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/)

# Task Configuration Avoidance

Contents

[What is the new API good for?](#4d34og8)

[What do we want you to do?](#2s8eyo1)

[Guidelines](#17dp8vu)

[Migration Guide](#3rdcrjn)

[Old vs New API overview](#26in1rg)

[Profiling with a Gradle Profiler Scenario File](#lnxbz9)

We are asking early adopters to try out a new API for creating and configuring tasks to iron out any issues and gather feedback. This chapter provides a quick introduction to the feature and explains some guidelines for migrating your build to use the new API. We intend for this new API to eventually replace the existing API through our usual deprecation process over several major releases.

| **✨** | The new Task configuration API is currently [incubating](http://docs.google.com/feature_lifecycle.html#feature_lifecycle). Please be aware that the DSL or API may change in minor Gradle versions. |
| --- | --- |

[What is the new API good for?](#4d34og8)

In a nutshell, the new API allows builds to avoid the cost of creating and configuring tasks during Gradle’s configuration phase when those tasks will never be executed. We identified task configuration as a contributor to overall configuration time. Configuration time affects every build, so making it faster is good for everyone. For example, if you’re only compiling, Gradle doesn’t need to also configure code quality, testing and publishing tasks. For tasks that aren’t used often by developers, configuration can be expensive (e.g., running git status or querying a webservice) and can slow down builds that do not need to execute them. This new API avoids configuring those tasks if they are not needed.

This feature is designed to be backwards compatible when mixed with the old and new APIs, so you can gradually migrate your plugins and build scripts to the new API. Instead of immediately creating a task and running all configuration actions, Gradle exposes a [Provider](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html) to a task. A task will only be created and configured when the task is required. If at any point, a task registered with the new API is accessed through the old API, the task will be created and configured just as if it had been created with the old API.

Many built-in Gradle plugins have been converted to use this new API already, so you may see some benefits without changing anything.

[What do we want you to do?](#2s8eyo1)

We’ve been using this feature in many of the built-in Gradle plugins, internal Gradle projects and the Gradle build itself. Now we would appreciate your help in evaluating it with your builds as well.

We’d like for you to take a few steps:

* Try your build with the latest nightly (see the version number at the top of this page).
* How many tasks do you see registered with the new API? See [How do I know it’s working?](#35nkun2) for how to find this.
* Which task types are most numerous? Which ones are created with the old API? Which ones are registered with the new API, but have been created/configured anyways?
* Following [Pitfalls and migration guidelines](#3rdcrjn), change your build scripts to use the new API to avoid creating/configuring tasks that are from the built-in Gradle plugins. Is there an impact to the number of tasks that are created?
* Convert some of your custom plugins/tasks to use the new API. Is there an impact to the number of tasks using the new API and the number of new API tasks that are created?
* Run the [gradle-profiler](https://github.com/gradle/gradle-profiler) to measure the difference between a build with less task configuration and a build where all tasks are configured. What do you see? Use [the scenario file below](#lnxbz9).

[Guidelines](#17dp8vu)

[How do I defer task creation?](#1ksv4uv)

This feature requires build authors to opt-in by migrating task creation from the [TaskContainer.create(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.lang.String-) APIs to the [TaskContainer.register(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#register-java.lang.String-) APIs. The register(…​) API registers a task to be created at a later time if and only if the task is needed. The create(…​) API continues to eagerly create and configure tasks when it is called.

Using the new creation API may not be enough to avoid all task configuration completely. You may need to change other code that configures tasks by name or by type, as explained in the following sections.

[How do I defer task configuration?](#44sinio)

Existing APIs like [DomainObjectCollection.all(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#all-org.gradle.api.Action-) and [DomainObjectCollection.withType(java.lang.Class, org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#withType-java.lang.Class-org.gradle.api.Action-) will immediately create and configure any registered tasks. To defer task configuration, you will need to migrate to a new API equivalent. See the [table below](#26in1rg) to identify the new API alternative.

