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/)

# Troubleshooting Dependency Resolution

Contents

[Resolving version conflicts](#4d34og8)

[Using dynamic versions and changing modules](#2s8eyo1)

[Controlling dependency caching programmatically](#17dp8vu)

[Controlling dependency caching from the command line](#3rdcrjn)

[Locking dependency versions](#26in1rg)

[Versioning of file dependencies](#lnxbz9)

Managing dependencies in a project can be challenging. This chapter describes techniques for troubleshooting issues you might encounter in your project as well as best practices for avoiding common problems.

[Resolving version conflicts](#4d34og8)

Gradle resolves version conflicts by picking the highest version of a module. [Build scans](https://scans.gradle.com/get-started) and the [dependency insight report](http://docs.google.com/inspecting_dependencies.html#sec:identifying_reason_dependency_selection) are immensely helpful in identifying why a specific version was selected. If the resolution result is not satisfying (e.g. the selected version of a module is too high) or it fails (because you configured [ResolutionStrategy.failOnVersionConflict()](http://docs.google.com/dsl/org.gradle.api.artifacts.ResolutionStrategy.html#org.gradle.api.artifacts.ResolutionStrategy:failOnVersionConflict())) you have the following possibilities to fix it.

* Configuring any dependency (transitive or not) as *forced*. This approach is useful if the dependency in conflict is a transitive dependency. See [Enforcing a particular dependency version](http://docs.google.com/managing_transitive_dependencies.html#sec:enforcing_dependency_version) for examples.
* Configuring dependency resolution to *prefer modules that are part of your build* (transitive or not). This approach is useful if your build contains custom forks of modules (as part of [multi-project builds](http://docs.google.com/multi_project_builds.html#multi_project_builds) or as include in [composite builds](http://docs.google.com/composite_builds.html#composite_builds)). See [ResolutionStrategy.preferProjectModules()](http://docs.google.com/dsl/org.gradle.api.artifacts.ResolutionStrategy.html#org.gradle.api.artifacts.ResolutionStrategy:preferProjectModules()) for more information.
* Using [dependency resolve rules](http://docs.google.com/customizing_dependency_resolution_behavior.html#sec:dependency_resolve_rules) for fine-grained control over the version selected for a particular dependency.

[Using dynamic versions and changing modules](#2s8eyo1)

There are many situations when you want to use the latest version of a particular module dependency, or the latest in a range of versions. This can be a requirement during development, or you may be developing a library that is designed to work with a range of dependency versions. You can easily depend on these constantly changing dependencies by using a *dynamic version*. A [dynamic version](http://docs.google.com/declaring_dependencies.html#sub:declaring_dependency_with_dynamic_version) can be either a version range (e.g. 2.+) or it can be a placeholder for the latest version available e.g. latest.integration.

Alternatively, the module you request can change over time even for the same version, a so-called [changing version](http://docs.google.com/declaring_dependencies.html#sub:declaring_dependency_with_changing_version). An example of this type of *changing module* is a Maven SNAPSHOT module, which always points at the latest artifact published. In other words, a standard Maven snapshot is a module that is continually evolving, it is a "changing module".

| **✨** | Using dynamic versions and changing modules can lead to unreproducible builds. As new versions of a particular module are published, its API may become incompatible with your source code. Use this feature with caution! |
| --- | --- |

By default, Gradle caches dynamic versions and changing modules for 24 hours. During that time frame Gradle does not contact any of the declared, remote repositories for new versions. If you want Gradle to check the remote repository more frequently or with every execution of your build, then you will need to change the time to live (TTL) threshold.

| **✨** | Using a short TTL threshold for dynamic or changing versions may result in longer build times due to the increased number of HTTP(s) calls. |
| --- | --- |

You can override the default cache modes using [command line options](#3rdcrjn). You can also [change the cache expiry times in your build programmatically](#17dp8vu) using the resolution strategy.

