

# Integration with Boozang

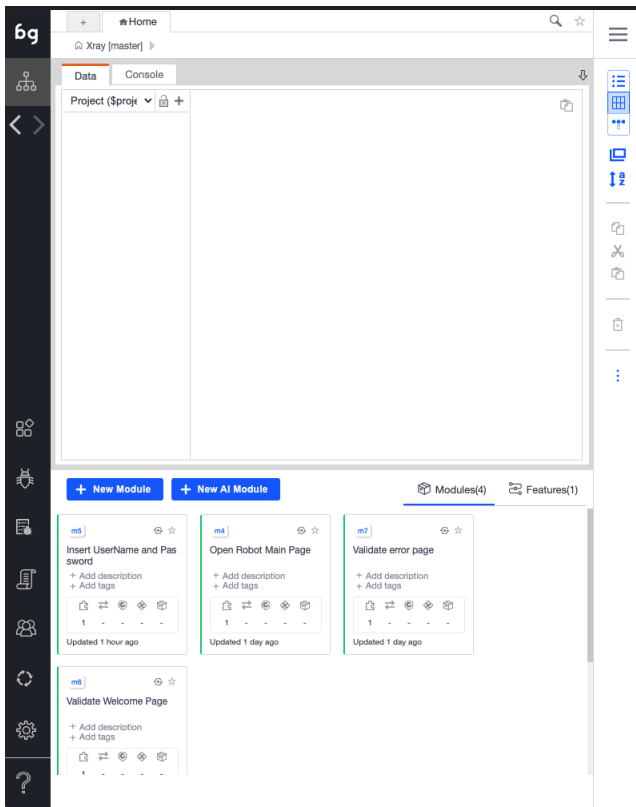
- [Overview](#)
- [Main features](#)
  - [Mapping of concepts](#)
- [Flow](#)
- [Setup](#)
- [Importing Xray Cucumber Tests into Boozang](#)
- [Automating Cucumber Tests in Boozang](#)
- [Import results to Xray](#)
  - [Jenkins](#)
  - [Command line](#)
    - [Authenticate](#)
    - [Send results to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



Home

Account ▾

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 and Server/DC
- Ability to export Cucumber tests from Xray to Boozang
- Automate tests in Boozang
- Import the automation results back to Xray

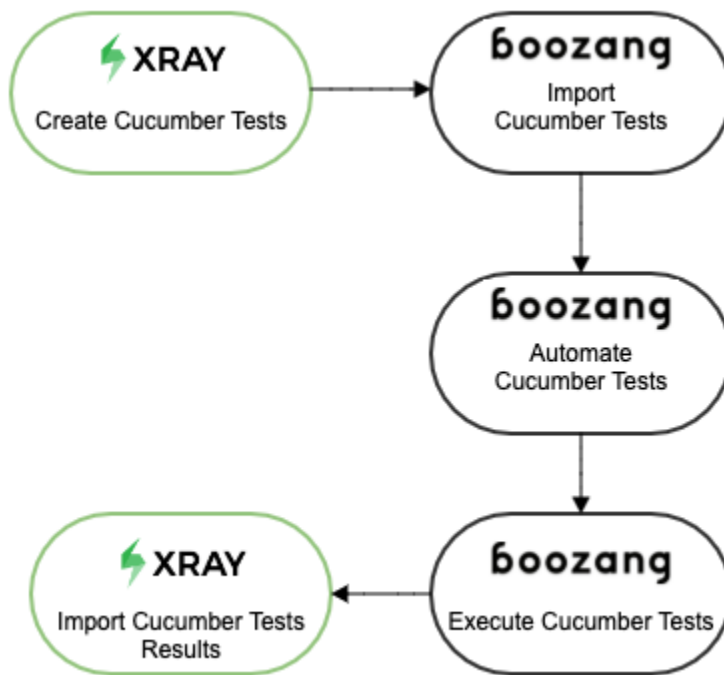
## Mapping of concepts

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; 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 the tests to be imported into Boozang - the integration is based on this filter. To do so select "Advanced issue search" from the "Filters" entry in the top menu.

Filters

- Search issues
- STARRED
  - Boozang
- OTHER
  - My open issues
  - Reported by me

STARRED

RECENT

- Filter for BTW board

View all filters

Advanced issue search

Assignee Reporter P Status Resolution Created Updated

XT-235	Execution results [1632825235092]	Cristiano Cunha	Cristiano Cunha	TO DO	Unresolved	28/Sep/21	28/Sep/21
--------	-----------------------------------	-----------------	-----------------	-------	------------	-----------	-----------

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:

Filters

- Search issues
- STARRED
- Boozang
- OTHER
  - My open issues
  - Reported by me
  - All issues
  - Open issues
  - Done issues
  - Viewed recently

Boozang

Save as Details

project = XT AND testType = Cucumber AND Key in (XT-8, XT-7) ORDER BY created DESC

Search Switch to basic

1-2 of 2

T Key Summary Assignee Reporter P Status Resolution Created Updated

XT-8	Invalid Login	Unassigned	Sérgio Freire	TO DO	Unresolved	17/Jun/21	17/Jun/21
XT-7	Valid Login	Unassigned	Sérgio Freire	TO DO	Unresolved	17/Jun/21	17/Jun/21

