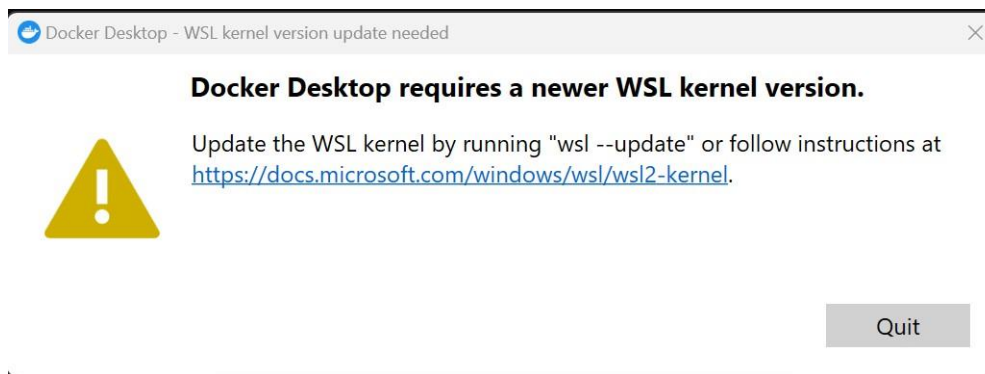


Test management tools Testlink and Jenkins

Remark: Testlink is not working for Mac.

1. Installing Jenkins and Testlink

- 1) Install Docker on your computer. <https://docs.docker.com/desktop/install/windows-install/>
- 2) Download the archive **vvss.zip**.
- 3) Unzip the archive in a directory.
- 4) Open a Command Prompt as administrator to the unzip directory.
- 5) Run the command: `docker-compose up -d`
- 6) If needed, use the newer WSL kernel version.



- 7) Accessing Testlink
<http://localhost:8090/>
user:student
password:student
- 8) Accessing Jenkins
<http://localhost:8080/>

