Gradle Artifactory Plugin

Overview

The Gradle Artifactory Plugin allows you to deploy your build artifacts and build information to Artifactory and also to resolve your build dependencies from Artifactory.

Latest Version

For the latest version number of the Gradle Artifactory Plugin, please refer to the **download page on Bintray**.

Download and Installation

Automatic Installation

Build script snippet for use in all Gradle versions

```
buildscript {
  repositories {
    jcenter()
  }
  dependencies {
    classpath "org.jfrog.buildinfo:build-info-extractor-gradle:latest.release"
  }
}
apply plugin: "com.jfrog.artifactory"
```

Build script snippet for use in Gradle 2.1 and above

```
// Please replace <plugin version> with the version of the Gradle Artifactory
Plugin.
plugins {
  id "com.jfrog.artifactory" version "<plugin version>"
}
```

Currently the "plugins" notation cannot be used for applying the plugin for sub projects, when used from the root build script

Page Contents

Integration Benefits

JFrog Artifactory and Gradle Repositories

Manual Installation

The latest plugin jar file can be downloaded from JFrog Bintray. Download and copy the *build-info-extractor-gradle-<x.y.z>-uber.jar* into your gradle home plugins directory (*~/.gradle/plugins*).

Then add the following line to your project build script:

```
buildscript.dependencies.classpath files(new File(gradle.gradleUserHomeDir, 'plugins/build-info-extractor-
gradle-<x.y.z>-uber.jar'))
```

Configuration

Using the Artifactory Plugin DSL

The Gradle Artifactory plugin is configured using its own Convention DSL inside the `build.gradle` script of your root project.

The syntax of the Convention DSL is described below:

We highly recommend also using our examples as a reference when configuring the DSL in your build scripts.

Mandatory items within the relevant context are prefixed with '+'. All other items are optional.

Dependencies Resolution

```
repositories {
        jcenter()
        maven {
                +url "http://repo.myorg.com/artifactory/libs-releases" // The Artifactory (preferably virtual)
repository to resolve from
                credentials {                                // Optional resolver credentials (leave out to use
anonymous resolution)
                        username = "resolver" // Artifactory user name
                      password = "resolverPaS*" // Password or API Key
                }
        }

        ivy {
                +url "http://localhost:8081/artifactory/ivy-releases"
                layout "pattern", {                // Optional section for configuring Ivy-style resolution.
                    ivy "[organization]/[module]/[revision]/ivy.xml"
                artifact "[organization]/[module]/[revision]/[module]-[revision](-[classifier]).[ext]"
                m2compatible = true        // Convert any dots in an [organization] layout value to path
separators, similar to Maven's groupId-to-path conversion. False if not specified.
            }
        }
}
```

Please follow this documentation for different ways to configure your repositories.

