Type safe build logic with the Gradle Kotlin DSL

Hans Dockter & Paul Merlin, Gradle

Who am I

```
speaker {
    name = "Paul Merlin"
    company = "Gradle Inc"
    oss  = "Apache Polygene PMC, former chair",
    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 },
    twitter = "@eskat0s",
    github = "eskatos"
}
```

What is Gradle?

Gradle’s purpose

Gradle Build Tool is a build and automation tool

Gradle's purpose

Gradle Build Tool is a build and automation tool

- JVM based

Gradle’s purpose

Gradle Build Tool is a build and automation tool

- JVM based
- Implemented in Java

Gradle’s purpose

Gradle Build Tool is a build and automation tool

- JVM based
- Implemented in Java
- 100% Free Open Source - Apache Standard License 2.0

Agnostic Build System

Agnostic Build System

- JVM ecosystem
  - Java, Kotlin, Groovy, Scala, …

Agnostic Build System

- JVM ecosystem
  - Java, Kotlin, Groovy, Scala, …
- Native ecosystem
  - C, C++, Swift, …

Agnostic Build System

- JVM ecosystem
  - Java, Kotlin, Groovy, Scala, …
- Native ecosystem
  - C, C++, Swift, …
- Android

Agnostic Build System

- JVM ecosystem
  - Java, Kotlin, Groovy, Scala, …
- Native ecosystem
  - C, C++, Swift, …
- Android
- Misc
  - Go, Python, JavaScript, Asciidoctor, …

Gradle in figures

Gradle in figures

- > 6M downloads / month

Gradle in figures

- > 6M downloads / month
- #17 OSS projects worldwide

Gradle in figures

- > 6M downloads / month
- #17 OSS projects worldwide
- 35+ Gradle Engineers

Gradle in figures

- > 6M downloads / month
- #17 OSS projects worldwide
- 35+ Gradle Engineers
- 300K builds/week @ LinkedIn

Gradle Inc.

The company behind Gradle

Gradle Inc.

The company behind Gradle

- Build Happiness

Gradle Inc.

The company behind Gradle

- Build Happiness
- Employs full time engineers

Gradle Inc.

The company behind Gradle

- Build Happiness
- Employs full time engineers
- Providing Gradle Build Scans and Gradle Enterprise

Gradle Inc.

The company behind Gradle

- Build Happiness
- Employs full time engineers
- Providing Gradle Build Scans and Gradle Enterprise
- (Gradle consulting, support, development services etc.)

Gradle Inc.

The company behind Gradle

- Build Happiness
- Employs full time engineers
- Providing Gradle Build Scans and Gradle Enterprise
- (Gradle consulting, support, development services etc.)
- (Training: online, public and in-house)

Gradle is hiring!

- Fully distributed development team
- Exciting project used by millions
- Build Tool team and Gradle Enterprise positions

If anything you hear from now on sounds like a great problem to solve,

Talk to us!