user:student
password:student

Steps for using Testlink

1. User

<http://localhost:8090/>

user:student

password:student

2. Create a project

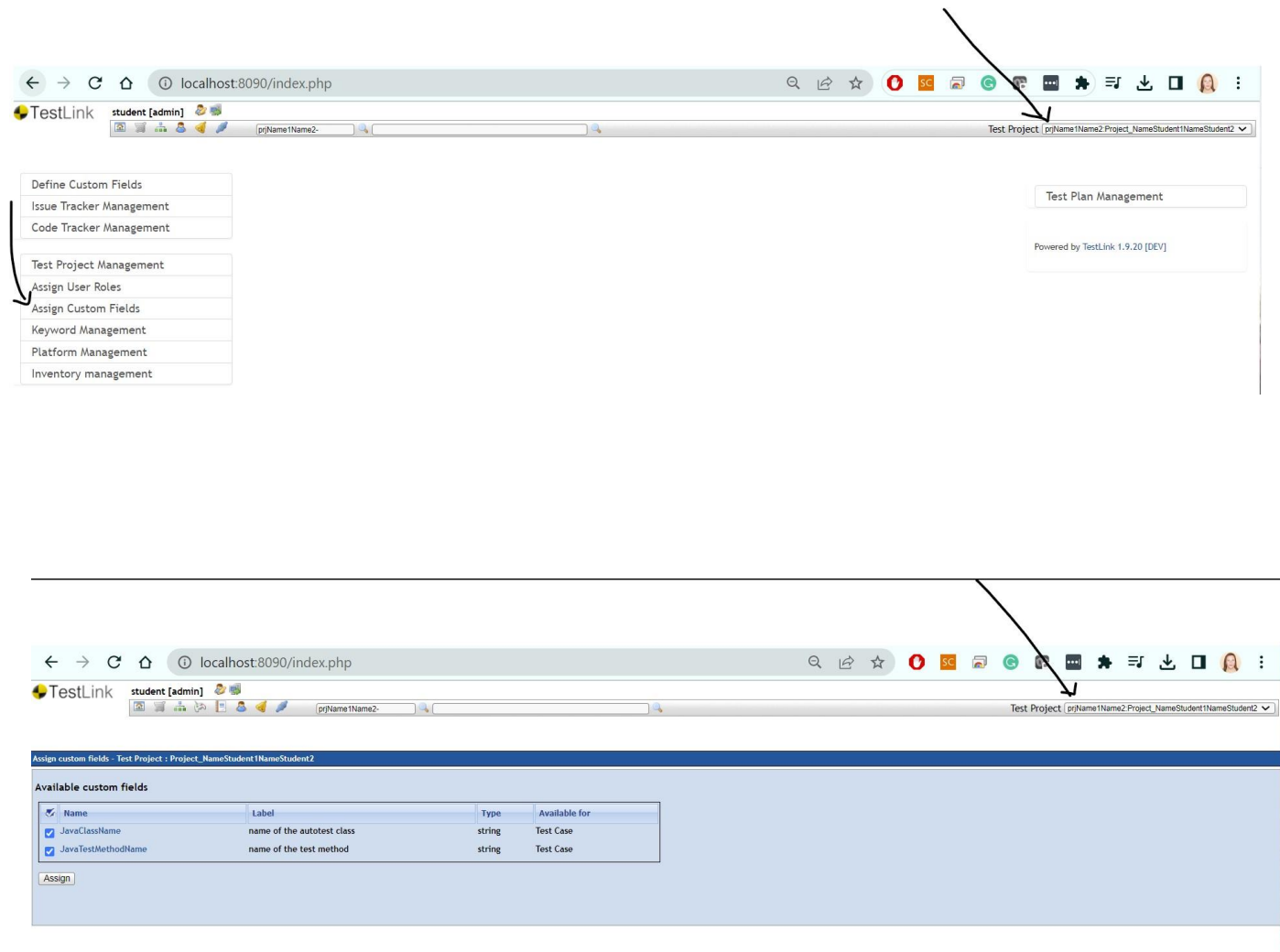
Each team will create a project for their SSVV project. Click the “Create” button and create
Example: Project_NameStudent1NameStudent2

The image shows two screenshots of the TestLink web application. The top screenshot displays the TestLink login page at localhost:8090/index.php. The user is logged in as 'student [admin]'. A dropdown menu is open, showing options: 'Define Custom Fields', 'Issue Tracker Management', 'Code Tracker Management', 'Test Project Management', 'Assign User Roles', 'Assign Custom Fields', 'Keyword Management', 'Platform Management', and 'Inventory management'. The bottom screenshot shows the 'Test Project Management > Create a new project' form. The form includes fields for 'Create from existing Test Project?' (set to 'No'), 'Name' (filled with 'Project_NameStudent1NameStudent2'), 'Prefix (used for Test case ID)' (filled with 'prjName1'), and 'Project description' (with a text area). Below these are 'Enhanced features' (checked: 'Enable Requirements feature', 'Enable Testing Priority', 'Enable Test Automation (API keys)', 'Enable Inventory'), 'Issue Tracker Integration' (message: '>> There are no Issue Tracker Systems defined <<'), 'Code Tracker Integration' (message: '>> There are no Code Tracker Systems defined <<'), and 'Availability' (checked: 'Active', 'Public'). At the bottom are 'Create' and 'Cancel' buttons. A note at the very bottom states '* Mandatory fields'.

3. Assign custom fields

In the Project Menu, in the upper right corner select from the combo box your Test Project.

In the Project Menu, select Assign Custom Fields, select both JavaClassName and JavaTestMethodName and click button Assign.



4. Create a Test Plan

Each team will create his/her own Test Plan, the name of the test plan will be composed by groupNumber followed by the names of the team members concatenated with TestPlan.
93X_Name01Name02_TestPlan

Select “Test Plan Management” and create the test plan using the naming convention above.

