

Test management tools

Jenkins Tutorial

Table of Contents

1. Access Jenkins	2
2. Connect to Jenkins	2
3. Create a new Job.....	2
4. Configure your Job	3
5. Build.....	8
6. Sending results from Jenkins to Testlink	10
7. View the state of a Test case in Testlink.....	10

1. Access Jenkins

<https://www.scs.ubbcluj.ro:9090>

2. Connect to Jenkins

Log in - user your scs_id_user

user: for example, for student Savu Victor Gabriel (**svie1164@scs.ubbcluj.ro**) the user is **svie1164**

password: the password of user **svie1164**

3. Create a new Job

Each student will create his/her own **Job** (**one for each TestPlan created in Testlink**), the name of the job will be the scs id user concatenated with the word Job

- for example, for student Savu Victor Gabriel (**svie1164@scs.ubbcluj.ro**) the name of the Test Plans will be: **svie1164Job_BBT**, **svie1164Job_WBT**, **svie1164Job_IntegrationT**

Create Job - STEPS

- Jenkins Menu
- Select **New Job**
 - Name:
 - scs_user_idJob_BBT
 - scs_user_idJob_WBT
 - scs_user_idJob_IntegrationT
 - Select *Build a free -style software project*
 - click button "ok"

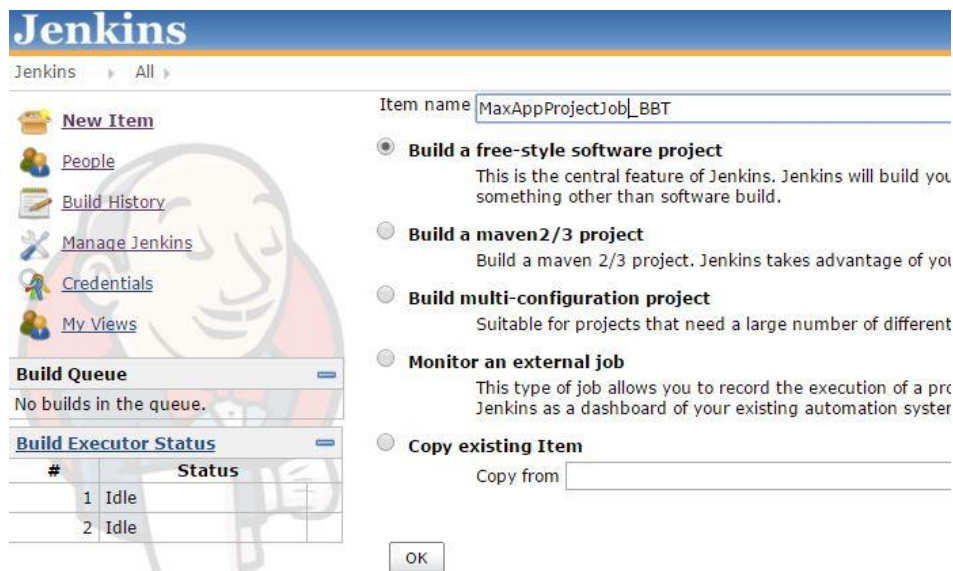


Figure 1. Jenkins – Creating new job

4. Configure your Job

- Jenkins Menu
- Select from the list of Jobs your previously created job
- Select from left menu the "Configure" section



Figure 2. Jenkins – Configure job

- In the **Source Code Management** section - select Subversion Modules
 - on the **Repository URL** section add the URL http://subversion.assembla.com/svn/ssvv2016/user_id
for example, for student Savu Victor Gabriel (svie1164@scs.ubbcluj.ro) the url will be <http://subversion.assembla.com/svn/ssvv2016/svie1164>
 - on the **Repository depth option** select "infinity"
 - on the **Check-out strategy** select "always check out a fresh copy"
 - on **Repository browser** select "Auto"

