

Setup Jenkins webhook triggers for a project (WIP).

- Setup Webhooks in BitBucket.
 - See the events requests.
- Jenkins Setup
 - Generate user token
 - Configure plugin for each of the jobs.
- Annex: Installing generic-webhook-trigger plugin

First we will be needing a user that has permissions to launch the jenkins jobs. The user for continuous integration is SG_MAD_EJOURNEY_CI. It is already setup for our jenkins instance. If a project is hosted in a different jenkins instance, then we would be needing to talk with somebody from IT (probably Alberto Robles), and ask him to generate a token for such user, and give it to us.

Setup Webhooks in BitBucket.

For this you will be needing administration permissions in Bitbucket for the project you need to setup. Anyone with these permissions will be able to modify these webhooks.

First thing, go to the webhooks tab for your repository, inside the repository settings:

The screenshot shows the BitBucket repository settings page for a specific project. On the left, there's a sidebar with various configuration tabs like 'Repository details', 'Security', 'Branch permissions', etc. The 'Webhooks' tab is highlighted with a red box. The main content area is titled 'Webhooks' and contains a table listing existing webhook configurations. Each row includes columns for Name, URL, Events, Level, Last response, Active status, and Actions. A 'Create webhook' button is located at the top right of the table area.

Name	URL	Events	Level	Last response	Active	Actions
jenkins_taik_server	http://01567991130628837269b3908a15a03...	Pull request merged, Pull request opened...	Repository	200	ACTIVE	...
taik_jenkins_bms_client	http://01567991130628837269b3908a15a03...	Pull request merged, Pull request opened...	Repository	200	ACTIVE	...
taik_jenkins_platform_client	http://01567991130628837269b3908a15a03...	Pull request merged, Pull request opened...	Repository	200	ACTIVE	...
taik_jenkins_platform_server	http://01567991130628837269b3908a15a03...	Pull request merged, Pull request opened...	Repository	200	ACTIVE	...
taik_jenkins_webportal	http://01567991130628837269b3908a15a03...	Pull request merged, Pull request opened...	Repository	200	ACTIVE	...
taik_jenkins_client_vms	http://01567991130628837269b3908a15a03...	Pull request merged, Pull request opened...	Repository	200	ACTIVE	...

Then push the button `Create Webhook`:

Here you will fill in

- the name for your hook,
- the url that needs to be invoked,
- the events that will trigger the webhook.

Of these three, the most important, and the most sensitive one is the url.

First let's choose the events that will trigger the webhook (bottom of the page):



with the checks shown the following events will trigger a build:

1. When there is a new commit in a branch.
2. When a new pull request is open, in the "from" branch.
3. When a pull request is updated with new code (amend, new commit, etc).
4. When a pull request is merged.

The url that needs to be filled looks like this:

namibia_ebms_client ACTIVE
http://SG_MAD_EJOURNEY_CI:11623a82c6f36f4e6315c31a3f164affee@10.75.105.211:8080/generic-webhook-trigger/invoke?token=namibia_unique_ebms_client

http://SG_MAD_EJOURNEY_CI:11623a82c6f36f4e6315c31a3f164affee@10.75.105.211:8080/generic-webhook-trigger/invoke?token=namibia_unique_ebms_client

Url analysis:

SG_MAD_EJOURNEY_CI	This is the jenkins user with permissions to trigger the jobs. This one we used here is the designated one for continuous integration.
11623a82c6f36f4e6315c31a3f164affee	This is the jenkins token for such user. Must be generated in jenkins. More information in the jenkins configuration part. This must be done for each user that triggers the builds and for each server. It is not inherited.
10.75.105.211:8080	This is the jenkins instance where your jobs are located.
generic-webhook-trigger/invoke? token=	This is a fixed part of the url that the plugin we are using imposes. That plugin is generic-webhook-trigger : Link
namibia_unique_ebms_client	This is an unique identifier of the job that will trigger. More information in jenkins part.

For each job you need to trigger, you will need one webhook, as we have seen there is a unique identifier that only identifies one job. So, if you have several jobs that need to be triggered when updating one repository, as is our case, then you will be needing several webhooks in one repository.

See the events requests.