[How do I reference a task without creating/configuring it?](#2jxsxqh)

Calling [Provider.get()](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html#get--) or looking up a task by name with [TaskCollection.getByName(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#getByName-java.lang.String-) will cause the task to be created and configured. Methods like [Task.dependsOn(java.lang.Object…​)](http://docs.google.com/dsl/org.gradle.api.Task.html#org.gradle.api.Task:dependsOn(java.lang.Object%5B%5D)) and [ConfigurableFileCollection.builtBy(java.lang.Object...)](http://docs.google.com/javadoc/org/gradle/api/file/ConfigurableFileCollection.html#builtBy-java.lang.Object...-) work with [TaskProvider](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskProvider.html) in the same way as [Task](http://docs.google.com/dsl/org.gradle.api.Task.html), so you do not need to unwrap a Provider for explicit dependencies to continue to work.

If you are configuring a task by name, you will need to use the new API equivalent. See the [table below](#26in1rg) to identify the new API alternative.

[How to get an instance of a Task?](#z337ya)

In the event you still need to get access to a Task instance, you can use [TaskCollection.named(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#named-java.lang.String-) and call [Provider.get()](http://docs.google.com/javadoc/org/gradle/api/provider/Provider.html#get--). This will cause the task to be created/configured, but everything should work as it has with the eager APIs.

Please [contact us](https://github.com/gradle/gradle/issues/5664) if you find tricky build logic that requires you to get a Task instance.

[How do I know if it’s working?](#35nkun2)

We provide a couple of internal flags to print out information after a build. This will eventually be captured and displayed in an easier to digest form.

**gradle -Dorg.gradle.internal.tasks.stats help**

Task counts: Old API 188, New API 587, total 775 # 1  
Task counts: created 613, avoided 162, %-lazy 21 # 2  
  
Task types that were created with the old API # 3  
class org.gradle.api.DefaultTask 31  
class org.jetbrains.kotlin.gradle.tasks.KotlinCompile 28  
class org.gradle.api.tasks.JavaExec 24  
class org.jlleitschuh.gradle.ktlint.KtlintCheck 24  
class org.gradle.api.tasks.bundling.Jar 16  
class org.gradle.kotlin.dsl.accessors.tasks.PrintAccessors 16  
class org.gradle.plugin.devel.tasks.ValidateTaskProperties 15  
class org.gradle.another.CoolTask 13  
class org.gradle.plugin.devel.tasks.GeneratePluginDescriptors 10  
class org.gradle.plugin.devel.tasks.PluginUnderTestMetadata 10  
class org.gradle.kotlin.dsl.accessors.tasks.UpdateProjectSchema 1  
  
Task types that were registered with the new API but were created anyways # 4  
class org.gradle.api.DefaultTask 142  
class org.gradle.api.tasks.Delete 106  
class org.gradle.api.tasks.compile.JavaCompile 32  
class org.gradle.language.jvm.tasks.ProcessResources 32  
class org.gradle.api.tasks.testing.Test 16  
class org.gradle.api.tasks.javadoc.Javadoc 16  
class org.gradle.plugins.ide.idea.GenerateIdeaModule 15  
class org.gradle.plugins.ide.eclipse.GenerateEclipseClasspath 15  
class org.gradle.plugins.ide.eclipse.GenerateEclipseJdt 15  
class org.gradle.plugins.ide.eclipse.GenerateEclipseProject 15  
class org.gradle.api.tasks.compile.GroovyCompile 10  
class org.gradle.api.tasks.javadoc.Groovydoc 5  
class org.gradle.api.plugins.quality.Checkstyle 4  
class org.gradle.api.plugins.quality.CodeNarc 2

1. 188 tasks are created with the old API and 587 tasks are registered with the new API for a total of 775 tasks.
2. 613 tasks were created and configured and only 162 tasks were avoided (never created or configured). 21% of all tasks were avoided (higher is better).
3. Lists of the type of tasks that were created with the old API. This is a good list to work down to increase the amount of possible avoidable task configuration.
4. Lists of the type of tasks that were created with the new API but were created/configured anyways. This is a good list to work down to increase the amount of task configuration that is avoided.