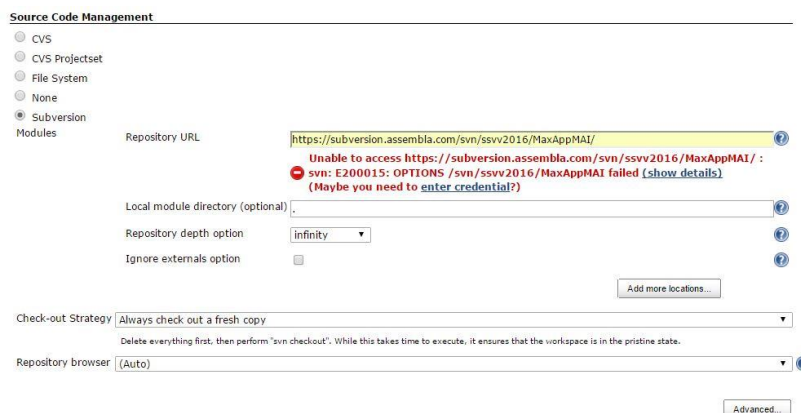
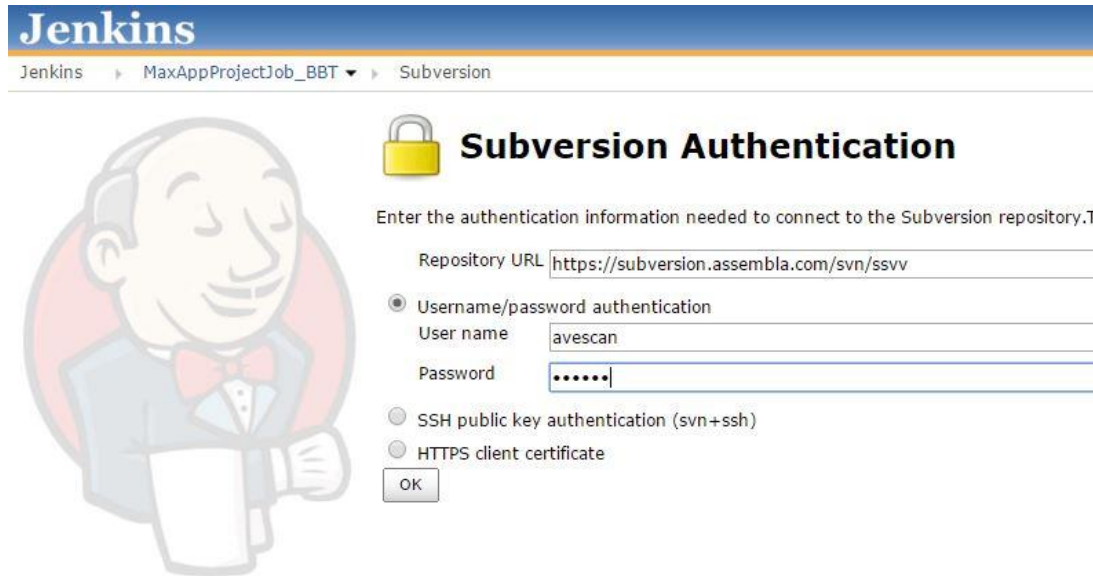


Figure 3. Jenkins – Configure job – url for svn

- enter credentials



The image shows the Jenkins 'Subversion Authentication' configuration page. On the left is a cartoon illustration of a man in a tuxedo. The main heading is 'Subversion Authentication' with a lock icon. Below the heading, it says 'Enter the authentication information needed to connect to the Subversion repository.' The 'Repository URL' is set to 'https://subversion.assembla.com/svn/ssvv'. Under 'Authentication method', 'Username/password authentication' is selected. The 'User name' is 'avescan' and the 'Password' is masked with dots. Other options like 'SSH public key authentication' and 'HTTPS client certificate' are unselected. An 'OK' button is at the bottom.

Figure 4. Jenkins – Configure job – enter credential

- In the **Build** Section
 - In the **Add build steps** select **Invoke top-level Maven targets** and
 - **Maven Version:** mvn
 - **Goals:** compile