Once such events are fired, the last one of them will be recorded, so that you can review it:

Webhooks

[Create webhook](#)

Use webhooks to send requests to your server (or another external service) when certain events occur in Bitbucket. You can configure webhooks to update an issue tracker, trigger CI builds, or even deploy to your production server. [Learn more about webhooks](#)

Level

All

Name	URL	Events	Level	Last response	Active	Actions
namibia_ebms_client	http://SG_MAD_EJOURNEY_CI:11623a82c6f36f...	Pull request merged, Pull request opened....	Repository	200	ACTIVE	...
namibia_unique_ebms_platformclient	http://SG_MAD_EJOURNEY_CI:11623a82c6f36f...	Pull request merged, Pull request opened....	Repository	200	AC	View details
namibia_unique_ebms_server	http://SG_MAD_EJOURNEY_CI:11623a82c6f36f...	Pull request merged, Pull request opened....	Repository	200	AC	Edit
namibia_unique_ebms_platformserver	http://SG_MAD_EJOURNEY_CI:11623a82c6f36f...	Pull request merged, Pull request opened....	Repository	200	AC	Delete
						...

namibia_ebms_client ACTIVE

http://SG_MAD_EJOURNEY_CI:11623a82c6f36f4e6315c31a3f164affee@10.75.105.211:8080/generic-webhook-trigger/invoke?token=namibia_unique_ebms_client

Event log

The last success and failure for each event are recorded in the table below (up to the last 30 days).

Event type	Last success	Last failure	Successful calls
Pull request merged	04 Nov 2025	Never failed	1/1 (100%)
Pull request opened	04 Nov 2025	Never failed	1/1 (100%)
Pull request source branch updated	04 Nov 2025	Never failed	1/1 (100%)
Repository refs updated	04 Nov 2025	Never failed	4/4 (100%)

And there, you can review the request, and the response from jenkins. In a request that works, we would see something like the following images, but more of the capture later.

Webhook event details

Request

Request details

- Event type: prmerged
- URL endpoint: http://10.75.105.211:8080/generic-webhook-trigger/invoke?token=namibia_unique_ebms_client

Headers

```
X-Request-Id: F90c9215-0881-4a10-a54d-84809e1e2a10
Content-Type: application/json; charset=utf-8
X-Event-Key: pr:merged
```

Body

```
{
  "date": "2025-11-08T15:20:17+0000",
  "actor": {
    "emailAddress": "jose.sco@external.thalesgroup.com",
    "displayName": "Seba-EXTERNAL Jose",
    "name": "SEBA-SEJOSE",
    "active": true,
    "id": "test_ci_jenkins2",
    "last": "test_ci_jenkins2"
  },
  "id": 49359,
  "type": "NORMAL",
  "slug": "58858799"
},
"eventkey": "pr:merged",
"pullrequest": {
  "author": {
    "emailAddress": "jose.sco@external.thalesgroup.com",
    "displayName": "Seba-EXTERNAL Jose",
    "name": "SEBA-SEJOSE",
    "active": true,
    "id": "test_ci_jenkins2",
    "last": "test_ci_jenkins2"
  }
}
```

Webhook event details

Response

Response details

- HTTP status: 200
- Duration: 117ms

Headers

```
Date: Tue, 04 Nov 2025 14:19:58 GMT
Server: Jetty(9.4.45.v20210920)
X-Content-Type-Options: nowrapff
Content-Type: application/json; charset=UTF-8
Via: 1.1 localhost (Apache-HttpClient/4.5.14 (cache))
Content-Length: 235
```

Body

```
{
  "jobs": {"Namibia-eBMS-Client-Integration-Git": [
    {"regExpFilterExpression": "",
     "triggered": true,
     "resolvedVariables": {"BRANCH_NAME": "test_ci_jenkins2"},
     "regExpFilterText": "",
     "id": 8829,
     "url": "queue/item/8823/"}
  ]},
  "message": "Triggered jobs."
}
```

Jenkins Setup

Generate user token

Tokens are generated in your user configuration. So first you must login with the desired user, then push on such user name (up-right). The screen you will see will be like this:

The screenshot shows the Jenkins user configuration interface. At the top, there's a navigation bar with the Jenkins logo and the user name "Seco-EXTERNAL Jose". Below the navigation bar is a sidebar with several options: People, Status, Builds, **Configure**, My Views, and Credentials. The "Configure" option is highlighted with a red box. The main content area displays the user's information: "Seco-EXTERNAL Jose" and "Jenkins User ID: s0156799".

