Docs

[User Manual](http://docs.google.com/userguide/userguide.html)

[Guides and Tutorials](https://guides.gradle.org)

[DSL Reference](http://docs.google.com/dsl/)

[Javadoc](http://docs.google.com/javadoc/)

[Release Notes](http://docs.google.com/release-notes.html)

[Forums](https://discuss.gradle.org/)

[Training](https://gradle.org/training/)

[Try Gradle Enterprise](https://gradle.com/enterprise)

[PDF](http://docs.google.com/userguide/userguide.pdf)

* [User Manual Home](http://docs.google.com/userguide/userguide.html)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Installing Gradle](http://docs.google.com/userguide/installation.html)
* [Tutorials](https://guides.gradle.org/)

### Reference

* [Groovy DSL Reference](http://docs.google.com/dsl/)
* [Gradle API Javadoc](http://docs.google.com/javadoc/)
* [Core Plugins](http://docs.google.com/userguide/plugin_reference.html)
* [Gradle & Third-party Tools](http://docs.google.com/userguide/third_party_integration.html)

### Getting Started

* [Creating New Gradle Builds](https://guides.gradle.org/creating-new-gradle-builds/)
* [Creating Build Scans](https://guides.gradle.org/creating-build-scans/)
* [Migrating From Maven](https://guides.gradle.org/migrating-from-maven/)

### Running Gradle Builds

* [Command-Line Interface](http://docs.google.com/userguide/command_line_interface.html)
* [Customizing Execution](#gjdgxs)
  + [Configuring the Build Environment](http://docs.google.com/userguide/build_environment.html)
  + [Configuring the Gradle Daemon](http://docs.google.com/userguide/gradle_daemon.html)
  + [Initialization Scripts](http://docs.google.com/userguide/init_scripts.html)
* [Directory Layout](http://docs.google.com/userguide/directory_layout.html)
* [Executing Multi-Project Builds](http://docs.google.com/userguide/intro_multi_project_builds.html)
* [Gradle Wrapper](http://docs.google.com/userguide/gradle_wrapper.html)
* [Troubleshooting](http://docs.google.com/userguide/troubleshooting.html)
* [Using Build Scans](https://docs.gradle.com/build-scan-plugin)
* [Enabling and Configuring the Build Cache](http://docs.google.com/userguide/build_cache.html)
* [Integrating Separate Gradle Builds (Composite Builds)](http://docs.google.com/userguide/composite_builds.html)

