

Setup Jenkins webhook triggers for a project (WIP).

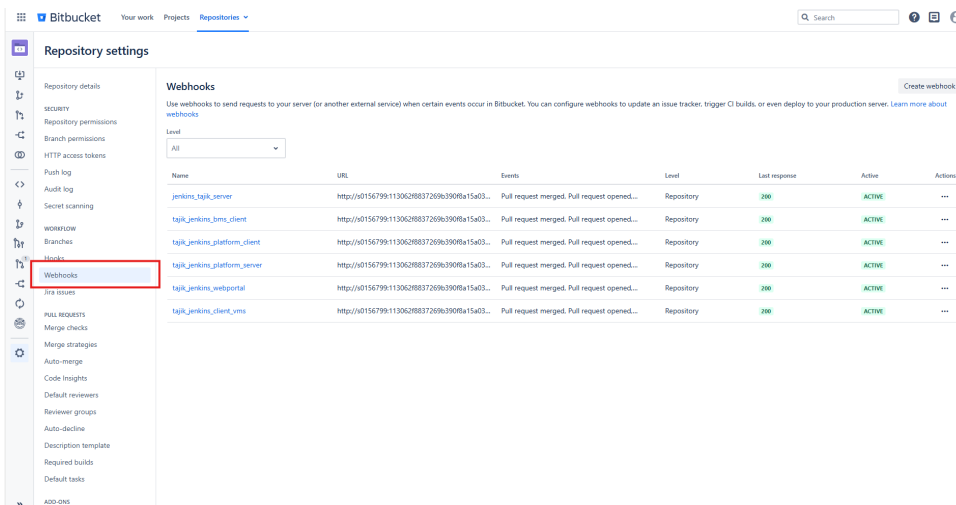
- [Setup Webhooks in BitBucket.](#)
 - [See the events requests.](#)
- [Jenkins Setup](#)
 - [Generate user token](#)
 - [Configure plugin for each of the jobs.](#)
- [Annex: Installing generic-webhook-trigger plugin](#)

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:



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.

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



- The url that needs to be filled looks like this:

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

Url analysis:

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

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	ACTIVE	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

Response

Request details

Event type: pr:merged

URL endpoint: http://s015679911306298837263b3908a15a03f62b145@10.75.105.211:8080/generic-webhook-trigger/invoke?token=namibia_unique_ebms_client

Headers

X-Request-Id: f06c9215-0883-4a18-a54d-6d66e1a23a10

Content-Type: application/json; charset=utf-8

X-Event-Key: pr:merged

Body

```
{
  "date": "2025-11-04T15:28:17+0000",
  "actor": {
    "emailAddress": "jose.seco@external-thalesgroup.com",
    "displayName": "Seco-EXTERNAL Jose",
    "name": "s0156799",
    "active": true,
    "links": [{"rel": ["href"], "https://bitbucket.gemalto.com/users/s0156799"}],
    "id": 40355,
    "type": "NORMAL",
    "slug": "s0156799"
  },
  "eventKey": "pr:merged",
  "pullRequest": {
    "author": {
```

Webhook event details

Request

Response

Response details

HTTP status: 200

Duration: 117ms

Headers

Date: Tue, 04 Nov 2025 14:19:58 GMT

Server: Jetty(9.4.43.v20210629)

X-Content-Type-Options: nosniff

Content-Type: application/json; charset=UTF-8

Via: 1.1 localhost (Apache/HTTPClient/4.5.14 [cache])

Content-Length: 235

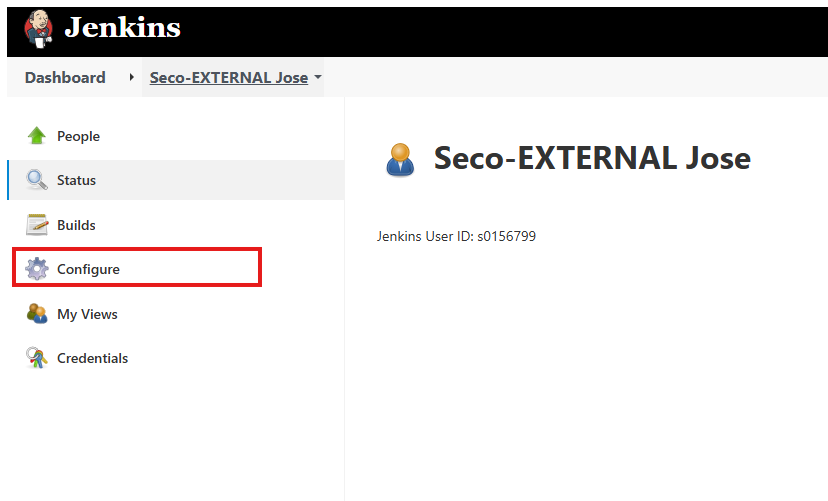
Body

```
{
  "jobs": {
    "Namibia-eBMS-Client-Integration-Git": {
      "regexFilterExpression": "",
      "triggered": true,
      "resolvedVariables": {
        "BRANCH_NAME": "test_ci_jenkins2"
      },
      "regexFilterText": "",
      "id": 8828,
      "url": "queue/item/8828/"
    }
  },
  "message": "Triggered jobs."
}
```

