Working with Gradle

Overview

Artifactory provides tight integration with Gradle. All that is needed is a simple modification of your `build.gradle` s cript file with a few configuration parameters.

Both the new and older publishing mechanisms of Gradle are supported, however some of the steps to configure the Gradle Artifactory Plugin depend on the version you are using, and these are detailed in the documentation pages.

The Gradle Artifactory Plugin can be used whether you are running builds using a CI server, or running standalone builds. In either case, you should note the following points:

1. **CI Server Integration**
   When running Gradle builds in your continuous integration server, we recommend using one of the Artifactory Plugins for Jenkins, TeamCity or Bamboo.
   You can use your build server UI to configure resolving and publishing artifacts through Artifactory to capture exhaustive build information.
2. **Standalone Integration**
   The Gradle Artifactory plugin offers a simple DSL to perform the following steps in your Gradle build:
   a. Define the default dependency resolution from Artifactory.
   b. Define configurations that publish artifacts to Artifactory after a full (multi-module) successful build.
   c. Define properties that should be attached to published artifacts in Artifactory metadata.
   d. Capture and publish a build-info object to the Artifactory build-info REST API to provide a fully traceable build context.

Source Code Available!

This Gradle Artifactory Plugin is an open source project on GitHub which you can freely browse and fork.

The following sections describe the main configuration steps and provide a sample Gradle script that shows the information you need to get started using Gradle with Artifactory.

Page Contents

Read more

- Gradle Artifactory Plugin

Integration Benefits

JFrog Artifactory and Gradle Reositories

Configuring Artifact Resolution

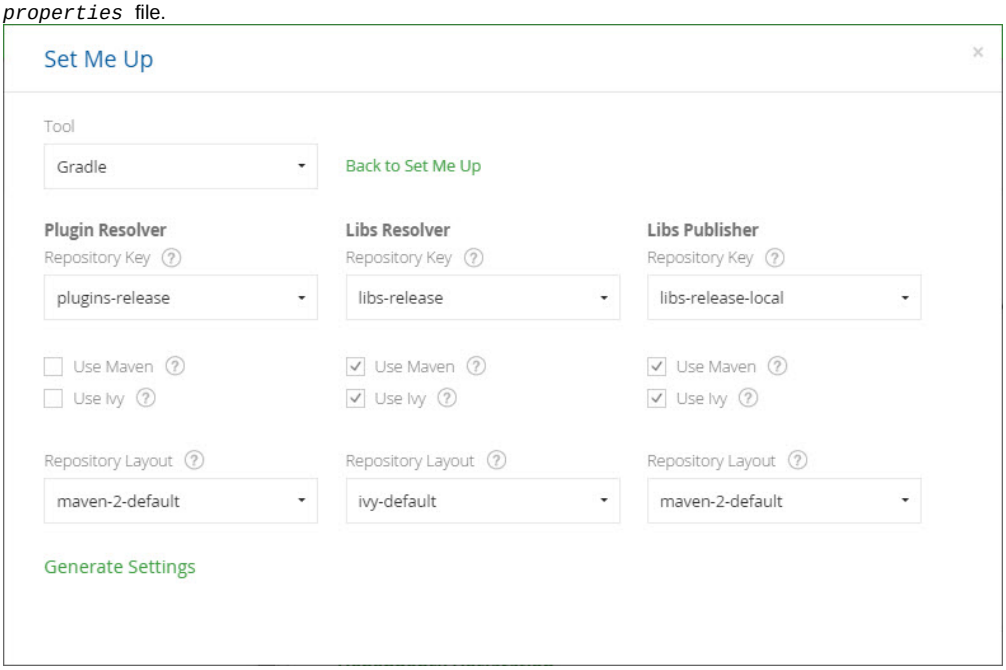
Using the Gradle Build Script Generator

With Artifactory's **Gradle Build Script Generator**, you can easily create a Gradle init script that handles resolution.

In the **Artifact Repository Browser** of the **Artifacts** module, select **Set Me Up.** In the **Set Me Up** dialog, set **Gradle** in the **Tool** field and click "Generate Gradle Settings". You can now specify the settings you want to configure for Gradle.

Plugin/Libs Resolver	The repository that should be used to resolve plugins /libraries
Use Maven /Use Ivy	When checked, specifies that resolving should be done using the Maven/Ivy pattern
Libs Publisher	The repository that should be used to publish libraries
Use Maven /Use Ivy	When checked, specifies that library should be published using a Maven/Ivy descriptor
Repository Layout	Specifies the layout of the corresponding repository

Once you have configured the settings for Gradle you can click "Generate Settings" to generate and save the `build.gradle` and `gradle.properties` file.



Provisioning Dynamic Settings for Users

Artifactory lets you deploy and provision a dynamic settings template for your users. Once downloaded, settings are generated according to your own logic and can automatically include user authentication information.