### Authoring Gradle Builds

* [Fundamentals](#30j0zll)
  + [Introducing the Basics of Build Scripts](http://docs.google.com/userguide/tutorial_using_tasks.html)
  + [Working with Tasks](http://docs.google.com/userguide/more_about_tasks.html)
  + [Learning More About Build Scripts](http://docs.google.com/userguide/writing_build_scripts.html)
  + [Working with Files](http://docs.google.com/userguide/working_with_files.html)
  + [Creating Custom Task Types](http://docs.google.com/userguide/custom_tasks.html)
  + [Using Gradle Plugins](http://docs.google.com/userguide/plugins.html)
  + [The Standard Gradle Plugins](http://docs.google.com/userguide/standard_plugins.html)
  + [Understanding the Build Lifecycle](http://docs.google.com/userguide/build_lifecycle.html)
  + [Working with Logging](http://docs.google.com/userguide/logging.html)
  + [Configuring Multi-Project Builds](http://docs.google.com/userguide/multi_project_builds.html)
* [Best Practices](#1fob9te)
  + [Authoring Maintainable Build Scripts](http://docs.google.com/userguide/authoring_maintainable_build_scripts.html)
  + [Organizing Gradle Projects](http://docs.google.com/userguide/organizing_gradle_projects.html)
  + [Optimizing Build Performance](https://guides.gradle.org/performance/)
  + [Using the Build Cache](https://guides.gradle.org/using-build-cache/)
* [Dependency Management](#3znysh7)
  + [Introduction to Dependency Management](http://docs.google.com/userguide/introduction_dependency_management.html)
  + [Dependency Management Terminology](http://docs.google.com/userguide/dependency_management_terminology.html)
  + [Dependency Types](http://docs.google.com/userguide/dependency_types.html)
  + [Repository Types](http://docs.google.com/userguide/repository_types.html)
  + [Declaring Dependencies](http://docs.google.com/userguide/declaring_dependencies.html)
  + [Declaring Repositories](http://docs.google.com/userguide/declaring_repositories.html)
  + [Inspecting Dependencies](http://docs.google.com/userguide/inspecting_dependencies.html)
  + [Managing Dependency Configurations](http://docs.google.com/userguide/managing_dependency_configurations.html)
  + [Managing Transitive Dependencies](http://docs.google.com/userguide/managing_transitive_dependencies.html)
  + [Dependency Locking](http://docs.google.com/userguide/dependency_locking.html)
  + [Troubleshooting Dependency Resolution](http://docs.google.com/userguide/troubleshooting_dependency_resolution.html)
  + [Customizing Dependency Resolution Behavior](http://docs.google.com/userguide/customizing_dependency_resolution_behavior.html)
  + [Dependency Cache Internals](http://docs.google.com/userguide/dependency_cache.html)
  + [Working with Dependencies](http://docs.google.com/userguide/working_with_dependencies.html)
* [Publishing Artifacts](http://docs.google.com/userguide/artifact_management.html)
* [C++ Projects](#2et92p0)
  + [Building Native Software](http://docs.google.com/userguide/native_software.html)
  + [Software Model Concepts](http://docs.google.com/userguide/software_model_concepts.html)
  + [Rule-based Model Configuration](http://docs.google.com/userguide/software_model.html)
  + [Implementing Model Rules in a Plugin](http://docs.google.com/userguide/rule_source.html)
  + [Extending the Software Model](http://docs.google.com/userguide/software_model_extend.html)
* [Java Projects](#tyjcwt)
  + [Building Java & JVM projects](http://docs.google.com/userguide/building_java_projects.html)
  + [Testing Java & JVM projects](http://docs.google.com/userguide/java_testing.html)
* [Advanced Techniques](#3dy6vkm)
  + [Configuring Tasks Lazily](http://docs.google.com/userguide/lazy_configuration.html)
  + [Developing Parallel Tasks](https://guides.gradle.org/using-the-worker-api/)
  + [Testing Your Build with TestKit](http://docs.google.com/userguide/test_kit.html)
  + [Using Ant from Gradle](http://docs.google.com/userguide/ant.html)
* [Sample Gradle builds](#1t3h5sf)
  + [Groovy DSL Samples](https://github.com/gradle/gradle/tree/master/subprojects/docs/src/samples)
  + [Kotlin DSL Samples](https://github.com/gradle/kotlin-dsl/tree/master/samples)

### Extending Gradle

* [Writing Custom Plugins](http://docs.google.com/userguide/custom_plugins.html)
* [Plugin Development Guides](https://gradle.org/guides/?q=Plugin+Development)

[Edit this page](https://github.com/gradle/gradle/edit/master/subprojects/docs/src/docs/userguide/)

# Maven Plugin

Contents

[Usage](#4d34og8)

[Tasks](#2s8eyo1)

[Dependency management](#17dp8vu)

[Convention properties](#3rdcrjn)

[Convention methods](#26in1rg)

[Interacting with Maven repositories](#lnxbz9)

| **✨** | This chapter describes deploying artifacts to Maven repositories using the *original* publishing mechanism available in Gradle 1.0: in Gradle 1.3 a new mechanism for publishing was introduced. This new mechanism introduces some new concepts and features that make Gradle publishing even more powerful and is now the preferred option for publishing artifacts.  You can read about the new publishing plugins in [Publishing Ivy](http://docs.google.com/publishing_ivy.html#publishing_ivy) and [Publishing Maven](http://docs.google.com/publishing_maven.html#publishing_maven). |
| --- | --- |

The Maven plugin adds support for deploying artifacts to Maven repositories.

[Usage](#4d34og8)

To use the Maven plugin, include the following in your build script:

[Example: Using the Maven plugin](#35nkun2)

**build.gradle**

apply plugin: 'maven'

[Tasks](#2s8eyo1)

The Maven plugin defines the following tasks:

install — [Upload](http://docs.google.com/dsl/org.gradle.api.tasks.Upload.html)

*Depends on*: All tasks that build the associated archives.

Installs the associated artifacts to the local Maven cache, including Maven metadata generation. By default the install task is associated with the archives configuration. This configuration has by default only the default jar as an element. To learn more about installing to the local repository, see [Installing to the local repository](#1ksv4uv)

[Dependency management](#17dp8vu)

The Maven plugin does not define any dependency configurations.