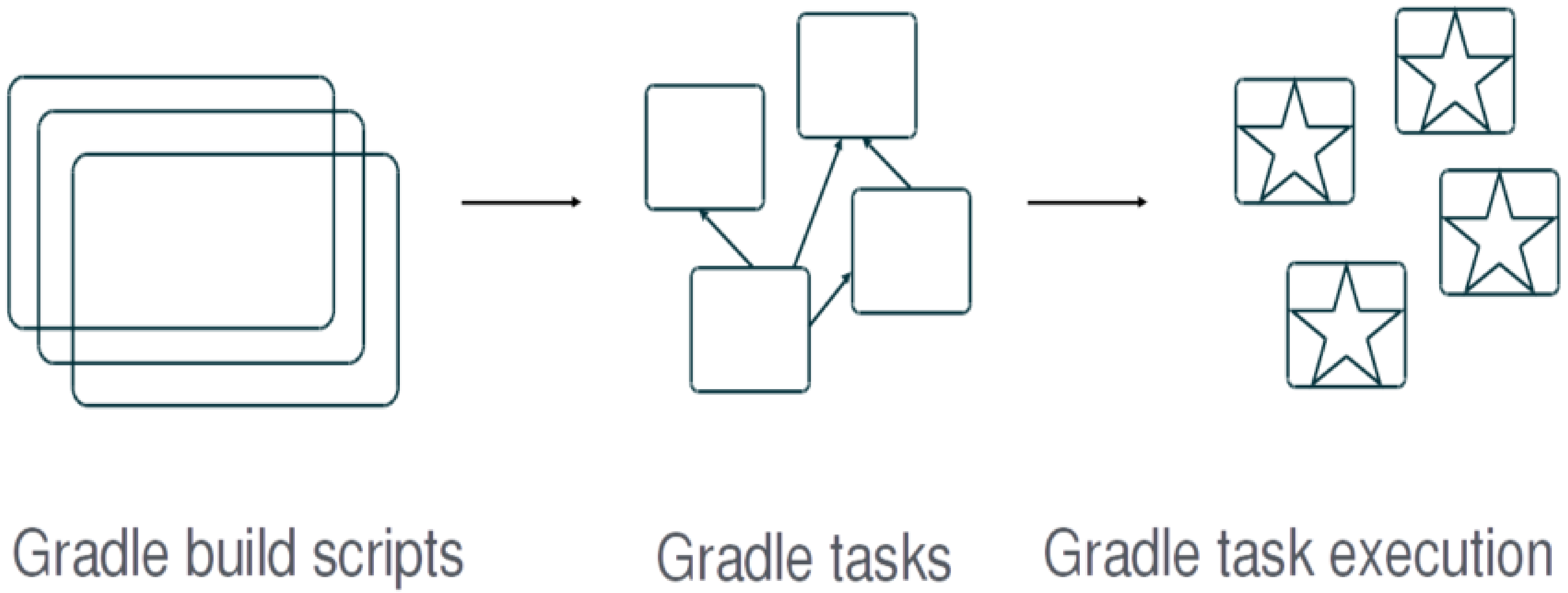
gradle.com/careers

Agenda

- Gradle Build Tool in a nutshell
- Type-safe build logic
- What makes this possible?
- Migrating from Groovy scripts
- Taking a step back
- Wrapping up

