Schnelle und wartbare Builds

für Projekte jeder Größenordnung

Stefan Wolf

JavaForum Stuttgart 2023

About me

Stefan Wolf

Principal Engineer @ Gradle

GitHub @wolfs

What is Gradle

Gradle Inc.

Gradle Build Tool

Gradle Build Tool is an open-source build system that automates the process of building software of any type, size, or complexity in a fast and reliable manner. What sets Gradle Build Tool apart is its elegant and extensible declarative build language that enables expressing any build in a clear and understandable way.

Gradle Build Tool

Software build tool released under the Apache License

It is one of the 20 most popular open source projects according to TechCrunch with nearly 30 million monthly downloads.

Various ecosystems

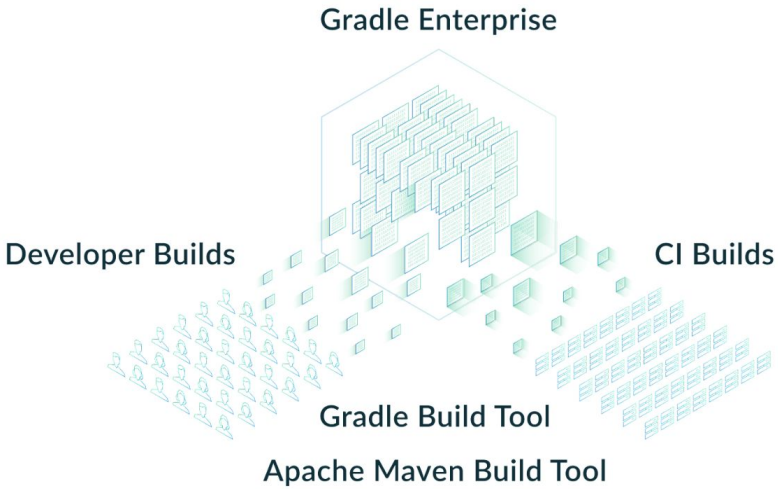
core etc.

Community etc…

Logos from Wikimedia or the respective official™ pages.

Gradle Enterprise

Gradle Enterprise, commercial product, is the first integrated solution of Developer Productivity Engineering (DPE)



Gradle Enterprise

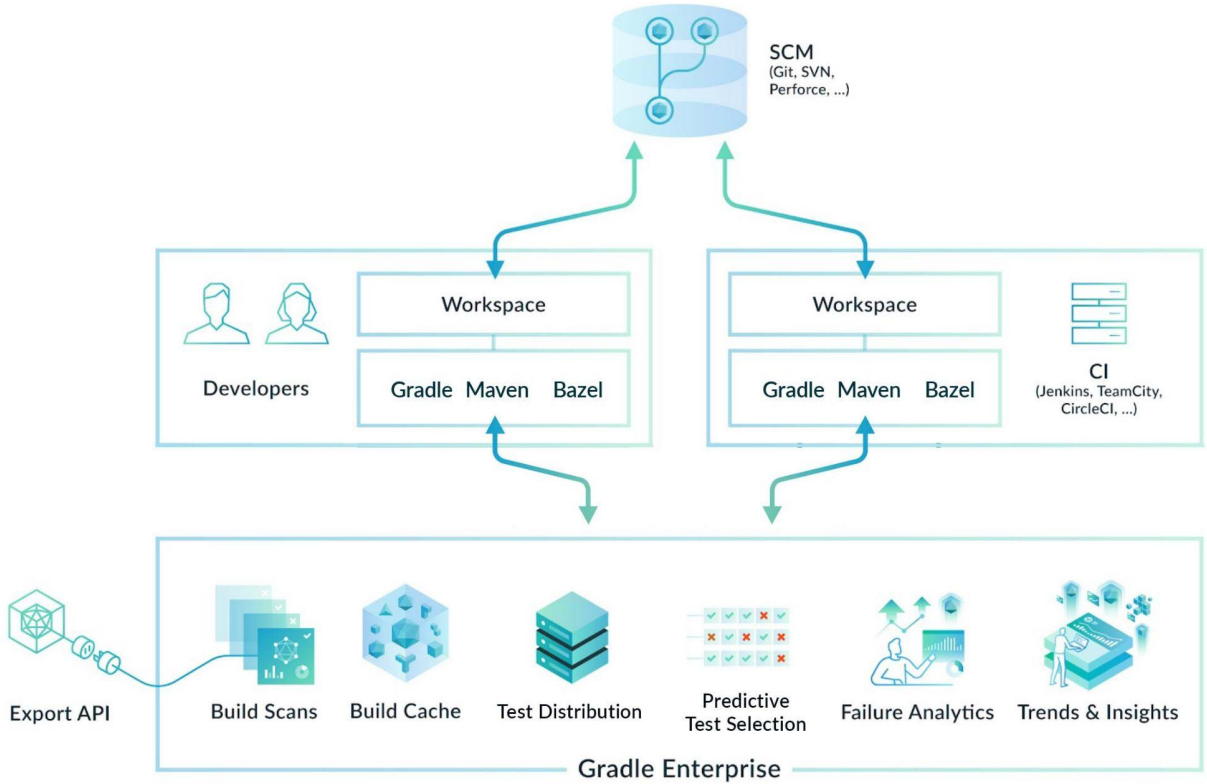
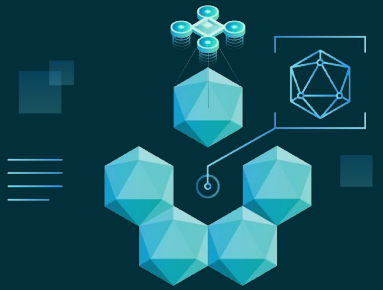


Table of Contents



Fundamentals

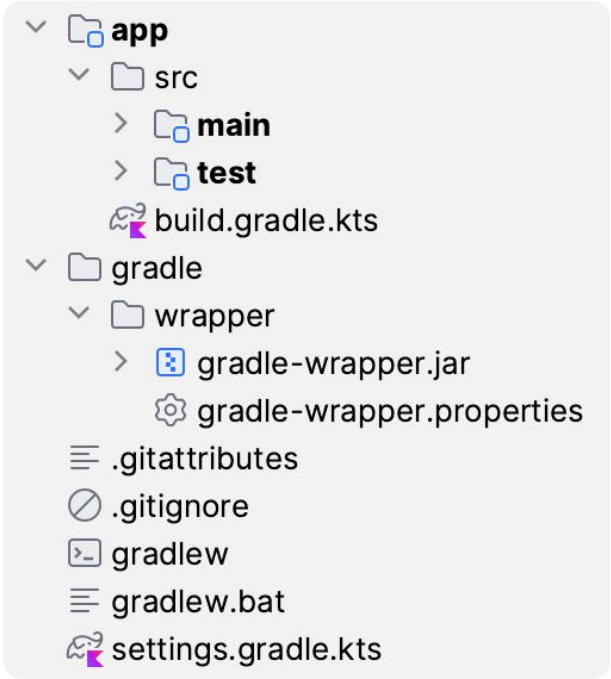
Gradle Init

```
❯ gradle init
```

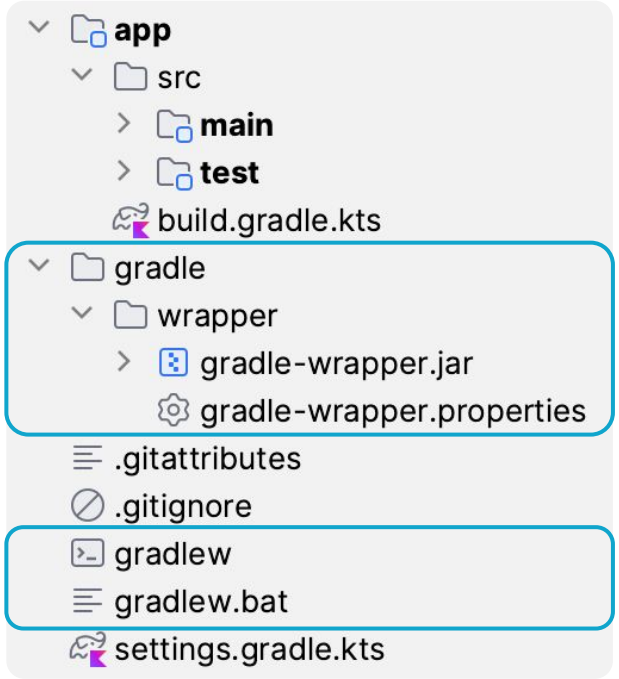
- Create a project via an interactive dialog
- Allows to configure
  - Project template
  - Test framework
  - Gradle setup

```
❯ gradle init --dsl kotlin --incubating \
  --type java-application --test-framework junit-jupiter \
  --package org.example --project-name my-project
```

