

# 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](#).

The screenshot shows the Xray [master] application interface. At the top, there is a header bar with a search icon, a star icon, and a menu icon. Below the header is a navigation bar with tabs for "Data" and "Console". The main area displays a "Project (\$projec")" view. On the left side, there is a sidebar with various icons for navigation and management. At the bottom, there are buttons for "New Module" and "New AI Module", and counts for "Modules(4)" and "Features(1)".

**Project (\$projec)**

**Modules(4)** **Features(1)**

**New Module** **New AI Module**

Module ID	Module Name	Description	Last Update
m5	Insert UserName and Pasword	+ Add description + Add tags	Updated 1 hour ago
m4	Open Robot Main Page	+ Add description + Add tags	Updated 1 day ago
m7	Validate error page	+ Add description + Add tags	Updated 1 day ago
m6	Validate Welcome Page	+ Add description + Add tags	Updated 1 hour ago

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

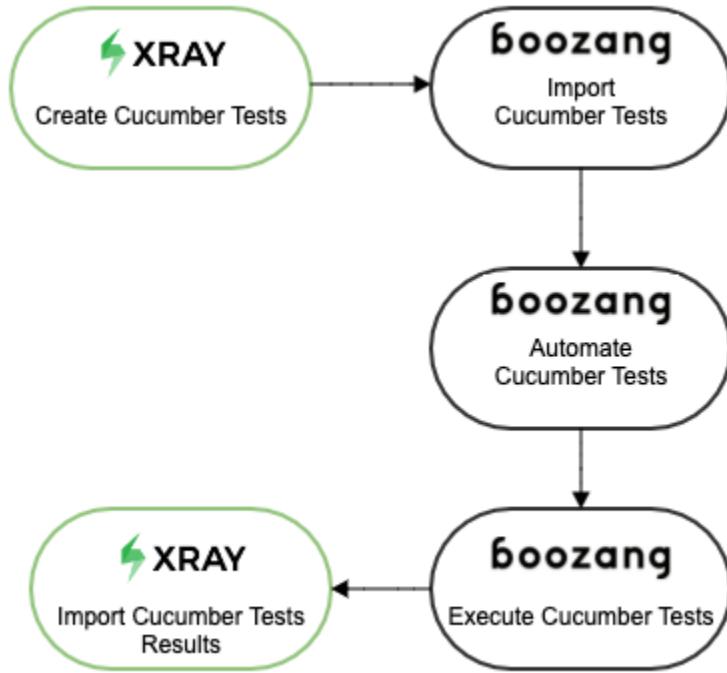
## Mapping of concepts

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 search interface. The 'Issues' dropdown menu is open, and the 'Search for issues' option is highlighted with a red box. The main search results table shows a single issue: XT-47 Valid Login, which is marked as 'Unresolved' and has a status of 'TO DO'. The table includes columns for P, Status, Resolution, Created, Due, Development, TestRunStatus, Test Type, TestEstimation, Original Estimate, and TestExecEstimation.

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:

The screenshot shows the Jira search interface. A search query is entered: "Test Type" in (Cucumber) and key = XT-47. The results table has columns: T, Key, Summary, P, Status, Resolution, Created, Due, Development, TestRunStatus, Test Type, TestEstimation, Original Estimate, and TestExecEstimation. One result is shown: XT-47, Valid Login, status To Do, Unresolved, created 16/Jun/21, test run status PASS, test type Cucumber.

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.

The screenshot shows the Jira search interface with the "Save as" button highlighted. A modal dialog box titled "Save filter" is open, showing a "Filter Name" input field containing "Boozang". The "Save" button is also highlighted.

The screenshot shows the Jira search interface with the filter name "Boozang" entered in the search bar. The results table shows the same issue XT-47 as before.

