

# Jenkins Integration

Class	<a href="#">Software Quality Management &amp; Assurance</a>
Assignment	<a href="#">Jobs in Jenkins</a>
Student	Vlădescu Alexandra-Bianca
Date	December 16, 2022

## Contents

---

### Prerequisites

- Build a Maven Project
- Publish the project on GitHub
- Install Jenkins

### Homework 1

- Create a Jenkins Job that connects to a GitHub repository
- Configure a parameter to decide which one of these tests to run

### Homework 2

- Create a Jenkins Job that connects to a GitHub repository
- Configure the build to run all tests

---

### Homework 1 - Description:

- Create a Jenkins job that connects to a GitHub repository where you have a minimum 2 tests.
- The Jenkins user should decide by a parameter which one of these tests to run. (you can have the tests in different TestCases).
- The repository name should be in the format: SQMA\_SecondName\_FirstName.  
Example: SQMA\_Zamfiroiu\_Alin
- Create a document with every step and the result of running this job.

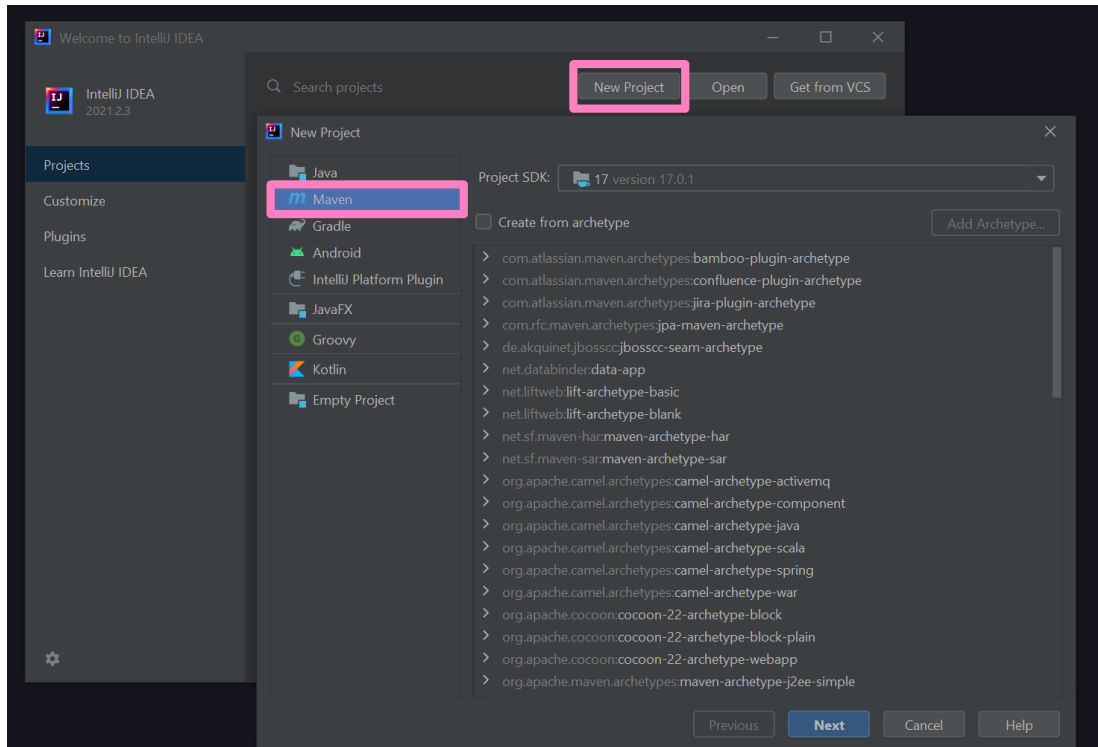
### Homework 2 - Description:

- You have to create more jobs in a pipeline that will run all tests from your repository.
- Create a document with every step and the result of running this pipeline.
- Please put in different sections of this document Homework 1 and Homework 2.

