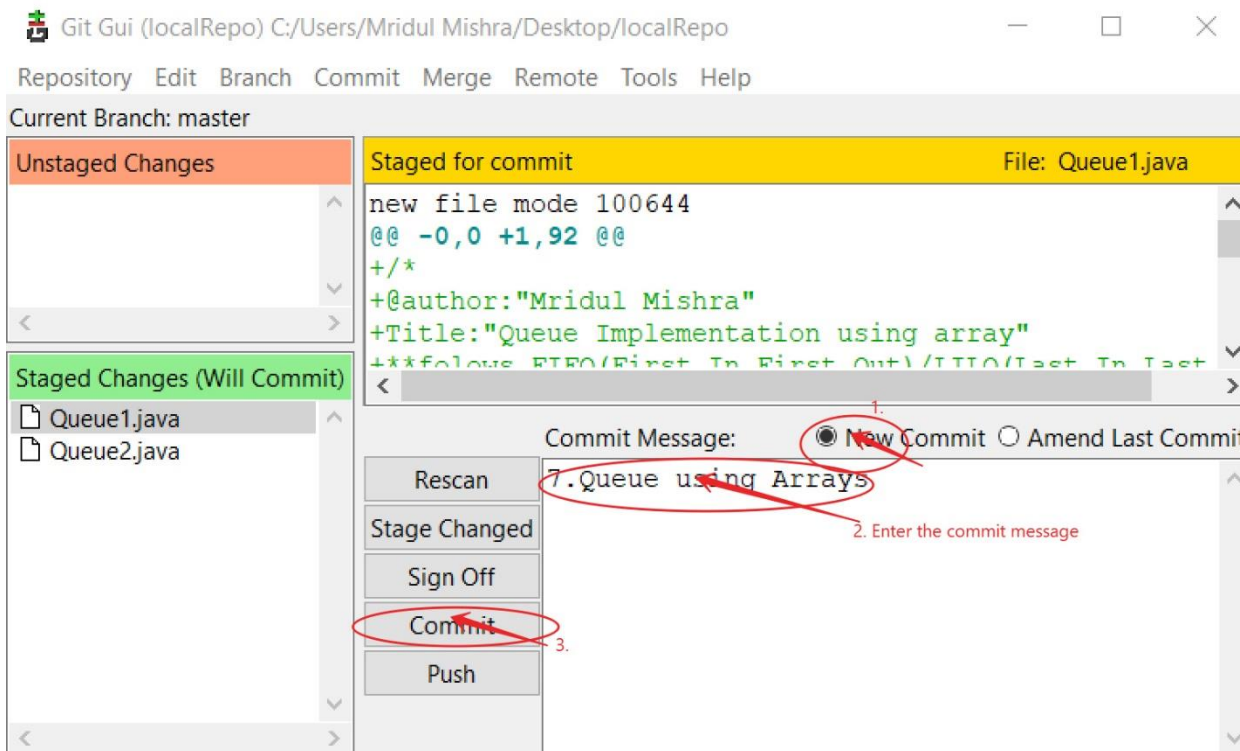
Git Equivalent Command:

```
git add file_name
```

```
git status
```

Commits

After we've staged your changes, we need to commit them to your local repository. Type a Commit Message that makes sense to the changes that were made. When we are done, press the Commit button.

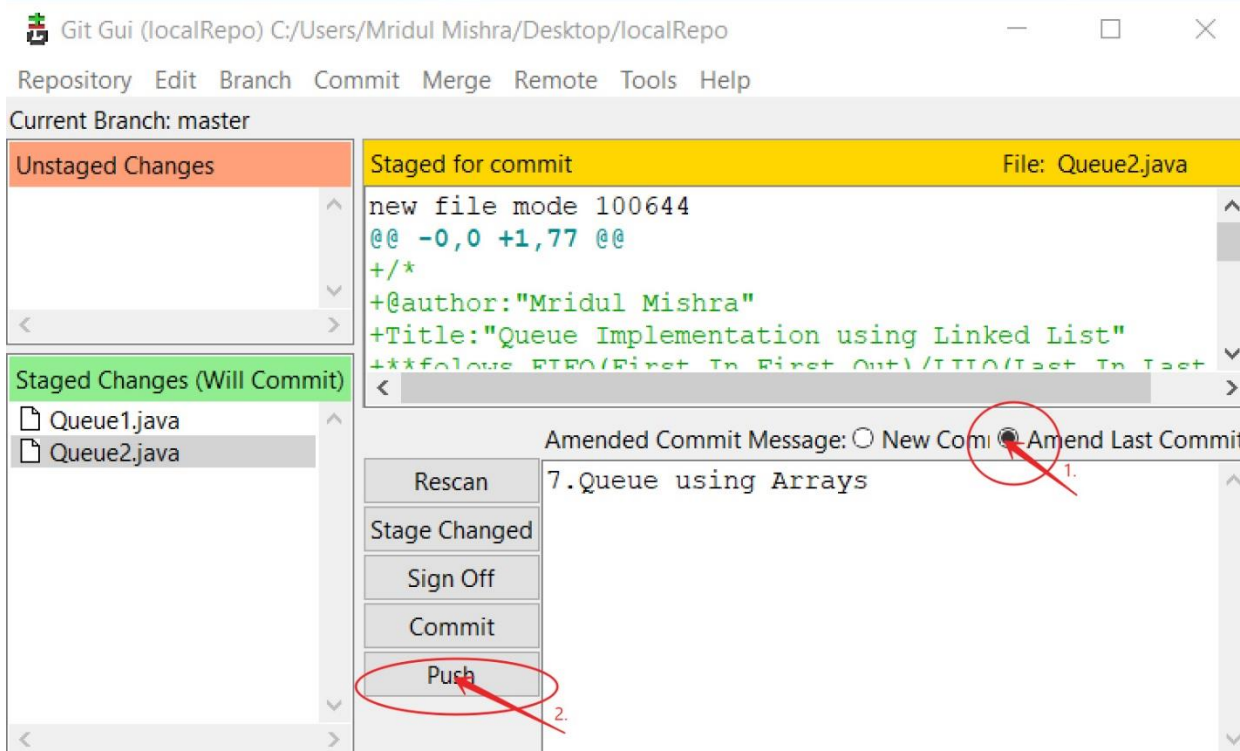


Git Equivalent Command:

`git commit -m "message"`

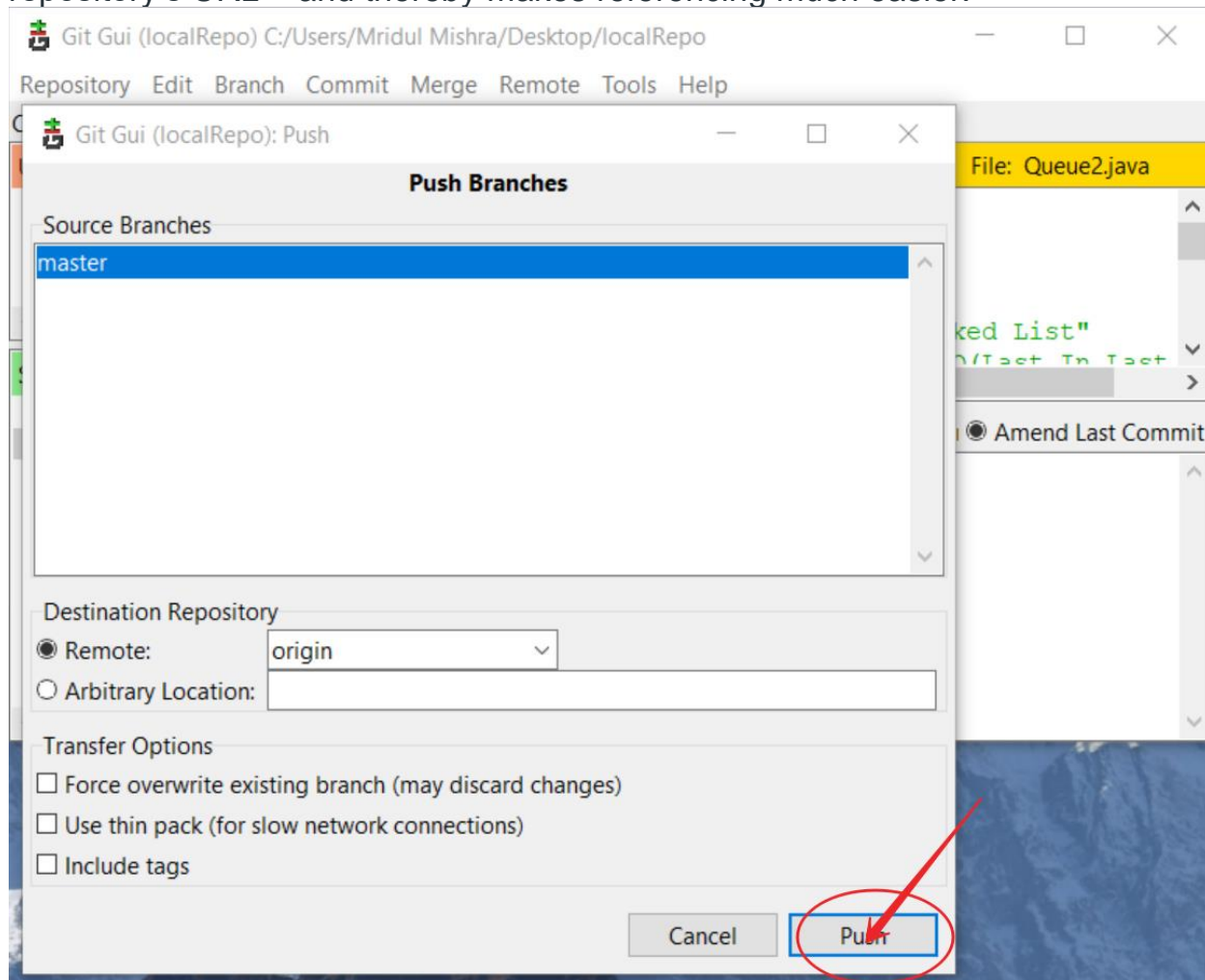
Pushing

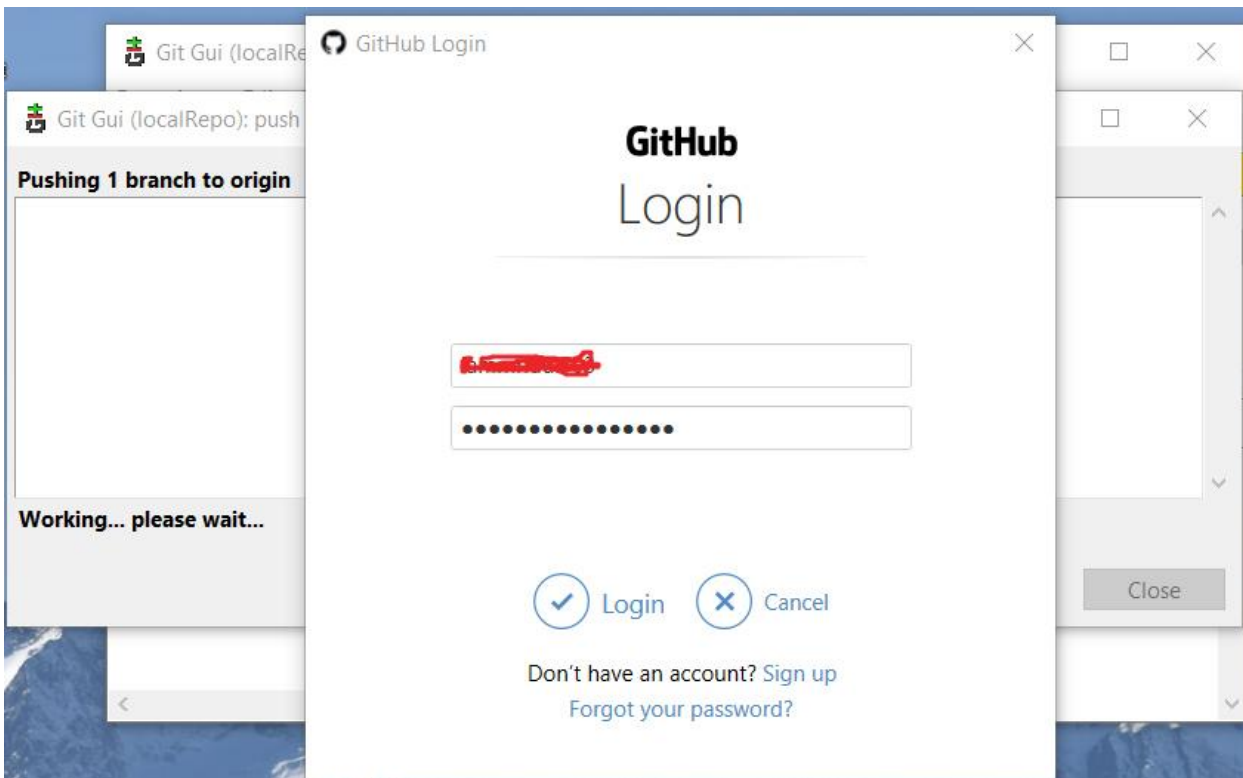
After we have committed all the codes in the local repository, we need to push these changes to our remote repository on GitHub. Without pushing the changes, others would not be able to access the code.



Before we can proceed to push, we need set up a location to push to. Most folks refer to this location as "origin". In Git, "origin" is a shorthand name for the remote repository that a project was originally cloned from. More precisely, it is used instead of that original

repository's URL – and thereby makes referencing much easier.





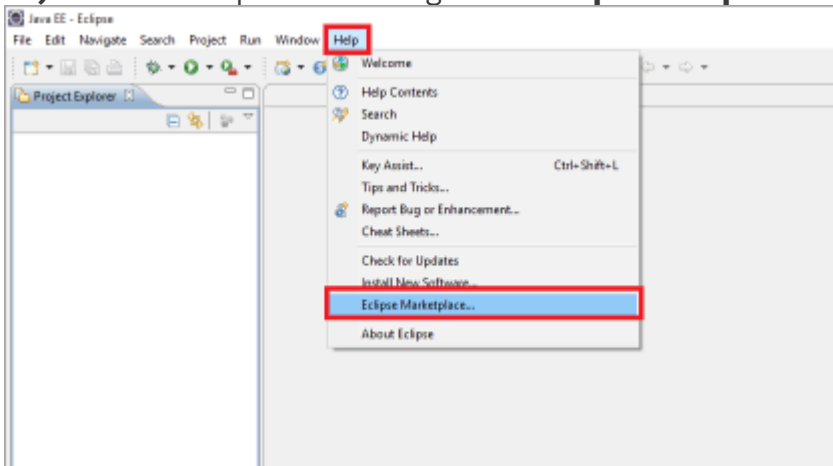
Git Equivalent Command:

```
git push -u origin master
```

