Kotlin, one language to build them all!

Paul Merlin

gradle.github.io/imaginate

pdf - sources

Agenda

- **Gradle & Kotlin**
  - What is this all about?
- **Imaginate**
  - An imaginary image generator
- **Gradle & Kotlin ❤**
  - Build logic

Who am I?

```
speaker {
    name = "Paul Merlin"
    company = "Gradle"
    joined = 2015
    position = "Kotlin DSL Project Lead, again \o/"
    previously = "Configuration Cache Project Lead"
    github = "eskatos"
    mastodon = "@eskatos@mastodon.social"
    successes = listOf(
        "BASIC 'Hello, World!'" in 1986,
        "C 'Hello, World!'" in 1989,
        "Java 'Hello, World!'" in 1996,
        "Kotlin 'Hello, World!'" in 2015,
        "tools", "daemons", "apps", "frameworks", "libs"
    )
    failures = generateSequence(code) { bugs }
}
```

Gradle

Since 2008, our mission is to
accelerate developer productivity.

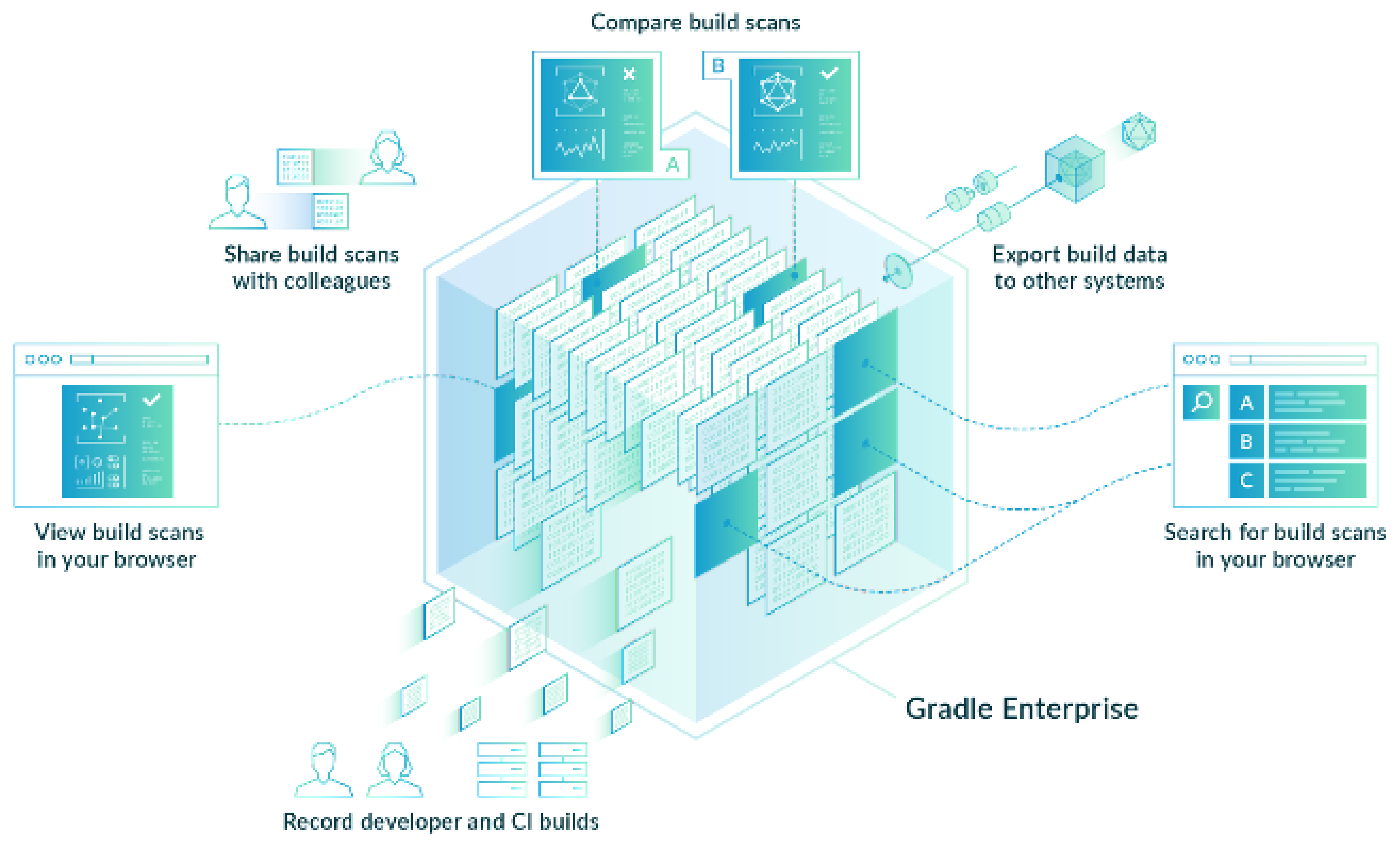
Gradle Build Tool

Apache licenced sofware build tool

With 30M+ monthly downloads, this is one of the top 20 popular open source projects according to TechCrunch.

Gradle Enterprise

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



Compare build scans
Share build scans with colleagues
Export build data to other systems
View build scans in your browser
Search for build scans in your browser
Gradle Enterprise
Record developer and CI builds

Gradle

Android

Maven

Bazel

Scala (2023)

Developer Productivity Engineering

DPE is an emerging software practice that relies on acceleration technologies and data analysis to improve developer productivity.

dpe handbook

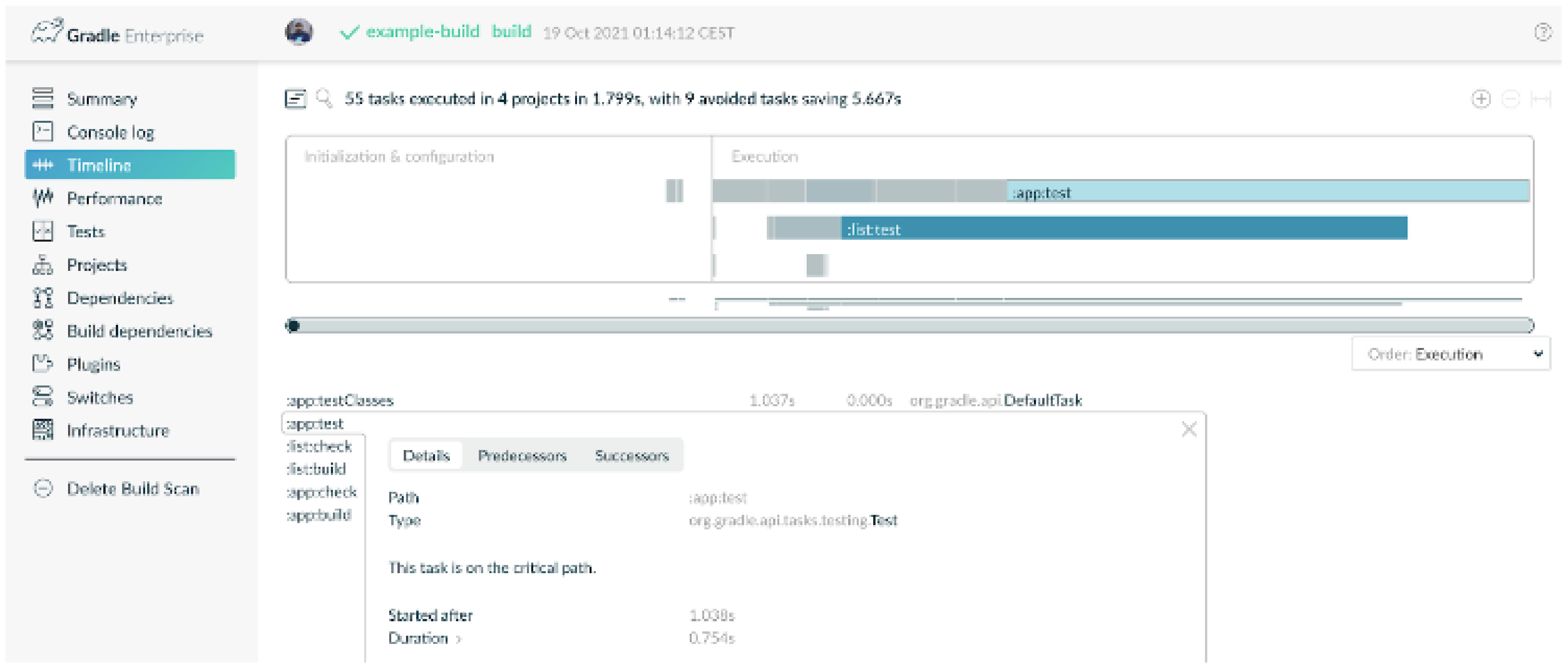
gradle.com/developer-productivity-engineering

dpesummit.com

DPE Lowdown - Youtube Playlist
DPE Showdown - Youtube Playlist

Build Scans

A permanent record
of what happens during a build.