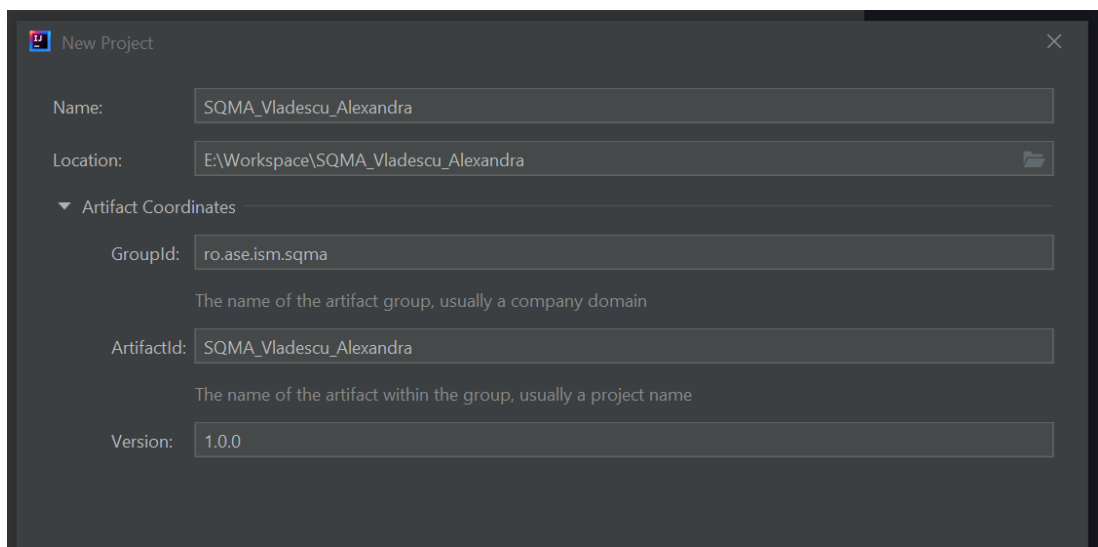
# Prerequisites

## Build a Maven Project

1. Create a new project with maven in IntelliJ.



2. Add a name that respects the format: SQMA\_SecondName\_FirstName



3. Add dependencies and write the code.

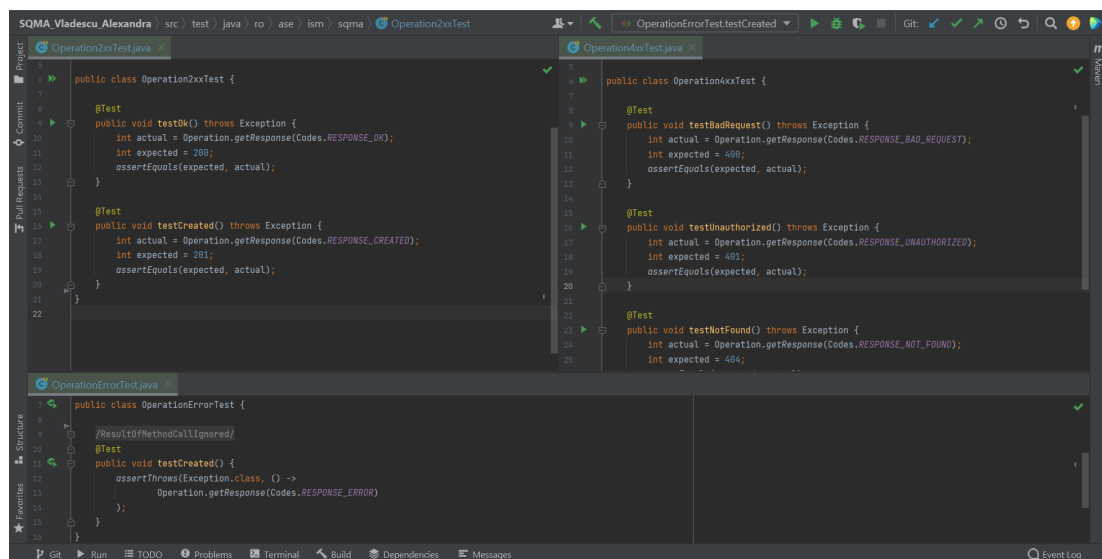
- Method to be tested.

```
package ro.ase.ism.sqma;

public class Operation {

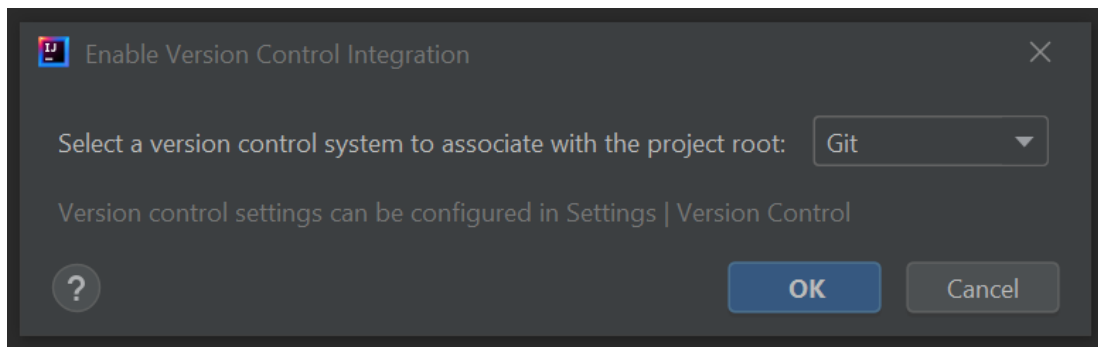
    public static int getResponse(Codes code) throws Exception {
        return switch (code) {
            case RESPONSE_OK -> 200;
            case RESPONSE_CREATED -> 201;
            case RESPONSE_BAD_REQUEST -> 400;
            case RESPONSE_UNAUTHORIZED -> 401;
            case RESPONSE_NOT_FOUND -> 404;
            default -> throw new Exception();
        };
    }
}
```

- Three sets of tests.

