Bitbucket Pipelines Artifactory Pipes

Overview

Artifactory brings continuous integration to Atlassian Bitbucket Pipelines through the **JFrog Artifactory** pipes.

The **JFrog Artifactory** pipes for Bitbucket Pipelines supports:

- Resolving your build dependencies from Artifactory.
- Deploying your build artifacts to Artifactory.
- Gaining full traceability of your builds by capturing your build-info from your builds and publishing to Artifactory.

About Pipes

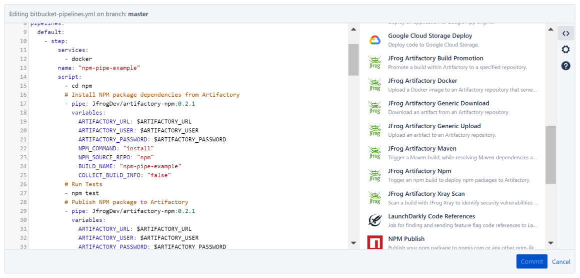
Pipes are Atlassian's way to simplify configuring your Bitbucket Pipeline. They're very useful for frequently performed actions that would otherwise take several lines of script, especially when working with a supporting tool like Artifactory or Xray.

Just like a subroutine or object method in code, you invoke a pipe to perform a specific function according to parameters you provide.

Page Contents

Setup and Use

No special setup is needed to use JFrog Artifactory pipes. They are readable in a public Bitbucket repository. You can select them, along with all other Atlassian-supported pipes, through the Bitbucket online editor, in the rightmost side panel.



Editing bitbucket-pipelines.yml on branch: **master**

```
pipelines:
  default:
    - step:
        services:
          - docker
        name: "npm-pipe-example"
        script:
          - cd npm
          # Install NPM package dependencies from Artifactory
          - pipe: JfrogDev/artifactory-npm:0.2.1
            variables:
              ARTIFACTORY_URL: $ARTIFACTORY_URL
              ARTIFACTORY_USER: $ARTIFACTORY_USER
              ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
              NPM_COMMAND: "install"
              NPM_SOURCE_REPO: "npm"
              BUILD_NAME: "npm-pipe-example"
              COLLECT_BUILD_INFO: "false"
          # Run Tests
          - npm test
          # Publish NPM package to Artifactory
          - pipe: JfrogDev/artifactory-npm:0.2.1
            variables:
              ARTIFACTORY_URL: $ARTIFACTORY_URL
              ARTIFACTORY_USER: $ARTIFACTORY_USER
              ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
```

Google Cloud Storage Deploy
Deploy code to Google Cloud Storage.

JFrog Artifactory Build Promotion
Promote a build within Artifactory to a specified repository.

JFrog Artifactory Docker
Upload a Docker image to an Artifactory repository that serve...

JFrog Artifactory Generic Download
Download an artifact from an Artifactory repository.

JFrog Artifactory Generic Upload
Upload an artifact to an Artifactory repository.

JFrog Artifactory Maven
Trigger a Maven build, while resolving Maven dependencies a...

JFrog Artifactory Npm
Trigger an npm build to deploy npm packages to Artifactory.

JFrog Artifactory Xray Scan
Scan a build with JFrog Xray to identify security vulnerabilities ...

LaunchDarkly Code References
Job for finding and sending feature flag code references to La...

NPM Publish
Publish your npm package to npmjs.com or any other npm-lik

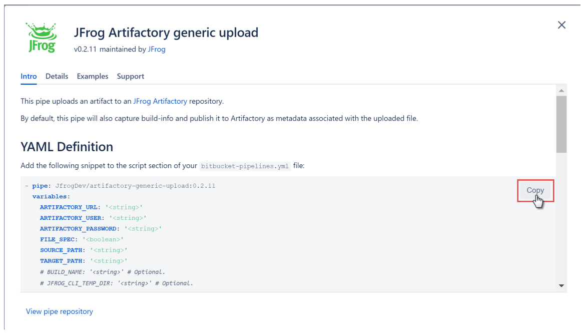
Prerequisites

To use JFrog Artifactory pipes, you will need a licensed installation of Artifactory, and to know its addressable URL, and your credentials for access.

How to use pipes

In the YAML file that configures your Bitbucket Pipeline, you specify a path to the pipe, followed by a few key pieces of information.

Selecting one of the available pipes will present you the information from the pipe's README.md that you will need to put it to use.



JFrog Artifactory generic upload
v0.2.11 maintained by JFrog

This pipe uploads an artifact to an JFrog Artifactory repository.

By default, this pipe will also capture build-info and publish it to Artifactory as metadata associated with the uploaded file.

