# **Practical Exercises: Building a Jenkins CI/CD pipeline with API Controller**

Training Objective

Learn how to build a Jenkins CI/CD pipeline for Dev first approach using WSO2 API Controller.

Business Scenario

PizzaShack wants to implement a CI/CD pipeline for their deployment using Jenkins

High-Level Steps

* Prerequisites
* Building Pipeline

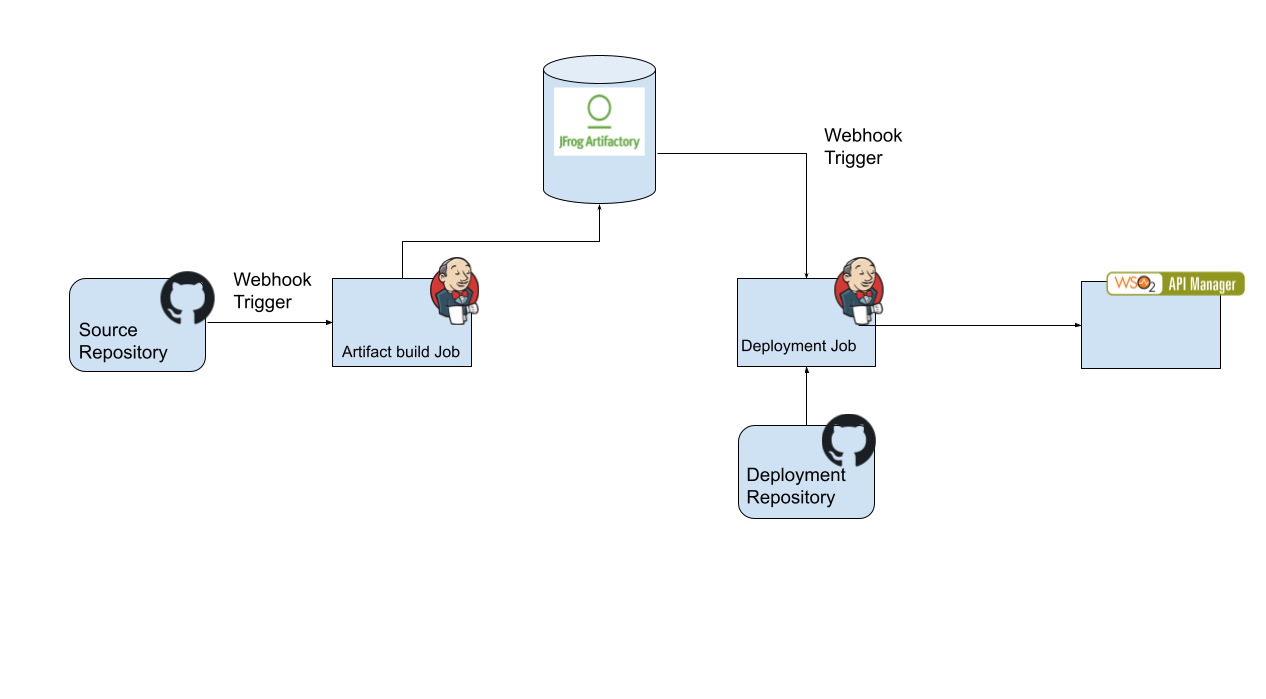
1. Write an Open API specification.
2. Initialize an API Project.
3. Update API data in the API Project.
4. Define environment specific data.
5. Commit the API Project.
6. Trigger Jenkins Pipeline.

Detailed Instructions

# **Building a CI/CD Pipeline for APIs Using WSO2 API Controller**

In the modern digital-era, many organizations adopt digital transformation technologies to keep up with the changes. APIs are one of the basic building blocks used in a digitally-driven organization. When the number of APIs managed by the organization grows, they need to have an automated process to handle the rapid API development process. Having a proper continuous integration and continuous deployment (CI/CD) process would give an added advantage to your organization.

In this section, you will learn how to build an automated process using the WSO2 API Controller (apictl) with WSO2 API Manager (WSO2 API-M). You will see how a solution is built using a version control system (Github), artifact repository(JFrog Artifactory) and CI/CD Tool (Jenkins).

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/cicd-jenkins-arch.png)

## **Setting up the environment**

### **Step 1 - Setup Jenkins**

Before you begin, you need to set up the Jenkins server on a remote machine which has public access. To use webhooks in Github and Artifactory repository to trigger Jenkins jobs you need a publicly accessible Jenkins server url. If you are trying this on a local machine, you may have to use a tool such as [socketxp](https://www.socketxp.com/" \t "_blank) to create a tunnel.

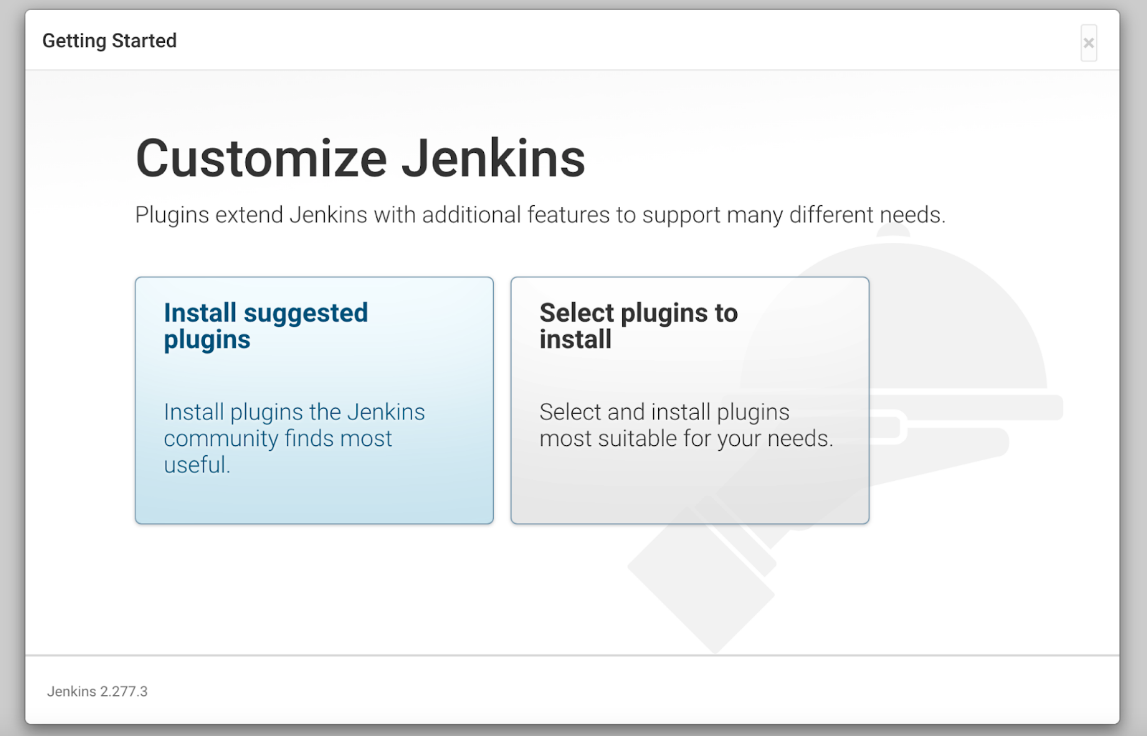
In this section, the instructions are tested out on an ubuntu server setup on <https://cloud.google.com/compute> .