Gradle Enterprise | example-build build 19 Oct 2021 01:14:12 CEST

- Summary
- Console log
- Timeline
- Performance
- Tests
- Projects
- Dependencies
- Build dependencies
- Plugins
- Switches
- Infrastructure
- Delete Build Scan

55 tasks executed in 4 projects in 1.799s, with 9 avoided tasks saving 5.667s

Initialization & configuration | Execution

:app:test
:list:test

Order: Execution

```
:app:testClasses      1.037s    0.000s    org.gradle.api.DefaultTask
:app:test
:list:check
:list:build
:app:check
:app:build
```

Details | Predecessors | Successors

Path	:app:test
Type	org.gradle.api.tasks.testing.Test

This task is on the critical path.

Started after	1.038s
Duration	0.754s

build scan

Gradle & Maven build speed challenge

Get some swag :)

We recruit!

If what we're going to talk about Today is of interest to you, come work with us!

gradle.com/careers

Gradle & Kotlin

What is this all about?

Gradle & Kotlin

- Gradle Build Tool
- Kotlin Programming Language

Gradle Build Tool

Gradle Build Tool is the fast and dependable open source build system that automates building software of any type, size or complexity.

The unique advantage of Gradle Build Tool is its elegant and extensible declarative build language.

Various ecosystems

Core

Java Groovy Scala checkstyle JACOCO Java Code Coverage

and more …

Community

android Kotlin nokee spring MICRONAUT node JS

Asciidoctor docker PostgreSQL sonarqube Flyway SpotBugs

and more …

Logos from Wikimedia or from product official™ pages.

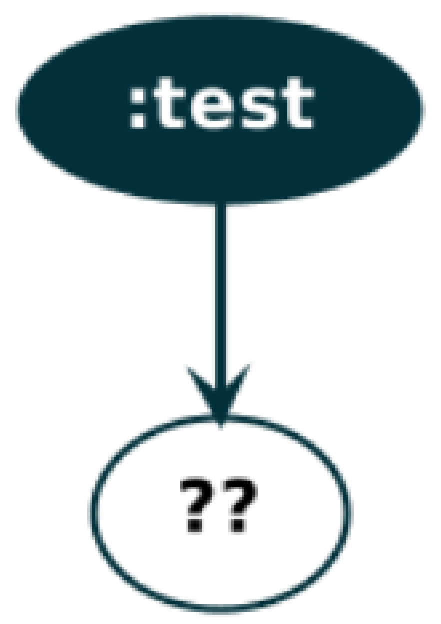
Gradle is…

A tool to automate building software

- A dependency resolution engine
- A task execution engine
- An extensible configuration model and DSL
- Plugins!

