Integration with Bamboo

- Overview
- Release Notes
- Installation
  - Automatic Installation
  - Manual Installation
- Configuration
  - Jira servers
- Tasks
  - Xray: Cucumber Features Export Task
    - Configuration
  - Xray: Results Import Task
    - Configuration
    - Additional fields
- Examples
  - Cucumber
    - Exporting Cucumber features
    - Importing the execution results
    - Importing the execution results with user-defined field values
  - JUnit
    - Importing the execution results
- Troubleshooting
  - The build process is failing with status code 403

Overview

Xray enables easy integration with Bamboo through the "Xray for JIRA add-on for Bamboo", providing the means for successful Continuous Integration by allowing users to report automated testing results.

Release Notes

- Xray Connector for Bamboo 1.2.2 Release Notes
- Xray Connector for Bamboo 1.2.1 Release Notes
- Xray Connector for Bamboo 1.2.0 Release Notes
- Xray Connector for Bamboo 1.1.3 Release Notes
- Xray Connector for Bamboo 1.1.2 Release Notes
- Xray for JIRA add-on for Bamboo 1.1.1 Release Notes
- Xray for JIRA add-on for Bamboo 1.1.0 Release Notes
- Xray for JIRA add-on for Bamboo 1.0.1 Release Notes
- Xray for JIRA add-on for Bamboo 1.0.0 Release Notes

Installation

The installation can be made either automatically or manually.

Requirements

Requires Java 1.8 to be installed in the Bamboo server.

The app was tested against Bamboo v5.9.x and it may not work properly with previous versions.

Automatic Installation

You can install Xray for Bamboo through the Universal Plugin Manager (UPM). Go to Bamboo's administration, then select Add-ons > Find new add-ons.

Manual Installation

If you have the actual xray-bamboo-X.X.X.jar file,

1. Go to the administration section of Bamboo, look for the Add-Ons > Manage Add-Ons menu.
2. Select option Upload Add-On and upload the xray-bamboo-X.X.X.jar file, where X.X.X is the file version.

Configuration

Xray for Bamboo is configured in Bamboo Administration page, in the Add-Ons > Xray for JIRA configuration.

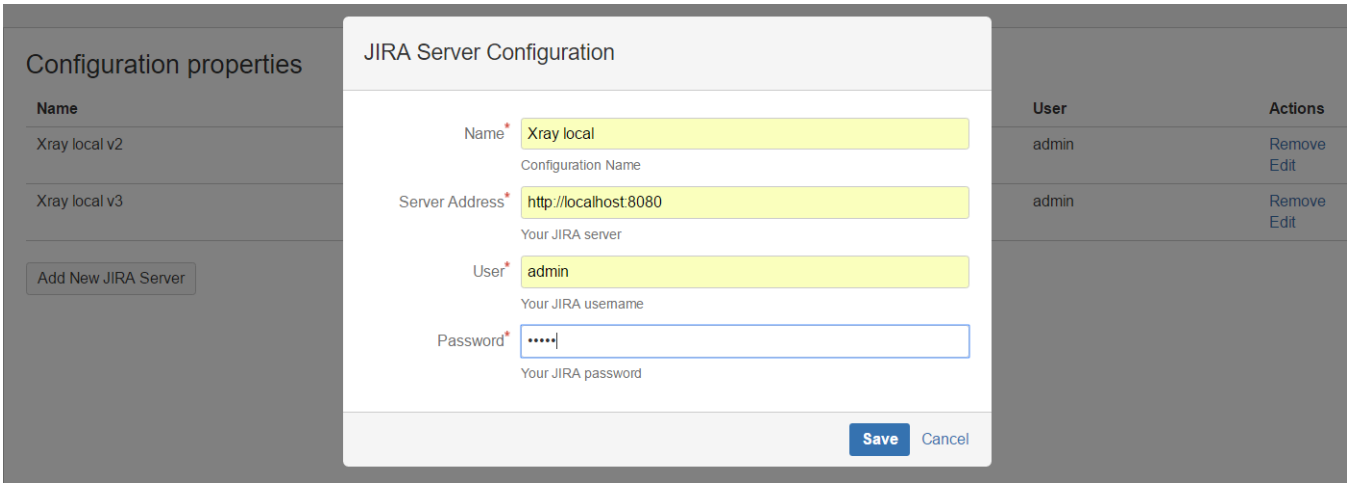
Jira servers

The Jira servers configuration defines connections with Jira instances.

