Sane and fast builds for projects of any scale

Alex Semin

W-JAX Munich 2022

About me

Alex Semin

Senior Engineer @ Gradle

Formerly @ dxFeed

Twitter @alllexist, GitHub @alllex

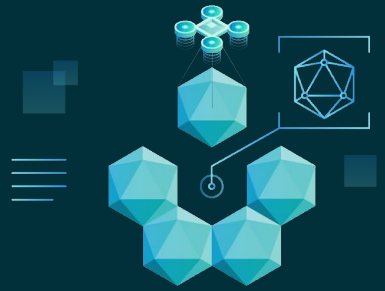
What is Gradle

Gradle Inc.

Gradle Build Tool

Gradle Enterprise

Table of Contents



Fundamentals

Gradle Init

```
❯ gradle init
```

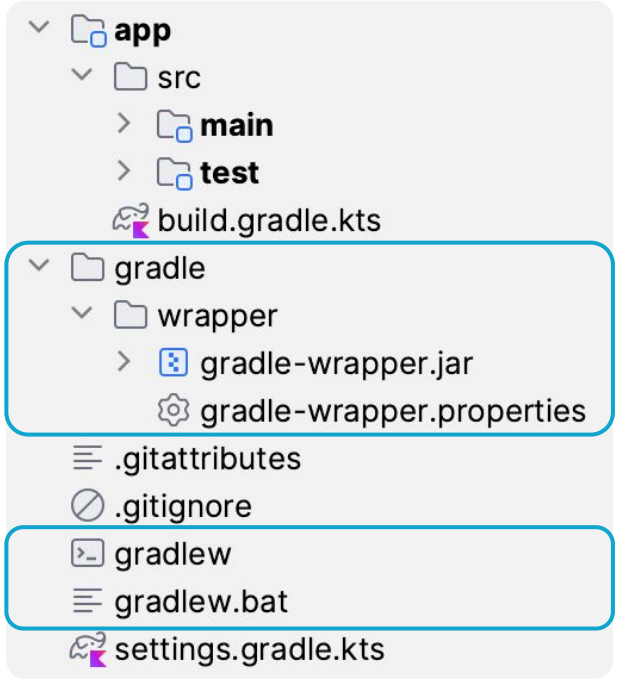
- Create a project via an interactive dialog
- Allows to configure
  - Project template
  - Test framework
  - Gradle setup

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

Project anatomy

- **app**
  - src
    - **main**
    - **test**
    - build.gradle.kts
- gradle
  - wrapper
    - gradle-wrapper.jar
    - gradle-wrapper.properties
  - .gitattributes
  - .gitignore
  - gradlew
  - gradlew.bat
  - settings.gradle.kts

Project anatomy



Gradle Wrapper

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

```
❯ ./gradlew wrapper --gradle-version 7.6
```

Project anatomy

- app
  - src
    - main
    - test
  - build.gradle.kts
- gradle
  - wrapper
    - gradle-wrapper.jar
    - gradle-wrapper.properties
- .gitattributes
- .gitignore
- gradlew
- gradlew.bat
- settings.gradle.kts

Build settings

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

include("app")
```

Project anatomy

- app
  - src
    - main
    - test
  - build.gradle.kts
- gradle
  - wrapper
    - gradle-wrapper.jar
    - gradle-wrapper.properties
- .gitattributes
- .gitignore
- gradlew
- gradlew.bat
- settings.gradle.kts

Build script

```
plugins {
    application
}

repositories {
    mavenCentral()
}

dependencies {
    // ...
}

application {
    mainClass.set("my.App")
}
```

Ready for development

- **app**
  - src
    - **main**
    - **test**
  - build.gradle.kts
- gradle
  - wrapper
    - gradle-wrapper.jar
    - gradle-wrapper.properties
- .gitattributes
- .gitignore
- gradlew
- gradlew.bat
- settings.gradle.kts

Compile sources

```
❯ ./gradlew classes
```

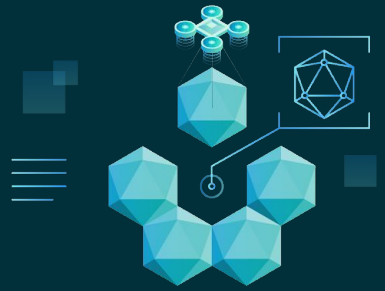
Execute tests

```
❯ ./gradlew check
```

Build full project

```
❯ ./gradlew build
```

Which Java version are we building for?



Java Toolchains

Java Toolchains

build.gradle.kts

```
plugins {
    application
    // ⇒ java
}

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

val testJavaVersion: String by project

tasks.withType<Test>().configureEach {
    javaLauncher.set(javaToolchains.launcherFor {
        languageVersion.set(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
  - Adoptium / AdoptOpenJDK distributions

Java Provisioning

settings.gradle.kts

```
toolchainManagement {
    jvm {
        javaRepositories {
            repository("adoptium") {
                resolverClass.set(
                    AdoptiumResolver::class.java
                )
            }
        }
    }
}
```

Coming soon: plugin to download toolchains based on Foojay Disco API

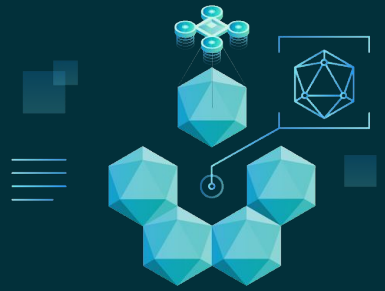
Testing a project

- Unit tests
- Property-based 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 Suits