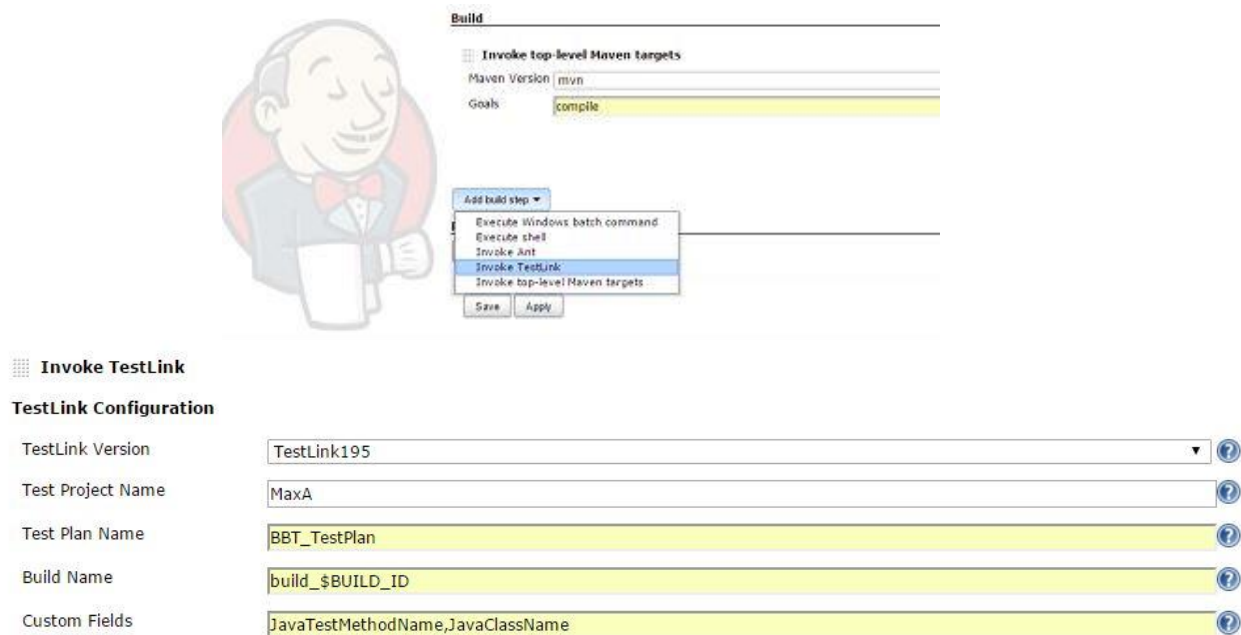
The image shows the 'Build' section of the Jenkins configuration. A dropdown menu for 'Add build step' is open, showing options: 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke TestLink', and 'Invoke top-level Maven targets' (which is highlighted). 'Save' and 'Apply' buttons are at the bottom of the dropdown.



The image shows the 'Build' section configuration for 'Invoke top-level Maven targets'. The 'Maven Version' is set to 'mvn' and the 'Goals' are set to 'compile'. There are 'Advanced...' and 'Delete' buttons on the right.

Figure 5. Jenkins – Configure job – Build-add build step -Maven

- In the **Add build steps** select **Invoke Testlink** and
 - **Testlink Version:** Testlink195
 - **Test Project Name:** groupNumberProject (for example Projects931, the project names from Testlink tutorial)
 - **Test Plan Name:** scs id user concatenated with the word TestPlan concatenated with the type of testing (for example, for student Savu Victor Gabriel (**svie1164@scs.ubbcluj.ro**) the name of the Test Plan will be: **svie1164TestPlan_BBT** or **svie1164TestPlan_WBT** or **svie1164TestPlan_IntegrationT**, the test plan from Testlink tutorial).
 - **Build Name:** build_\${BUILD_ID}
 - **Custom Fields:** JavaTestMethodName,JavaClassName (the custom field created in Testlink, from tutorial)



Build

Invoke top-level Maven targets

Maven Version: myn

Goals: compile

Add build step ▾

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke TestLink
- Invoke top-level Maven targets