[Convention properties](#3rdcrjn)

The Maven plugin defines the following convention properties:

mavenPomDir — File

The directory where the generated POMs are written to. *Default value*: *${project.buildDir}*/poms

conf2ScopeMappings — [Conf2ScopeMappingContainer](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/Conf2ScopeMappingContainer.html)

Instructions for mapping Gradle configurations to Maven scopes. See [Dependency mapping](#44sinio).

These properties are provided by a [MavenPluginConvention](http://docs.google.com/dsl/org.gradle.api.plugins.MavenPluginConvention.html) convention object.

[Convention methods](#26in1rg)

The maven plugin provides a factory method for creating a POM. This is useful if you need a POM without the context of uploading to a Maven repo.

[Example: Creating a standalone pom.](#2jxsxqh)

**build.gradle**

task writeNewPom {  
 doLast {  
 pom {  
 project {  
 inceptionYear '2008'  
 licenses {  
 license {  
 name 'The Apache Software License, Version 2.0'  
 url 'http://www.apache.org/licenses/LICENSE-2.0.txt'  
 distribution 'repo'  
 }  
 }  
 }  
 }.writeTo("$buildDir/newpom.xml")  
 }  
}

Amongst other things, Gradle supports the same builder syntax as polyglot Maven. To learn more about the Gradle Maven POM object, see [MavenPom](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/MavenPom.html). See also: [MavenPluginConvention](http://docs.google.com/dsl/org.gradle.api.plugins.MavenPluginConvention.html)

[Interacting with Maven repositories](#lnxbz9)

[Introduction](#z337ya)

With Gradle you can deploy to remote Maven repositories or install to your local Maven repository. This includes all Maven metadata manipulation and works also for Maven snapshots. In fact, Gradle’s deployment is 100 percent Maven compatible as we use the native Maven Ant tasks under the hood.

Deploying to a Maven repository is only half the fun if you don’t have a POM. Fortunately Gradle can generate this POM for you using the dependency information it has.

[Deploying to a Maven repository](#3j2qqm3)

Let’s assume your project produces just the default jar file. Now you want to deploy this jar file to a remote Maven repository.

[Example: Upload of file to remote Maven repository](#1y810tw)

**build.gradle**

apply plugin: 'maven'  
  
uploadArchives {  
 repositories {  
 mavenDeployer {  
 repository(url: "file://localhost/tmp/myRepo/")  
 }  
 }  
}

That is all. Calling the uploadArchives task will generate the POM and deploys the artifact and the POM to the specified repository.

There is more work to do if you need support for protocols other than file. In this case the native Maven code we delegate to needs additional libraries. Which libraries are needed depends on what protocol you plan to use. The available protocols and the corresponding libraries are listed in [Protocol JARs for Maven deployment](#4i7ojhp) (those libraries have transitive dependencies which have transitive dependencies).[[1](#2xcytpi)] For example, to use the ssh protocol you can do:

[Example: Upload of file via SSH](#1ci93xb)

**build.gradle**

configurations {  
 deployerJars  
}  
  
repositories {  
 mavenCentral()  
}  
  
dependencies {  
 deployerJars "org.apache.maven.wagon:wagon-ssh:2.2"  
}  
  
uploadArchives {  
 repositories.mavenDeployer {  
 configuration = configurations.deployerJars  
 repository(url: "scp://repos.mycompany.com/releases") {  
 authentication(userName: "me", password: "myPassword")  
 }  
 }  
}

There are many configuration options for the Maven deployer. The configuration is done via a Groovy builder. All the elements of this tree are Java beans. To configure the simple attributes you pass a map to the bean elements. To add bean elements to its parent, you use a closure. In the example above *repository* and *authentication* are such bean elements. [Configuration elements of Maven deployer](#3whwml4) lists the available bean elements and a link to the Javadoc of the corresponding class. In the Javadoc you can see the possible attributes you can set for a particular element.

In Maven you can define repositories and optionally snapshot repositories. If no snapshot repository is defined, releases and snapshots are both deployed to the repository element. Otherwise snapshots are deployed to the snapshotRepository element.

