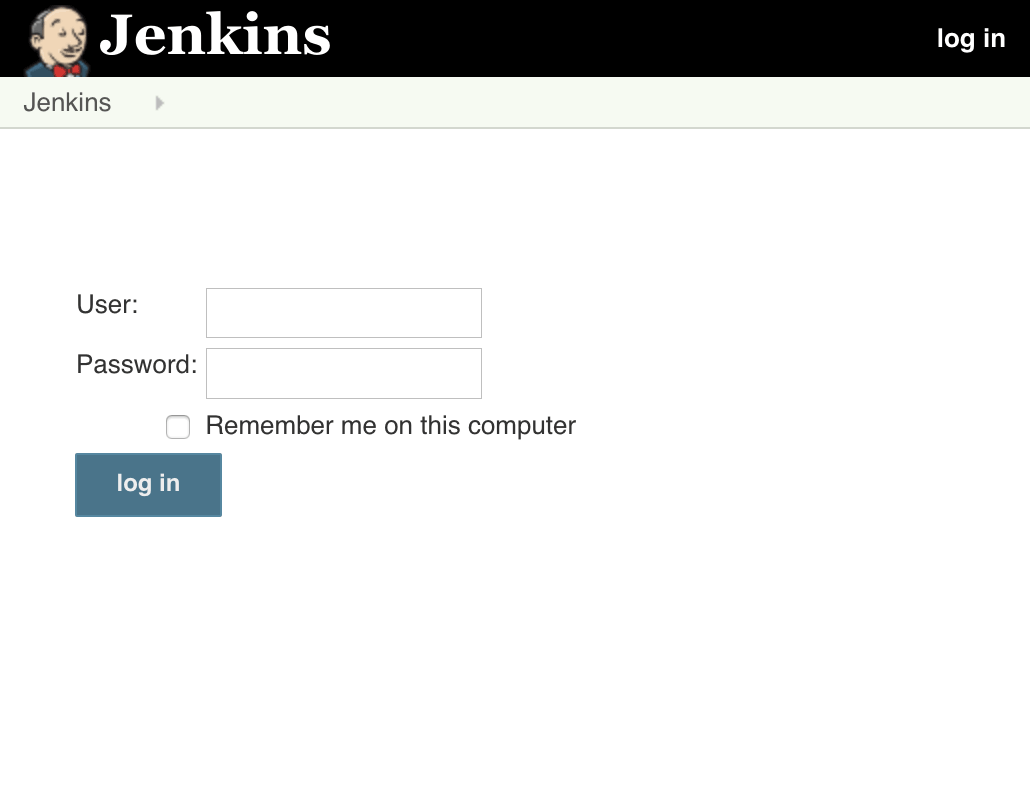
### **­Lab : Create A Continuous Integration Pipeline With Jenkins And GitHub**

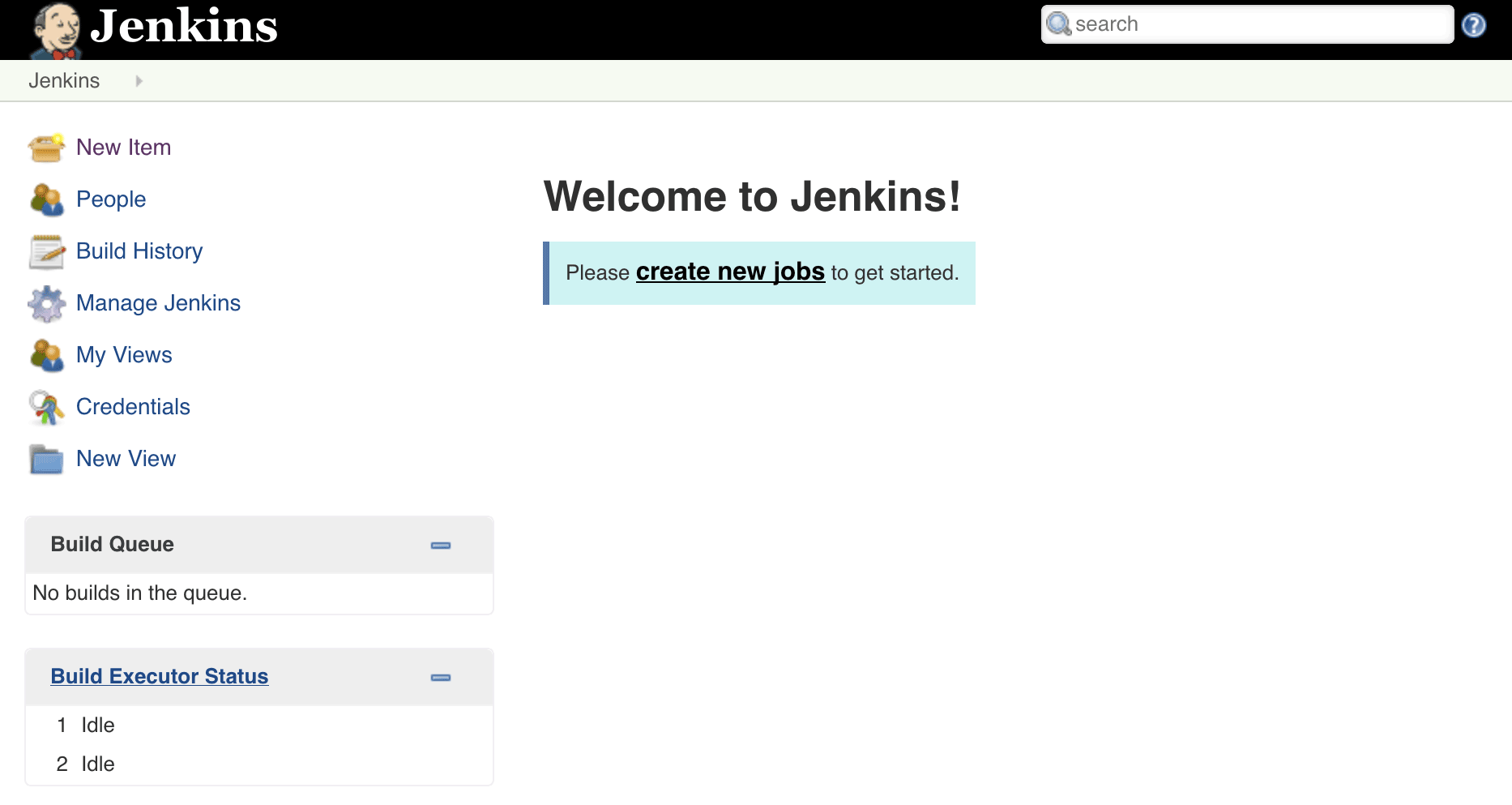
### **Assumptions And Pre-Requisites**

* You have launched a Jenkins
* You have a GitHub

### **Step 1: Log In To Jenkins**

[](https://docs.bitnami.com/images/img/apps/jenkins/login-jenkins.png)

­

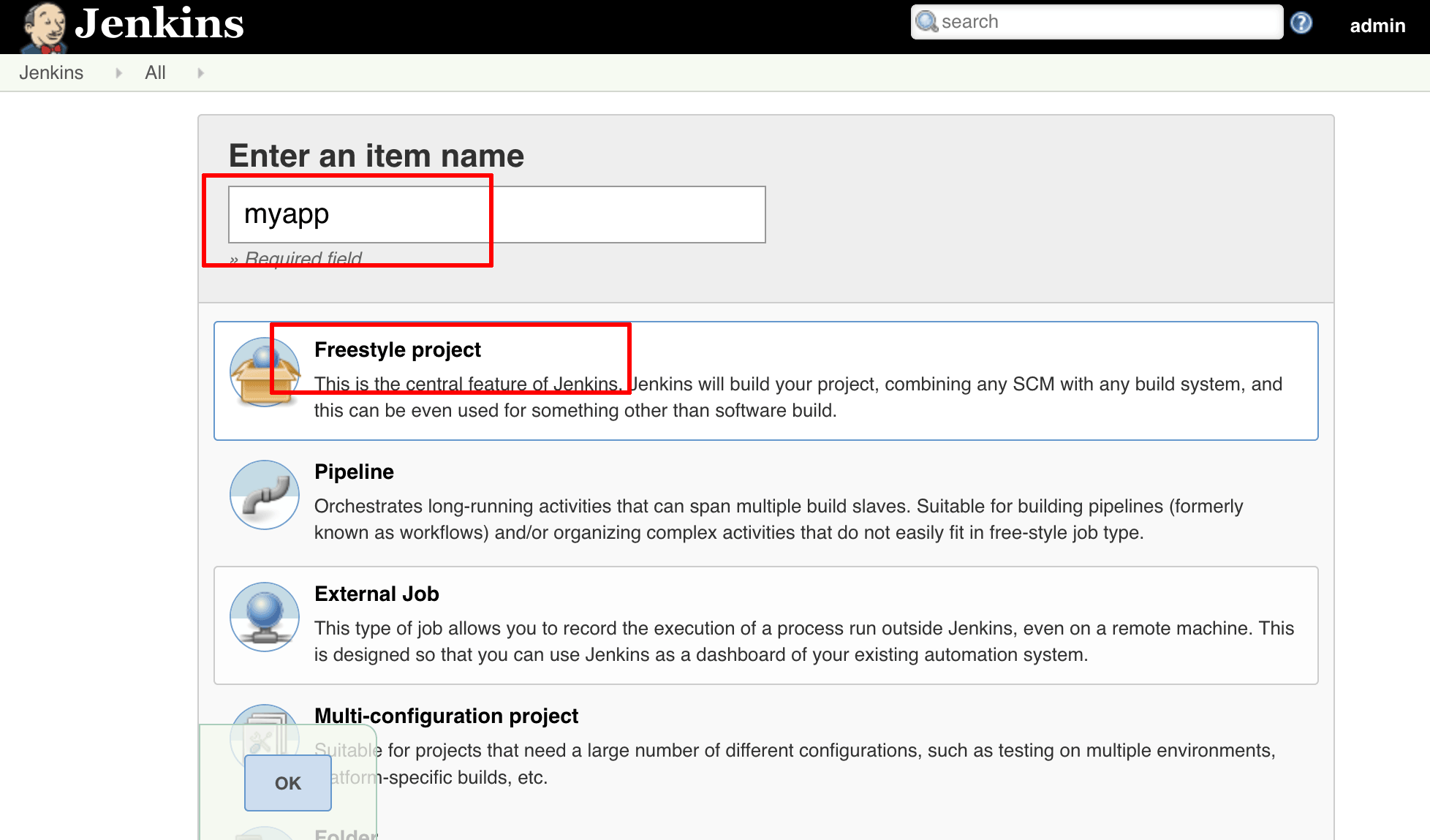
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-3.png)

The Jenkins dashboard is your primary control point for interacting with Jenkins. It includes controls for creating and running builds, inspecting build output, working with credentials and configuring Jenkins. You’ll notice that at the moment, there are no projects or queued builds. You’ll change that in the next step.