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

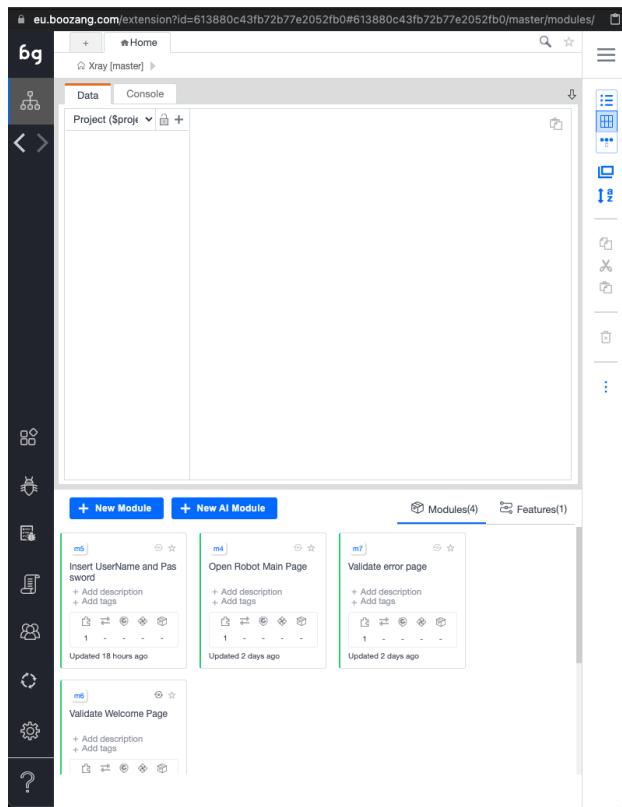
This is Region Europe. Select a project below to view installation instructions, manage your project team, view bugs or add project versions.

**Xray**

**Create new Project**

**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

The screenshot shows the Boozang application's configuration interface. On the left, there's a sidebar with various icons. One icon, which looks like a gear, is highlighted with a red box. The main area is titled 'Integration' and contains several configuration sections:

- Test case in Alias**: An input field with a '+' button.
- Notifications**: An input field with a '+' button.
- Monitor Cucumber Report Content**: Includes a 'Filter' dropdown and an 'Email content' checkbox.
- Feature file server**: An input field containing a URL (`https://xray.cloud.xpand-it.com/api/v2/export/cucumber?filter=10005&fz=true`) with a red box around it.
- Upload Report Data to specific Server**: Includes a 'Console output for Cucumber Report' dropdown and an 'Accept to be monitored' checkbox.

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	<code>https://xray-demo3.xpand-it.com/rest/raven/1.0/export/test?filter=12600&amp;fz=true</code>
Token	asd:asd
Client ID	
Client Secret	
Match File	<code>*.feature</code>
<input checked="" type="checkbox"/> In Zip	
<input type="button" value="Done"/> <input type="button" value="Check"/> <input type="button" value="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"`

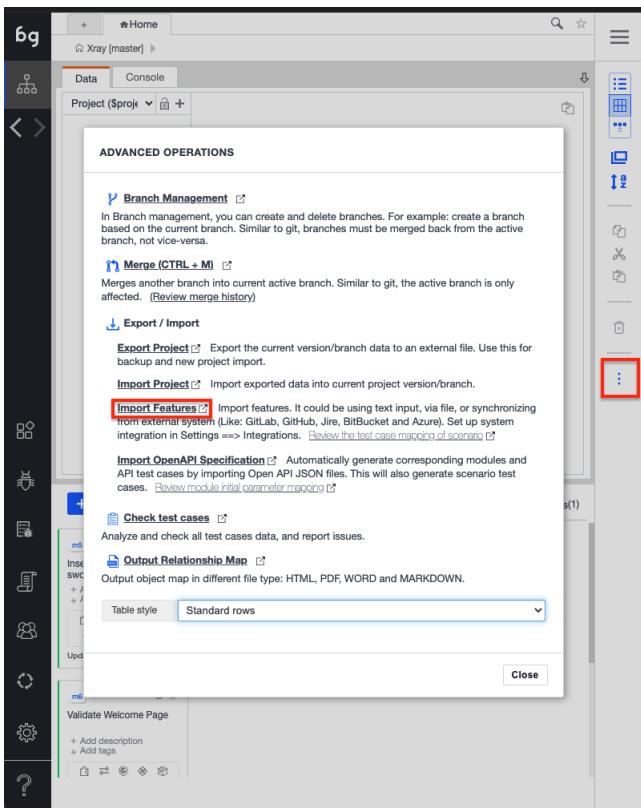
You can select Check to validate that the configuration is OK, and once everything is done click on Done.