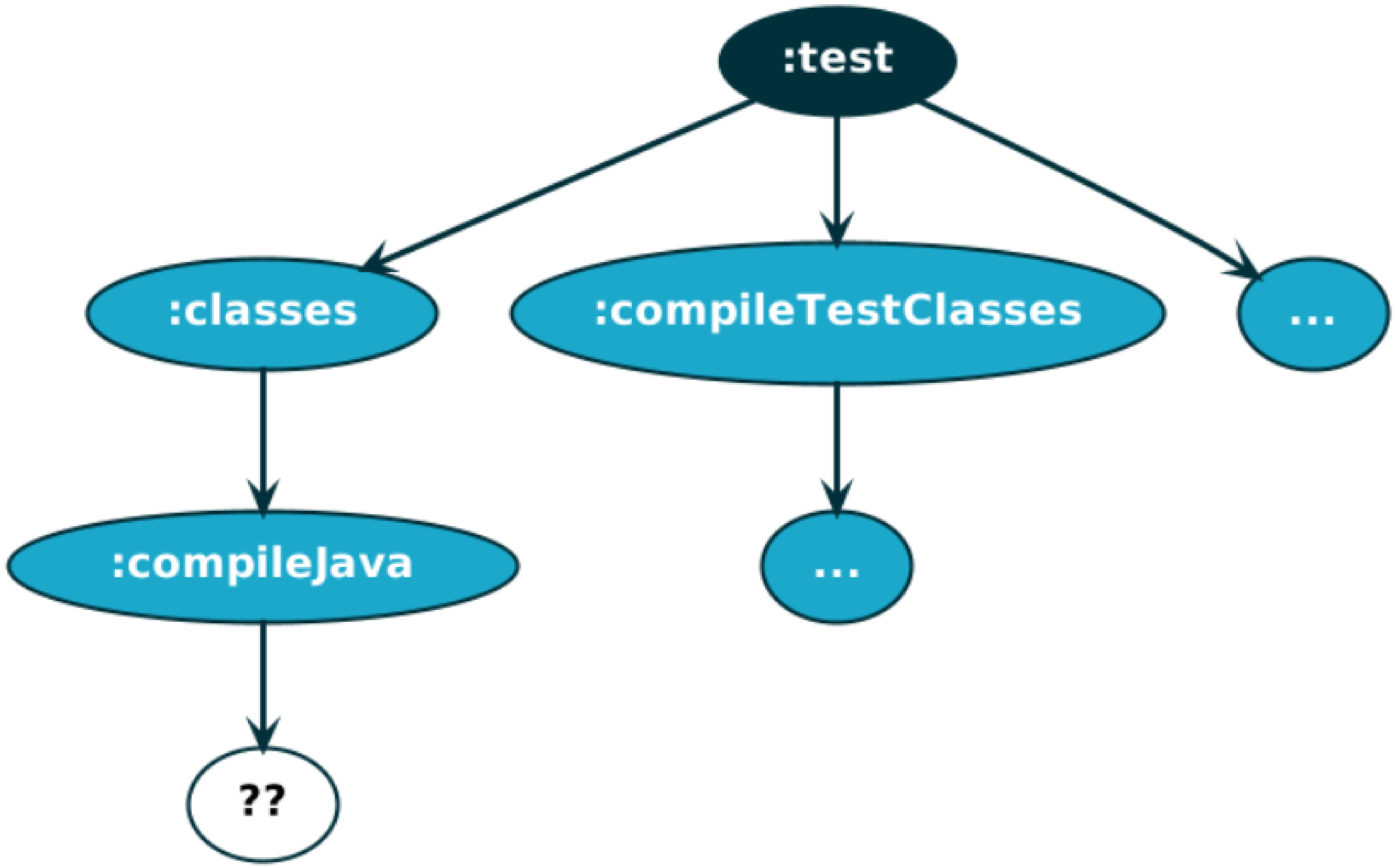
Task dependency resolution

Starts with the invoked task



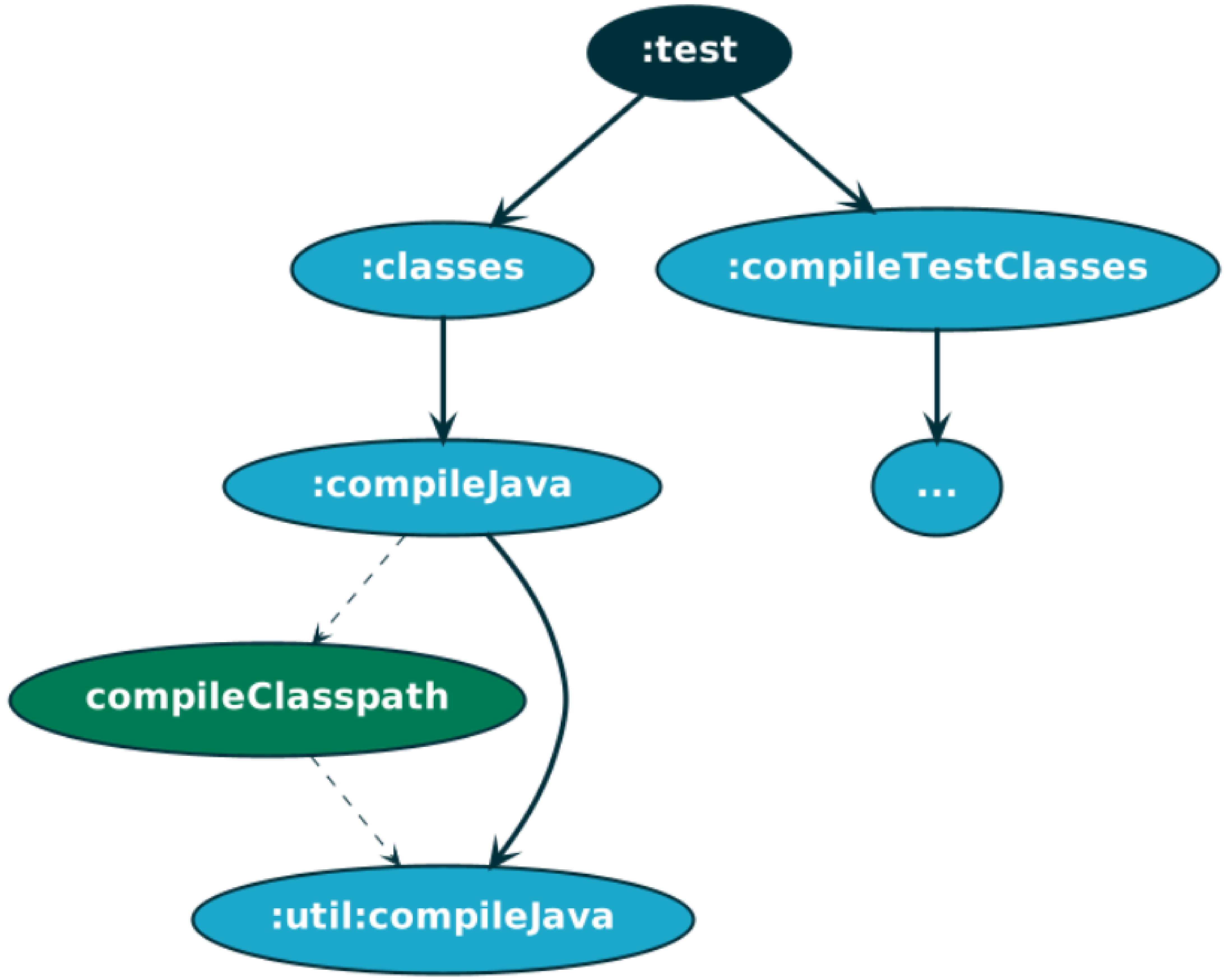
Task dependency resolution

Then connects that task with its direct task dependencies



Task dependency resolution

Includes indirect ones, potentially from a different project



Task execution

Never do something that was done before

Avoid rework

`UP-TO-DATE`

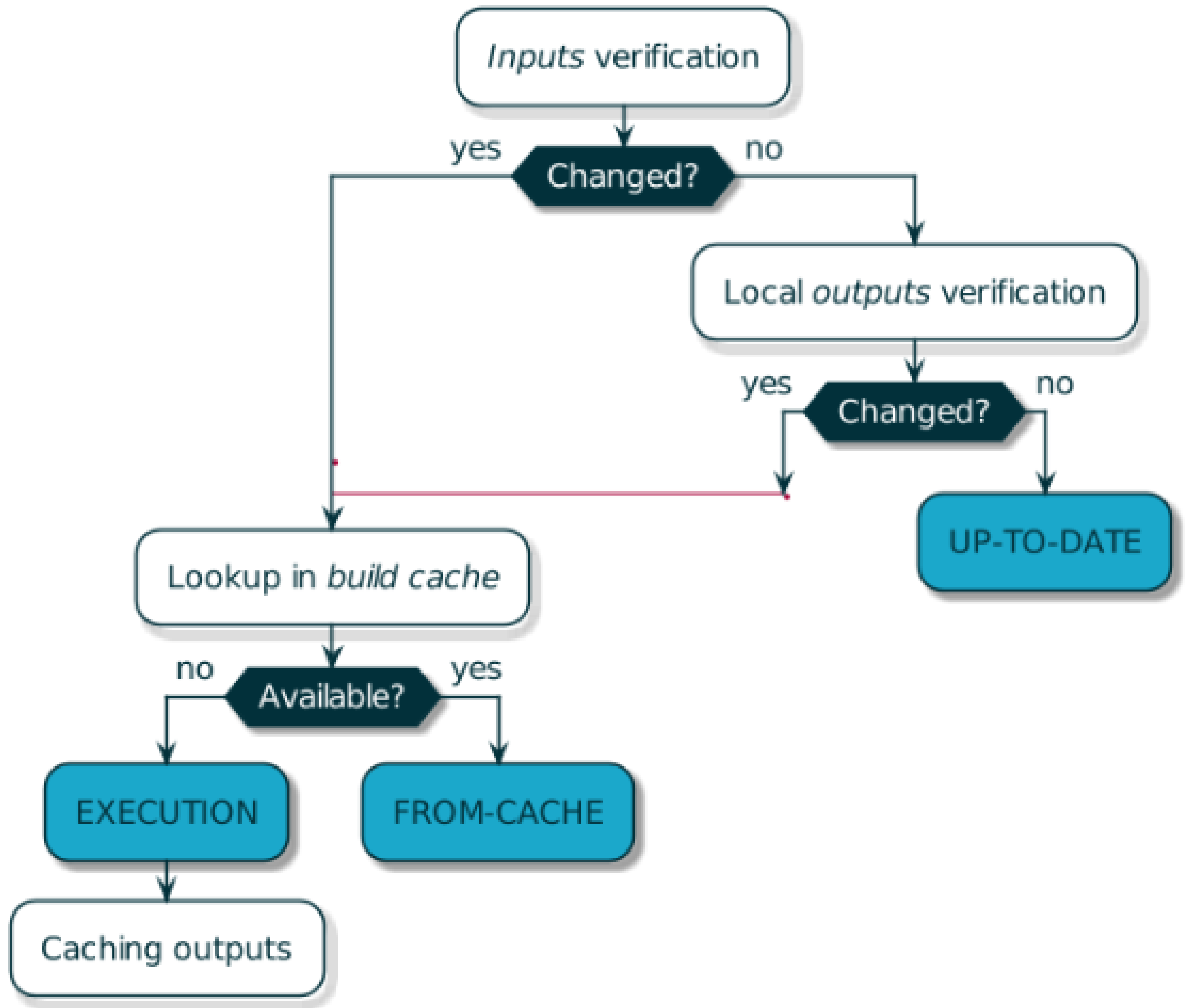
Build cache

`FROM-CACHE`

Do the work

`EXECUTION`

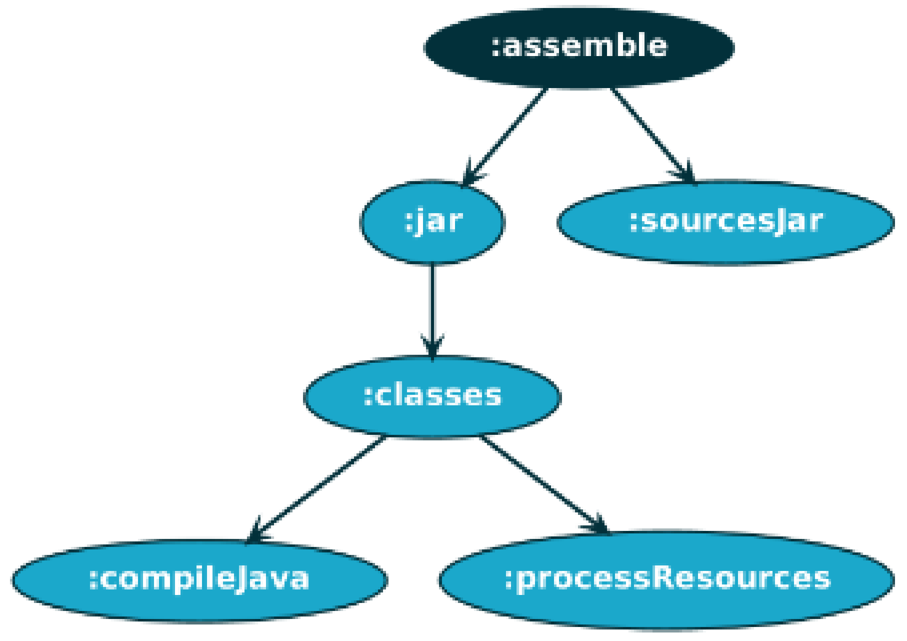
Incremental tasks



Extensible configuration

- Extensible configuration model
- Plugin system
- Dynamic DSL
- Modelling of a build vs. scripting tasks

```
plugins {
    id("java")
}
java {
    withSourcesJar()
}
```