The screenshot displays the TestLink web application interface. The top navigation bar includes a sidebar with options like 'Define Custom Fields', 'Issue Tracker Management', 'Code Tracker Management', 'Test Project Management', 'Assign User Roles', and 'Assign Custom Fields'. The main content area shows the 'Test Plan Management' section, which is highlighted by a black arrow. Below this, the 'Edit Test Plan - 93X_Name01Name02_TestPlan' form is visible. The form includes fields for 'Name' (93X_Name01Name02_TestPlan), 'Description' (a rich text editor), 'Active' (checked), 'Public' (checked), and 'API Key' (fac085552017ab456af38a20d1aeecd55103ef573bed55505824ff96d6049d66). There are 'Update' and 'Cancel' buttons. Below the form, there is a section for 'Attached files' and a 'File' section with a 'Choose File' button and an 'Upload file' button. At the bottom, a footer note states: 'Test plans should encompass a (set of) clearly defined tasks with a timeframe and content. They can be created for everything from simple change requests to new product versions. It is recommended that the description field be used to document links to project plans and related documentation, lists of features to be tested, risks, etc. You can create a new test plan from an existing one. The items that are copied include: builds, test cases, priorities, milestones, and user roles. Test plans can be deactivated (i.e., editing and changing of results change are not allowed). Deactivated test plans are visible only via 'Reporting' and this page.'

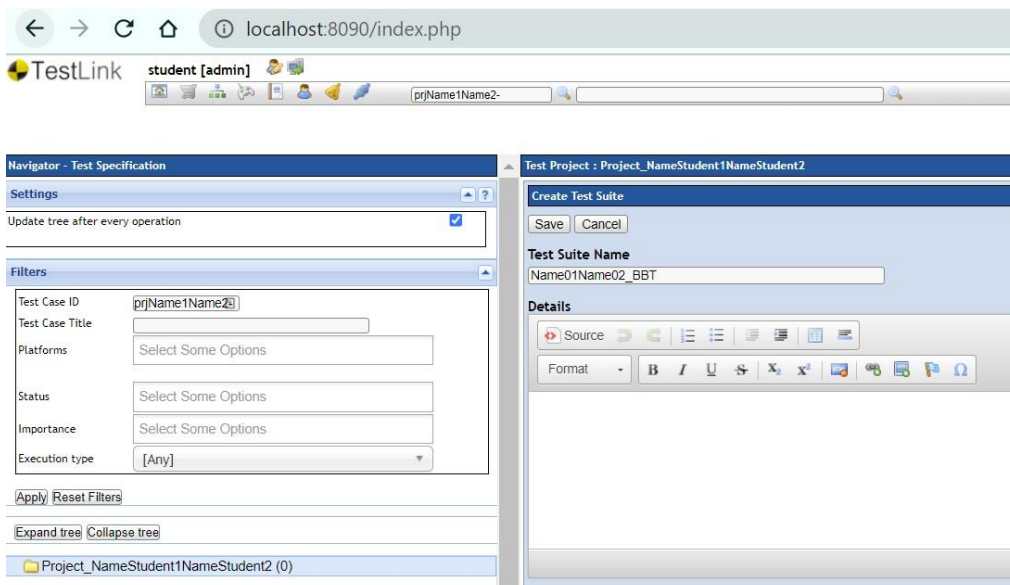
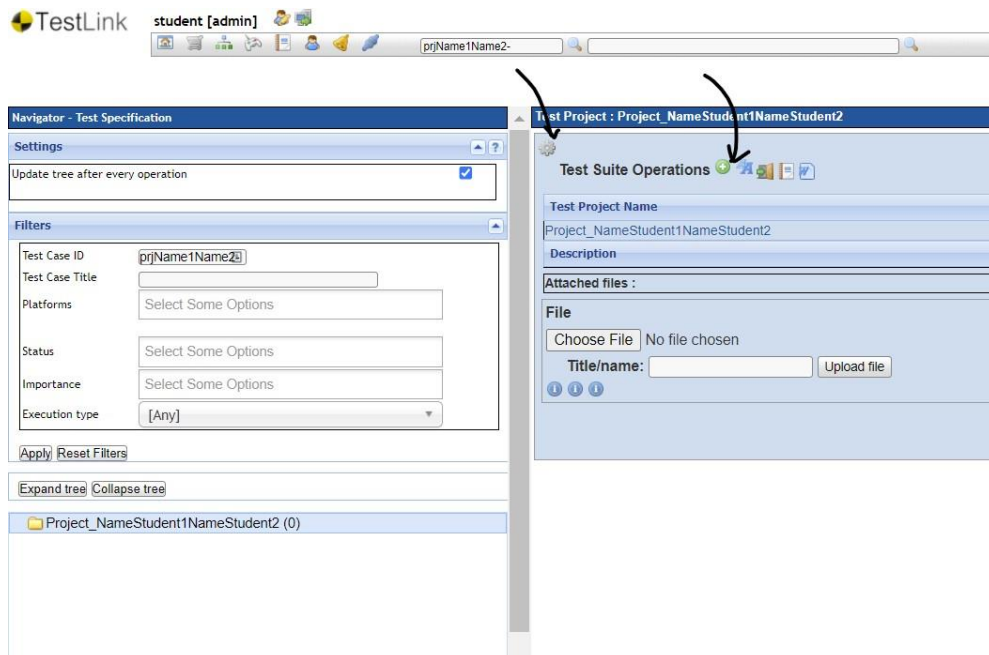
5. Create Test Suites/ Test Cases