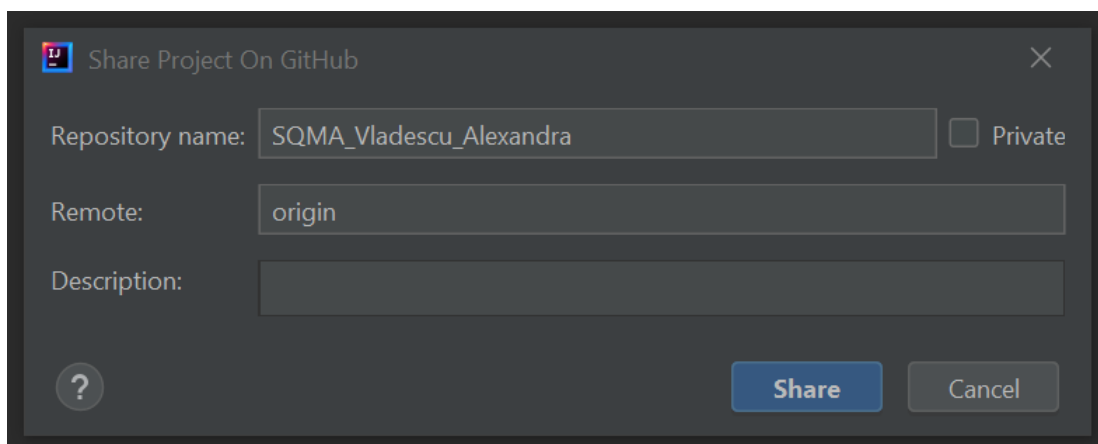


## Publish the project on GitHub

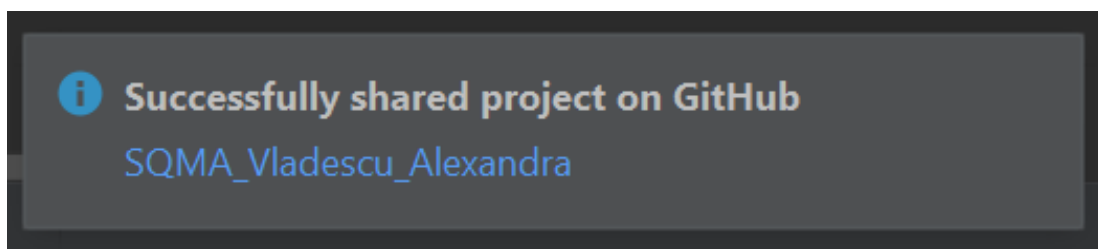
1. From IntelliJ: VCS -> Enable Version Control Integration
  - Select **Git** then click **OK**