To add a new Jira instance connection, specify these properties:

1. **Name**: Configuration alias
2. **Server Address**: the address of the JIRA Server where Xray is running
3. Authentication information:
    a. **User**: username
    b. **Password**.

> **Please note**
>
> The user present in this configuration must exist in the JIRA instance and have permission to Create Test and Test Execution Issues



Tasks

The app provides a task for exporting Cucumber Scenario/Scenario Outlines from Jira as .feature files, and another task for importing execution results.

> **Please note**
>
> The fields of the tasks may take advantage of Bamboo variables, which can be used to populate fields such as the "Revision" for specifying the source code's revision. For more information, please see Bamboo variables.

Xray: Cucumber Features Export Task

This task will export the Cucumber Tests (i.e., Scenario/Scenario Outlines) in .feature or bundled in a .zip file. The rules for exporting are defined here.

It invokes Xray's Export Cucumber Tests REST API endpoint (see more information here).

Configuration

Some fields need to be configured in order to export the Cucumber Tests. As input, you can either specify issue keys (see the endpoint documention here) or the ID of the saved filter in Jira.

field	description
Task description	A short task description
Jira instance	The Jira instance where Xray is running
Issue keys	Set of issue keys separated by ";"
Filter ID	A number that indicates the filter ID
File path	The relative path of the directory where the features should be exported to; normally, this corresponds to the "features" folder of the Cucumber project that has the implementation steps. Note: The directory will be created if it does not exist.

Xray: Results Import Task

The app provides easy access to Xray's Import Execution Results REST API endpoints (see more information here). Therefore, it mimics the endpoints input parameters.

It supports importing results in Xray's own JSON format, Cucumber, Behave, JUnit, and NUnit, among others.

Configuration

field	description
Task description	A short task description
Jira instance	The Jira instance where Xray is running
Format	A list of test result formats and it's specific endpoint
Execution Report File	The relative path of results file, including file name. Note: regex is not supported.

Additional fields

Depending on the chosen test result format, some additional fields may need to be configured.

format and specific endpoint	field	description
Behave JSON multipart Cucumber JSON multipart NUnit XML multipart JUnit XML multipart Robot XML multipart	Test execution fields	An object (JSON) specifying the fields for the issue. You may either specify the object directly in the field or in the file path. **Learn more** The custom field IDs can be discovered using the Jira REST API Browser tool included in Jira. Each ID is of the form "**customfield_ID**". Another option, which does not require Jira administration rights, is to invoke the "Get edit issue meta" in an existing issue (e.g., in a Test issue) as mentioned here. Example: GET http://yourserver/rest/api/2/issue/CALC-1/editmeta

NUnit XML JUnit XML Robot XML	Project key	Key of the project where the Test Execution (if the **Test Execution Key** field wasn't provided) and the Tests (if they aren't created yet) are going to be created
	Test execution key	Key of the Test Execution
	Test plan key	Key of the Test Plan
	Test environments	List of Test Environments separated by ";"
	Revision	Source code's revision being targeted by the Test Execution
	Fix version	The Fix Version associated with the test execution (it supports only one value)

Please note

The **Xray Import Execution Results** task should be configured as a final task in order to ensure that results are imported to Xray even when there are failures during test execution.

Examples

Cucumber

In a typical Cucumber Workflow, after having created a Cucumber project and the Cucumber tests specified in Jira, you may want to have a plan that **exports** the features from Jira, executes the automated tests on a CI environment, and then **imports** back its results.