build scan

Configuration building blocks

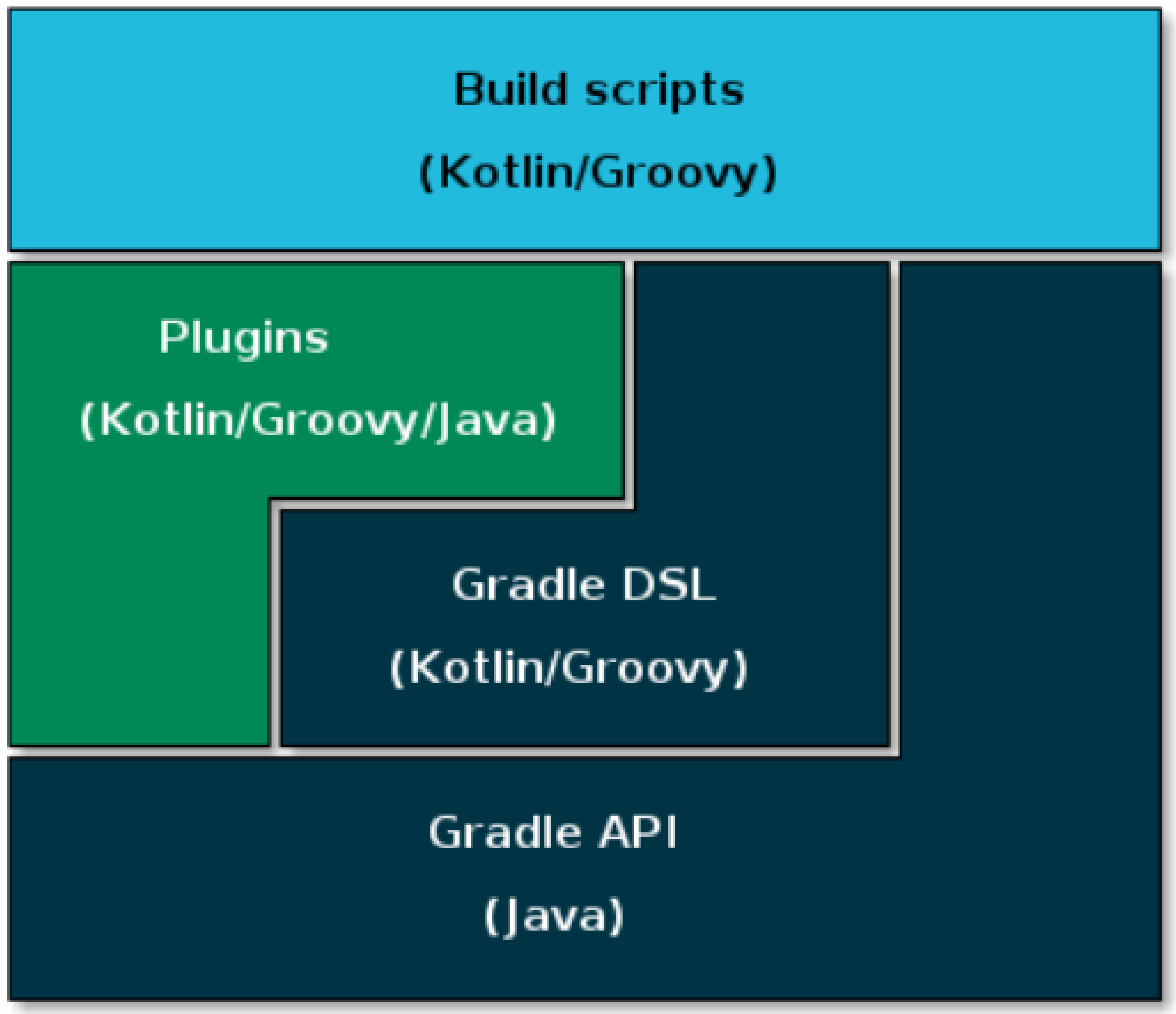
Build
Build logic ensemble

Settings
Build and project hierarchy

Projects
Individual modules

Composite builds
Compose multiple builds together

Implementation languages



Build scripts vs. Plugins

Configuration
Build scripts are declarative

Conventions
Local build plugins implement the build configuration logic

plugins.gradle.org
Build logic can leverage external plugins

Writing plugins

A Gradle plugin is built by a Gradle build

Plugin development plugins for:

- Model Gradle plugin building
- Validate plugins, DSL extensions and tasks
- Simplify integration testing
- Declare plugin metadata
- Publish plugins to plugins.gradle.org

Build script of a plugin in Java

```
plugins {
    id("java-gradle-plugin")
}
```

Build script of a plugin in Groovy

```
plugins {
    id("groovy-gradle-plugin")
}
```

Build script of a plugin in Kotlin

```
plugins {
    id("kotlin-dsl")
}
```

Programming model

Don’t call us, we’ll call you!
Sugarloaf

Abstract types, instantiated and decorated by Gradle

Injected Gradle Services

Something.groovy

```
abstract class Something implements GradleApiType {

    abstract Property<String> getSomeProperty()

    @Inject
    abstract ExecOperations getExecOps()

    @Override
    def someAction() {
        execOps.exec {
            commandLine "git", "status"
        }
        println someProperty.get()
    }
}
```

build.gradle

```
def some = objects.newInstance(Something)
```

Plugins

Plugin types

- `Project`
- `Settings`
- `Gradle`

Script equivalents

- *project script*
- *settings script*
- *init script*

MyPlugin.java

```
class MyPlugin implements Plugin<Project> {
    @Override
    public void apply(Project project) {
        /* ... Uses the Gradle API ... */
    }
}
```

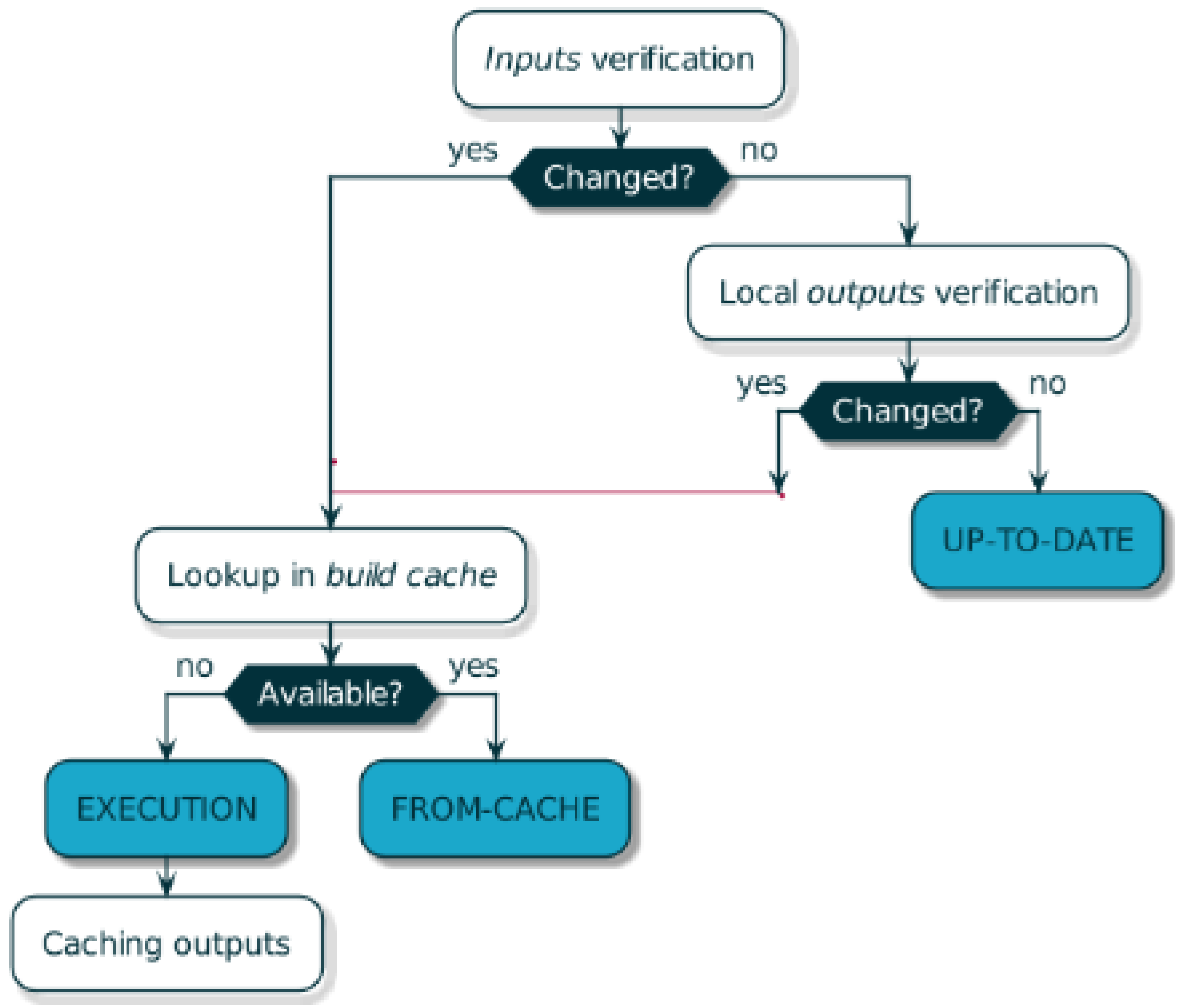
Writing tasks

Tasks are functions

What are the task *inputs*?