2. From IntelliJ: VCS -> GitHub -> Share Project On GitHub

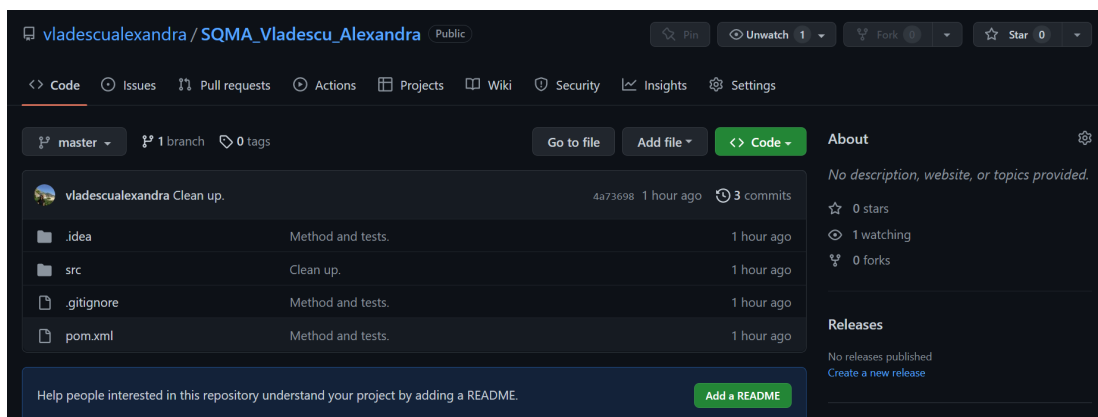


3. A popup will appear after the project is published.



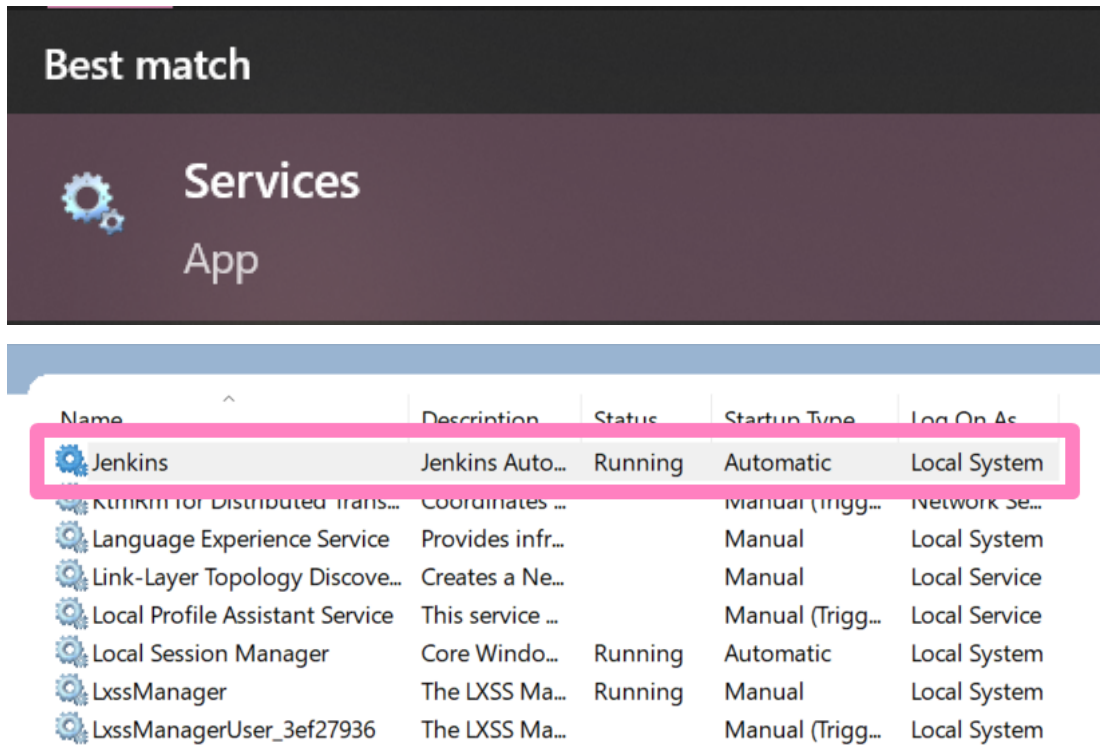
4. Click the link to open the project in GitHub.

This project: [https://github.com/vladescualexandra/SQMA\\_Vladescu\\_Alexandra](https://github.com/vladescualexandra/SQMA_Vladescu_Alexandra)