If you push "Configure", there will be a section for generating a token:

The screenshot shows the "API Token" section of the Jenkins configuration page. It includes fields for "Full Name" (set to "Seco-EXTERNAL Jose") and "Description". Below these, under "Current token(s)", there is a table showing one token named "token1". The table includes columns for the token name, creation date ("Created 15 day(s) ago"), usage statistics ("Used 22 time(s), last time was 13 day(s) ago"), and a delete button. A red box highlights this entire section.

token1	Created 15 day(s) ago	Used 22 time(s), last time was 13 day(s) ago	Delete
Add new Token			

Configure plugin for each of the jobs.

1

This needs to be done in each of the jobs that will be triggered.

The general way of working is that it will capture data from the body of the request that is sent from Bitbucket to Jenkins. The body of such request is in json format. With the data you capture, you can override the values of the variables of the job, or define new variables that you want to use for something.

In the configure tab of the job, you will get a new Build Trigger, that has been added by this plugin:

Build Triggers

Build after other projects are built ?
 Build periodically ?
 Build whenever a SNAPSHOT dependency is built ?
 Generic Webhook Trigger ?

Don't have any other trigger active. Specially don't have this one, that makes the whole thing not work:

Trigger builds remotely (e.g., from scripts) ?

That way, it will be setup to be triggered just manually, or with this webhook trigger.

When you push the "Generic Webhook Trigger" check, it will expand for the configuration of such webhook trigger.

First thing you will usually want to capture, is the branch, the variable name will change from repository to repository. As we have different events that we are capturing in Bitbucket, the requests will have different bodies, so we will need to be search for such branch in different places. As we don't know which request we are receiving, we need to define the same variable as many times as needed to match all of the requests.

First one would go to this web, to test their capture of the variable, using the request body copied from Bitbucket: [Link](#)

The screenshot shows a web-based JSON processing interface. On the left, there's a large text area for pasting JSON data. It contains a complex JSON object representing a pull request. Below this is a 'JSONPath Expression' input field containing 'pullRequest.fromRef.displayId'. To the right of the input field is a 'Process' button. Above the input field, there are dropdown menus for 'JSON Data/URL', 'JSON Template' (set to '3 Space Tab'), and 'Implementation' (set to 'JSONPath 0.5.0'). At the bottom of the interface, there are several navigation icons and a status bar indicating '#2 November 13th 2025, 11:32:44 am'. The results panel on the right shows the output of the JSONPath expression, which is a single value: 'test_ci_jenkins2'.

And then would capture that variable, testing with as many bodies of the requests, as evens are. In our case, we only need to capture 3 ways, because for the Pull Requests events, we can find the branch in the same place every time.

Post content parameters

Variable
BRANCH_NAME
Name of variable

Expression
\$changes[0].ref.displayId
 JSONPath
 XPath

Variable

BRANCH_NAME
Name of variable

Expression

\$pullRequest.fromRef.displayId

JSONPath

XPath

Additionally you can redefine other variables that you want, using an incorrect JSONPath, or define new variables that you want for some reason. This way we can make sure Fortify is always triggered, for example:

Variable

RUN_FORTIFY
Name of variable

Expression

JSONPath

XPath

Expression to evaluate in POST content. Use [JSONPath](#) for JSON or [XPath](#) for XML.

Value filter

Optional. Anything in the evaluated value, matching this [regular expression](#), will be removed. Having [^0-9] would only allow numbers. The regexp syntax is documented [here](#).

Default value

true
Optional. This value will be used if expression does not match anything.

Then we have the unique token that is used as unique identifier in the request:

Token

namibia_unique_ebms_client
Optional token. If it is specified then this job can only be triggered if that token is supplied when invoking http://JENKINS_URL/generic-webhook-trigger/invoke. It can be supplied as a:

And then, finally we have a check that allows us to debug the captured variables in the log, if enabled:

- Print contributed variables
- Print contributed variables in job log.