Artifacts and BuildInfo Publication

```
artifactory {
  +contextUrl = 'http://repo.myorg.com/artifactory'   //The base Artifactory URL if not overridden by the
publisher/resolver
  publish {
    contextUrl = 'http://repo.myorg.com/artifactory'   //The base Artifactory URL for the publisher
    //A closure defining publishing information
    repository {
      +repoKey = 'integration-libs'   //The Artifactory repository key to publish to
      +username = 'deployer'          //The publisher user name
      password = 'deployerPaS*'       //The publisher password or API key
      ivy {
        //Optional section for configuring Ivy publication. Assumes Maven repo layout if not specified
        ivyLayout = '[organization]/[module]/[revision]/[type]s/ivy-[revision].xml'
        artifactLayout = '[organization]/[module]/[revision]/[module]-[revision](-[classifier]).[ext]'
        mavenCompatible = true //Convert any dots in an [organization] layout value to path separators, similar
to Maven's groupId-to-path conversion. True if not specified
      }
    }
    defaults {
                //List of Gradle Publications (names or objects) from which to collect the list of artifacts to
be deployed to Artifactory.
        publications ('ivyJava','mavenJava','foo')
                ////List of Gradle Configurations (names or objects) from which to collect the list of
artifacts to be deployed to Artifactory.
                publishConfigs('archives', 'published')
        properties = ['qa.level': 'basic', 'q.os': 'win32, deb, osx']  //Optional map of properties to attach
to all published artifacts
        /*
        The properties closure in the "defaults" task uses the following syntax:
        properties {
            publicationName 'group:module:version:classifier@type', key1:'value1', key2:'value2', ...
        }
        publicationName: A valid name for a publication of the project. You can use all to apply the properties
to all publications.
        group:module:version:classifier@type: A filter that specifies the artifacts to which properties should
be attached.
                The filter may contain wildcards: * for all characters or ? for a single character.
        key:'value': A list of key/value properties that will be attached to to the published artifacts
matching the filter.
        */
        properties {                                                   //Optional closure to attach properties
to artifacts based on a list of artifact patterns per project publication
            foo '*:*:*:*@*', platform: 'linux', 'win64'                //The property platform=linux,win64 will
be set on all artifacts in foo publication
            mavenJava 'org.jfrog:*:*:*@*', key1: 'val1'                //The property key1=val1 will be set on
all artifacts part of the mavenJava publication and with group org.jfrog
            all 'org.jfrog:shared:1.?:*@*', key2: 'val2', key3: 'val3' //The properties key2 and key3 will be
set on all published artifacts (all publications) with group:artifact:version

//equal to org.jfrog:shared:1.?
        }
        publishBuildInfo = true   //Publish build-info to Artifactory (true by default)
        publishArtifacts = true   //Publish artifacts to Artifactory (true by default)
        publishPom = true   //Publish generated POM files to Artifactory (true by default).
        publishIvy = true   //Publish generated Ivy descriptor files to Artifactory (true by default).
                publishForkCount = 8 //Number of threads to use for artifacts publishing (8 by default).
Supported since version 4.10.0.
    }
  }
  // Redefine basic properties of the build info object
  clientConfig.setIncludeEnvVars(true)
  clientConfig.setEnvVarsExcludePatterns('*password*,*secret*')
  clientConfig.setEnvVarsIncludePatterns('*not-secret*')
  clientConfig.info.addEnvironmentProperty('test.adding.dynVar',new java.util.Date().toString())
  clientConfig.info.setBuildName('new-strange-name')
  clientConfig.info.setBuildNumber('' + new java.util.Random(System.currentTimeMillis()).nextInt(20000))
  clientConfig.timeout = 600 // Artifactory connection timeout (in seconds). The default timeout is 300 seconds.
}
```

Controlling how environment variables are exposed

As shown in the example above, you can control which environment variables are exposed in `clientConfig.setIncludeEnvVars` using `clientConfig.setEnvVarsExcludePatterns` and `clientConfig.setEnvVarsIncludePatterns`. These calls specify which environment variables should be excluded or included respectively using a parameter which is a comma-separated list of expressions to exclude or include. The expressions can use a star ('*') wildcard to specify multiple environment variables.