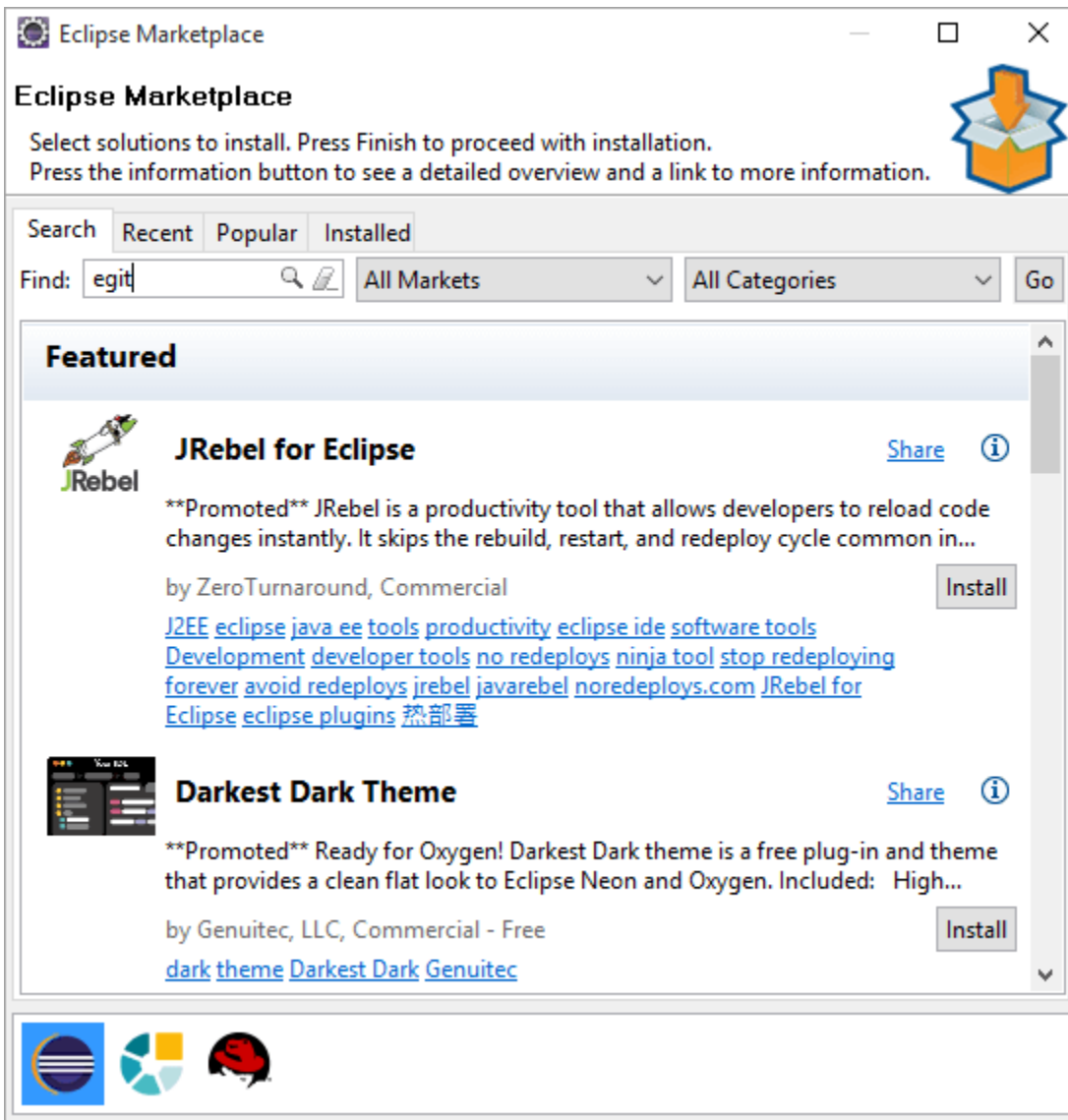
This way GUI makes it a lot easier to work with GIT for the users that do not prefer the command line.

Set Up Eclipse with Git Plugin

#1) Launch Eclipse and navigate to **Help => Eclipse Marketplace**.



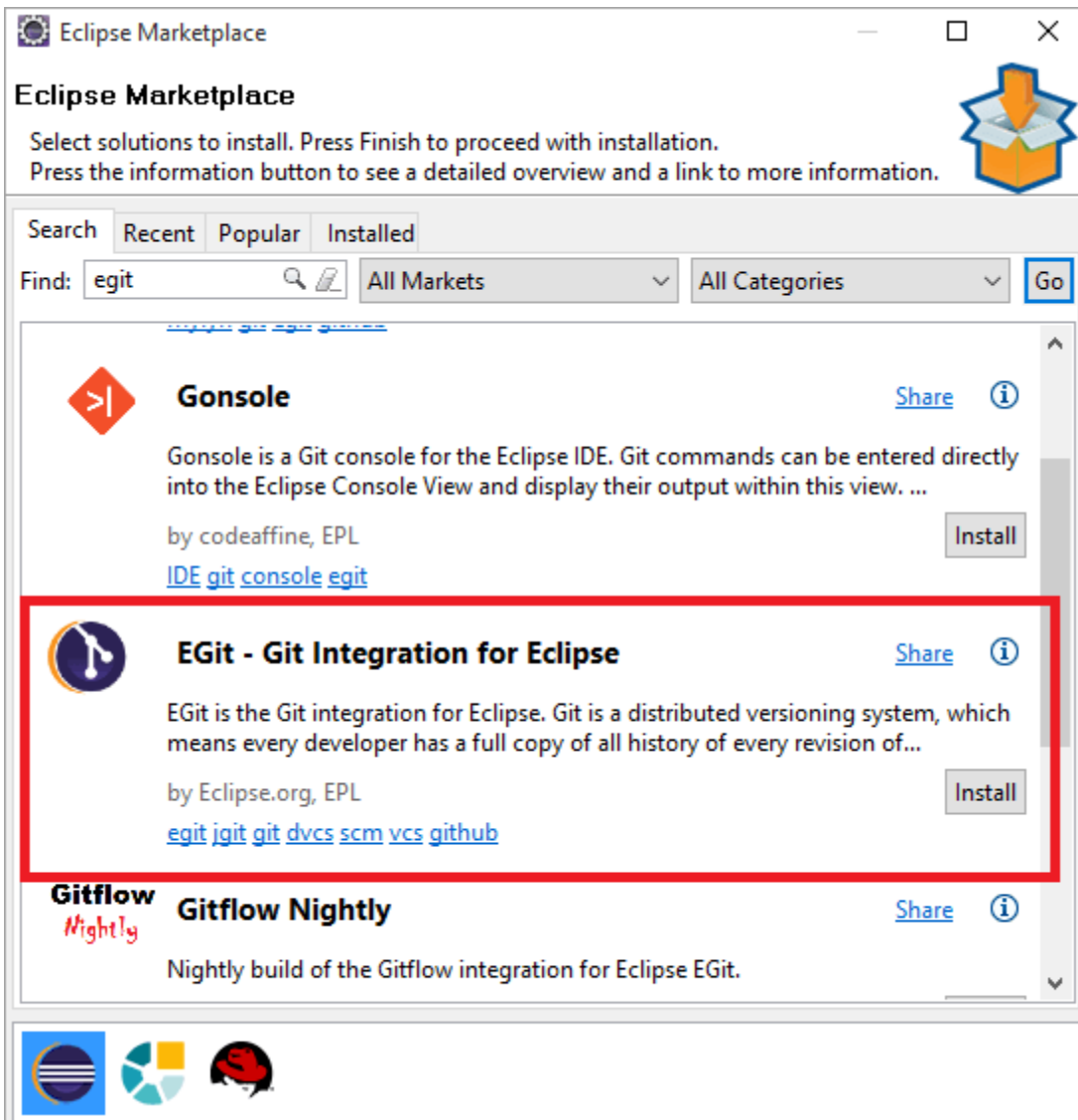
#2) The following screen will be displayed as shown in the below image.