Table 1. Protocol jars for Maven deployment

| **Protocol** | **Library** |
| --- | --- |
| http | org.apache.maven.wagon:wagon-http:2.2 |
| ssh | org.apache.maven.wagon:wagon-ssh:2.2 |
| ssh-external | org.apache.maven.wagon:wagon-ssh-external:2.2 |
| ftp | org.apache.maven.wagon:wagon-ftp:2.2 |
| webdav | org.apache.maven.wagon:wagon-webdav:1.0-beta-2 |
| file | - |

Table 2. Configuration elements of the MavenDeployer

| **Element** | **Javadoc** |
| --- | --- |
| root | [MavenDeployer](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/MavenDeployer.html) |
| repository | [org.apache.maven.artifact.ant.RemoteRepository](http://www.docjar.com/docs/api/org/apache/maven/artifact/ant/RemoteRepository.html) |
| authentication | [org.apache.maven.artifact.ant.Authentication](http://www.docjar.com/docs/api/org/apache/maven/artifact/ant/Authentication.html) |
| releases | [org.apache.maven.artifact.ant.RepositoryPolicy](http://www.docjar.com/docs/api/org/apache/maven/artifact/ant/RepositoryPolicy.html) |
| snapshots | [org.apache.maven.artifact.ant.RepositoryPolicy](http://www.docjar.com/docs/api/org/apache/maven/artifact/ant/RepositoryPolicy.html) |
| proxy | [org.apache.maven.artifact.ant.Proxy](http://www.docjar.com/docs/api/org/apache/maven/artifact/ant/Proxy.html) |
| snapshotRepository | [org.apache.maven.artifact.ant.RemoteRepository](http://www.docjar.com/docs/api/org/apache/maven/artifact/ant/RemoteRepository.html) |

[Installing to the local repository](#1ksv4uv)

The Maven plugin adds an install task to your project. This task depends on all the archives task of the archives configuration. It installs those archives to your local Maven repository. If the default location for the local repository is redefined in a Maven settings.xml, this is considered by this task.

[Maven POM generation](#2bn6wsx)

When deploying an artifact to a Maven repository, Gradle automatically generates a POM for it. The groupId, artifactId, version and packaging elements used for the POM default to the values shown in the table below. The dependency elements are created from the project’s dependency declarations.

Table 3. Default Values for Maven POM generation

| **Maven Element** | **Default Value** |
| --- | --- |
| groupId | project.group |
| artifactId | uploadTask.repositories.mavenDeployer.pom.artifactId (if set) or archiveTask.baseName. |
| version | project.version |
| packaging | archiveTask.extension |

Here, uploadTask and archiveTask refer to the tasks used for uploading and generating the archive, respectively (for example uploadArchives and jar). archiveTask.baseName defaults to project.archivesBaseName which in turn defaults to project.name.

| **✨** | When you set the “archiveTask.baseName” property to a value other than the default, you’ll also have to set uploadTask.repositories.mavenDeployer.pom.artifactId to the same value. Otherwise, the project at hand may be referenced with the wrong artifact ID from generated POMs for other projects in the same build. |
| --- | --- |

Generated POMs can be found in <buildDir>/poms. They can be further customized via the [MavenPom](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/MavenPom.html) API. For example, you might want the artifact deployed to the Maven repository to have a different version or name than the artifact generated by Gradle. To customize these you can do:

[Example: Customization of pom](#qsh70q)

**build.gradle**

uploadArchives {  
 repositories {  
 mavenDeployer {  
 repository(url: "file://localhost/tmp/myRepo/")  
 pom.version = '1.0Maven'  
 pom.artifactId = 'myMavenName'  
 }  
 }  
}

To add additional content to the POM, the pom.project builder can be used. With this builder, any element listed in the [Maven POM reference](http://maven.apache.org/pom.html) can be added.

[Example: Builder style customization of pom](#3as4poj)

**build.gradle**

uploadArchives {  
 repositories {  
 mavenDeployer {  
 repository(url: "file://localhost/tmp/myRepo/")  
 pom.project {  
 licenses {  
 license {  
 name 'The Apache Software License, Version 2.0'  
 url 'http://www.apache.org/licenses/LICENSE-2.0.txt'  
 distribution 'repo'  
 }  
 }  
 }  
 }  
 }  
}

Note: groupId, artifactId, version, and packaging should always be set directly on the pom object.