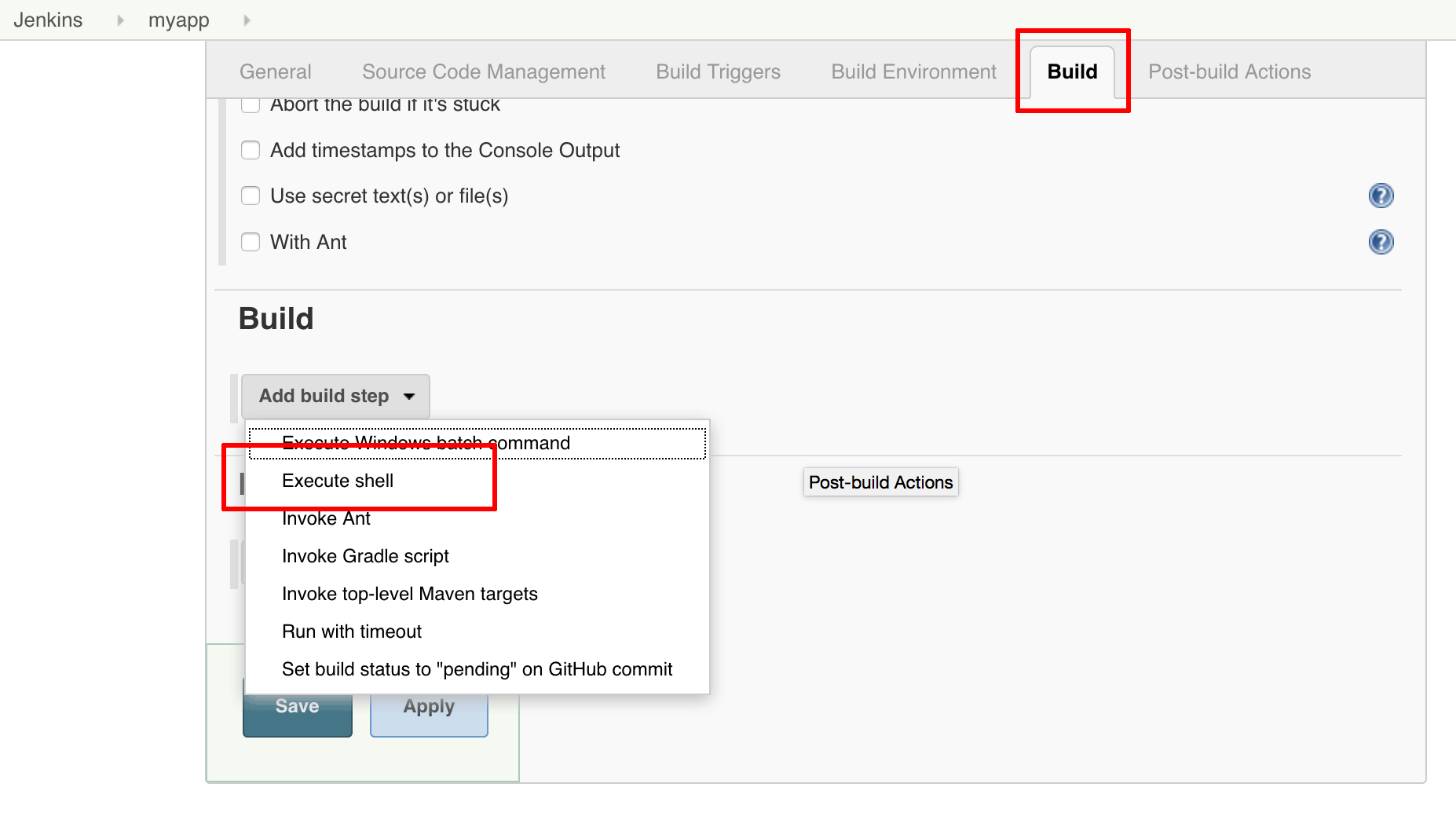
### **Step 2: Create And Build A Simple Project**

Jenkins makes it easy to automatically build, test and deploy software. It does this by allowing you to define “projects” and then specifying “steps” to build, test and deploy them. It also lets you automatically trigger builds on specific events (such as a repository change) or on a fixed schedule, and perform predefined actions (such as sending an email notification) on build success or failure.

To see how this works in practice, begin by clicking the “New Item” link in the Jenkins dashboard. You should be presented with a form to define a new project, as shown below. Enter a name for the new project (in this example, “myapp”) and select “Freestyle project” as the project type. Click “OK” to proceed.

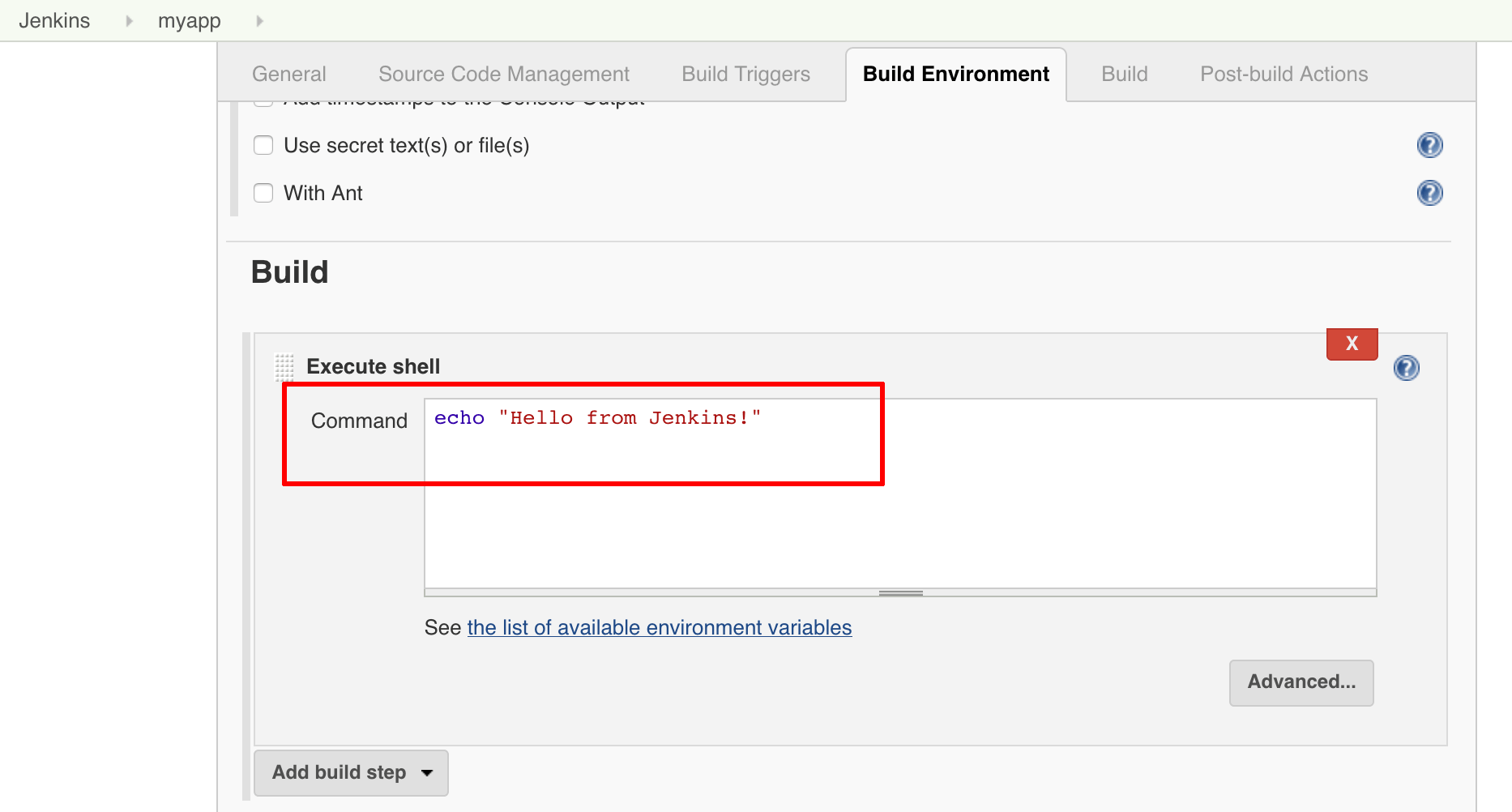
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-4.png)

You should now see the project configuration page, which allows you to define various aspects of the build. For the moment, leave everything at default values and head for the “Build” section. Click the “Add build step” button and add a step to “Execute shell”.