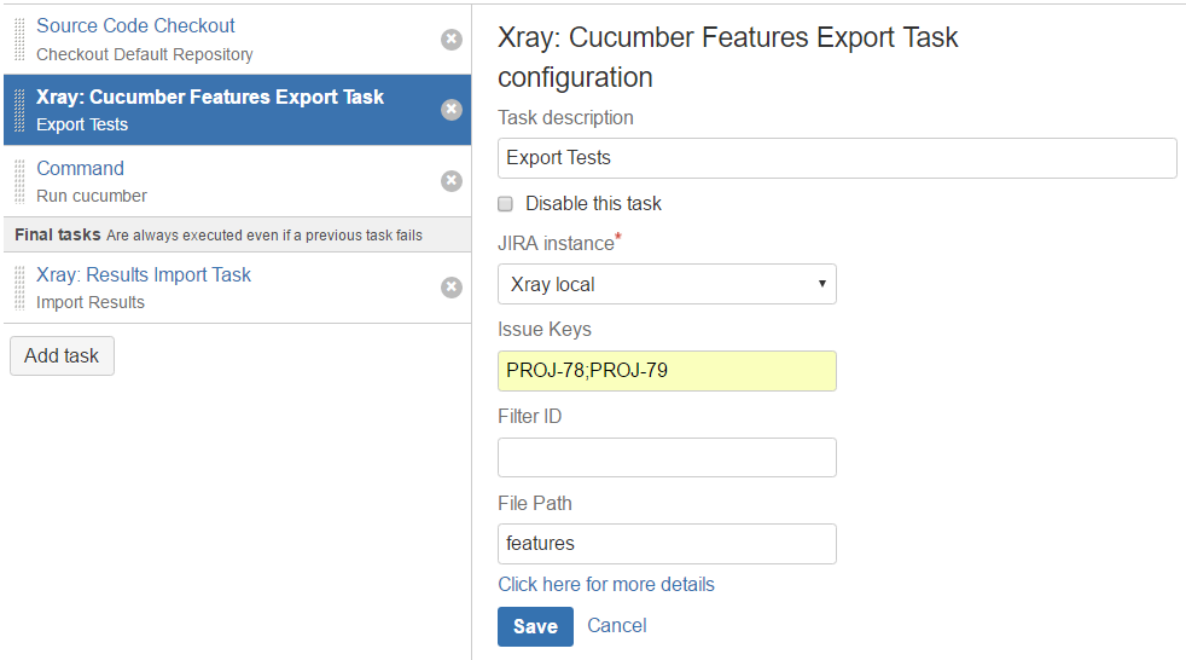
For this scenario, the Bamboo plan would be configured with a set of tasks responsible for:

1. Pulling the Cucumber project
2. **Exporting Cucumber features from Jira to your Cucumber project**
3. Executing the tests in the CI environment
4. **Importing the execution results back to Jira**

Exporting Cucumber features

To start the configuration, add the task *Xray: Cucumber Features Export Task.* After that, configure it and click Save.

