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 Environment

Contents

[Gradle properties](#4d34og8)

[System properties](#2s8eyo1)

[Environment variables](#17dp8vu)

[Project properties](#3rdcrjn)

[Configuring JVM memory](#26in1rg)

[Configuring a task using project properties](#lnxbz9)

[Accessing the web through a HTTP proxy](#35nkun2)

Gradle provides multiple mechanisms for configuring behavior of Gradle itself and specific projects. The following is a reference for using these mechanisms.

When configuring Gradle behavior you can use these methods, listed in order of highest to lowest precedence (first one wins):

* [Command-line flags](http://docs.google.com/command_line_interface.html#command_line_interface) such as --build-cache. These have precedence over properties and environment variables.
* [System properties](#2s8eyo1) such as systemProp.http.proxyHost=somehost.org stored in a gradle.properties file.
* [Gradle properties](#4d34og8) such as org.gradle.caching=true that are typically stored in a gradle.properties file in a project root directory or GRADLE\_USER\_HOME environment variable.
* [Environment variables](#17dp8vu) such as GRADLE\_OPTS sourced by the environment that executes Gradle.

Aside from configuring the build environment, you can configure a given project build using [Project properties](#3rdcrjn) such as -PreleaseType=final.

[Gradle properties](#4d34og8)

Gradle provides several options that make it easy to configure the Java process that will be used to execute your build. While it’s possible to configure these in your local environment via GRADLE\_OPTS or JAVA\_OPTS, it is useful to store certain settings like JVM memory configuration and Java home location in version control so that an entire team can work with a consistent environment.

Setting up a consistent environment for your build is as simple as placing these settings into a gradle.properties file. The configuration is applied in following order (if an option is configured in multiple locations the *last one wins*):

* gradle.properties in project root directory.
* gradle.properties in GRADLE\_USER\_HOME directory.
* system properties, e.g. when -Dgradle.user.home is set on the command line.

The following properties can be used to configure the Gradle build environment:

org.gradle.caching=(true,false)

When set to true, Gradle will reuse task outputs from any previous build, when possible, resulting is much faster builds. Learn more about [using the build cache](http://docs.google.com/build_cache.html#build_cache).

org.gradle.caching.debug=(true,false)

When set to true, individual input property hashes and the build cache key for each task are logged on the console. Learn more about [task output caching](http://docs.google.com/build_cache.html#sec:task_output_caching).

org.gradle.configureondemand=(true,false)

Enables incubating [configuration on demand](http://docs.google.com/multi_project_builds.html#sec:configuration_on_demand), where Gradle will attempt to configure only necessary projects.

org.gradle.console=(auto,plain,rich,verbose)

Customize console output coloring or verbosity. Default depends on how Gradle is invoked. See [command-line logging](http://docs.google.com/command_line_interface.html#sec:command_line_logging) for additional details.

org.gradle.daemon=(true,false)

When set to true the [Gradle Daemon](http://docs.google.com/gradle_daemon.html#gradle_daemon) is used to run the build. Default is true.

org.gradle.daemon.idletimeout=(# of idle millis)

Gradle Daemon will terminate itself after specified number of idle milliseconds. Default is 10800000 (3 hours).

org.gradle.debug=(true,false)

When set to true, Gradle will run the build with remote debugging enabled, listening on port 5005. Note that this is the equivalent of adding -agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=5005 to the JVM command line and will suspend the virtual machine until a debugger is attached. Default is false.

org.gradle.java.home=(path to JDK home)

Specifies the Java home for the Gradle build process. The value can be set to either a jdk or jre location, however, depending on what your build does, using a JDK is safer. A reasonable default is used if the setting is unspecified.

org.gradle.jvmargs=(JVM arguments)

Specifies the JVM arguments used for the Gradle Daemon. The setting is particularly useful for [configuring JVM memory settings](#26in1rg) for build performance.

org.gradle.logging.level=(quiet,warn,lifecycle,info,debug)

When set to quiet, warn, lifecycle, info, or debug, Gradle will use this log level. The values are not case sensitive. The lifecycle level is the default. See [Choosing a log level](http://docs.google.com/logging.html#sec:choosing_a_log_level).

org.gradle.parallel=(true,false)

When configured, Gradle will fork up to org.gradle.workers.max JVMs to execute projects in parallel. To learn more about parallel task execution, see [the Gradle performance guide](https://guides.gradle.org/performance/#parallel_execution).

org.gradle.warning.mode=(all,none,summary)

When set to all, summary or none, Gradle will use different warning type display. See [Command-line logging options](http://docs.google.com/command_line_interface.html#sec:command_line_logging) for details.

org.gradle.workers.max=(max # of worker processes)

When configured, Gradle will use a maximum of the given number of workers. Default is number of CPU processors. See also [performance command-line options](http://docs.google.com/command_line_interface.html#sec:command_line_performance).