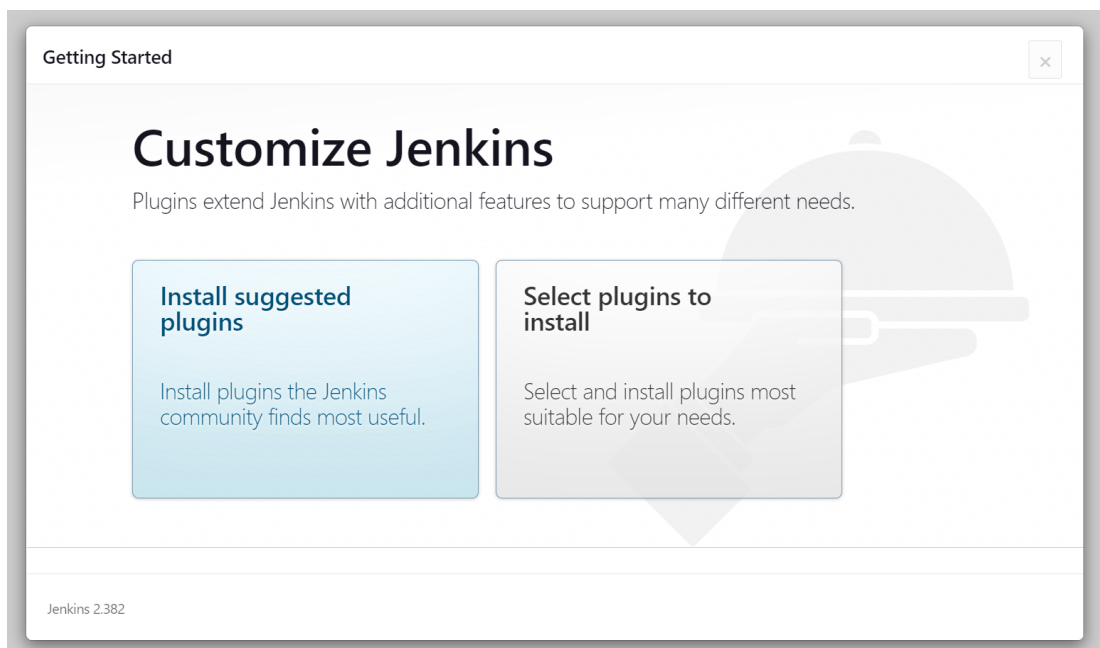


## Install Jenkins

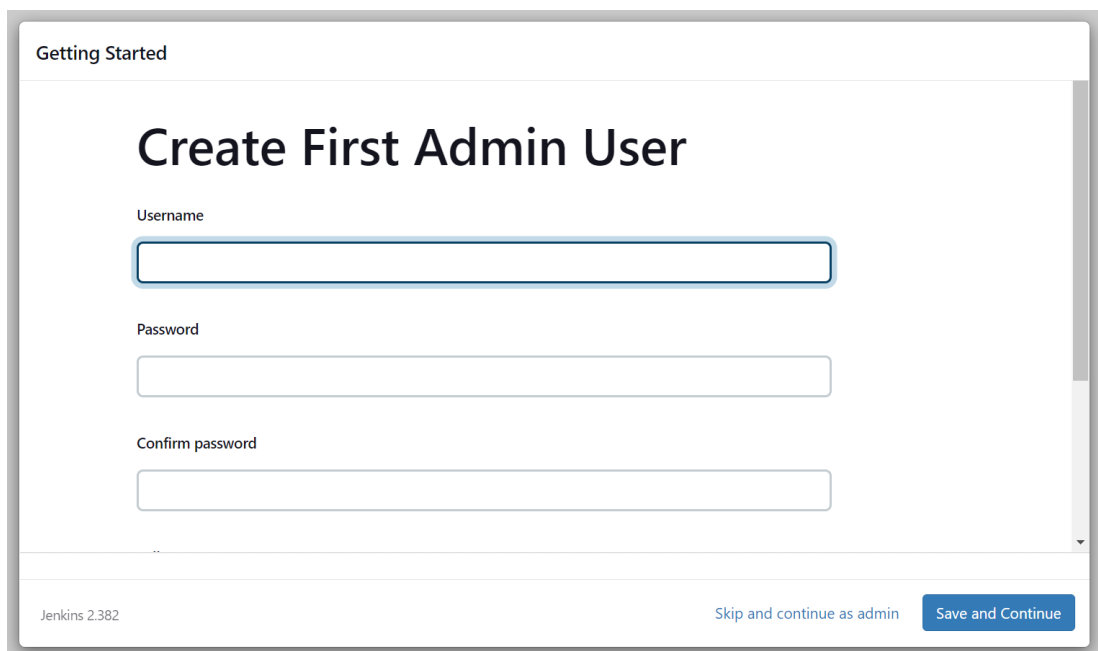
1. Follow instructions from: <https://www.jenkins.io/doc/book/installing/windows/>
2. To check that Jenkins is running, from Windows find Services, then look for the Jenkins service and the status needs to be Running.:



3. Access jenkins on: <http://localhost:8080/> and login
4. Install suggested plugins:

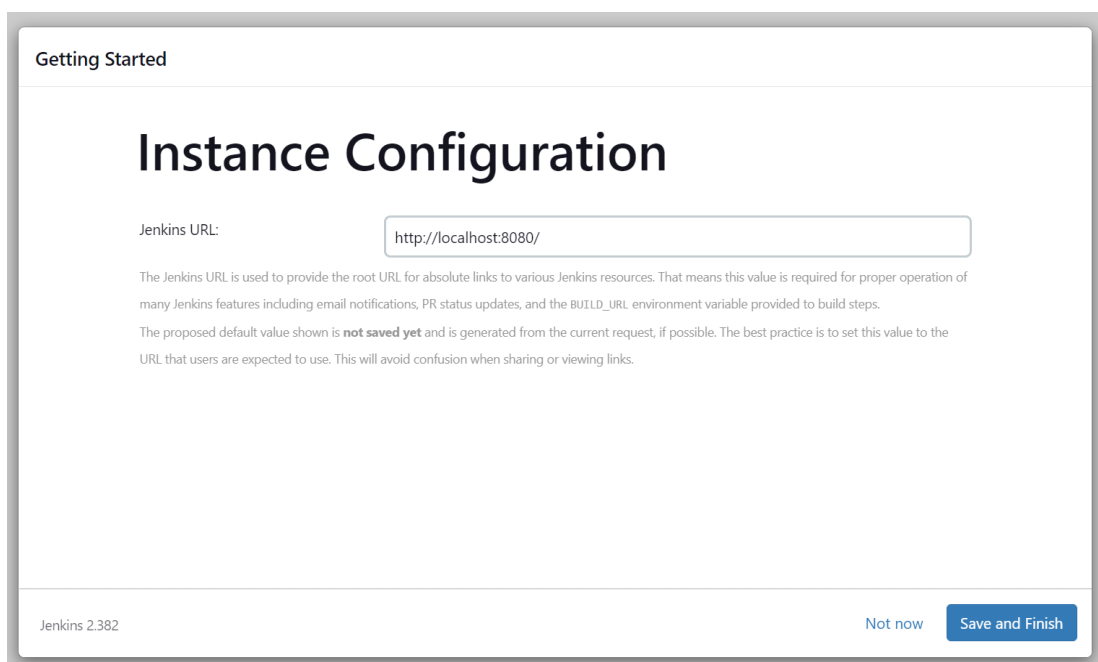


5. Create an user or click on **Skip and continue as admin**



The screenshot shows the 'Getting Started' section of the Jenkins installation wizard. The main heading is 'Create First Admin User'. Below this, there are three input fields: 'Username', 'Password', and 'Confirm password'. The 'Username' field is currently selected. At the bottom of the screen, there is a footer with 'Jenkins 2.382' on the left, and two buttons on the right: 'Skip and continue as admin' and 'Save and Continue'.

6. Configure the URL.



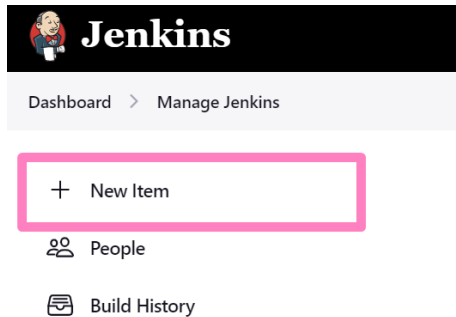
The screenshot shows the 'Getting Started' section of the Jenkins installation wizard. The main heading is 'Instance Configuration'. Below this, there is a label 'Jenkins URL:' followed by a text input field containing 'http://localhost:8080/'. Below the input field, there is a paragraph of text explaining the purpose of the Jenkins URL. At the bottom of the screen, there is a footer with 'Jenkins 2.382' on the left, and two buttons on the right: 'Not now' and 'Save and Finish'.

