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/)

# Build Init Plugin

Contents

[Tasks](#4d34og8)

[What to set up](#2s8eyo1)

[Build init types](#17dp8vu)

| **✨** | The Build Init plugin is currently [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle). Please be aware that the DSL and other configuration may change in later Gradle versions. |
| --- | --- |

The Gradle Build Init plugin can be used to bootstrap the process of creating a new Gradle build. It supports creating brand new projects of different types as well as converting existing builds (e.g. An Apache Maven build) to be Gradle builds.

Gradle plugins typically need to be *applied* to a project before they can be used (see [Using plugins](http://docs.google.com/plugins.html#sec:using_plugins)). The Build Init plugin is an automatically applied plugin, which means you do not need to apply it explicitly. To use the plugin, simply execute the task named init where you would like to create the Gradle build. There is no need to create a “stub” build.gradle file in order to apply the plugin.

It also leverages the wrapper task to [generate the Gradle Wrapper files](http://docs.google.com/gradle_wrapper.html#sec:adding_wrapper) for the project.

[Tasks](#4d34og8)

The plugin adds the following tasks to the project:

init — [InitBuild](http://docs.google.com/dsl/org.gradle.buildinit.tasks.InitBuild.html)

*Depends on*: wrapper

Generates a Gradle project.

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

Generates Gradle wrapper files.

[What to set up](#2s8eyo1)

The init supports different build setup *types*. The type is specified by supplying a --type argument value. For example, to create a Java library project simply execute: gradle init --type java-library.

If a --type parameter is not supplied, Gradle will attempt to infer the type from the environment. For example, it will infer a type value of “pom” if it finds a pom.xml to convert to a Gradle build.

If the type could not be inferred, the type “basic” will be used.

The init plugin also supports generating build scripts using either the Gradle Groovy DSL or the Gradle Kotlin DSL. The build script DSL to use defaults to the Groovy DSL and is specified by supplying a --dsl argument value. For example, to create a Java library project with Kotlin DSL build scripts simply execute: gradle init --type java-library --dsl kotlin.

All build setup types include the setup of the Gradle Wrapper.

Note that the migration from Maven builds only supports the Groovy DSL for generated build scripts.

[Build init types](#17dp8vu)

| **✨** | As this plugin is currently [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle), only a few build init types are currently supported. More types will be added in future Gradle releases. |
| --- | --- |

[pom (Maven conversion)](#3rdcrjn)

The “pom” type can be used to convert an Apache Maven build to a Gradle build. This works by converting the POM to one or more Gradle files. It is only able to be used if there is a valid “pom.xml” file in the directory that the init task is invoked in or, if invoked via the “-p” [command line option](http://docs.google.com/command_line_interface.html#command_line_interface), in the specified project directory. This “pom” type will be automatically inferred if such a file exists.

The Maven conversion implementation was inspired by the [maven2gradle tool](https://github.com/jbaruch/maven2gradle) that was originally developed by Gradle community members.

The conversion process has the following features:

* Uses effective POM and effective settings (support for POM inheritance, dependency management, properties)
* Supports both single module and multimodule projects
* Supports custom module names (that differ from directory names)
* Generates general metadata - id, description and version
* Applies maven, java and war plugins (as needed)
* Supports packaging war projects as jars if needed
* Generates dependencies (both external and inter-module)
* Generates download repositories (inc. local Maven repository)
* Adjusts Java compiler settings
* Supports packaging of sources and tests
* Supports TestNG runner
* Generates global exclusions from Maven enforcer plugin settings

[java-application](#26in1rg)

The “java-application” build init type is not inferable. It must be explicitly specified.

It has the following features:

* Uses the “application” plugin to produce a command-line application implemented using Java
* Uses the “jcenter” dependency repository
* Uses [JUnit](http://junit.org) for testing
* Has directories in the conventional locations for source code
* Contains a sample class and unit test, if there are no existing source or test files

Alternative test framework can be specified by supplying a --test-framework argument value. To use a different test framework, execute one of the following commands:

* gradle init --type java-application --test-framework spock: Uses [Spock](http://code.google.com/p/spock/) for testing instead of JUnit
* gradle init --type java-application --test-framework testng: Uses [TestNG](http://testng.org/doc/index.html) for testing instead of JUnit

[java-library](#lnxbz9)

The “java-library” build init type is not inferable. It must be explicitly specified.

It has the following features:

* Uses the “java” plugin to produce a library Jar
* Uses the “jcenter” dependency repository
* Uses [JUnit](http://junit.org) for testing
* Has directories in the conventional locations for source code
* Contains a sample class and unit test, if there are no existing source or test files

Alternative test framework can be specified by supplying a --test-framework argument value. To use a different test framework, execute one of the following commands:

* gradle init --type java-library --test-framework spock: Uses [Spock](http://code.google.com/p/spock/) for testing instead of JUnit
* gradle init --type java-library --test-framework testng: Uses [TestNG](http://testng.org/doc/index.html) for testing instead of JUnit

[scala-library](#35nkun2)

The “scala-library” build init type is not inferable. It must be explicitly specified.

It has the following features:

* Uses the “scala” plugin to produce a library Jar
* Uses the “jcenter” dependency repository
* Uses Scala 2.10
* Uses [ScalaTest](http://www.scalatest.org) for testing
* Has directories in the conventional locations for source code
* Contains a sample scala class and an associated ScalaTest test suite, if there are no existing source or test files
* Uses the Zinc Scala compiler by default

[groovy-library](#1ksv4uv)

The “groovy-library” build init type is not inferable. It must be explicitly specified.

It has the following features:

* Uses the “groovy” plugin to produce a library Jar
* Uses the “jcenter” dependency repository
* Uses Groovy 2.x
* Uses [Spock testing framework](http://spockframework.org) for testing
* Has directories in the conventional locations for source code
* Contains a sample Groovy class and an associated Spock specification, if there are no existing source or test files

[groovy-application](#44sinio)

The “groovy-application” build init type is not inferable. It must be explicitly specified.

It has the following features:

* Uses the “groovy” plugin
* Uses the “application” plugin to produce a command-line application implemented using Groovy
* Uses the “jcenter” dependency repository
* Uses Groovy 2.x
* Uses [Spock testing framework](http://spockframework.org) for testing
* Has directories in the conventional locations for source code
* Contains a sample Groovy class and an associated Spock specification, if there are no existing source or test files

[basic](#2jxsxqh)

The “basic” build init type is useful for creating a fresh new Gradle project. It creates a sample build.gradle file, with comments and links to help get started.

This type is used when no type was explicitly specified, and no type could be inferred.

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/)