1-2 of 2

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

Your work

Projects

Filters

Dashboards

People

Apps

+

Q Search

Filters

Search issues

STARRED

Boozang

OTHER

My open issues

Reported by me

All issues

Open issues

Done issues

Boozang

Save as

Details

Share

Export

project = XT AND testType = Cucumber AND Key in (XT-8, XT-7) ORDER BY created DESC

Search

Switch to basic

1-2 of 2

Columns

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated
	XT-8	Invalid Login	Unassigned	Sérgio Freire		TO DO	Unresolved	17/Jun/21	17/Jun/21
	XT-7	Valid Login	Unassigned	Sérgio Freire		TO DO	Unresolved	17/Jun/21	17/Jun/21

1-2 of 2

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

# Save Filter

Filter Name\*

Boozang

Enter a name for this Filter

Submit

Cancel

The screenshot shows the Atlassian Xray web interface. The browser address bar displays the URL `xraytutorials.atlassian.net/issues?filter=10005`. The top navigation bar includes links for 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', and 'Apps'. A search bar is also present. On the left sidebar, under the 'Filters' section, the 'Boozang' filter is selected. The main content area shows the filter details: 'project = XT AND testType = Cucumber AND Key in (XT-8, XT-7) ORDER BY created DESC'. Below this, a table lists the filtered issues:

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated
	XT-8	Invalid Login	Unassigned	Sérgio Freire	==	TO DO	Unresolved	17/Jun/21	17/Jun/21
	XT-7	Valid Login	Unassigned	Sérgio Freire	==	TO DO	Unresolved	17/Jun/21	17/Jun/21

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:

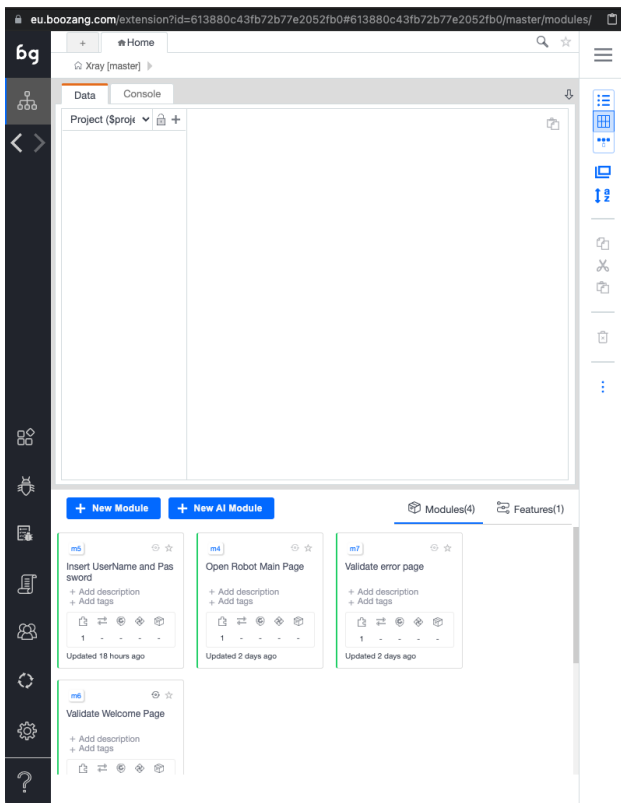
## Welcome Cristiano Cunha

[Create new Project](#)

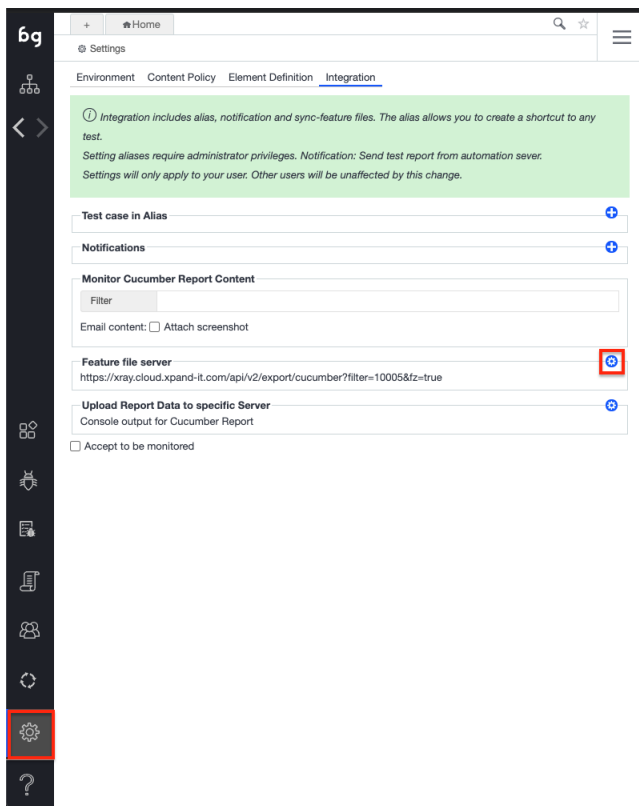
This is **Region Europe**. Select a project below to view installation instructions, manage your project team, view bugs or add project versions.