Log in to the server and install following.

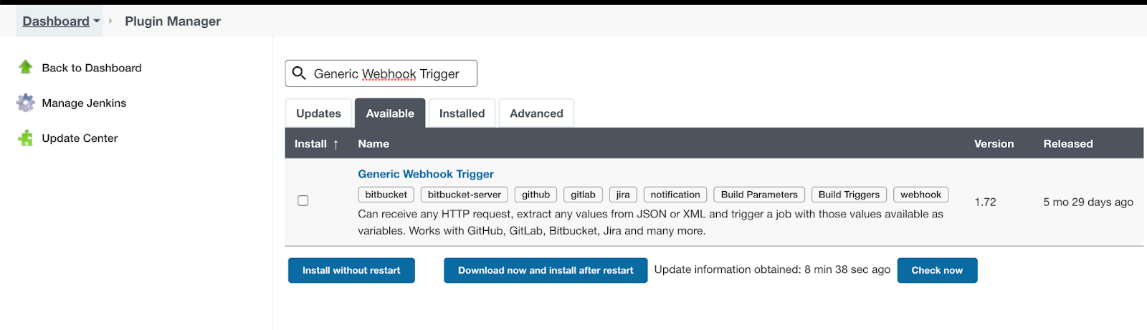
1. Install jq using the command given below.

sudo apt-**get** install jq

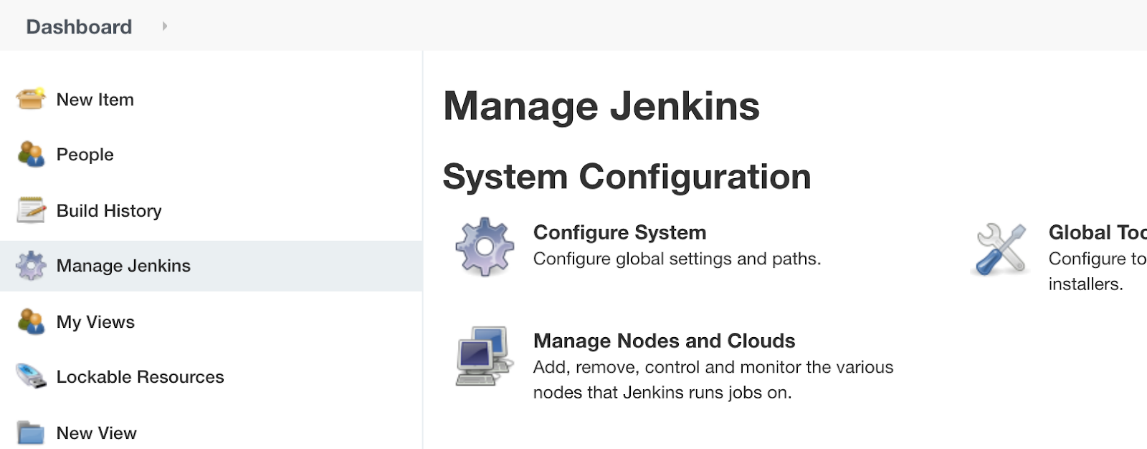
1. Install the [apictl tool](https://apim.docs.wso2.com/en/latest/install-and-setup/setup/api-controller/getting-started-with-wso2-api-controller/" \l "download-and-initialize-the-apictl). Extract to a location henceforth referred as CTL\_HOME.
2. Install java.
3. Setup the Jenkins server following the instructions given [here](https://www.jenkins.io/doc/book/installing/linux/). You can choose to install the suggested default plugins. This will install the commonly-used plugins.

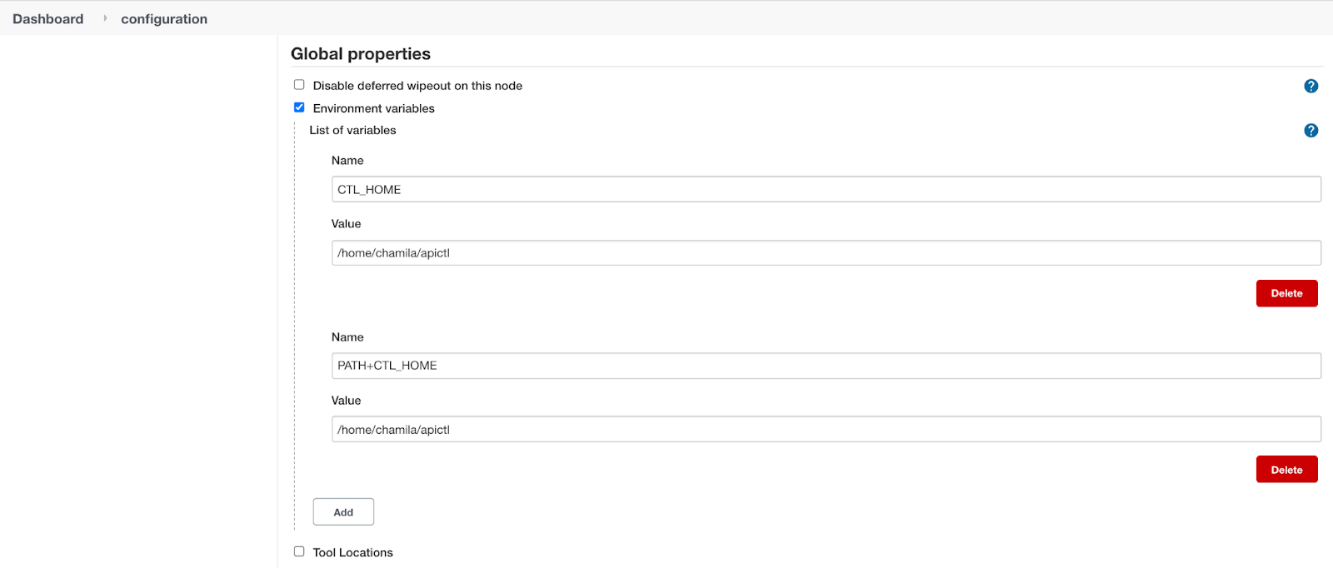
[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/customize-jenkins.png)

1. Register a webhook in the JForg Artifact repository to notify changes. To capture these events and trigger a Jenkins job, install a generic-webhook-trigger plugin.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/webhook-trigger-plugin.png)

1. Configure apictl home as an environment variable. Go to the **Manage Jenkins** section and select **Configure System**.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/configure-system.png)

Under the **Global properties** select **Environment variables** and set the APICTL home as below and click **Save**. [](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/environment-variables.png)

Now the Jenkins server is set up.

### **Step 2 - Create Github repositories**

In this section, two repositories are created to illustrate this scenario.