Project anatomy



Project anatomy

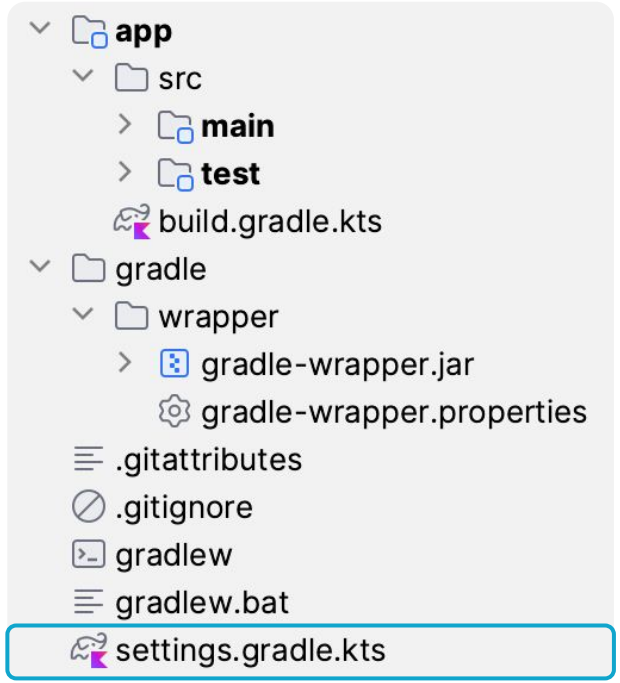


Gradle Wrapper

- Ties the project to a Gradle version
- Downloads Gradle distribution
- Allows upgrading Gradle in the project

```
❯ ./gradlew wrapper --gradle-version 8.2
```

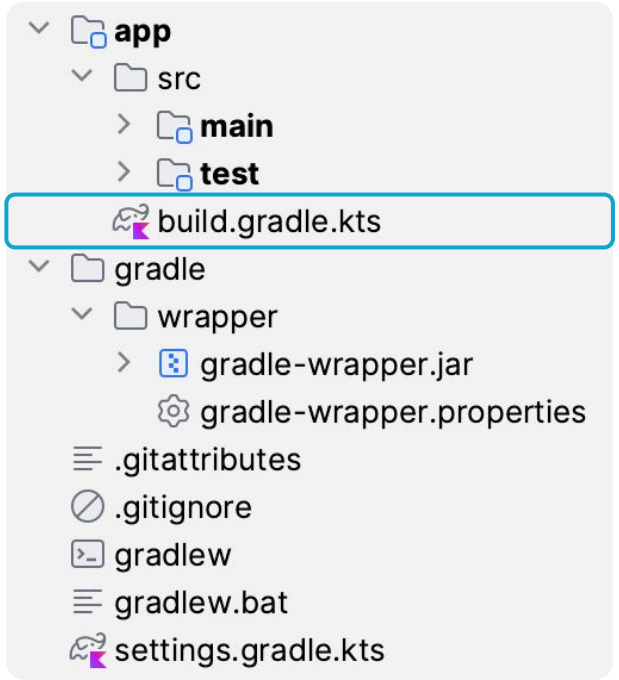
Project anatomy



Build settings

```
rootProject.name = "my-project"

include("app")
```

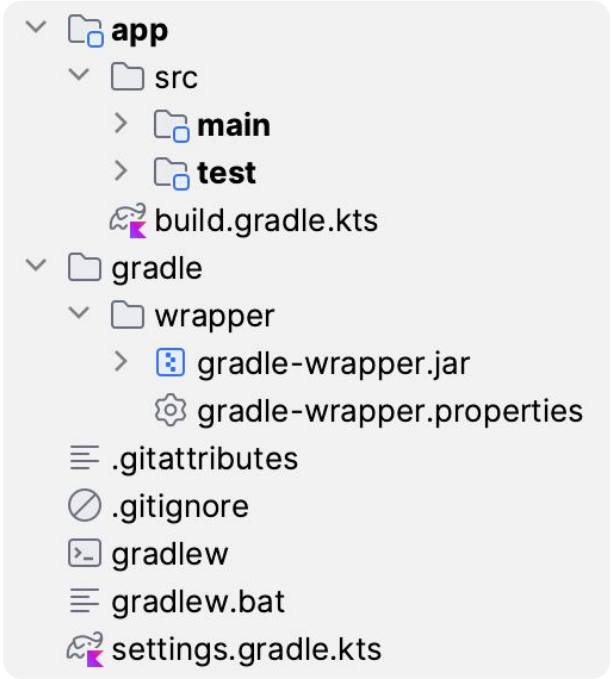
Project anatomy



Build script

```
plugins {
    application
}

repositories {
    mavenCentral()
}

dependencies {
    // ...
}

application {
    mainClass = "my.App"
}
```

Ready for development



Compile sources

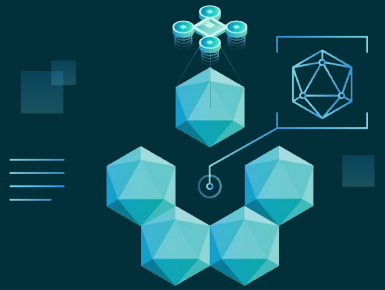
```
❯ ./gradlew classes
```

Execute tests

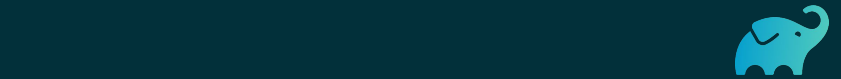
```
❯ ./gradlew check
```

Build full project

```
❯ ./gradlew build
```



Test Suites



Testing a project

- Unit tests
- Integration tests
- End-to-end tests
- Performance tests
- …

Works out-of-the-box,
but requires dependencies

Could live with unit tests
but would run even if those fail

Live in separate non-test project
or require manual source sets setup

Test Suites

build.gradle.kts

```
testing {
    suites {
        val test by getting(JvmTestSuite::class) {
            useJUnitJupiter()
        }
    }
}
```

Test Suites

build.gradle.kts

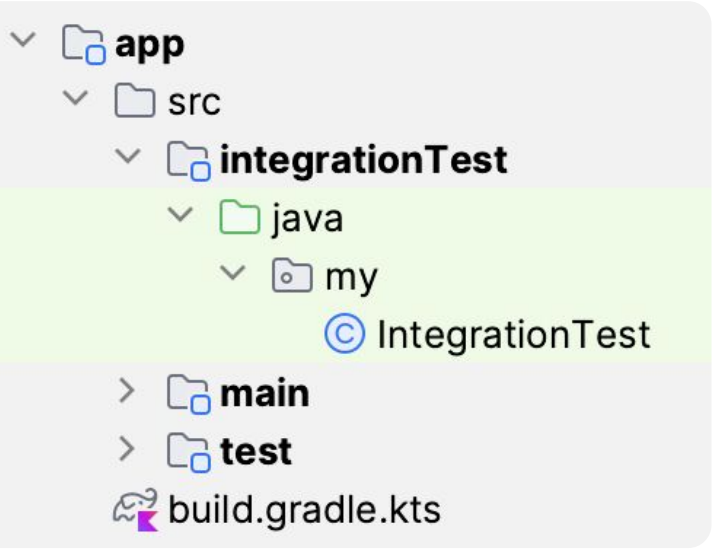
```
testing {
    suites {
        val test by getting(JvmTestSuite::class) {
            useJUnitJupiter()
        }

        register<JvmTestSuite>("integrationTest") {
            dependencies {
                implementation(project())
            }
            useJUnitJupiter("5.8.2")
            targets.all { testTask.configure { shouldRunAfter(test) } }
        }
    }
}
```

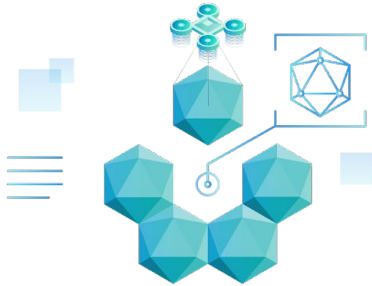
Test Suites

build.gradle.kts

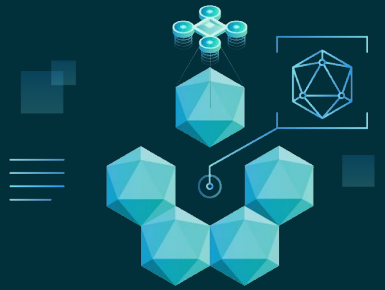
```
testing {
    suites {
        // ...
    }
}

tasks.named("check") {
    dependsOn(testing.suites.named("integrationTest"))
}
```

```
❱ ./gradlew check
…
> Task :app:test
> Task :app:integrationTest
> Task :app:check
…
```

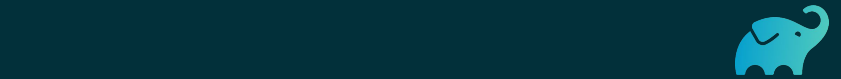




```
class IntegrationTest {

    @Test
    void integrationTest() {
        // ...
    }
}
```