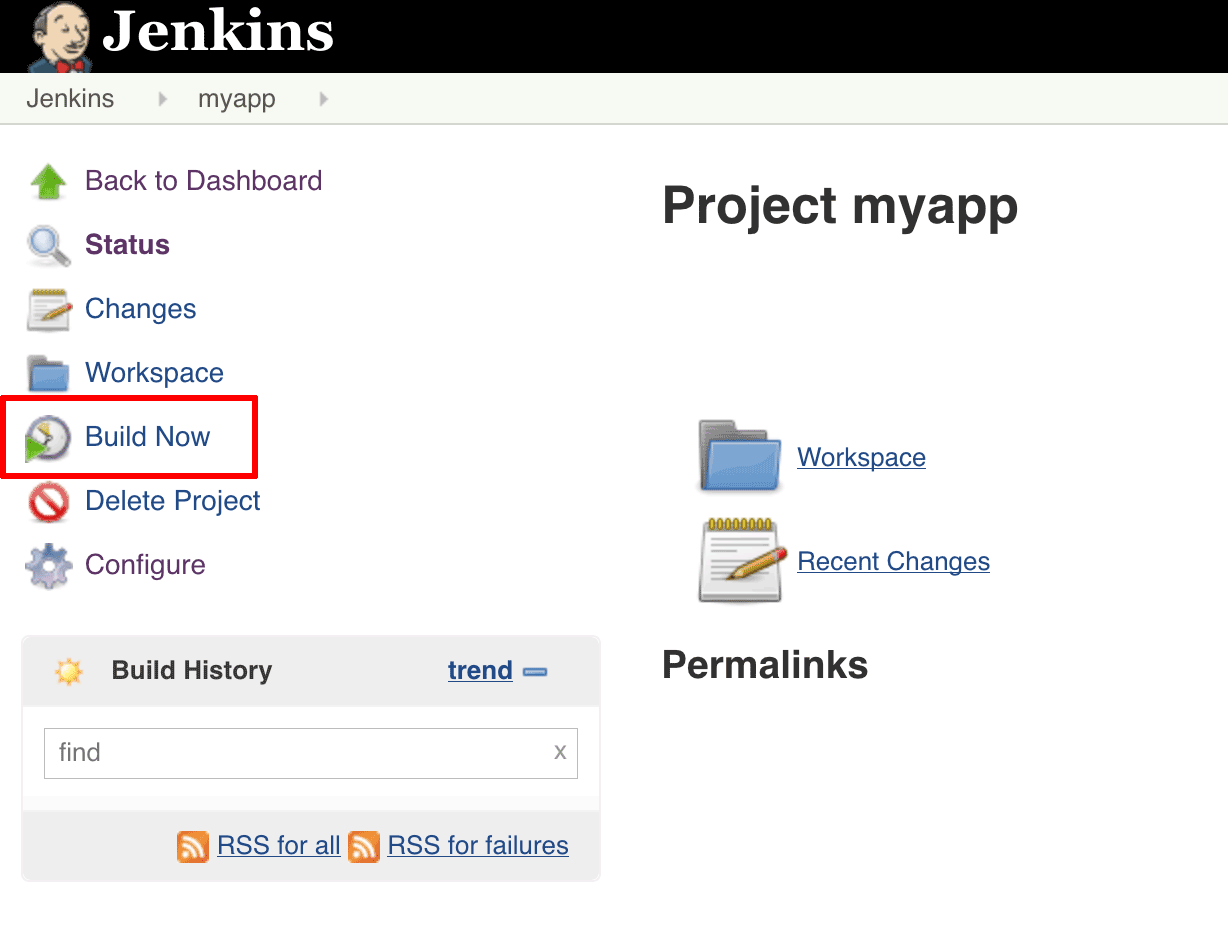
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-5.png)

In the resulting “Command” box, enter a shell command that displays output, like the one below, and click “Save” to save the changes:

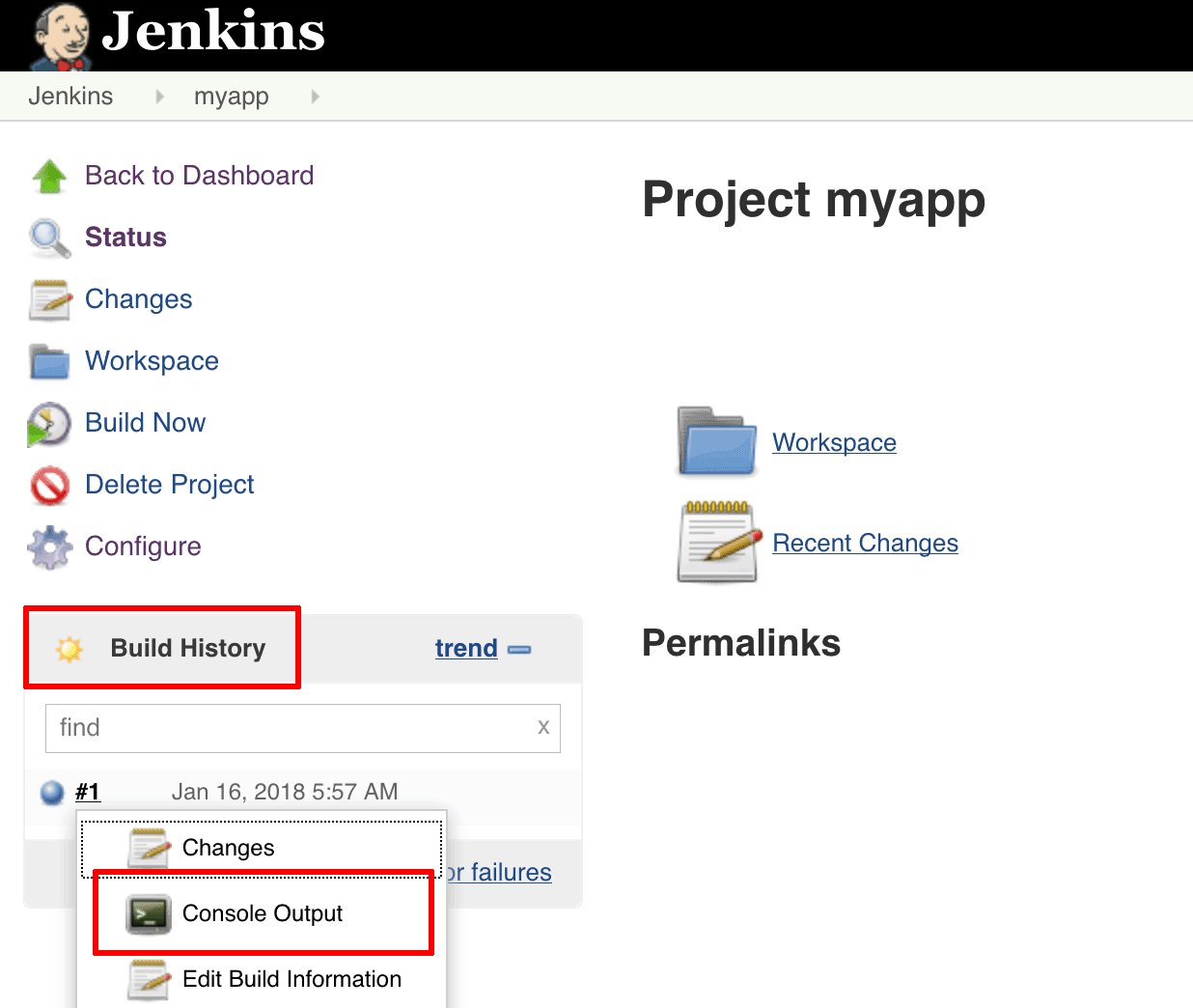
echo "Hello from Jenkins!"

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-6.png)

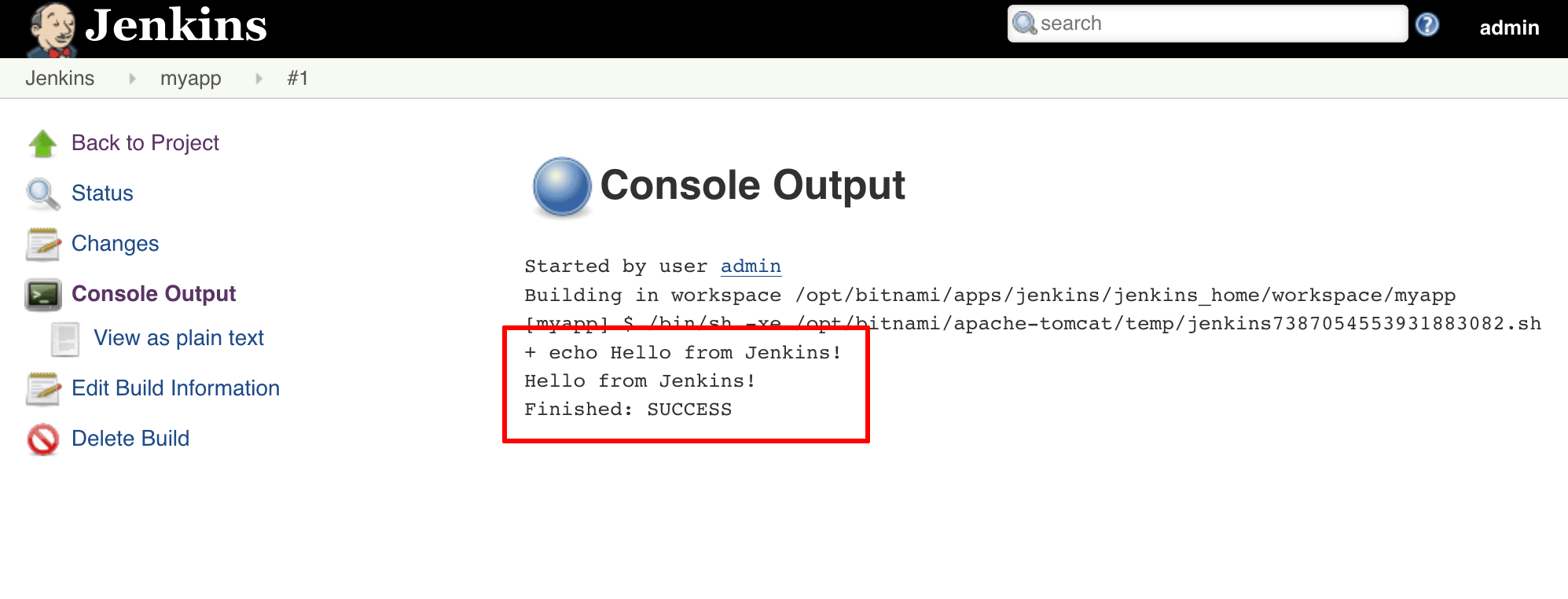
Head back to the project page and click the “Build now” link.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-7.png)

Jenkins should begin building the project. The “Build History” panel in the bottom left corner of the page will show you the status of the build (blue indicates success, red indicates failure).