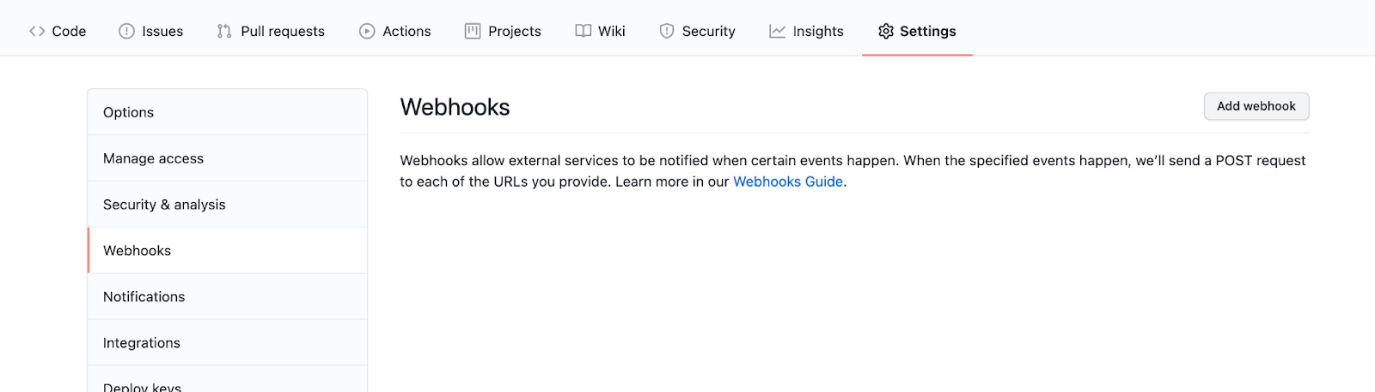
1. Source Repository -<https://github.com/chamilaadhi/poc-cicd-source-repo>
2. Deployment Repository - <https://github.com/chamilaadhi/poc-cicd-deployment-repo>

Source repository contains the files and metadata related to the API. Deployment repository contains the configurations related to each environment this APIs is getting deployed.

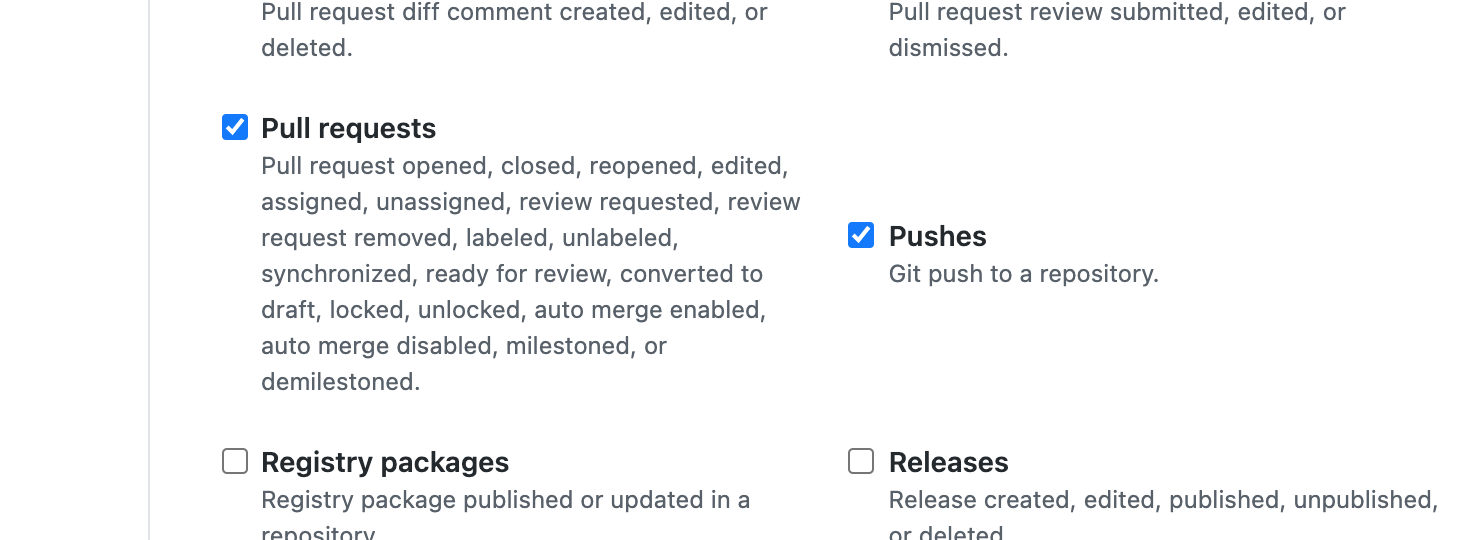
#### Step 2.1 - Setup source repository

This section shows how to trigger a Jenkins job when a change is made to the repository. Lets use [Github webhooks](https://docs.github.com/en/developers/webhooks-and-events/webhooks" \t "_blank) to trigger a Jenkins job.

1. Go to the **source** repository and under the **Settings**, select **Webhooks** and add a webhook to the jenkins server.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/select-webhooks.png)

1. Use your jenkins server URL (e.g., http://&lt;public\_ip>:8080/github-webhook/) as the URL and **Content type** as application/json.
2. Under the **Which events would you like to trigger this webhook?** section, select **Let me select individual events**, and select **pull request** and **pushes**.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/jenkins-server-url.png)

Source repository is now configured.

Now let’s initialize the source repository. This will be done from the developer’s local machine.

**Note**

If you haven’t setup API Controller, set it up using the [apictl tool](https://apim.docs.wso2.com/en/latest/install-and-setup/setup/api-controller/getting-started-with-wso2-api-controller/" \l "download-and-initialize-the-apictl)

1. Clone the source repository. Navigate to the repository folder.

$ git clone https://github.com/chamilaadhi/poc-cicd-source-repo.git

$ cd poc-cicd-source-repo

1. Execute the following command.

$ apictl vcs init

This will create a vcs.yaml file in the repository. Commit it to the source repository.

Now the source repository is configured.

#### Step 2.2 - Setup deployment repository

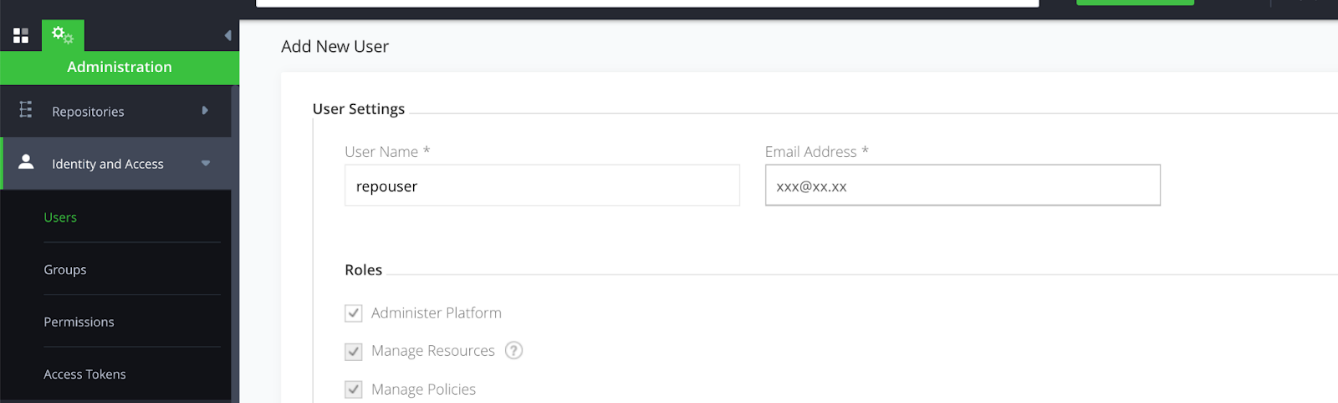
Deployment repository will be used to gather configurations for each deployment. To separate out the configuration, we will use git branches. Create a branch named ‘dev’ to keep the configuration related to the dev environment.

