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/)

# Initialization Scripts

Contents

[Basic usage](#4d34og8)

[Using an init script](#2s8eyo1)

[Writing an init script](#17dp8vu)

[External dependencies for the init script](#3rdcrjn)

[Init script plugins](#26in1rg)

Gradle provides a powerful mechanism to allow customizing the build based on the current environment. This mechanism also supports tools that wish to integrate with Gradle.

Note that this is completely different from the “init” task provided by the “build-init” incubating plugin (see [Build Init Plugin](http://docs.google.com/build_init_plugin.html#build_init_plugin)).

[Basic usage](#4d34og8)

Initialization scripts (a.k.a. *init scripts*) are similar to other scripts in Gradle. These scripts, however, are run before the build starts. Here are several possible uses:

* Set up enterprise-wide configuration, such as where to find custom plugins.
* Set up properties based on the current environment, such as a developer’s machine vs. a continuous integration server.
* Supply personal information about the user that is required by the build, such as repository or database authentication credentials.
* Define machine specific details, such as where JDKs are installed.
* Register build listeners. External tools that wish to listen to Gradle events might find this useful.
* Register build loggers. You might wish to customize how Gradle logs the events that it generates.

One main limitation of init scripts is that they cannot access classes in the buildSrc project (see [Using buildSrc to extract imperative logic](http://docs.google.com/organizing_gradle_projects.html#sec:build_sources) for details of this feature).

[Using an init script](#2s8eyo1)

There are several ways to use an init script:

* Specify a file on the command line. The command line option is -I or --init-script followed by the path to the script. The command line option can appear more than once, each time adding another init script. The build will fail if any of the files specified on the command line does not exist.
* Put a file called init.gradle in the *USER\_HOME*/.gradle/ directory.
* Put a file that ends with .gradle in the *USER\_HOME*/.gradle/init.d/ directory.
* Put a file that ends with .gradle in the *GRADLE\_HOME*/init.d/ directory, in the Gradle distribution. This allows you to package up a custom Gradle distribution containing some custom build logic and plugins. You can combine this with the [Gradle wrapper](http://docs.google.com/gradle_wrapper.html#gradle_wrapper) as a way to make custom logic available to all builds in your enterprise.

If more than one init script is found they will all be executed, in the order specified above. Scripts in a given directory are executed in alphabetical order. This allows, for example, a tool to specify an init script on the command line and the user to put one in their home directory for defining the environment and both scripts will run when Gradle is executed.

[Writing an init script](#17dp8vu)

Similar to a Gradle build script, an init script is a Groovy script. Each init script has a [Gradle](http://docs.google.com/dsl/org.gradle.api.invocation.Gradle.html) instance associated with it. Any property reference and method call in the init script will delegate to this Gradle instance.

Each init script also implements the [Script](http://docs.google.com/dsl/org.gradle.api.Script.html) interface.

[Configuring projects from an init script](#lnxbz9)

You can use an init script to configure the projects in the build. This works in a similar way to configuring projects in a multi-project build. The following sample shows how to perform extra configuration from an init script *before* the projects are evaluated. This sample uses this feature to configure an extra repository to be used only for certain environments.

[Example: Using init script to perform extra configuration before projects are evaluated](#35nkun2)

**build.gradle**

repositories {  
 mavenCentral()  
}  
  
task showRepos {  
 doLast {  
 println "All repos:"  
 println repositories.collect { it.name }  
 }  
}

**init.gradle**

allprojects {  
 repositories {  
 mavenLocal()  
 }  
}

**Output of gradle --init-script init.gradle -q showRepos**

> gradle --init-script init.gradle -q showRepos  
All repos:  
[MavenLocal, MavenRepo]

[External dependencies for the init script](#3rdcrjn)

In [External dependencies for the build script](http://docs.google.com/tutorial_using_tasks.html#sec:build_script_external_dependencies) it was explained how to add external dependencies to a build script. Init scripts can also declare dependencies. You do this with the initscript() method, passing in a closure which declares the init script classpath.

[Example: Declaring external dependencies for an init script](#1ksv4uv)

**init.gradle**

initscript {  
 repositories {  
 mavenCentral()  
 }  
 dependencies {  
 classpath group: 'org.apache.commons', name: 'commons-math', version: '2.0'  
 }  
}

The closure passed to the initscript() method configures a [ScriptHandler](http://docs.google.com/javadoc/org/gradle/api/initialization/dsl/ScriptHandler.html) instance. You declare the init script classpath by adding dependencies to the classpath configuration. This is the same way you declare, for example, the Java compilation classpath. You can use any of the dependency types described in [Declaring Dependencies](http://docs.google.com/declaring_dependencies.html#declaring_dependencies), except project dependencies.

Having declared the init script classpath, you can use the classes in your init script as you would any other classes on the classpath. The following example adds to the previous example, and uses classes from the init script classpath.

[Example: An init script with external dependencies](#44sinio)

**init.gradle**

import org.apache.commons.math.fraction.Fraction  
  
initscript {  
 repositories {  
 mavenCentral()  
 }  
 dependencies {  
 classpath group: 'org.apache.commons', name: 'commons-math', version: '2.0'  
 }  
}  
  
println Fraction.ONE\_FIFTH.multiply(2)

**Output of** gradle --init-script init.gradle -q doNothing

> gradle --init-script init.gradle -q doNothing  
2 / 5

[Init script plugins](#26in1rg)

Similar to a Gradle build script or a Gradle settings file, plugins can be applied on init scripts.

[Example: Using plugins in init scripts](#2jxsxqh)

**init.gradle**

apply plugin:EnterpriseRepositoryPlugin  
  
class EnterpriseRepositoryPlugin implements Plugin<Gradle> {  
  
 private static String ENTERPRISE\_REPOSITORY\_URL = "https://repo.gradle.org/gradle/repo"  
  
 void apply(Gradle gradle) {  
 // ONLY USE ENTERPRISE REPO FOR DEPENDENCIES  
 gradle.allprojects{ project ->  
 project.repositories {  
  
 // Remove all repositories not pointing to the enterprise repository url  
 all { ArtifactRepository repo ->  
 if (!(repo instanceof MavenArtifactRepository) ||  
 repo.url.toString() != ENTERPRISE\_REPOSITORY\_URL) {  
 project.logger.lifecycle "Repository ${repo.url} removed. Only $ENTERPRISE\_REPOSITORY\_URL is allowed"  
 remove repo  
 }  
 }  
  
 // add the enterprise repository  
 maven {  
 name "STANDARD\_ENTERPRISE\_REPO"  
 url ENTERPRISE\_REPOSITORY\_URL  
 }  
 }  
 }  
 }  
}

**build.gradle**

repositories{  
 mavenCentral()  
}  
  
 task showRepositories {  
 doLast {  
 repositories.each {  
 println "repository: ${it.name} ('${it.url}')"  
 }  
 }  
}

**Output of gradle --init-script init.gradle -q showRepositories**

> gradle --init-script init.gradle -q showRepositories  
repository: STANDARD\_ENTERPRISE\_REPO ('https://repo.gradle.org/gradle/repo')

The plugin in the init script ensures that only a specified repository is used when running the build.

When applying plugins within the init script, Gradle instantiates the plugin and calls the plugin instance’s [Plugin.apply(T)](http://docs.google.com/javadoc/org/gradle/api/Plugin.html#apply-T-) method. The gradle object is passed as a parameter, which can be used to configure all aspects of a build. Of course, the applied plugin can be resolved as an external dependency as described in [External dependencies for the init script](#3rdcrjn)

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/)