Jenkins Setup

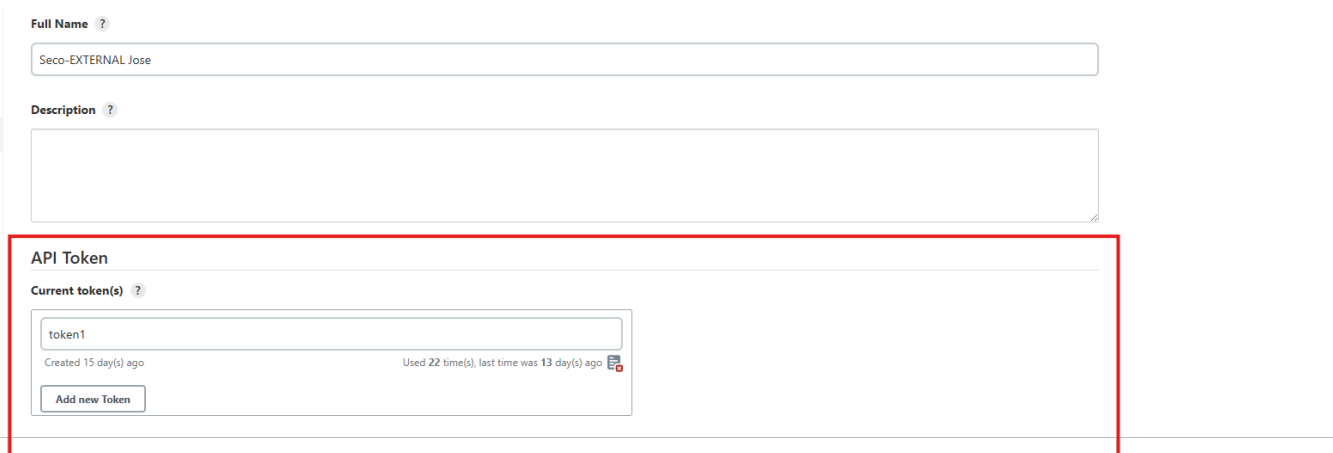
Generate user token

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 image shows the Jenkins user configuration page for a user named 'Seco-EXTERNAL Jose'. The page has a dark header with the Jenkins logo and name. Below the header, there is a navigation bar with 'Dashboard' and 'Seco-EXTERNAL Jose'. On the left side, there is a sidebar with links to 'People', 'Status', 'Builds', 'Configure' (highlighted with a red box), 'My Views', and 'Credentials'. The main content area shows the user's profile with a blue icon, the name 'Seco-EXTERNAL Jose', and the Jenkins User ID: s0156799.

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



The image shows the Jenkins user configuration page for the 'Seco-EXTERNAL Jose' user, specifically the 'API Token' section. The 'Full Name' field is filled with 'Seco-EXTERNAL Jose'. The 'Description' field is empty. The 'API Token' section is highlighted with a red box and contains a table with one token entry:

Current token(s) ?	
token1	Created 15 day(s) ago Used 22 time(s), last time was 13 day(s) ago

Below the table is an 'Add new Token' button.

Configure plugin for each of the jobs.

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](#)

Paste in JSON or a URL, drop a file, browse, or load an example to begin.

JSON Data/URL

```
{  "requests": [    {      "id": "1",      "state": "MERGED",      "locked": false,      "open": false,      "properties": {        "mergeCommit": {          "id": "8937cc388db4dc7cc8b82a46f3985235b99d12d",          "displayId": "8937cc388db"        }      }    }  ]}
```

JSON Template

3 Space Tab

Implementation

JSONPath 0.5.0

JSONPath Expression

.pullRequest.fromRef.displayId

Process

Discover more: [programming languages](#) [Validator](#) [Python](#) [Programming language](#) [Email](#)