Test Suites

build.gradle.kts

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

Test Suites

build.gradle.kts

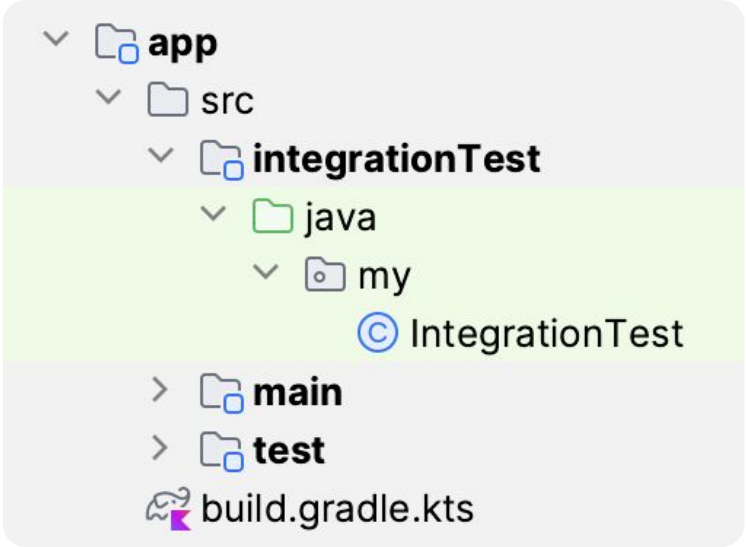
```
testing {
    suites {
        val test by getting(JvmTestSuite::class) {
            useJUnitJupiter()
        }

        val integrationTest by registering(JvmTestSuite::class) {
            dependencies {
                implementation(project())
            }
            useJUnitJupiter("5.8.2")
            targets.all { testTask.configure { shouldRunAfter(test) } }
        }
    }
}
```