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

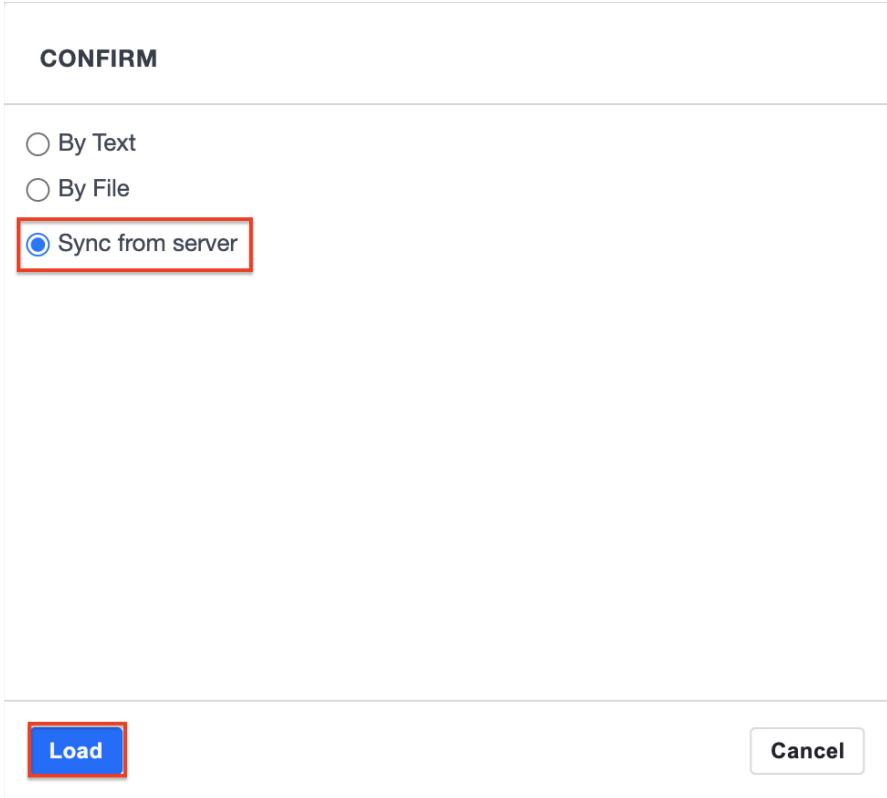
The screenshot shows the Boozang application interface. On the left is a vertical sidebar with icons for Home, Settings, Environment, Content Policy, Element Definition, Integration (which is selected), Test case in Alias, Notifications, Monitor Cucumber Report Content, Feature file server, Upload Report Data to specific Server, and Help. The main content area displays the 'Integration' configuration page. It includes sections for 'Test case in Alias' and 'Notifications' with '+' buttons to add items. Under 'Monitor Cucumber Report Content', there is a 'Filter' button and an 'Email content:' field with an 'Attach screenshot' checkbox. The 'Feature file server' section shows the URL <https://xray.cloud.xpand-it.com/api/v2/export/cucumber?filter=10005&z=true>. The 'Upload Report Data to specific Server' section shows 'Console output for Cucumber Report' and an 'Accept to be monitored' checkbox.