What are the task *outputs*?

What happens when input or output change?



Writing tasks

Executed each time

MyTask.kt

```kotlin
abstract class MyTask : DefaultTask() {

  abstract val inputs: ConfigurableFileTree

  abstract val output: DirectoryProperty

  @TaskAction
  fun action() {
    /* ... */
  }
}
```

UP-TO-DATE

MyTask.kt

```kotlin
abstract class MyTask : DefaultTask() {

  @get:InputFiles (1)
  abstract val inputs: ConfigurableFileTree

  @get:OutputDirectory (2)
  abstract val output: DirectoryProperty

  @TaskAction
  fun action() {
    /* ... */
  }
}
```

1. Declare this property as an *input*
2. Declare this property as an *output*

Cached tasks

MyTask.kt

```kotlin
@CacheableTask ❶
abstract class MyTask : DefaultTask() {

  @get:InputFiles
  @get:PathSensitive(RELATIVE) ❷
  abstract val inputs: ConfigurableFileCollection

  @get:OutputDirectory
  abstract val output: DirectoryProperty

  @TaskAction
  fun action() {
    /* ... */
  }
}
```

❶ Declare task implementation as *cacheable*

❷ Specifies *input* path sensitivity

doc/build_cache

Writing tasks

Many opt-ins to consider

Gradle helps by validating plugins and tasks

- At runtime → warnings / failures / deprecations
- While developing plugins → `:validatePlugins`

Validation evolves by checking more and more things while preserving compatibility

Gradle is …

A tool to automate building software

- A dependency resolution engine
- A task execution engine
- An extensible configuration model and DSL
- Plugins!

Kotlin Programming Language

Kotlin is a multi-platform, statically typed, null-safe, general-purpose functional and object oriented programming language with type inference.

Kotlin is…

Multi-Platform != Cross-Platform

- Targets

  JVM, Android, JS, WASM, Native (Linux, Windows, Apple via LLVM)

- Interoperable

  Java - JavaScript/TypeScript - C

- Kotlin Common

  The language, all platforms, `expect/actual`

- Standard Library

  Common, JVM, JS/WASM, Native

- Platform APIs

  JVM, JS/WASM, Posix, Windows, Apple etc…

Kotlin is…

Multi-Platform != Cross-Platform

- Ecosystem

  CLI, networking, structured concurrency, serialization, GUI etc…

- Share as little as you want

  DTOs, Networking, Storage etc…

- Share as much as you want

  View Model/Presenter/Controller, UI etc…

- Gradle Plugins!

  Dependency management, variants publication/consumption etc…

Kotlin shared UI toolkit

Compose Multiplatform

- Originate from Android

  Joint effort between JetBrains and Google

- Reactive UI toolkit

  Similar to React or SwiftUI

- Now supports all platforms

  JVM on Linux/Windows/Mac

  "Native" on Android

  Native on ios

  Canvas on WASM in the browser

Kotlin Programming Language

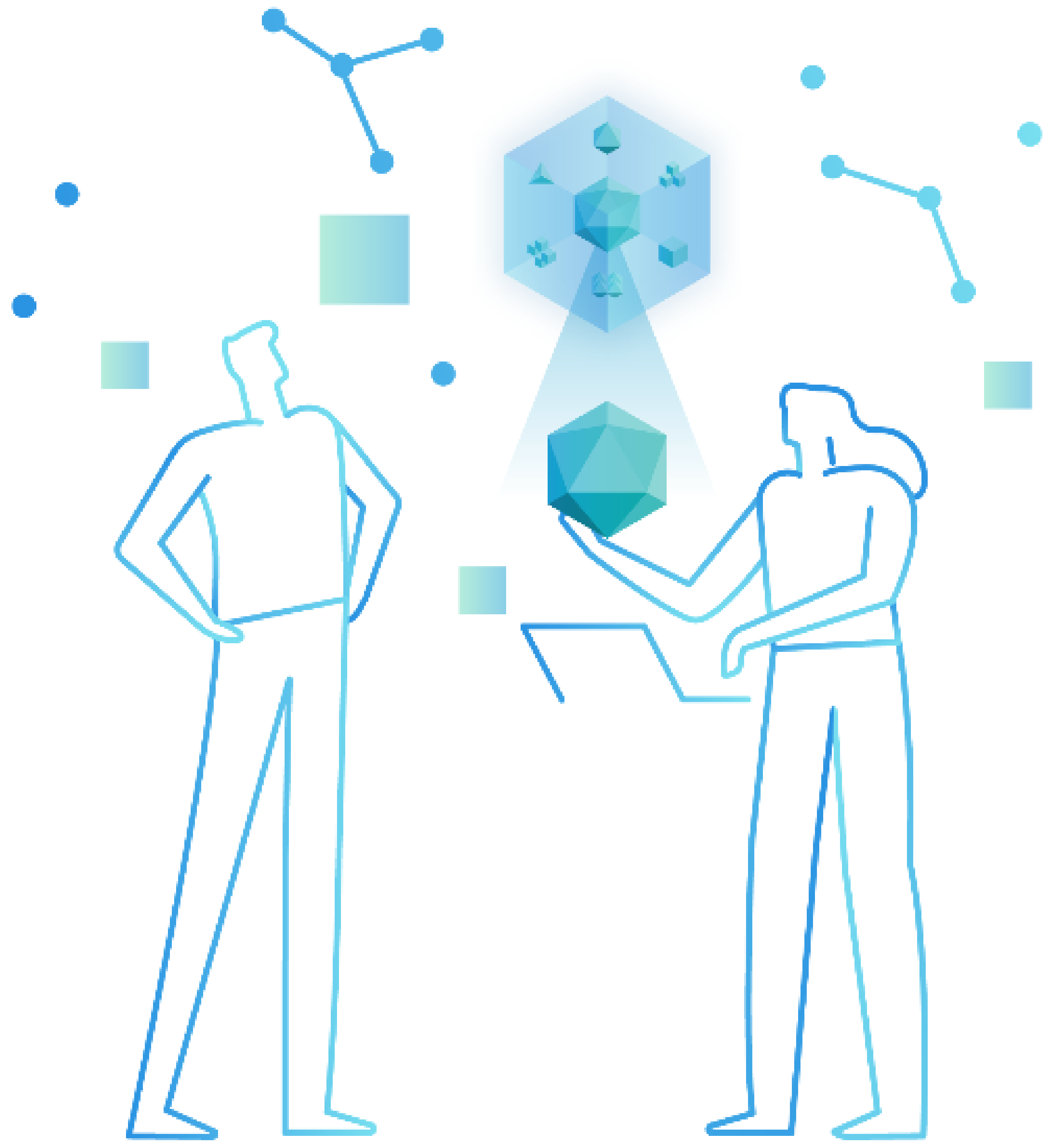
Kotlin is a multi-platform, statically typed, null-safe, general-purpose functional and object oriented programming language with type inference.