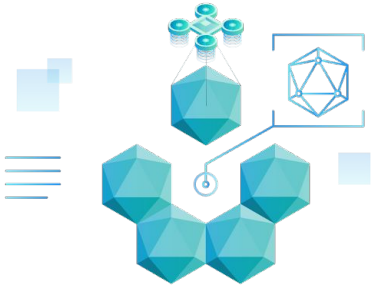
Test Suites

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() {
        // ...
    }
}
```

Multi-project build

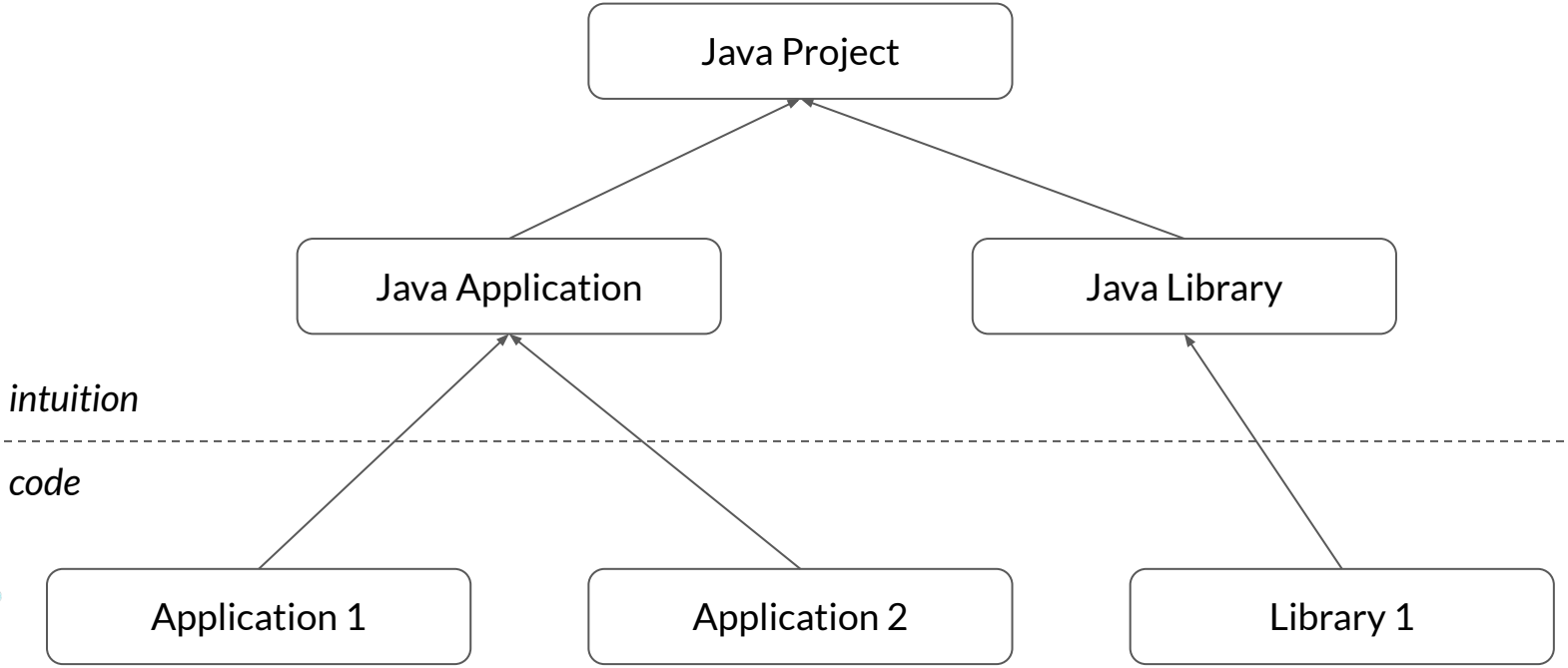
Application 1

Application 2