## 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.



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 Boozang application interface. At the top, there's a navigation bar with links like 'Edit', 'Q Comment', 'Assign', 'More', 'To Do', 'In Progress', 'Done', and 'Admin'. A red box highlights the 'Valid Login' link in the top left. On the right, there are buttons for '1 of 1', 'Return to search', 'Doc. Generator', and 'Export'. Below the navigation, there's a section titled 'Test Details' with fields for 'Type: Cucumber', 'Scenario Type: Scenario', and a 'Scenario' block containing a Cucumber test step: 'Given browser is opened to login page When user "demo" logs in with password "mode" Then welcome page should be open'. To the right of this, there are sections for 'Assignee: Unassigned', 'Reporter: Xpand IT Admin', 'Votes: 0', 'Watchers: 1 Stop watching this issue', and 'Created: 16/Jun/21 2:17 PM', 'Updated: 17/Jun/21 10:00 AM'. There are also 'Edit Steps' and 'Agile' buttons. On the left side, there are expandable sections for 'Pre-Conditions', 'Test Sets', 'Test Plans', 'Test Runs', 'Attachments', and 'Issue Links'. Under 'Issue Links', a red box highlights a link to 'XT-45 As a user, I can login the application'. At the bottom, there's a 'TO DO' button. A sidebar on the left contains icons for Home, Data, Console, Project (\$proj), and other tools.

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

This screenshot shows a modal dialog box titled 'PREVIEW' over a larger application window. The dialog lists requirements: 'All' and 'As a user, I can login the application'. It has 'Start' and 'Cancel' buttons. The background application window shows 'ADVANCED OPERATIONS' and 'UPDATING ITEMS' sections, with a message 'Merg affected Loading ...'. The sidebar on the left includes icons for Home, Data, Console, Project (\$proj), and other tools.

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.

The screenshot shows the Xray application interface. On the left is a sidebar with various icons. The main area has a header bar with tabs for 'Data' and 'Console'. Below the header is a 'Project (\$projec)' section. A 'New Feature' card is displayed, containing a user story: 'As a user, I can login the application'. This card has a red box around its 'Features(1)' button.

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.

This screenshot shows two side-by-side views of the Xray interface. The left view shows a 'Module info' card for a feature named 'As a user, I can login the application'. The right view shows a detailed 'Test setting' screen for a specific test case. The 'Main' tab is selected in the right panel. A red box highlights the 'Main' tab and the first step of the test scenario, which is a 'Given' step: 'User: (( ... )) to insert data/script'.

# 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.

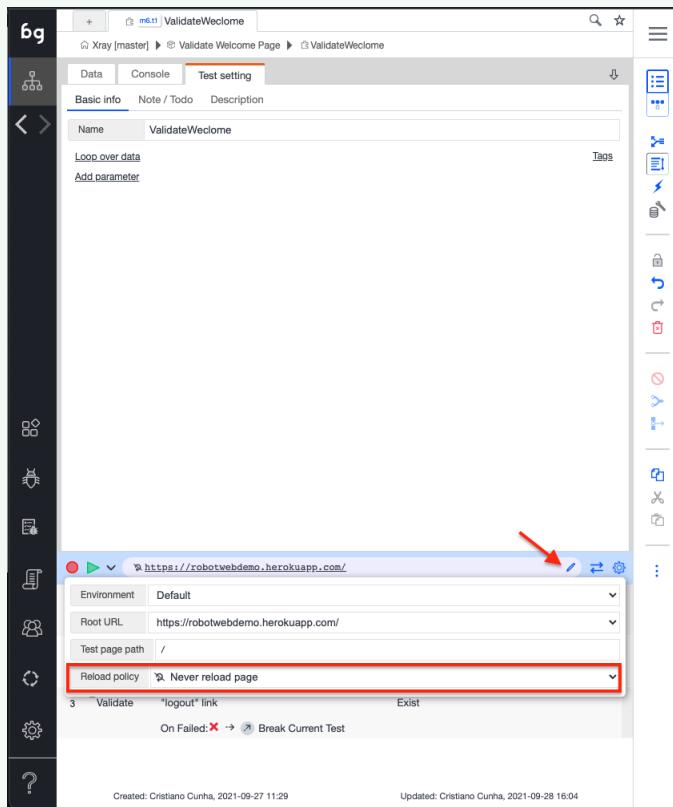
For this example, what matters is to have the steps automated so that they can be executed and produce a report of the execution which will be imported into Xray.