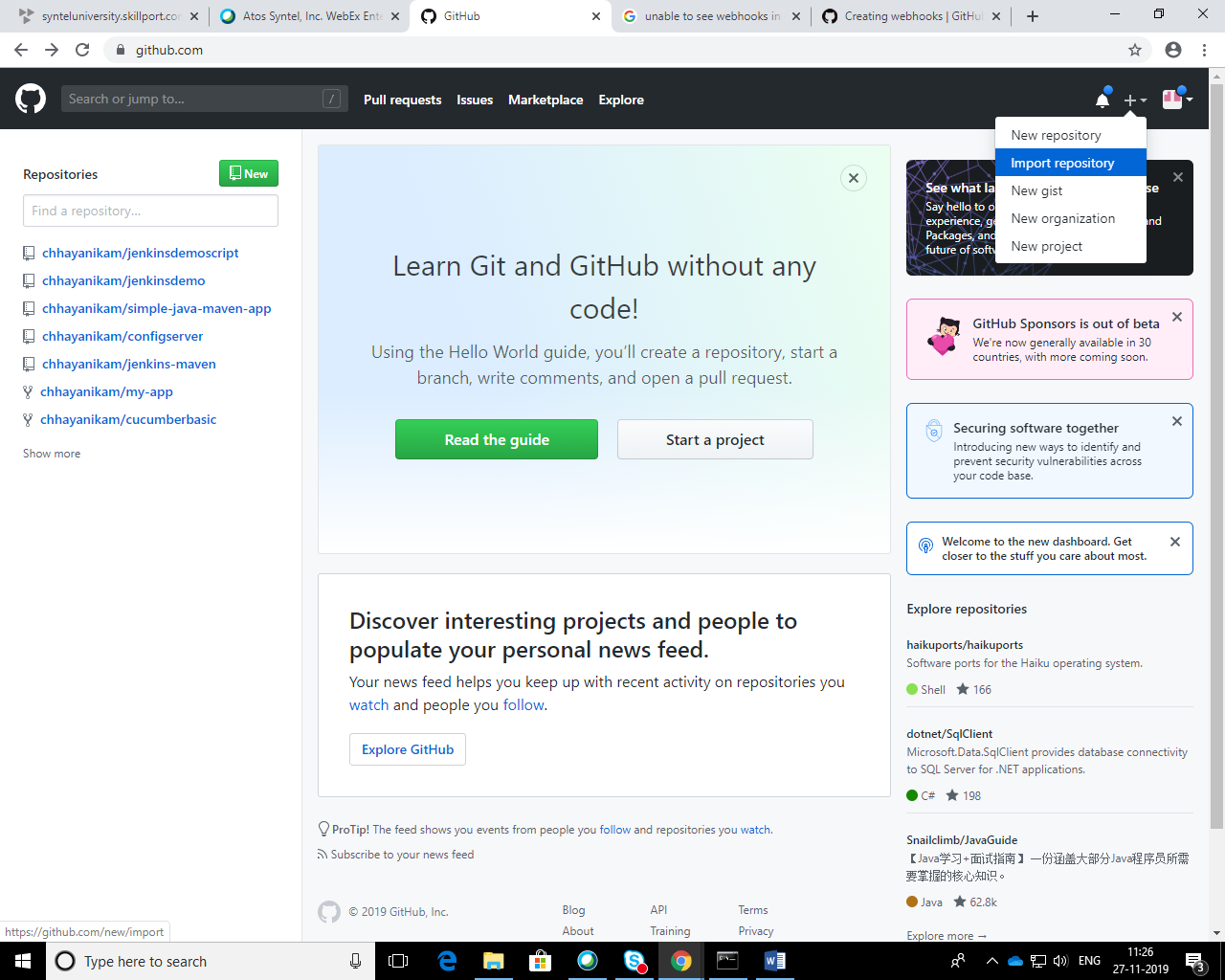
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-8.png)

Once the build is complete, select the “Console Output” menu item to access a detailed build log. The build log should display the output of the shell command you specified earlier.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-9.png)

Congratulations! You just completed your first successful build with Jenkins.

### **Step 3: Build A GitHub Project**



Project to import:

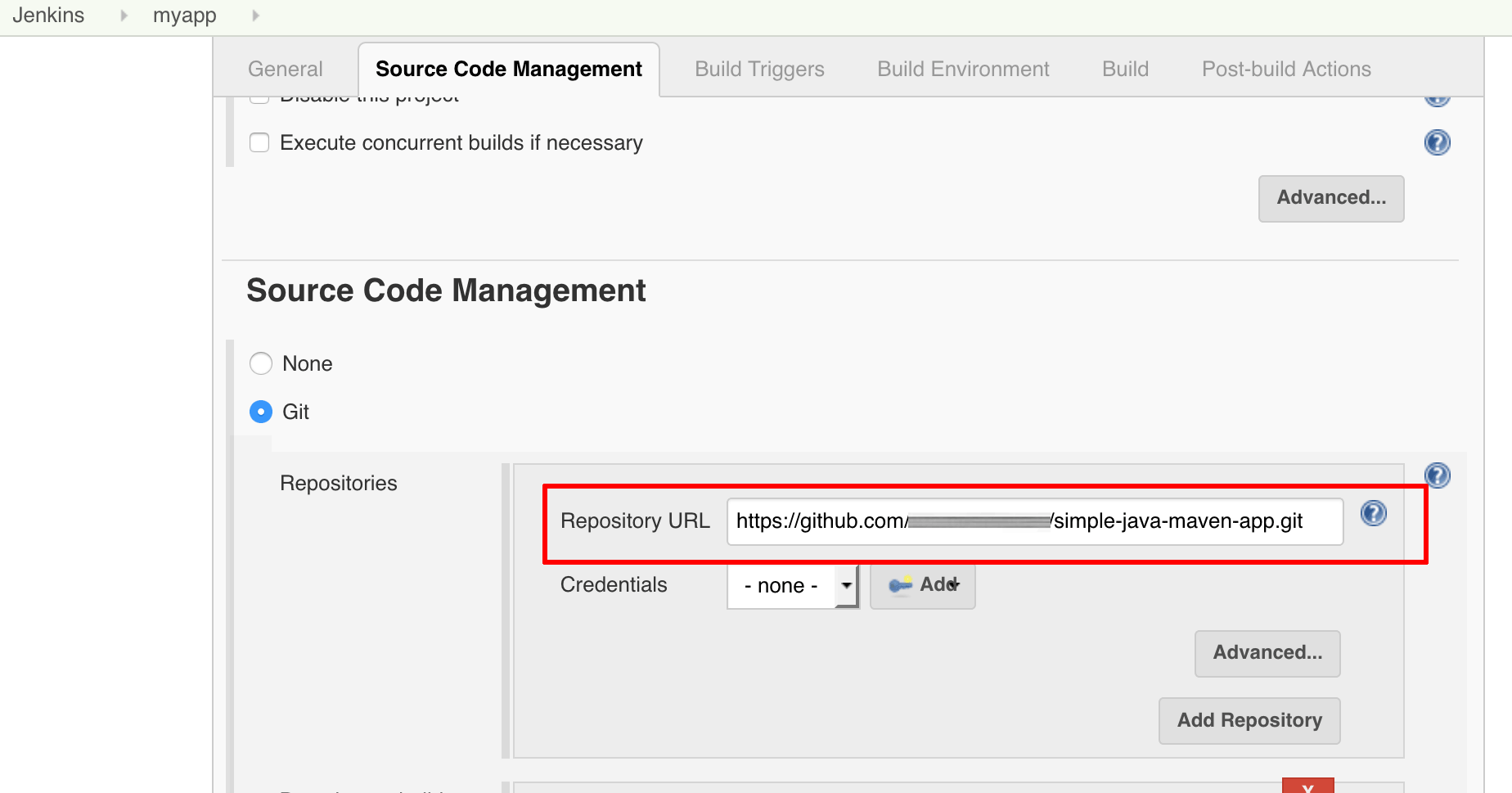
<https://github.com/jenkins-docs/simple-java-maven-app>

My cloned repository (use import project from git)

<https://github.com/chhayanikam/simple-java-maven-app>

NOTE: This guide uses a simple Java project hosted on GitHub as an example. However, you can follow the steps described here for any public GitHub project, remembering to ensure that all the necessary build tools are correctly configured in Jenkins.