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

This will open the tool unless it's the first time you utilize 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.cloud.xpand-it.com/api/v2/export/cucumber?filter=10005&fz=true
Token	Authorization: Bearer
Client ID	BC00BF01FC514BFA90CF436
Client Secret	.....
Match File	*.feature
<input checked="" type="checkbox"/> In Zip	

Done

Check

Cancel

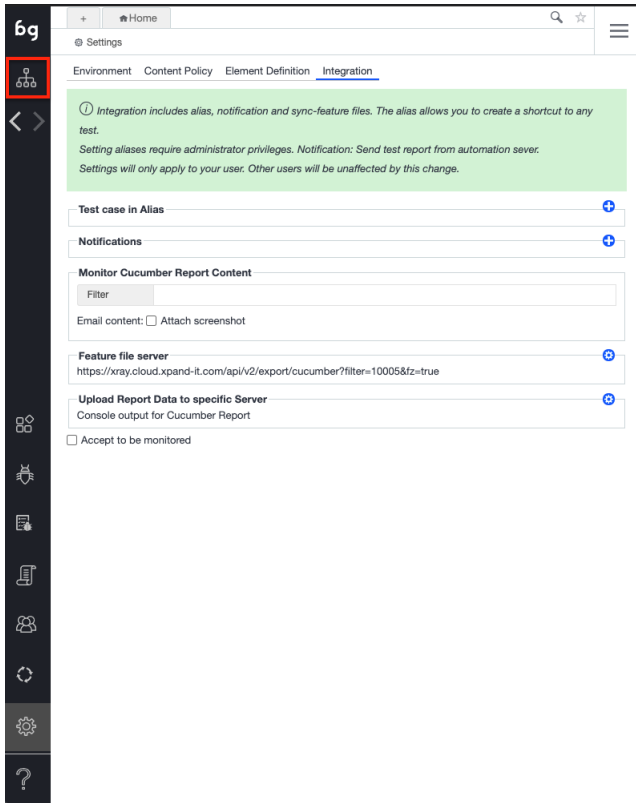
In more detail:

- Type: Choose Jira/Xray option
- File List Url: insert the JQL filter that we have saved in Jira/Xray
- Token: Leave it blank (it will be filled automatically)
- Client ID: Client ID from Xray
- Client Secret: Client Secret from Xray
- Match File: Leave the "\*.feature"

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

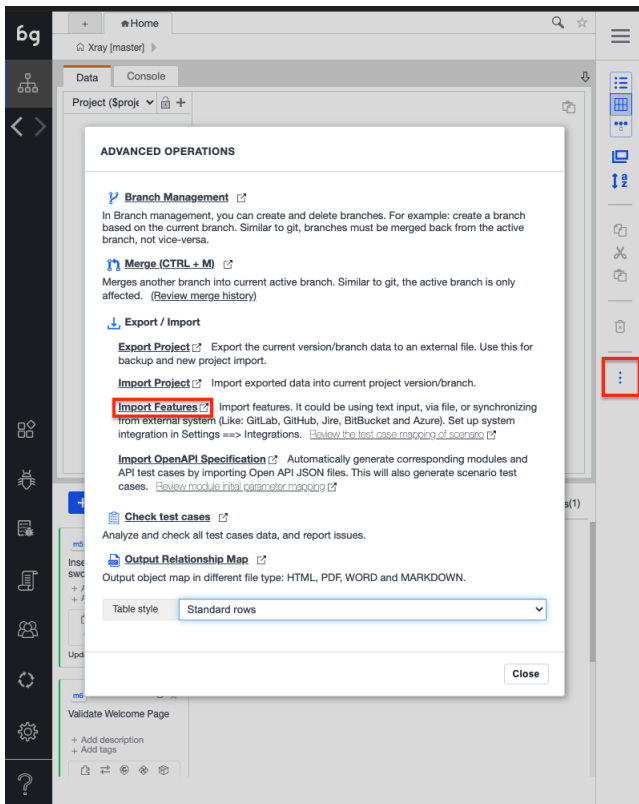


Now head back to the main page of the Boozang application by selecting the first option in the left-hand side menu.



## Importing 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 the "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-5 User Story.

Projects / Xray Tutorials / XT-5

As a user, I can login the application

Description

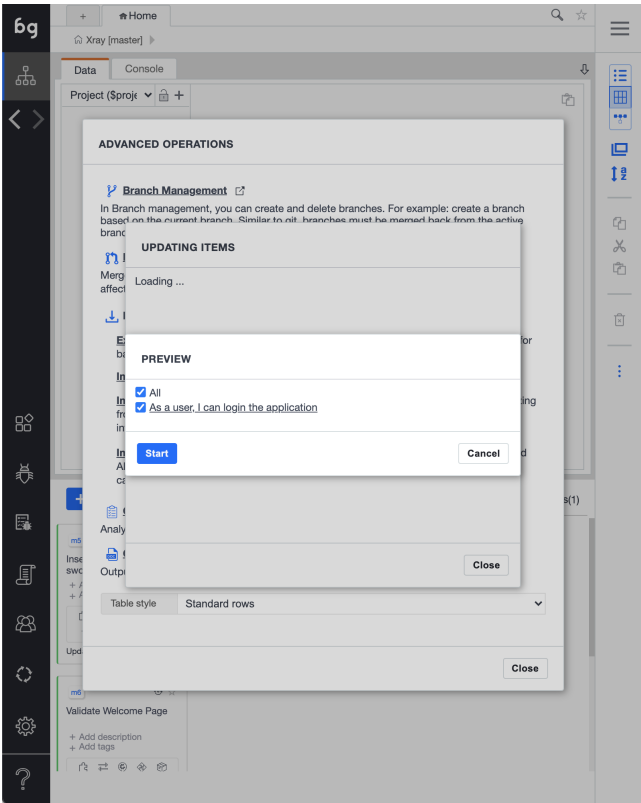
Add a description...