Library 1

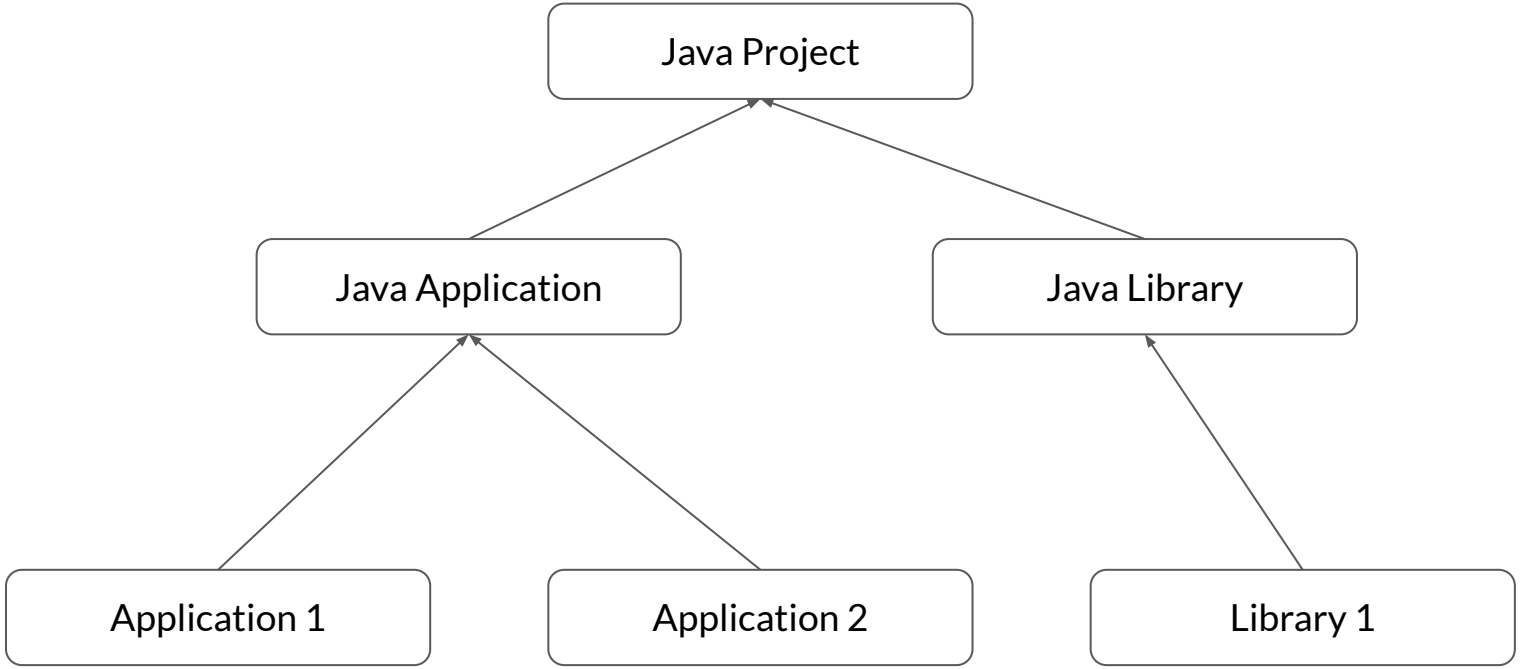

Ctrl
C
V

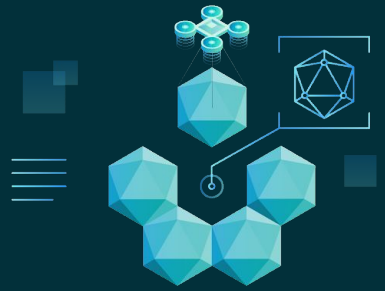
Multi-project build



Multi-project build

code

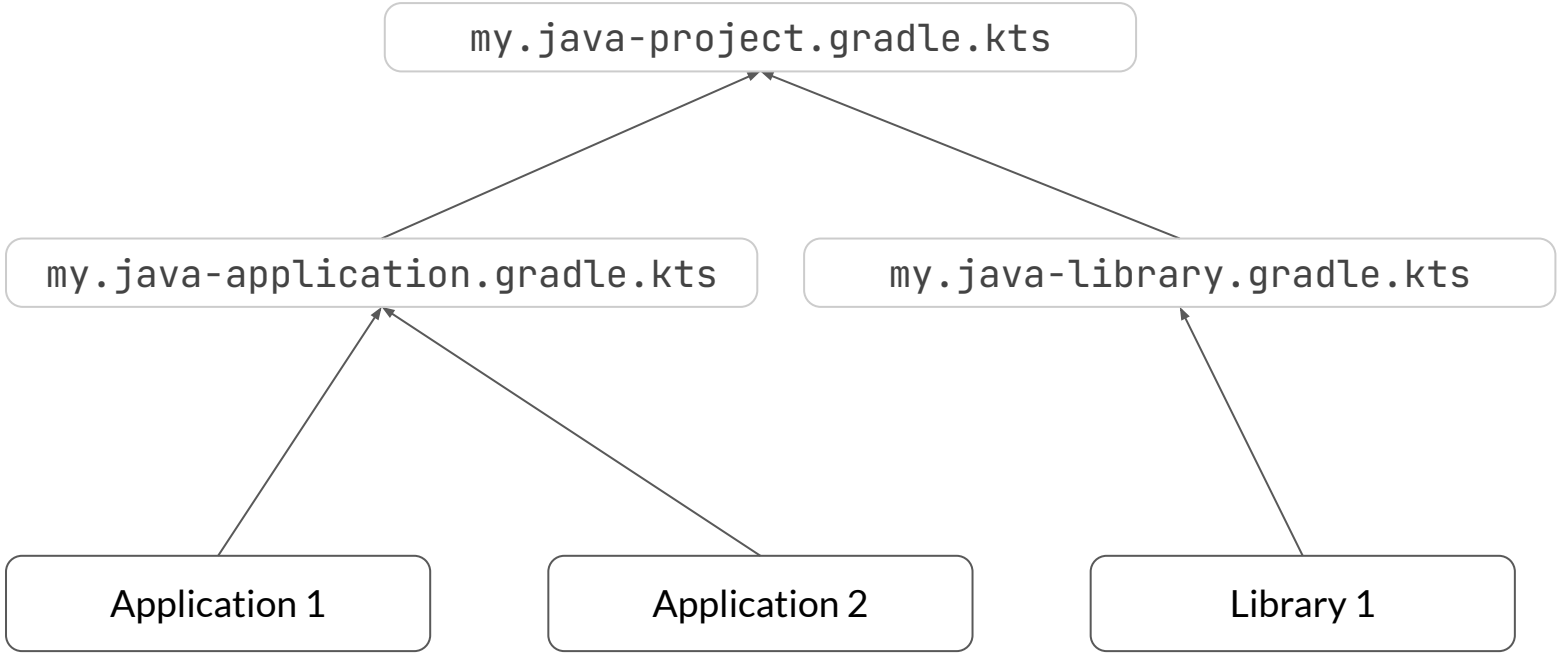




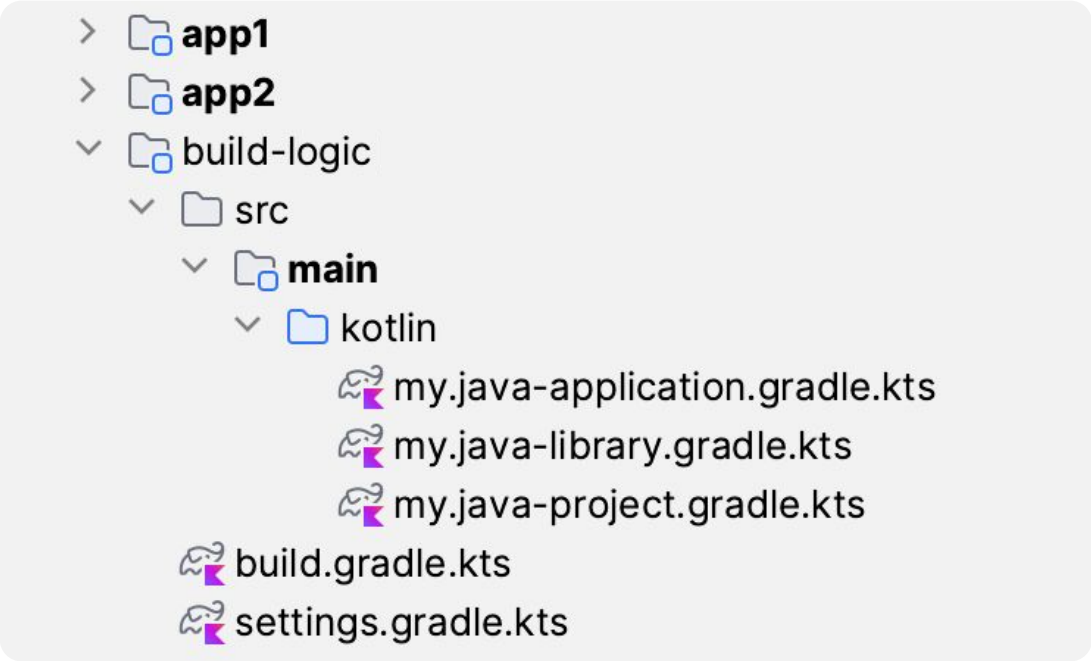
