Build Integration

Overview

JFrog Artifactory supports build integration whether you are running builds on one of the common CI servers in use today, on cloud-based CI servers or standalone without a CI server. Integration of Artifactory into your build ecosystem provides important information that supports fully reproducible builds through visibility of artifacts deployed, dependencies and information on the build environment.

The Artifactory Build Integration provides a set of plugins you can use with industry standard CI systems and build tools that enable you to:

- View all the builds that are published in Artifactory.
- Explore the modules of each build, including published artifacts and corresponding dependencies.
- Obtain information about the build environment.
- Treat all the artifacts and/or dependencies from a specific build as a single unit and perform build promotion.
- Search for artifacts in the context of a build using AQL or File Specs.
- Delete old artifacts by defining build retention.
- Receive bidirectional links between build and artifact information inside the build server and Artifactory pages.

Behind the scenes, the JFrog Artifactory client for your CI server performs these major tasks:

1. Resolves all dependencies from a resolution repository in Artifactory.
2. Deploys all the artifacts to Artifactory.
3. Collects the Build-Info from the build agents and saves it in Artifactory.

What is Build-Info?

Build-info is all the information collected by the build agent which includes details about the build. The build-info includes the list of project modules, artifacts, dependencies, environment variables and more. When using one of the JFrog clients to build the code, the client can collect the build-info and publish it to Artifactory. When the build-info is published to Artifactory, all the published details become visible in the Artifactory UI.

Supported CI Integrations

The following tools that are currently supported:

- Jenkins
- TeamCity
- Bamboo
- Artifactory Azure DevOps Extension
- JFrog CLI

JFrog Build Offerings By Package Type

The following table lists the JFrog Build offering by package type with active links directly to the relevant offering.

<table>
<thead>
<tr>
<th>Package Type</th>
<th>CI Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jenkins</td>
</tr>
<tr>
<td>Generic</td>
<td>☑️</td>
</tr>
<tr>
<td>Conan</td>
<td>☑️</td>
</tr>
<tr>
<td>Docker</td>
<td>☑️</td>
</tr>
</tbody>
</table>
Inspecting Builds

You can view all your CI server projects that deploy their output to the JFrog Platform in the **Builds** page under the **Application** module under **Artifactory | Builds**.

<table>
<thead>
<tr>
<th>Go</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradle</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ivy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Maven</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>npm</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NuGet</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Selecting a build item to display all runs of that build reflecting the build history in the CI server, and a complete build-level information.

**Permissions**

To view build information you must have the ‘Read’ permission on some repository path.

**Viewing Build Number Information**

You can select the **Build Name | Build Number** to drill down into a specific build number. This displays detailed information about the build number, and enables you to compare it with another build as described in the following sections.

The Build Info Details screen is divided in these panels:

- The **horizontal panel** along the top displays basic information about the selected build number.
- The **Build details**. Select a file to see its details in the panel on the right.
There are three categories of information:

1. General build information about the build and its environment.
2. Build modules along with their artifacts and dependencies.
3. Generic view of the build information in JSON format.

**General Build Information**

The horizontal panel displays general information about the build.

**Published Modules**

The Published Modules tab displays the modules published into Artifactory as a result of the build, along with the number of artifacts and dependencies that they contain.
Module Artifacts and Dependencies

Selecting a published module that was built will display its artifacts and dependencies. You can group these by type or scope by clicking the corresponding column header.

You can click any item to download it directly, or click its Repo Path to view it in the Tree Browser.

Environment

The Environment tab displays an extensive list of properties and environment settings defined for the selected build. You can use these to reproduce the environment precisely if you need to rerun the build.
Xray Data

Xray scanning requires Artifactory Pro X, Enterprise with Xray, or an Enterprise+ license.

For more information, see Viewing Xray data on Builds.

Issues
The Issues tab displays information about issues that were dealt with in the build. The CI clients that currently support this feature are: Jenkins, Bamboo and JFrog CLI. When using Jenkins CI, if you set the Enable JIRA Integration option in the Jenkins Artifactory Plugin, the Issues tab will display any JIRA issues that have been addressed by this build.

### Build Diff

The Diff tab allows you to compare the selected build with any other build. Once you select a build number in the Select A Build To Compare Against field, Artifactory displays all the differences between the builds that were detected including new artifacts added, dependencies deleted, properties changed and more.

### Release History

The Release History tab displays promote information.
Build Info JSON

Generic BuildInfo View

This tab displays the raw buildInfo JSON representation of the build information in Artifactory. This data can be accessed via the REST API or used for debugging and is also available in the Artifactory OSS version.