YAML Definition

Add the following snippet to the script section of your `bitbucket-pipelines.yml` file:

```
- pipe: JfrogDev/artifactory-generic-upload:0.2.11
  variables:
    ARTIFACTORY_URL: '<string>'
    ARTIFACTORY_USER: '<string>'
    ARTIFACTORY_PASSWORD: '<string>'
    FILE_SPEC: '<boolean>'
    SOURCE_PATH: '<string>'
    TARGET_PATH: '<string>'
    # BUILD_NAME: '<string>' # Optional.
    # JFROG_CLI_TEMP_DIR: '<string>' # Optional.
```

To use a pipe in a Bitbucket Pipeline, copy the offered snippet to the Bitbucket YAML file, and replace the placeholder parameter text with the needed information. A best practice is to reference commonly needed parameters, such the base Artifactory URL, and your credentials, as environment settings.

Publishing build-info to Artifactory

Most pipes, including the **Artifactory Generic Upload** and **Artifactory Generic Download** pipes, can be directed to collect build-info and publish it to Artifactory by setting the COLLECT_BUILD_INFO varaible to "true". This is, in fact, the default setting for this variable, so it isn't necessary to specify it to publish build-info to Artifactory.

You can also publish (or prevent publishing) other information as part of the build-info, such as environment variables and Git revision information by setting variables to "true" (or "false"). By default, all information will be included in the build-info.

If you need to prevent build-info from being published to Artifactory, then set COLLECT_BUILD_INFO to "false".

For more information about build-info, see Artifactory Build Integration.

Upload Generic Artifacts to Artifactory

The **Artifactory Generic Upload** pipe uploads your generated build artifact(s) from the local file system to an Artifactory repository.

The task triggers JFrog CLI to perform the upload in the background.

The artifacts to upload may be defined using File Specs. By default, the pipe will also capture build-info and publish it to Artifactory as metadata associated with the uploaded file.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. Whether you will use a File Spec or directly specify the artifact(s) to upload.
4. The path to the artifcact(s) to upload (wild cards may be used), or the path to the File Spec.
5. The target path of the Artifactory repository that will receive the upload.
6. Other pipe options, such as a build name to associate with the artifact.

Example: Upload generic artifacts to Artifactory

```
# Upload artifacts to Artifactory
- pipe: JfrogDev/artifactory-generic-upload:0.2.11
  variables:
    ARTIFACTORY_URL: $ARTIFACTORY_URL
    ARTIFACTORY_USER: $ARTIFACTORY_USER
    ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
    FILE_SPEC: "false"
    SOURCE_PATH: "generic/*.zip"
    TARGET_PATH: "generic-local/"
    BUILD_NAME: "generic-pipe-example"
```

See the pipe's README.md for the full list of variables you may specify.

Downloading Generic Artifacts from Artifactory

The **Artifactory Generic Download** pipe downloads your built artifact(s) from an Artifactory repository to the local file system.

The task triggers JFrog CLI to perform the upload in the background.

The artifacts to download may be defined using File Specs. By default, the pipe will also capture build-info published to Artifactory as metadata associated with the downloaded file.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. Whether you will use a File Spec or directly specify the artifact(s) to download.
4. The source path of the Artifactory repository where the file to download is stored. You may use wildcards to specify multiple artifacts.
5. Other pipe options, such as the local file system target path for the downloaded artifact.

Example: Download generic artifacts from Artifactory

```
# Download artifacts from Artifactory
- pipe: JfrogDev/artifactory-generic-download:0.2.11
  variables:
    ARTIFACTORY_URL: $ARTIFACTORY_URL
    ARTIFACTORY_USER: $ARTIFACTORY_USER
    ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
    FILE_SPEC: "false"
    SOURCE_PATH: "generic-local/*.zip"
    TARGET_PATH: "./generic/"
    BUILD_NAME: "generic-pipe-example"
```

See the pipe's README.md for the full list of variables you may specify.

Triggering Builds

You can trigger the following builds.

Triggering Maven Builds

The **Artifactory Maven** pipe triggers a Maven build, while resolving Maven dependencies and deploying Maven packages from and to Artifactory as a Maven repository.

By default, this pipe will also capture build-info and publish it to Artifactory as metadata associated with the built artifacts.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. The Artifactory Maven repository that will be used to resolve snapshots.
4. The Artifactory Maven repository that will be used to resolve releases.
5. Other pipe options, such as a build name to associate with the artifacts.