The following example demonstrates usage of various properties.

[Example: Setting properties with a gradle.properties file](#1ksv4uv)

**gradle.properties**

gradlePropertiesProp=gradlePropertiesValue  
sysProp=shouldBeOverWrittenBySysProp  
systemProp.system=systemValue

**build.gradle**

task printProps {  
 doLast {  
 println commandLineProjectProp  
 println gradlePropertiesProp  
 println systemProjectProp  
 println System.properties['system']  
 }  
}

**Output of gradle -q -PcommandLineProjectProp=commandLineProjectPropValue -Dorg.gradle.project.systemProjectProp=systemPropertyValue printProps**

> gradle -q -PcommandLineProjectProp=commandLineProjectPropValue -Dorg.gradle.project.systemProjectProp=systemPropertyValue printProps  
commandLineProjectPropValue  
gradlePropertiesValue  
systemPropertyValue  
systemValue

[System properties](#2s8eyo1)

Using the -D command-line option, you can pass a system property to the JVM which runs Gradle. The -D option of the gradle command has the same effect as the -D option of the java command.

You can also set system properties in gradle.properties files with the prefix systemProp.

[Example: Specifying system properties in gradle.properties](#44sinio)

systemProp.gradle.wrapperUser=myuser  
systemProp.gradle.wrapperPassword=mypassword

The following system properties are available. Note that command-line options take precedence over system properties.

gradle.wrapperUser=(myuser)

Specify user name to download Gradle distributions from servers using HTTP Basic Authentication. Learn more in [Authenticated wrapper downloads](http://docs.google.com/gradle_wrapper.html#sec:authenticated_download).

gradle.wrapperPassword=(mypassword)

Specify password for downloading a Gradle distribution using the Gradle wrapper.

gradle.user.home=(path to directory)

Specify the Gradle user home directory.

In a multi project build, “systemProp.” properties set in any project except the root will be ignored. That is, only the root project’s gradle.properties file will be checked for properties that begin with the “systemProp.” prefix.

[Environment variables](#17dp8vu)

The following environment variables are available for the gradle command. Note that command-line options and system properties take precedence over environment variables.

GRADLE\_OPTS

Specifies [command-line arguments](http://docs.google.com/command_line_interface.html#command_line_interface) to use when starting the Gradle client. This can be useful for setting the properties to use when running Gradle.

GRADLE\_USER\_HOME

Specifies the Gradle user home directory (which defaults to $USER\_HOME/.gradle if not set).

JAVA\_HOME

Specifies the JDK installation directory to use.

[Project properties](#3rdcrjn)

You can add properties directly to your [Project](http://docs.google.com/dsl/org.gradle.api.Project.html) object via the -P command line option.

Gradle can also set project properties when it sees specially-named system properties or environment variables. If the environment variable name looks like ORG\_GRADLE\_PROJECT*\_prop*=somevalue, then Gradle will set a prop property on your project object, with the value of somevalue. Gradle also supports this for system properties, but with a different naming pattern, which looks like org.gradle.project.*prop*. Both of the following will set the foo property on your Project object to "bar".

[Example: Setting a project property via gradle.properties](#2jxsxqh)

org.gradle.project.foo=bar

[Example: Setting a project property via environment variable](#z337ya)

ORG\_GRADLE\_PROJECT\_foo=bar

| **✨** | The properties file in the user’s home directory has precedence over property files in the project directories. |
| --- | --- |

This feature is very useful when you don’t have admin rights to a continuous integration server and you need to set property values that should not be easily visible. Since you cannot use the -P option in that scenario, nor change the system-level configuration files, the correct strategy is to change the configuration of your continuous integration build job, adding an environment variable setting that matches an expected pattern. This won’t be visible to normal users on the system.

You can access a project property in your build script simply by using its name as you would use a variable.

| **✨** | If a project property is referenced but does not exist, an exception will be thrown and the build will fail.  You should check for existence of optional project properties before you access them using the [Project.hasProperty(java.lang.String)](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:hasProperty(java.lang.String)) method. |
| --- | --- |

[Configuring JVM memory](#26in1rg)

Gradle defaults to 1024 megabytes maximum heap per JVM process (-Xmx1024m), however, that may be too much or too little depending on the size of your project. There are many JVM options (this [blog post on Java performance tuning](https://dzone.com/articles/java-performance-tuning) and [this reference](http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html) may be helpful).

You can adjust JVM options for Gradle in the following ways:

The JAVA\_OPTS environment variable is used for the Gradle client, but not forked JVMs.

