

# Integration with Boozang

- [Overview](#)
- [Main features](#)
  - [Mapping of concepts](#)
- [Flow](#)
- [Setup](#)
- [Import Xray Cucumber Tests into Boozang](#)
- [Automating Cucumber Tests in Boozang](#)
- [Import results to Xray](#)
  - [Jenkins](#)
  - [Command line](#)
    - [Authenticate](#)
    - [Send results to Xray](#)
- [Learn more](#)

## Overview

[Boozang](#) is a codeless testing tool that allows you to define and execute UI/API automated tests without the need to code them. It also supports Cucumber tests and has a modular approach towards testing.

Integration with CI/CD tools is also possible through scripts generated in Boozang to be used in your CI/CD tool.

More details about Boozang [here](#).

6g

+

Home

Xray [master]

Data

Console

Project (\$proj€

+

+ New Module

+ New AI Module

Modules(4)

Features(1)

m5

Insert UserName and Pas sword

+ Add description

+ Add tags

1

-

-

-

-

Updated 1 hour ago

m4

Open Robot Main Page

+ Add description

+ Add tags

1

-

-

-

-

Updated 1 day ago

m7

Validate error page

+ Add description

+ Add tags

1

-

-

-

-

Updated 1 day ago

m6

Validate Welcome Page

+ Add description

+ Add tags

1

-

-

-

-

Xray

(Branch: master-auto-20210923, Project Application Language: English)

Launch Tool

Installation

Team


Details

Requirements


Bugs

Choose the installation option that suits your needs:


**Run The Boozang tool in Chrome**  
Install the Chrome extension and get started testing any site within minutes. Use this option to get familiar with the tool.

  
Install Chrome extension

**I have file system access**  
Install the HTML snippet to test my own site. Get the full benefits of the Boozang tool, such as cross-browser testing, linkable tests and [CI server](#) integration.

  
[Download the Fragment](#)

**My teammate have file system access**  
Email the HTML snippet to a team member so they can install it for me.

  
[Email the Fragment to Team member](#)

## Main features

This integration provides:

- Integration with Xray cloud or Server/DC
- Ability to export Cucumber tests from Xray to Boozang
- Automate the tests in Boozang
- Import the automation results back to Xray

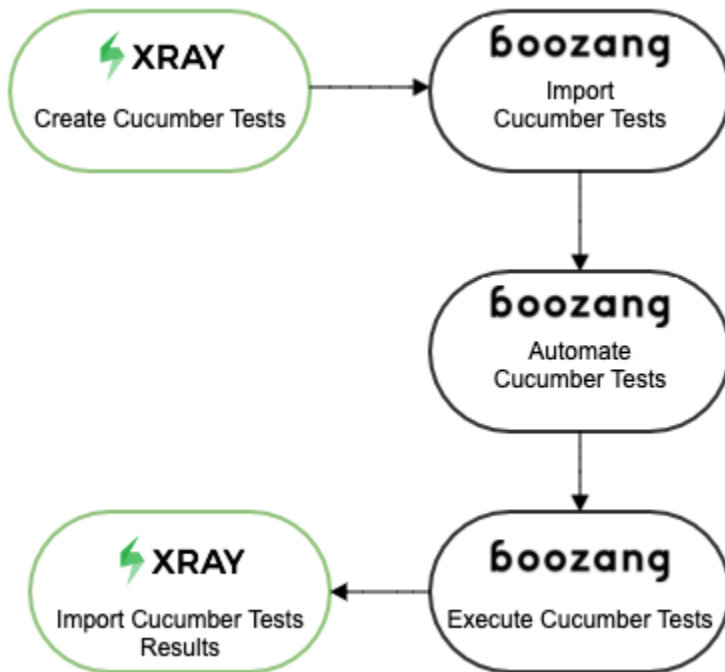
## Mapping of concepts

Boozang	Xray
Test (Cucumber)	Test Case (Cucumber)
Test Execution	Test Execution

## Flow

If you are using Xray as the [master of information](#) (i.e. defining your Cucumber tests and writing those in [Gherkin](#) with Xray), then you will import that specification into Boozang to automate the steps and execute them.

Once the execution is done you will import the test results back to Xray; so the flow will be the below one:



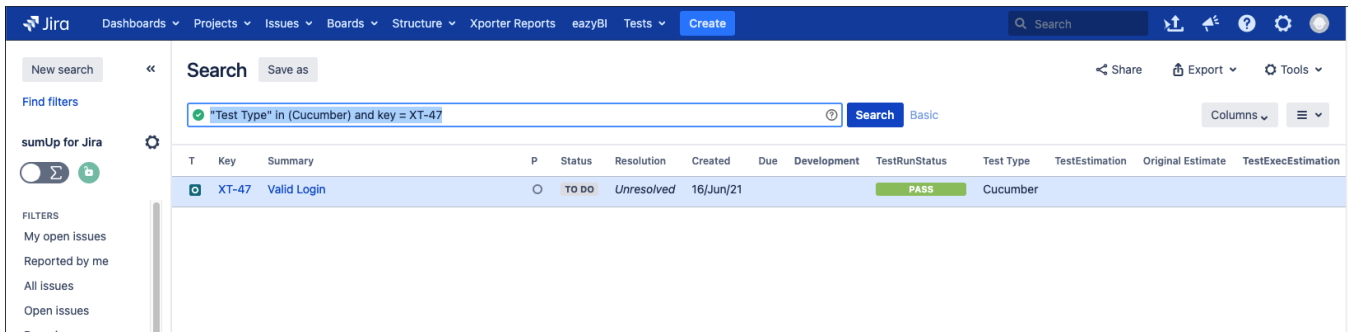
## Setup

Start by having your Cucumber Tests defined in Xray. If you do not have those defined yet, you can follow this [article](#) in order to add your Cucumber tests.

The second step is to define a JQL query, in Jira/Xray, to select those tests in order to be imported into Boozang - the integration is based on this filter. To do so select "Search for issues" from the "Filters" entry in the top menu