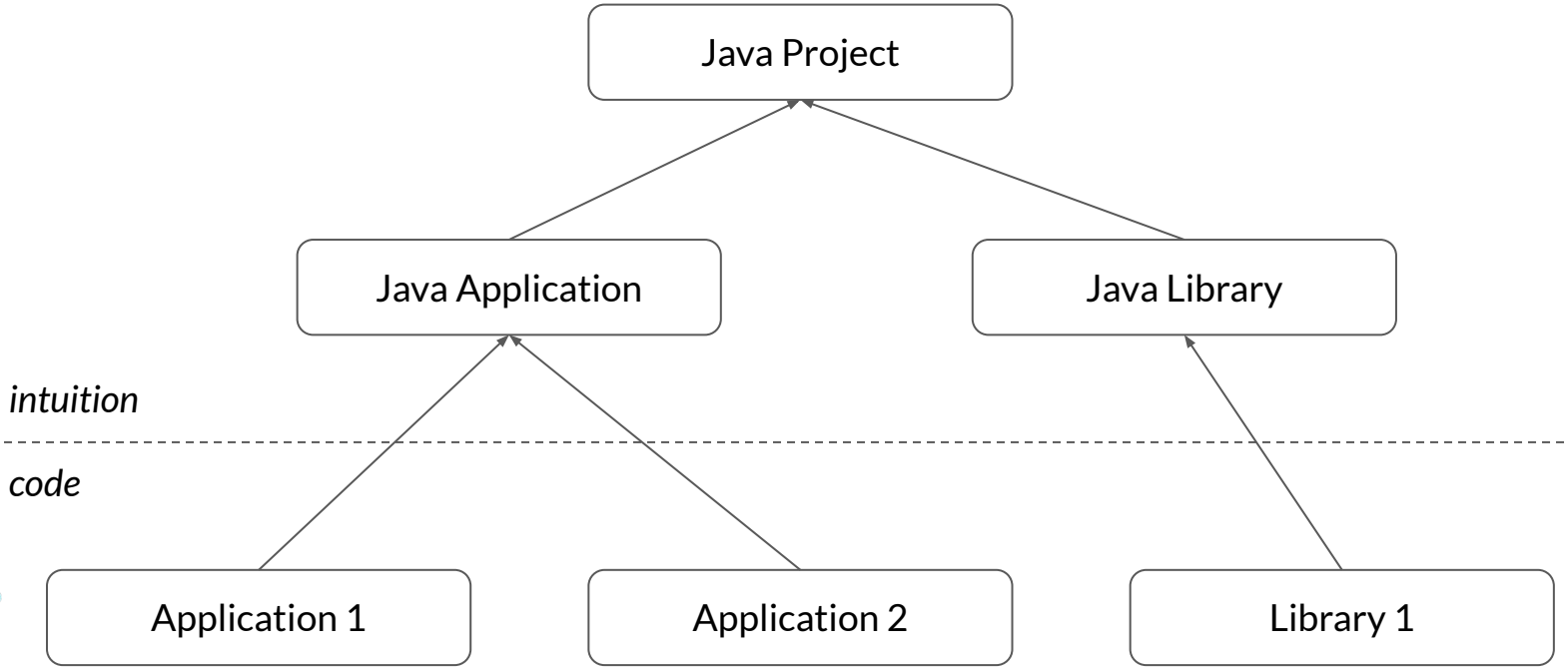
Convention Plugins



Multi-project build

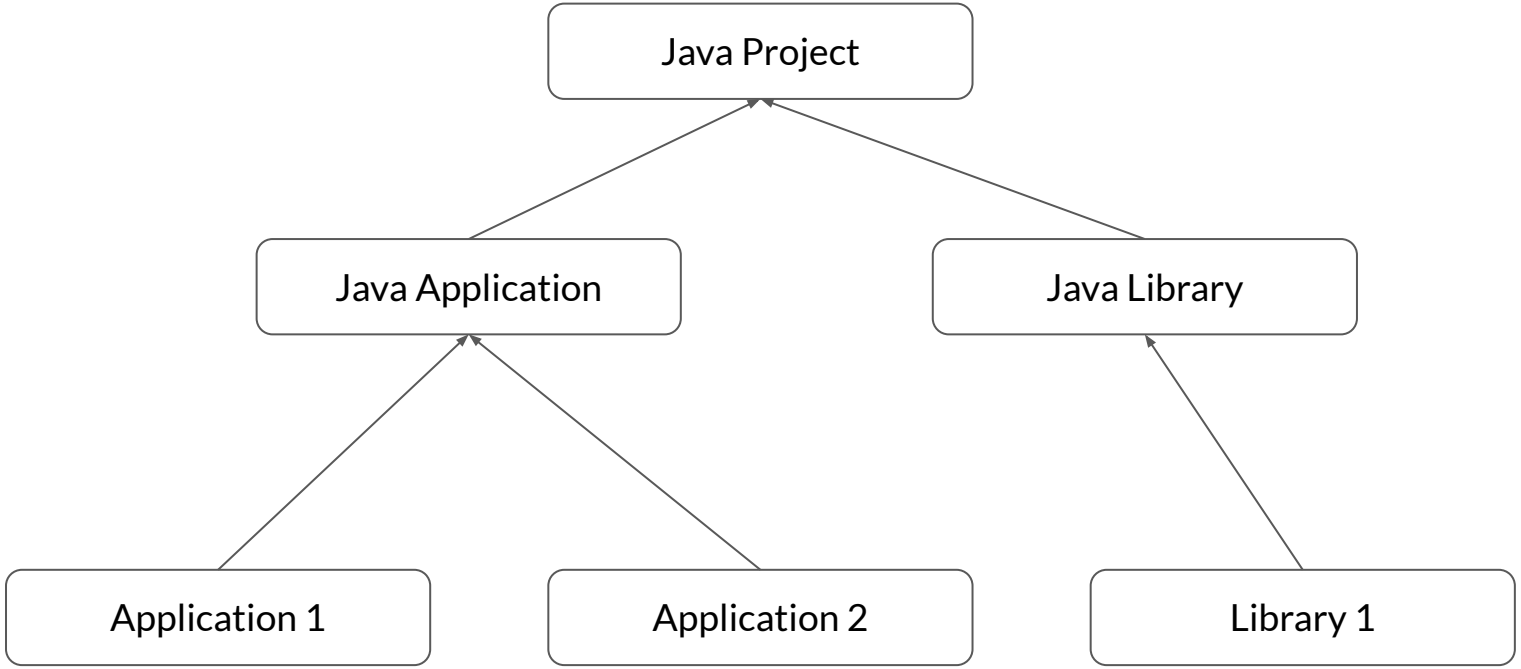

Ctrl
C
V

Multi-project build



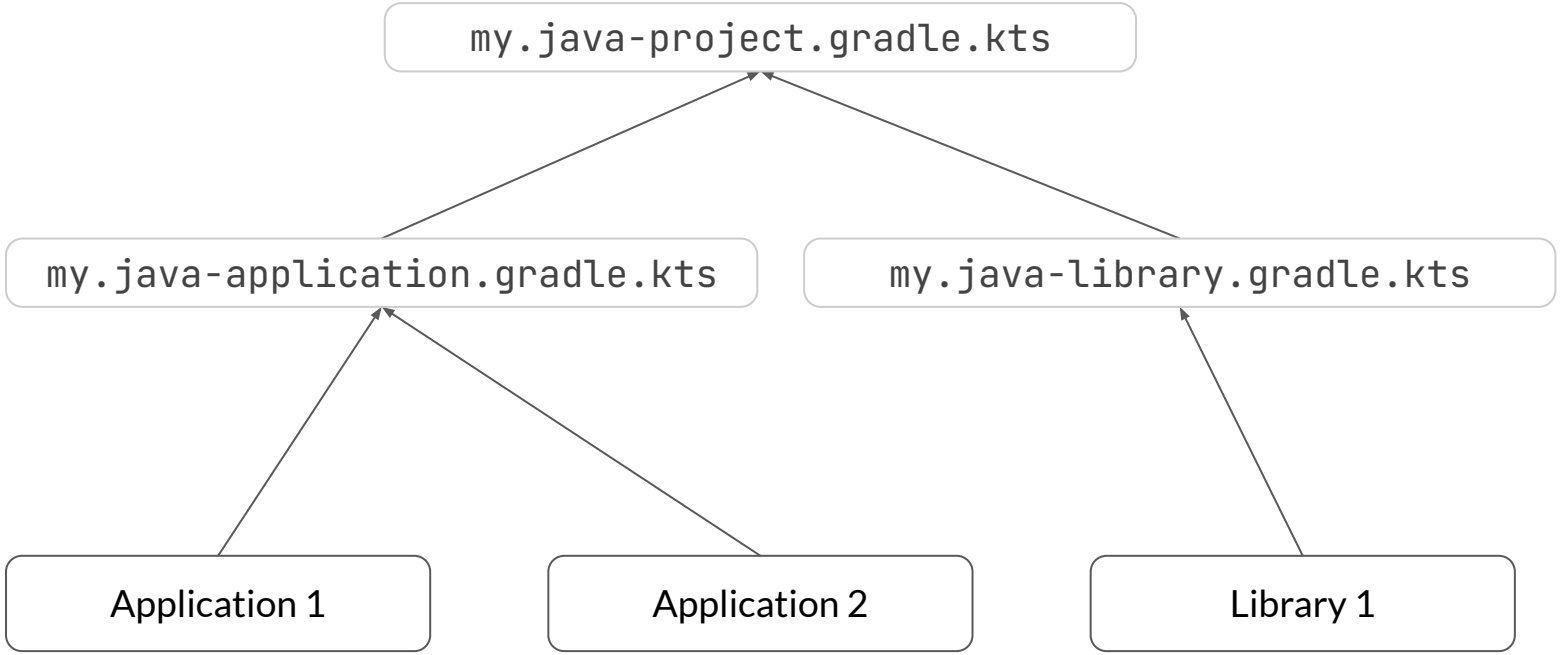
Multi-project build

code

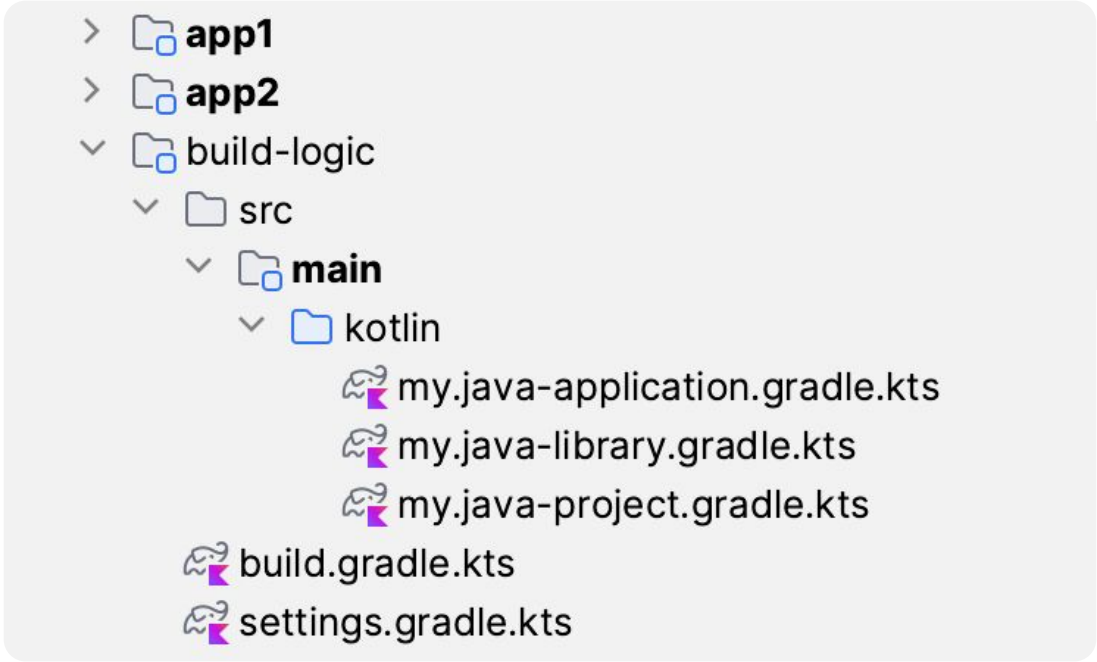


Convention Plugins

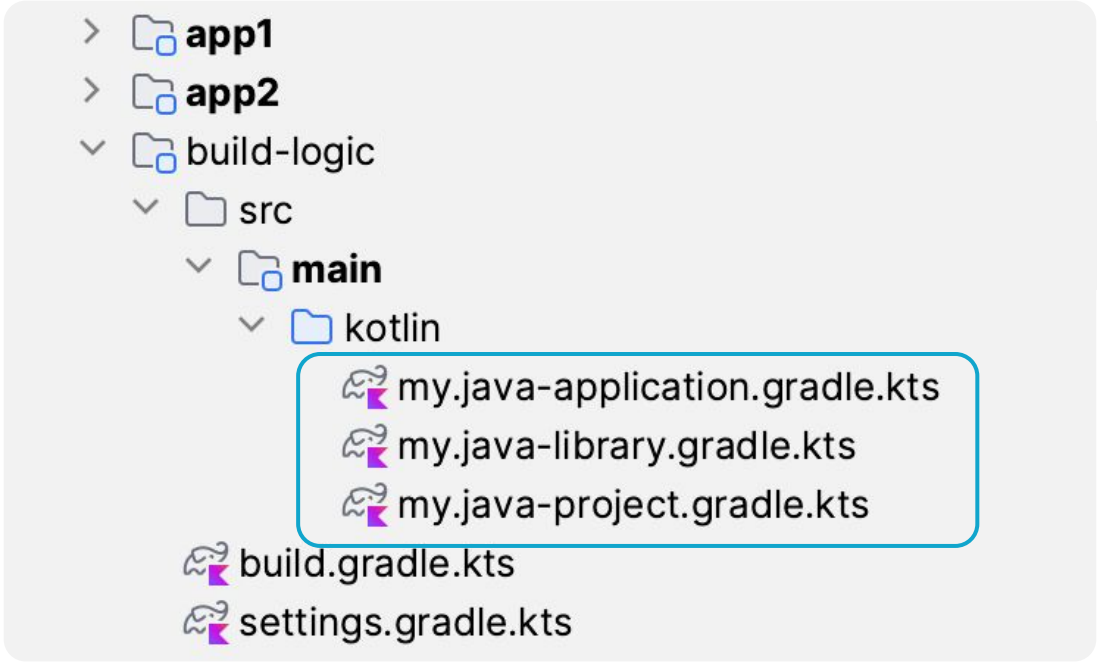
code



Build Logic Subproject



Build Logic Subproject