Linked issues +

is tested by

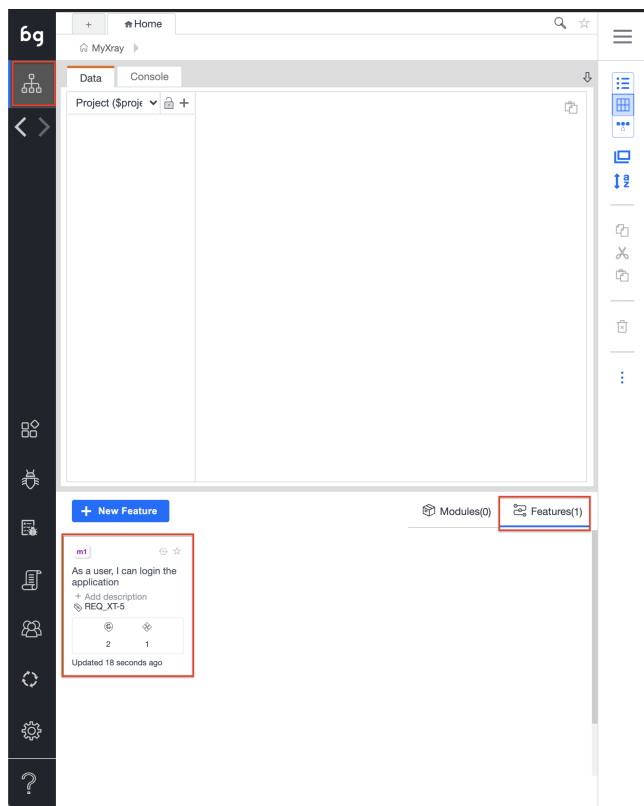
XT-7 Valid Login	=		TO DO ▾
XT-8 Invalid Login	=		TO DO ▾

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

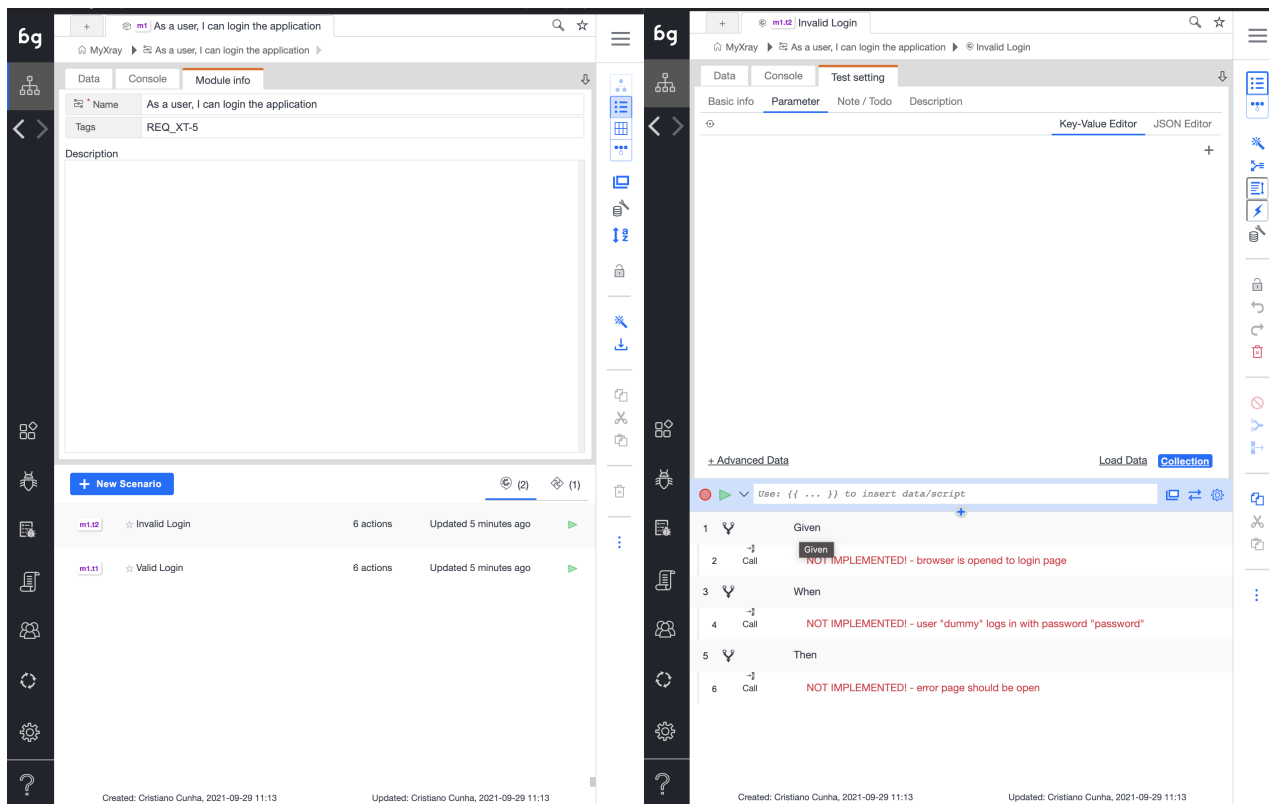


When selecting "Start," Boozang will show you the items it will create and proceed with the creation of the features.