#2 November 13th 2025, 11:32:44 am
pullRequest.fromRef.displayId

JSONPath Result

```
{  "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.displayName

☒ 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.

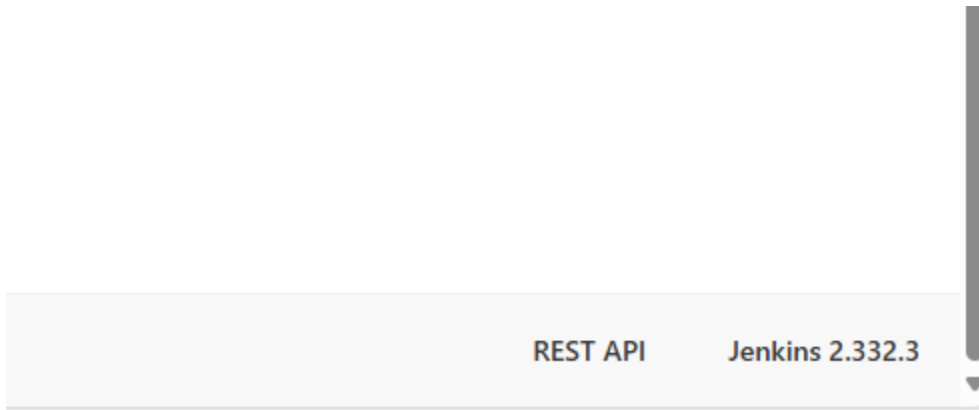
Contributing variables:

```
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:

Jenkinscd

BlogSuccess StoriesContributor SpotlightDocumentationPluginsCommunitySubprojectsSecurityAboutSupport

Generic Webhook Trigger

DocumentationReleasesIssuesDependenciesHealth Score

buildpassing

This is a Jenkins plugin that can:

1. Receive any HTTP request, `30W3NS_URL/generic-webhook-trigger/invoke`

2. Extract values

- From `POST` content with `JSONPath` or `XPath`
- From the `query` parameters
- From the `headers`

3. Trigger a build with those values contribute as variables

There is an optional feature to trigger jobs only if a supplied regular expression matches the extracted variables. Here is an example, let's say the post content looks like this:

```
{  "before": "1846f1236ae15769e6b33e9c4477d8158b03453",  "after": "5cab33339eeb324eb08c7b775e9d27b51ef13d",  "ref": "refs/heads/dev33op"}
```

Then you can have a variable, resolved from post content, named `ref` of type `350WPath` and with expression like `1.ref`. The optional filter text can be set to `3.rer` and the filter regexp set to `^(refs/heads/develop|refs/heads/feature/.+)$` to trigger builds only for develop and feature-branches.

There are more examples of use cases [here](#).

Video showing an example usage:


Version: 2.4.1

Released: 3 months ago

Requires Jenkins 2.479.3

ID: generic-webhook-trigger

Installed on 13.8% of controllers



[View detailed version information](#)

Links

[Github](#)

[Open issues \(Github\)](#)

[Report an issue \(Github\)](#)

[Open issues \(Jira\)](#)

[Report an issue \(Jira\)](#)

[Javadoc](#)

Labels

[bitbucket](#)

[bitbucket-server](#)

[github](#)

[gitlab](#)

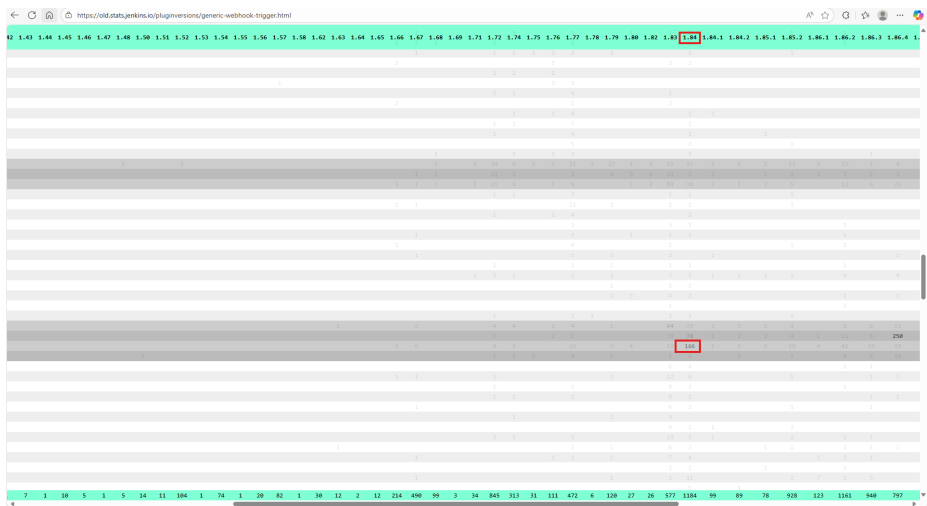
[jira](#)

[notification](#)

[Build Parameters](#)

[Build Triggers](#)

[webhook](#)



So version of the plugin we must install is 1.84.