Imaginate

An imaginary image generator

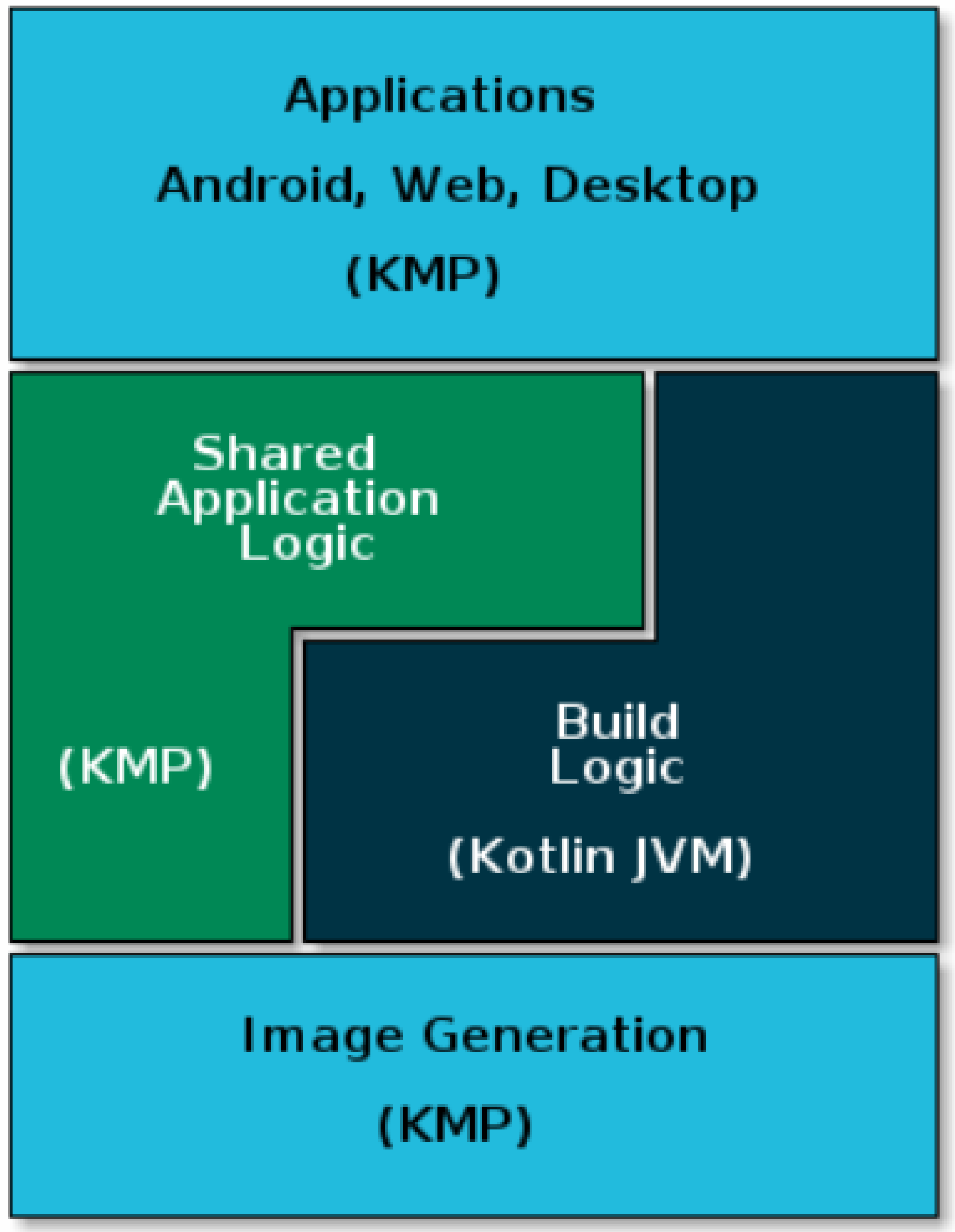
Imaginate - Demo

Imaginate - Exploration



gradle/imaginate#DesktopMain.kt

Imaginate - Overview



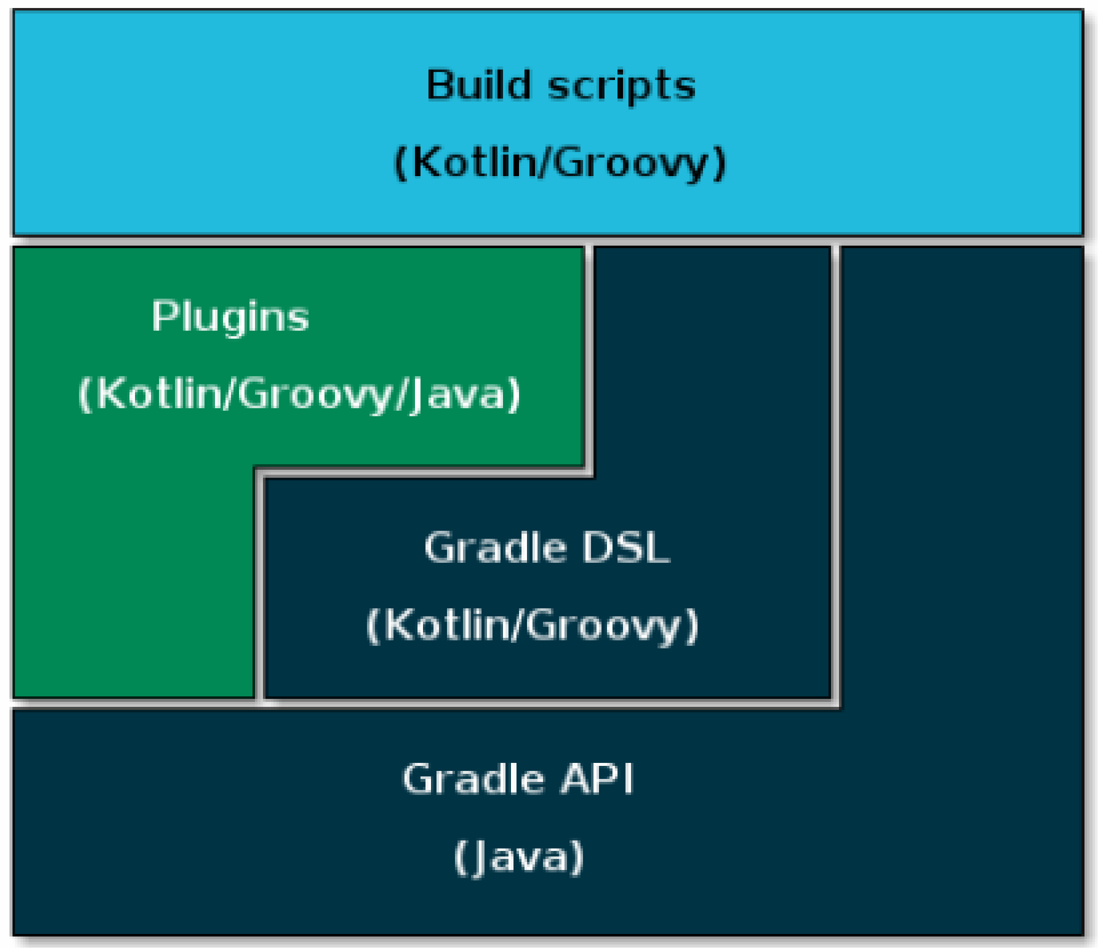
Gradle & Kotlin ❤

Build logic

Build logic - Kotlin!

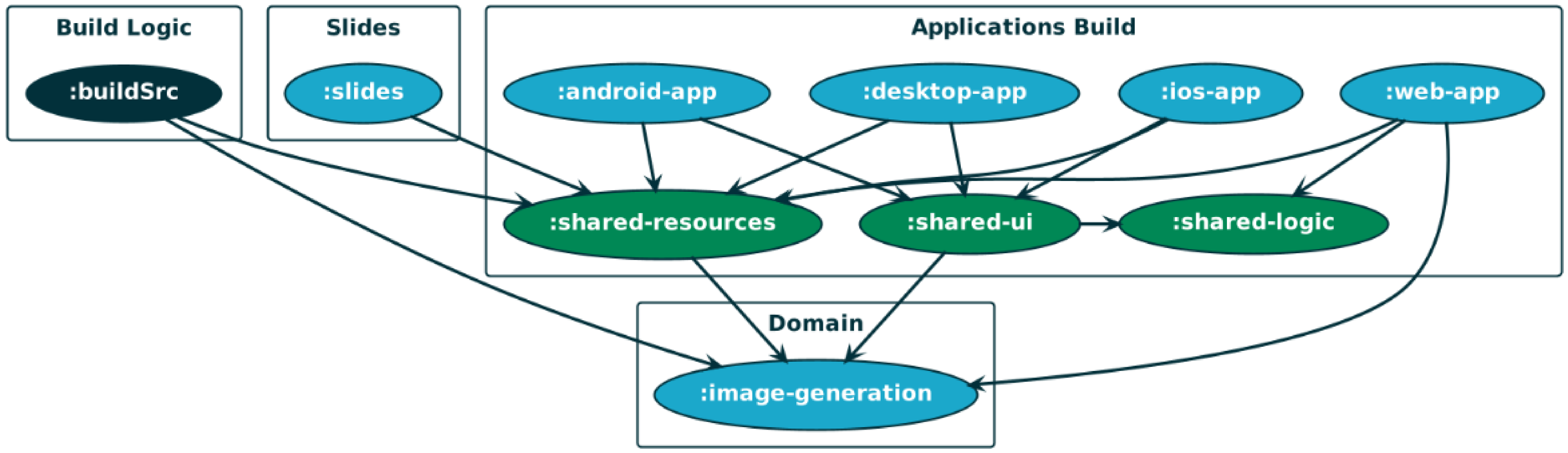
Gradle's Kotlin DSL

- Kotlin DSL on top of Gradle's Java API
- Low ceremony thanks to Kotlin Scripting
- Official Kotlin compiler plugins
- IDE assistance, code navigation etc...