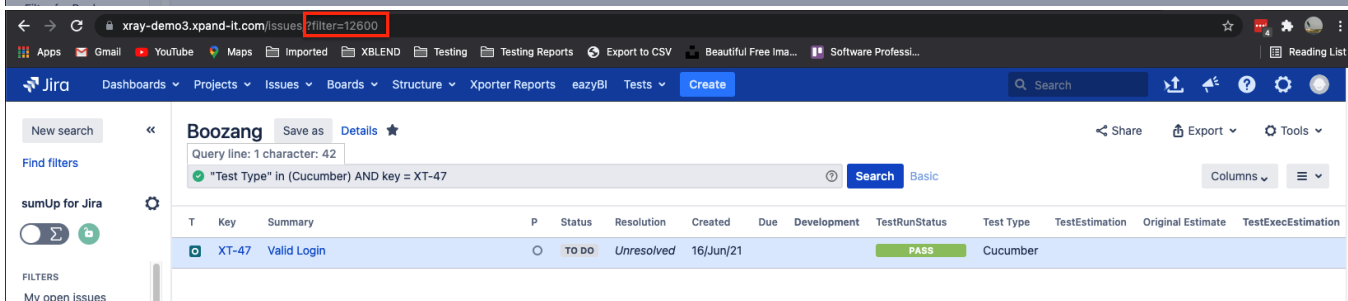
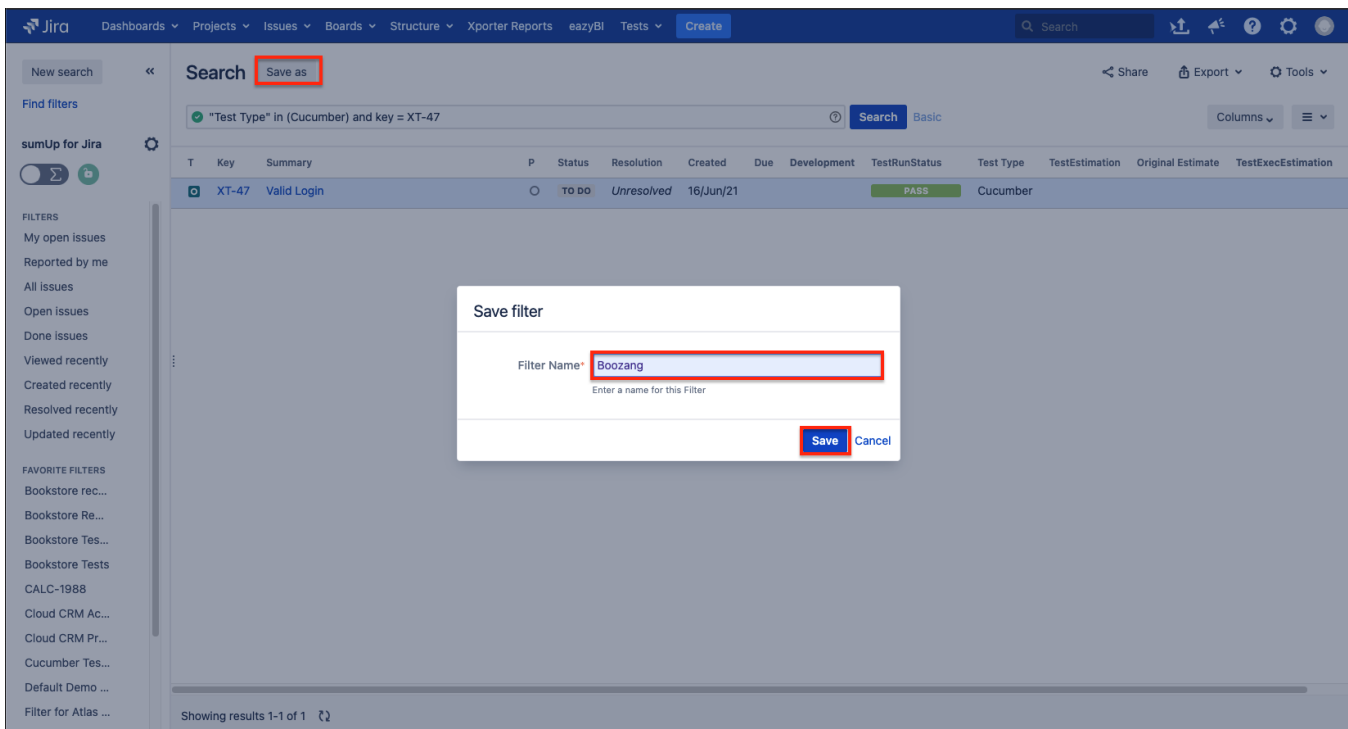
The screenshot shows the Jira/Xray interface. The 'Issues' menu is highlighted, and the 'Search for issues' option is selected. The search results show a list of issues, including 'XT-47 Valid Login', 'XT-145 As a user, I can log in...', 'XT-142 login feature', 'XT-226 Test Login feature', and 'XT-313 Test Execution for Test...'. The 'XT-47' issue is highlighted, and its details are shown in the right pane. The details include the issue key, status (TO DO), resolution (Unresolved), created date (16/Jun/21), and test run status (PASS). The test type is Cucumber.

Introduce the query that will allow you to select the Cucumber Tests that you want to export to Boozang (in order to automate those), in our case it will look like this:



Once you have the filter defined save it using the "Save as" option.

Provide a name and save the id that Jira will associate to your filter. This is especially important as we will use this id in the integration with Boozang to import the Cucumber Tests.



At this stage we have defined the Cucumber Tests in Xray and created a filter to extract those fields, so it is now time to switch to the Boozang application.

Access [Boozang](#) via your region and create a new project designated for your test automation. In the example below, an Xray project was created. Next, open the Boozang tool by selecting the "Launch Tool" option:

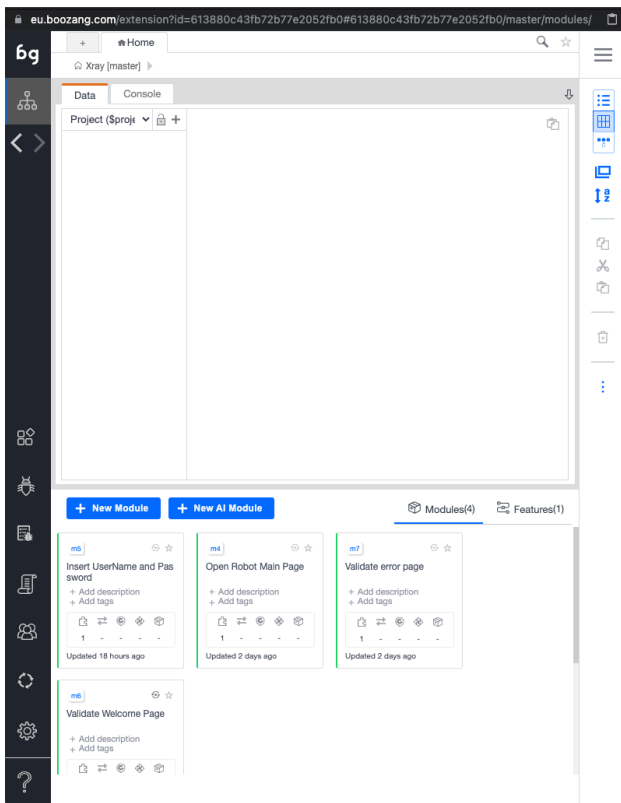
## Welcome Cristiano Cunha

[Create new Project](#)