These statistics are printed out once per build. Projects with buildSrc and composite builds will display this information multiple times. In a build that uses the new APIs perfectly, we should see 0 tasks created with the old API and only 1 created/configured task because we are only executing the help task. If you run other tasks (like build), you should expect many more tasks to be created and configured.

You can use the list of task types to guide which tasks would provide the biggest bang for your buck when you migrate to the new API.

To approximate the time it takes to configure a build without executing tasks, you can run gradle help. Please use the Gradle Profiler to measure your build as described [further on](#lnxbz9).

[Migration Guide](#3rdcrjn)

The following sections will go through some [general guidelines](#3j2qqm3) to adhere when migrating the build logic as well as [the steps](#1y810tw) we recommend following. We also cover some [troubleshooting](#4i7ojhp) and [pitfalls](#2xcytpi) to help you work around some issues you may encountered during the migration.

[General](#3j2qqm3)

1. Use help task as a benchmark during the migration. The help task is the perfect candidate to benchmark your migration process. In a completely lazy build, [a build scan should show no task created immediately or created during configuration](https://scans.gradle.com/s/o7qmlmmrsfxz4/performance/configuration?openScriptsAndPlugins=WzFd). Only a single task, the help task, should be created during task graph calculation. Be mindful of [the pitfall around the lazy API usage in the build scan plugin](#1ci93xb).
2. Only mutate the current task inside configuration action. Because the task configuration action can now run immediately, later or never, mutating anything other than the current task will expose flaky-like behavior in your build. In the future, Gradle will consider these flaky-mutations as failures. It also means the configuration needs to flow from the extensions to the tasks. Thus, the task should be configured by the extension and not the other way around.
3. Prefer small incremental changes. Smaller changes are easier to sanity check. If you ever break your build logic, it will be easier to analyze the changelog since the last successful verification.
4. Ensure a good plan is established for validating the build logic. Usually, a simple build task invocation should do the trick to validate all corner of your build logic. However, if all automation aspect is centralized around Gradle, you may need to validate other task execution path.
5. Prefer automatic testing to manual testing. It’s good practice to write integration test for your build logic using TestKit.
6. Avoid referencing a task by name. In the majority of the case, referencing a task by name is a fragile pattern and should be avoided. Although task name is available on the TaskProvider, effort should be made to use references from a strongly typed model instead.
7. Use the lazy task API as much as possible. Although realizing a task within a lazy configuration action may not have any immediate impact, it makes misuse of the lazy API more costly. The lazy task API is an opt-in feature, meaning someone may be realizing task eagerly which would cause the realization to realize other tasks transitively. Using TaskProvider helps create an indirection that protects transitive realization.
8. Some APIs may be disallowed if you try to access them from the new API’s configuration blocks. For example, [Project.afterEvaluate()](http://docs.google.com/dsl/org.gradle.api.Project.html#org.gradle.api.Project:afterEvaluate(org.gradle.api.Action)) cannot be called when configuring a task registered with the new API. Since afterEvaluate is used to delay configuring a Project, mixing delayed configuration with the new API can cause hard to track down configuration errors because tasks registered with the new API are not always configured, but an afterEvaluate block may be expected to always execute.

[Migration Steps](#1y810tw)

The first part of the migration process is to go through the code and manually migrate eager task creation and configuration to use lazy APIs. The following explores the recommended steps for a successful migration. While going through those steps, keep in mind [the guidelines](#3j2qqm3).

| **✨** | Using the new API in a plugin will require users to use Gradle 4.9 or newer. Plugin authors should refer to [Supporting older versions of Gradle](#3whwml4) section. |
| --- | --- |