Save Apply

Invoke TestLink

TestLink Configuration

TestLink Version: TestLink195

Test Project Name: MaxA

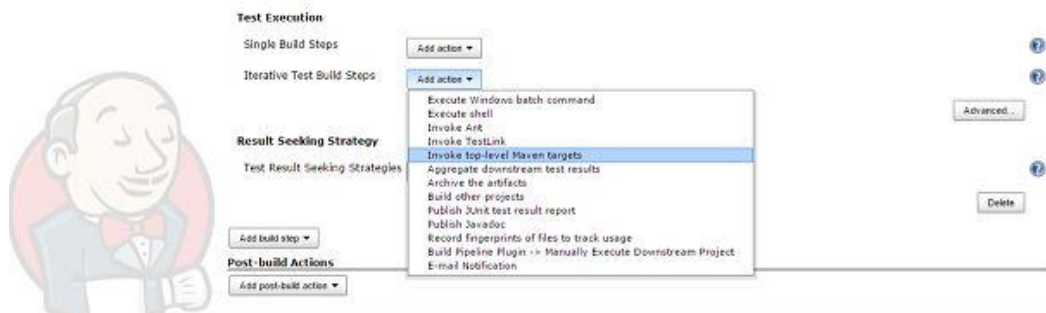
Test Plan Name: BBT_TestPlan

Build Name: build_\${BUILD_ID}

Custom Fields: JavaTestMethodName,JavaClassName

Figure 6. Jenkins – Configure job – Build-add build step -Testlink

- In the **Test Execution** section select **Iterative Test Build Steps**



Test Execution

Single Build Steps: Add action ▾

Iterative Test Build Steps: Add action ▾

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke TestLink
- Invoke top-level Maven targets
- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Publish Javadoc
- Record fingerprints of files to track usage
- Build Pipeline Huger -> Manually Execute Downstream Project
- E-mail Notification

Test Result Seeking Strategies

Test Result Seeking Strategies: Add build step ▾

Post-build Actions: Add post-build action ▾

Advanced ...

Delete

Figure 7. Jenkins – Configure job – Test execution

- In the **Add steps** select **Invoke top-level Maven targets**
 - **Maven Version:** mvn
 - **Goals:** verify

The screenshot shows the Jenkins 'Test Execution' section. On the left, there are tabs for 'Single Build Steps' and 'Iterative Test Build Steps'. An 'Add action' button is visible. The 'Invoke top-level Maven targets' step is selected and expanded. It contains two input fields: 'Maven Version' with the value 'mvn' and 'Goals' with the value 'verify'. To the right of these fields are 'Advanced...' and 'Delete' buttons. There are also several question mark icons on the right side of the interface.

Figure 7. Jenkins – Configure job – Test execution-Maven

- In the **Advances section**
 - **Properties:** test=\$TESTLINK_TESTCASE_JavaTestMethodName

This screenshot shows the same Jenkins 'Test Execution' section as Figure 7, but with the 'Advanced...' button clicked. The 'Invoke top-level Maven targets' step is expanded to show additional configuration options. The 'Maven Version' is 'mvn' and 'Goals' is 'verify'. The 'POM' field is empty. The 'Properties' field contains the text 'test=\$TESTLINK_TESTCASE_JavaTestMethodName'. Below this, there are fields for 'JVM Options' (empty), 'Use private Maven repository' (unchecked checkbox), and 'Settings file' (set to 'Use default maven settings'). Question mark icons are present on the right.

Figure 8. Jenkins – Configure job – Test execution-Maven-properties

- In the **Add steps** select **Execute Shell:**

This screenshot shows the 'Add steps' dropdown menu in Jenkins. The menu is open, displaying several options: 'Execute Windows batch command', 'Execute shell' (which is highlighted in blue), 'Invoke Ant', 'Invoke TestLink', 'Invoke top-level Maven targets', and 'Aggregate downstream test results'. In the background, the 'Test Result Seeking Strategy' section is visible, showing a 'Test Result Seeking Strategy' dropdown and 'Save' and 'Cancel' buttons. The URL at the bottom of the browser window is 'http://www.cs.ubbcluj.ro/avescan/MavenAppProjectJob_SBT/configure'.

Figure 9. Jenkins – Configure job – Test execution-Maven-Add steps

- **Command:**
#!/bin/bash
mv target/surefire-reports/TEST-\$TESTLINK_TESTCASE_JAVAClassName.xml target/surefire-reports/TEST-\$TESTLINK_TESTCASE_JAVAClassName.\$TESTLINK_TESTCASE_ID.xml

Test Execution

Single Build Steps

Iterative Test Build Steps



The screenshot shows the Jenkins configuration page for 'Test Execution'. On the left, there are two tabs: 'Single Build Steps' and 'Iterative Test Build Steps'. The 'Single Build Steps' tab is active. Below the tabs, there is an 'Add action' button. The main configuration area contains two sections: 'Invoke top-level Maven targets' and 'Execute shell'. The 'Invoke top-level Maven targets' section has a 'Maven Version' dropdown set to 'mvn' and a 'Goals' dropdown set to 'verify'. There are 'Advanced...' and 'Delete' buttons to the right. The 'Execute shell' section has a 'Command' text area containing the following script:

```
#!/bin/bash
mv target/surefire-reports/TEST-$TESTLINK_TESTCASE_JAVAClassName.xml target/surefire-reports/TEST-$TESTLINK_TESTCASE_JAVAClassName.$TESTLINK_TESTCASE_ID.xml
```

 There is a 'See the list of available environment variables' link below the command area. On the right side of the configuration area, there are several help icons (question marks) and a 'Delete' button.