### **Step 3 - Setup JFrog Artifactory**

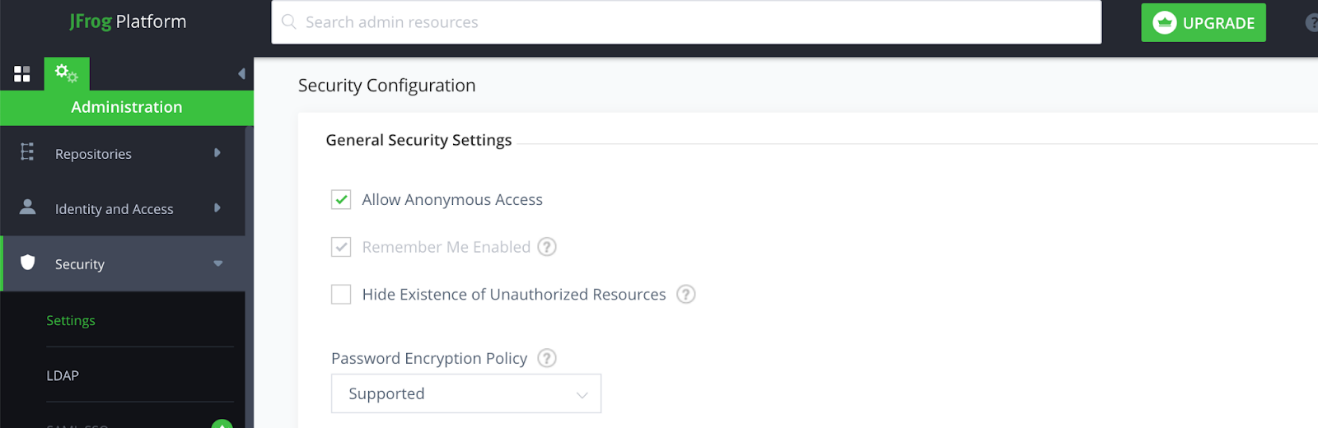
**Before you begin**

Create a free cloud repository using <https://jfrog.com/start-free/#saas>. After the registration is complete, Login to https://&lt;name>.jfrog.io/ui/

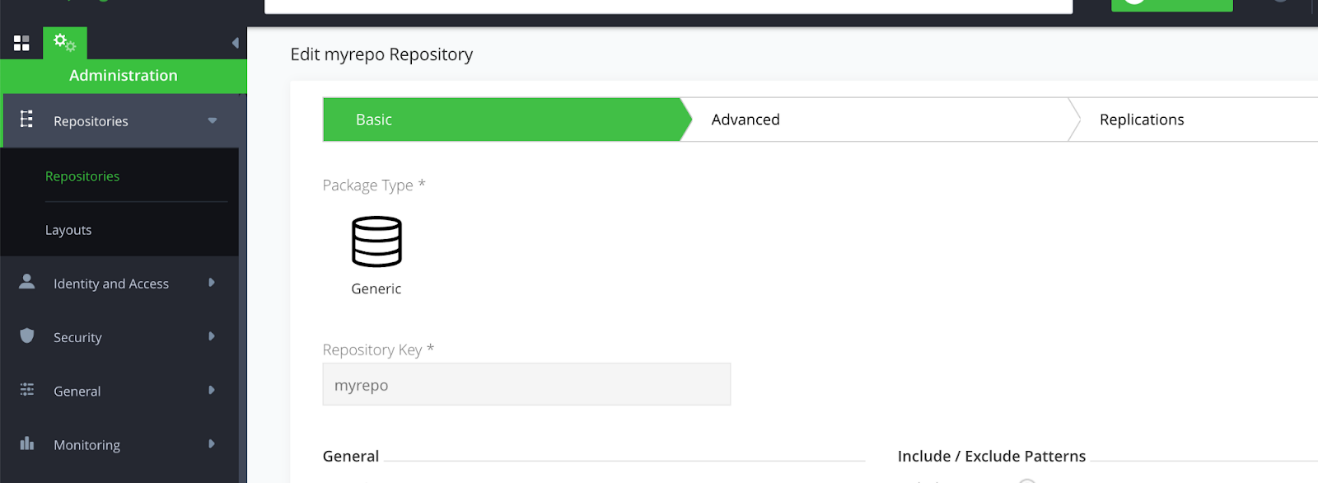
1. Create a user to upload artifacts. Login using admin credentials and go to **Identity and Access** → **Users** and create a user. All roles are selected in this scenario.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/create-a-user.png)

1. Make the repo accessible public. Go to **Security → Settings** and select **Allow Anonymous Access**

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/repo-public-access.png)

1. Create a repository to upload the artifacts. Go to **Repositories** and select **Add Repositories**. Select **Generic** type and provide a name (e.g., myrepo).

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/create-repository.png)

After saving, You should be able to access https://<your\_org>.jfrog.io/artifactory/<repo\_name>/.

Try uploading a sample resource. An example is given below.

$ touch resource.txt

$ curl -u repouser:<password> -X PUT https://xxxx.jfrog.io/artifactory/myrepo/ -T resource.txt

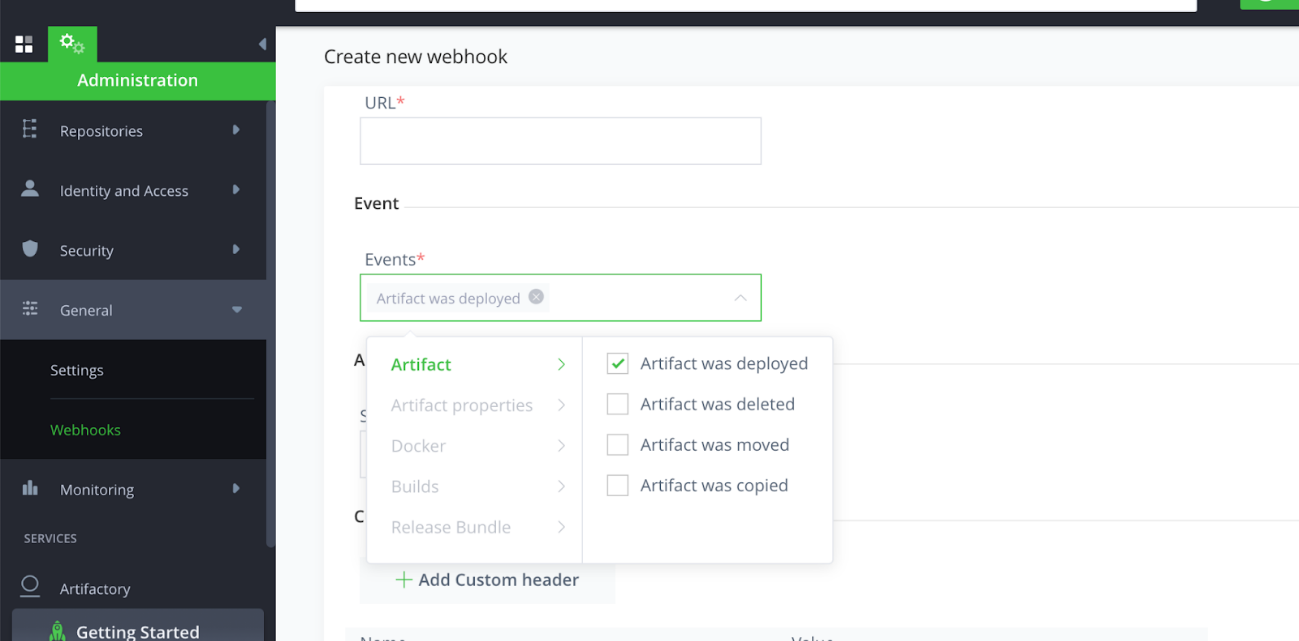
Access the repository URL to see the resource.