For more details, please refer to Provisioning Build Tool Settings section under Filtered Resources.

Sample Build Script and Properties

You can download sample scripts from the JFrog GitHub public repository.

Running Gradle

For Gradle to build your project and upload generated artifacts to Artifactory, you need to run the following command:

```
gradle artifactoryPublish
```

For more details on building your projects with Gradle, please refer to the Gradle Documentation.

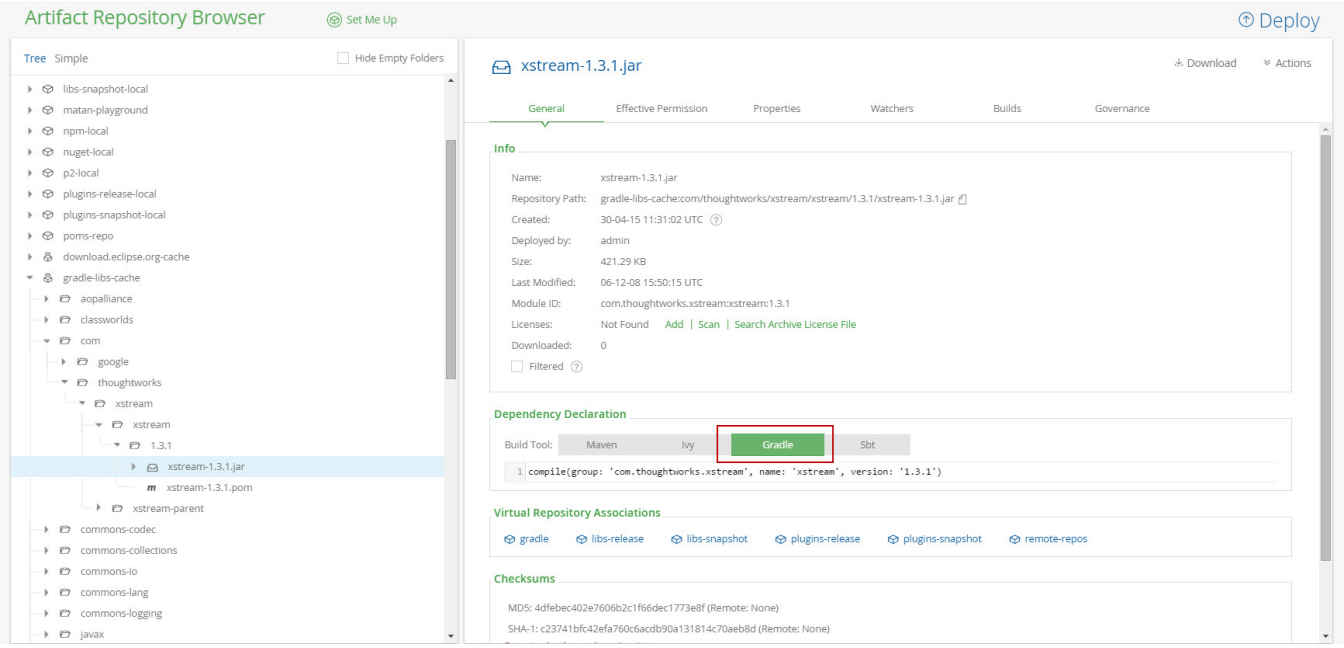
Getting debug information from Gradle

We highly recommend running Gradle with the `-d` option to get useful and readable information if something goes wrong with your build.

Dependency Declaration Snippets

Artifactory can provide you with dependency declaration code snippets that you can simply copy into the **Gradle Dependency Declaration** section of your *`build.gradle`* file.

In the **Artifact Repository Browser** of the **Artifacts** module, drill down in the repository tree and select a relevant artifact. Under the **Dependency Declaration** section, select **Gradle** to display the corresponding dependency declaration that you can copy into your *`build.gradle`* file.



Optimizing Gradle Builds

From V3.5, Gradle introduces a build cache feature that lets you reuse outputs produced by other builds, instead of rebuilding them, and dramatically reduce build time. This feature supports not only your local filesystem cache, but also remote caches that can be shared across your organization.

The Gradle team has measured an average reduction of 25% in total build time, and even a reduction of 80% with some of their commits!