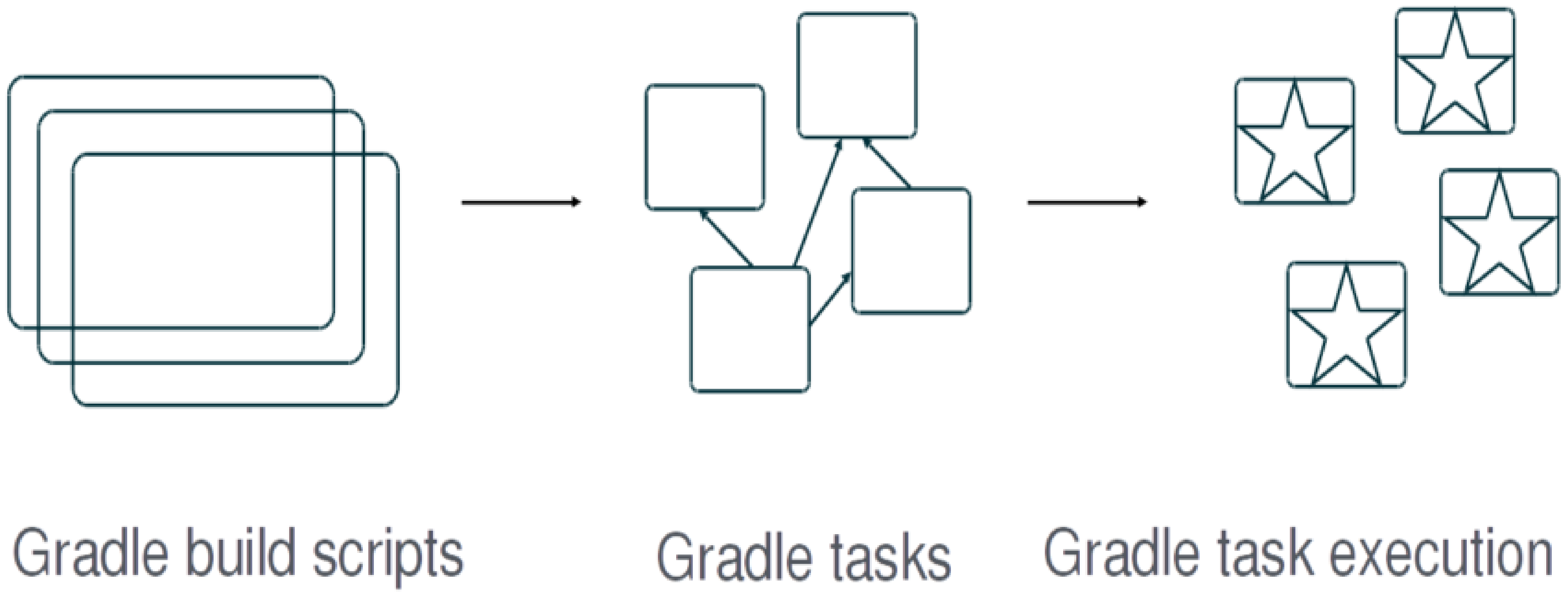
Gradle in a nutshell

Gradle in a nutshell



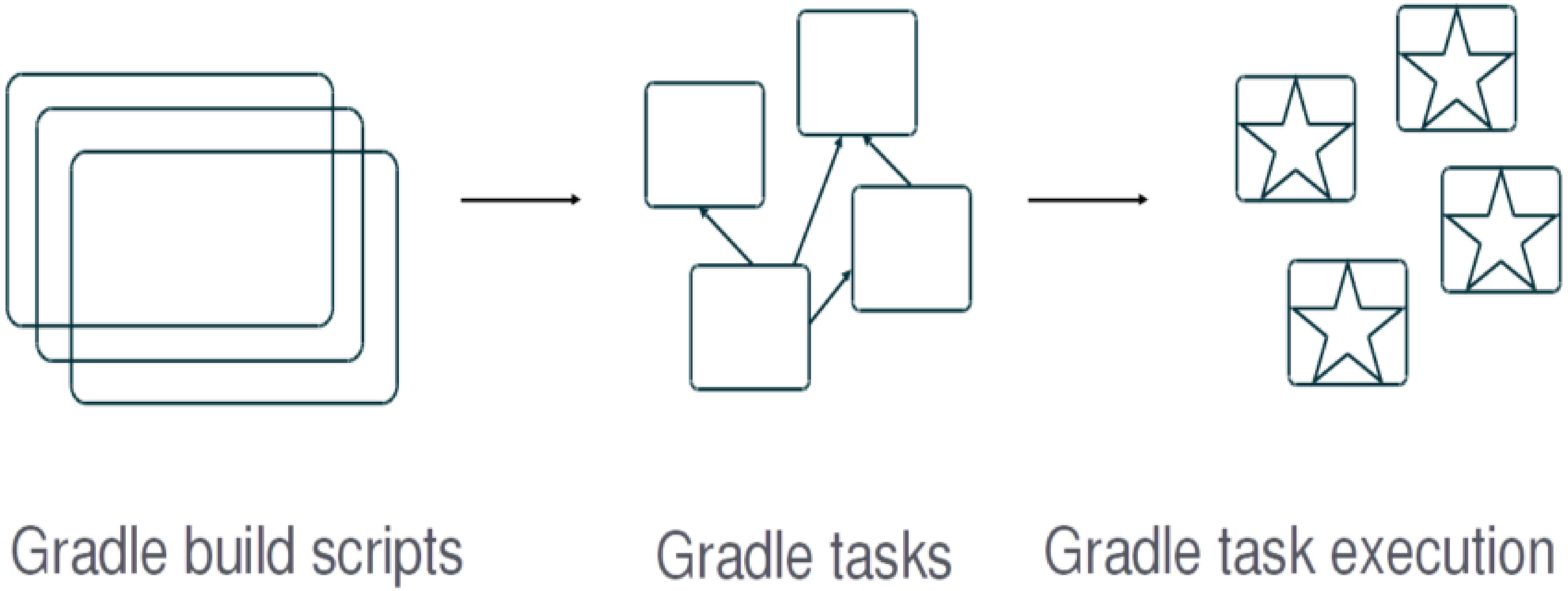
Gradle in a nutshell