Figure 10. Jenkins – Configure job – Test execution-Maven-Execute shell -command

- In the **Result Seeking Strategy** section
 - In the **Add strategy** select **JUnit method name**
 - **Include Pattern:** `**/TEST-*.xml`
 - **Key Custom Field:** `JavaTestMethodName`
 - check **Attach JUnit XML** and **Include test notes**

Result Seeking Strategy

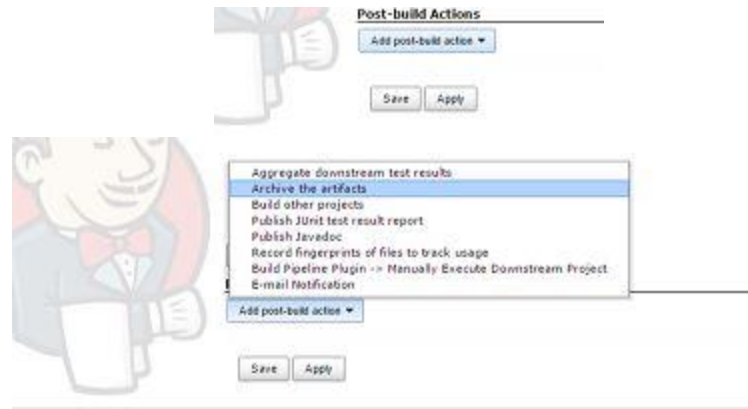
Test Result Seeking Strategies



The screenshot shows the Jenkins configuration page for 'Result Seeking Strategy'. On the left, there is a tab 'Test Result Seeking Strategies'. The main configuration area contains a section titled 'JUnit method name'. It has four fields: 'Include Pattern' with the value '**/TEST-*.xml', 'Key Custom Field' with the value 'JavaTestMethodName', 'Attach JUnit XML' with a checked checkbox, and 'Include test notes' with a checked checkbox. There are help icons (question marks) to the right of each field. At the bottom right, there is a 'Delete' button. At the bottom left, there is an 'Add strategy' button.

Figure 12. Jenkins – Configure job – Result seeking strategy- information

- In the **Post-build Actions** section
 - For **Archive the artifacts**
 - **Files to archive:** `**/TEST-*.xml`



Post-build Actions

Archive the artifacts

Files to archive

Advanced...

Delete

Figure 13. Jenkins – Configure job – Post build actions

- **Click Save button**

5. Build

- Jenkins Menu
- Select from the list of Job your previously created and configured job
- Click on **Build Now** functionality and wait for the build to finished



Figure 14. Jenkins – Build job

- Click on **Build History** functionality and wait for the build to finished



Figure 15. Jenkins – Build job

- Select **Console Output**. You will see if the build was successful or not.

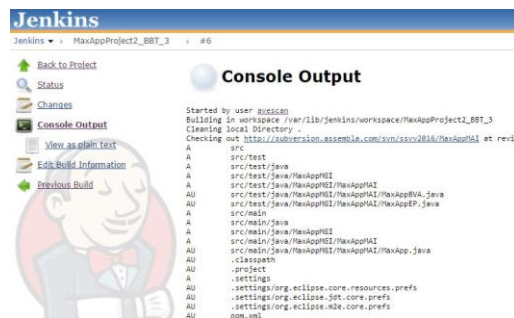


Figure 16. Jenkins – Console output

- If yes, click the **Testlink results** from the left menu to see the result of the tests.



Figure 17. Jenkins – Testlink results

- If selecting again the project and from the Build history, the last build a table with a list of executed test cases is listed.