Each student will create in total 3 test suites:

- 1 test suite from laboratory 2 - black-box testing
 - Test Suite Name = **Name01Name02_BBT**
- 1 test case from laboratory 3 - white-box testing
 - Test Suite Name = **Name01Name02_WBT**
- 1 test case from laboratory 4 - integration testing.
 - Test Suite Name **Name01Name02_IntegrationT**

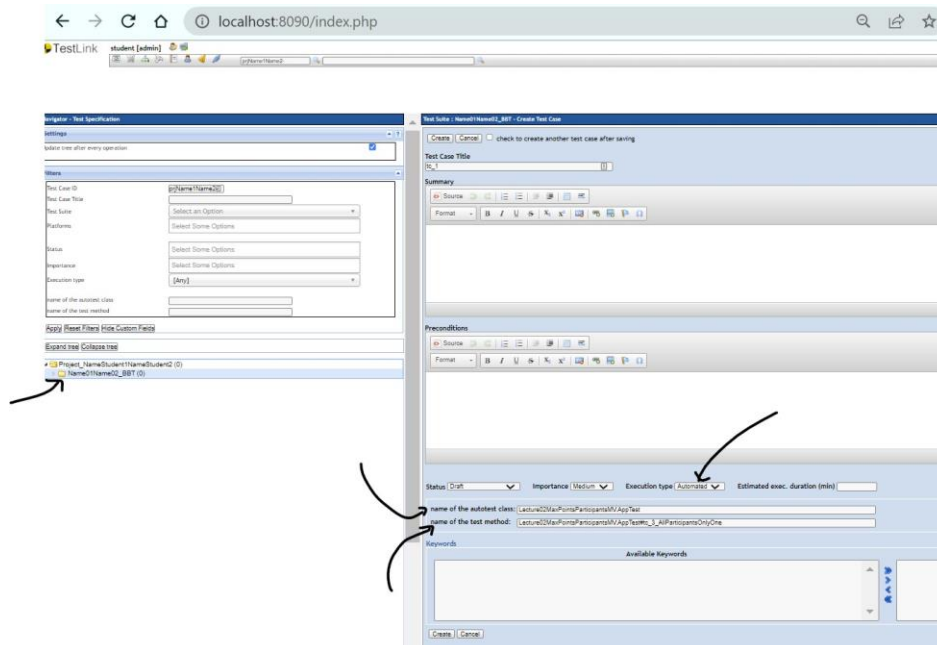
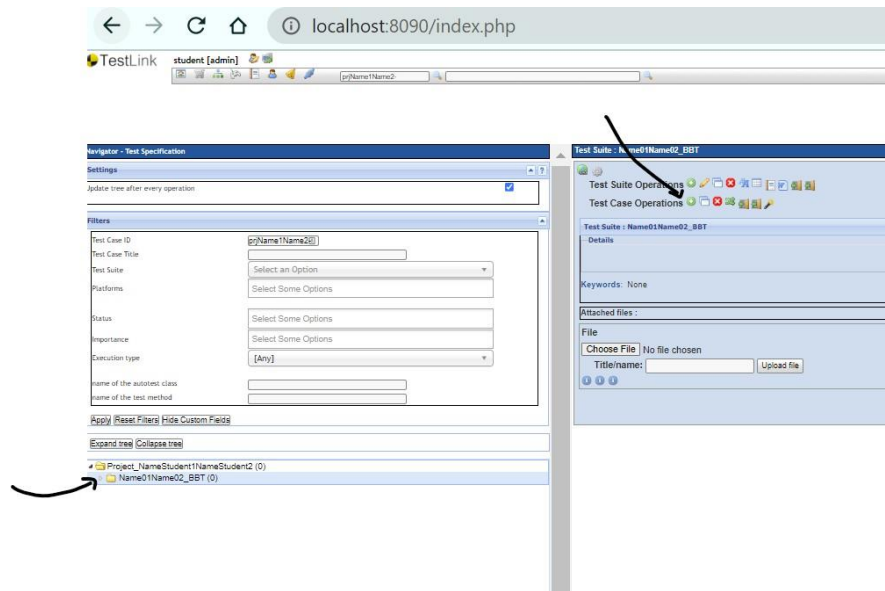
Create Test Suites - STEPS