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.

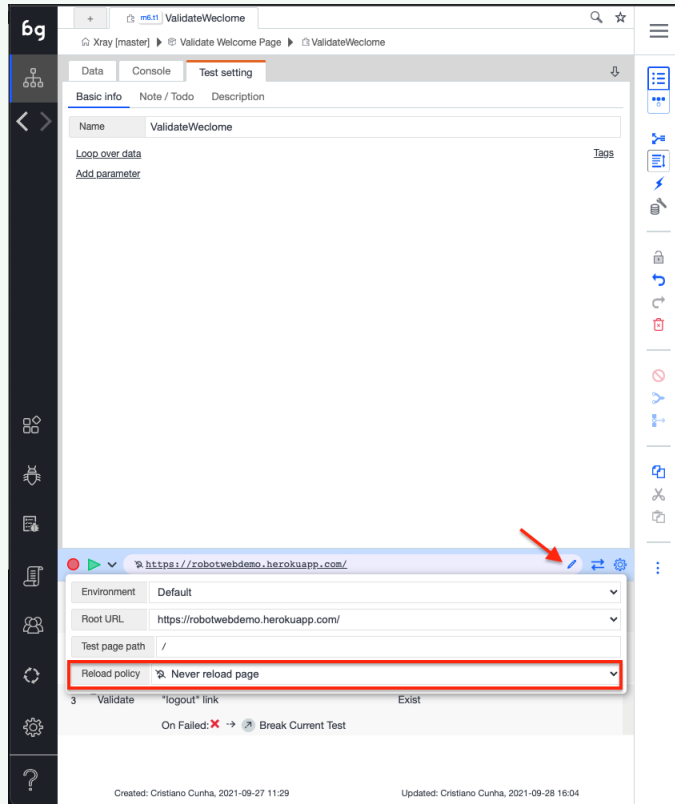
For this example, what matters is to have the steps automated so that they can be executed and produce a report of the execution which will be imported into Xray.

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.

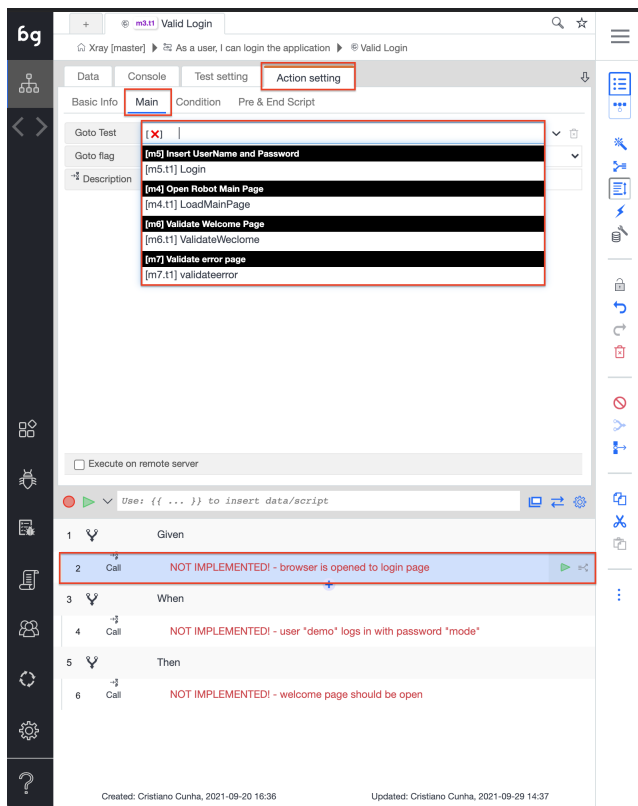
## Tip

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 has ended. To do that, set the Parameter inside the "*Reload Policy*" of the detail of the Test to "*Never*."