Convention Plugins

Convention Plugins

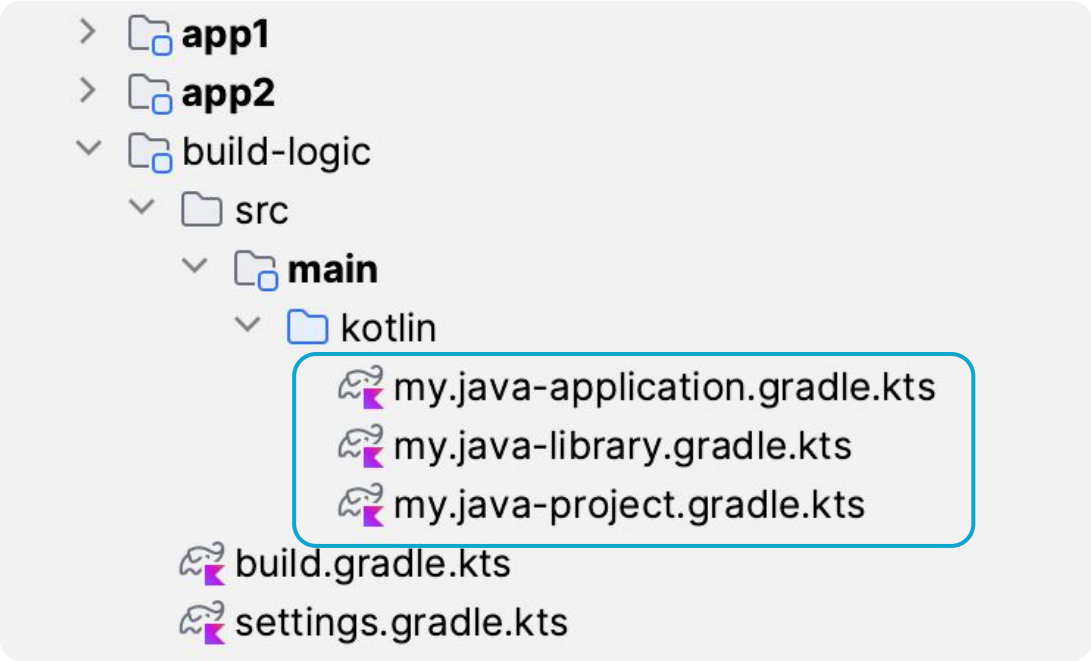
code



Build Logic Subproject



Build Logic Subproject



Extracting Build Logic

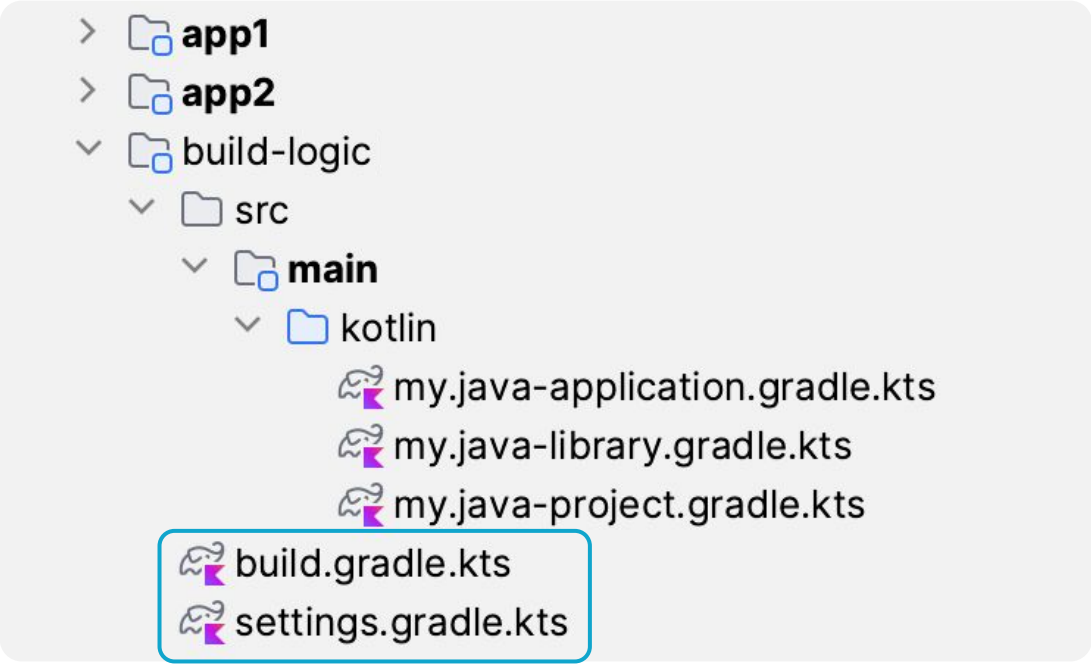
my.java-project.gradle.kts