- Project/Desktop menu --> Test Specification section --> click on Test Specification
- Select your test project in navigation tree (left side screen)
- In the Test Suite Operations panel click on the Action button and then on the Create button.

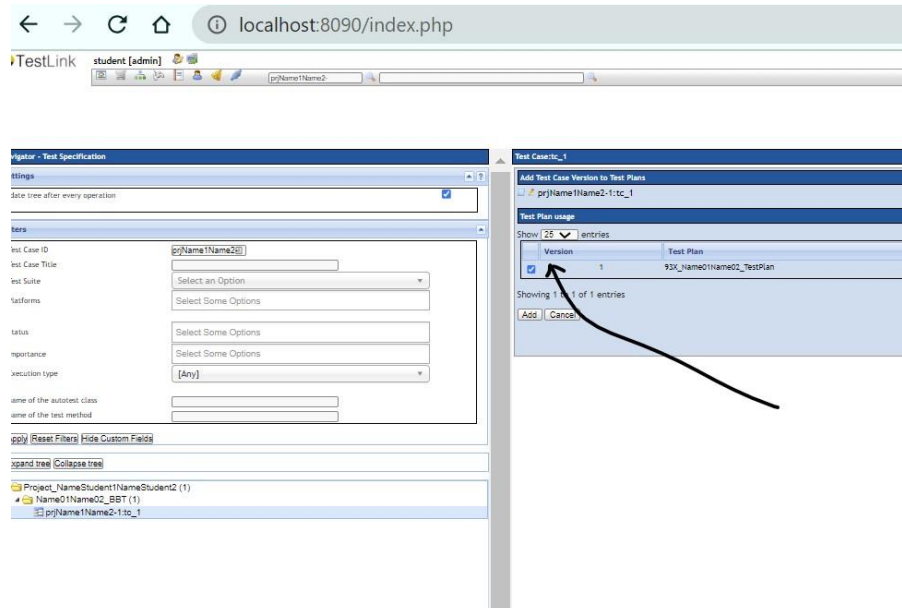
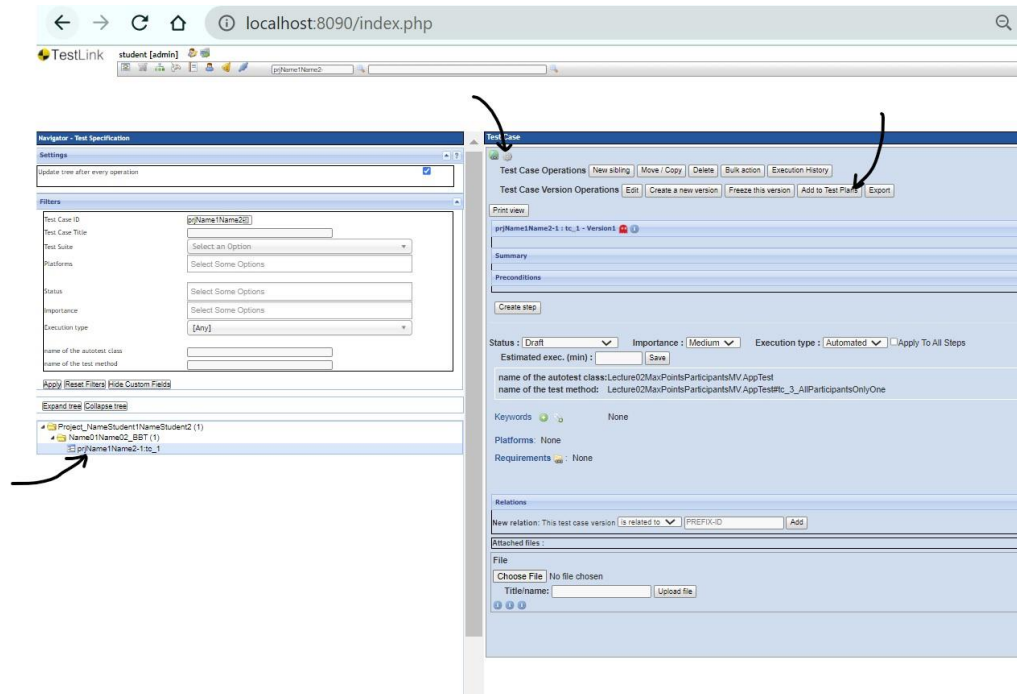


6. Create Test Cases - STEPS

- Select your test suite in navigation tree (left side screen)
- In the Test Suite Operations/ Test Case Operations panel click on the Create button (from Test Case Operation). Figure 6a and Figure 6b. Remember to modify Execution type to automated.



7. Add a Test Case to a Test Plan



Steps for using Jenkins

1. User

Accessing Jenkins

<http://localhost:8080/>

user:student

password:student

2. Create a new Job

Each team will create his/her own **Job (for the TestPlan created in Testlink)**, the name of the job will be composed of: groupNumber followed by the names of the team members concatenated with the word Job. (93X_Name01Name02_Job)

- for example, for one of the team in group 931 Pop and Popescu the name of the job will be:
 - 931_PopPopescu_Job.

Create a new Job from other existing job

1. In the Dashboard, click on the +NewItem
2. Use “Copy from”: Job_Lecture02_BBT

localhost:8080/view/all/newJob

Jenkins

Dashboard > All >

Enter an item name

93X_Name1Name2_Job

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (declarative and scripted).

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is a virtual container, items are actually stored in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

Job_Lecture02_BBT

OK

3. Modify in your Job the following elements

a. Click on the created Job and click on the Configure (left side menu)

b. Github link with your Github link

c. Test Project and Test Plan with

- **Lecture01_Demo_BBT** with [Project_NameStudent1NameStudent2](#) (the name of the Project that you created in Testlink)

- **Lecture02_TestPlan_BBT** with [93X_Name01Name02_TestPlan](#) (the name of the Test Plan that you created in Testlink).

localhost:8080/job/93x_Name1Name2_Job/configure

Dashboard > 93x_Name1Name2_Job > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

☐ Discard old builds ?

☒ GitHub project

Project url ?

https://github.com/andreeavescan/2023_Lecture02_BBT/

Advanced...