Build logic - Structure

A composite build bringing 4 included builds together.



Build logic - Slides

"capital letter K. elephant. pop art."

"elephant in the Kotlin island. caravaggio."

Build logic - Overview

settings.gradle.kts

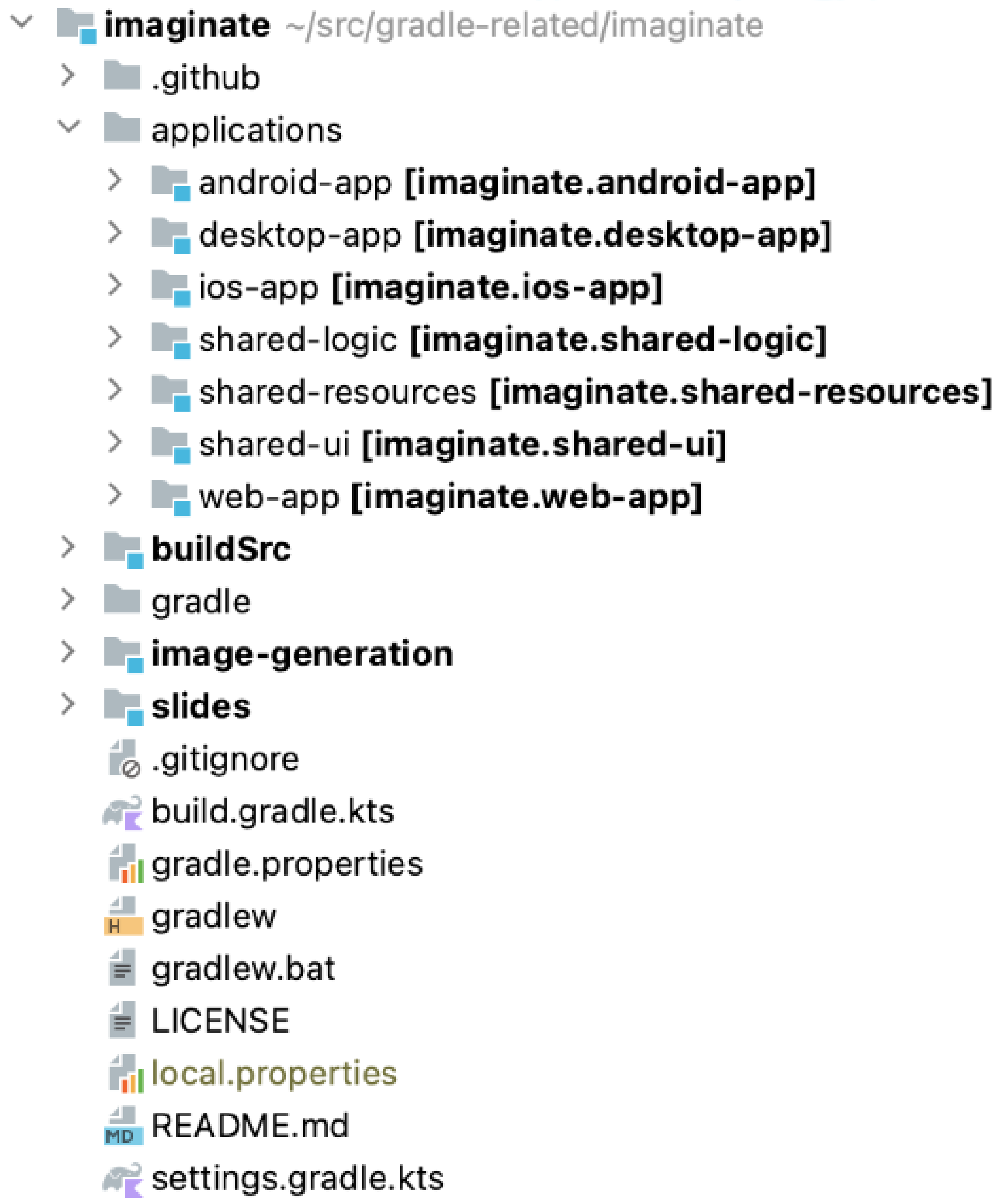
```
includeBuild("image-generation")
includeBuild("slides")

include("shared-resources")
include("shared-logic")
include("shared-ui")
include("desktop-app")
include("android-app")
include("ios-app")
include("web-app")

for (project in rootProject.children) {
    project.projectDir = file("applications/${project.name}")
}
```

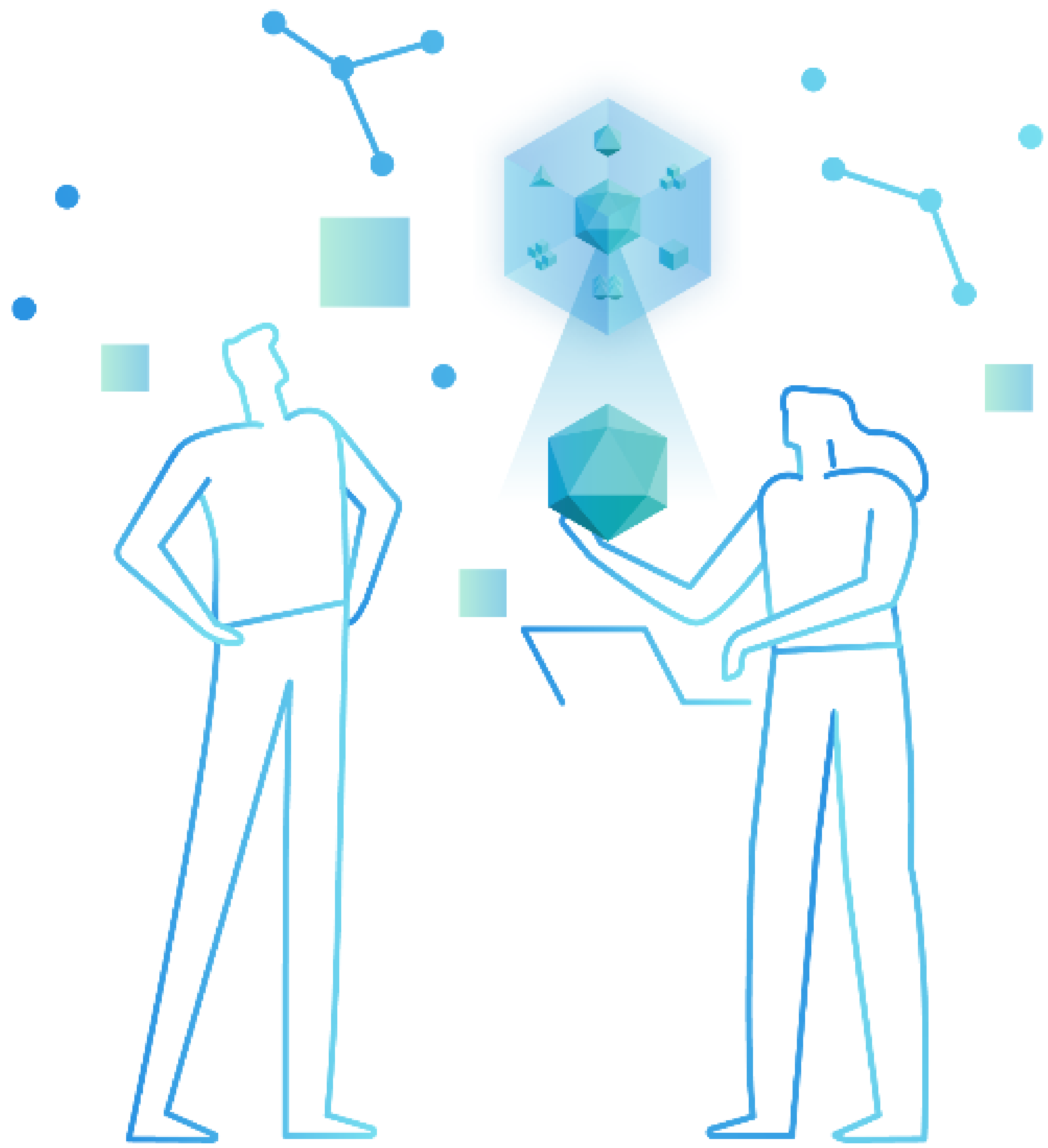
buildSrc/settings.gradle.kts

```
includeBuild("../image-generation")
```



```
imaginate ~/src/gradle-related/imaginate
  .github
  applications
    android-app [imaginate.android-app]
    desktop-app [imaginate.desktop-app]
    ios-app [imaginate.ios-app]
    shared-logic [imaginate.shared-logic]
    shared-resources [imaginate.shared-resources]
    shared-ui [imaginate.shared-ui]
    web-app [imaginate.web-app]
  buildSrc
  gradle
  image-generation
  slides
  .gitignore
  build.gradle.kts
  gradle.properties
  gradlew
  gradlew.bat
  LICENSE
  local.properties
  README.md
  settings.gradle.kts
```