```
plugins {
    java
}

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 {
    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.set("my.App1")
}
```

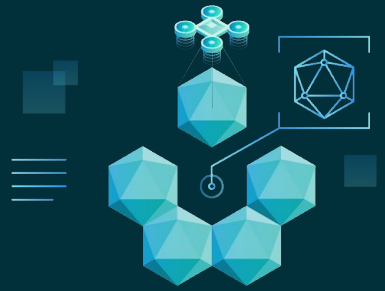
Convention plugins

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

Composite Builds

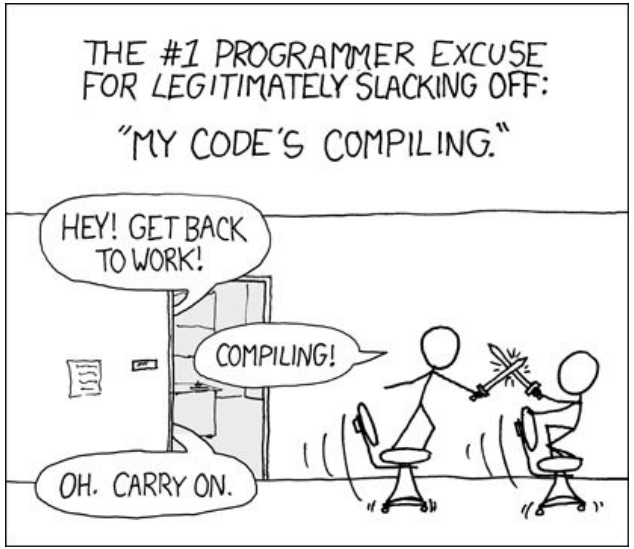
```
includeBuild("/path/to/lib/from/another/repo")
```

- 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

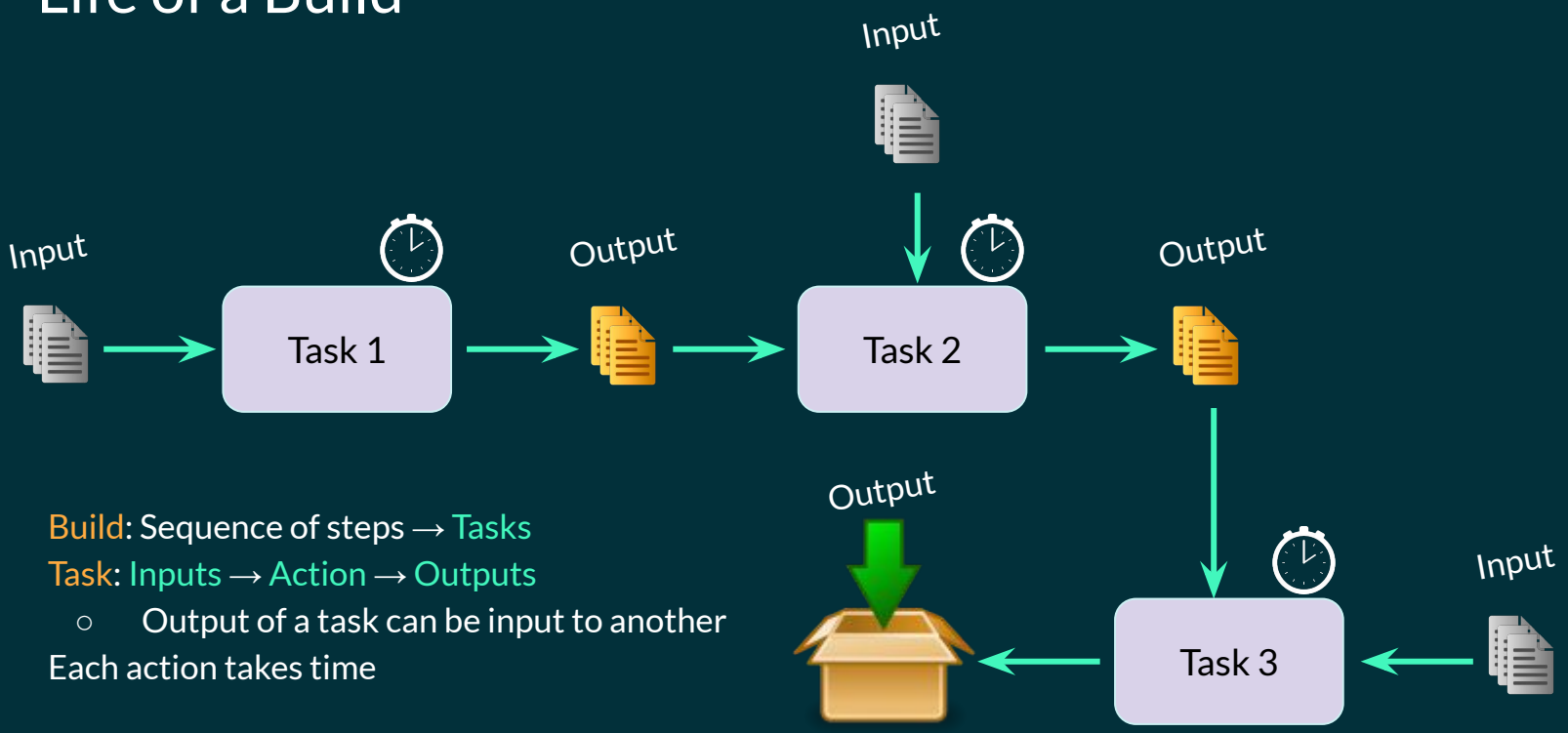


https://xkcd.com/303/

Performance Improvements

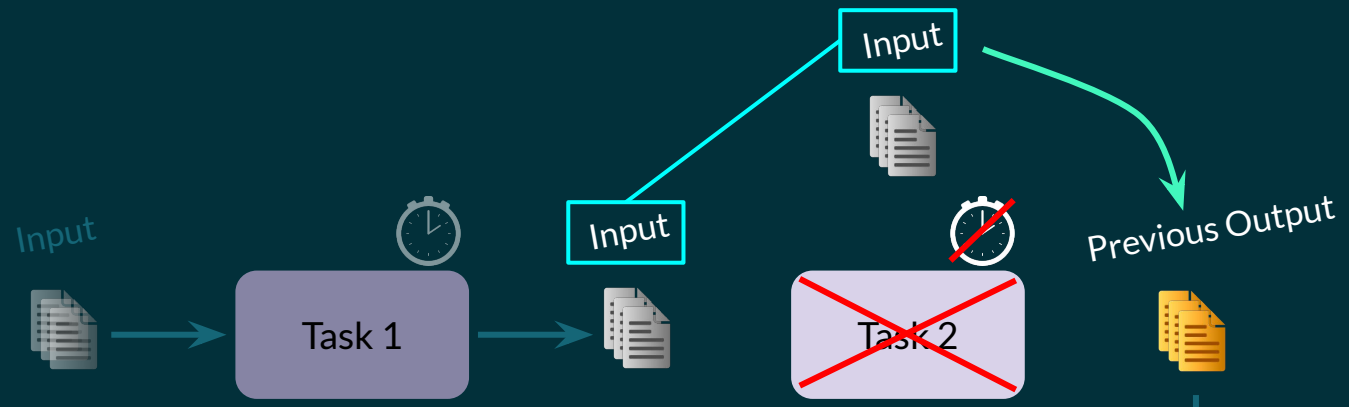
- Don’t do the same work again – *work avoidance*
- Don’t do the whole work if part is enough – *incremental builds*
- Use more resources to do the work faster – *run in parallel*

Life of a Build

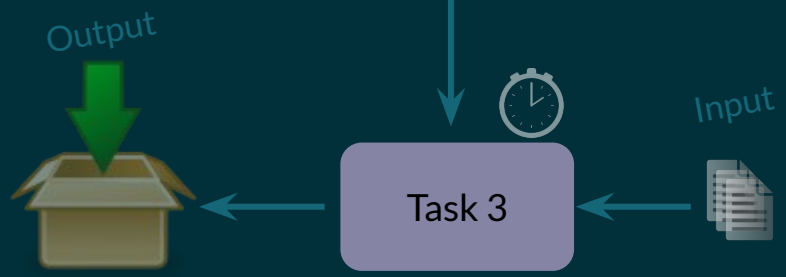