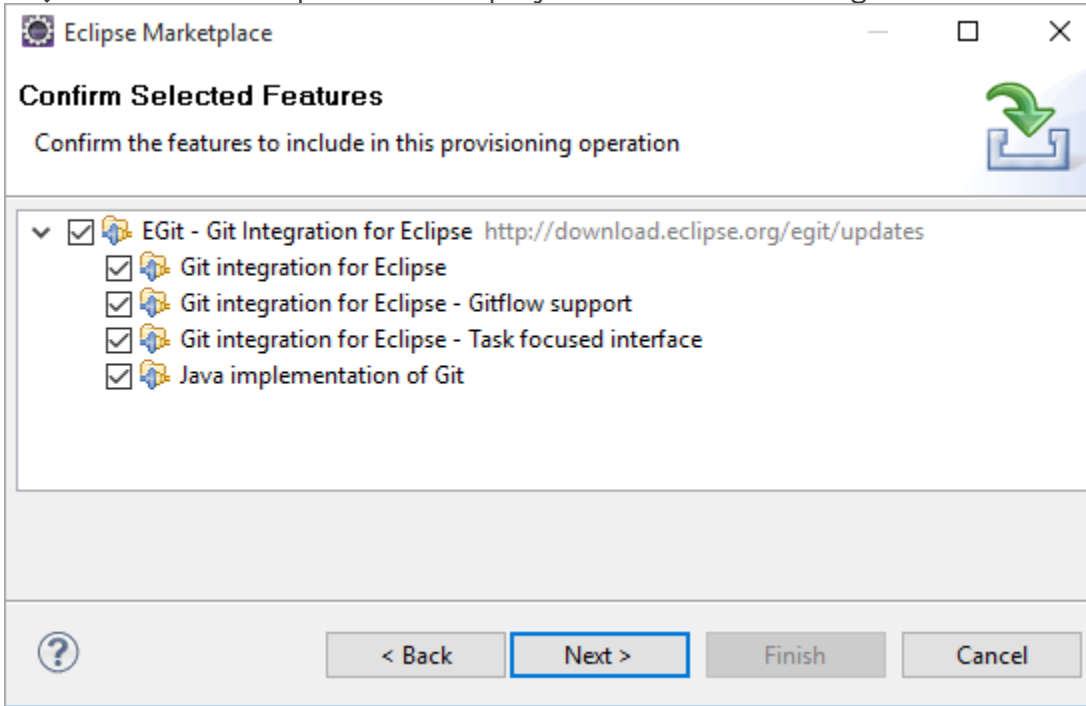
#3) Type “EGit” in the find section as shown in the below image.

#4) Click on Go.

#5) Click on the install button to Install “**EGit – Git integration for Eclipse**”.



#6) Select all the options as displayed in the below image. Click on the Next button.



#7) Accept the license agreement and click on the Finish button.

Thus, you have successfully installed the Git plugin on your PC.

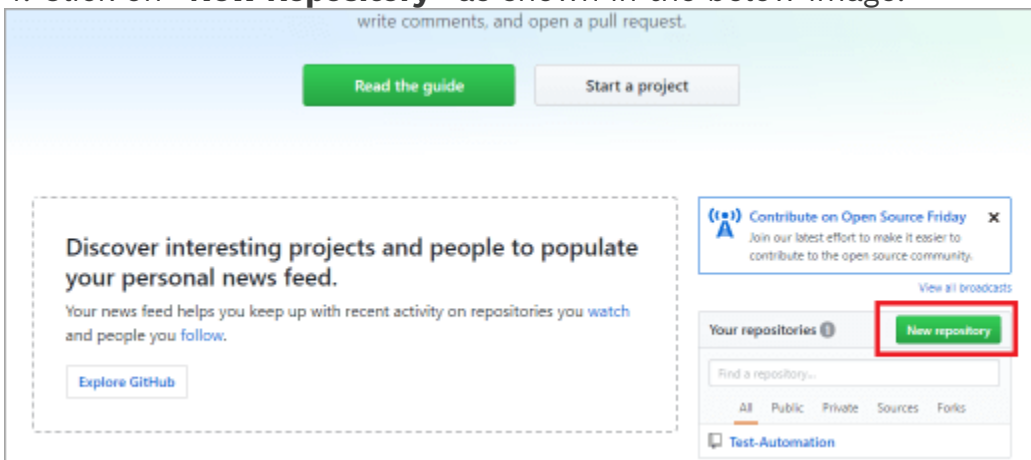
Create Repository on GitHub

Before learning the process of creating a repository, let us understand what a repository is.


A Repository in context with GitHub means a central location where all data, files, etc., can be stored. It is particularly used to efficiently co-ordinate the activities of a project. You can add java files, screenshots, videos, Excel sheets, documents, etc., in short, all that which your project needs can be added to a repository.

Follow the below steps to create a repository on GitHub:

1. Navigate to <https://github.com/>.
2. Complete the sign-up process.
3. Log in with valid credentials.
4. Click on **“New Repository”** as shown in the below image.





5. Enter the name of the repository in the **“Repository name”** text box.
6. Provide a description (optional) of the repository.
7. Click on the **“Create Repository”** button.

Owner  / Repository name ✓


Great repository names are Your new repository will be created as Videocon-CRM turbo-succotash.

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ 

Create repository

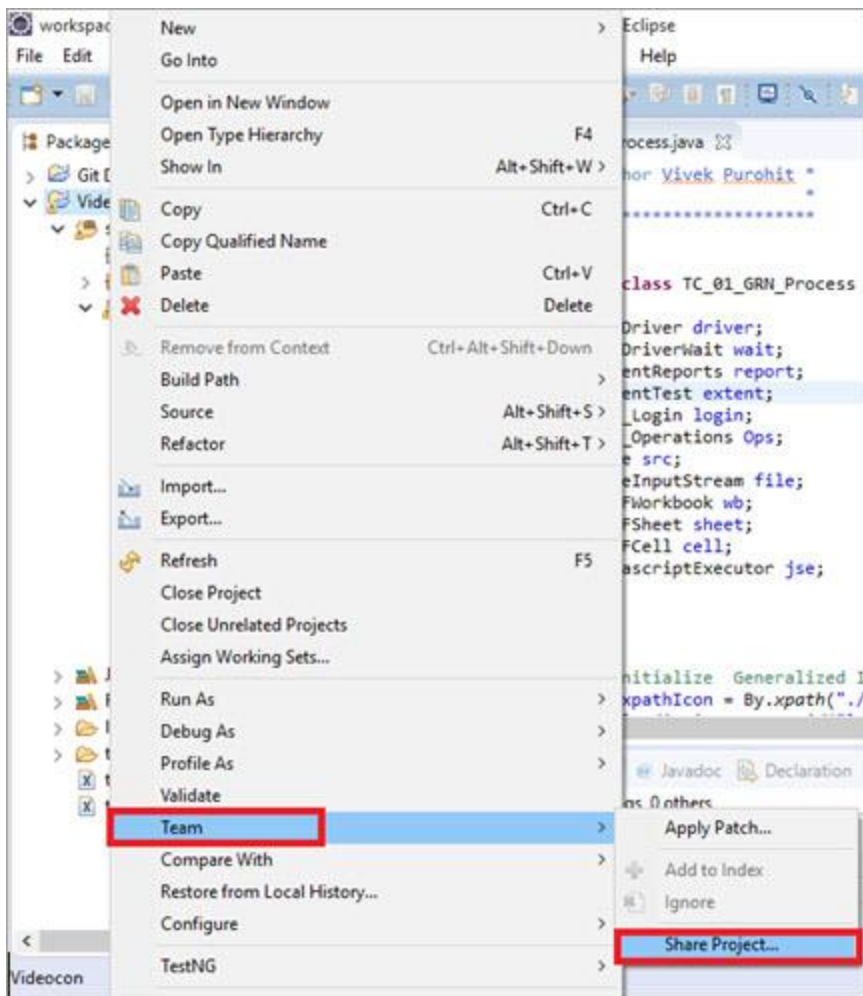
Thus a repository gets created.