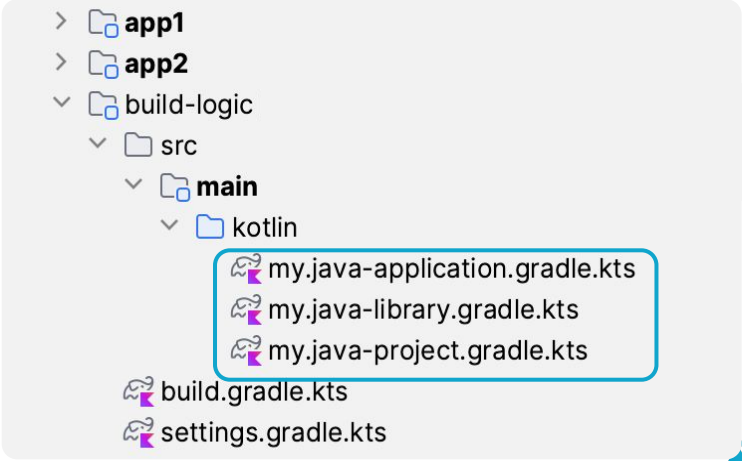
Extracting Build Logic

my.java-project.gradle.kts

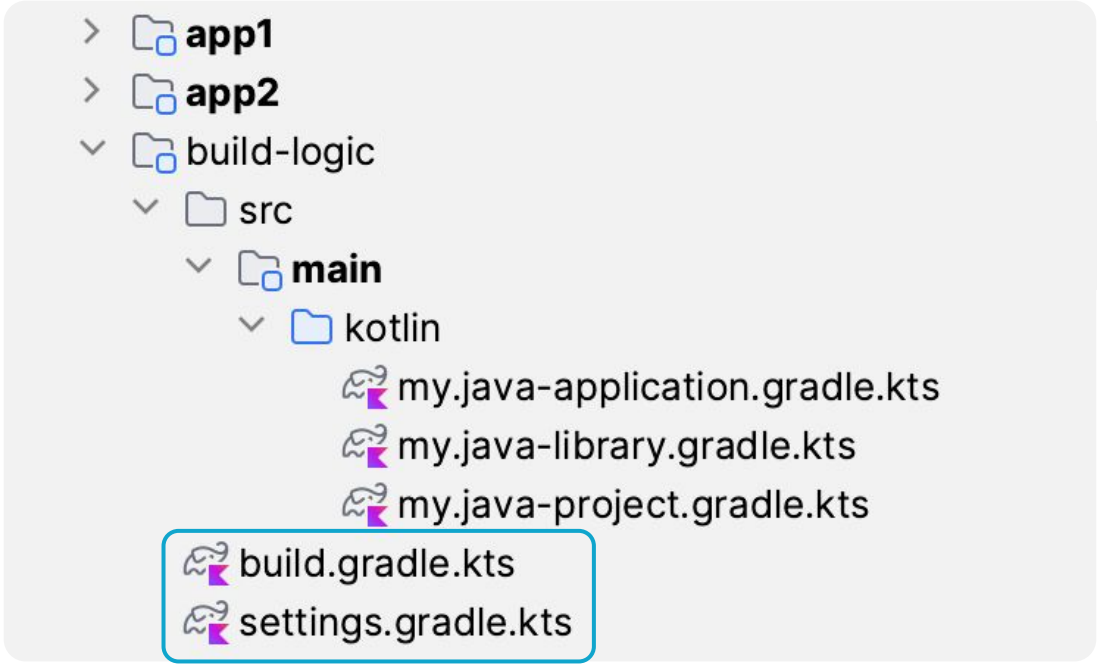
```
plugins {
    `java-base`
}

repositories {
    mavenCentral()
}

java {
    toolchain { /* ... */ }
}

testing {
    suites { /* ... */ }
}
```

my.java-application.gradle.kts

```
plugins {
    id("my.java-project")
    application
}
```