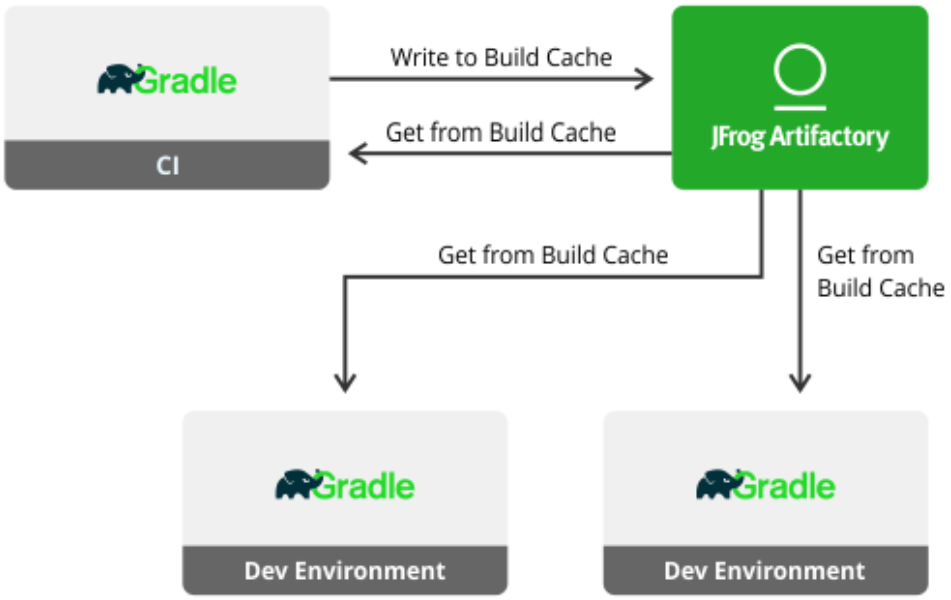
To optimize your Gradle builds:

1. Configure Artifactory to be your Gradle build cache
2. Configure Gradle to use the build cache in Artifactory

Configuring Artifactory

Artifactory can be used as the Gradle build cache by simply creating a generic repository in Artifactory.

For example, the following is a simple use case where the CI server builds a project and stores the build cache in Artifactory for later use by the following builds. This will greatly improve the build time in your local developer environments.



Configuring Gradle

Configure Gradle to use the build cache and point it to Artifactory.

gradle.properties

```
artifactory_user=admin
artifactory_password=password
artifactory_url=http://localhost:8081/artifactory
org.gradle.caching=true
gradle.cache.push=false
```

settings.gradle

Set the *gradle.cache.pus*h property to true, on the CI server, by overriding it using *-Pgradle.cache.push=true*.

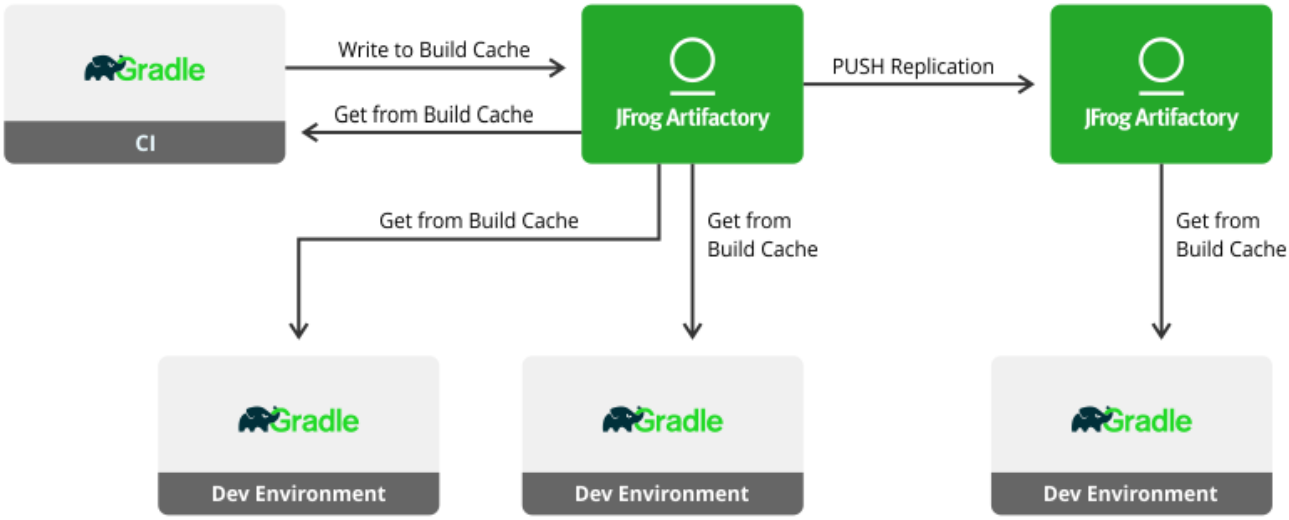
```
include "shared", "api", "services:webservice"

ext.isPush = getProperty('gradle.cache.push')

buildCache {
  local {
    enabled = false
  }
  remote(HttpBuildCache) {
   url = "${artifactory_url}/gradle-cache-example/"
   credentials {
    username = "${artifactory_user}"
    password = "${artifactory_password}"
   }
   push = isPush
  }
}
```

Replication Across Different Sites

You can also use Artifactory as a distributed cache that's synchronized across both local and remote teams using push and pull repository replication, and improve both your local and remote build times.



Watch the Screencast