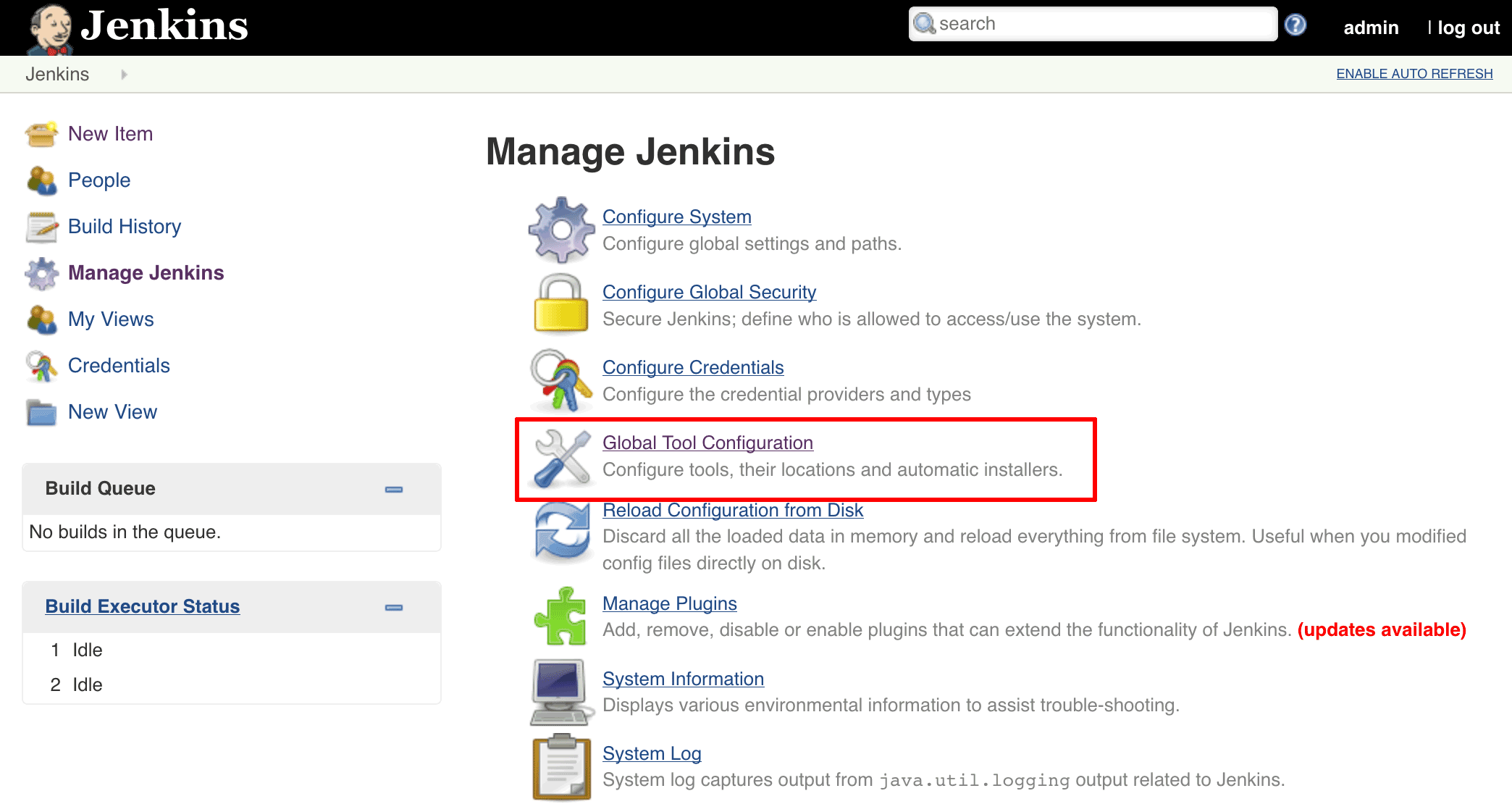
* Click the Jenkins logo to access the dashboard. Select the project you created in [Step 2](https://docs.bitnami.com/oci/how-to/create-ci-pipeline-jenkins-oracle/#step-2-create-and-build-a-simple-project) and click the “Configure” button for the project.
* On the project configuration page, in the “Source Code Management” section, select “Git” as the option and enter the clone URL for your GitHub repository.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-14.png)

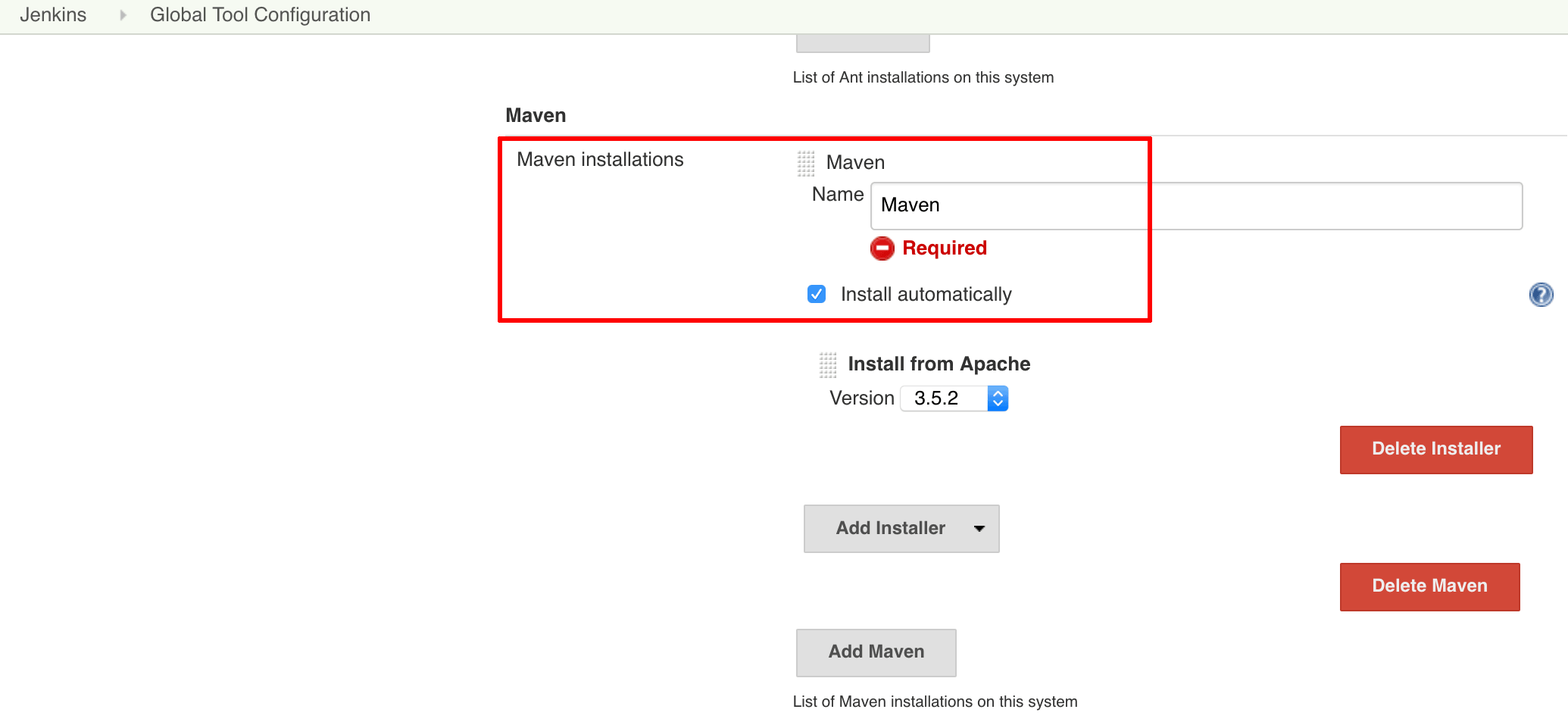
The example project requires [Maven](https://maven.apache.org/) and so, before you can build it, you must install Maven in Jenkins.

Install the Maven Integration Plugin (Manage Jenkins🡪 Manage Plugins)

* Click the Jenkins logo to return to the Jenkins dashboard, then click the “Manage Jenkins -> Global Tool Configuration” option.

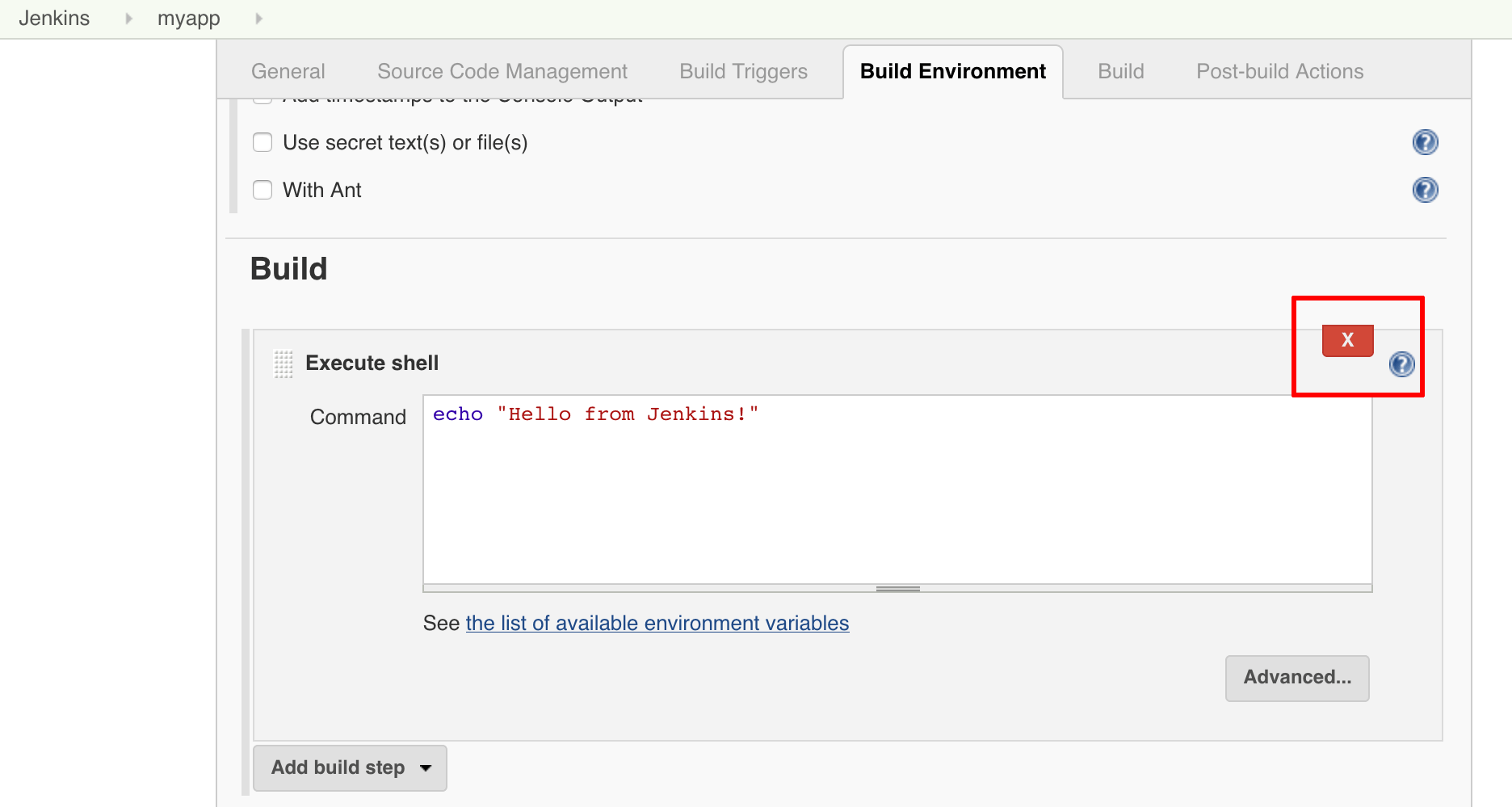
[](https://docs.bitnami.com/images/img/apps/jenkins/tool-configuration-jenkins.png)

