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/)

# Overview

Contents

[Features](#4d34og8)

[Why Groovy?](#2s8eyo1)

[Features](#4d34og8)

Here is a list of some of Gradle’s features.

Declarative builds and build-by-convention

At the heart of Gradle lies a rich extensible Domain Specific Language (DSL) based on Groovy. Gradle pushes declarative builds to the next level by providing declarative language elements that you can assemble as you like. Those elements also provide build-by-convention support for Java, Groovy, OSGi, Web and Scala projects. Even more, this declarative language is extensible. Add your own new language elements or enhance the existing ones, thus providing concise, maintainable and comprehensible builds.

Language for dependency based programming

The declarative language lies on top of a general purpose task graph, which you can fully leverage in your builds. It provides utmost flexibility to adapt Gradle to your unique needs.

Structure your build

The suppleness and richness of Gradle finally allows you to apply common design principles to your build. For example, it is very easy to compose your build from reusable pieces of build logic. Inline stuff where unnecessary indirections would be inappropriate. Don’t be forced to tear apart what belongs together (e.g. in your project hierarchy). Avoid smells like shotgun changes or divergent change that turn your build into a maintenance nightmare. At last you can create a well structured, easily maintained, comprehensible build.

Deep API

From being a pleasure to be used embedded to its many hooks over the whole lifecycle of build execution, Gradle allows you to monitor and customize its configuration and execution behavior to its very core.

Gradle scales

Gradle scales very well. It significantly increases your productivity, from simple single project builds up to huge enterprise multi-project builds. This is true for structuring the build. With the state-of-art incremental build function, this is also true for tackling the performance pain many large enterprise builds suffer from.

Multi-project builds

Gradle’s support for multi-project build is outstanding. Project dependencies are first class citizens. We allow you to model the project relationships in a multi-project build as they really are for your problem domain. Gradle follows your layout not vice versa.

Gradle provides partial builds. If you build a single subproject Gradle takes care of building all the subprojects that subproject depends on. You can also choose to rebuild the subprojects that depend on a particular subproject. Together with incremental builds this is a big time saver for larger builds.

Many ways to manage your dependencies

Different teams prefer different ways to manage their external dependencies. Gradle provides convenient support for any strategy. From transitive dependency management with remote Maven and Ivy repositories to jars or directories on the local file system.

Gradle is the first build integration tool

Ant tasks are first class citizens. Even more interesting, Ant projects are first class citizens as well. Gradle provides a deep import for any Ant project, turning Ant targets into native Gradle tasks at runtime. You can depend on them from Gradle, you can enhance them from Gradle, you can even declare dependencies on Gradle tasks in your build.xml. The same integration is provided for properties, paths, etc …​

Gradle fully supports your existing Maven or Ivy repository infrastructure for publishing and retrieving dependencies. Gradle also provides a converter for turning a Maven pom.xml into a Gradle script. Runtime imports of Maven projects will come soon.

Ease of migration

Gradle can adapt to any structure you have. Therefore you can always develop your Gradle build in the same branch where your production build lives and both can evolve in parallel. We usually recommend to write tests that make sure that the produced artifacts are similar. That way migration is as less disruptive and as reliable as possible. This is following the best-practices for refactoring by applying baby steps.

Groovy

Gradle’s build scripts are written in Groovy or Kotlin, not XML. But unlike other approaches this is not for simply exposing the raw scripting power of a dynamic language. That would just lead to a very difficult to maintain build. The whole design of Gradle is oriented towards being used as a language, not as a rigid framework. And Groovy is our glue that allows you to tell your individual story with the abstractions Gradle (or you) provide. Gradle provides some standard stories but they are not privileged in any form. This is for us a major distinguishing feature compared to other declarative build systems. Our Groovy support is not just sugar coating. The whole Gradle API is fully Groovy-ized. Adding Groovy results in an enjoyable and productive experience.

The Gradle wrapper

The Gradle Wrapper allows you to execute Gradle builds on machines where Gradle is not installed. This is useful for example for some continuous integration servers. It is also useful for an open source project to keep the barrier low for building it. The wrapper is also very interesting for the enterprise. It is a zero administration approach for the client machines. It also enforces the usage of a particular Gradle version thus minimizing support issues.

Free and open source

Gradle is an open source project, and is licensed under the [Apache License 2.0](https://github.com/gradle/gradle/blob/master/LICENSE).

[Why Groovy?](#2s8eyo1)

We think the advantages of an internal DSL (based on a dynamic language) over XML are tremendous when used in *build scripts*. There are a couple of dynamic languages out there. Why Groovy? The answer lies in the context Gradle is operating in. Although Gradle is a general purpose build tool at its core, its main focus are Java projects. In such projects the team members will be very familiar with Java. We think a build should be as transparent as possible to *all* team members.

In that case, you might argue why we don’t just use Java as the language for build scripts. We think this is a valid question. It would have the highest transparency for your team and the lowest learning curve, but because of the limitations of Java, such a build language would not be as nice, expressive and powerful as it could be.[[1](#17dp8vu)] Languages like Python, Groovy or Ruby do a much better job here. We have chosen Groovy as it offers by far the greatest transparency for Java people. Its base syntax is the same as Java’s as well as its type system, its package structure and other things. Groovy provides much more on top of that, but with the common foundation of Java.

For Java developers with Python or Ruby knowledge or the desire to learn them, the above arguments don’t apply. The Gradle design is well-suited for creating another build script engine in JRuby or Jython. It just doesn’t have the highest priority for us at the moment. We happily support any community effort to create additional build script engines.

[1](#3rdcrjn). At <http://www.defmacro.org/ramblings/lisp.html> you find an interesting article comparing Ant, XML, Java and Lisp. It’s funny that the 'if Java had that syntax' syntax in this article is actually the Groovy syntax.

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/)