[Example: Modifying auto-generated content](#1pxezwc)

**build.gradle**

def installer = install.repositories.mavenInstaller  
def deployer = uploadArchives.repositories.mavenDeployer  
  
[installer, deployer]\*.pom\*.whenConfigured {pom ->  
 pom.dependencies.find {dep -> dep.groupId == 'group3' && dep.artifactId == 'runtime' }.optional = true  
}

If you have more than one artifact to publish, things work a little bit differently. See [Multiple artifacts per project](#49x2ik5).

To customize the settings for the Maven installer (see [Installing to the local repository](#1ksv4uv)), you can do:

[Example: Customization of Maven installer](#2p2csry)

**build.gradle**

install {  
 repositories.mavenInstaller {  
 pom.version = '1.0Maven'  
 pom.artifactId = 'myName'  
 }  
}

[Multiple artifacts per project](#49x2ik5)

Maven can only deal with one artifact per project. This is reflected in the structure of the Maven POM. We think there are many situations where it makes sense to have more than one artifact per project. In such a case you need to generate multiple POMs. In such a case you have to explicitly declare each artifact you want to publish to a Maven repository. The [MavenDeployer](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/MavenDeployer.html) and the MavenInstaller both provide an API for this:

[Example: Generation of multiple poms](#147n2zr)

**build.gradle**

uploadArchives {  
 repositories {  
 mavenDeployer {  
 repository(url: "file://localhost/tmp/myRepo/")  
 addFilter('api') {artifact, file ->  
 artifact.name == 'api'  
 }  
 addFilter('service') {artifact, file ->  
 artifact.name == 'service'  
 }  
 pom('api').version = 'mySpecialMavenVersion'  
 }  
 }  
}

You need to declare a filter for each artifact you want to publish. This filter defines a boolean expression for which Gradle artifact it accepts. Each filter has a POM associated with it which you can configure. To learn more about this have a look at [PomFilterContainer](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/PomFilterContainer.html) and its associated classes.

[Dependency mapping](#44sinio)

The Maven plugin configures the default mapping between the Gradle configurations added by the Java and War plugin and the Maven scopes. Most of the time you don’t need to touch this and you can safely skip this section. The mapping works like the following. You can map a configuration to one and only one scope. Different configurations can be mapped to one or different scopes. You can also assign a priority to a particular configuration-to-scope mapping. Have a look at [Conf2ScopeMappingContainer](http://docs.google.com/javadoc/org/gradle/api/artifacts/maven/Conf2ScopeMappingContainer.html) to learn more. To access the mapping configuration you can say:

[Example: Accessing a mapping configuration](#3o7alnk)

**build.gradle**

task mappings {  
 doLast {  
 println conf2ScopeMappings.mappings  
 }  
}

Gradle exclude rules are converted to Maven excludes if possible. Such a conversion is possible if in the Gradle exclude rule the group as well as the module name is specified (as Maven needs both in contrast to Ivy). Per-configuration excludes are also included in the Maven POM, if they are convertible.

[1](#23ckvvd). It is planned for a future release to provide out-of-the-box support for this

Docs

* [User Manual](http://docs.google.com/userguide/userguide.html)
* [DSL Reference](http://docs.google.com/dsl/)
* [Release Notes](http://docs.google.com/release-notes.html)
* [Javadoc](http://docs.google.com/javadoc/)

News

* [Blog](https://blog.gradle.org/)
* [Newsletter](https://newsletter.gradle.com/)
* [Twitter](https://twitter.com/gradle)

Products

* [Build Scans](https://gradle.com/build-scans)
* [Build Cache](https://gradle.com/build-cache)
* [Enterprise Docs](https://gradle.com/enterprise/resources)

Get Help

* [Forums](https://discuss.gradle.org/c/help-discuss)
* [GitHub](https://github.com/gradle/)
* [Training](https://gradle.org/training/)
* [Services](https://gradle.org/services/)

Subscribe for important Gradle updates and news

Subscribe

By entering your email, you agree to our [Terms](https://gradle.org/terms/) and [Privacy Policy](https://gradle.org/privacy/), including receipt of emails. You can unsubscribe at any time.

© [Gradle Inc.](https://gradle.com) 2018 All rights reserved.

[Careers](https://gradle.com/careers) | [Privacy](https://gradle.org/privacy) | [Terms of Service](https://gradle.org/terms) | [Contact](https://gradle.org/contact/)