* On the “Global Tool Configuration” page, find the “Maven” section and click the “Add Maven” button. Enter a name for the new Maven installation (here, simply “Maven”) and ensure that the “Install automatically” button is ticked. Click “Save” to save the changes and install Maven.

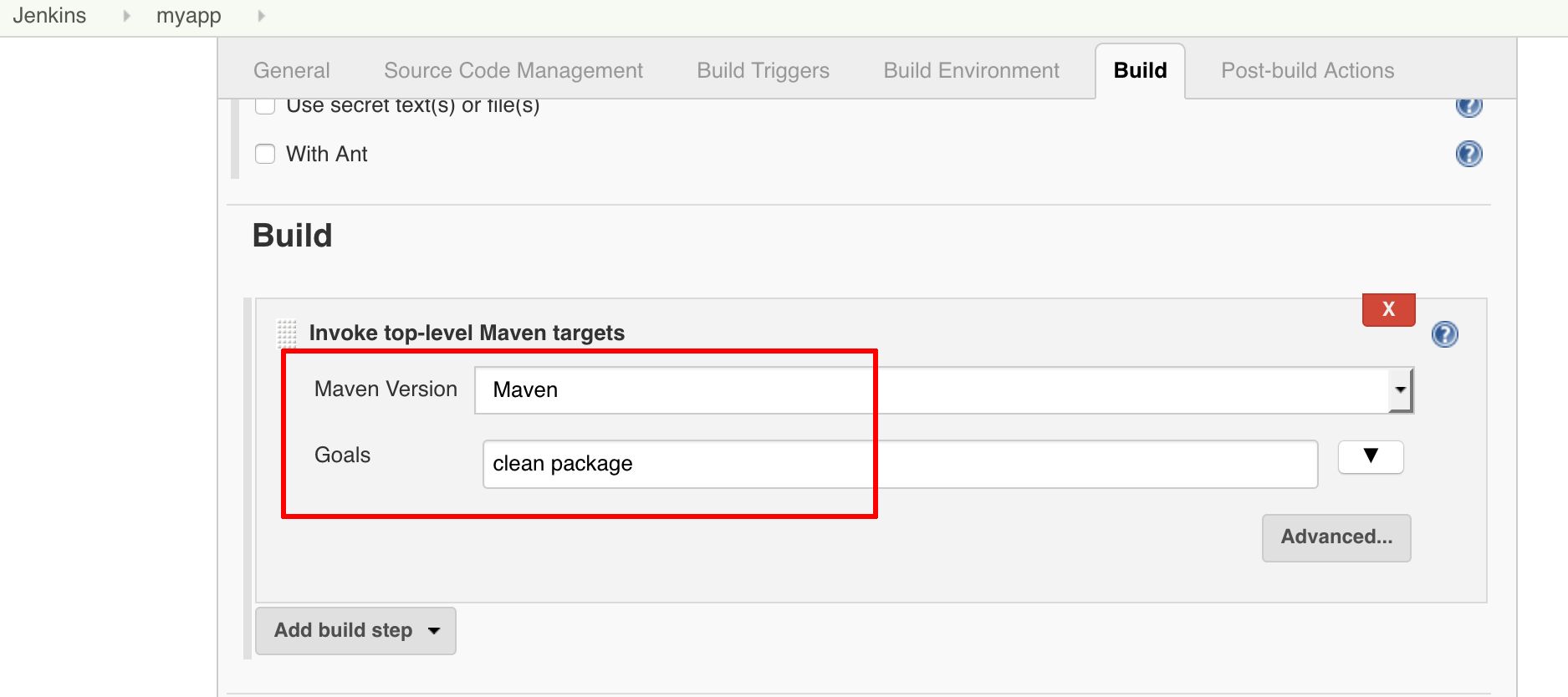
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-13.png)

Once Maven is installed, you can proceed to configure Jenkins to download and build the example application.

* Return to the project configuration page, and in the “Build” section, remove the “Execute shell” build step created in [Step 2](https://docs.bitnami.com/oci/how-to/create-ci-pipeline-jenkins-oracle/#step-2-create-and-build-a-simple-project), by clicking the red “X” in the top right corner of the section.

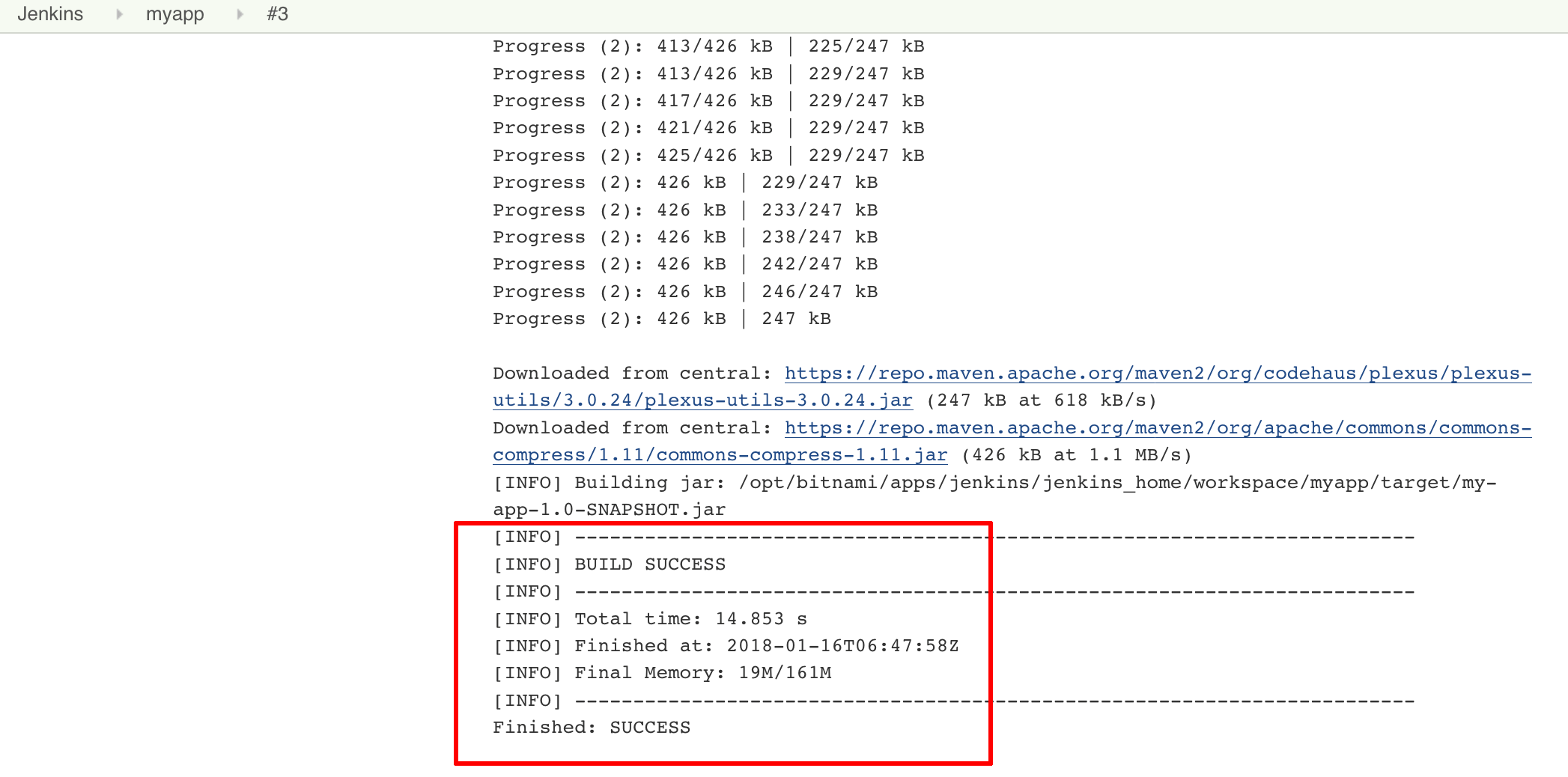
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-15.png)

* Add a new build step to “Invoke top-level Maven targets”.
  + In the “Maven Version” field, select the name of the Maven installation you just created.
  + In the “Goals” field, specify the Maven goals as “clean package”.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-16.png)

* Click “Save” to save the changes.