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 screenshot shows the Xray interface for a test case titled "Valid Login". The "Action setting" tab is selected. In the "Basic Info" section, the "Goto Test" field contains "[m3.t1] Insert UserName and Password". The "Description" field includes steps like "Open Robot Main Page", "LoadMainPage", "Validate Welcome Page", and "Validate error page". The main test steps are:

- Given
- Call NOT IMPLEMENTED! - browser is opened to login page
- When
- Call NOT IMPLEMENTED! - user "demo" logs in with password "mode"
- Then
- Call NOT IMPLEMENTED! - welcome page should be open

Created: Cristiano Cunha, 2021-09-20 16:36 Updated: Cristiano Cunha, 2021-09-29 14:37

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

The screenshot shows the Xray interface for the same test case. The "Basic Info" section now has "Goto Test" set to "[m4.t1/] LoadMainPage" and a "Description" field stating "browser is opened to login page". The "Key-Value Editor" tab is selected. The main test steps are:

- Given
- Call browser is opened to login page
- When
- Call user "demo" logs in with password "mode"
- Then
- Call welcome page should be open

Created: Cristiano Cunha, 2021-09-20 16:36 Updated: Cristiano Cunha, 2021-09-29 14:44

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 application interface, which includes a left sidebar with various icons for navigation and management, and two main panes for test configuration and execution results.

**Top Left Pane (Test Configuration):**

- Title:** m3.t1 | Valid Login
- Path:** Xray [master] > As a user, I can login the application > Valid Login
- Tab:** Test setting (selected)
- Content:** Shows a JSON editor with the following code:

```
1 | {  
2 |   "User_Name": "demo",  
3 |   "Password": "adasd",  
4 |   "Password1": "mode"  
5 | }
```

**Bottom Left Pane (Test Results):**

- Title:** eu.boozang.com/extension?id=613880c43fb72b77e2052fb0#613880c43fb72b77e2052fb0/master/m3/t1/2
- Path:** Xray [master] > As a user; I can login the application > Valid Login
- Tab:** Action setting (selected)
- Content:** Basic info tab shows Name: Valid Login and Tags: TEST\_XT-7.
- Test Runner:** Shows a green play button with the text "Use: {{ ... }} to insert data/script".
- Test Steps:** A tree view of test steps:
  - Given: browser is opened to login page
  - When: user "demo" logs in with password "mode"
  - Then: welcome page should be open
- Log:** Shows the log message "Login succeeded. Now you can logout."

**Bottom Right Pane (Test Results):**

- Title:** eu.boozang.com/extension?id=613880c43fb72b77e2052fb0#613880c43fb72b77e2052fb0/master/m3/t1/2
- Path:** Xray [master] > As a user; I can login the application > Valid Login
- Tab:** Current Test (selected)
- Content:** Shows the same test steps as the bottom left pane.
- Log:** Shows the log message "Login succeeded. Now you can logout."