1. Configure a webhook to triggerb the Jenkins job.

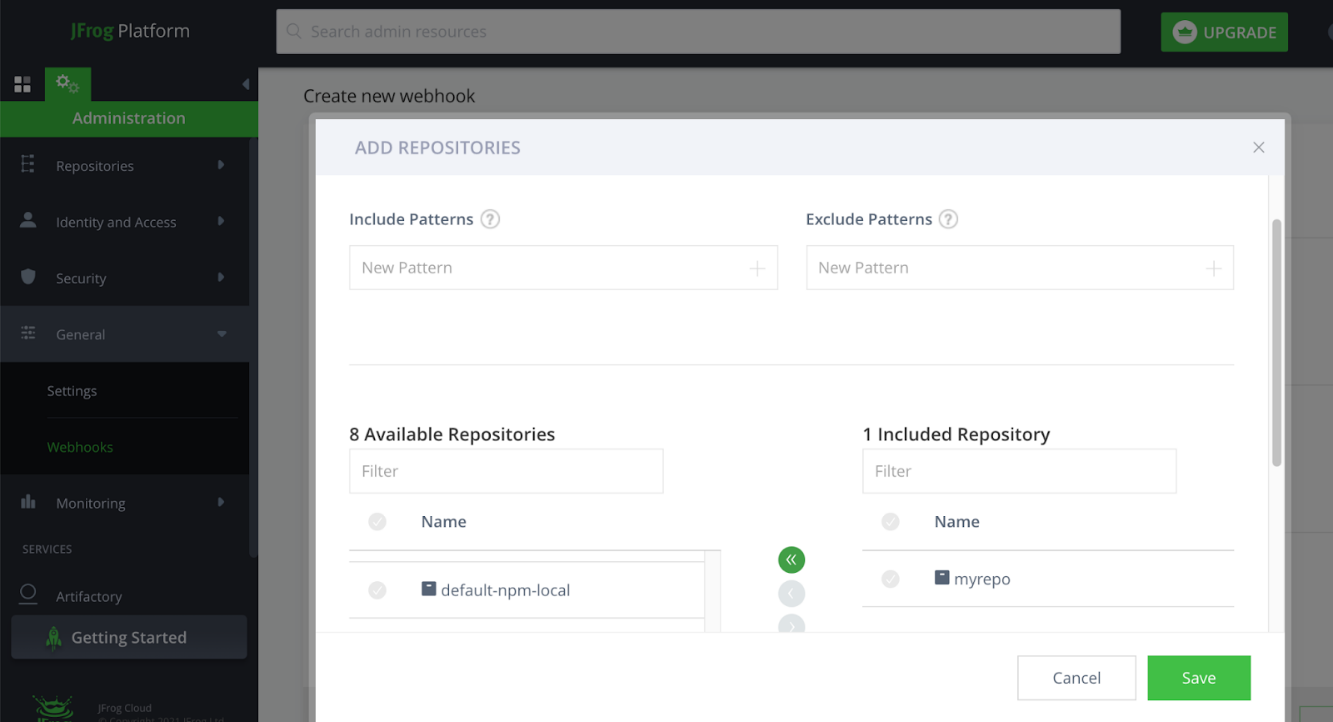
Create a webhook to trigger the Jenkins job when a new resource is uploaded to the repository. Go to **General** → **Webhooks** and create a new one.

URL - http://<jenkins\_host>:8080/generic-webhook-trigger/invoke?token=123\*\*

Add the **Event set** as **Artifact was deployed**.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/set-event.png)

Select the repository from the new window that appears.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/select-repository.png)

### **Step 4 - Setup API-M instances**

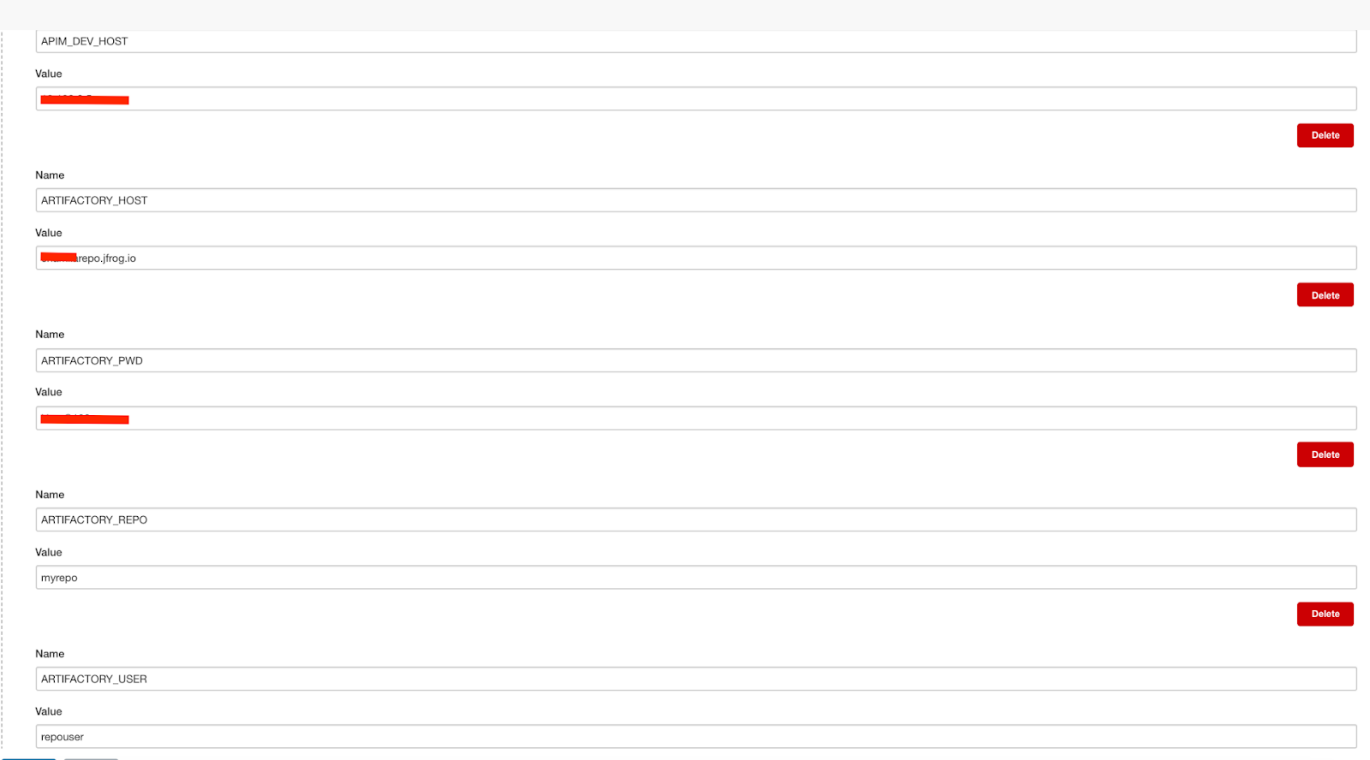
Download the WSO2 API Manager 4.2.0 [here](https://apim.docs.wso2.com/en/latest/install-and-setup/setup/api-controller/getting-started-with-wso2-api-controller/#download-and-initialize-the-apictl) and start the instance. Set this in a separate instance as the dev instance.