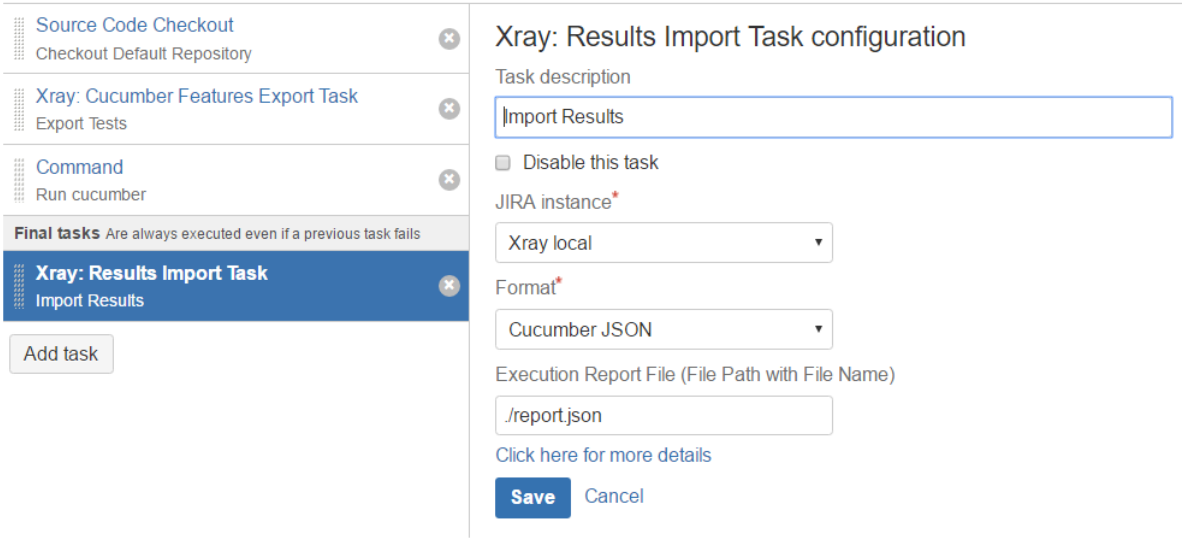
In this example, we configured the task to extract the *features* from a set of issues (PROJ-78 and PROJ-79) to the folder that holds the Cucumber project.



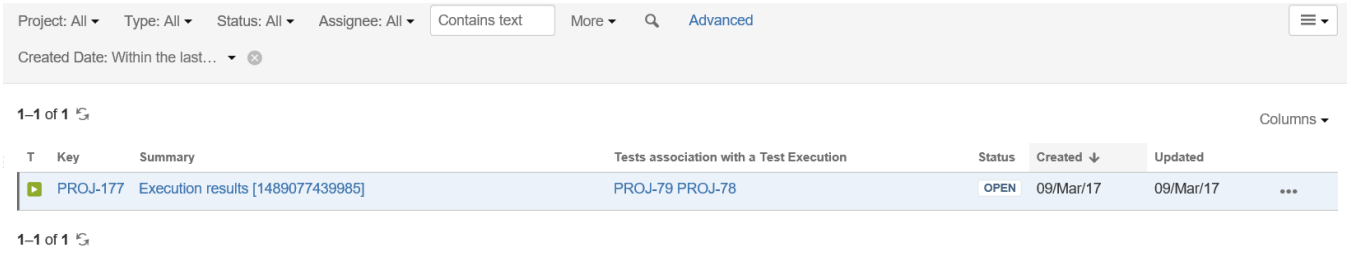
Importing the execution results

To start the configuration, add the task *Xray: Results Import Task*. After that, configure it and click Save.

In this example, we configured the task to import the **Cucumber JSON** results back to Jira.



After running the plan, the expected result is a new Test Execution issue created in the Jira instance.



Importing the execution results with user-defined field values

For Cucumber, Behave, JUnit, NUnit and Robot, Xray for Bamboo allows you to create new Test Executions and have control over newly-created Test Execution fields. You can send two files, the normal execution result file and a JSON file similar to the one Jira uses to create new issues. More details on how Jira creates new issues can be found here.