Selenium Integration With GitHub Using Eclipse

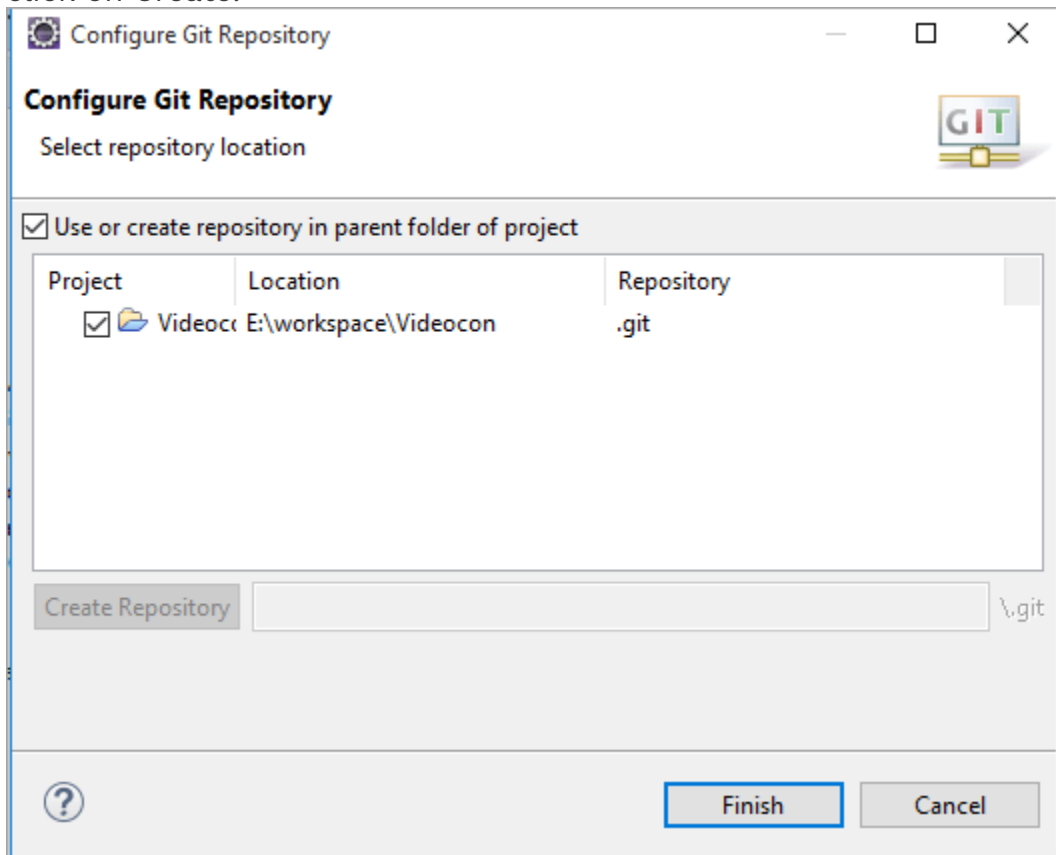
Given below are the steps involved in the Integration of Selenium Automation Script with GitHub Using Eclipse IDE

#1) To integrate Selenium with GitHub, launch Eclipse IDE and navigate to the Selenium Automation project which is to be synced with GitHub.

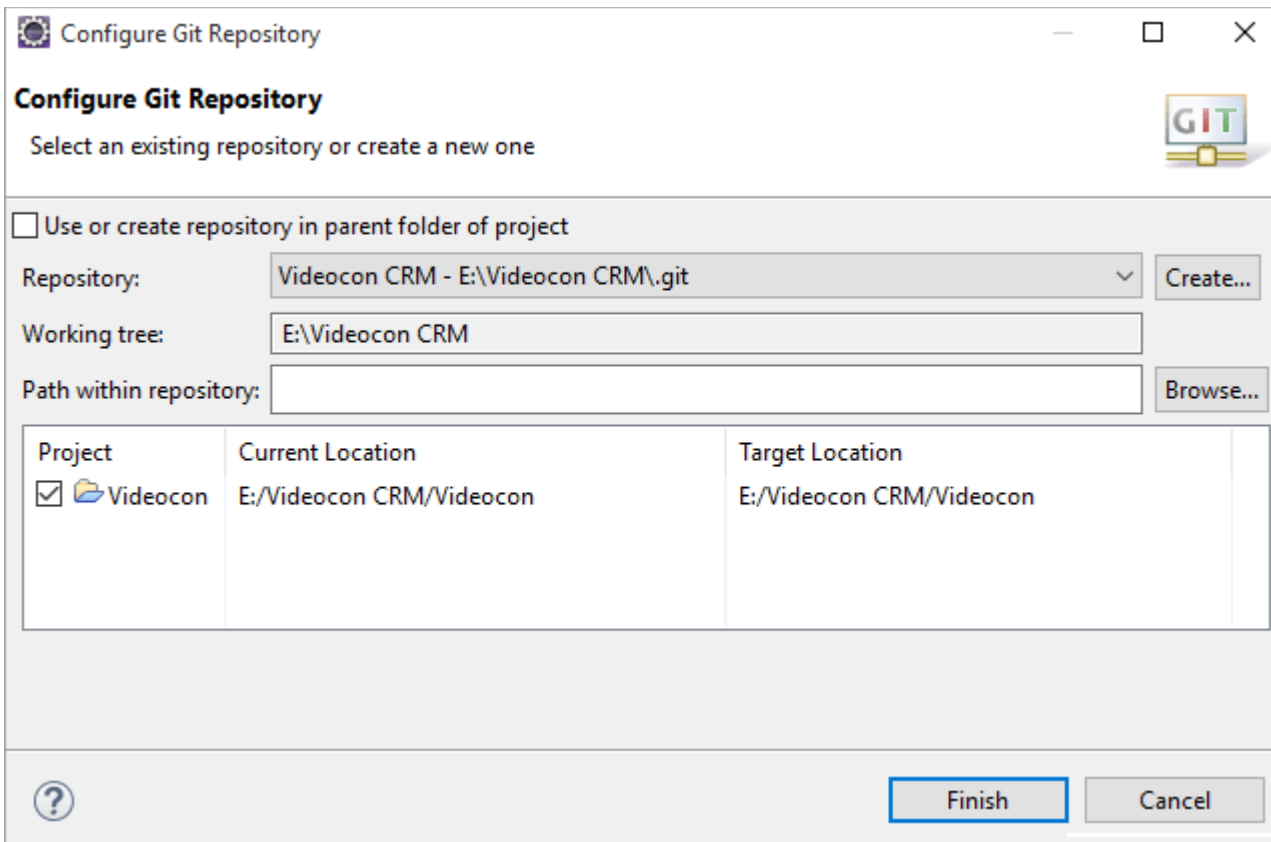
#2) Right-click on the project and navigate to **Team => Share Project**.



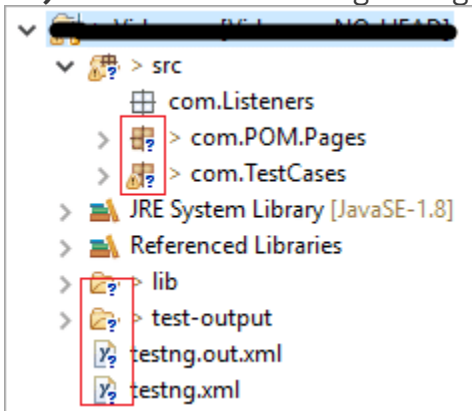
#3) Select the Repository from the drop-down. If no data is displayed in the dropdown, then click on Create.