1. Migrate task configuration that affects all tasks (tasks.all {}) or subsets by type (tasks.withType(…​) {}). This will cause your build to create fewer built-in Gradle task types. Note that despite some built-in Gradle task aren’t using the lazy APIs, the configuration affecting those task type should still be migrated. We recommend using regex code search capability offered by modern IDE. For example, the Intellij regex \.all\s\*((\(?\s\*\{)|\(\s\*new) will identify every call to .all of a DomainObjectCollection and subtype. We will need to go through those result and identify and migrate only the method calls affecting the TaskContainer or TaskCollection. Be aware of the [common pitfall around deferred configuration](#2xcytpi).
2. Migrate tasks configured by name. Just like the previous point, it will cause your build to create fewer built-in Gradle tasks. For example, logic that uses TaskContainer#getByName(String, Closure/Action) would be converted to TaskContainer#named(String).configure(Closure/Action). It also include [task configuration as DSL blocks](#2bn6wsx). Be aware of the [common pitfall around deferred configuration](#2xcytpi).
3. Migrate tasks creation to register(…​). It will cause your build to create fewer tasks in general. Be aware of the [common pitfall around deferred configuration](#2xcytpi).

At this point, we should see a decent improvement regarding build configuration. We recommend [profiling your build and note the improvement](#35nkun2). Under some circumstance, you may [notice no improvement](#qsh70q). It’s unfortunate, and we feel your frustration. Keep reading onto the next section.

[Troubleshooting](#4i7ojhp)

* The build became unstable after migrating to lazy task configuration. It’s a common issue that happens during the migration. Follow these next steps to find out the issue:
  1. Is the build succeed when all tasks are eagerly resolved on creation using the command line flag (-Dorg.gradle.internal.tasks.eager=true)? Verify that [each task configuration only mutating the task](#3as4poj).
  2. Was there any build logic accidentally deleted? Verify [your change log](#1pxezwc) for disparity in the configuration that may have been deleted.
  3. It may be a real issue. For example, [the issue of FileCollection using TaskProvider was a real discrepancy](https://github.com/gradle/gradle-native/issues/737) found while migrating a build to lazy task API. As a rule of thumb, a TaskProvider should be accepted everywhere a Task instance is accepted.
* Where a task was resolved? As we keep developing the feature, more reporting, and troubleshooting information will be made available to answer this question. In the meantime, [build scan is the best way to answer this question](https://gradle.com/enterprise/releases/2018.3#reduce-configuration-time-by-leveraging-task-creation-avoidance). Follow these steps:
  1. [Create a build scan](https://scans.gradle.com/). Execute the Gradle command using the --scan flag.
  2. Navigate to the configuration performance tab.  
     *Figure 1. Navigate to configuration performance tab in build scan*
     1. Navigate to the performance card from the left side menu.
     2. Navigate to the configuration tab from the top of the performance card.
  3. All the information requires will be presented.  
     *Figure 2. Configuration performance tab in build scan annotated*
     1. Total tasks present when each task are created or not.
        + Created immediately represent all the task created using the eager task APIs.
        + Created during configuration represent all the tasks that were created using the lazy task APIs, but was realized explicitly (via TaskProvider#get()) or implicitly using the eager task query APIs.
        + Both "Created immediately" and "Created during configuration" numbers are considered the "bad" numbers that should be minimized as much as possible.
        + Created during task graph calculation number represent all the task created while building the execution task graph. Typically, this number should be equal to the number of tasks that is executed.
        + Not created number represents all the tasks that were avoided in this build session.
     2. The next section helps answer the question of where a task was realized. For each scripts, plugins and lifecycle callbacks, the last column represents the tasks that were created either immediately or during configuration. Ideally, this column should be empty. The migration should be focusing on the build logic under your control.
     3. Extending any scripts, plugins, or lifecycle callbacks subsection shows a break down in which project it was applied.
* The build performance haven’t improved after migrating to lazy API. It is important to note that lazy task configuration is an opt-in feature. It means the entire configuration chain needs to be lazy to get the benefit. If at any point, an eager API is used for a specific named task, task type or, the worst, the entire task container, all benefits are mitigated. The first step is to have a look at the statistics to see how much more the build can be improved. If the statistic shows the build is completely lazy (work-avoided) [please contact us](https://github.com/gradle/gradle/issues/5664) so we can discuss where this feature falls short and improve it for the next Gradle version.