- Task configuration and execution



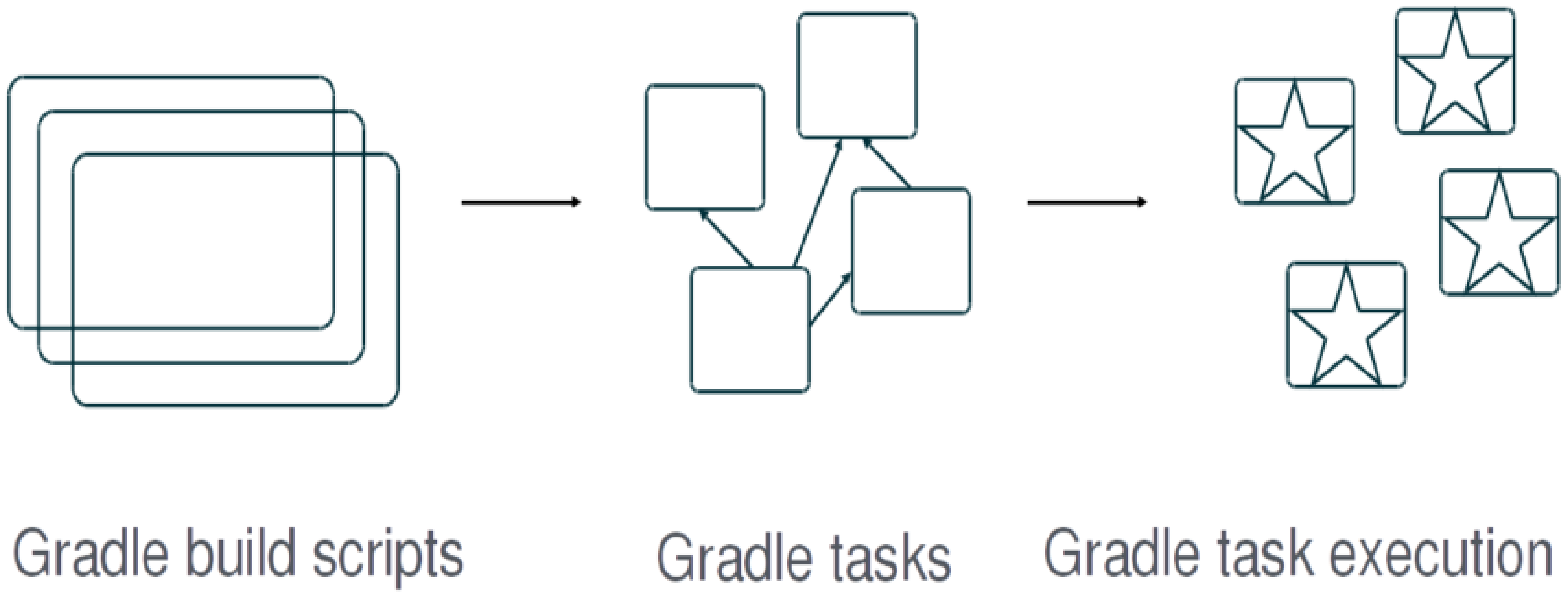
Gradle in a nutshell

- Task configuration and execution
- Dependency resolution