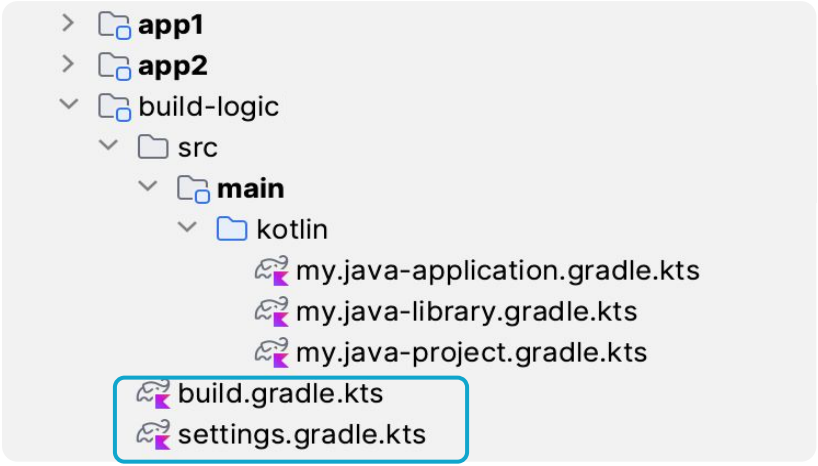
Build Logic Subproject



Extracting Build Logic

build-logic/settings.gradle.kts

```
rootProject.name = "build-logic"
```



build-logic/build.gradle.kts

```
plugins {
    `kotlin-dsl`
}

repositories {
    mavenCentral()
    gradlePluginPortal()
}
```

Including build logic

settings.gradle.kts

```
rootProject.name = "monorepo"

includeBuild("build-logic")

include("app1", "app2", "lib1")
```

app1/build.gradle.kts

```
plugins {
    id("my.java-application")
}

dependencies {
    // ...
}

application {
    mainClass = "my.App1"
}
```

Convention plugins

- Orchestrate applied plugins
- Configure defaults for you, your project, your company
- Inside the project or published

Composite Builds

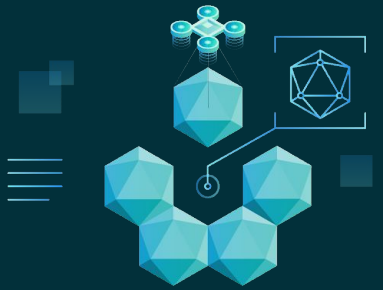
```
includeBuild("/path/to/lib/from/another/repo")
```

- Library changes are available directly in your project

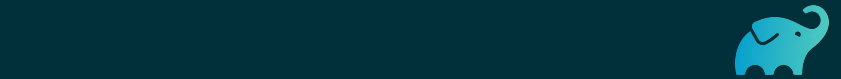
  without local publishing
- Including library as a temporary Gradle module in IDE provides

  cross-project navigation and refactorings
- Works via dependency substitution and

  supports substitution overrides



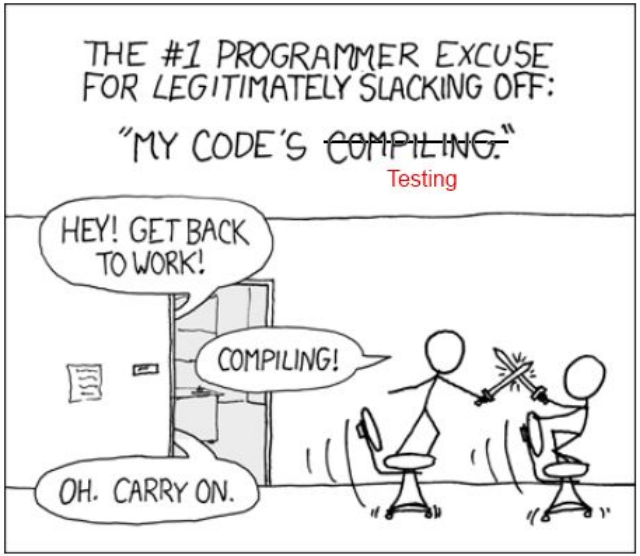
Performance



Anti-performance

https://xkcd.com/303/

Anti-performance



https://xkcd.com/303/

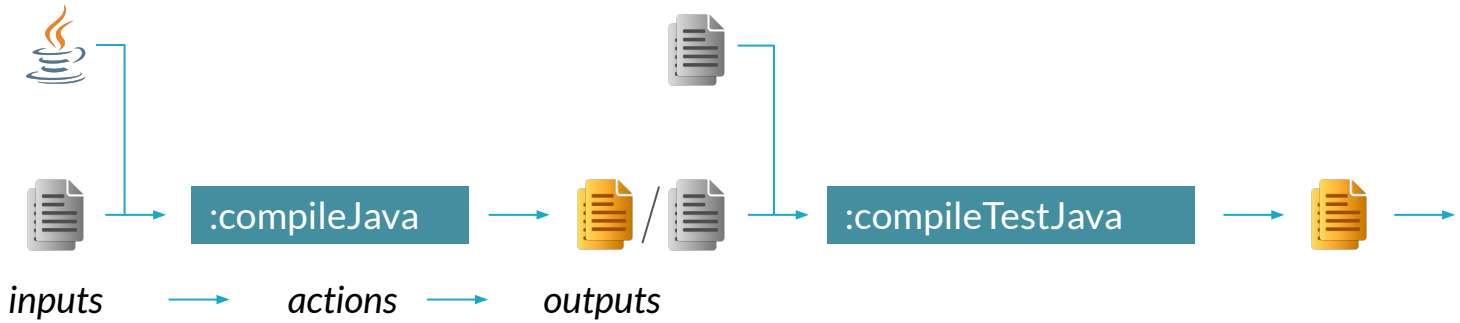
Performance Improvements

- Don’t do the same work again –

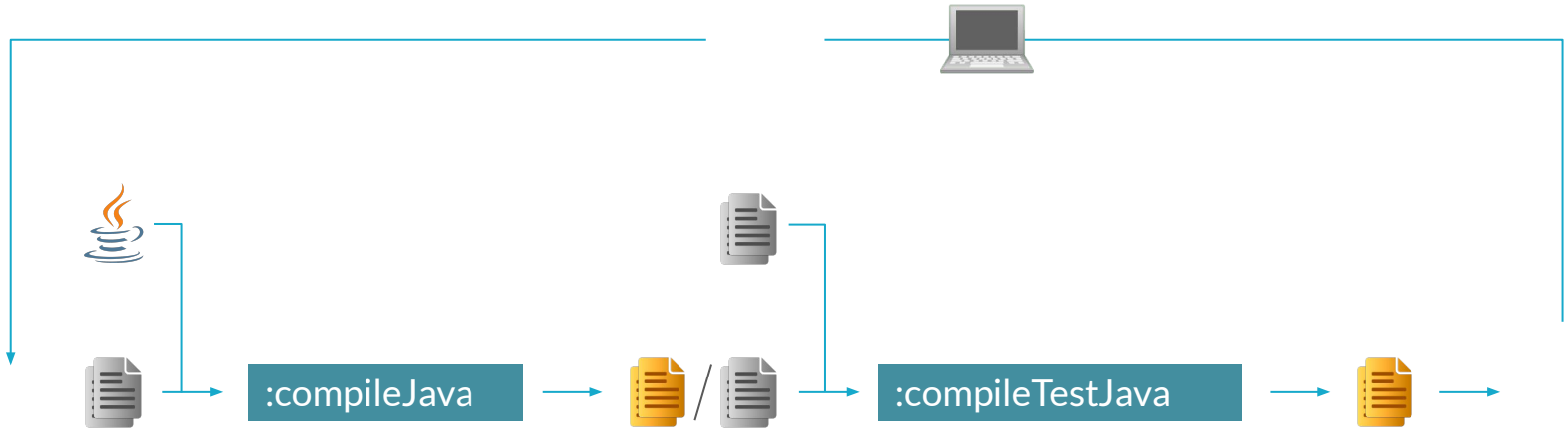
  work avoidance: incremental build/incremental tasks/caching
- Use more resources to do the work faster –