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced...

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/andreeavescan/2023_Lecture02_BBT.git

localhost:8080/job/93x_Name1Name2_Job/configure

Dashboard > 93x_Name1Name2_Job > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Advanced...

☒ Invoke TestLink

TestLink Configuration

TestLink Version ?

localhost

Test Project Name ?

Lecture01_Demo_BBT

Test Plan Name ?

Lecture02_TestPlan_BBT

Build Name ?

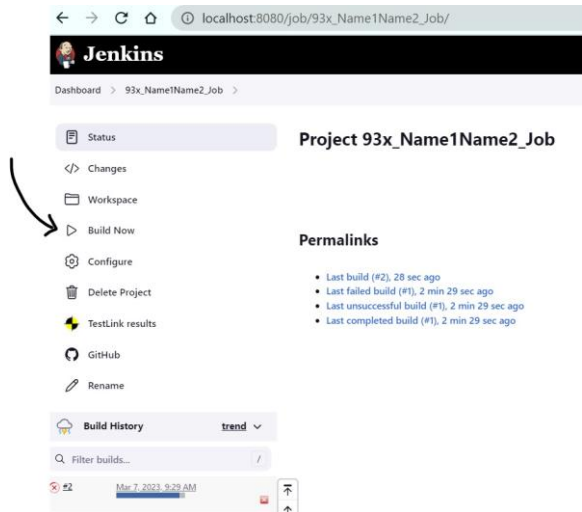
build_\${BUILD_ID}

Custom Fields ?

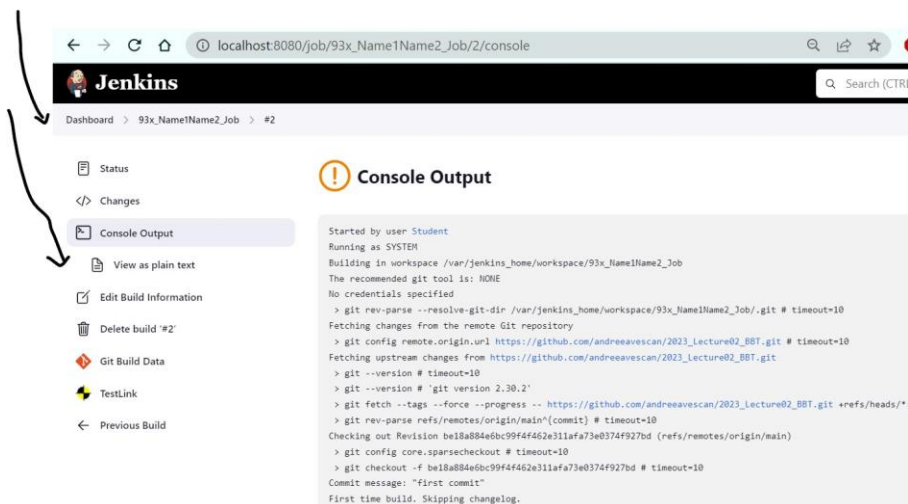
JavaClassName.JavaTestMethodName

4. Build

- Select the Job
- Select Build Now



- Console Output



5. Sending results from Jenkins to Testlink

After successfully running the job, in Testlink it is updated automatically the status of the test cases in Not Run, Passed, Failed, or Blocked.

6. View the state of a Test case in Testlink

In the Execute Tests menu, within the chosen project, eg Projects931, a desired Test Suite is selected, e.g., abie1234_BBT, and a test case.

The screenshot displays the Testlink web interface. On the left, the 'Execute Tests' menu is visible, showing a tree structure of projects and test cases. The selected test case is 'prjName1Name2-1-ic_1'. The main panel shows the 'Test Results on Build build_3' page. The page includes a table with the following data:

Date	Build	Tested by	Status	Exec (min)
03/07/2023 09:39:51	build_3	student	Passed	

Arrows point to the selected test case in the sidebar and the 'Passed' status in the table.