7. Save and finish

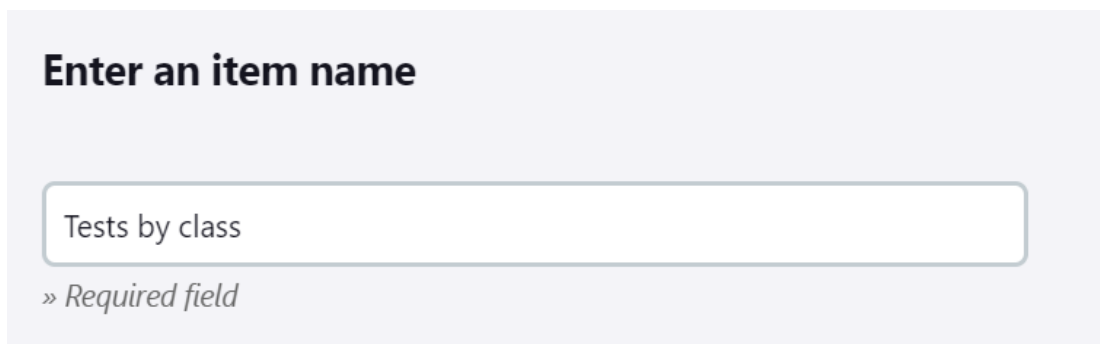
# Homework 1

## Create a Jenkins Job that connects to a GitHub repository

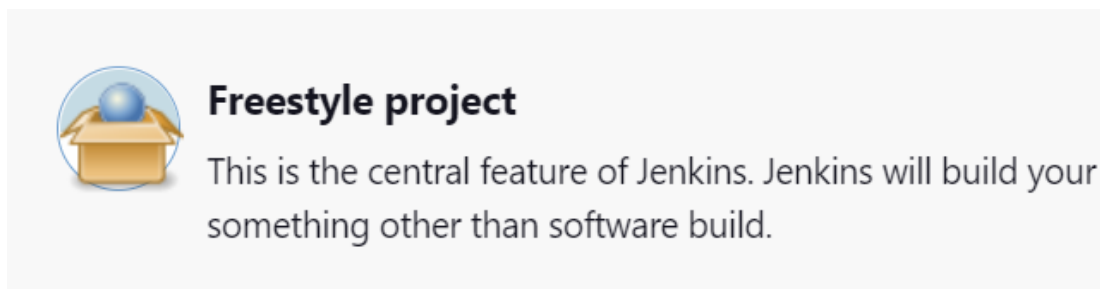
1. From Dashboard > Manage Jenkins > New Item



2. Enter a name for the item.

The image shows a form titled 'Enter an item name'. It has a single text input field containing the text 'Tests by class'. Below the input field, there is a small text note that reads '» Required field'.

3. Choose a Freestyle Project.



4. Source code management -> Select Git

- Repository URL: [https://github.com/vladescualexandra/SQMA\\_Vladescu\\_Alexandra](https://github.com/vladescualexandra/SQMA_Vladescu_Alexandra)

The image shows the 'Source code management' section of the Jenkins job configuration. The 'Git' option is selected with a radio button. Below it, there is a section titled 'Repositories' with a question mark icon. Inside this section, there is a 'Repository URL' field with a question mark icon. The URL 'https://github.com/vladescualexandra/SQMA\_Vladescu\_Alexandra' is entered in the field. A red 'X' icon is visible in the top right corner of the repository configuration area.

Configure a parameter to decide which one of these tests to run.

1. From General -> check "This project is parameterised"

☒ This project is parameterised ?

Add Parameter ▾

2. Click on Add Parameter -> Choice

Name	Class
Choices	Operation2xxTest Operation4xxTest OperationErrorTest

☒ This project is parameterised ?

≡ Choice Parameter ?

Name ?

Class

Choices ?

Operation2xxTest  
Operation4xxTest  
OperationErrorTest

3. Add Build Steps:

Add Build Steps	Execute Windows batch command
Command	E: cd Workspace\SQMA_Vladescu_Alexandra C:\Software\apache-maven-3.6.0\bin\mvn clean -Dtest=%Class% test

Build Steps

≡ Execute Windows batch command ?