- Build: Sequence of steps → Tasks
- Task: Inputs → Action → Outputs
  - Output of a task can be input to another
- Each action takes time

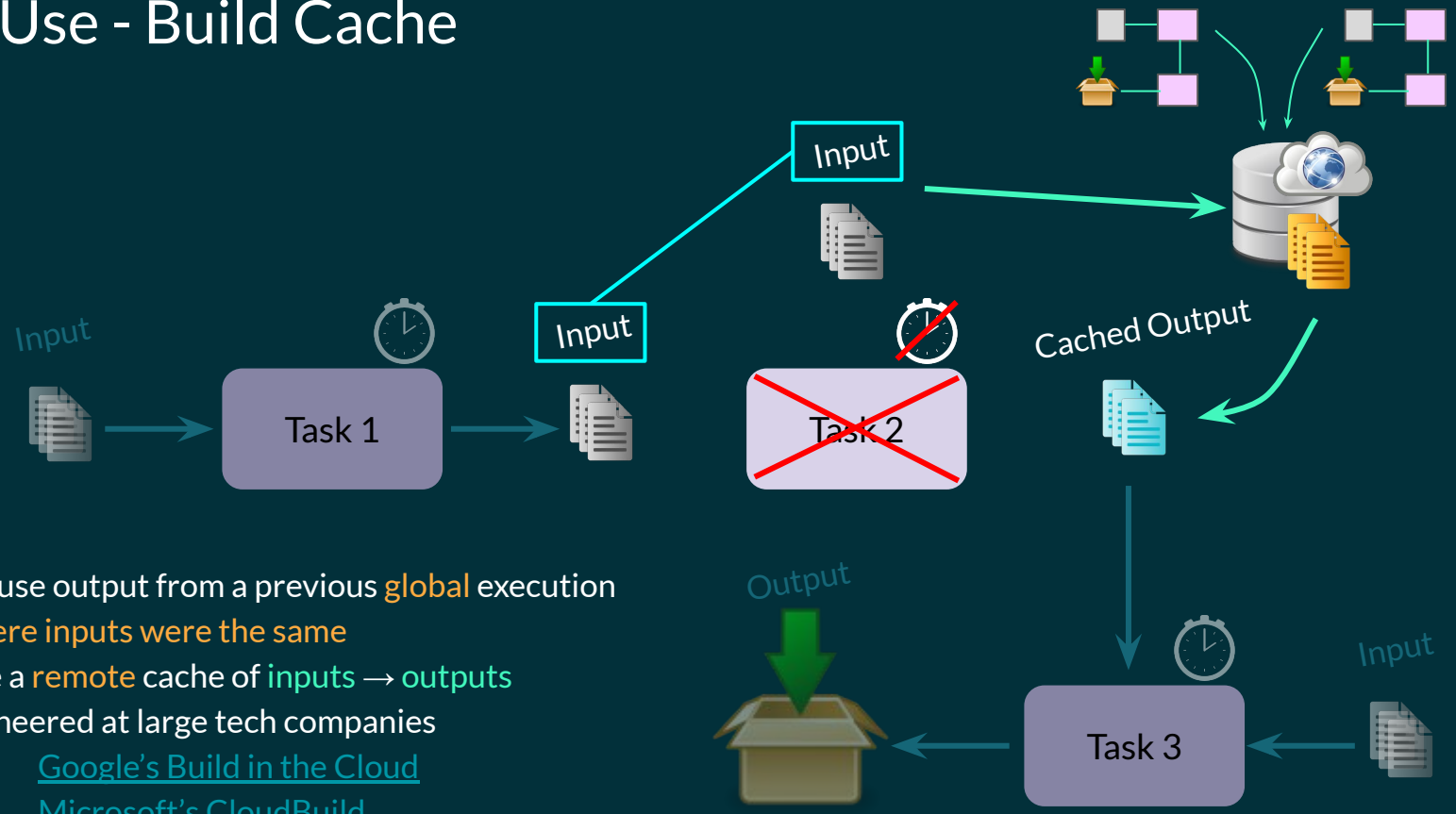
Re-Use - Incremental Build



- If task is deterministic and inputs haven't changed, re-use output from previous execution
  - Save time from running action
- Limitation: Only uses immediate previous build



Re-Use - Build Cache



- Re-use output from a previous global execution where inputs were the same
- Use a remote cache of inputs → outputs
- Pioneered at large tech companies
  - Google's Build in the Cloud
  - Microsoft's CloudBuild

Local Build Cache

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

  `org.gradle.caching=true`
- Configure location and retention policy:

settings.gradle.kts

