

Gradle 0.9 Release Notes

1. [#New and Noteworthy](#)
2. [#Migrating from 0.8](#)
3. [#Fixed Jira Issues](#)

New and Noteworthy

Incremental builds

Gradle will now skip a task if its input files have not changed since the task last ran. It will do this for all built-in tasks and for any custom tasks that you write. It can even do this for the ad hoc tasks that you define in your build script. Writing builds which are both large and performant has never been easier.

We've put a lot of effort into making sure that incremental builds are reliable. No more running `clean` before you run a build. Gradle will take care of things such as cleaning up old classes and resources, and recompiling dependent classes when source files change. This means no more running stale tests or building archives that contain the stale files.

See the [user guide](#) for some details about how to use this for your tasks.

Simple build composition using scripts

You can now easily apply any number of build scripts to a project. This allows you to do lots of useful things: you can split a large build script into smaller pieces, you can use it to apply common configuration to multiple projects, or to multiple builds across your organization. You can even implement plugins using scripts.

Scripts can be applied from any URL, which means you can share scripts using HTTP.

You can apply scripts to any arbitrary object, so you can use them to configure any object, such as tasks, repositories or your own custom build objects.

See the [user guide](#)

The Gradle Daemon

This release includes the experimental Gradle daemon. The daemon runs in the background, and performs builds. This way it avoids the startup costs.

See the [user guide](#)

Gradle build language reference guide

We've added a reference guide for the Gradle build language. This reference guide documents all the properties, methods and script blocks for the main classes that make up the Gradle DSL, along with all of the task implementations.

See <http://gradle.org/0.9/docs/dsl/index.html>

Gradle Plugin for IDEA

A plugin for IDEA is now available which allows you to explore and run Gradle builds from within IDEA. The plugin is available via the plugin manager in IDEA.

IDE Plugins

We've add an 'idea' plugin which generates the project files for IDEA, and we've rewritten the 'eclipse' plugin. Both plugins automatically discover and download the source and javadoc JARs for all your dependencies. And both plugins offer lots of hooks for flexible configuration.

See the [idea plugin](#) and [eclipse plugin](#) for details.

New archive task API

The archive tasks now share the same API as the copy task, based on the powerful `CopySpec` interface. This means that all the capabilities you can use when copying are now available when you create an archive. This includes being able to filter the contents or rename certain files as they are being added to the archive. You can share `CopySpecs` between tasks, so you can, for example, use the same definition to generate a WAR and an exploded WAR. Or ZIP, TAR and exploded distributions.

In addition, the `Jar` and `War` tasks have been much simplified, so that it is much easier to define the contents of the META-INF and WEB-INF directories. The `Jar` and `War` tasks are now easily used in builds which don't use the Java or War plugins.

Manifest generation is also much improved. You can share manifest definitions between tasks, import from existing manifest, or merge manifests together in flexible ways.

See [working with files](#) and [java plugin](#)

Testing

Parallel test execution

The `Test` task now supports parallel test execution. Enabling this is as easy as setting the `maxParallelForks` property. So now, you can easily take advantage of those extra cores that weren't doing anything, to speed up your test execution.

Execute a single test

You can run Gradle with `-Dtest.single=SomeClassName` to run a single test class.

Maximum tests per process

You can also specify the maximum number of test classes to execute in a forked test process. The process is restarted when the limit is reached. Fork modes 'per test' and 'once' are just special cases of this. This provides an efficient alternative to adding more and more heap space for your test process, without the performance hit of forking per test.

Test listeners

Finally, you can now register a listener to receive notification as tests execute, to do custom reporting or some other test result analysis.

These test features work equally well for JUnit and TestNG test suites.

See the [user guide](#)

Antlr support

The new Antlr plugin adds support for generating source from Antlr grammars.

See the [user guide](#)

Announce plugin

The new Announce plugin adds support for providing notifications from your build to various destinations, such as twitter, or locally using snarl, libnotify, or growl.

See the [user guide](#)

Maven POM generation

You can change or extend what Gradle adds to generated Maven POMs, using exactly the same syntax that polygot Maven uses.

File handling

More useful methods have been added to various file handling APIs. For example, you can now obtain the contents of a ZIP file as a file tree, which you can pass to any task which understands file collections (pretty much all of them). The file DSL is now available in all scripts which Gradle executes, not just build scripts.

See the [user guide](#)

New tasks

We've added a `Sync` task, which extends the `Copy` task to delete any files from the destination directory which were not part of the copy operation. This is useful for things like installing your distribution, or for maintaining a copy of your dependencies in a particular directory.

The general purpose `Delete` task replaces the specific `Clean` and `EclipseClean` tasks.

There's also a new `GradleBuild` task, which you can use to execute another Gradle build from within your build. We've added an `Exec` task to execute arbitrary commands, and a `JavaExec` task to execute arbitrary Java applications.

See the [DSL reference](#)

Build profiler

A profiler is now built into Gradle, to help you profile your builds if you have performance problems.

See the [user guide](#)

Writing plugins

Added `ProjectBuilder` to help you test your custom task and plugin implementations.

See the [user guide](#)

Updated dependencies

Build scripts are now executed using Groovy 1.7.6 and Ant 1.8.1

Performance

- Incremental builds will help your average build time.
- Parallel test execution and reforking can make a huge difference for a long running test suite.
- Start-up time has improved thanks to the Gradle daemon and Groovy start-up improvements.
- Large multi-project builds with complex dependency graphs will see significant improvements.
- The test task is highly concurrent internally, so that things like detecting test files, launching processes, and writing results files are all done concurrently.
- Copy and the archive tasks perform streaming to keep memory requirements down.

Migrating from 0.8

[Gradle 0.9 Breaking Changes](#)

Fixed Jira Issues

Error rendering macro 'jiraissues' : JIRA project does not exist or you do not have permission to view it.