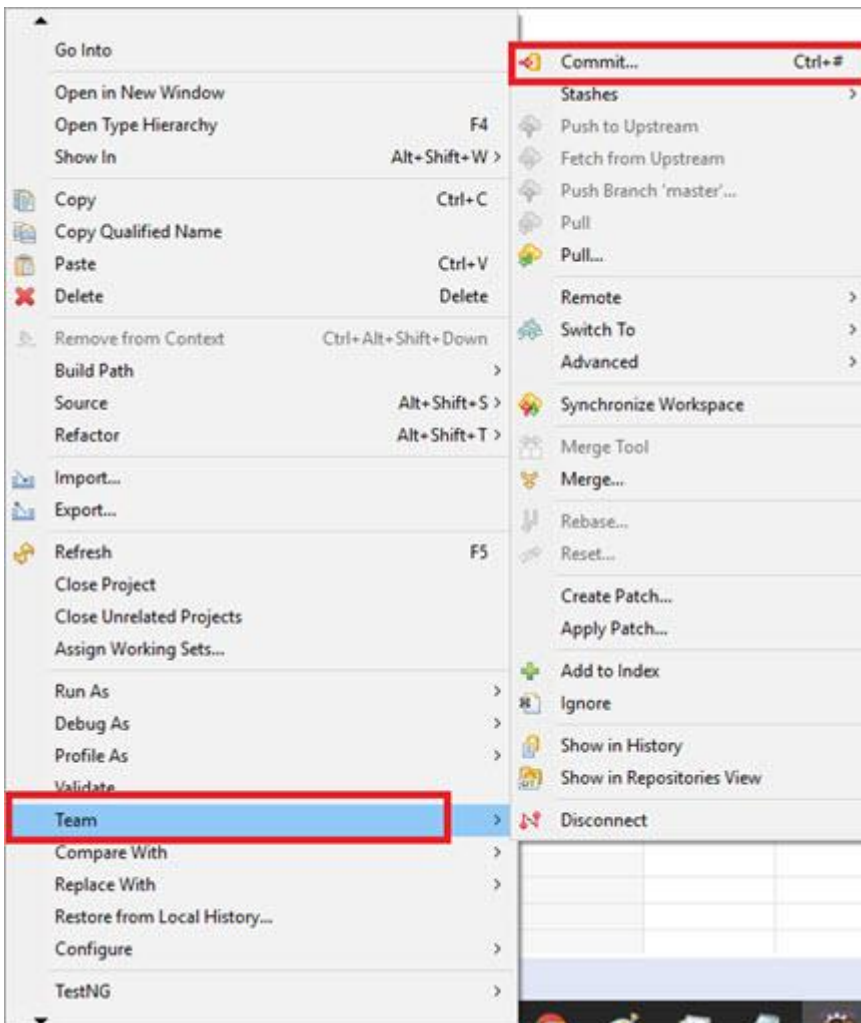
#4) Click on the Selenium Automation project which you want to integrate with GitHub. Click on the **Finish** button.



#5) Notice the following change in the structure of your Selenium project.

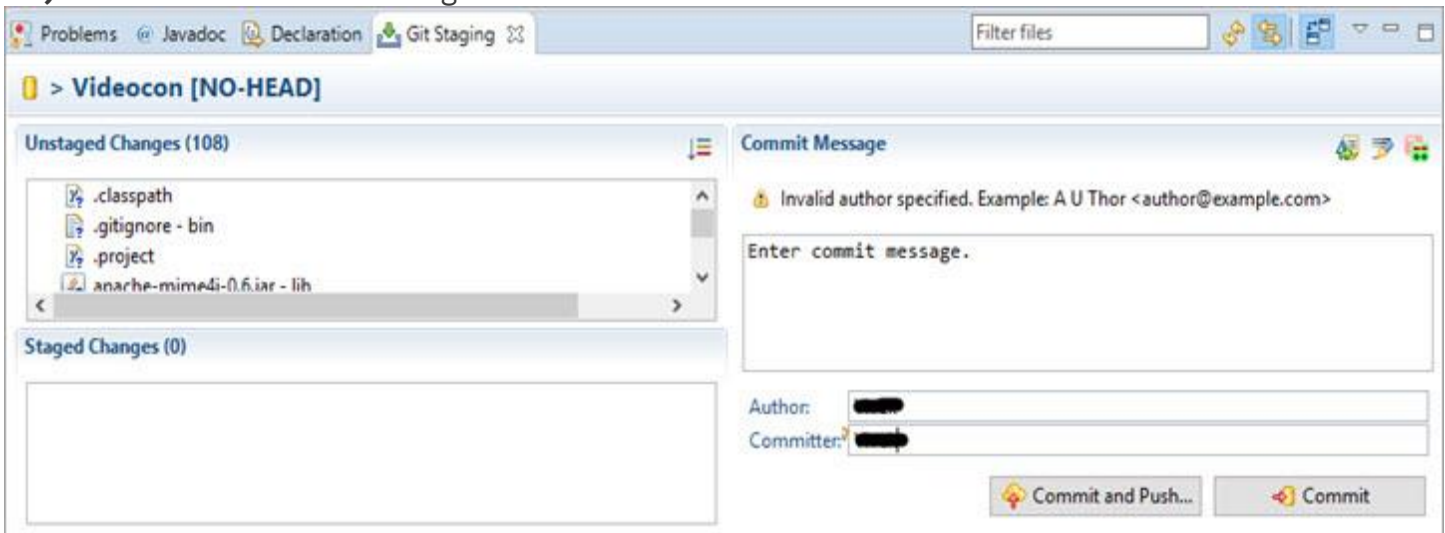


#6) Right-click on the project which was configured with the Git Repository. Navigate to **Team** => **Commit**.



#7) Right-click on the Selenium automation project and navigate to **Team => Add to Index**.

#8) Enter the commit message and click on the **Commit** button.



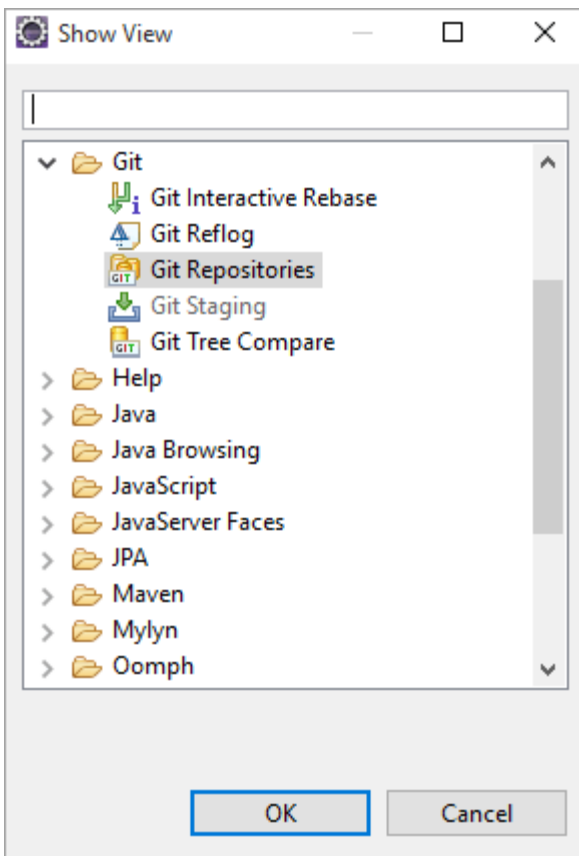
This will add all your Test Cases files to staged changes.

#9) Open Git repository tab in Eclipse.

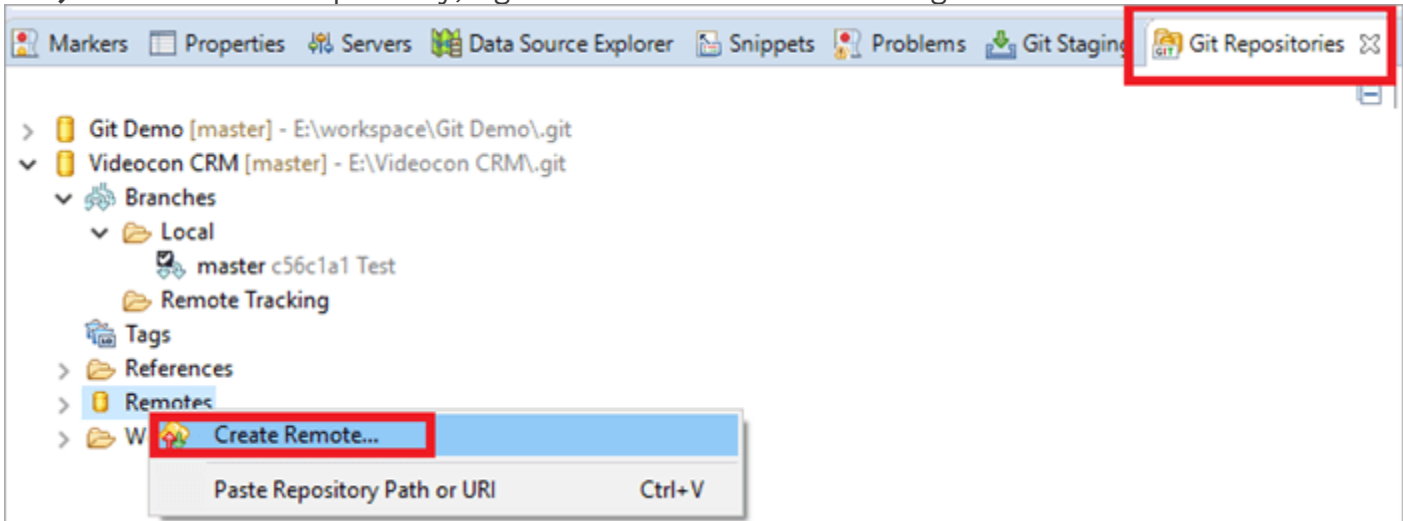
Note: If the tab does not open by default, then follow the below steps.

a) Navigate to **Windows => Show view => Other**.

b) Under the Git folder select, **Git repositories and Git Staging** and click on the OK button.