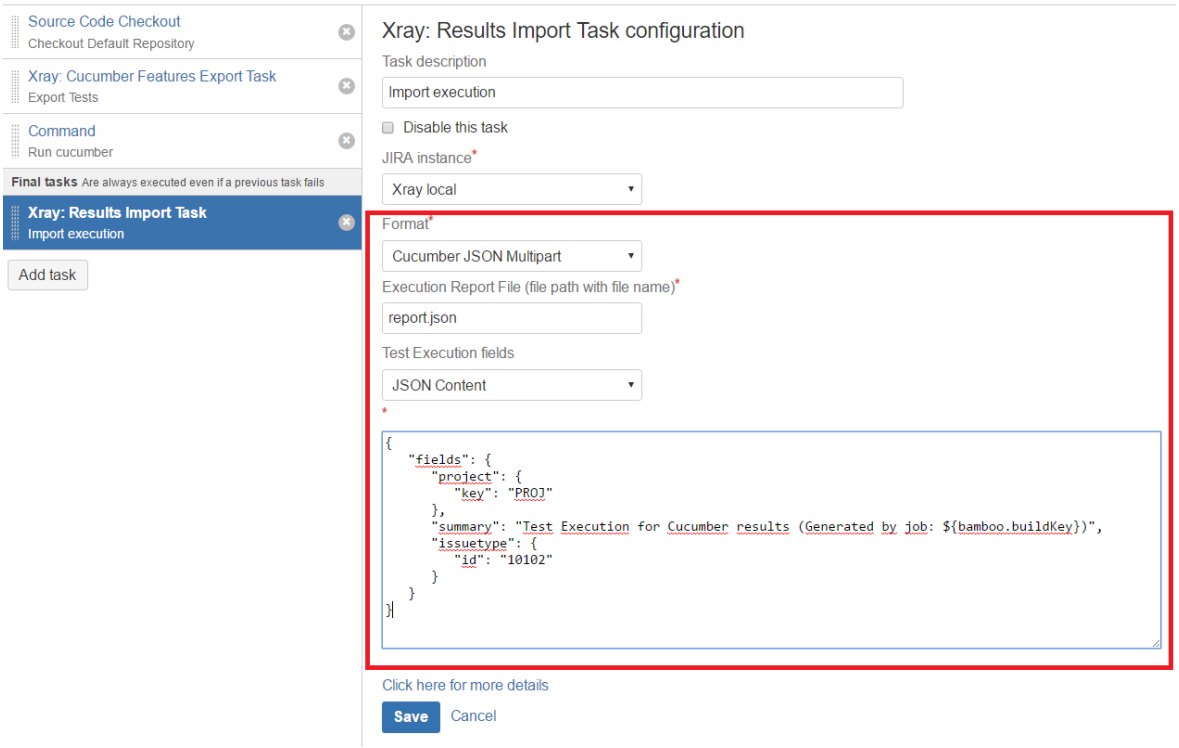
For this scenario and example, the import task needs to be configured with **Cucumber JSON Multipart** format. When selecting this option, you can, additionally, configure the *Test Execution fields* in two ways:

- Insert the relative **path** to the JSON file containing the information, or
- Insert the **JSON content** directly in the field.

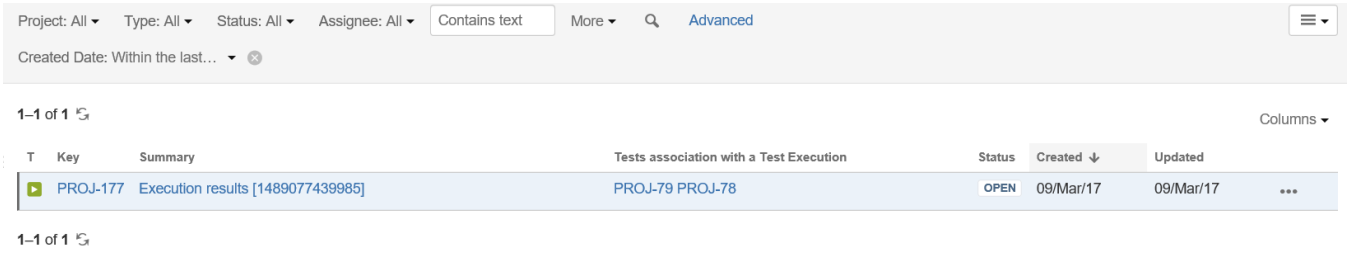
In this example, we configured the following object:

```
{
   "fields": {
      "project": {
         "key": "PROJ"
      },
      "summary": "Test Execution for Cucumber results (Generated by job: ${bamboo.buildKey})",
      "issuetype": {
         "id": "10102"
      }
   }
}
```

And configured the task to import the **Cucumber JSON Multipart** results back to Jira.



After running the plan, the expected result is a new Test Execution issue created in the Jira instance, with the Test Execution fields as specified in the Bamboo task configuration.



