Pipeline Example: Maven Build

Overview

This quickstart demonstrates a simple pipeline that builds and publishes a Maven package. An example Pipelines DSL is used to show how to use integrations, resources, and a combination of native steps to build an application and publish it to Artifactory.

This tutorial walks you through the following steps to run this application using JFrog Pipelines:

- Create a GitHub Integration and Artifactory Integration.
- Add a Pipeline Source.
- Build Maven components using the MvnBuild native step.
- Publish build info using the PublishBuildInfo native step.

Page Contents

A successful run of the pipeline in this Quickstart looks like this:

Before you Begin

Before trying this example, ensure that you have:

- A GitHub account. This is required for forking the sample repository.
- A JFrog Platform account, or self-hosted JFrog Pipelines.
- At least one node pool. This is the set of nodes that all pipeline steps will execute in. For more information, see Managing Pipelines Node Pools.

> If you have a Cloud account, a node pool will already be available as part of your subscription.

Running This Example

Perform the steps below to build your Maven artifact:

1. ### Fork the repository

   This Pipelines sample is available in the jfrog-pipelines-maven-sample repository in the JFrog GitHub account. The configuration is included in the YAML files at the root of the repository:
   - `pipelines.yml`, which contains the declarations for all the resources and steps required to run the pipeline. This configuration is written in template format, so you will not need to change anything in this file.
   - values.yml, which contains custom values that will be populated into the template to create your pipeline

   For a full breakup of all the resources, pipelines, and steps used in the yml file, see the pipeline definition section below. Fork this repository to your account or organization. This is important since you need admin access to repositories that are used as in your pipelines, to enable us to add webhooks to these repositories and listen for change events.

2. ### Sign in to Artifactory

   Sign in to JFrog Platform with your Artifactory credentials.

3. ### Create a local Maven repository

Create a local Maven repository and write down the repository name, since you will need to use it in your pipeline configuration.

4. ## Add Integrations

a. Go to **Administration | Pipelines | Integrations** to add two integrations:

- **GitHub Integration**: This integration is used to add the Pipeline source, as well as the GitRepo resource.
- **Artifactory Integration**: This integration is used to authenticate with Artifactory to download Maven depedencies from Artifactory, and to pack and upload the built package to Artifactory.

b. Write down the names of both GitHub and Artifactory integrations as these are required for the next step. Ensure that the names are unique and easy to remember.

5. ## Update pipeline definitions

Since your pipelines.yml config file is templatized, as shown in the table below, update the values.yml in your forked repository:

Tag	Description	Example
`gitProvider`	Provide the name of the Github integration you added in Step 4.	`gitProvider: my_github`
`repoPath`	Provide the path to your fork of this repository.	`repoPath: myuser/jfrog-pipelines-maven-sample`
`artifactory`	Provide the name of the Artifactory integration you added in the previous Step 4.	`artifactory: demoArt`
`deployerRepo`	Provide the name of the local Maven repository in Artifactory you created in Step 3.	`deployerRepo: maven-local`

And that's it. Your configuration is ready to go!

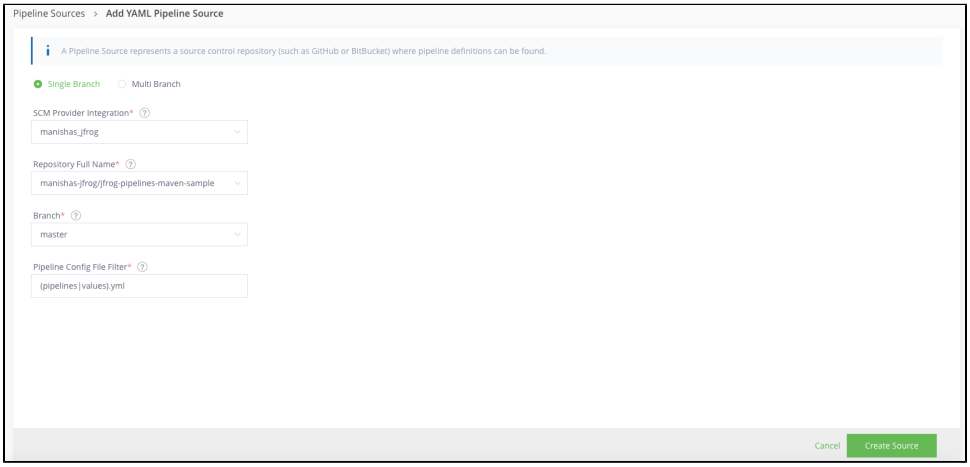
> ⚠ **All pipeline definitions are global across JFrog Pipelines within a Project.** The names of your pipelines and resources need to be unique within the Project in JFrog Pipelines.