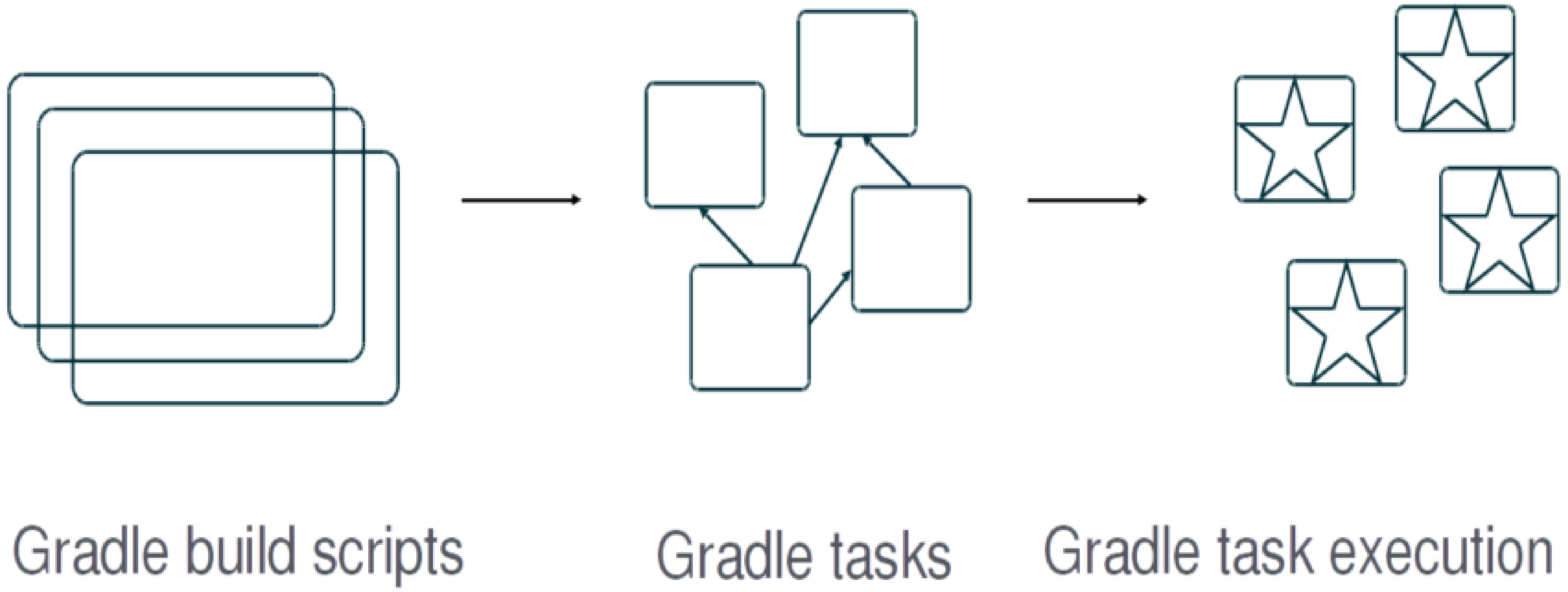


Gradle in a nutshell

- Task configuration and execution
- Dependency resolution
- Work avoidance