  run in parallel

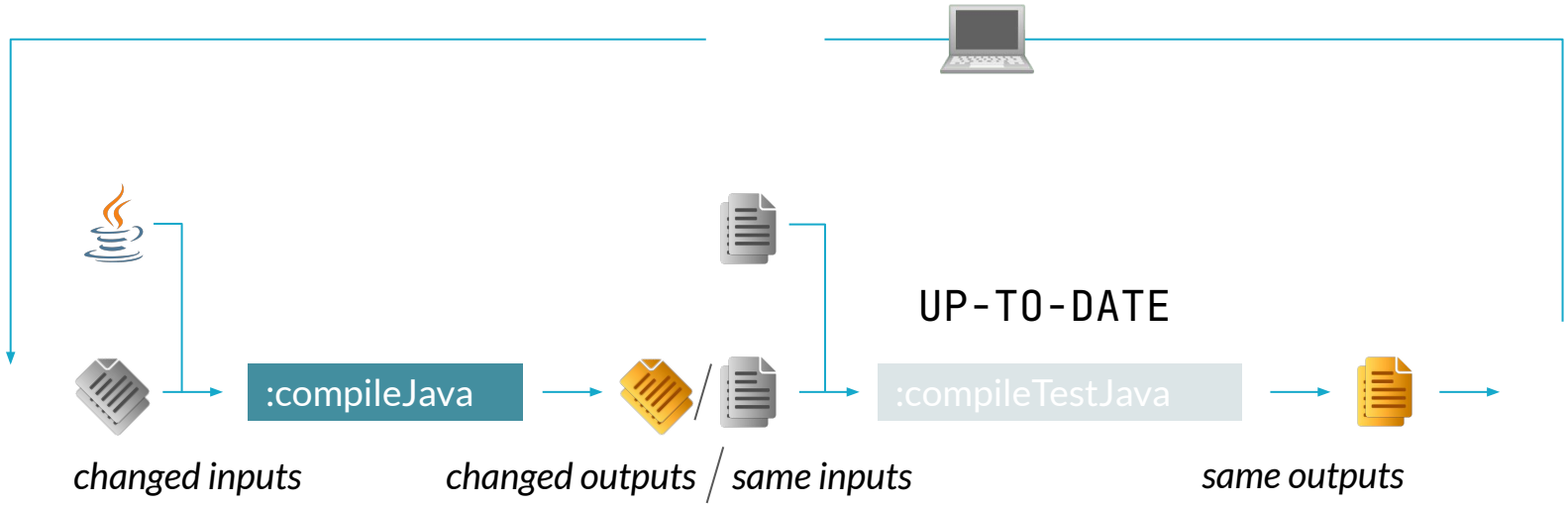
Gradle Tasks



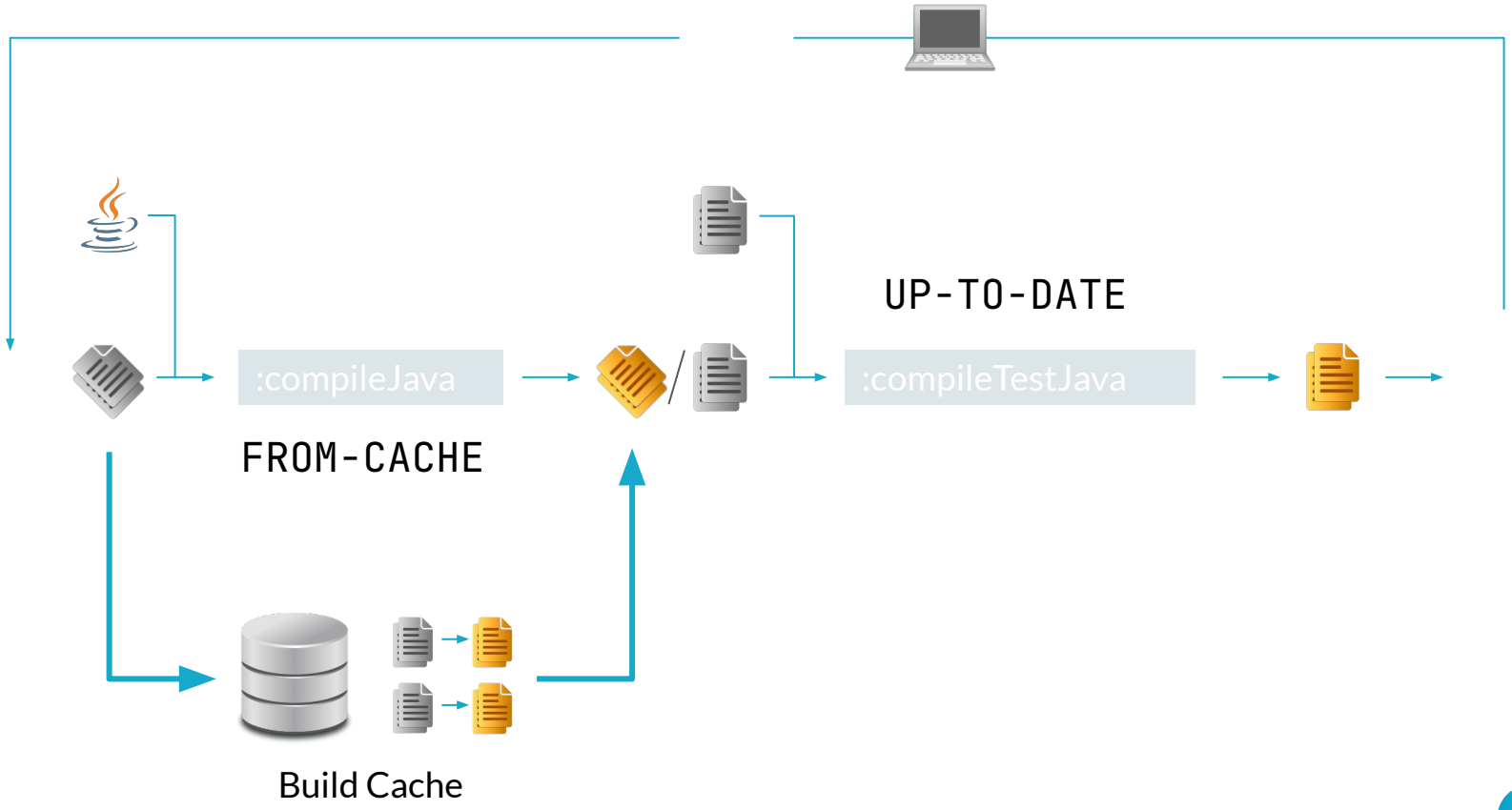
Gradle Tasks



Incremental Build



Build Cache


:compileJava
FROM-CACHE
UP-TO-DATE
:compileTestJava
Build Cache

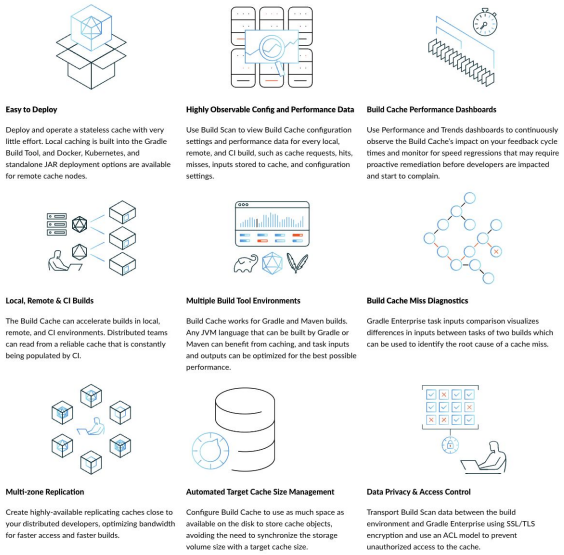
Build Cache

- Enable for single build invocation with `--build-cache`
- Enable for all builds via `gradle.properties`

```
org.gradle.caching=true
```

Remote Build Cache

- https://hub.docker.com/r/gradle/build-cache-node
- Better with Gradle Enterprise



Easy to Deploy

Deploy and operate a stateless cache with very little effort. Local caching is built into the Gradle Build Tool, and Docker, Kubernetes, and standalone JAR deployment options are available for remote cache nodes.

Highly Observable Config and Performance Data

Use Build Scan to view Build Cache configuration settings and performance data for every local, remote, and CI build, such as cache requests, hits, misses, inputs stored to cache, and configuration settings.

Build Cache Performance Dashboards

Use Performance and Trends dashboards to continuously observe the Build Cache's impact on your feedback cycle times and monitor for speed regressions that may require proactive remediation before developers are impacted and start to complain.

Local, Remote & CI Builds

The Build Cache can accelerate builds in local, remote, and CI environments. Distributed teams can read from a reliable cache that is constantly being populated by CI.