JUnit

Apart from supporting Cucumber natively, Xray enables you to take advantage of many other testing frameworks like JUnit. In this sense, Xray for Bamboo lets you import other results in other formats besides Cucumber JSON.

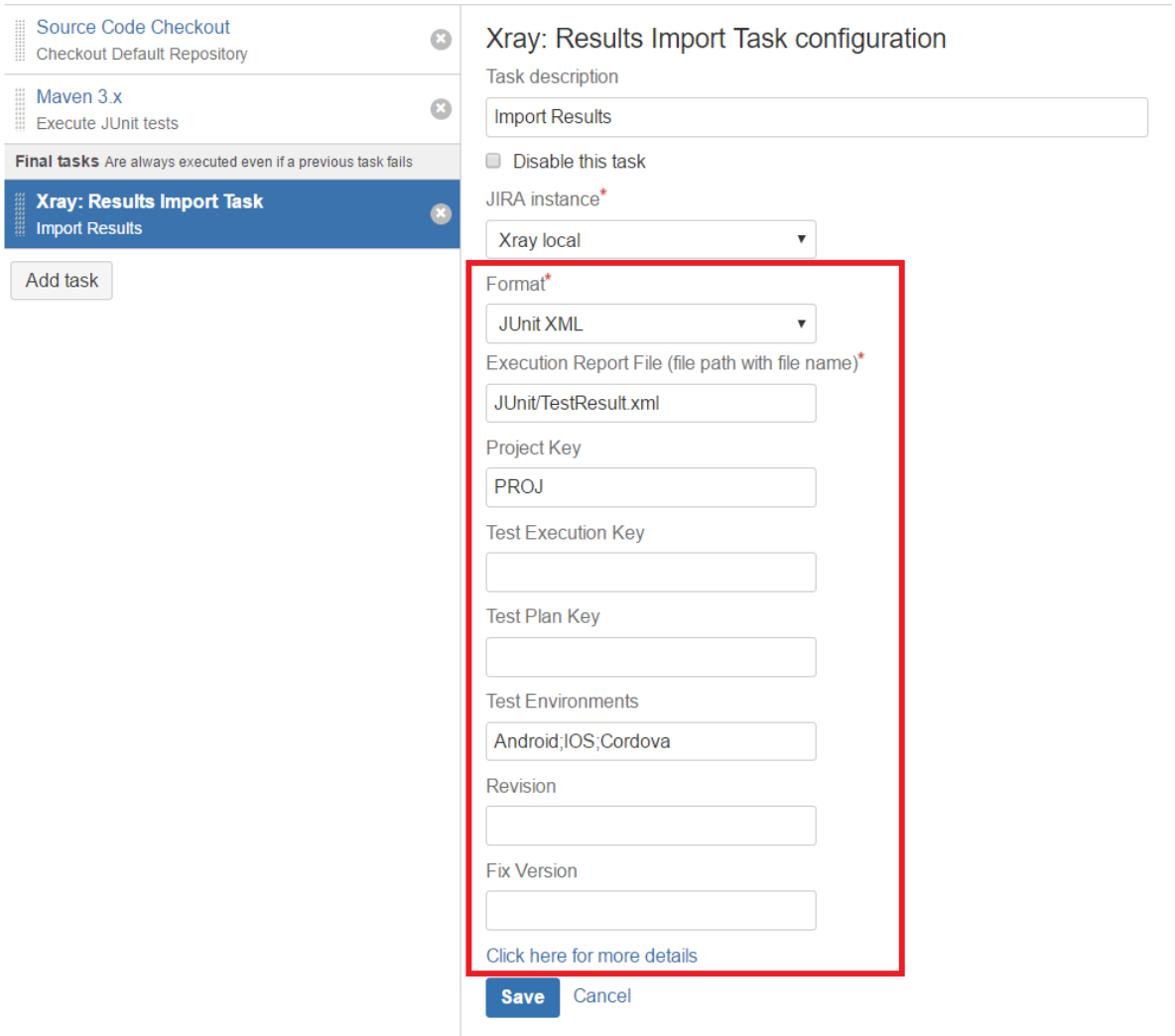
If you want to import **JUnit XML reports,** a typical Job outline would be:

1. Pulling the JUnit project
2. Executing the tests in the CI environment
3. **Importing the execution results, including Tests, to Jira.**

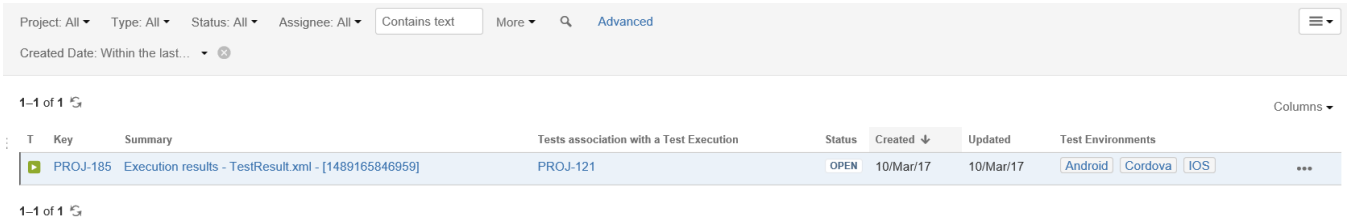
Importing the execution results

To start the configuration, add the task *Xray: Results Import Task*. After that, configure it and click Save.

In this example, we have a configuration where the **JUnit XML** format is chosen.



After running the plan, the expected result is a new Test Execution issue created in the Jira instance.



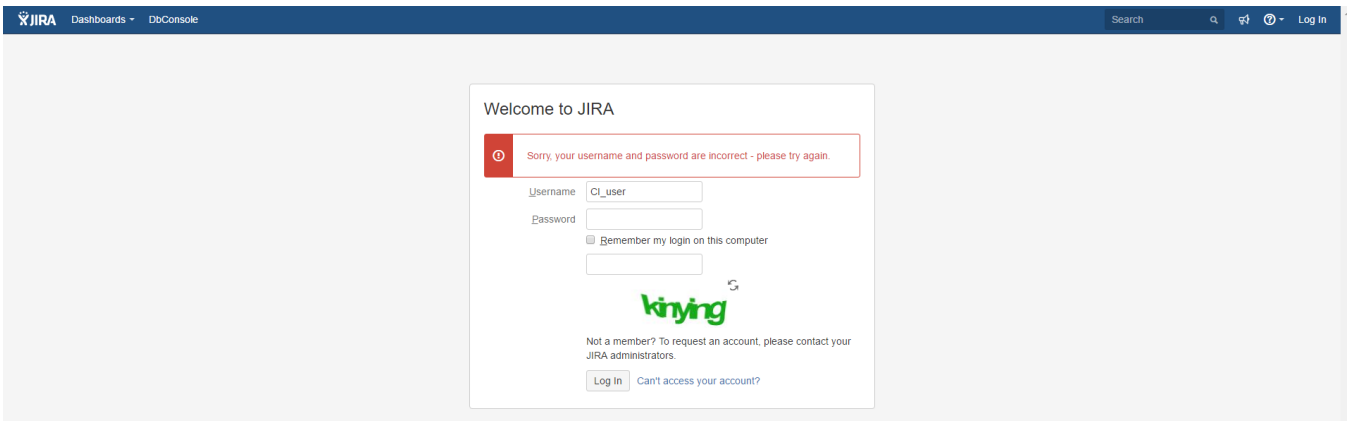
Troubleshooting

*The build process is failing with status code **403***

When you check its log, it has the following:

By default, when you have successively tried to log into Jira with the wrong credentials, the Jira instance will prompt you to provide a CAPTCHA the next time you try to do so. It is not possible to provide this information via the build process, so it will fail with status code **403 Forbidden**.

You will need to log in to Jira via the browser and provide the CAPTCHA.



If you are a Jira administrator, you can go to Jira administration > User Management and reset the failed login.