Build logic - Exploration



gradle/imaginate#ImageGenerator.kt

Build logic - Kotlin DSL

Gradle plugins contribute to a dynamic model

Kotlin DSL provides static accessors

- extensions and tasks
- projects and version catalogs

IDE assistance, code navigation etc...

applications/web-app/build.gradle.kts

```
plugins {
  id("web-app")
}

dependencies {
  // Configuration defined in the
  // `shared-resources-consumer` convention plugin
  sharedBitmaps(projects.sharedResources)
}

kotlin {
  sourceSets {
    // Source set defined in `web-app` convention plugi
    jsMain {
      dependencies {
        // Static project accessor
        implementation(projects.sharedLogic)
        // Static version catalog accessors
        implementation(libs.imaginate.imageGeneration)
      }
    }
  }
}
```

Build logic - Building blocks

Convention plugins

- Put `.gradle.kts` files in `buildSrc/src/main/kotlin`
- Apply them in project build scripts

Helpers

- Put `.kt` files in `buildSrc/src/main/kotlin`
- Prefer `internal` visibility

Custom DSL

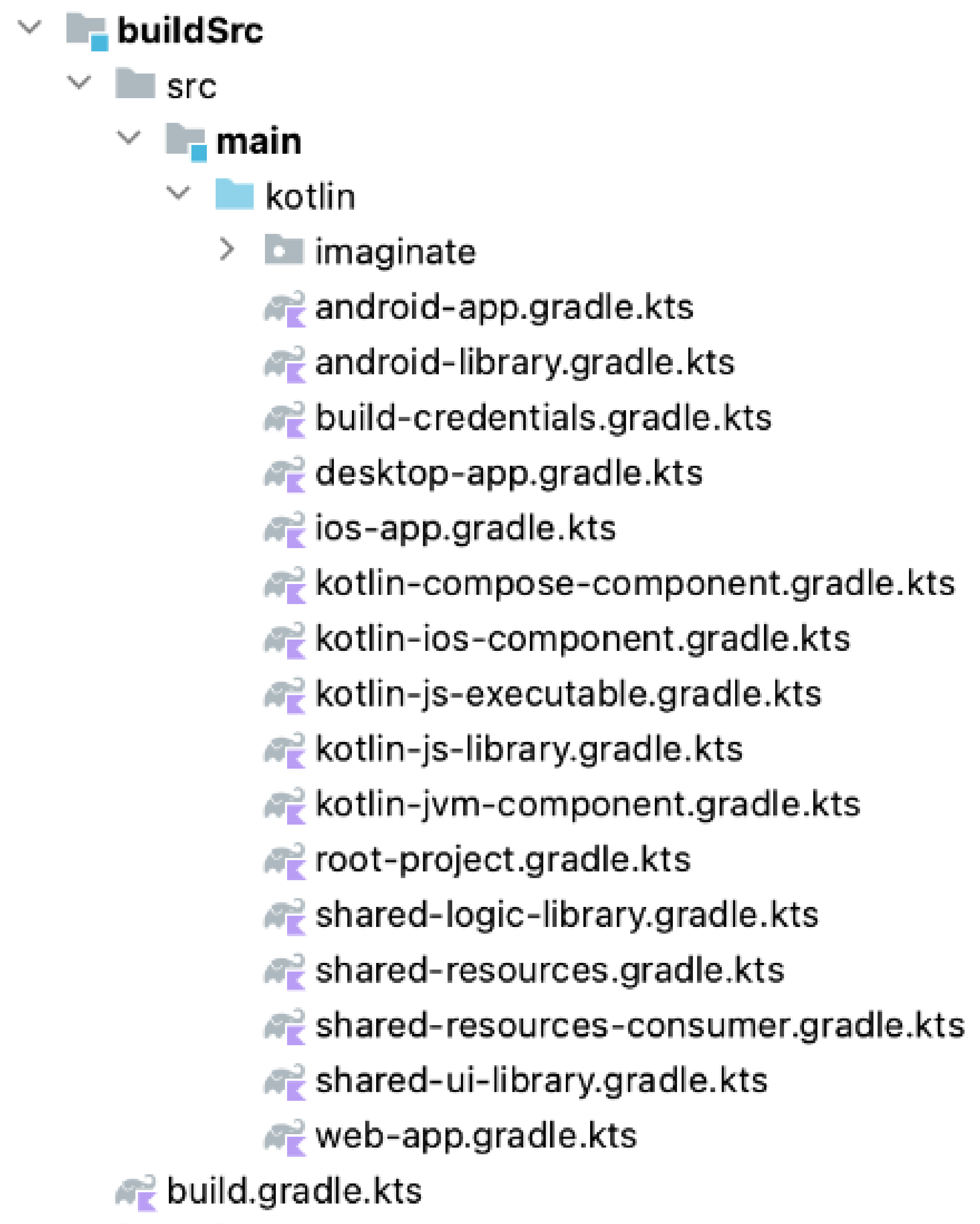
- Use Gradle extensions and containers

Custom Tasks

- Use the Worker API for isolation and parallelism

Build logic - `buildSrc`

Keep imperative logic out of project build scripts.
Expose declarative DSL from convention plugins instead.



```
buildSrc
  src
    main
      kotlin
        imaginate
        android-app.gradle.kts
        android-library.gradle.kts
        build-credentials.gradle.kts
        desktop-app.gradle.kts
        ios-app.gradle.kts
        kotlin-compose-component.gradle.kts
        kotlin-ios-component.gradle.kts
        kotlin-js-executable.gradle.kts
        kotlin-js-library.gradle.kts
        kotlin-jvm-component.gradle.kts
        root-project.gradle.kts
        shared-logic-library.gradle.kts
        shared-resources.gradle.kts
        shared-resources-consumer.gradle.kts
        shared-ui-library.gradle.kts
        web-app.gradle.kts
  build.gradle.kts
```

`applications/android-app/build.gradle.kts`

```
plugins {
    id("android-app")
}

android {
    namespace = "imaginate.android"
}

dependencies {
    implementation(projects.sharedLogic)
}
```

Build logic - buildSrc?

buildSrc vs includeBuild for build logic

buildSrc

Everything is made available in the owner build's project scripts.

You can directly use anything.

includeBuild

Nothing is made available by default.

You have to declare dependencies.

Convention plugins

```
plugins {
    id("my-convention-plugin")
}
```

From buildSrc

```
dependencies {
    implementation("my:included-build:latest")
}
```

Since Gradle 8.0, buildSrc is much more like included builds
docs.gradle.org/8.0/release-notes.html#improvements-for-buildsrc-builds

Build logic - `buildSrc` first

Start with `buildSrc`,
move to included builds when you need to

- share logic between builds,
- isolate a portion of build logic.

Transition is simple
if you only use convention plugins

Publish your convention plugins for sharing them across repositories

Build logic - Learn more

- Gradle’s Kotlin DSL
  - Documentation
  - DSL Reference
- Build organization
  - Structuring individual builds
  - Structuring software products
  - Jendrik’s Understanding Gradle videos
  - Tony's and Cédric's blog posts

What’s next?

Usability

- Simpler container DSL
- Kotlin and Kotlin DSL version management
- And more…

Performance

- Script compilation avoidance
- `plugins {}` block interpreter
- Faster precompiled script plugins build
- K2 - Kotlin Compiler 2.0
- Quicker first-use
- Configuration cache by default for Android
- Project Isolation

We recruit!

If what we talked about Today is of interest to you, come work with us!

gradle.com/careers

Thank you!