Multiple Build Tool Environments

Build Cache works for Gradle and Maven builds. Any JVM language that can be built by Gradle or Maven can benefit from caching, and task inputs and outputs can be optimized for the best possible performance.

Build Cache Miss Diagnostics

Gradle Enterprise task inputs comparison visualizes differences in inputs between tasks of two builds which can be used to identify the root cause of a cache miss.

Multi-zone Replication

Create highly-available replicating caches close to your distributed developers, optimizing bandwidth for faster access and faster builds.

Automated Target Cache Size Management

Configure Build Cache to use as much space as available on the disk to store cache objects, avoiding the need to synchronize the storage volume size with a target cache size.

Data Privacy & Access Control

Transport Build Scan data between the build environment and Gradle Enterprise using SSL/TLS encryption and use an ACL model to prevent unauthorized access to the cache.

Building in parallel

- Maximum parallelism `--max-workers=16`
  - Dependencies, artifact transforms, tasks using Worker API
- Parallelism between projects with `--parallel`
- Parallel test execution

```
tasks.test { maxParallelForks = 16 }
```

Understanding build execution

- Why did the build take this long?
- Which part or the build takes the most time?
- Were there any cache misses due to a misconfiguration?
- What was the historical performance of this test?

```
❯ gradle build --scan
```

publishes build scan to scans.gradle.com

Summary
Console log
Failure
Deprecations
Timeline
Performance
Tests
Projects
Dependencies
Build dependencies
Plugins
Custom values
Switches
Infrastructure

See before and after
Compare Build Scan

771 tasks executed in 97 projects in 13.089s, with 488 avoided tasks saving 4m 41.921s

Order: Execution | By task path | By task type

:configuration-cache:compileIntegTestGroovy	FROM-CACHE	2.992s	0.262s	org.gradle.api.tasks.compile.GroovyCompile
:configuration-cache:integTestClasses	UP-TO-DATE	3.254s	0.000s	org.gradle.api.DefaultTask

:configuration-cache:embeddedIntegTest ...tionTest

Showing 601-771 out of 771 total items

Details | Predecessors | Successors

Path :configuration-cache:embeddedIntegTest
Type gradlebuild.integrationtests.tasks.IntegrationTest

This task is on the critical path.

Started after 3.255s
Duration 9.698s

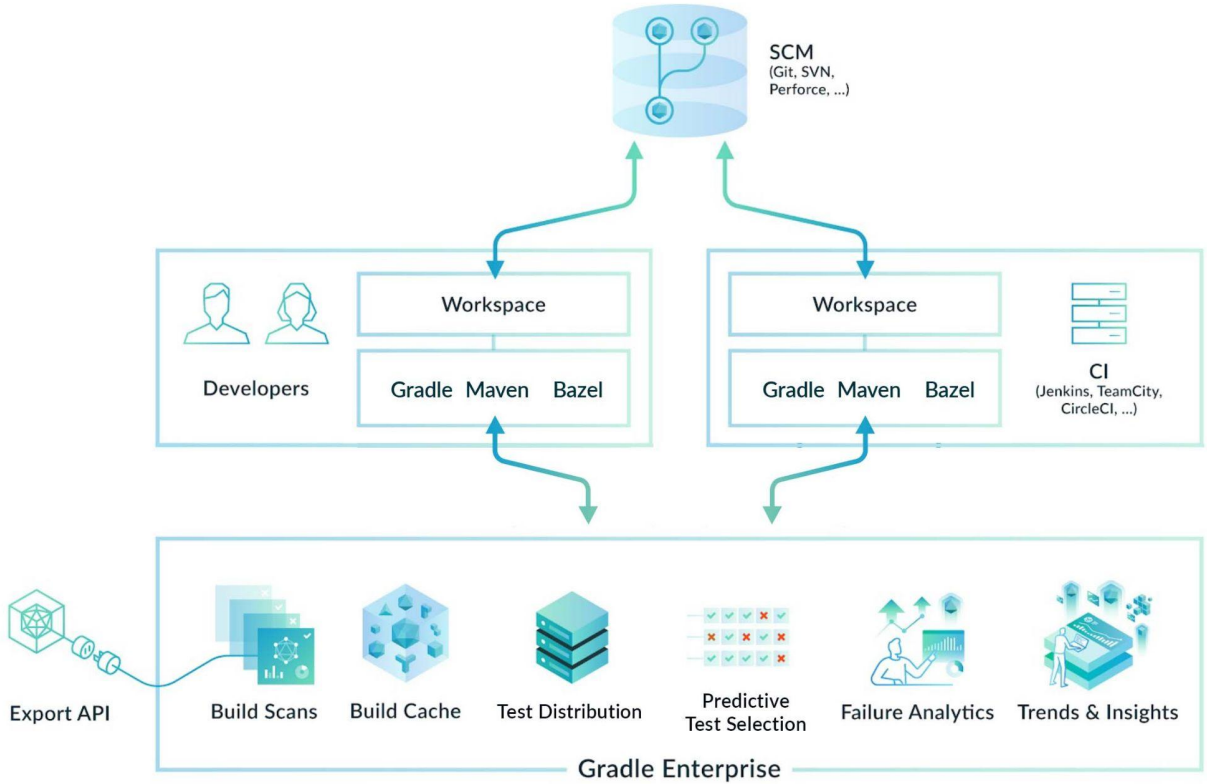
The task was not up-to-date because Task.upToDateWhen was false.

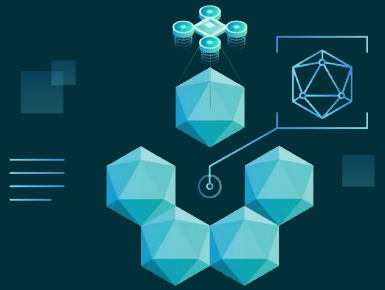
Build cache result Store (local)

Java toolchain Oracle 11.0.2+9 (x86_64)

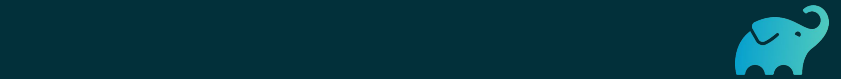
View task in console log Focus on task in timeline

