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/)

# Dependency Locking

Contents

[Enabling locking on configurations](#4d34og8)

[Generating and updating dependency locks](#2s8eyo1)

[Lock state location and format](#17dp8vu)

[Running a build with lock state present](#3rdcrjn)

[Selectively updating lock state entries](#26in1rg)

[Disabling dependency locking](#lnxbz9)

[Locking limitations](#35nkun2)

[Nebula locking plugin](#1ksv4uv)

Use of dynamic dependency versions (e.g. 1.+ or [1.0,2.0)) makes builds non-deterministic. This causes builds to break without any obvious change, and worse, can be caused by a transitive dependency that the build author has no control over.

To achieve [reproducible builds](https://reproducible-builds.org/), it is necessary to *lock* versions of dependencies and transitive dependencies such that a build with the same inputs will always resolve the same module versions. This is called *dependency locking*.

It enables, amongst others, the following scenarios:

* Companies dealing with multi repositories no longer need to rely on -SNAPSHOT or changing dependencies, which sometimes result in cascading failures when a dependency introduces a bug or incompatibility. Now dependencies can be declared against major or minor version range, enabling to test with the latest versions on CI while leveraging locking for stable developer builds.
* Teams that want to always use the latest of their dependencies can use dynamic versions, locking their dependencies only for releases. The release tag will contain the lock states, allowing that build to be fully reproducible when bug fixes need to be developed.

Locking is enabled per [dependency configuration](http://docs.google.com/managing_dependency_configurations.html#managing_dependency_configurations). Once enabled, you must create an initial lock state. It will cause Gradle to verify that resolution results do not change, resulting in the same selected dependencies even if newer versions are produced. Modifications to your build that would impact the resolved set of dependencies will cause it to fail. This makes sure that changes, either in published dependencies or build definitions, do not alter resolution without adapting the lock state.

| **✨** | Dependency locking makes sense only with [dynamic versions](http://docs.google.com/declaring_dependencies.html#sub:declaring_dependency_with_dynamic_version). It will have no impact on [changing versions](http://docs.google.com/declaring_dependencies.html#sub:declaring_dependency_with_changing_version) (like -SNAPSHOT) whose coordinates remain the same, though the content may change. Gradle will even emit a warning when persisting lock state and changing dependencies are present in the resolution result. |
| --- | --- |

[Enabling locking on configurations](#4d34og8)

Locking of a configuration happens through the [ResolutionStrategy](http://docs.google.com/dsl/org.gradle.api.artifacts.ResolutionStrategy.html):

[Example: Locking a specific configuration](#44sinio)

**build.gradle**

configurations {  
 compileClasspath {  
 resolutionStrategy.activateDependencyLocking()  
 }  
}

Or the following, as a way to lock all configurations:

[Example: Locking all configurations](#2jxsxqh)

**build.gradle**

dependencyLocking {  
 lockAllConfigurations()  
}

| **✨** | Only configurations that can be resolved will have lock state attached to them. Applying locking on non resolvable-configurations is simply a no-op. |
| --- | --- |

[Generating and updating dependency locks](#2s8eyo1)

In order to generate or update lock state, you specify the --write-locks command line argument in addition to the normal tasks that would trigger configurations to be resolved. This will cause the creation of lock state for each resolved configuration in that build execution. Note that if lock state existed previously, it is overwritten.

[Lock all configurations in one build execution](#z337ya)

When locking multiple configurations, you may want to lock them all at once, during a single build execution.

For this, you have two options:

* Run gradle dependencies --write-locks. This will effectively lock all resolvable configurations that have locking enabled. Note that in a multi project setup, dependencies only is executed on *one* project, the root one in this case.
* Declare a custom task that will resolve all configurations

[Example: Resolving all configurations](#3j2qqm3)

**build.gradle**

task resolveAndLockAll {  
 doFirst {  
 assert gradle.startParameter.writeDependencyLocks  
 }  
 doLast {  
 configurations.findAll {  
 // Add any custom filtering on the configurations to be resolved  
 it.canBeResolved  
 }.each { it.resolve() }  
 }  
}

That second option, with proper choosing of configurations, can be the only option in the native world, where not all configurations can be resolved on a single platform.

[Lock state location and format](#17dp8vu)

Lock state will be preserved in a file located in the folder gradle/dependency-locks inside the project or subproject directory. Each file is named by the configuration it locks and has the lockfile extension.

The content of the file is a module notation per line, with a header giving some context. Module notations are ordered alphabetically, to ease diffs.

[Example: Lockfile content](#1y810tw)

**gradle/dependency-locks/compileClasspath.lockfile**

# This is a Gradle generated file for dependency locking.  
# Manual edits can break the build and are not advised.  
# This file is expected to be part of source control.  
org.springframework:spring-beans:5.0.5.RELEASE  
org.springframework:spring-core:5.0.5.RELEASE  
org.springframework:spring-jcl:5.0.5.RELEASE

which matches the following dependency declaration:

[Example: Dynamic dependency declaration](#4i7ojhp)

**build.gradle**

dependencies {  
 implementation 'org.springframework:spring-beans:[5.0,6.0)'  
}

[Running a build with lock state present](#3rdcrjn)

The moment a build needs to resolve a configuration that has locking enabled and it finds a matching lock state, it will use it to verify that the given configuration still resolves the same versions.

A successful build indicates that the same dependencies are used as stored in the lock state, regardless if new versions matching the dynamic selector have been produced.

The complete validation is as follows:

* Existing entries in the lock state must be matched in the build
  + A version mismatch or missing resolved module causes a build failure
* Resolution result must not contain extra dependencies compared to the lock state

[Selectively updating lock state entries](#26in1rg)

In order to update only specific modules of a configuration, you can use the --update-locks command line flag. It takes a comma (,) separated list of module notations. In this mode, the existing lock state is still used as input to resolution, filtering out the modules targeted by the update.

❯ gradle classes --update-locks org.apache.commons:commons-lang3,org.slf4j:slf4j-api

Wildcards, indicated with \*, can be used in the group or module name. They can be the only character or appear at the end of the group or module respectively. The following wildcard notation examples are valid:

* org.apache.commons:\*: will let all modules belonging to group org.apache.commons update
* \*:guava: will let all modules named guava, whatever their group, update
* org.springframework.spring\*:spring\*: will let all modules having their group starting with org.springframework.spring and name starting with spring update

| **✨** | The resolution may cause other module versions to update, as dictated by the Gradle resolution rules. |
| --- | --- |

[Disabling dependency locking](#lnxbz9)

1. Make sure that the configuration for which you no longer want locking is not configured with locking.
2. Remove the file matching the configurations where you no longer want locking.

If you only perform the second step above, then locking will effectively no longer be applied. However, if that configuration happens to be resolved in the future at a time where lock state is persisted, it will once again be locked.

[Locking limitations](#35nkun2)

* It is currently not possible to lock the [classpath configuration](http://docs.google.com/plugins.html#sec:applying_plugins_buildscript) used for script plugins.
* Locking can not yet be applied to source dependencies.

[Nebula locking plugin](#1ksv4uv)

This feature is inspired by the [Nebula Gradle dependency lock plugin](https://github.com/nebula-plugins/gradle-dependency-lock-plugin).

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/)