```
buildCache {
    local {
        directory = File(rootDir, "build-cache")
        removeUnusedEntriesAfterDays = 30
    }
}
```

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

Gradle Enterprise

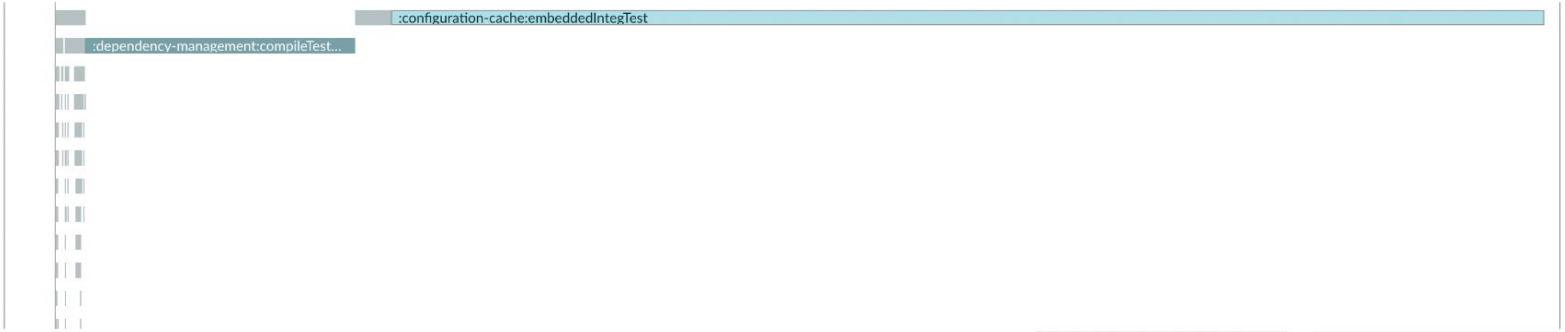
g... :configuration-cache:embeddedIntegTest --tests org.gradle.configurationcache.ConfigurationCacheD... 2. Nov. 2022 15:51:42 CET

Build Scans

- 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

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.

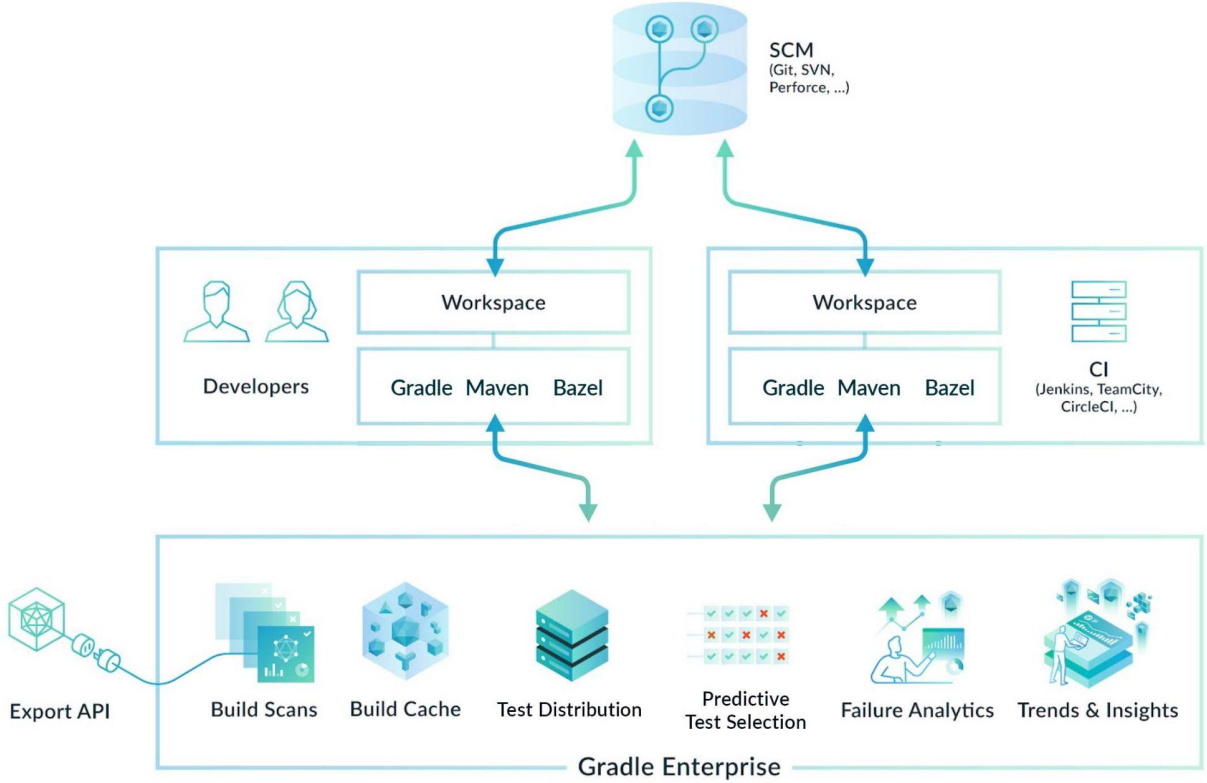
Started after	3.255s
Duration	9.698s

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



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

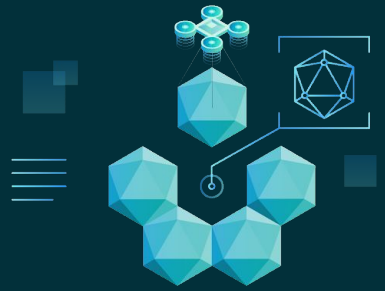
Showing 1001-1017 out of 1017 total items «Firs

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

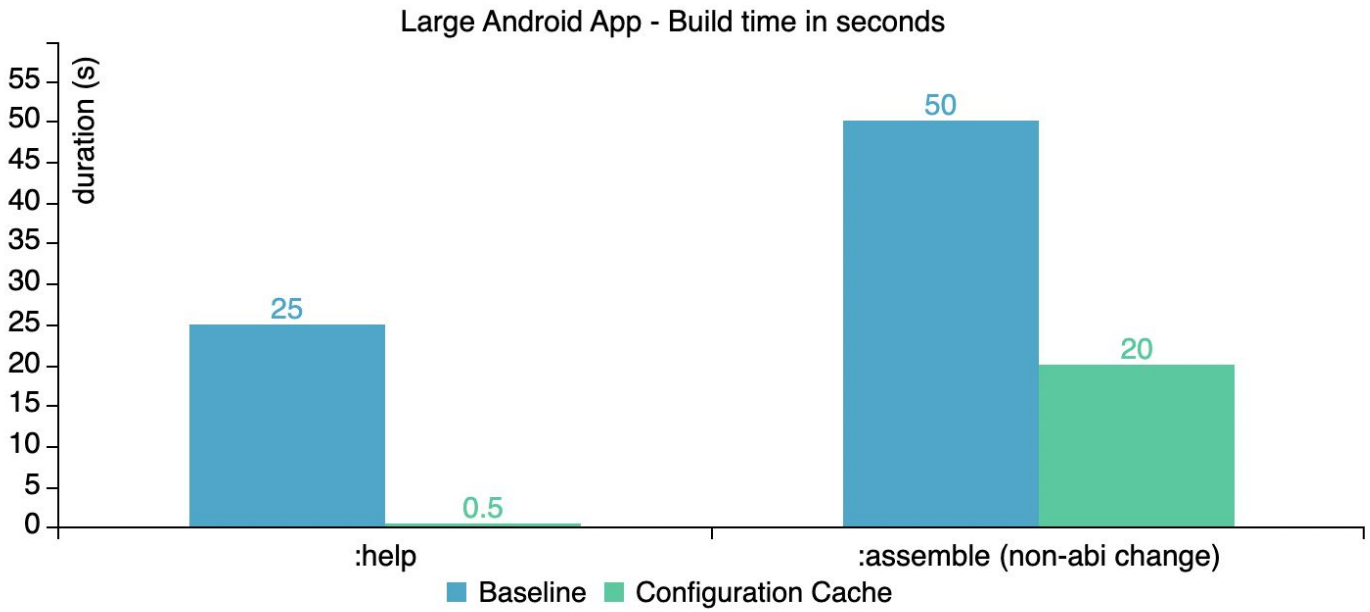
Configuration Cache Principles

- Caches the result of the configuration and computation of the task graph, and reuse it for subsequent builds.
- Detects build logic inputs for invalidation
- Task isolated from the mutable model and from each other

Configuration Cache Benefits

- When nothing changed, the whole configuration phase is skipped
- Less memory pressure because the build model can be garbage collected
- Executes all tasks in parallel (incl. intra-projects)

Configuration Cache



Configuration Cache

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

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

  `org.gradle.unsafe.configuration-cache-problems=warn`

Configuration Cache Constraints

On *configuration* logic

- Environment, configuration files, external processes (like `git`)
- Build listeners registered at configuration, notified during execution

On *execution* logic

- No references to build model during task execution like `Project`, `Task` ...
- No live objects in inputs like `InputStream`, `Socket` ...

Configuration Cache Strength

Forces good practices

- Clear separation between configuration and execution
- Correct declaration of inputs
- No cross-dependencies between tasks

Configuration Cache Compatibility

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

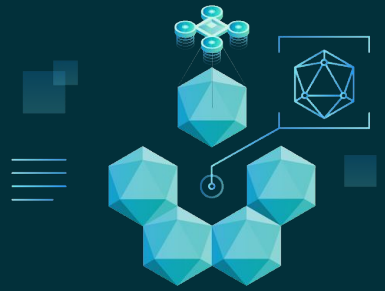
Configuration Cache Roadmap

- Stable in Gradle 8.x

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

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

  (without opt-out)



Going Forward

Resources

- Release Notes
- User Manual
- Public Roadmap
- Community Platforms

Thank you!

Twitter @alllexist

asemin@gradle.com