#10) Under the Git Repository, right-click on remote and navigate to create remote.

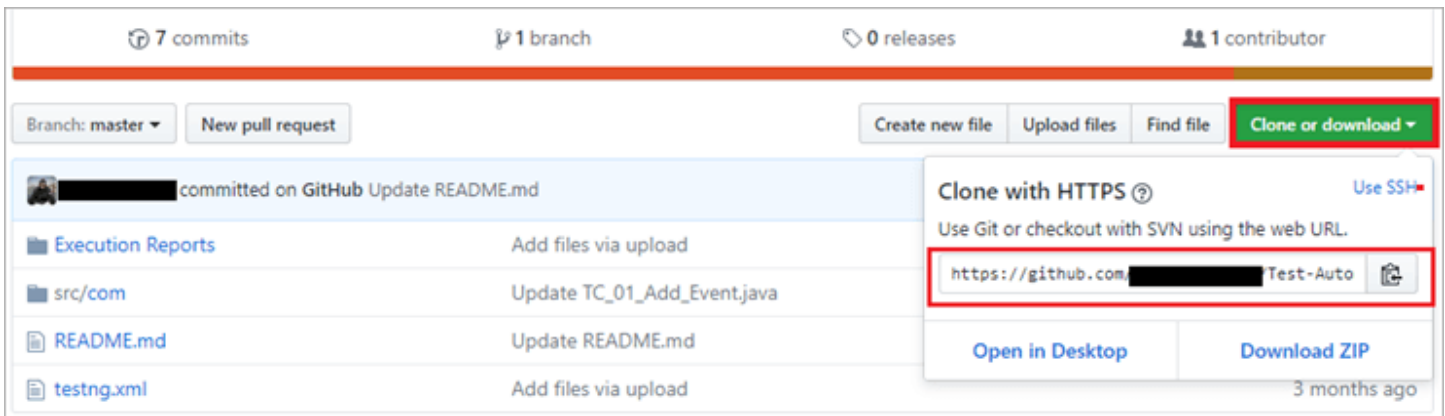


#11) A new pop-up window will open, provide the remote name. Leave the other settings unchanged and click on the OK button.

#12) Another pop-up window will open, provide the URL of the GitHub repository which can be copied by following the below-mentioned steps:

a) Navigate to the **created repository** on GitHub.

b) Click on Clone or download as shown in the below image.

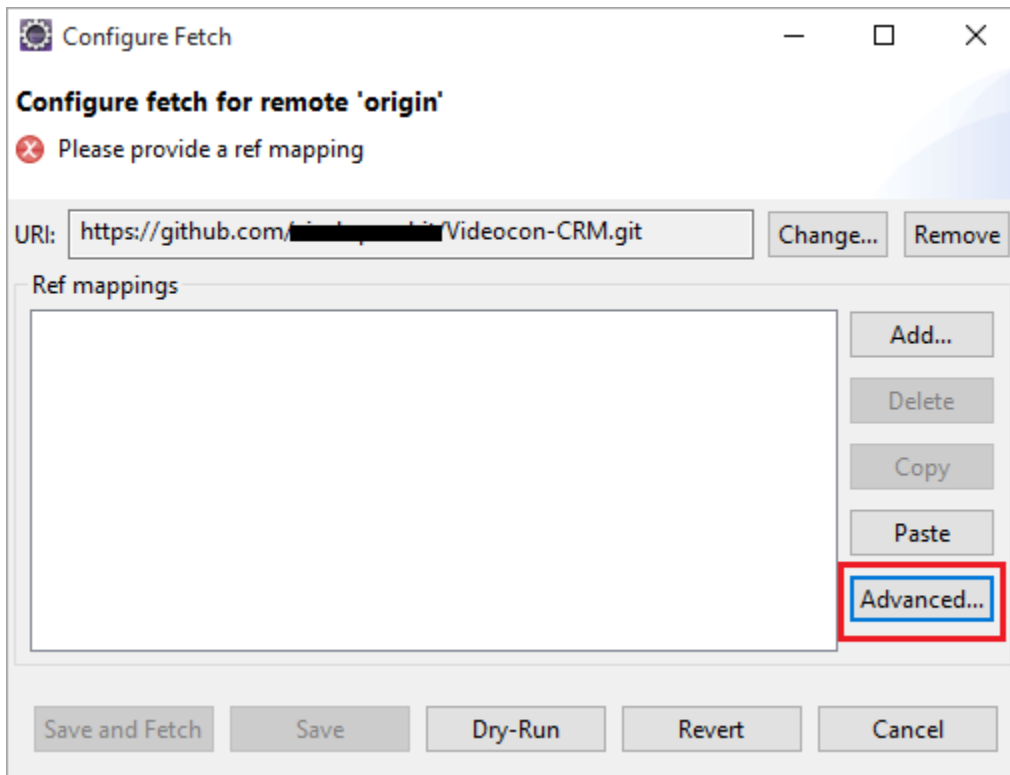


c) Copy the URL.

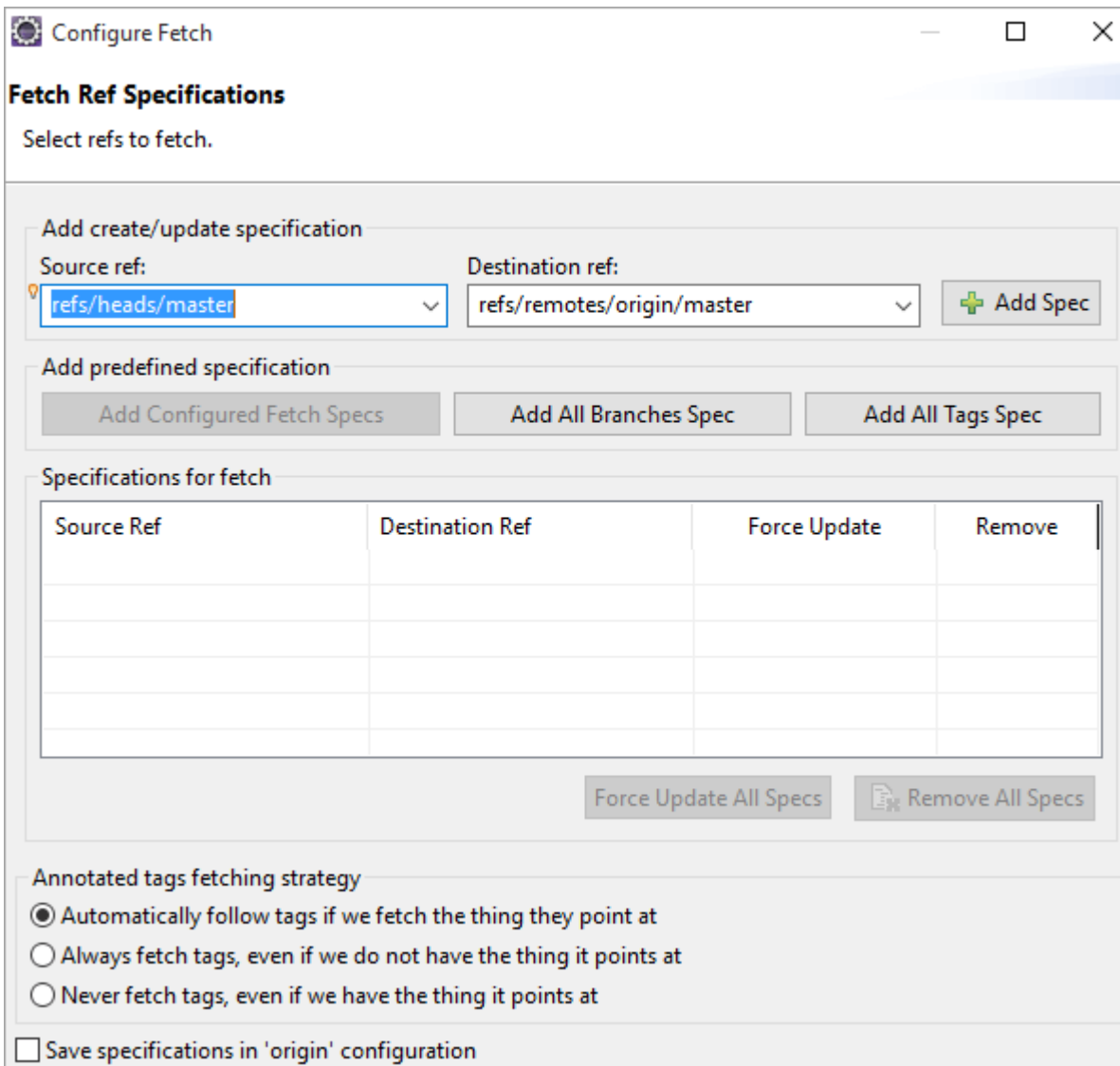
#13) Provide the copied URL and the other details including your login credentials of the GitHub account as shown in the below image, and click on the Finish button.