### **Step 5 - Configure Jenkins Jobs**

#### Step 5.1 - Setup Global variables

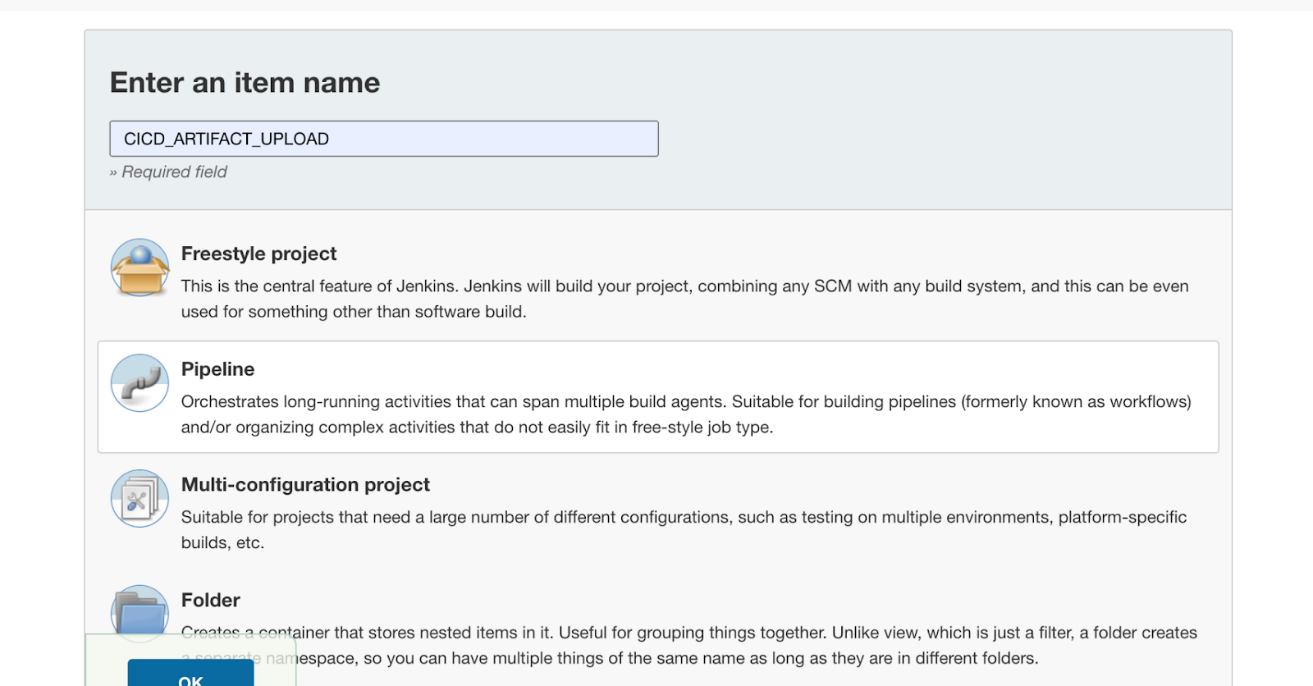
As shown in the **Setup Jenkins** Section, Go to **Manage Jenkins** section and select **Configure System**. Under the **Global properties** select **Environment variables** and set following variables with your environment related details.

|  |  |
| --- | --- |
| Variable | Description |
| APIM\_DEV\_HOST | IP or host name of the dev instance (ex 10.100.5.96 ) |
| APIM\_PROD\_HOST | IP or host name of the prod instance (ex 10.100.5.97 ) |
| ARTIFACTORY\_HOST | Artifactory host name (ex: testrepo.jfrog.io) |
| ARTIFACTORY\_PWD | Artifact uploading user’s password |
| ARTIFACTORY\_REPO | Repo name.(ex: myrepo) |
| ARTIFACTORY\_USER | Artifact uploading user’s username |

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/global-variables.png)

#### Step 5.2 - Setup Artifact build and upload job

1. Create a jenkins **Pipeline** project.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/create-pipeline.png)

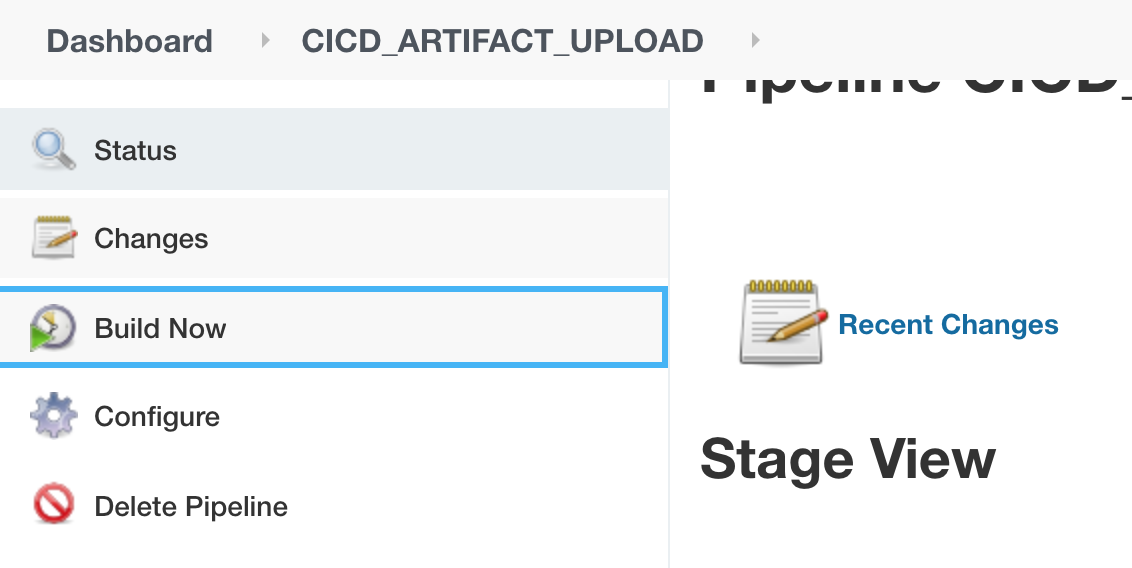
1. Use [this](https://gist.github.com/chamilaadhi/def68ba36cedec6b901731f32bbad532) script for the pipeline.