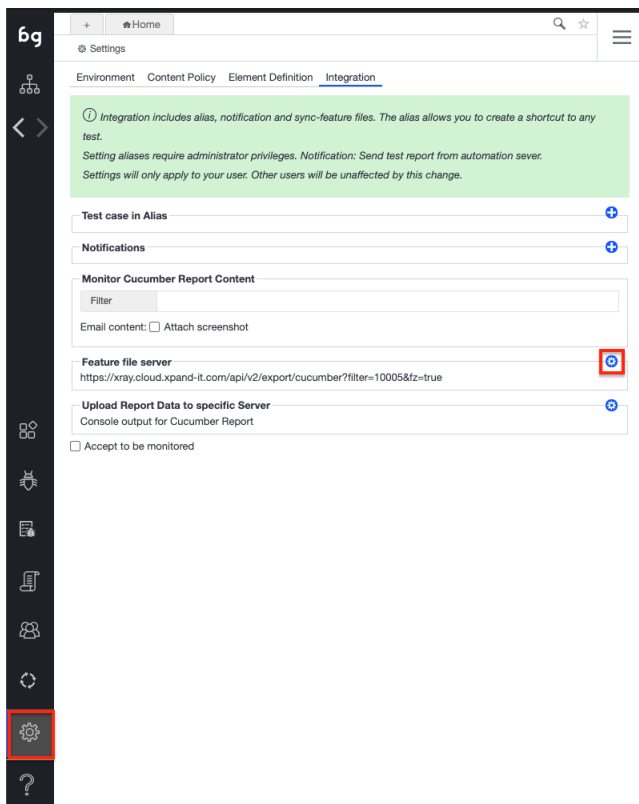
This is **Region Europe**. Select a project below to view installation instructions, manage your project team, view bugs or add project versions.

[Xray](#)[Launch Tool](#)

This will open the tool unless it's the first time you press this option. If so, it will take you to the installation option of the Boozang AI Chrome extension and after you've installed it you'll be taken to the tool.



In order to configure the integration with Xray, we must access the main page and select the configuration option in the bottom left-hand corner and then configure the *Feature File Server* by clicking on the configuration icon next to it



This will open a configuration pop up with the different integrations available in Boozang, in our case we are going to choose the "Jira/Xray" option

### LOAD FEATURES FROM VCS

Type	Jira / Xray	▼
File List Url	https://xray-demo3.xpand-it.com/rest/raven/1.0/export/test?filter=12600&fz=true	
Token	asd:asd	
Client ID		
Client Secret		
Match File	*.feature	
<input checked="" type="checkbox"/> In Zip		

Done

Check

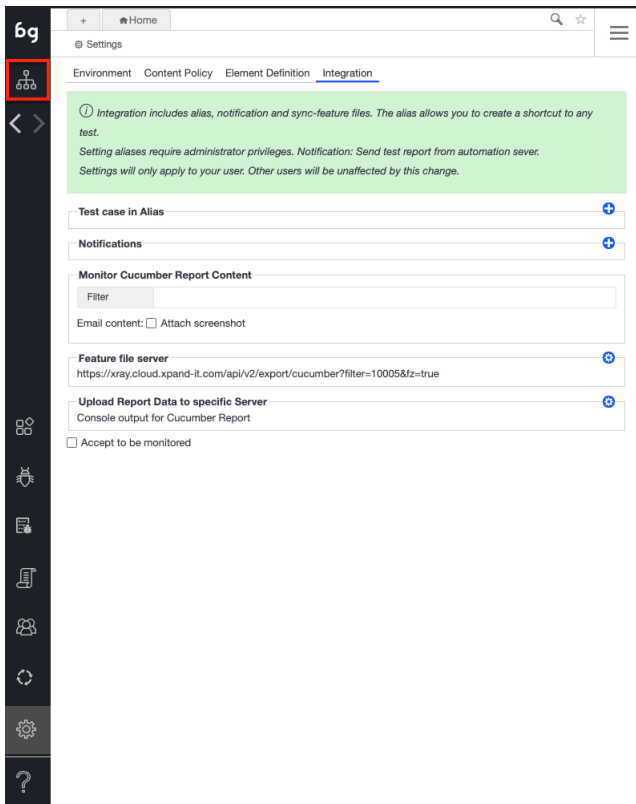
Cancel

In more detail:

- Type: Choose Jira/Xray option
- File List Url: Replace by your Jira instance URL plus "/rest/raven/1.0/export/test?filter" and the JQL filter that we have saved in Jira/Xray
- Token: Insert the username and password for your Jira instance
- Client ID: Leave blank
- Client Secret: Leave blank
- Match File: Leave the "\*.feature"

You can select Check to validate that the configuration is OK, and once everything is done click on Done.

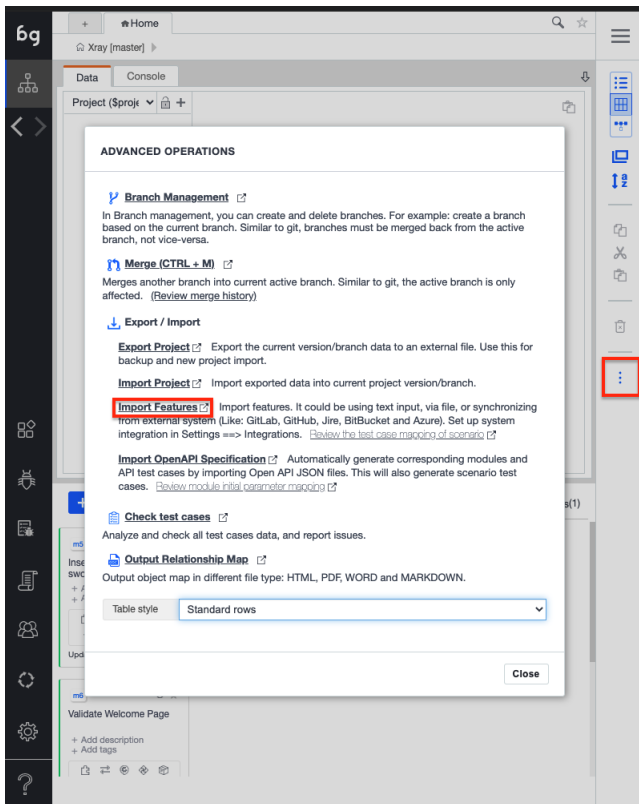
Now get back to the main page of Boozang application by selecting the first option in the left-hand side menu.



## Import Xray Cucumber Tests into Boozang

Now that the configuration is set, we can export the Cucumber tests from Xray and import them into Boozang. To do so, select the option in the right-hand side menu as shown in the screenshot below, and choose "Import Features" option in the pop up that comes up.





Another pop up will appear asking you to choose the import type, in our case, we will choose "Sync from server" and select Load.

**CONFIRM**

☐ By Text

☐ By File

☒ Sync from server

Load

Cancel

At this stage Boozang will connect to Xray and list all the requirements that are covered by the Cucumber tests present in the filter you have provided. In this case, two Cucumber Tests are covering the XT-45 User Story.