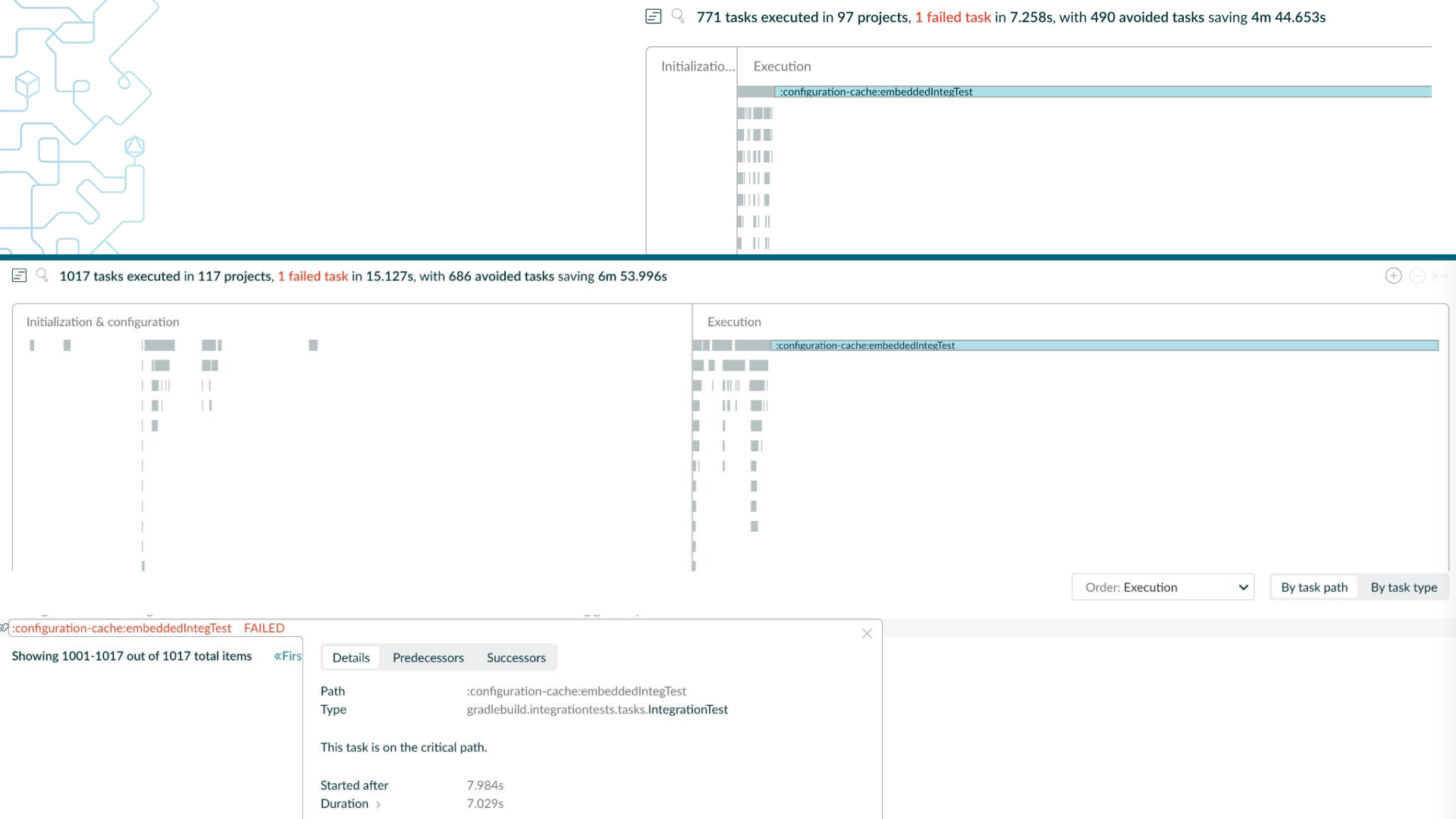
Gradle Enterprise





Configuration Cache

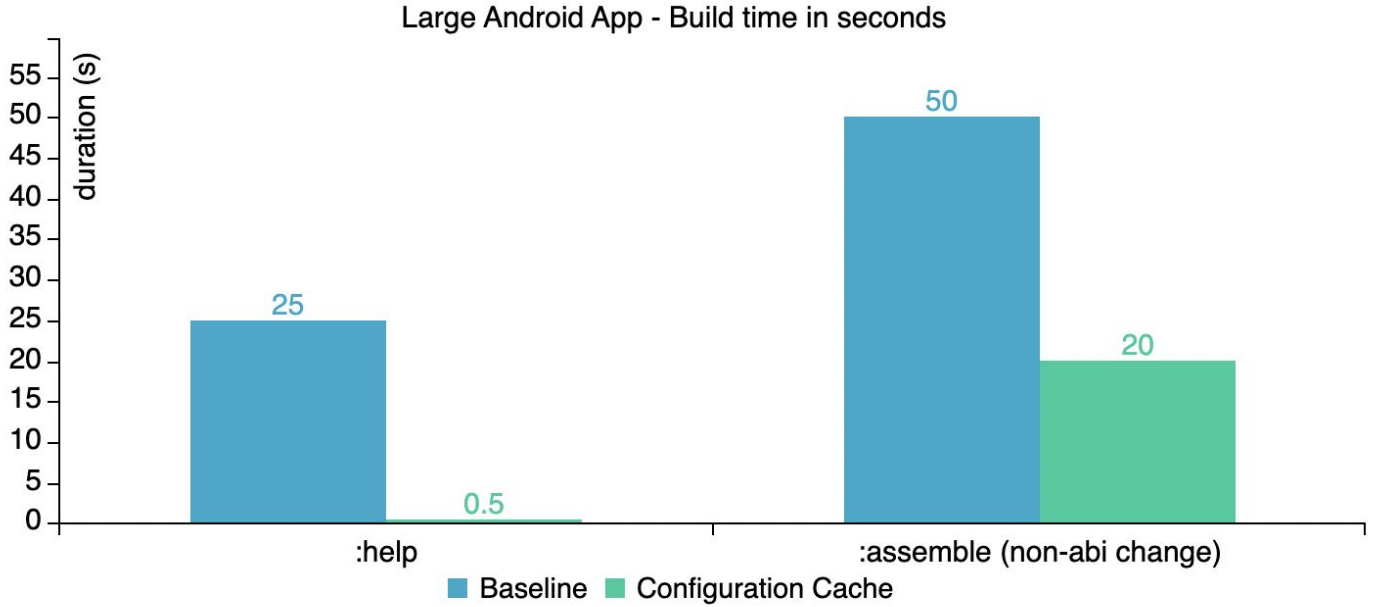



771 tasks executed in 97 projects, 1 failed task in 7.258s, with 490 avoided tasks saving 4m 44.653s
Initializatio...
Execution
:configuration-cache:embeddedIntegTest
1017 tasks executed in 117 projects, 1 failed task in 15.127s, with 686 avoided tasks saving 6m 53.996s
Initialization & configuration
Execution
:configuration-cache:embeddedIntegTest
Order: Execution
By task path
By task type
:configuration-cache:embeddedIntegTest FAILED
Showing 1001-1017 out of 1017 total items
«Firs
Details
Predecessors
Successors
Path :configuration-cache:embeddedIntegTest
Type gradlebuild.integrationtests.tasks.IntegrationTest
This task is on the critical path.
Started after 7.984s
Duration 7.029s

Configuration Cache

- Caches the result of the configuration and the task graph
  - When nothing changed, the whole configuration phase is skipped
- Detects build logic inputs for invalidation
- Task isolated from the mutable model and from each other
  - Executes *all* tasks in parallel (incl. intra-projects)

Configuration Cache



Configuration Cache

- Enable for single build invocation with `--configuration-cache`
- Enable for all builds via Gradle property

  `org.gradle.configuration-cache=true`
- Report failures as warnings with Gradle property

  `org.gradle.configuration-cache=warn`

Configuration Cache Compatibility

Forces good practices

- Downside: You'll probably need to change your build
- Clear separation between configuration and execution
- Correct declaration of inputs
- No cross-dependencies between tasks

CC Compatibility

- Core JVM plugins ✅
- Other core plugins ⏳
- Kotlin ✅
- Android ✅
- Community Plugins 🌈

JVM languages and frameworks	Native languages	Packaging and distribution
✓ Java	✖ C++ Application	✓ Application
✓ Java Library	✖ C++ Library	✓ WAR
✓ Java Platform	✖ C++ Unit Test	✖ EAR
✓ Groovy	✖ Swift Application	⚠ Maven Publish
✓ Scala	✖ Swift Library	⚠ Ivy Publish
✓ ANTLR	✖ XCTest	✓ Distribution
		✓ Java Library Distribution

Code analysis	IDE project files generation	Utility
✓ Checkstyle	✖ Eclipse	✓ Base
✓ CodeNarc	✖ IntelliJ IDEA	✓ Build Init
✓ JaCoCo	✖ Visual Studio	✓ Signing
✓ JaCoCo Report Aggregation	✖ Xcode	⚠ Java Plugin Development
✓ PMD		✓ Groovy DSL Plugin Development
✓ Test Report Aggregation		✓ Kotlin DSL Plugin Development
		✓ Project Report Plugin

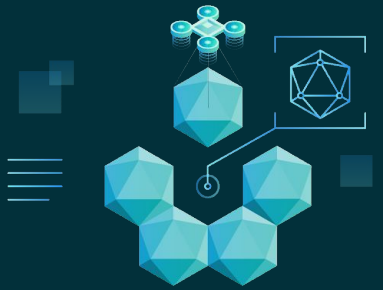
Configuration Cache Roadmap

- Stable since Gradle 8.1

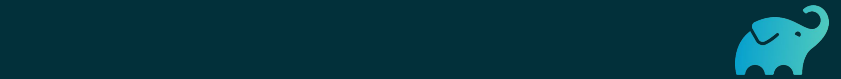
  (and opt-in)
- Activated by default in Gradle 9.0

  (with opt-out)
- Only mode in Gradle x.x

  (without opt-out)



What is next?



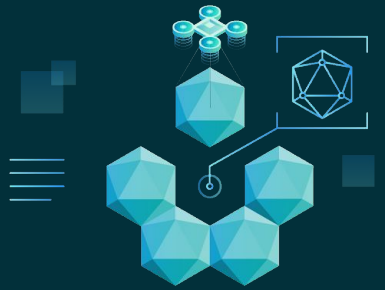
What's next

- Faster IDE Sync: Isolated Projects
- Public Roadmap
- gradle.org

Thank you!

wolf@gradle.com

Gradle



Java Toolchains

Java Toolchains

build.gradle.kts

```
plugins {
    application
    // ⇒ java
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
        vendor = JvmVendorSpec.ADOPTIUM
    }
}

val testJavaVersion: String by project

tasks.withType<Test>().configureEach {
    javaLauncher = javaToolchains.launcherFor {
        languageVersion = JavaLanguageVersion.of(testJavaVersion)
    }
}
```

Which Java toolchain does Gradle detect?

- Autodetected defaults:
  - Per OS: Linux, macOS, Windows
  - Package managers: Asdf-vm, Jabba, SDKMAN!
  - Maven toolchains
- Explicit configuration:
  - `org.gradle.java.installations.fromEnv`
  - `org.gradle.java.installations.paths`
- Automatic toolchain download
  - foojay Disco API

Auto Provisioning

settings.gradle.kts

```
plugins {
  id("org.gradle.toolchains.foojay-resolver-convention")
}
```

Download toolchains using the Foojay Disco API