Command

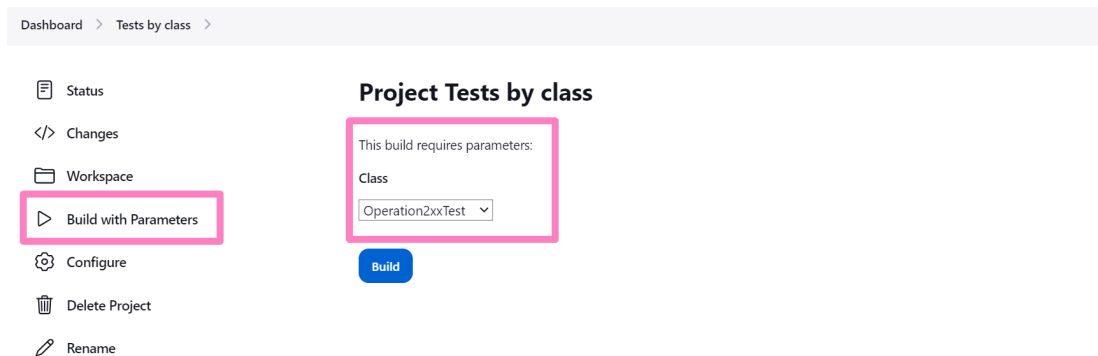
See [the list of available environment variables](#)

E:  
cd Workspace\SQMA\_Vladescu\_Alexandra  
C:\Software\apache-maven-3.6.0\bin\mvn clean -Dtest=%Class% test

Advanced...



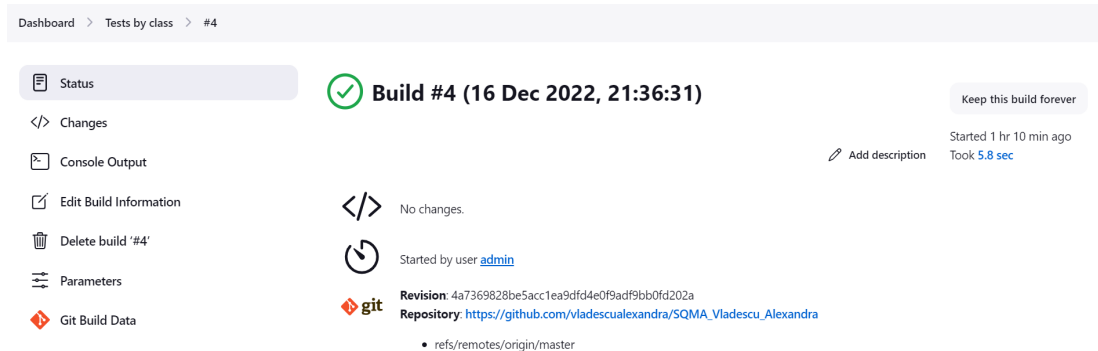
4. Save, then the pipeline will be created.
5. Click on Build with Parameters
6. Choose a class from the list then click on Build



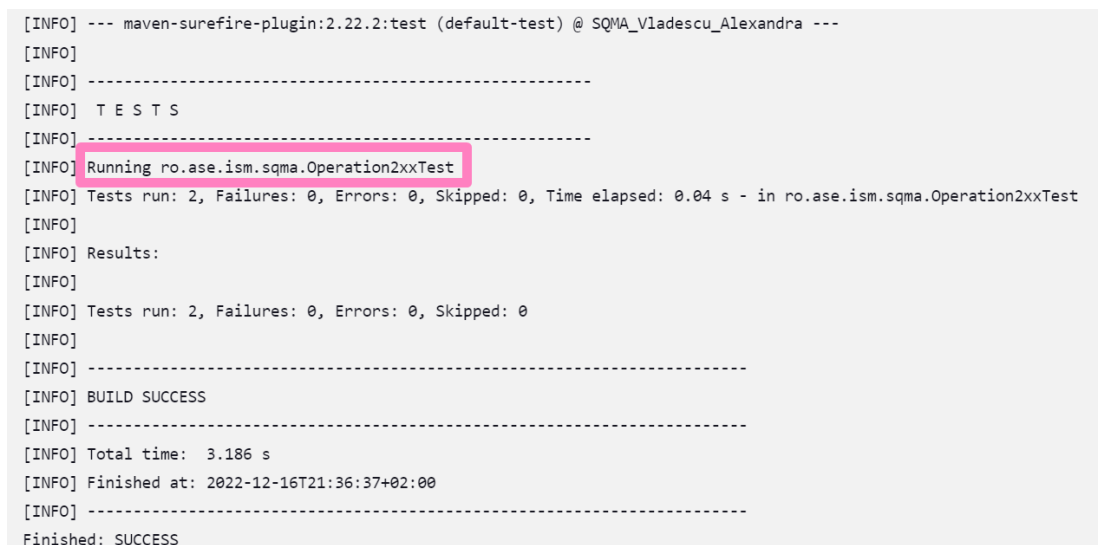
7. You can see the build running.



8. Click on the build number, where you can see the status, then go to Console Output.



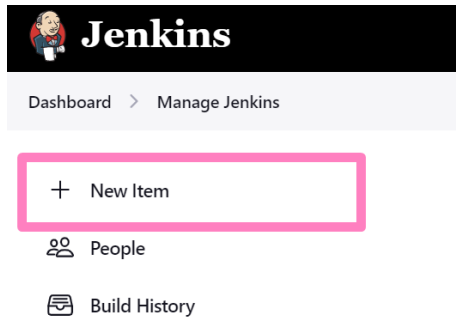
9. In the logs, we can see that only the test from class Operation2xxTest ran.