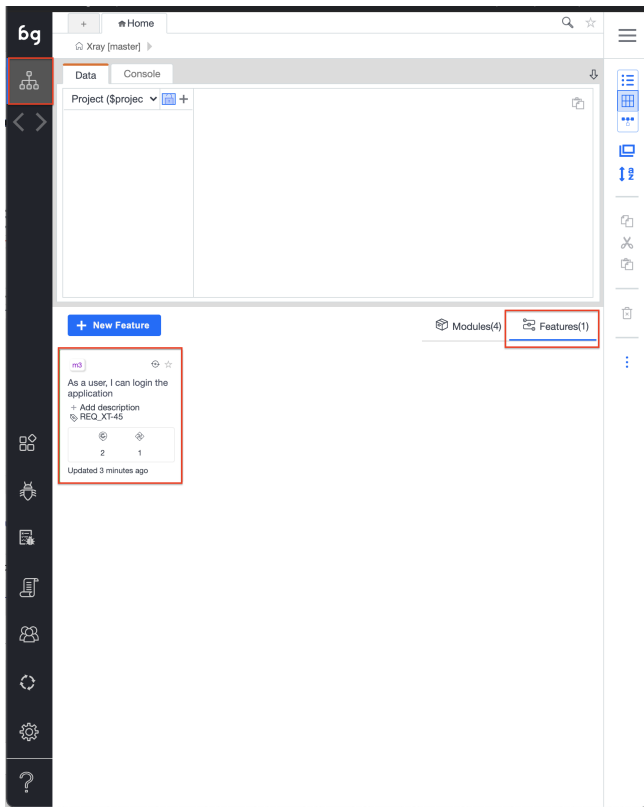
The screenshot shows the Xray web interface. At the top, the breadcrumb navigation includes 'Xray Tutorials' and 'XT-47', with 'Valid Login' highlighted. Below this is a toolbar with buttons for 'Edit', 'Comment', 'Assign', 'More', 'To Do', 'In Progress', 'Done', and 'Admin'. The main content area is divided into two columns. The left column contains a 'Test Details' section with the following information: Type: Cucumber, Scenario Type: Scenario, and Scenario: Given browser is opened to login page, When user "demo" logs in with password "mode", Then welcome page should be open. Below this are sections for Pre-Conditions, Test Sets, Test Plans, Test Runs, Attachments, and Issue Links. The 'Issue Links' section shows a link to 'XT-45 As a user, I can login the application'. The right column contains a 'Dates' section with 'Created: 16/Jun/21 2:17 PM' and 'Updated: 17/Jun/21 10:00 AM', and an 'Agile' section with a 'View on Board' link. At the bottom, there is a 'Structure' section.

The pop up it will show a preview of the requirement and where it will import the Tests from.

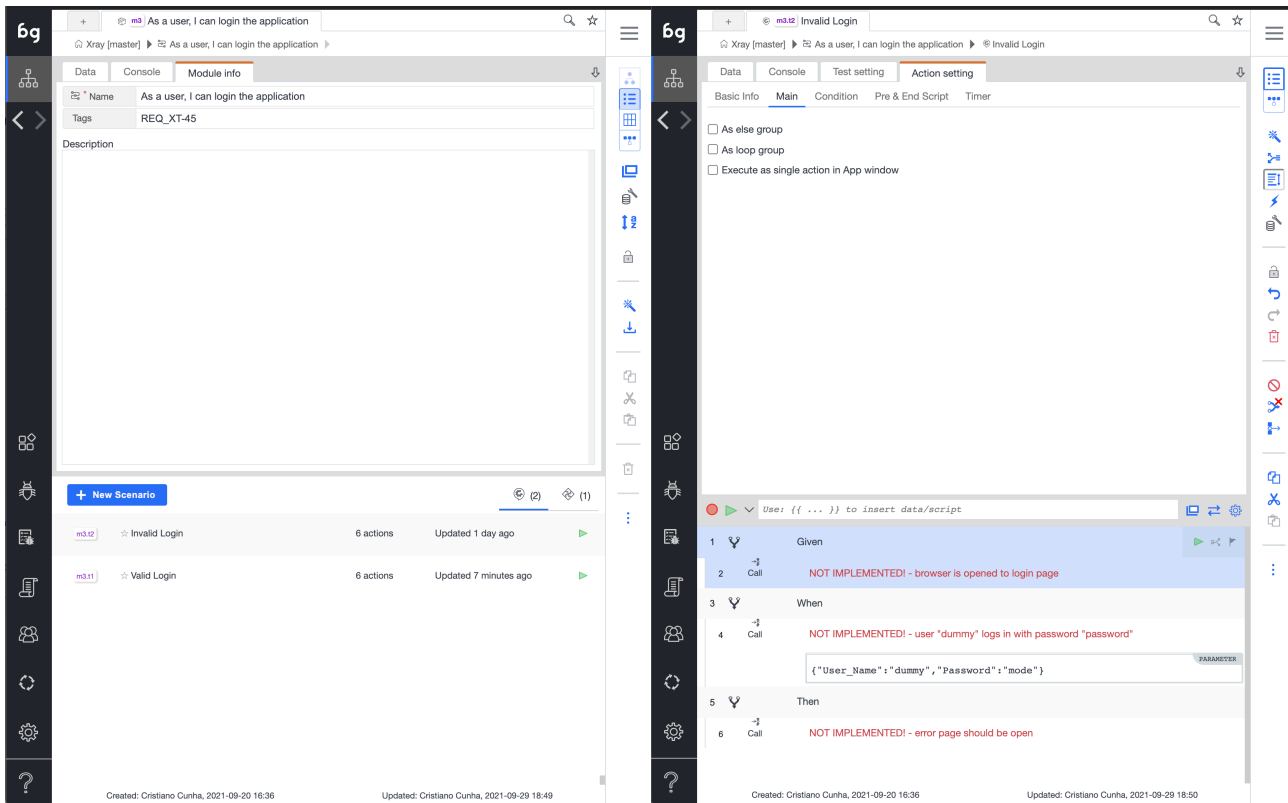
The screenshot shows the Boozang web interface. A modal dialog titled 'PREVIEW' is open, displaying a list of requirements. The requirements are: 'All' and 'As a user, I can login the application'. The 'As a user, I can login the application' requirement is selected. Below the list are buttons for 'Start' and 'Cancel'. The background shows the Boozang interface with a sidebar and a main content area.

When selecting "Start", Boozang will show you the items it will create and proceed with the creation of the features.

Now that the features have been imported into Boozang, the next step would be to implement the code that will be executed to perform the actual validation. Head back to the main page and review the Features created.



When clicking on *Feature*, the actual Tests that were imported will come up, and selecting either of them will show the detailed view of the Tests with the steps that need to be automated - displayed in red below.



# Automating Cucumber Tests in Boozang

There are several ways to automate tests and link them to steps in Boozang however for this section, we will focus on one. For other options more suited to your requirements, we suggest taking a look at Boozang's documentation.