**Note**

Jenkins server default workspace URL is /var/lib/jenkins/workspace/. If you have installed the Jenkins server in a different location, change this path in the script accordingly.

1. Under the **Build Triggers** set the GitHub hook trigger for **GITScm polling**.
2. Save the configuration. Now the Jenkins job is configured to listen to any change in the source repository and upload any new update to the artifact repository.

To set up, Execute a build job. Note that this will fail.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/build-now.png)

#### Step 5.3 - Setup Artifact deployment Job

The artifact deployment job will listen to any new updates in the Artifactory repository and deploy the artifact with the configurations in the Deployment git repository to the dev API-M environment.

1. Create a Jenkins pipeline using [this](https://gist.github.com/chamilaadhi/81241bf2e9c46b720ef61fb516e00249) script .
2. Save the script and execute a build to setup. Note that this will fail.

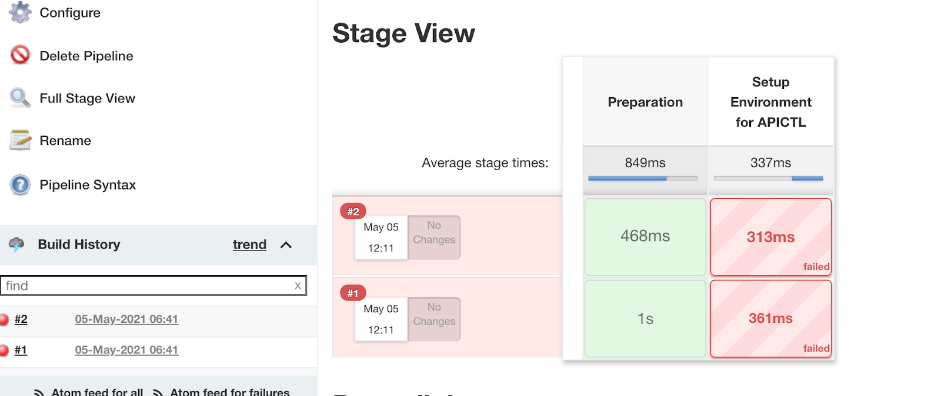
[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/build-dev-now.png)

1. To test whether the webhook between Artifactory and the Jenkins job is working, you could execute the same upload test we did under Setup JFrog Artifactory section

$ touch resource.txt

$ curl -u repouser:<password> -X PUT https://xxxxx.jfrog.io/artifactory/myrepo/ -T resource.txt

If the configuration is working, this will trigger a new build.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/trigger-new-build.png)

## **Test the scenario**

To test this setup, use the OpenAPI definition based method to create the APIs.

1. Create API using an OAS Definition. This will create a project with name **PetstoreAPI**. A sample command is given below.

$apictl init PetstoreAPI --oas https://raw.githubusercontent.com/OAI/OpenAPI-Specification/main/examples/v2.0/yaml/petstore.yaml

1. Generate deployment resource using the command given below.

$apictl gen deployment-dir --source /Path/To/PetstoreAPI --destination <deployment\_repo\_path>/poc-cicd-deployment-repo

You will see the resources created with the following structure.

├── **DeploymentArtifacts\_SwaggerPetstore-1**.0.0

│ ├── **api\_meta**.yaml

│ ├── **certificates**

│ └── **params**.yaml

1. Open the params.yaml file and update the content with the code given below.

environments:

- name: dev

configs:

endpoints:

production:

url: 'https://petstore.swagger.io/v2/'

deploymentEnvironments:

- displayOnDevportal: true

deploymentEnvironment: Default

deploymentVhost : localhost

The environment name is set as dev. This configuration will be used for the dev environment.

1. Commit this to the **dev** branch in the deployment repository.
2. Copy the **PetstoreAPI** folder to the source repository git location.
3. Create a file named **meta.yaml** inside the **PetstoreAPI** folder and add a version similar to following.

**version**: 1.0.0

This version is used to deploy the bundle created using the **PetstoreAPI** API in the Artifactory repository.