6. ## Add Pipeline Sources

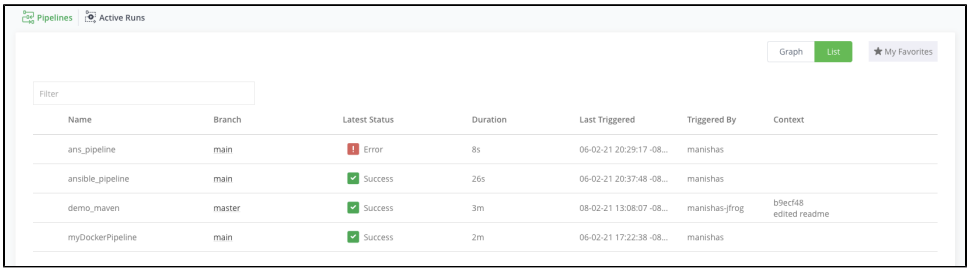
The Pipeline Source represents the git repository where our pipelines definition files are stored. A pipeline source connects to the repository through an integration, which we added in Step 4.

In your left navigation bar, go to **Administration | Pipelines | Pipeline Sources**. Click **Add a Pipeline Source** and then choose **From YAML** . Follow instructions to add a Pipeline Source. This automatically adds your configuration to the platform and pipelines are created based on your YAML.

An example for adding a pipeline source is shown below. Ensure that **Repository Full Name** points to your forked repository and **Pipeline Config File Filter** is entered correctly as (pipelines|values).yml so that both your config files are included.



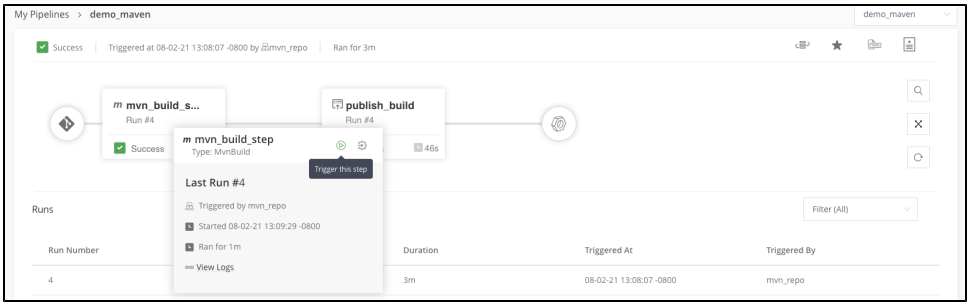
After your pipeline source syncs successfully, navigate to **Pipelines | My Pipelines** in the left navbar to view the newly added pipeline. In this example, `demo_maven` is the names of your pipeline.



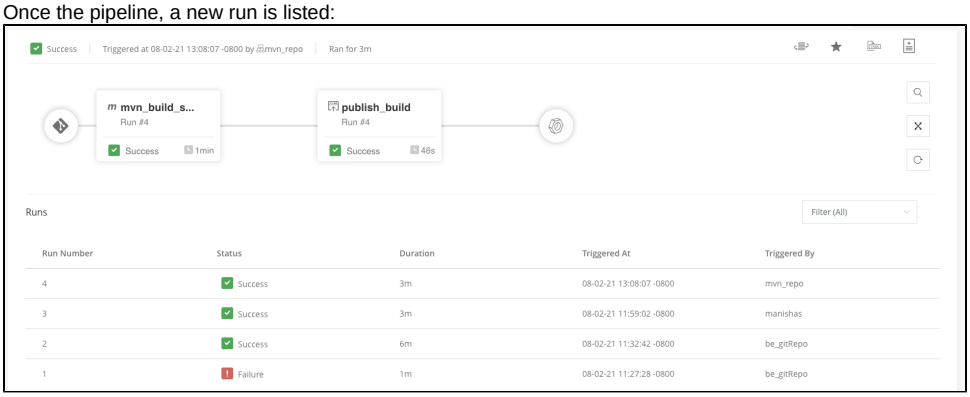
Click the name of the pipeline. This renders a real-time, interactive, diagram of the pipeline and the results of its most current run.

7. ## Execute the Pipeline

You can trigger the pipeline by committing a change to your repository, or by manually triggering it through the UI. Multiple steps can execute in parallel if the node pool has multiple build nodes available.



Once the pipeline, a new run is listed:



How the Pipeline Definition Works

Let us now take a look at the pipeline definition files and what each section means.

The **pipelines.yml** file contains the templatized definition of your pipeline. This consists of the following:

- Resources are entities that contain information that is consumed or generated by pipeline steps. In our example, we use the following resources:
  - A GitRepo resource pointing to the source control repository where your application code is present. You can configure this resource to trigger dependent steps on specific events. For more information, see GitRepo.
  - A BuildInfo resource is a pointer to the Build on Artifactory. This is automatically created by the PublishBuildInfo step. For more information, see BuildInfo.
- Steps are executable units that form your pipeline. In our example, the pipeline consists of the following steps:
  - A MvnBuild native step that builds your Maven project and optionally deploys it to Artifactory. This step is a pre-packaged step (that is, native step) that is available to be used with simple configuration and without the need for custom scripting. For more information, see MvnBuild.
  - A PublishBuildInfo step is a native step that gathers build metadata and pushes it to Artifactory. Artifactory Builds provide a manifest and include metadata about included modules, dependencies and other environment variables. For more information, see PublishBuildInfo.