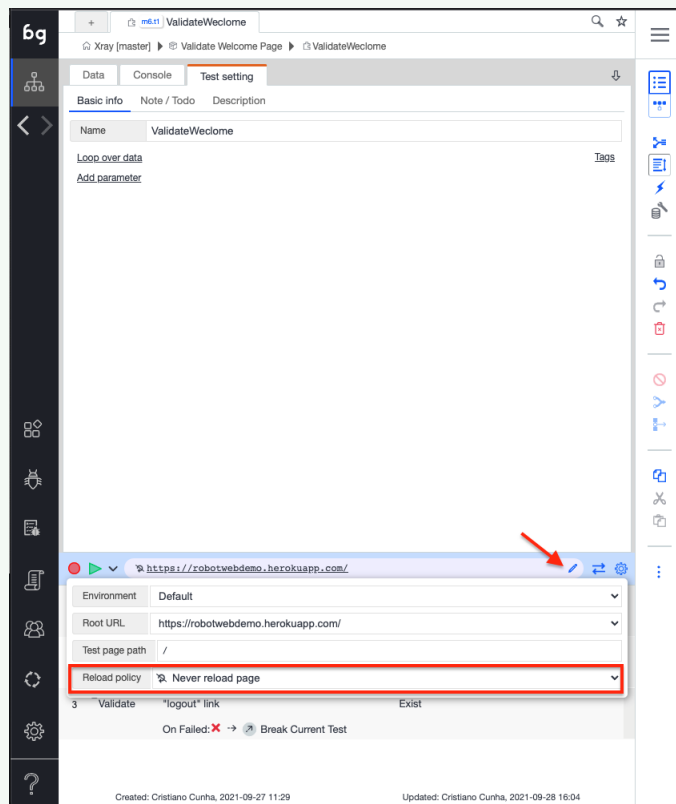
For this example, what matters is to have the steps automated so that they can be executed and produce a report of the execution which will be imported into Xray.

The approach is to create new Modules with a test in them that will represent each step in the Cucumber scenario, that way Tests can be reused in other scenarios with the same description.

You can use the recording capability of Boozang to create each Test. The recorder allows you to define the Test steps and the validations required for each Test.

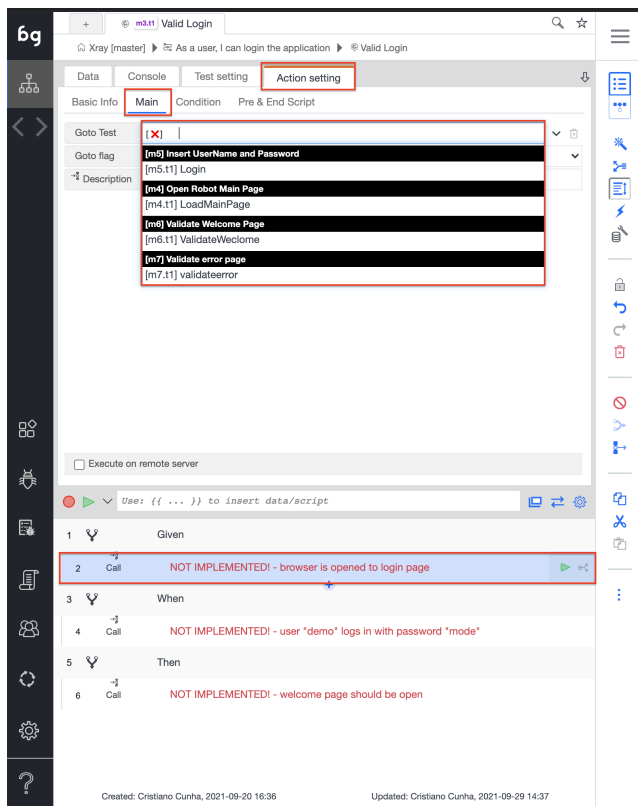
## Tip

In order to reuse some Test steps, you must assure that when the Test starts it will not reload the URL but start where the last Test have ended. To do that, set the Parameter inside the "Reload Policy" of the detail of the Test to "Never".

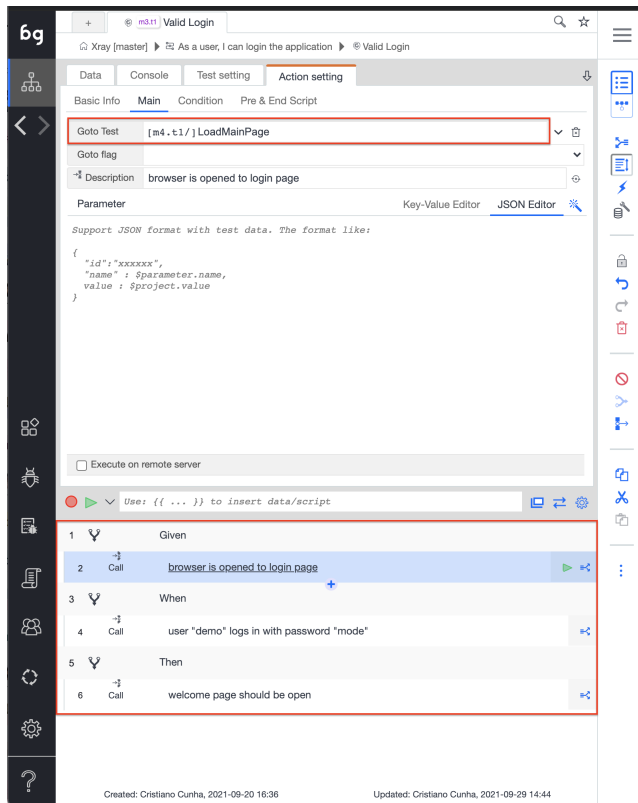


Once the Modules and Tests are ready, they'll need to be linked to each Step in the Cucumber Test. In order to achieve this, access the details of each Cucumber Test and click in the "NOT IMPLEMENTED! -..." which will load in the upper part of the application, under the *Main* submenu of the "Action Setting" tab.

Click on the "Goto Test" entry which Test will be associated to this step via a dropdown available with a list of tests to choose from:



The final result looks like the following screenshot, where all the steps have automated Tests associated and are ready to be executed.



Once you have associated all steps, you can validate that everything is running as expected by clicking on the *Play* button. This option will execute the test and report the results back.

When executing the Tests, Boozang will open up a new browser window where you can follow the Test execution and report back to the Test window.

The screenshot displays the Boozang test runner interface, which is used for creating and executing tests. The interface is divided into several sections:

- Test Setting:** The top section shows the test name "Valid Login" and the test data. The data is a JSON object: 

```
{ 1: { 2: "User_Name": "demo", 3: "Password": "asdaad", 4: "Password1": "mode" 5: } }
```
- Current Test:** The middle section shows the test steps in a Gherkin format:

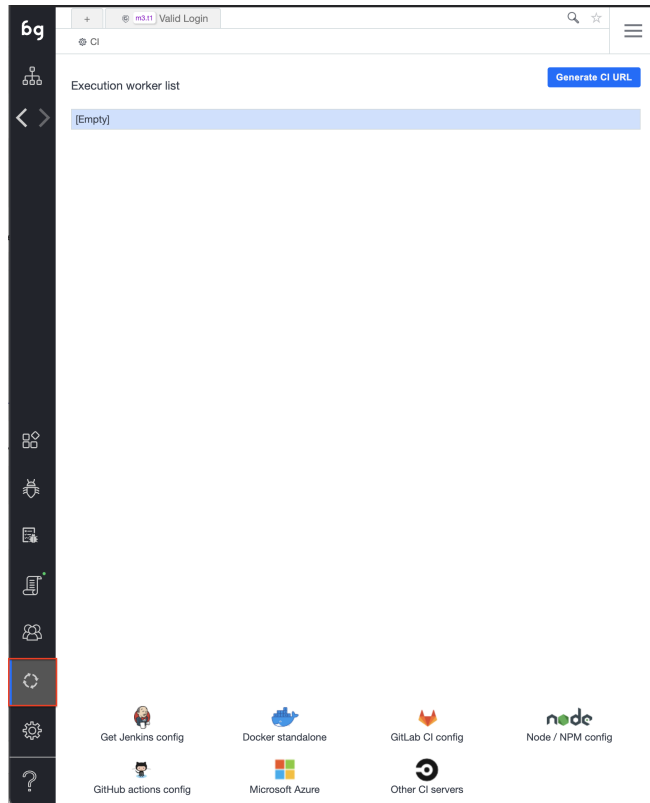
```
1 Given 2 Call browser is opened to login page 3 When 4 Call user "demo" logs in with password "mode" 5 Then 6 Call welcome page should be open
```
- Test Execution:** The bottom section shows the test execution results. The test is marked as "Passed" (green checkmark). The execution log shows the following steps:
  - 1. Given
  - 2. Call browser is opened to login page
  - 3. When
  - 4. Call user "demo" logs in with password "mode"
  - 5. Then
  - 6. Call welcome page should be open

The test execution results are displayed in a table format, showing the step number, the step type (Given, When, Then), the step description, and the step status (Passed, Failed, Skipped).

# Import results to Xray

Finally, the Tests which have been exported from Xray into Boozang are automated. The results from the executions have to be pushed back into Xray. This can be achieved via Boozang's CI where the ability to generate the necessary scripts for different CI/CD tools that will execute the Boozang Test Runner is available.

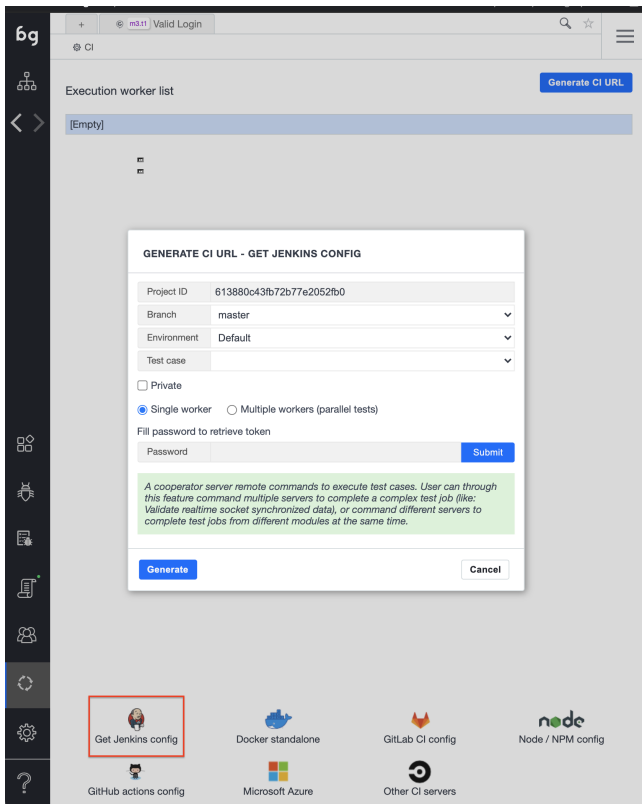
In Boozang, access the CI entry present in the bottom of the left menu.



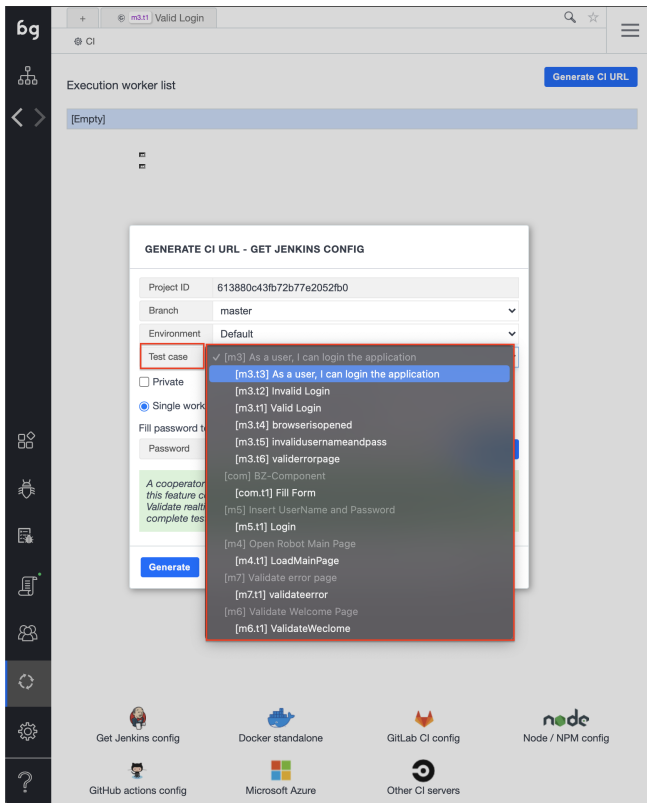
## Jenkins

In this example, and to take advantage of the Xray Jenkins Plugin available, the script needed to include in Jenkins was generated by clicking the bottom link "*Get Jenkins config*".

A new pop up will appear requiring details to generate the script.

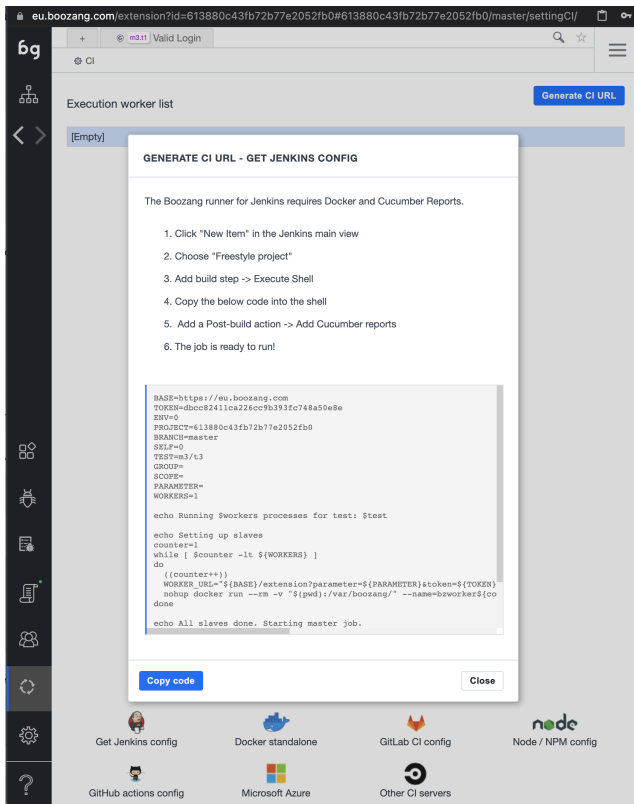


The majority of the options will be the default ones, let's choose the "Test case" to be the feature we have automated (from the drop down list).