[Pitfalls](#2xcytpi)

* Lazy task configuration is an opt-in feature. Any eager realization of a task will mitigate the benefit of this new API. In the far future, eager realizations will become an error, until them, it’s up to the build author to avoid eager realization.
* Beware of the hidden eager task realization. Gradle has lots of ways to configure a task eagerly. TaskContainer#getByName(String, Action) is an explicit configuration by name. Configuring a task as a DSL block is an alias to the previous explicit configuration, meaning a lazy task will be prematurely resolved:  
  // Given a task lazily created with  
  tasks.register "someTask"  
    
  // Some time later, the task is configured using a DSL block like  
  someTask {  
   // The task is resolved immediately, and this closure is executed immediately  
  }  
  You will have to migrate the DSL task configuration to the following:  
  tasks.named("someTask").configure {  
   // ...  
   // Beware of the pitfalls here  
  }  
  While migrating to the lazy task configuration, you may still have eager task configuration that references a task by name using Groovy’s property to task reference syntactic sugar. If this lookup is done from anything other than a lazy configuration action, the lazy task reference by the name will be prematurely realized:  
  tasks.register "someTask"  
    
  // Sometime later, an eager task is configured like  
  task aEagerTask {  
   // The task is resolved immediately as it will be treated as a property of the project and will result in an eager task lookup by name  
   dependsOn someTask  
  }  
  You can solve the following in three ways:
  + Use TaskProvider variable. Useful when the task is referenced multiple time in the same build script.  
    def someTask = tasks.register "someTask"  
      
    task aEagerTask {  
     dependsOn someTask  
    }
  + Migrate consumer task. It will only work under specific circumstances. If the task gets resolve later or the realization may happen in a configuration action with no ways of actually knowing where the task is created or configured, this solution won’t be helpful.  
    tasks.register "someTask"  
      
    tasks.register("aEagerTask") {  
     dependsOn someTask  
    }
  + Lookup the task lazily. Useful when the task registration isn’t co-located with its eager usage.  
    tasks.register "someTask"  
      
    task aEagerTask {  
     dependsOn tasks.named("someTask")  
    }
* The jar task is eagerly realized, always. The following [issue is tracking this pitfall](https://github.com/gradle/gradle-native/issues/730). Even though this issue wasn’t released with Gradle 4.9, you should reference the jar task lazily.
* The build scan plugin buildScanPublishPrevious task is eager until version 1.15. Upgrade the build scan plugin in your build to use the latest version.

[Supporting older versions of Gradle](#3whwml4)

This section describes two ways to keep your plugin backward compatible with older version of Gradle if you must maintain compatibility with versions of Gradle older than 4.9.

| **✨** | Although backward compatibility is good for users, we still recommended to upgrade to newer Gradle releases in a timely manner. This will reduce your maintenance burden. |
| --- | --- |

The first method to maintain compatibility is to compile your plugin against the Gradle 4.9 API and conditionally call the right APIs with Groovy ([example](https://github.com/melix/jmh-gradle-plugin/blob/a034aa88805b7a06fa9c5a825d573554b2aa23e2/src/main/groovy/me/champeau/gradle/JMHPlugin.groovy#L289-L296)).

The second method is to use Java reflection to cope with the fact that the APIs are unavailable during compilation ([example](https://github.com/tbroyer/gradle-apt-plugin/tree/432509ec85d1ab49296d4f9b21fad876523c6a8a/src/main/java/net/ltgt/gradle/apt)).

It is highly recommended to have cross-version test coverage using [TestKit](http://docs.google.com/test_kit.html#sub:gradle-runner-gradle-version) and multiple versions of Gradle.