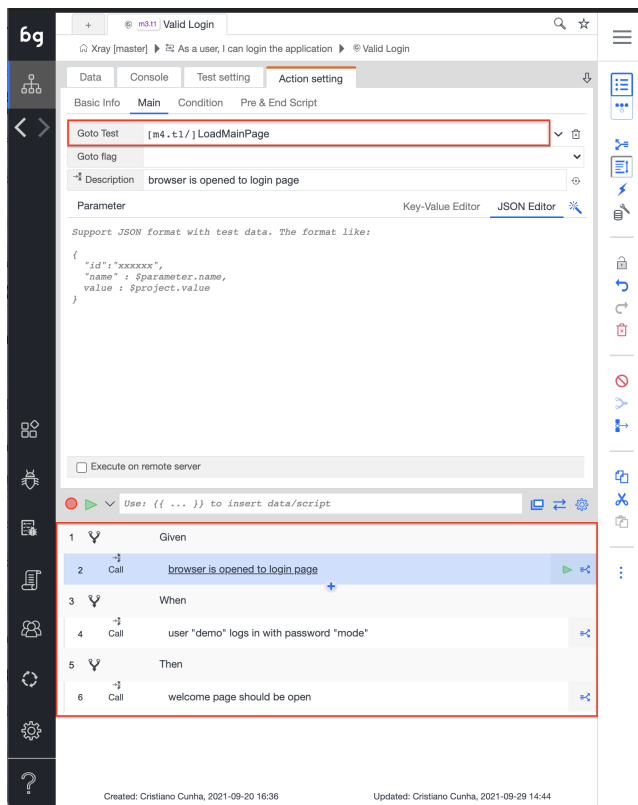


Once the Modules and Tests are ready, they'll need to be linked to each Step in the Cucumber Test. In order to achieve this, access the details of each Cucumber Test and click on "*NOT IMPLEMENTED! -...*" which will load in the upper part of the application, under the *Main* submenu of the "*Action Setting*"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

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



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



Once you have associated all the 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 shows the Boozang test editor interface. The top bar displays the test name 'Valid Login'. Below it, there are tabs for 'Data', 'Console', and 'Test setting'. The 'Test setting' tab is active, showing a 'Parameter' section with a JSON configuration:

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

Below the JSON editor, there is a 'Current Test' section with a 'Result list' and 'Failed Actions (0)'. The test steps are listed as follows:

- 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 bottom of the interface shows the creation and update timestamps: 'Created: Cristiano Cunha, 2021-09-20 16:36' and 'Updated: Cristiano Cunha, 2021-09-29 14:44'.

The screenshot shows the Boozang test execution interface. The top bar displays the test name 'Valid Login'. Below it, there are tabs for 'Data', 'Console', 'Test setting', and 'Action setting'. The 'Test setting' tab is active, showing a 'Basic info' section with the following details:

- Name: Valid Login
- Tags: TEST\_XT-7

Below the 'Basic info' section, there is a 'Current Test' section with a 'Result list' and 'Failed Actions (0)'. The test steps are listed as follows:

- 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 bottom of the interface shows the creation and update timestamps: 'Created: Cristiano Cunha, 2021-09-20 16:36' and 'Updated: Cristiano Cunha, 2021-09-29 14:44'.

On the left side of the screenshot, there is a browser window showing the 'Welcome Page' of the 'robotwebdemo.herokuapp.com' application. The page content is:

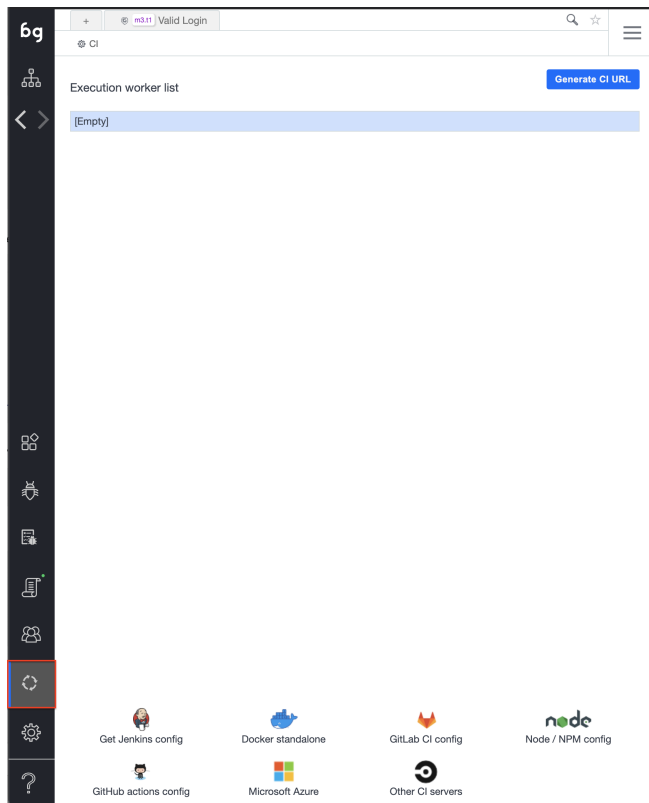
**Welcome Page**  
Login succeeded. Now you can [logout](#).