Example: Build and publish Maven packages to Artifactory

```
# Build and publish Maven packages to Artifactory
- pipe: JfrogDev/artifactory-maven:0.2.9
  variables:
    ARTIFACTORY_URL: $ARTIFACTORY_URL
    ARTIFACTORY_USER: $ARTIFACTORY_USER
    ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
    MAVEN_SNAPSHOT_REPO: "libs-snapshot"
    MAVEN_RELEASE_REPO: "libs-release"
    BUILD_NAME: "maven-pipe-example"
    EXTRA_BAG_ARGS: "../"
```

See the pipe's README.md for the full list of variables you may specify.

Triggering Npm Builds

The **Artifactory Maven** pipe triggers a Maven build, while resolving Maven dependencies and deploying Maven packages from and to Artifactory as a Maven repository.

By default, this pipe will also capture build-info and publish it to Artifactory as metadata associated with the built artifacts.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. The npm command to perform (*install* or *publish*)
4. The Artifactory npm repository that will be used to download and push artifacts.
5. Other pipe options, such as a build name to associate with the artifacts.

Example: Build and publish npm packages to Artifactory

```
# Publish NPM package to Artifactory
- pipe: JfrogDev/artifactory-npm:0.2.3
    variables:
      ARTIFACTORY_URL: $ARTIFACTORY_URL
      ARTIFACTORY_USER: $ARTIFACTORY_USER
      ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
      NPM_COMMAND: "publish"
      NPM_TARGET_REPO: "npm-local"
      BUILD_NAME: "npm-pipe-example"
```

See the pipe's README.md for the full list of variables you may specify.

Pushing and Pulling Docker Images to and from Artifactory

The **Artifactory Docker** pipe pushes a docker image to a docker repository in Artifactory.

By default, this pipe will also capture build-info and publish it to Artifactory as metadata associated with the docker image. In addition to details about the build and the build environment, the build info includes the image layers as build dependencies and build artifacts.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. The Docker image tag.
4. The Artifactory Docker registry to store the Docker image.
5. Other pipe options, such as a build name to associate with the Docker image.

Example: Build and publish a Docker image to Artifactory

```
# Build and publish Docker image to Artifactory
- pipe: JfrogDev/artifactory-docker:0.2.11
  variables:
    ARTIFACTORY_URL: $ARTIFACTORY_URL
    ARTIFACTORY_USER: $ARTIFACTORY_USER
    ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
    DOCKER_IMAGE_TAG: "art-docker.mycompany.team/docker-pipe-example:${BITBUCKET_BUILD_NUMBER}"
    DOCKER_TARGET_REPO: "docker-stage-local"
    BUILD_NAME: "docker-pipe-example"
```

See the pipe's README.md for the full list of variables you may specify.

Promoting Builds in Artifactory

To support the artifacts life-cycle, Artifactory supports promoting published builds from one repository to another.

The **Artifactory Build Promotion** pipe promotes a build, by either copying or moving the build artifacts and/or dependencies to a target repository.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. The Artifactory repository to receive the promoted build's artifacts
4. The status message to set during promotion.
5. Other pipe options, such as a build name to associate with the promoted build.

Example: Promote a build in Artifactory

```
# Promote Build in Artifactory
- pipe: JfrogDev/artifactory-build-promotion:0.2.10
    variables:
      ARTIFACTORY_URL: $ARTIFACTORY_URL
      ARTIFACTORY_USER: $ARTIFACTORY_USER
      ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
      TARGET_REPO: "npm-prod-local"
      STATUS: "Staged"
      BUILD_NAME: "npm-pipe-example"
```

See the pipe's README.md for the full list of variables you may specify.

Scanning Published Builds with JFrog Xray

The **Artifactory Xray Scan** pipe triggers a build scan with JFrog Xray.

When the scan is triggered, Artifactory will pass on the build to Xray, which will then scan the build artifacts. The action is synchronous, meaning it will wait for the scan to finish before returning control to the script.

If the **Allow fail build** task option is set and Xray is configured to fail the build, the build pipeline will fail, if vulnerabilities are found.

When configuring the pipe you must specify:

1. Your Artifactory service URL
2. Your credentials
3. Other pipe options, such as the name of the build to scan.

Example: Promote a build in Artifactory

```
# Scan published build through Xray
- pipe: JfrogDev/artifactory-xray-scan:0.2.7
    variables:
      ARTIFACTORY_URL: $ARTIFACTORY_URL
      ARTIFACTORY_USER: $ARTIFACTORY_USER
      ARTIFACTORY_PASSWORD: $ARTIFACTORY_PASSWORD
      BUILD_NAME: "npm-pipe-example"
```

See the pipe's README.md for the full list of variables you may specify.