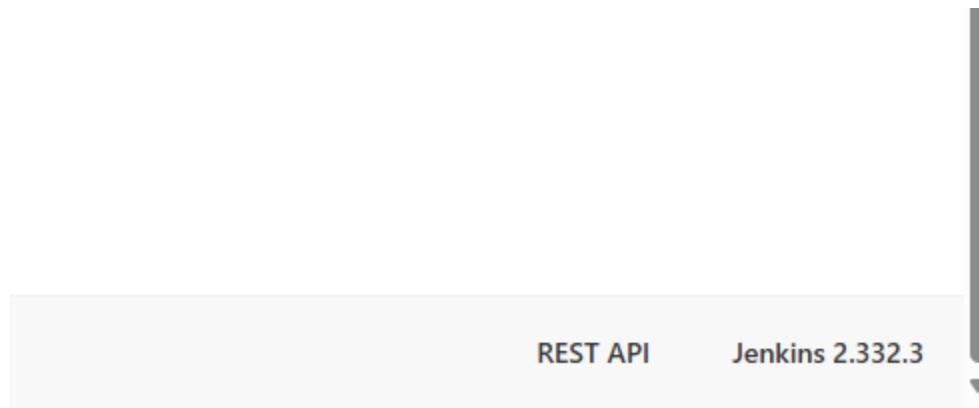
Contributing variables:

```
BRANCH_NAME = test_ci_jenkins2
RUN_FORTIFY = true
```

Annex: Installing generic-webhook-trigger plugin

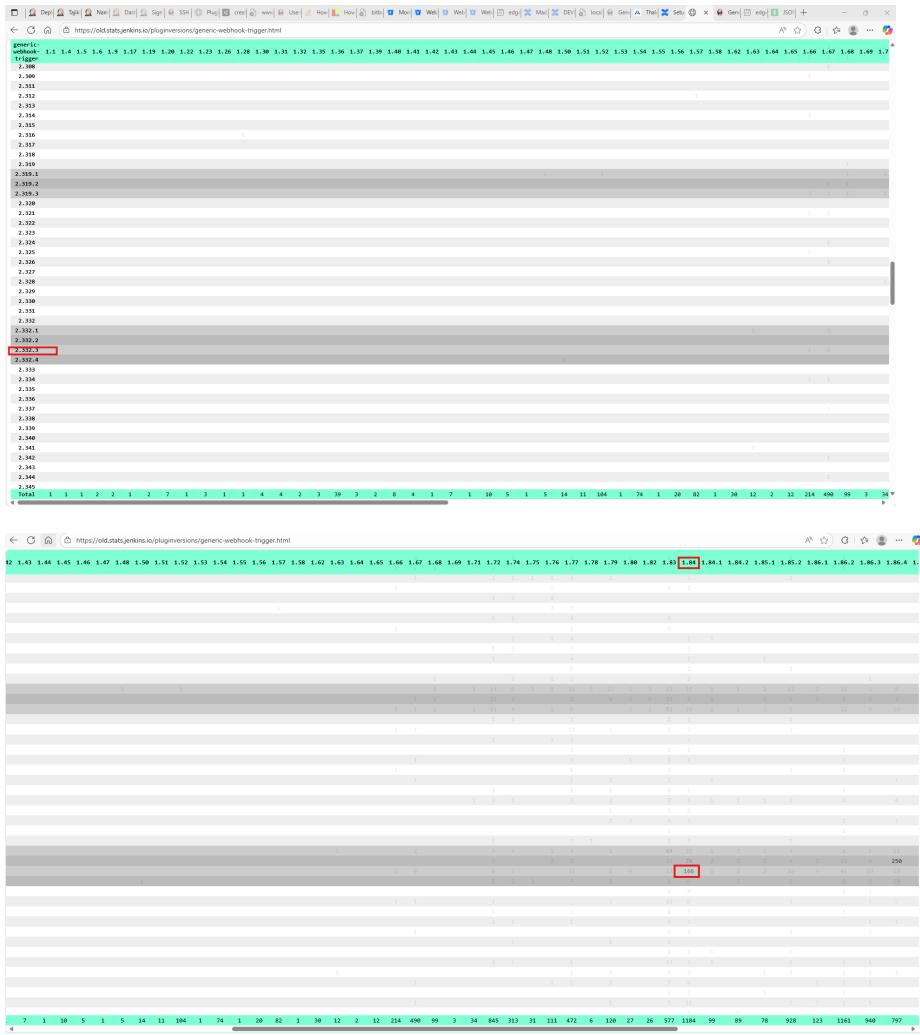
In case you need to install the [plugin](#) in a new instance, the most important thing is to install a plugin version that is compatible with your current jenkins version:

You can check the jenkins version in the bottom right corner:



And check the compatibility chart in the plugin page:

A screenshot of the Jenkins Generic Webhook Trigger plugin page. It shows the plugin's version (2.4.1), release date (3 months ago), and required Jenkins version (2.479.3). A progress bar indicates it is installed on 13.8% of controllers. There are also links to GitHub, Jira, and JavaDoc, and a section for labels.



So version of the plugin we must install is 1.84.