# 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.

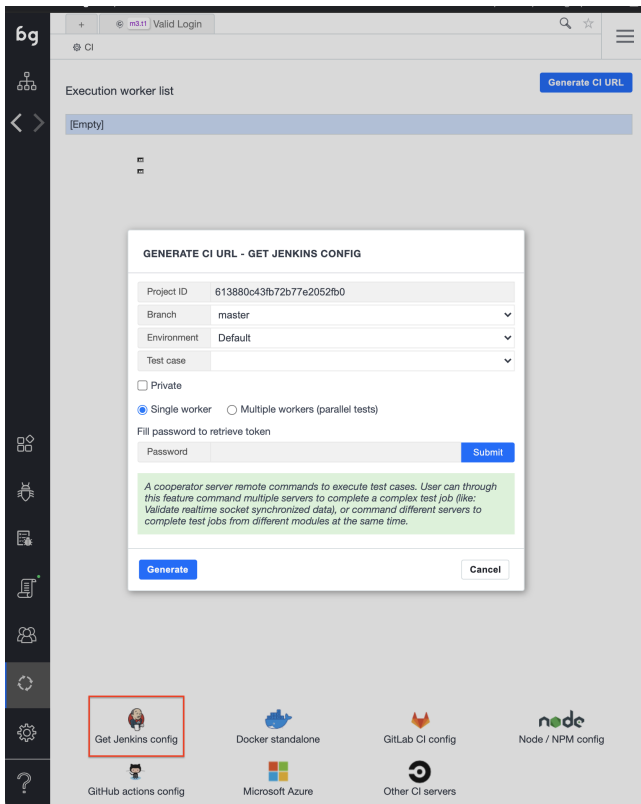
In Boozang, access the CI entry present at the bottom of the left menu.



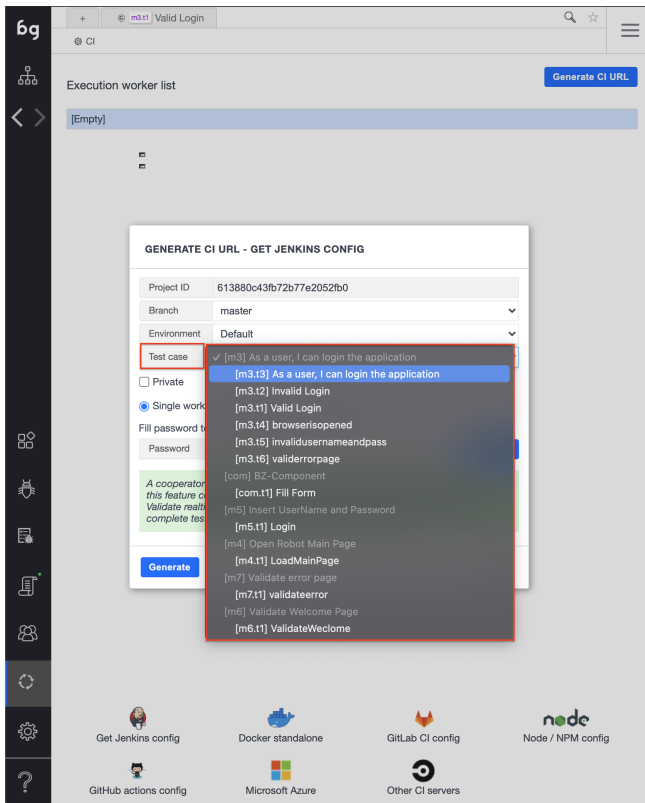
## Jenkins

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*".

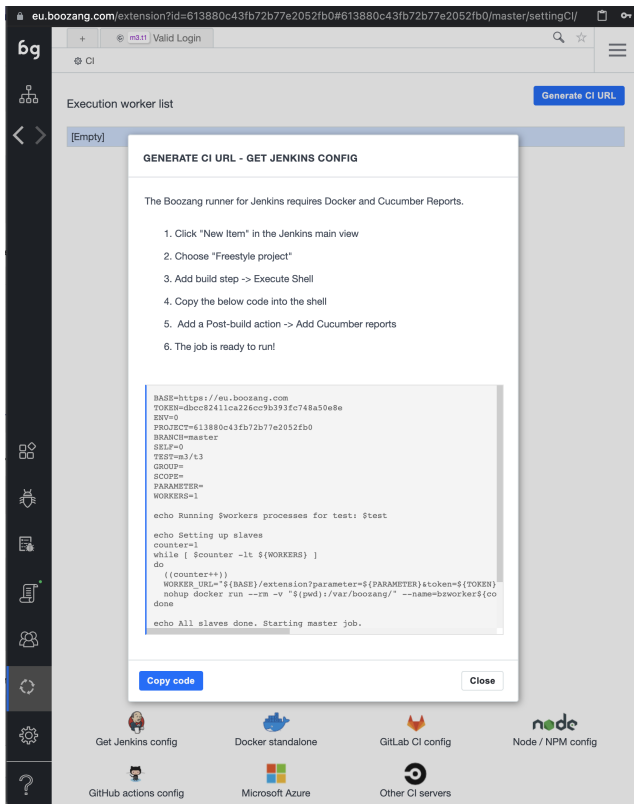
A new pop-up will appear requiring details to generate the script.



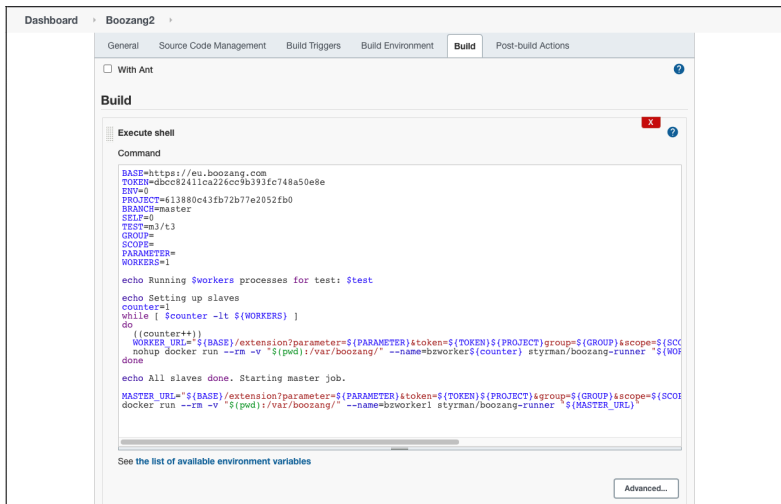
The majority of the options will be default ones, let's choose "Test case" to be the feature we have automated (from the drop-down list).