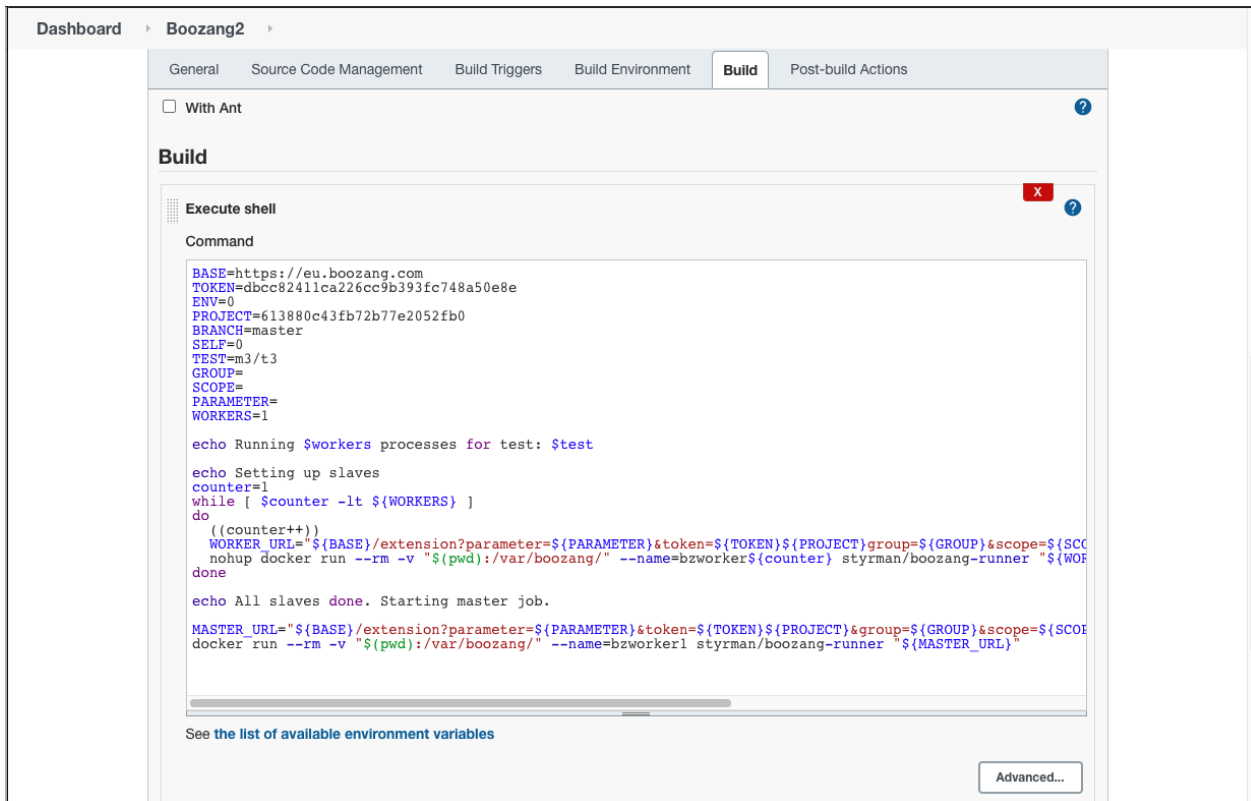


Fill in the password, click "Submit" and click on "Generate". This will produce the necessary script that we will be copied and pasted into Jenkins.





In Jenkins, we created a simple Project where an additional step was added to insert the script generated. More specifically by adding a "Execute shell" in the build step section and inserting the generated code there.



#### Tip

If you have defined several Tests the output of the execution will generate one report per Test. In order to upload the results, the results need to be merged into a single report. This can be achieved by using a tool that will merge two Cucumber reports into one.

You can find that tool [here](#), the usage is simple:

```
cucumber-json-merge-multiworker report_cucumber-m3-t1.json report_cucumber-m3-t2.json
```

After the execution, you can find the report in the default output file "*merged-test-results.json*" or specify the name of the output in the parameter "-o".

The last step, before executing the pipeline, is to add the Xray [Plugin](#) to import the results back to Xray. First, make sure you have previously installed the [plugin](#) and in your pipeline, just add in the Post Build Actions "*Xray: Results Import Task*" and configure properly with:

- the Jira/Xray instance that you will use to upload the results
- the Format of the results file you want to import, in our case Cucumber JSON
- The path and name of the file to be uploaded

X

**Xray: Results Import Task**

Jira Instance

https://xray-demo3.xpand-it.com/

Format

Cucumber JSON

Parameters

Execution Report File (file path with file name)

/var/jenkins\_home/workspace/Boozang2/report\_cucumber-m3-t1.json

☒ Import in parallel

Import all results files in parallel, using all available CPU cores.

[Click here for more details](#)

Once this pipeline is built, it will execute the tests in Boozang and import the results into Xray as you can verify in the "*Console Output*" of the build:

Dashboard > Boozang2 > #31

274: go to close, status: doing  
275: BZ-LOG: report all tasks status,  
276: post data from 552,1444  
277: method: updateServerStatus, current tasks: 0  
278: <div>+++</div> /api/coop/ <div>+++</div> without response  
279: call all worker to close  
280: post data from 1448  
281: method: taskDone, current tasks: 0  
282: <div>+++</div> /api/coop/ <div>+++</div> without response  
283: task-done  
284: One-Task Completed!  
The test failed. Docker return code set to 1.  
Build step 'Execute shell' marked build as failure  
Archiving artifacts  
[CucumberReport] Using Cucumber Reports version 5.6.0  
[CucumberReport] JSON report directory is ""  
[CucumberReport] Copied 0 properties files from workspace "/var/jenkins\_home/workspace/Boozang2" to reports directory  
"/var/jenkins\_home/jobs/Boozang2/builds/31/cucumber-html-reports/.cache"  
[CucumberReport] Copied 2 files from workspace "/var/jenkins\_home/workspace/Boozang2" to reports directory  
"/var/jenkins\_home/jobs/Boozang2/builds/31/cucumber-html-reports/.cache"  
[CucumberReport] Processing 2 json files:  
[CucumberReport] /var/jenkins\_home/jobs/Boozang2/builds/31/cucumber-html-reports/.cache/report\_cucumber-m3-t1.json  
[CucumberReport] /var/jenkins\_home/jobs/Boozang2/builds/31/cucumber-html-reports/.cache/report\_cucumber-m3-t2.json  
[CucumberReport] Found 33.333333 failed steps, while expected not more than 0.000000 percent  
[CucumberReport] Build status is left unchanged  
Starting XRAY: Results Import Task...