Build Info JSON

```
1 {
2     "properties": {
3         "buildInfo.env.BUILD_NUMBER": "234-19",
4         "buildInfo.env.HOSTNAME": "b50e624071e",
5         "buildInfo.env.BUILD_USER": "jfrog",
6         "buildInfo.env.ENVOID": "71",
7         "buildInfo.env.YARN_VERSION": "1.19.2",
8         "buildInfo.env.ROOT": "/root",
9         "buildInfo.env.VERSION": "4.234.19",
10        "buildInfo.env.xCTrivy": "/usr/local/bin/trivy",
11        "buildInfo.env.KIND_VERSION": "12.3.0",
12        "buildInfo.env.xCTrivyUser": "admin",
13        "buildInfo.env.buildNumber": "234",
14        "buildInfo.env.PATH": "/usr/local/bin:/usr/local/bin:/usr/bin:/usr/sbin:/usr/bin:/sbin:/bin"
15      }
16  }
```

Effective Permissions
Displays the list of permissions in the context of users, groups or permission targets on the build level. For more information, see Permissions.

Managing Builds

Searching Builds

The global Application Search includes a dedicated Build search bar allowing you to search for specific builds within a specified data range.

For more information, see the Application Search.

Deleting a Build

You can delete a single build or multiple builds from the UI, select the builds and click Delete or hover over a build number and click Delete.
Viewing Xray Data on Builds

Xray scanning requires Artifactory Pro X, Enterprise with Xray, or an Enterprise+ license.

JFrog Xray can be integrated into an organization’s CI/CD pipeline to make sure that build jobs containing vulnerabilities are stopped early on in the process. As part of a fully automated process, Xray receives information about a build that has just been run by your CI server, it then runs a deep recursive scan on the build down to the deepest level dependency, and if any vulnerabilities are found, Xray will return an indication to the calling CI server.

For your build numbers, you can view the Xray scan status and the severity level if a vulnerability was detected in the Application module under Artifactory | Builds | [build_name] | [build_number].

Under the Xray Data tab, you can view these dedicated Xray related tabs with the option to run a set of actions on the build. For detailed information on each tab, see Analyzing Resource Scan Results.

Click a Build ID to view detailed Xray data for the build number.
Viewing Associated Builds on Artifacts

When viewing an artifact within the Tree Browser, you can see all of the builds with which that artifact is associated, whether directly or as a dependency in the Builds tab.
Build-Info Repository

The Build Info repository is a local repository that stores your build info json files as artifacts, providing you with:

- **user / group permission management** equivalent to managing permissions on repositories with include/exclude patterns on build names
- **build info replication** to other instances, since they are stored as artifacts in a repository
- **improved accessibility** to the build info json files and overall **performance** of the Builds module

Whenever a new build is uploaded to Artifactory (either by a CI server plugin, directly using a Build REST API or from the Artifactory UI), it will automatically get deployed to this default `artifactory-build-info` repository.

Build info json files should be uploaded to the root of the repository. Artifactory will automatically place them in the repository using the following structure: `artifactory-build-info > [build name] > [build number]-[deploy timestamp].json`

**Using the artifactory-build-info repository**

The `artifactory-build-info` repository is automatically created and used by default. It cannot be removed. Removing it will cause all your build info to be deleted and a new default build info repository will be recreated.

**Cleanup Policies**

If you have existing cleanup policies, defined in Artifactory User Plugins, that delete artifacts from your local repositories according to a policy, they will also affect the `artifactory-build-info` repository and should be updated accordingly. Exclude this repository from your cleanup policies to ensure that your build info files are not deleted.

To view the `artifactory-build-info` repository:

From the **Application** module, click **Artifactory | Artifacts** and search for the `artifactory-build-info` repository.
Migrating to the artifactory-build-info Repository

The migration process may be configured through the following system properties. The migration will run on the primary node.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>artifactory.migration.job.dbQueryLimit</td>
<td>[Default: 100] Specifies the number of rows that should be retrieved each time the migration job queries the database for build info json entries that exist in the database but not yet in the build-info repository values.</td>
</tr>
<tr>
<td>artifactory.migration.job.batchSize</td>
<td>[Default: 10] Build info json artifacts are deployed concurrently in batches, then a sleep cycle is initiated. This property specifies the number of artifacts in each batch.</td>
</tr>
<tr>
<td>artifactory.build.info.migration.job.queue.workers</td>
<td>[Default: 2] Specifies the number of concurrent threads that should execute actual build info artifact updates. Each concurrent artifact update will incur a read from the database to retrieve the build info json, followed by an upload to the build-info repo.</td>
</tr>
<tr>
<td>artifactory.migration.job.sleepIntervalMillis</td>
<td>[Default: 5000 milliseconds] Specifies the duration of the sleep cycle which is initiated after each batch of updates.</td>
</tr>
<tr>
<td>artifactory.migration.job.waitForCluster.sleepIntervalMillis</td>
<td>[Default: 30000 milliseconds] The build-info repository migration can only start when the cluster has been fully upgraded and all nodes run the same minimal required version (6.6.0), where applicable. This parameter specifies the amount of time the master node will wait between each check for all of the other nodes' current version.</td>
</tr>
</tbody>
</table>