Fill in the password, click "Submit," and click on "Generate." This will produce the necessary script that 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 an "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 use this 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 into Xray. First, make sure you have previously installed the [plugin](#) and in your pipeline, just add 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 this case, Cucumber JSON
- The path and name of the file to be uploaded

Xray: Results Import Task

Jira Instance

https://xraytutorials.atlassian.net/

Format

Cucumber JSON

Parameters

Execution Report File (file path with file name)

./merged-test-results.json

☒ Import in parallel

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

[Click here for more details](#)

Add post-build action

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 > #30

```
464: BE-LOG: report all tasks status,
465: post data from 552,1444
466: method: updateServerStatus, current tasks: 0
467: +++> /api/coop/ +++> without response
468: call all worker to close
469: post data from 1448
470: method: taskDone, current tasks: 0
471: +++> /api/coop/ +++> without response
472: task-done
473: 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/30/cucumber-html-reports/.cache"
[CucumberReport] Copied 2 files from workspace "/var/jenkins_home/workspace/Boozang2" to reports directory
"/var/jenkins_home/jobs/Boozang2/builds/30/cucumber-html-reports/.cache"
[CucumberReport] Processing 2 json files:
[CucumberReport] /var/jenkins_home/jobs/Boozang2/builds/30/cucumber-html-reports/.cache/report_cucumber-m3-
t1.json
[CucumberReport] /var/jenkins_home/jobs/Boozang2/builds/30/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...
#####
###      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) {"id":"10269","key":"XT-
238","self":"https://xraytutorials.atlassian.net/rest/api/2/issue/10269"}
Successfully imported Cucumber JSON results from report_cucumber-m3-t1.json
XRAY_IS_REQUEST_SUCCESSFUL: true
XRAY_RAW_RESPONSE: {"id":"10269","key":"XT-
238","self":"https://xraytutorials.atlassian.net/rest/api/2/issue/10269"}
XRAY_TESTS:
XRAY_ISSUES_MODIFIED: XT-238
XRAY_TEST_EXECS: XT-238
Finished: FAILURE
```

REST APIJenkins 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 view 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**

search

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					Total	Scenarios			Features	
	Passed	Failed	Skipped	Pending	Undefined		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 there were several Tests it will be the overall result of all the Tests.

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

- The requirement that is covered by the Test, in this case, XT-5
- The scenario definition
- The details of the execution of each step

## 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

You can start the authentication process by accessing your ClientID and Client Secret from your Jira/Xray cloud (more information [here](#)) and then use the following command to obtain the token:

```
curl -H "Content-Type: application/json" -X POST --data '{ "client_id": "BC00BF01FC514BFA90CF43067B0035C86", "client_secret": "4aed9234c12b166e5c7e7b45a5535845df37f5a750fa170e3ee9813f9a6a249bb" }' https://xray.cloud.getxray.app/api/v2/authenticate
```

This will return a token that you must save to use in the following requests.

## Send results to Xray

Finally, you must enable the above token in the following request in order to upload the results back into Xray:

```
curl -H "Content-Type: application/json" -X POST -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzJ9...EmSHPTxO2NXBDG1Sn5qfoydMKW7oncRvrzbV6e26fU0" --data @ 'xrayResults.json' 'https://xray.cloud.getxray.app/api/v2/import/execution'
```

When successful, the answer will have information of the Test Execution created, like the following one that you can use to check for in Xray:

```
{"id": "10268", "key": "XT-237", "self": "https://xraytutorials.atlassian.net/rest/api/2/issue/10268"}
```

We can see the detailed results in Xray:

The screenshot displays the 'Test details' page in Xray, specifically for a CUCUMBER scenario. The page is divided into several sections: 'Test Issue Links', 'Scenario', and 'Results'.

- Test Issue Links:** Shows a link to 'XT-5 As a user, I can login the application' with a 'To Do' status.
- Scenario:** Lists three steps:
  - 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 results of the scenario. The overall status is 'FAILED'.

Context	Duration	Status
Steps	9s 193ms	FAILED
Given Browser is opened to login page	2 secs	PASSED
When User "demo" logs in with password "mode"	2 secs	PASSED
Then Welcome page should be open	4 secs	FAILED

The 'Then Welcome page should be open' step is highlighted with a red box, indicating the failure. The error message states: 'Element is not found or syntax error exists in the code'. The error details include a failed action 'm7.t1.1, Validate [exist]: "Error Page" main heading' and an error hash 'A11BE61D39D0929293CC1826C756C691'.

## Learn more

- [Boozang home page](#)



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