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/)

# Inspecting Dependencies

Contents

[Listing dependencies in a project](#4d34og8)

[Identifying which dependency version was selected and why](#2s8eyo1)

[Justifying dependency declarations with custom reasons](#17dp8vu)

Gradle provides sufficient tooling to navigate large dependency graphs and mitigate situations that can lead to [dependency hell](https://en.wikipedia.org/wiki/Dependency_hell). Users can chose to render the full graph of dependencies as well as identify the selection reason and origin for a dependency. The origin of a dependency can be a declared dependency in the build script or a transitive dependency in graph plus their corresponding configuration. Gradle offers both capabilities through visual representation via build scans and as command line tooling.

[Listing dependencies in a project](#4d34og8)

A project can declare one or more dependencies. Gradle can visualize the whole dependency tree for every [configuration](http://docs.google.com/managing_dependency_configurations.html#sub:what_is_a_configuration) available in the project.

Rendering the dependency tree is particularly useful if you’d like to identify which dependencies have been resolved at runtime. It also provides you with information about any dependency conflict resolution that occurred in the process and clearly indicates the selected version. The dependency report always contains declared and transitive dependencies.

Let’s say you’d want to create tasks for your project that use the [JGit library](https://www.eclipse.org/jgit/) to execute SCM operations e.g. to model a release process. You can declare dependencies for any external tooling with the help of a [custom configuration](http://docs.google.com/managing_dependency_configurations.html#managing_dependency_configurations) so that it doesn’t doesn’t pollute other contexts like the compilation classpath for your production source code.

[Example: Declaring the JGit dependency with a custom configuration](#3rdcrjn)

**build.gradle**

repositories {  
 jcenter()  
}  
  
configurations {  
 scm  
}  
  
dependencies {  
 scm 'org.eclipse.jgit:org.eclipse.jgit:4.9.2.201712150930-r'  
}

A [build scan](https://scans.gradle.com/) can visualize dependencies as a navigable, searchable tree. Additional context information can be rendered by clicking on a specific dependency in the graph.



*Figure 1. Dependency tree in a build scan*

Every Gradle project provides the task dependencies to render the so-called *dependency report* from the command line. By default the dependency report renders dependencies for all configurations. To pair down on the information provide the optional parameter --configuration.

[Example: Rendering the dependency report for a custom configuration](#26in1rg)

**Output of** gradle -q dependencies --configuration scm

> gradle -q dependencies --configuration scm  
  
------------------------------------------------------------  
Root project  
------------------------------------------------------------  
  
scm  
\--- org.eclipse.jgit:org.eclipse.jgit:4.9.2.201712150930-r  
 +--- com.jcraft:jsch:0.1.54  
 +--- com.googlecode.javaewah:JavaEWAH:1.1.6  
 +--- org.apache.httpcomponents:httpclient:4.3.6  
 | +--- org.apache.httpcomponents:httpcore:4.3.3  
 | +--- commons-logging:commons-logging:1.1.3  
 | \--- commons-codec:commons-codec:1.6  
 \--- org.slf4j:slf4j-api:1.7.2  
  
A web-based, searchable dependency report is available by adding the --scan option.

The dependencies report provides detailed information about the dependencies available in the graph. Any dependency that could not be resolved is marked with FAILED in red color. Dependencies with the same coordinates that can occur multiple times in the graph are omitted and indicated by an asterisk. Dependencies that had to undergo conflict resolution render the requested and selected version separated by a right arrow character.

[Identifying which dependency version was selected and why](#2s8eyo1)

Large software projects inevitably deal with an increased number of dependencies either through direct or transitive dependencies. The [dependencies report](#4d34og8) provides you with the raw list of dependencies but does not explain *why* they have been selected or *which* dependency is responsible for pulling them into the graph.

Let’s have a look at a concrete example. A project may request two different versions of the same dependency either as direct or transitive dependency. Gradle applies version conflict resolution to ensure that only one version of the dependency exists in the dependency graph. In this example the conflicting dependency is represented by commons-codec:commons-codec.

[Example: Declaring the JGit dependency and a conflicting dependency](#lnxbz9)

**build.gradle**

repositories {  
 jcenter()  
}  
  
configurations {  
 scm  
}  
  
dependencies {  
 scm 'org.eclipse.jgit:org.eclipse.jgit:4.9.2.201712150930-r'  
 scm 'commons-codec:commons-codec:1.7'  
}

The dependency tree in a [build scan](https://scans.gradle.com/) renders the selection reason (conflict resolution) as well as the origin of a dependency if you click on a dependency and select the "Required By" tab.



*Figure 2. Dependency insight capabilities in a build scan*

Every Gradle project provides the task dependencyInsight to render the so-called *dependency insight report* from the command line. Given a dependency in the dependency graph you can identify the selection reason and track down the origin of the dependency selection. You can think of the dependency insight report as the inverse representation of the dependency report for a given dependency. When executing the task you have to provide the mandatory parameter --dependency to specify the coordinates of the dependency under inspection. The parameters --configuration and --singlepath are optional but help with filtering the output.

[Example: Using the dependency insight report for a given dependency](#35nkun2)

**Output of** gradle -q dependencyInsight --dependency commons-codec --configuration scm

> gradle -q dependencyInsight --dependency commons-codec --configuration scm  
commons-codec:commons-codec:1.7  
 variant "default+runtime" [  
 org.gradle.status = release (not requested)  
 ]  
 Selection reasons:  
 - Was requested  
 - By conflict resolution : between versions 1.7 and 1.6  
  
commons-codec:commons-codec:1.7  
\--- scm  
  
commons-codec:commons-codec:1.6 -> 1.7  
\--- org.apache.httpcomponents:httpclient:4.3.6  
 \--- org.eclipse.jgit:org.eclipse.jgit:4.9.2.201712150930-r  
 \--- scm  
  
A web-based, searchable dependency report is available by adding the --scan option.

[Justifying dependency declarations with custom reasons](#17dp8vu)

When you declare a [dependency](http://docs.google.com/declaring_dependencies.html#declaring_dependencies) or a [dependency constraint](http://docs.google.com/managing_transitive_dependencies.html#sec:dependency_constraints), you can provide a custom reason for the declaration. This makes the dependency declarations in your build script and the dependency insight report easier to interpret.

[Example: Giving a reason for choosing a certain module version in a dependency declaration](#1ksv4uv)

**build.gradle**

apply plugin: 'java-library'  
  
repositories {  
 jcenter()  
}  
  
dependencies {  
 implementation('org.ow2.asm:asm:6.0') {  
 because 'we require a JDK 9 compatible bytecode generator'  
 }  
}

[Example: Using the dependency insight report with custom reasons](#44sinio)

**Output of** gradle -q dependencyInsight --dependency asm

> gradle -q dependencyInsight --dependency asm  
org.ow2.asm:asm:6.0  
 variant "compile" [  
 org.gradle.status = release (not requested)  
 org.gradle.usage = java-api  
 ]  
 Selection reasons:  
 - Was requested : we require a JDK 9 compatible bytecode generator  
  
org.ow2.asm:asm:6.0  
\--- compileClasspath  
  
A web-based, searchable dependency report is available by adding the --scan option.

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/)