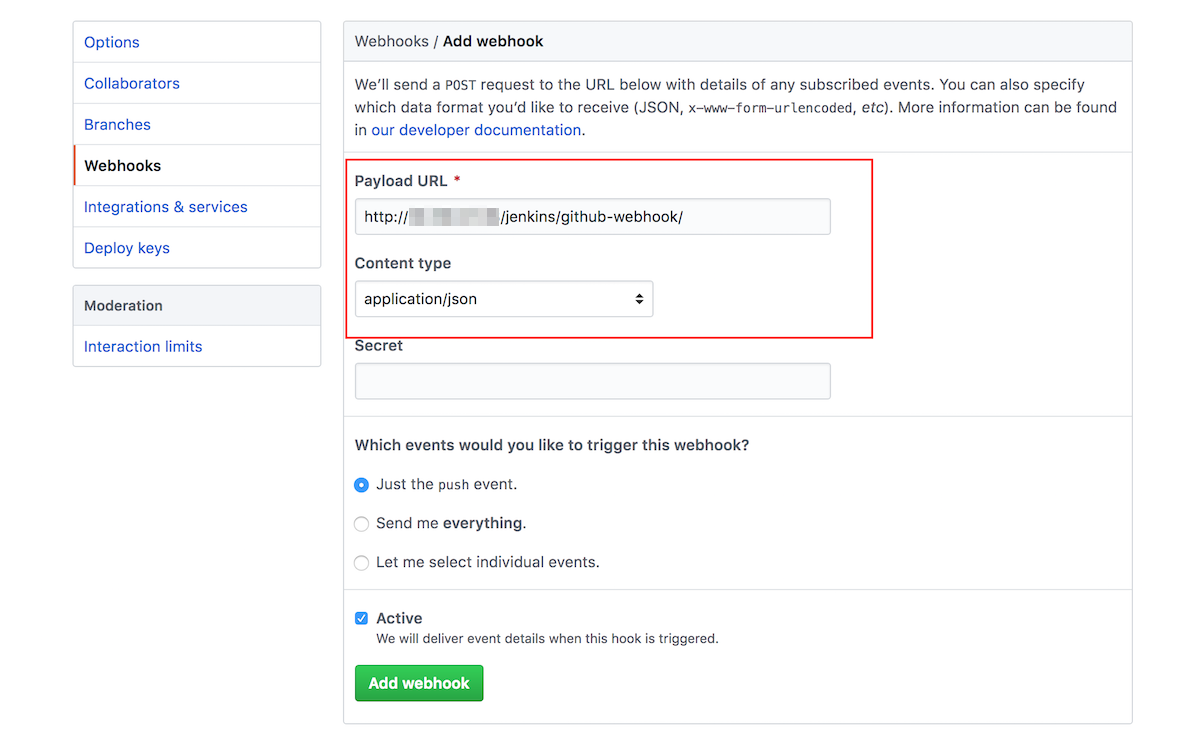
Head back to the project page and click the “Build now” link. Jenkins should clone your GitHub repository and build it with Maven. Once the build is complete, select the “Console Output” menu item (as done previously in [Step 2](https://docs.bitnami.com/oci/how-to/create-ci-pipeline-jenkins-oracle/#step-2-create-and-build-a-simple-project)) to access detailed build information.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-17.png)

### **Step 4: Create A Continuous Integration Pipeline For A GitHub Project**

The next step is to create a Continuous Integration (CI) pipeline that connects your GitHub repository with Jenkins. Once this is in place, any changes to your GitHub repository will trigger a new build in Jenkins. Any errors resulting from the changes will result in a build failure, giving you an immediate heads-up on potential issues or bugs.

* Navigate to your GitHub repository and click on the “Settings” tab.
* Move in your repository
* From the left side repository, click “Webhooks”. On the resulting screen, click the “Add webhook” button.
* On the Webhook page, enter the following information:
  + Payload URL: enter the URL to your Jenkins installation with the additional /github-webhook/ suffix. For example, if your Jenkins installation is available at http://SERVER-IP/jenkins, enter the Jenkins hook URL as http://SERVER-IP/jenkins/github-webhook/.
  + Content type: select the “application/json” option.
  + Which events would you like to trigger this webhook?: select the “Just the push event”.

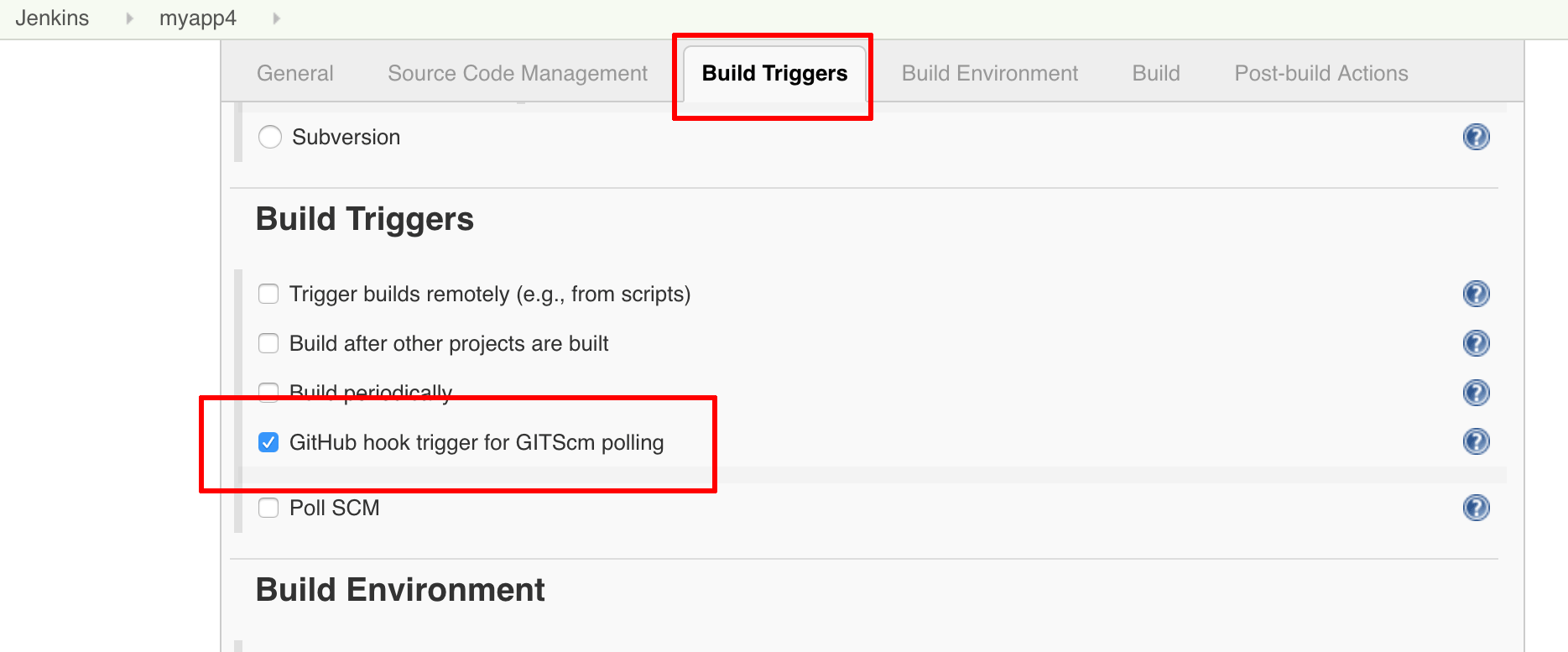
[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/add-github-webhook.png)

* Click the “Add webhook” button to create the new webhook.

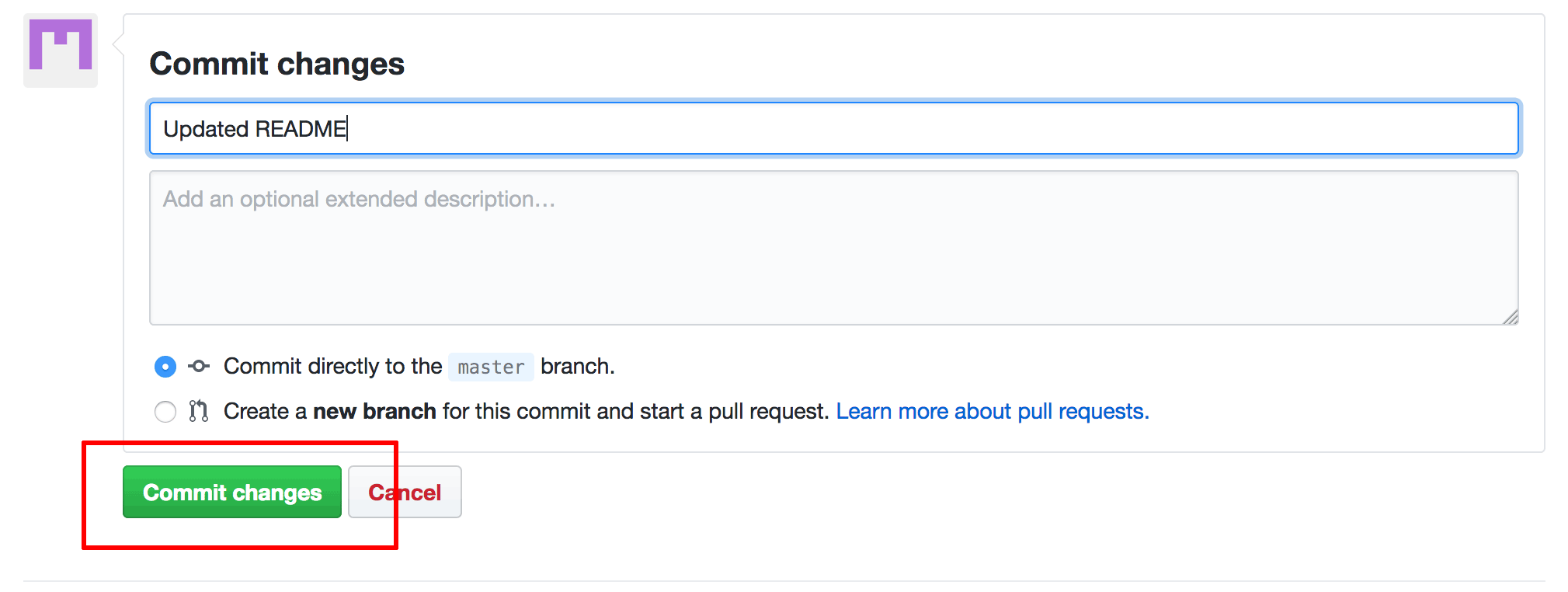
NOTE: It may take a few minutes for the integration to begin working.