[Example: Changing JVM settings for Gradle client JVM](#3j2qqm3)

JAVA\_OPTS="-Xmx2g -XX:MaxPermSize=256m -XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8"

You need to use the org.gradle.jvmargs Gradle property to configure JVM settings for the [Gradle Daemon](http://docs.google.com/gradle_daemon.html#gradle_daemon).

[Example: Changing JVM settings for forked Gradle JVMs](#1y810tw)

org.gradle.jvmargs=-Xmx2g -XX:MaxPermSize=256m -XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8

| **✨** | Many settings (like the Java version and maximum heap size) can only be specified when launching a new JVM for the build process. This means that Gradle must launch a separate JVM process to execute the build after parsing the various gradle.properties files.  When running with the [Gradle Daemon](http://docs.google.com/gradle_daemon.html#gradle_daemon), a JVM with the correct parameters is started once and reused for each daemon build execution. When Gradle is executed without the daemon, then a new JVM must be launched for every build execution, unless the JVM launched by the Gradle start script happens to have the same parameters. |
| --- | --- |

Certain tasks in Gradle also fork additional JVM processes, like the test task when using [Test.setMaxParallelForks(int)](http://docs.google.com/javadoc/org/gradle/api/tasks/testing/Test.html#setMaxParallelForks-int-) for JUnit or TestNG tests. You must configure these through the tasks themselves.

[Example: Set Java compile options for](#4i7ojhp) [JavaCompile](http://docs.google.com/dsl/org.gradle.api.tasks.compile.JavaCompile.html) tasks

**build.gradle**

apply plugin: "java"  
  
tasks.withType(JavaCompile) {  
 options.compilerArgs += ["-Xdoclint:none", "-Xlint:none", "-nowarn"]  
}

See other examples in the [Test](http://docs.google.com/dsl/org.gradle.api.tasks.testing.Test.html) API documentation and [test execution in the Java plugin reference](http://docs.google.com/java_testing.html#sec:test_execution).

[Build scans](https://scans.gradle.com) will tell you information about the JVM that executed the build when you use the --scan option.



[Configuring a task using project properties](#lnxbz9)

It’s possible to change the behavior of a task based on project properties specified at invocation time.

Suppose you’d like to ensure release builds are only triggered by CI. A simple way to handle this is through an isCI project property.

[Example: Prevent releasing outside of CI](#2xcytpi)

**build.gradle**

task performRelease {  
 doLast {  
 if (project.hasProperty("isCI")) {  
 println("Performing release actions")  
 } else {  
 throw new InvalidUserDataException("Cannot perform release outside of CI")  
 }  
 }  
}

**Output of** gradle performRelease -PisCI=true --quiet

> gradle performRelease -PisCI=true --quiet  
Performing release actions

[Accessing the web through a HTTP proxy](#35nkun2)

Configuring an HTTP or HTTPS proxy (for downloading dependencies, for example) is done via standard JVM system properties. These properties can be set directly in the build script; for example, setting the HTTP proxy host would be done with System.setProperty('http.proxyHost', 'www.somehost.org'). Alternatively, the properties can be [specified in gradle.properties](#4d34og8).

**Configuring an HTTP proxy using gradle.properties**

systemProp.http.proxyHost=www.somehost.org  
systemProp.http.proxyPort=8080  
systemProp.http.proxyUser=userid  
systemProp.http.proxyPassword=password  
systemProp.http.nonProxyHosts=\*.nonproxyrepos.com|localhost

There are separate settings for HTTPS.

**Configuring an HTTPS proxy using gradle.properties**

systemProp.https.proxyHost=www.somehost.org  
systemProp.https.proxyPort=8080  
systemProp.https.proxyUser=userid  
systemProp.https.proxyPassword=password  
systemProp.https.nonProxyHosts=\*.nonproxyrepos.com|localhost

You may need to set other properties to access other networks. Here are 2 references that may be helpful:

* [ProxySetup.java in the Ant codebase](https://git-wip-us.apache.org/repos/asf?p=ant.git;a%3Dblob;f%3Dsrc/main/org/apache/tools/ant/util/ProxySetup.java;hb%3DHEAD)
* [JDK 7 Networking Properties](http://download.oracle.com/javase/7/docs/technotes/guides/net/properties.html)

[NTLM Authentication](#1ci93xb)

If your proxy requires NTLM authentication, you may need to provide the authentication domain as well as the username and password. There are 2 ways that you can provide the domain for authenticating to a NTLM proxy:

* Set the http.proxyUser system property to a value like *domain*/*username*.
* Provide the authentication domain via the http.auth.ntlm.domain system property.

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/)