[Old vs New API overview](#26in1rg)

| **✨** | * Methods that take a groovy.lang.Closure are covered in the new API with methods taking org.gradle.api.Action. * More convenience methods may be added in the future based on user feedback. * Some old API methods may never have a direct replacement in the new API. * Some APIs may be restricted when accessed in a lazy configuration block. |
| --- | --- |

| **Old vs New API** | **Description** |
| --- | --- |
| Instead of: task myTask(type: MyTask) {} | There is not a shorthand Groovy DSL for using the new API. |
| Use: tasks.register("myTask", MyTask) {} |
| Instead of: [TaskContainer.create(java.util.Map)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.util.Map-) | Use one of the alternatives below. |
| Use: No direct equivalent. |
| Instead of: [TaskContainer.create(java.util.Map, groovy.lang.Closure)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.util.Map-groovy.lang.Closure-) | Use one of the alternatives below. |
| Use: No direct equivalent. |
| Instead of: [TaskContainer.create(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.lang.String-) | This returns a TaskProvider instead of a Task. |
| Use: [TaskContainer.register(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#register-java.lang.String-) |
| Instead of: [TaskContainer.create(java.lang.String, groovy.lang.Closure)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.lang.String-groovy.lang.Closure-) | This returns a TaskProvider instead of a Task. |
| Use: [TaskContainer.register(java.lang.String, org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#register-java.lang.String-org.gradle.api.Action-) |
| Instead of: [TaskContainer.create(java.lang.String, java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.lang.String-java.lang.Class-) | This returns a TaskProvider instead of a Task. |
| Use: [TaskContainer.register(java.lang.String, java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#register-java.lang.String-java.lang.Class-) |
| Instead of: [TaskContainer.create(java.lang.String, java.lang.Class, org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.lang.String-java.lang.Class-org.gradle.api.Action-) | This returns a TaskProvider instead of a Task. |
| Use: [TaskContainer.register(java.lang.String, java.lang.Class, org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#register-java.lang.String-java.lang.Class-org.gradle.api.Action-) |
| Instead of: [TaskContainer.create(java.lang.String, java.lang.Class, java.lang.Object…​)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#create-java.lang.String-java.lang.Class-java.lang.Object...-) | This returns a TaskProvider instead of a Task. |
| Use: [TaskContainer.register(java.lang.String, java.lang.Class, java.lang.Object…​)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#register-java.lang.String-java.lang.Class-java.lang.Object...-) |
| Instead of: [TaskCollection.getByName(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#getByName-java.lang.String-) | This returns a TaskProvider instead of a Task. |
| Use: [TaskCollection.named(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#named-java.lang.String-) |
| Instead of: [TaskCollection.getByName(java.lang.String, groovy.lang.Closure)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#getByName-java.lang.String-groovy.lang.Closure-) | This returns a TaskProvider instead of a Task. |
| Use: named(java.lang.String).configure(Action) |
| Instead of: [TaskContainer.getByPath(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#getByPath-java.lang.String-) | Accessing tasks from another project requires a specific ordering of project evaluation. |
| Use: No direct equivalent. |
| Instead of: [NamedDomainObjectCollection.findByName(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/NamedDomainObjectCollection.html#findByName-java.lang.String-) | named(String) is the closest equivalent, but will fail if the task does not exist. Using findByName(String) will cause tasks registered with the new API to be created/configured. |
| Use: No direct equivalent. |
| Instead of: [TaskContainer.findByPath(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#findByPath-java.lang.String-) | See getByPath(String) above. |
| Use: No direct equivalent. |
| Instead of: [TaskCollection.withType(java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#withType-java.lang.Class-) | This is OK to use because it does not require tasks to be created immediately. |
| Use: *OK* |
| Instead of: withType(java.lang.Class).getByName(java.lang.String) | This returns a TaskProvider instead of a Task. |
| Use: withType(java.lang.Class).named(java.lang.String) |
| Instead of: [DomainObjectCollection.withType(java.lang.Class, org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#withType-java.lang.Class-org.gradle.api.Action-) | This returns void, so it cannot be chained. |
| Use: withType(java.lang.Class).configureEach(org.gradle.api.Action) |
| Instead of: [DomainObjectCollection.all(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#all-org.gradle.api.Action-) | This returns void, so it cannot be chained. |
| Use: [DomainObjectCollection.configureEach(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#configureEach-org.gradle.api.Action-) |
| Instead of: TaskCollection.whenTaskAdded(org.gradle.api.Action) | This returns void, so it cannot be chained. |
| Use: [DomainObjectCollection.configureEach(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#configureEach-org.gradle.api.Action-) |
| Instead of: DomainObjectCollection.whenObjectAdded(org.gradle.api.Action) | This returns void, so it cannot be chained. |
| Use: [DomainObjectCollection.configureEach(org.gradle.api.Action)](http://docs.google.com/javadoc/org/gradle/api/DomainObjectCollection.html#configureEach-org.gradle.api.Action-) |
| Instead of: [NamedDomainObjectSet.findAll(groovy.lang.Closure)](http://docs.google.com/javadoc/org/gradle/api/NamedDomainObjectSet.html#findAll-groovy.lang.Closure-) | Avoid calling this method. matching(Spec) and configureEach(Action) are more appropriate in most cases. |
| Use: *OK*, with issues. |
| Instead of: [TaskCollection.matching(groovy.lang.Closure)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#matching-groovy.lang.Closure-) | This is OK to use because it does not require tasks to be created immediately. |
| Use: *OK* |
| Instead of: [TaskCollection.getAt(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#getAt-java.lang.String-) | Avoid calling this directly as it’s a Groovy convenience method. The alternative returns a TaskProvider instead of a Task. |
| Use: [TaskCollection.named(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskCollection.html#named-java.lang.String-) |
| Instead of: iterator() or implicit iteration over the Task collection | Avoid doing this as it requires creating and configuring all tasks. See findAll(Closure) above. |
| Use: *OK*, with issues. |
| Instead of: remove(org.gradle.api.Task) | Avoid calling this. The behavior of remove with the new API may change in the future. |
| Use: *OK*, with issues. |
| Instead of: [TaskContainer.replace(java.lang.String)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#replace-java.lang.String-) | Avoid calling this. The behavior of replace with the new API may change in the future. |
| Use: *OK*, with issues. |
| Instead of: [TaskContainer.replace(java.lang.String, java.lang.Class)](http://docs.google.com/javadoc/org/gradle/api/tasks/TaskContainer.html#replace-java.lang.String-java.lang.Class-) | Avoid calling this. The behavior of replace with the new API may change in the future. |
| Use: *OK*, with issues. |