Using the old Gradle publishing mechanism?

```
If you are using the old Gradle publishing mechanism, you need to replace the above defaults closure with the
following one:

defaults {
        //This closure defines defaults for all 'artifactoryPublish' tasks of all projects the plugin is
applied to
        publishConfigs ('a','b','foo')                                  //Optional list of configurations (names
or objects) to publish.
                                                                        //The 'archives' configuration is used
if it exists and no configuration is specified
        mavenDescriptor = '/home/froggy/projects/proj-a/fly-1.0.pom'   //Optional alternative path for a POM to
be published (can be relative to project baseDir)
        ivyDescriptor = 'fly-1.0-ivy.xml'                               //Optional alternative path for an ivy
file to be published (can be relative to project baseDir)
        properties = ['qa.level': 'basic', 'q.os': 'win32, deb, osx']  //Optional map of properties to attach
to all published artifacts
        /*
        The properties closure in the "defaults" task uses the following syntax:
        properties {
            configuration 'group:module:version:classifier@type', key1:'value1', key2:'value2', ...
        }
        configuration: A configuration that is a valid name of a configuration of the project. You can use all
to apply the properties to all configurations.
        group:module:version:classifier@type: An artifact specification filter for matching the artifacts to
which properties should be attached.
                The filter may contain wildcards: * for all characters or ? for a single character.
        key:'value': A list of key/value(s) properties that are attached to to the published artifacts matching
the filter.
        */
        properties {                                                    //Optional closure to attach properties
to artifacts based on a list of artifact patterns per project configuration
            foo '*:*:*:*@*', platform: 'linux', 'win64'                //The property platform=linux,win64 will
be set on all artifacts in foo configuration
            archives 'org.jfrog:*:*:*@*', key1: 'val1'                 //The property key1=val1 will be set on
all artifacts part of the archives configuration and with group org.jfrog
            all 'org.jfrog:shared:1.?:*@*', key2: 'val2', key3: 'val3' //The properties key2 and key3 will be
set on all published artifacts (all configurations) with group:artifact:version
                                                                        //equal to org.jfrog:shared:1.?
        }
        publishBuildInfo = true   //Publish build-info to Artifactory (true by default)
        publishArtifacts = true   //Publish artifacts to Artifactory (true by default)
        publishPom = true         //Publish generated POM files to Artifactory (true by default)
        publishIvy = false        //Publish generated Ivy descriptor files to Artifactory (false by default)
    }
```

The Artifactory Project Publish Task

The Artifactory Publishing Plugin creates an **`artifactoryPublish`** Gradle task for each project the plugin is applied to. The task is configured by the `publish` closure of the plugin.

You can configure the project-level task directly with the task's `artifactoryPublish` closure, which uses identical Syntax to that of the plugin's `publish.defaults` closure.

```
artifactoryPublish {
    skip = false //Skip build info analysis and publishing (false by default)
        contextUrl = 'http://repo.myorg.com/artifactory'
    publications ('a','b','c')
    properties = ['qa.level': 'basic', 'q.os': 'win32, deb, osx']
    properties {
        c '**:**:**:*@*', cProperty: 'only in c'
    }
        clientConfig.publisher.repoKey = 'integration-libs'
        clientConfig.publisher.username = 'deployer'
        clientConfig.publisher.password = 'deployerPaS'
}
```

Controlling Publication in Sub-Projects

The Gradle Artifactory Plugin allows you to define different publication configuration for sub projects. You may also define the configuration once for the whole project by defining the **artifactory** closure only in the root project. The plugin also lets you disable publication for a sub-module.

- When defining the configuration anywhere in the hierarchy, all sub-projects beneath it inherit the configuration and can override it whether it is defined in the root or in a sub-project.
- Each sub-project can override the `publish` closure or the `repositories` closure, or both of them.

Example for overriding publication only

```
artifactory {
        publish {
                contextUrl = 'http://localhost:8081/artifactory'
                repository {
                        repoKey = "libs-snapshot-local"
                        username = "user"
                        password = "pass"
                }
        }
}
```

- For buildInfo to be published, a publish closure must be defined in the root project.
- Use the `artifactoryPublish.skip` flag to deactivate analysis and publication.
- Activate the corresponding **`artifactoryPublish`** Gradle task manually for each project to which you wish to apply the plugin. For example in our Gradle project example you can run:

Activating the plugin manually

```
./gradlew clean api:artifactoryPublish shared:artifactoryPublish
```

Controlling the Build Name and Number

By default, BuildInfo is published with a build name constructed from the name of your root project and a build number that is the start date of the build. You can control the build name and number values by specifying the following properties respectively:

Specifying the build name and number

```
buildInfo.build.name=my-super-cool-build
buildInfo.build.number=r9001
```

The above properties should be added to your project's gradle.properties file.

Examples

Project examples which use the Gradle Artifactory Plugin are available here.