To set up this pipeline, you need to enable the GitHub hook trigger in Jenkins for your project. Then, you will configure GitHub to automatically notify Jenkins on any change to the repository.

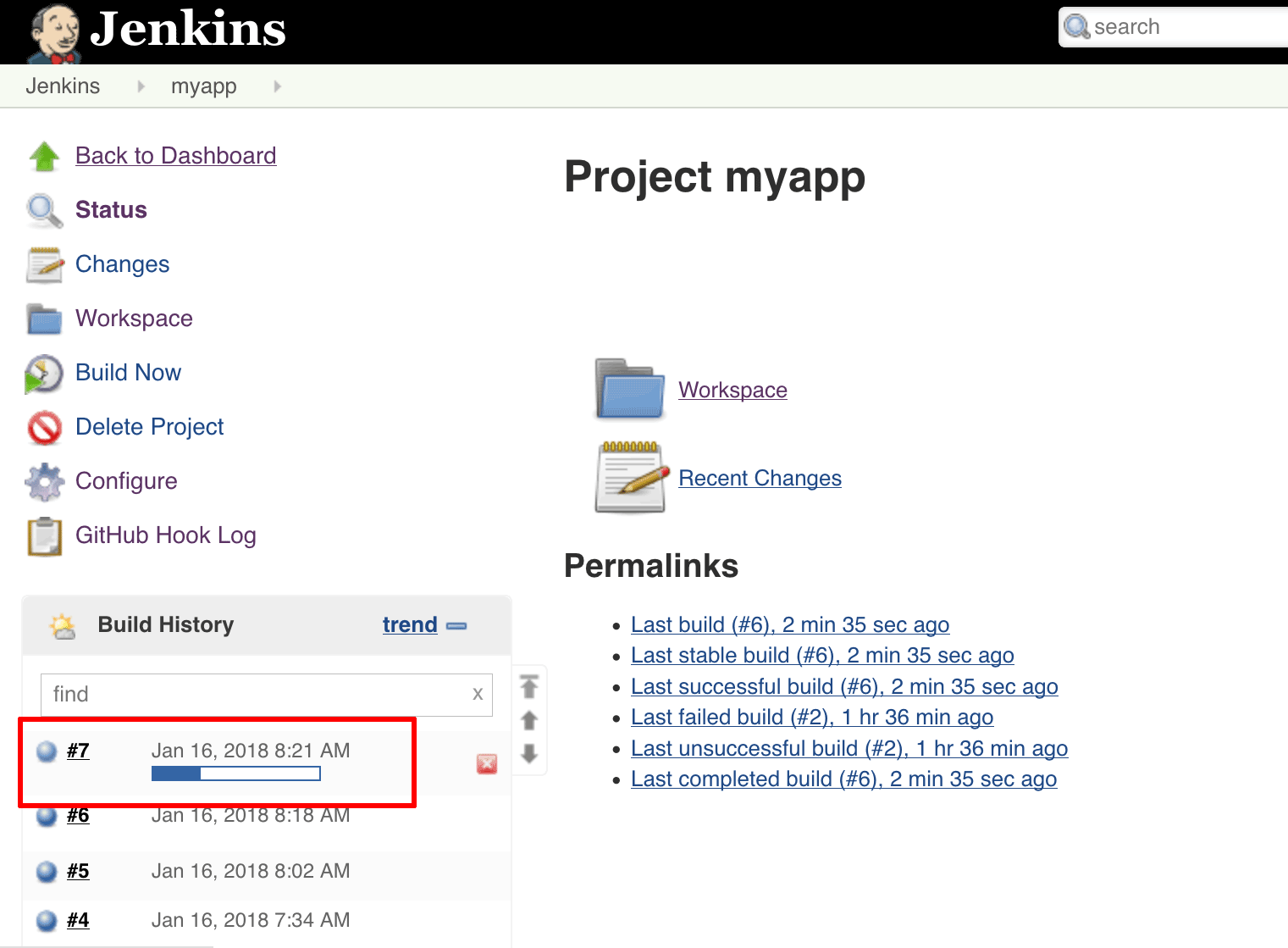
* From the Jenkins dashboard, select the project you created in [Step 3](https://docs.bitnami.com/oci/how-to/create-ci-pipeline-jenkins-oracle/#step-3-build-a-github-project). It redirects you to the project configuration page.
* From the left side menu, click the “Configure” option. On the next screen, scroll down until the “Build Triggers” section. Enable the “Github hook trigger for GITScm polling” option and save the changes.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-18.png)

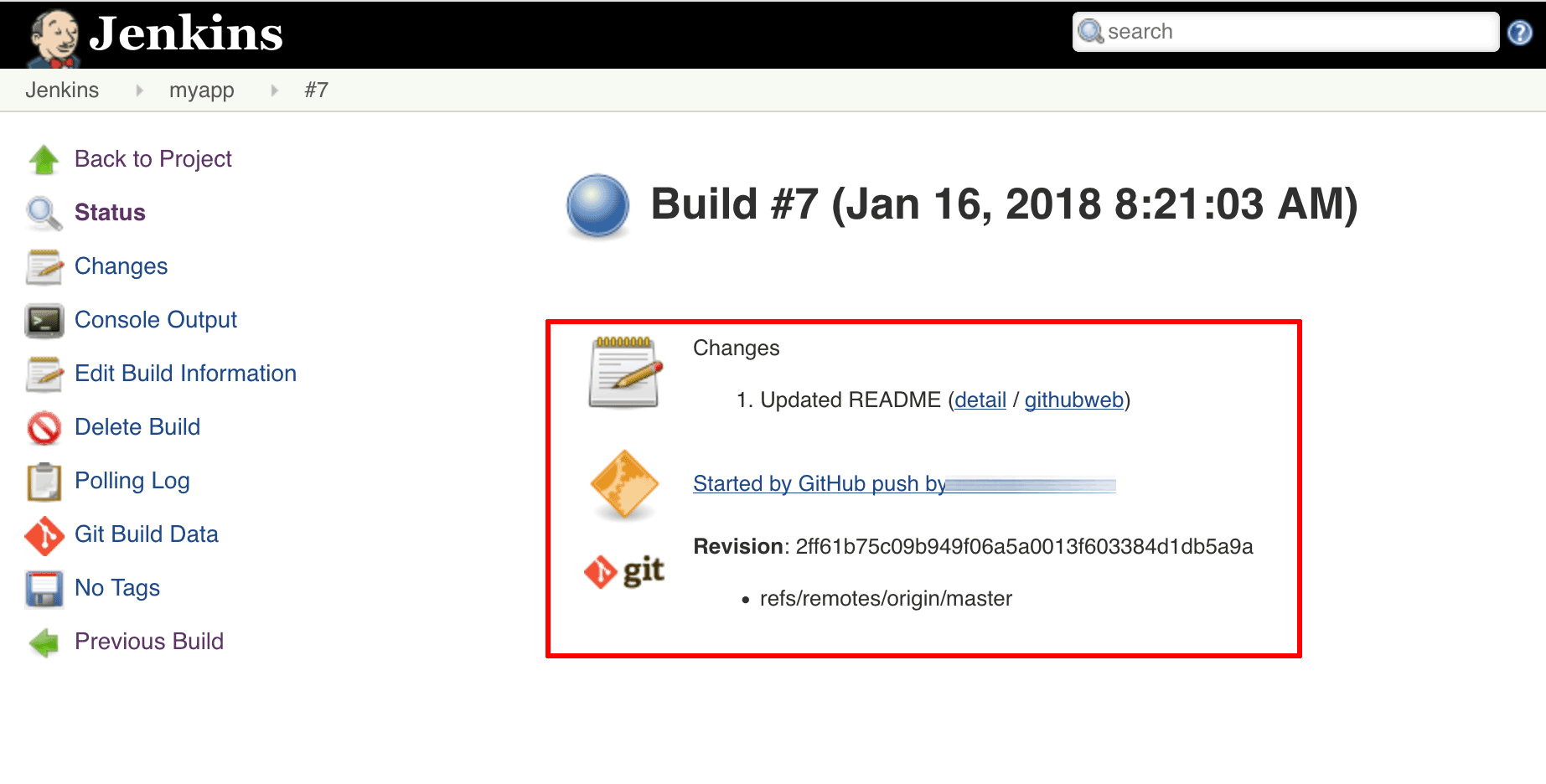
Your pipeline is now configured, and all that remains is to test it. To do this, make a change to the repository - for example, by adding a line to the README file and committing the change.

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-22.png)

Your change should trigger a new build in Jenkins, as shown below:

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-23.png)

Clicking the build number will show you the details of the commit that triggered the build:

[](https://docs.bitnami.com/images/img/how_to_guides/create-ci-pipeline-jenkins-oracle/create-ci-pipeline-jenkins-oracle-24.png)

Congratulations! Your CI pipeline is now complete, and any subsequent changes to the repository will be automatically built by Jenkins.