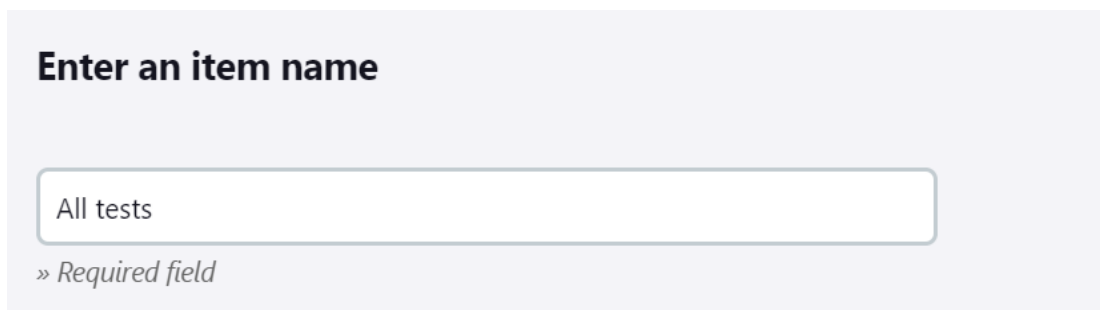
## Homework 2

### Create a Jenkins Job that connects to a GitHub repository

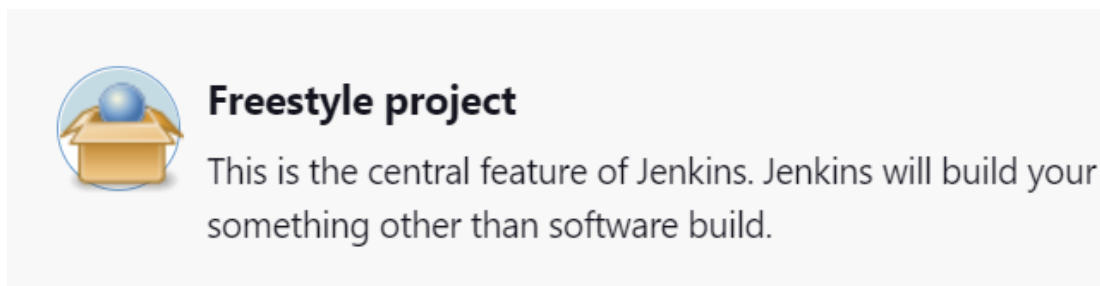
1. From Dashboard > Manage Jenkins > New Item



2. Enter a name for the item.

A screenshot of the 'Enter an item name' form in Jenkins. The title 'Enter an item name' is at the top. Below it is a text input field containing the text 'All tests'. At the bottom of the form, there is a note that says '» Required field'.

3. Choose a Freestyle Project



4. Source code management -> select Git

- Repository URL: [https://github.com/vladescualexandra/SQMA\\_Vladescu\\_Alexandra](https://github.com/vladescualexandra/SQMA_Vladescu_Alexandra)

A screenshot of the Jenkins configuration page. At the top, there's a section for 'Source code management' with a radio button selected for 'Git'. Below this, there's a section for 'Repositories' with a question mark icon. Inside this section, there's a label 'Repository URL' with a question mark icon, followed by a text input field containing the URL 'https://github.com/vladescualexandra/SQMA\_Vladescu\_Alexandra'. A red 'X' icon is visible in the top right corner of the input field area.

## Configure the build to run all tests

1. Add Build Steps:

Add Build Steps	Execute Windows batch command
Command	E: cd Workspace\SQMA_Vladescu_Alexandra C:\Software\apache-maven-3.6.0\bin\mvn clean test

### Build Steps

≡ Execute Windows batch command ?

Command

See [the list of available environment variables](#)

E:  
cd Workspace\SQMA\_Vladescu\_Alexandra  
C:\Software\apache-maven-3.6.0\bin\mvn clean test

2. Save.
3. From Project Status, click on Build now.

Dashboard > All tests >

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Project All tests

Permalinks

4. Observe that the build is running.

... #1

16 Dec 2022, 21:44

5. Click on the build number, then go to Console Output.

Dashboard > All tests > #1

Status

</> Changes

**Console Output**

Edit Build Information

Delete build '#1'

Git Build Data

Next Build

**Build #1 (16 Dec 2022, 21:44:14)**

Add description

No changes.

Started by user [admin](#)

**Revision:** 4a7369828be5acc1ea9dfd4e0f9adf9bb0fd202a  
**Repository:** [https://github.com/vladescualexandra/SQMA\\_Vladescu\\_Alexandra](https://github.com/vladescualexandra/SQMA_Vladescu_Alexandra)

- refs/remotes/origin/master

6. Observe that all tests ran.

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running ro.ase.ism.sqma.Operation2xxTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.032 s - in ro.ase.ism.sqma.Operation2xxTest
[INFO] Running ro.ase.ism.sqma.Operation4xxTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in ro.ase.ism.sqma.Operation4xxTest
[INFO] Running ro.ase.ism.sqma.OperationErrorTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in ro.ase.ism.sqma.OperationErrorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.247 s
[INFO] Finished at: 2022-12-16T21:44:21+02:00
[INFO] -----
Finished: SUCCESS
```

7. In the dashboard we can see all the projects with their last statuses.

Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		All tests	4 min 17 sec <a href="#">#2</a>	N/A	6.1 sec
		Tests by class	12 min <a href="#">#4</a>	N/A	5.8 sec