[Controlling dependency caching programmatically](#17dp8vu)

You can fine-tune certain aspects of caching programmatically using the [ResolutionStrategy](http://docs.google.com/javadoc/org/gradle/api/artifacts/ResolutionStrategy.html) for a configuration. The programmatic approach is useful if you would like to change the settings permanently.

By default, Gradle caches dynamic versions for 24 hours. To change how long Gradle will cache the resolved version for a dynamic version, use:

[Example: Dynamic version cache control](#35nkun2)

**build.gradle**

configurations.all {  
 resolutionStrategy.cacheDynamicVersionsFor 10, 'minutes'  
}

By default, Gradle caches changing modules for 24 hours. To change how long Gradle will cache the meta-data and artifacts for a changing module, use:

[Example: Changing module cache control](#1ksv4uv)

**build.gradle**

configurations.all {  
 resolutionStrategy.cacheChangingModulesFor 4, 'hours'  
}

[Controlling dependency caching from the command line](#3rdcrjn)

You can control the behavior of dependency caching for a distinct build invocation from the command line. Command line options are helpful for making a selective, ad-hoc choice for a single execution of the build.

[Avoiding network access with offline mode](#44sinio)

The --offline command line switch tells Gradle to always use dependency modules from the cache, regardless if they are due to be checked again. When running with offline, Gradle will never attempt to access the network to perform dependency resolution. If required modules are not present in the dependency cache, build execution will fail.

[Forcing all dependencies to be re-resolved](#2jxsxqh)

At times, the Gradle Dependency Cache can become out of sync with the actual state of the configured repositories. Perhaps a repository was initially misconfigured, or perhaps a "non-changing" module was published incorrectly. To refresh all dependencies in the dependency cache, use the --refresh-dependencies option on the command line.

The --refresh-dependencies option tells Gradle to ignore all cached entries for resolved modules and artifacts. A fresh resolve will be performed against all configured repositories, with dynamic versions recalculated, modules refreshed, and artifacts downloaded. However, where possible Gradle will check if the previously downloaded artifacts are valid before downloading again. This is done by comparing published SHA1 values in the repository with the SHA1 values for existing downloaded artifacts.

[Locking dependency versions](#26in1rg)

The use of [dynamic dependencies](http://docs.google.com/declaring_dependencies.html#sub:declaring_dependency_with_dynamic_version) in a build is convenient. The user does not need to know the latest version of a dependency and Gradle automatically uses new versions once they are published. However, dynamic dependencies make builds non-reproducible, as they can resolve to a different version at a later point in time. This makes it hard to reproduce old builds when debugging a problem. It can also disrupt development if a new, but incompatible version is selected. In the best case the CI build catches the problem and someone needs to investigate. In the worst case, the problem makes it to production unnoticed.

Gradle offers [dependency locking](http://docs.google.com/dependency_locking.html#dependency_locking) to solve this problem. The user can run a build asking to persist the resolved versions for every module dependency. This file is then checked in and the versions in it are used on all subsequent runs until the lock is updated or removed again.

[Versioning of file dependencies](#lnxbz9)

Legacy projects sometimes prefer to consume [file dependencies](http://docs.google.com/declaring_dependencies.html#sec:declaring_file_dependency) instead of [module dependencies](http://docs.google.com/declaring_dependencies.html#sec:declaring_dependency_to_module). File dependencies can point to any file in the filesystem and do not need to adhere a specific naming convention. It is recommended to clearly express the intention and a concrete version for file dependencies. File dependencies are not considered by Gradle’s [version conflict resolution](http://docs.google.com/introduction_dependency_management.html#sec:dependency_resolution). Therefore, it is extremely important to assign a version to the file name to indicate the distinct set of changes shipped with it. For example commons-beanutils-1.3.jar lets you track the changes of the library by the release notes.

As a result, the dependencies of the project are easier to maintain and organize. It’s much easier to uncover potential API incompatibilities by the assigned version.

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/)