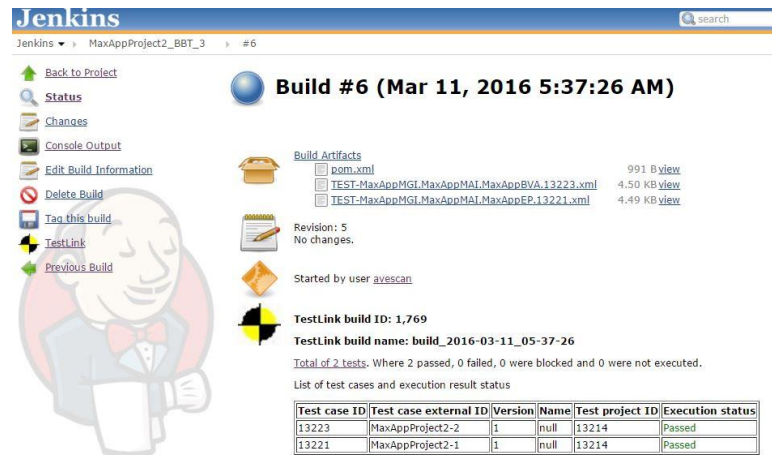


Figure 18. Jenkins – Testlink results for the last build

6. Sending results from Jenkins to Testlink

- After **Jenkins build** (if successful) in the Testlink management tool the status of the test case should be updated to "Passed" or "Failed"

7. View the state of a Test case in Testlink

- In Testlink, select **Test execution** menu

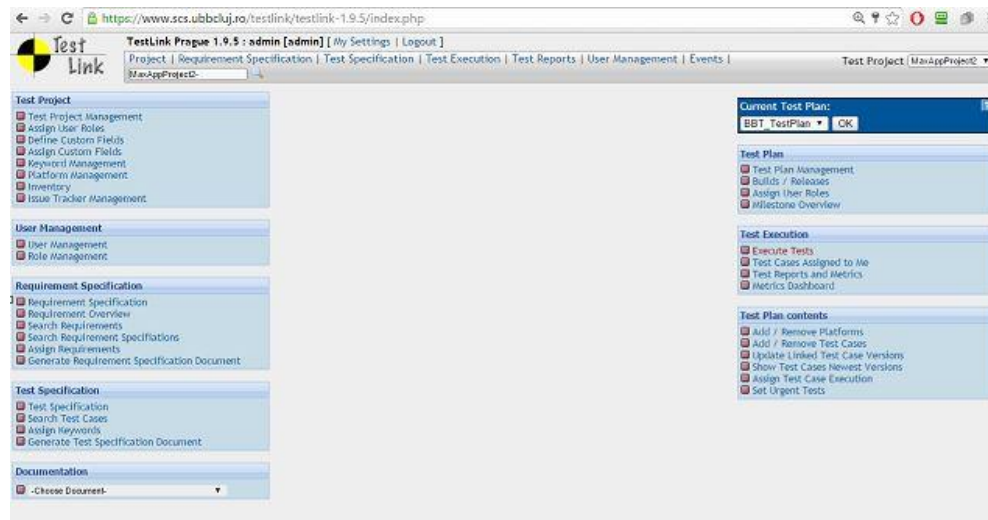


Figure 19. Testlink results from Jenkins

- in your project (groupNumberProject) select your **TestPlan** (scs_user_idTestPlan) and then select your **Test Case**.

The screenshot displays the TestLink web application interface. The top navigation bar includes links for Project, Requirement Specification, Test Specification, Test Execution, Test Reports, User Management, Events, and a dropdown for MaxAppProject2. The main content area is divided into several sections:

- Settings:** Shows the selected Test Plan as 'BBT_TestPlan' and the Build to execute as 'build_2016-03-11_05-37-26'. It also includes options for 'Update one after every operation' and 'Export Test Plan'.
- Filters:** A sidebar with filters for Test Case ID, Test Case Title, Test Suite, Priority, Execution type, Assigned to, Tests, JavaTestMethodname, Result, and an 'on' button for 'Build chosen for execution'.
- Test Results on Build build_2016-03-11_05-37-26:** This section shows the test suite 'MaxAppProject2_BBT/' and the selected test case 'Test Case ID MaxAppProject2-1 :: Version : 1 (TC-SC-1)'. It indicates 'No tester assigned' and shows the last execution (any build) as 'Build : build_2016-03-11_05-37-26' with a status of 'Passed'.
- Execution History Table:** A table with columns: Date, Build, Tested by, Status, Test Case Version, attachments, and Run mode. The first row shows a successful execution on 11/03/2016 at 05:37:54 for build 'build_2016-03-11_05-37-26' by user 'admin'.
- Requirements:** A section showing the requirement '[MaxAppProject2_ReqSpec] MaxAppProject2_ReqSpec_1 : MaxAppProject2_ReqSpec_1'.

Figure 20. Testlink results from Jenkins – for a selected test case