1. Following is the content for the PetstoreAPI. Commit all the files to the git repository

├── **PetstoreAPI**

│ ├── **Definitions**

│ │ └── **swagger**.yaml

│ ├── **README**.md

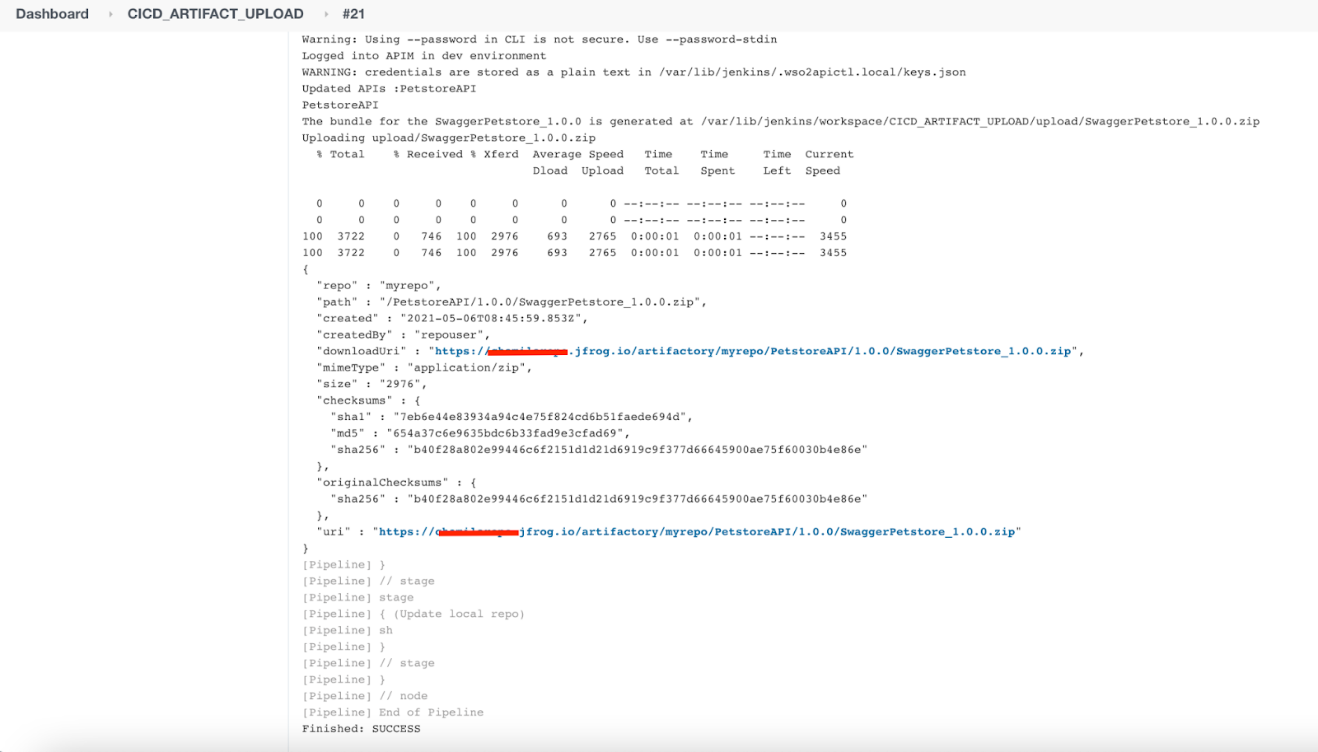
│ ├── **api**.yaml

│ ├── **api\_meta**.yaml

│ ├── **deployment\_environments**.yaml

│ └── **meta**.yaml

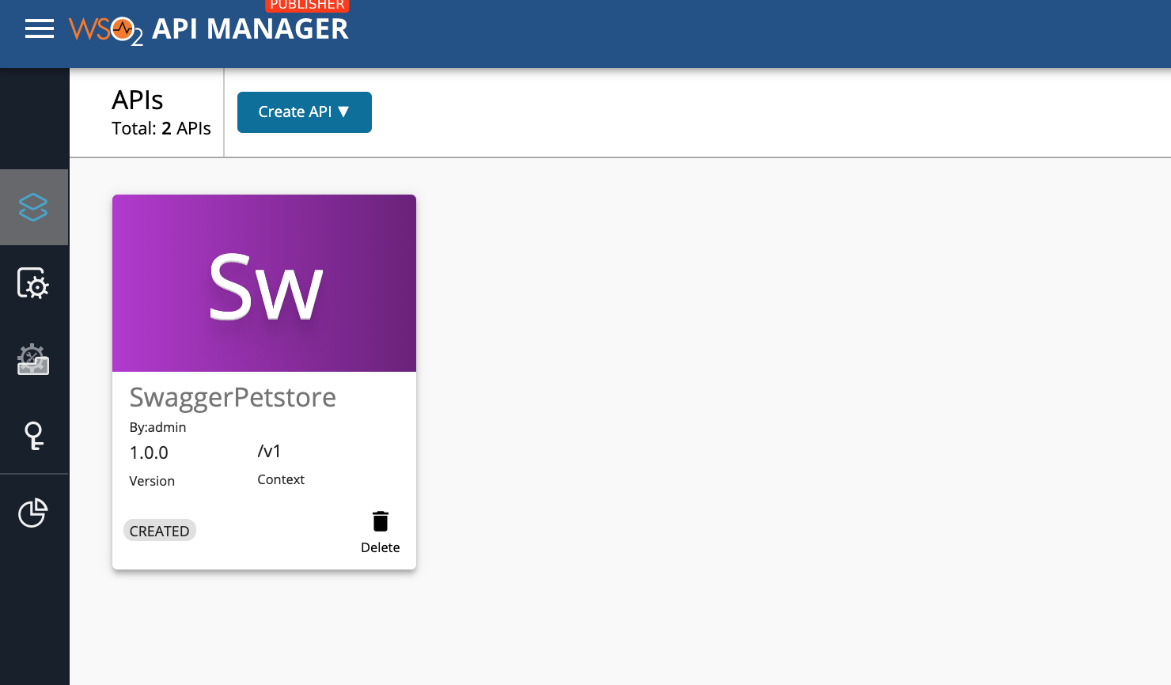
When you commit to the source repository, the CICD\_ARTIFACT\_UPLOAD will get triggered first and upload the built component to the Artifactory. This is displayed in the console log of the Jenkins job.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/console-log-jenkins.png)

Once the bundle is uploaded to the Artifactory repository, it will trigger the deployment Jenkins job. The bundle getting deployed in API-M is displayed in the logs.

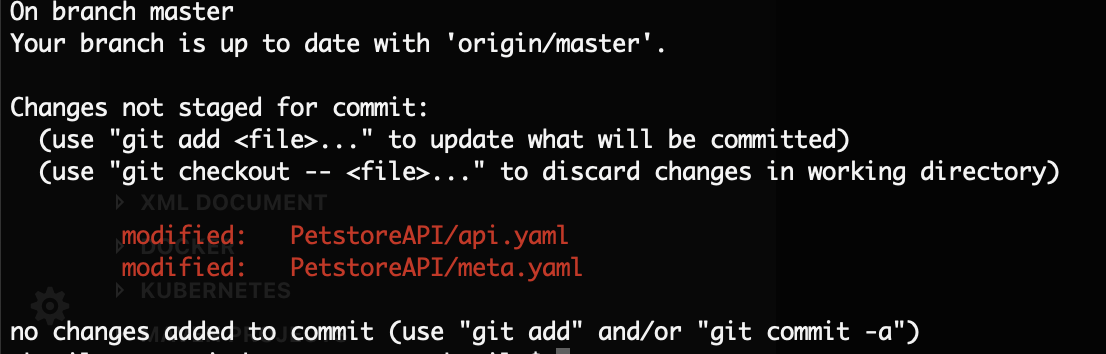
[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/apim-logs.png)

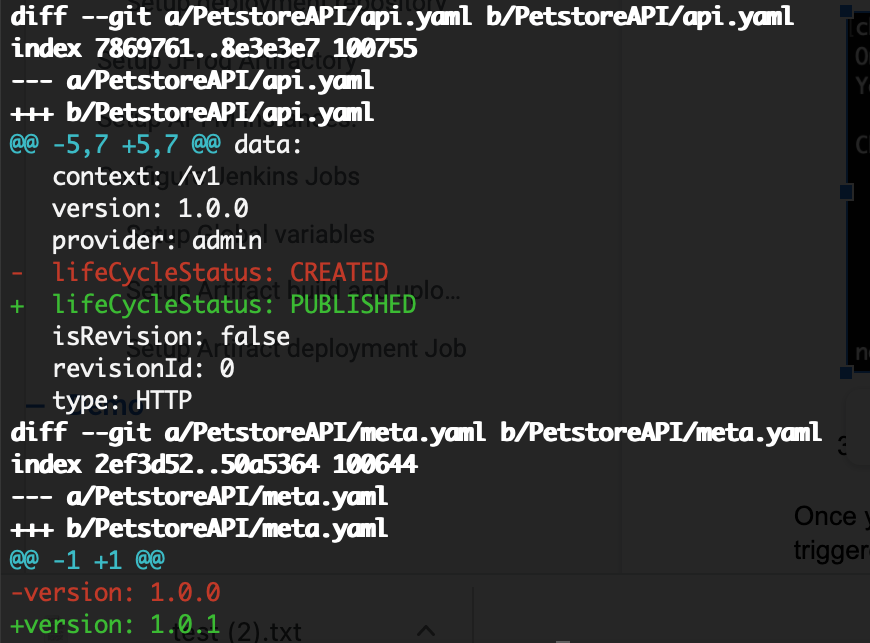
Log in to the API Manager instance and you should see the API is created.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/created-api.png)

Let's update the API and see the changes. The API is in **CREATED** state. Let’s publish this API by following the steps given below.

1. Open the **PetstoreAPI/api.yaml** file and change the **lifeCycleStatus** to **PUBLISHED**
2. Open **PetstoreAPI/meta.yaml** file and change the version to some new value (let’s say 1.0.1) .

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/create-file-git.png)

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/api-state.png)

1. Commit both files.

Once you commit the changes to the source repository, you would see the jenkins jobs getting triggered. Log in to the Publisher Portal and check the changes.

If you go to the artifactory repository and check the **PetstoreAPI** you will see two versions in the repository.

[](https://apim.docs.wso2.com/en/latest/assets/img/learn/api-controller/two-versions.png)