[Profiling with a Gradle Profiler Scenario File](#lnxbz9)

The [Gradle Profiler](https://github.com/gradle/gradle-profiler) is a tool to measure build times for Gradle builds in a predictable and reproducible manner. The tool automates collecting profiling and benchmark information from a Gradle build and mitigates environmental impacts to measuring build time (like JIT warmups and cached dependencies). Clone and build gradle-profiler locally.

To measure the impact of the new API on your build, we’ve included a sample scenario file you can use. This scenario runs gradle help on your build with a special flag to enable/disable the new API to make it easier to measure improvements. gradle help approximates the time it takes Gradle to configure your build by running only a single, simple task.

**Save as help.scenario**

defaults {  
 tasks = ["help"]  
 warm-ups = 20  
}  
eagerHelp = ${defaults} {  
 gradle-args = ["-Dorg.gradle.internal.tasks.eager=true"]  
}  
lazyHelp = ${defaults} {  
 gradle-args = ["-Dorg.gradle.internal.tasks.eager=false"]  
}

Run gradle-profiler in the root of your build. The results will go into a file called profile-out-N where N is unique for each invocation.

* When measuring your build with gradle-profiler, you should make sure the machine running the benchmark is not also busy doing other things. You may get false positives/negatives if resources are spent doing other things.
* Get a baseline for how long your build takes before making any changes. Run

gradle-profiler --benchmark  
 --iterations 20  
 --gradle-version [some Gradle version]  
 --scenario-file help.scenario  
 eagerHelp lazyHelp

* In the profile-out-N directory, gradle-profiler will generate a CSV and a HTML file to display the results of the benchmarking.
* After making some changes to decrease the number of tasks that are created and configured, re-run the command above.
* For the Gradle build itself, we saw improvements after 50% of the tasks were no longer configured each time. Your mileage may vary depending on how expensive particular tasks are to create and configure.
* Please provide feedback on [this issue](https://github.com/gradle/gradle/issues/5664). Or send us an email at [performance@gradle.com](mailto:performance@gradle.com).

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/)