## 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.

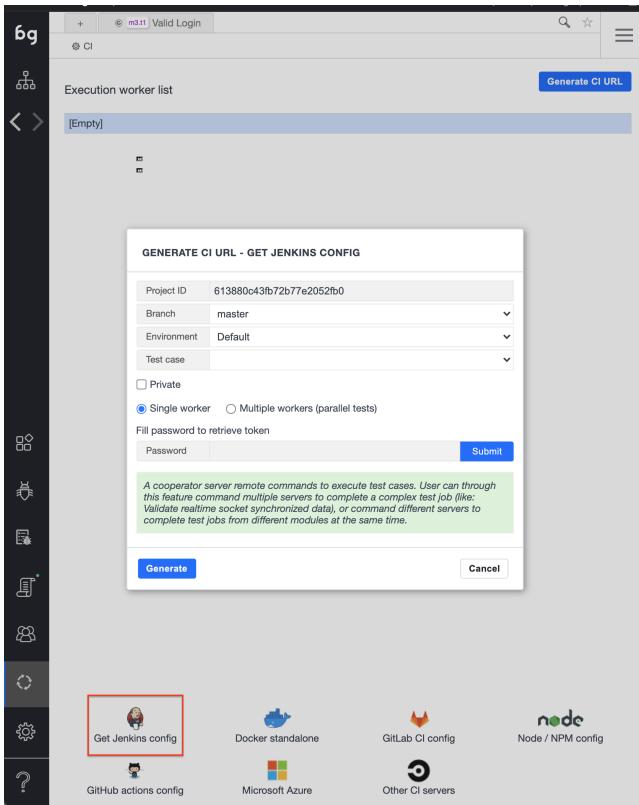
The screenshot shows the Boozang application interface. On the left is a dark sidebar with various icons: a magnifying glass, a triangle, a square, a lightbulb, a document, a person, a circular arrow (highlighted with a red box), a gear, and a question mark. The main area has a header with a search bar and a 'Generate CI URL' button. Below the header is a section titled 'Execution worker list' with a message '[Empty]'. At the bottom, there are several configuration options:

- Get Jenkins config (Jenkins icon)
- Docker standalone (Docker icon)
- GitLab CI config (GitLab icon)
- Node / NPM config (Node.js icon)
- GitHub actions config (GitHub icon)
- Microsoft Azure (Microsoft Azure icon)
- Other CI servers (generic icon)

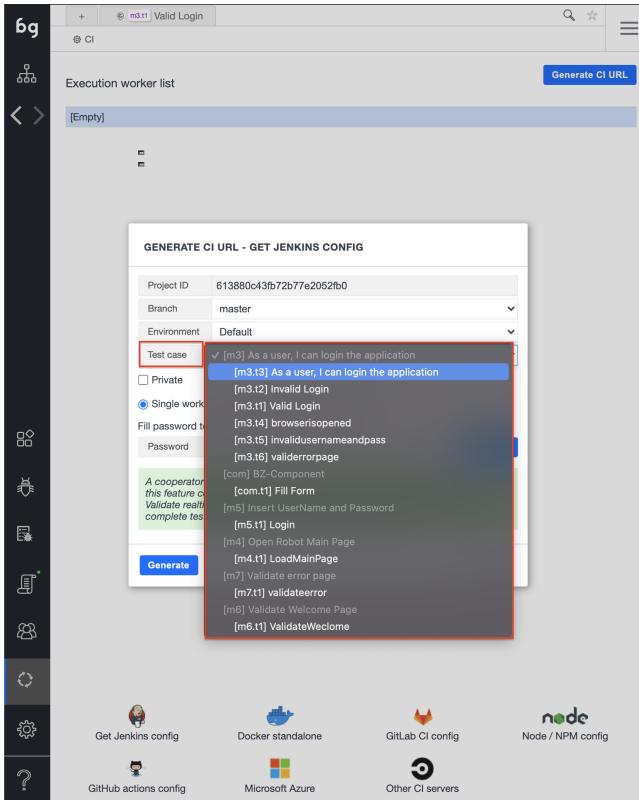
## Jenkins

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.

The Boozang runner for Jenkins requires Docker and Cucumber Reports.

1. Click "New Item" in the Jenkins main view
2. Choose "Freestyle project"
3. Add build step -> Execute Shell
4. Copy the below code into the shell
5. Add a Post-build action -> Add Cucumber reports
6. The job is ready to run!

```

BASE=https://eu.boozang.com
TOKEN=dbcc8241lca226cc9b393fc748a50e8e
ENV=0
PROJECT=613880c43fb72b77e2052fb0
BRANCH=master
SELF=0
TEST=m3/t3
GROUP=
SCOPE=
PARAMETER=
WORKERS=1

echo Running $workers processes for test: $test
echo Setting up slaves
counter=1
while [ $counter -lt ${WORKERS} ]
do
((counter++))
    WORKER_URL="${BASE}/extension?parameter=${PARAMETER}&token=${TOKEN}&group=${GROUP}&scope=${SCOPE}&PROJECT=${PROJECT}"
    nohup docker run --rm -v "${pwd}:/var/boozang/" --name=bzworker${counter} boozang-runner "${WORKERS}" >> /dev/null &
done
echo All slaves done. Starting master job.