<div>#####</div>  
<div>### Xray is importing the execution results ###</div>  
<div>#####</div>  
File: /var/jenkins\_home/workspace/Boozang2/report\_cucumber-m3-t1.json  
Starting to import results from report\_cucumber-m3-t1.json  
Response: (200) {"testExecIssue":{"id":"21831","key":"XT-315","self":"https://xray-demo3.xpand-it.com/rest/api/2/issue/21831"},"testIssues":{"success":  
[{"id":"21257","key":"XT-47","self":"https://xray-demo3.xpand-it.com/rest/api/2/issue/21257"}]}, "infoMessages":["Could not make transition from workflow  
status <b>To Do</b> to workflow status <b>Resolved</b>."]}  
Successfully imported Cucumber JSON results from report\_cucumber-m3-t1.json  
XRAY\_IS\_REQUEST\_SUCCESSFUL: true  
XRAY\_RAW\_RESPONSE: {"testExecIssue":{"id":"21831","key":"XT-315","self":"https://xray-demo3.xpand-it.com/rest/api/2/issue/21831"},"testIssues":{"success":  
[{"id":"21257","key":"XT-47","self":"https://xray-demo3.xpand-it.com/rest/api/2/issue/21257"}]}, "infoMessages":["Could not make transition from workflow  
status <b>To Do</b> to workflow status <b>Resolved</b>."]}  
XRAY\_TESTS: XT-47  
XRAY\_ISSUES\_MODIFIED: XT-315;XT-47  
XRAY\_TEST\_EXECS: XT-315  
Finished: FAILURE</div>


REST API


Jenkins 2.303.1


We can see the response from Xray where we can find the Test Execution created for this execution.

If you want to see the results in Jenkins, install the [Cucumber Reports Plugin](#) that will automatically process and generate a view in Jenkins for you to go over the results.


**Jenkins**


**1**


 cristiano cunha


 log out


Dashboard > Boozang2 > #26


 Back to Project


 Status


 Changes


 Console Output


 Edit Build Information

 Delete build '#26'


 Cucumber reports

 Open Blue Ocean

 Previous Build


 **Build #26 (Sep 28, 2021, 5:41:04 PM)**


[Keep this build forever](#)

 add description

Started 20 hr ago

Took **38 sec**

 No changes.

 Started by user [cristiano.cunha](#)

Cucumber Report

Jenkins

Previous results

Latest results

Features

Tags

Steps

Trends


Failures

Project	Number	Date
Boozang2	26	28 Sep 2021, 17:41

### Features Statistics

The following graphs show passing and failing statistics for features

#### Scenarios



Feature	Steps						Scenarios			Features	
	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
[m3] As a user, I can login the application	5	1	0	0	0	6	1	1	2	16.313	Failed
	5	1	0	0	0	6	1	1	2	16.313	1
	83.33%	16.67%	0.00%	0.00%	0.00%		50.00%	50.00%			0.00%

generate Cucumber HTML reports via: [Jenkins Plugin](#) | [Standalone](#) | [Sandwich](#) | [Maven](#)

If we verify the Test Execution that was created in Xray, the uploaded results with the total number of Tests and the Overall Execution Status is available - in this case, it's not relevant since there is one Test but if they were several Tests it will be the overall result of all the Tests.

**Execution results [1632938223874]**

**Details**  
 Type: Test Execution  
 Priority: Trivial  
 Labels: None  
 Test Plan: None  
 Test Environments: None  
 Status: TO DO (View Workflow)  
 Resolution: Unresolved

**Description**  
 Click to add description

**Tests**  
 Add Tests

**Overall Execution Status**  
 1 FAIL

**Total Tests: 1**  
 Filter(s)

Rank	Key	Summary	Test Type	#Req	#Def	Assignee	Dataset	Status
1	XT-47	Valid Login	Cucumber	1	0	Xpand IT Admin		FAIL

Showing 1 to 1 of 1 entries

**Attachments**

Click on the Details button (below the red arrow) will give users a better understanding of the characteristics of the execution since it navigates to the execution panel of the Test Execution with the following information:

- The requirement that is covered by the Test, in this case, XT-45
- The scenario definition
- The details of the execution of each step

**Valid Login**

**Test Issue Links**  
 XT-45 As a user, I can login the application

**tests**  
 Test Type: Cucumber  
 Scenario Type: Scenario  
 Scenario:  
 1 Given browser is opened to login page  
 2 When user "demo" logs in with password "mode"  
 3 Then welcome page should be open

**Results**

Context	Duration	Status
-	9 sec	FAIL

**Steps**

Step	Duration	Status
Given Browser is opened to login page	2510.000 ms	PASS
When User "demo" logs in with password "mode"	2238.000 ms	PASS
Then Welcome page should be open	4537.000 ms	FAIL

Element is not found or syntax error exists in the code  
 ["B2.TW.document",

## Command line

If Jenkins (or another CI/CD tool) is not part of your workflow you can use the Xray [API](#) to import the result back into Xray through the command line. To do so, please follow the next steps:

- Authenticate to obtain the token
- Send the results to Xray using the token

## Authenticate

The authentication in the Server/DC version is done using the username and password in every request, so no special step to perform before executing the import request.

## Send results to Xray

Finally to upload the results back into Xray you must use the following request:

```
curl -H "Content-Type: application/json" -X POST -u admin:admin --data @"xrayResults.json" 'https://xray-demo3.xpand-it.com/api/v2/import/execution'
```

When successful, the answer will have information of the Test Execution created.

We can see the detailed results in Xray:

The screenshot displays the Xray web interface. On the left is a sidebar with navigation links: XT board, Backlog, Active sprints, Releases, Reports, Issues, Components, Structure, Xray Reports, Xray Test Repository, Xray Test Plan Board, Automated Steps Library, Add-ons, PROJECT SHORTCUTS, Add a link to useful information for your whole team to see., Add link, and Project settings. The main content area is titled 'Valid Login' and shows 'Test Execution: XT-315 / Test: XT-47'. Below this, the 'Test Issue Links' section highlights 'XT-45 As a user, I can login the application'. The 'tests' section shows 'Test Type: Cucumber' and 'Scenario Type: Scenario'. The 'Scenario' section lists three steps: 1. Given browser is opened to login page, 2. When user 'demo' logs in with password 'mode', and 3. Then welcome page should be open. The 'Results' section shows a table with columns 'Context', 'Duration', and 'Status'. The table has one row for the scenario, with a duration of 9 sec and a status of 'FAIL'. Below this, the 'Steps' section shows a table with columns 'Steps', 'Duration', and 'Status'. The steps are: 'Given Browser is opened to login page' (2510.000 ms, PASS), 'When User 'demo' logs in with password 'mode'' (2238.000 ms, PASS), and 'Then Welcome page should be open' (4537.000 ms, FAIL). The 'Then' step is highlighted with a red box, and the error message 'Element is not found or syntax error exists in the code' is visible below it.

Context	Duration	Status
-	9 sec	FAIL

Steps	Duration	Status
Given Browser is opened to login page	2510.000 ms	PASS
When User "demo" logs in with password "mode"	2238.000 ms	PASS
Then Welcome page should be open	4537.000 ms	FAIL

Element is not found or syntax error exists in the code

```
[
  "B2.TW.document",
  ...
]
```

## Learn more

- [Boozang home page](#)
- [Boozang features overview](#)
- [Boozang videos](#)