The screenshot shows a 'Select a URI' dialog box with the title 'Source Git Repository'. It prompts the user to 'Enter the location of the source repository.' The dialog is divided into three main sections: 'Location', 'Connection', and 'Authentication'.
 - **Location:** The 'URI' field contains 'https://github.com/[redacted]Videocon-CRM.git'. There are also fields for 'Host' (github.com) and 'Repository path' ([redacted]Videocon-CRM.git). A 'Local File...' button is present.
 - **Connection:** The 'Protocol' is set to 'https'. The 'Port' field is empty.
 - **Authentication:** The 'User' field contains '[redacted]'. The 'Password' field is masked with dots. There is a checkbox labeled 'Store in Secure Store' which is checked.
 At the bottom, there are 'Finish' and 'Cancel' buttons.

#14) Once the configuration process is done, we need to select the branch in which we will commit changes. Click on the **Advanced** button as shown in the image.

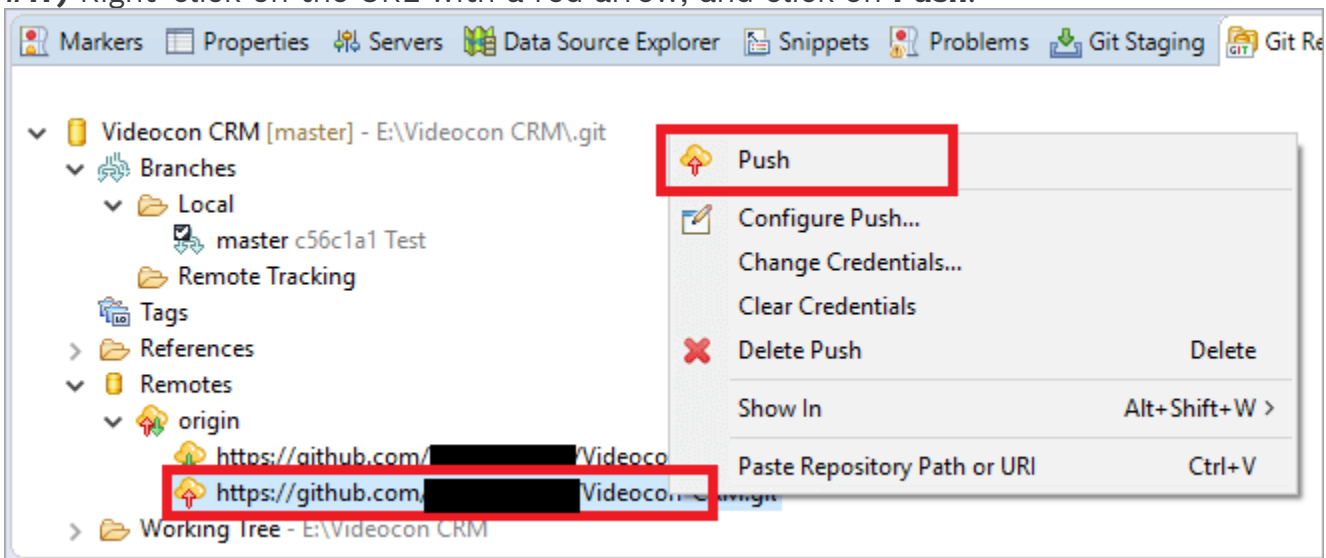


#15) Select your branch and click on the **Add Spec** button.

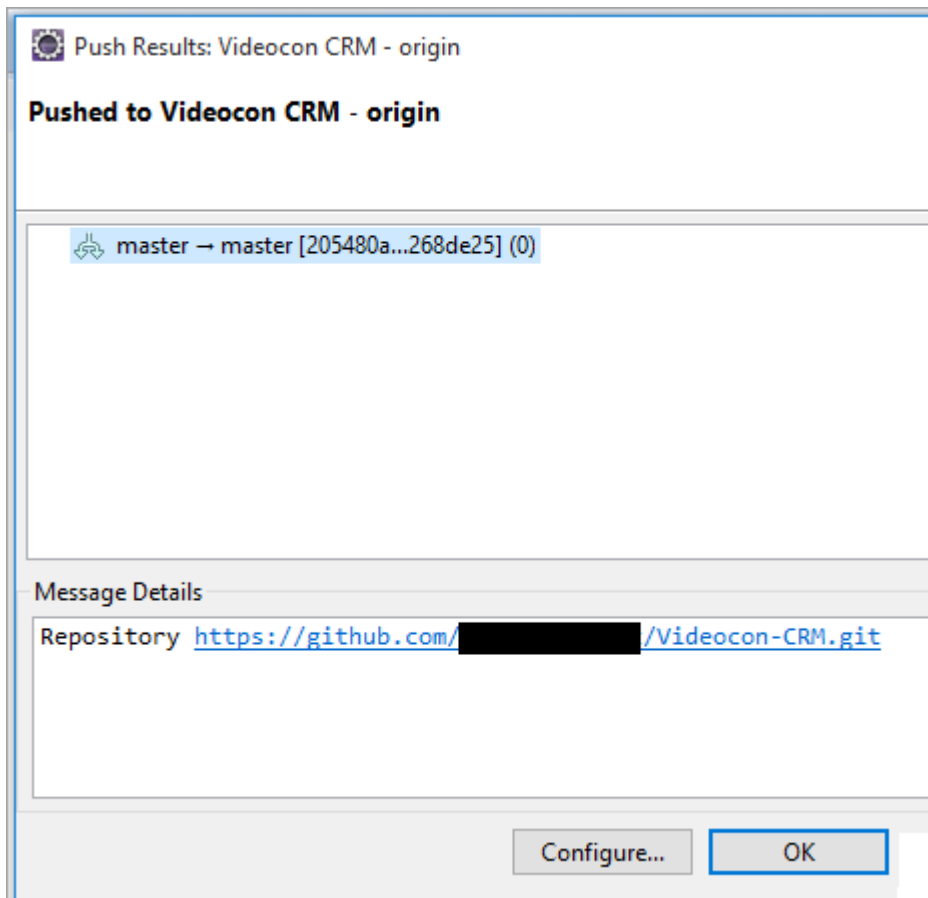


#16) Finally, click on the Finish button. Now under Remote, you will find a folder which is the name you provided in Step 11.

#17) Right-click on the URL with a red arrow, and click on **Push**.



#18) Thus, all the changes that were made and the Test Cases of your Selenium Project will get committed to the repository.



#19) Verify the updates in your GitHub account.