```

[Copy code](#) [Close](#)

Get Jenkins config Docker standalone GitLab CI config Node / NPM config

GitHub actions config Microsoft Azure Other CI servers

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.

Dashboard > Boozang2 >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

With Ant [?](#)

**Build**

**Execute shell**

Command

```

BASE=https://eu.boozang.com
TOKEN=dbcc8241lca226cc9b393fc748a50e8e
ENV=0
PROJECT=613880c43fb72b77e2052fb0
BRANCH=master
SELF=0
TEST=m3/t3
GROUP=
SCOPE=
PARAMETER=
WORKERS=1

echo Running $workers processes for test: $test
echo Setting up slaves
counter=1
while [ $counter -lt ${WORKERS} ]
do
((counter++))
    WORKER_URL="${BASE}/extension?parameter=${PARAMETER}&token=${TOKEN}&group=${GROUP}&scope=${SCOPE}&PROJECT=${PROJECT}"
    nohup docker run --rm -v "${pwd}:/var/boozang/" --name=bzworker${counter} boozang-runner "${WORKERS}" >> /dev/null &
done
echo All slaves done. Starting master job.
MASTER_URL="${BASE}/extension?parameter=${PARAMETER}&token=${TOKEN}&group=${GROUP}&scope=${SCOPE}&PROJECT=${PROJECT}"
docker run --rm -v "${pwd}:/var/boozang/" --name=bzworker1 boozang-runner "${MASTER_URL}"

```

[X](#) [?](#)

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

### 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 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

The screenshot shows the configuration for the 'Xray: Results Import Task'. It includes fields for 'Jira Instance' (set to 'https://xray-demo3.xpand-it.com/'), 'Format' (set to 'Cucumber JSON'), and 'Parameters' (with 'Execution Report File' set to '/var/jenkins\_home/workspace/Boozang2/report\_cucumber-m3-t1.json' and 'Import in parallel' checked). A link 'Click here for more details' is also visible at the bottom.

**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:

```

274: go to close, status: running
275: BZ-LOG: report all tasks status,
276: post data from 552,1444
277: method: updateServerStatus, current tasks: 0
278: +--- /api/coop/ +--- without response
279: call all worker to close
280: post data from 1448
281: method: taskDone, current tasks: 0
282: +--- /api/coop/ +--- 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 33.33333 failed steps, while expected not more than 0.000000 percent
[CucumberReport] Build status is left unchanged
Starting XRAY: Results Import Task...
#####
##### Xray is importing the execution results #####
#####
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

```

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

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

No changes.

Started by user [cristiano cunha](#)

[add description](#) Started 20 hr ago  
Took 38 sec

### Cucumber Report

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

### Features Statistics

The following graphs show passing and failing statistics for features

#### Scenarios

Feature	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]

**Overall Execution Status**

1 FAIL

Total Tests: 1

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

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

Test Execution: XT-315 / Test: XT-45

**Valid Login**

**Test Issue Links**

tests

XT-45 As a user, I can login the application

**Results**

Context	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 [ "BZ.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 shows the Xray interface with the following details:

- Project:** Xray Tutorials
- Test Execution:** XT-315 / Test: XT-47
- Test Name:** Valid Login
- Test Issue Links:** XT-45 As a user, I can login the application
- Test Type:** Cucumber
- Scenario Type:** Scenario
- Scenario:** Given browser is opened to login page  
When user "demo" logs in with password "mode"  
Then welcome page should be open
- Results:** A table showing the execution of steps:

Context	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 [ "BZ.TW.document",		

## Learn more

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