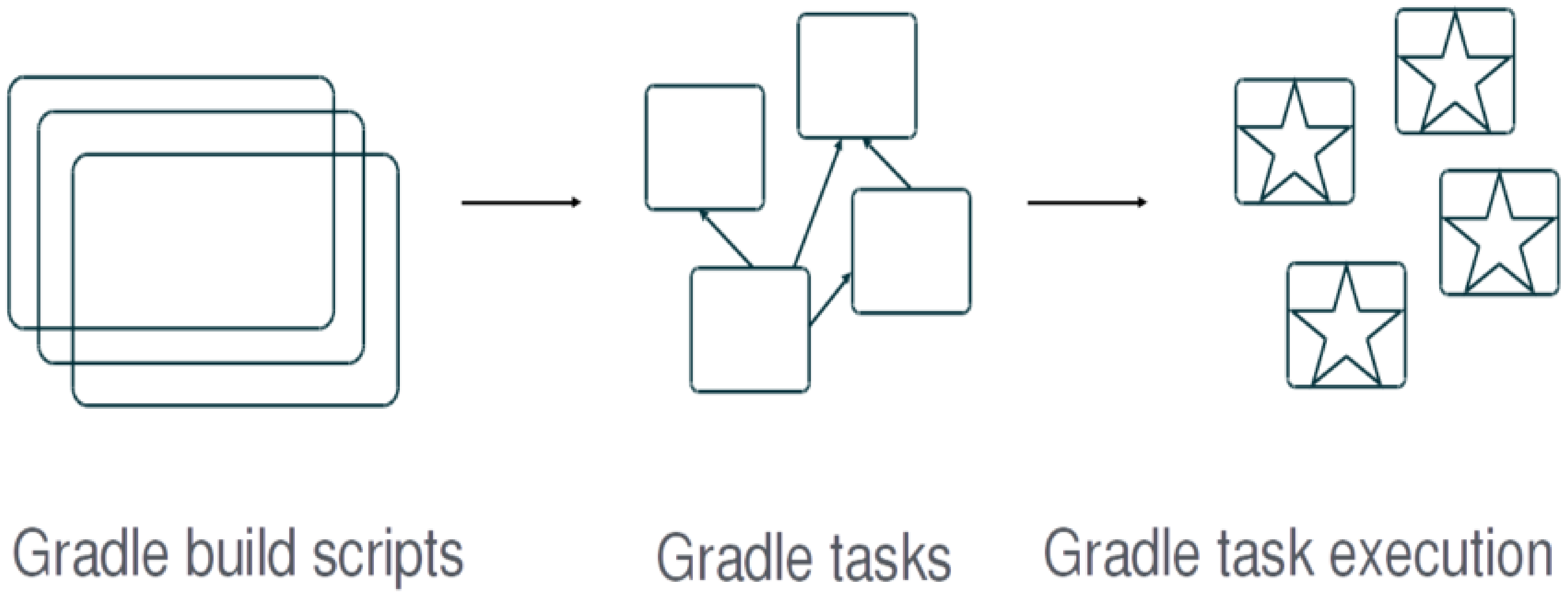


Gradle Plugins



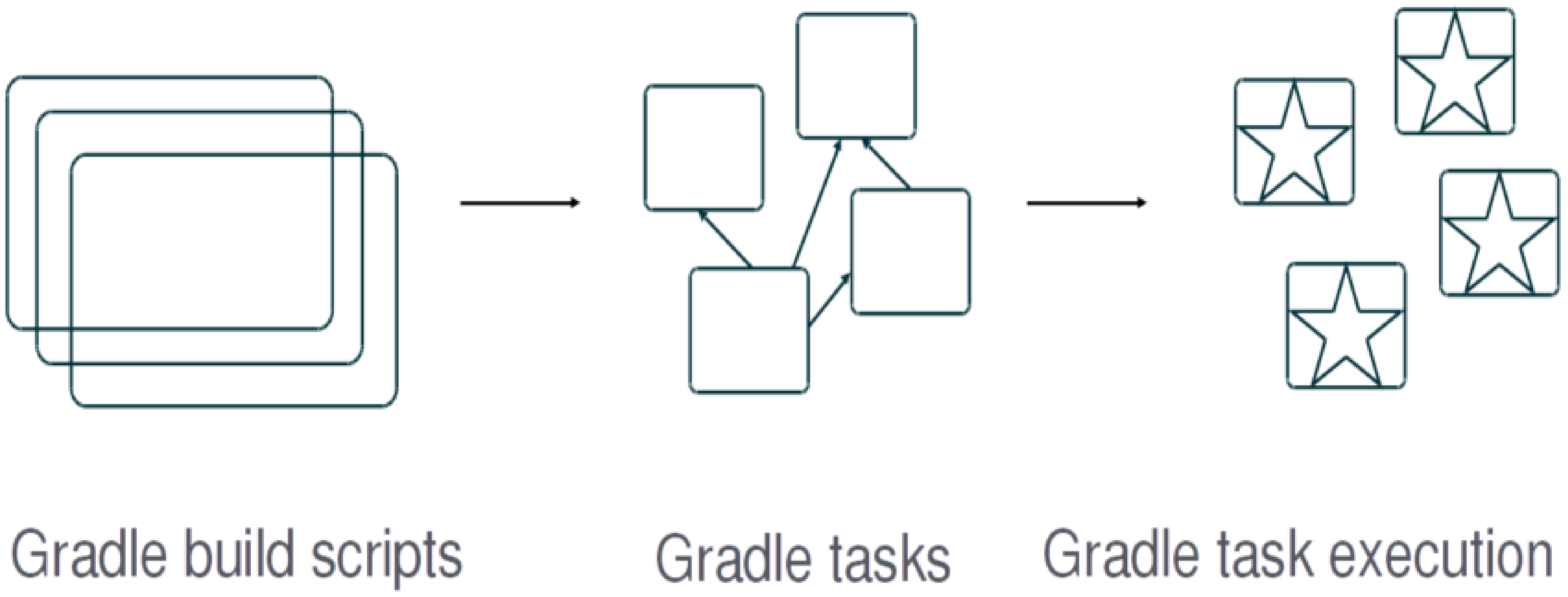
Gradle Plugins

- Core Plugins (`java`, `jacoco`, `maven-publish` ...)



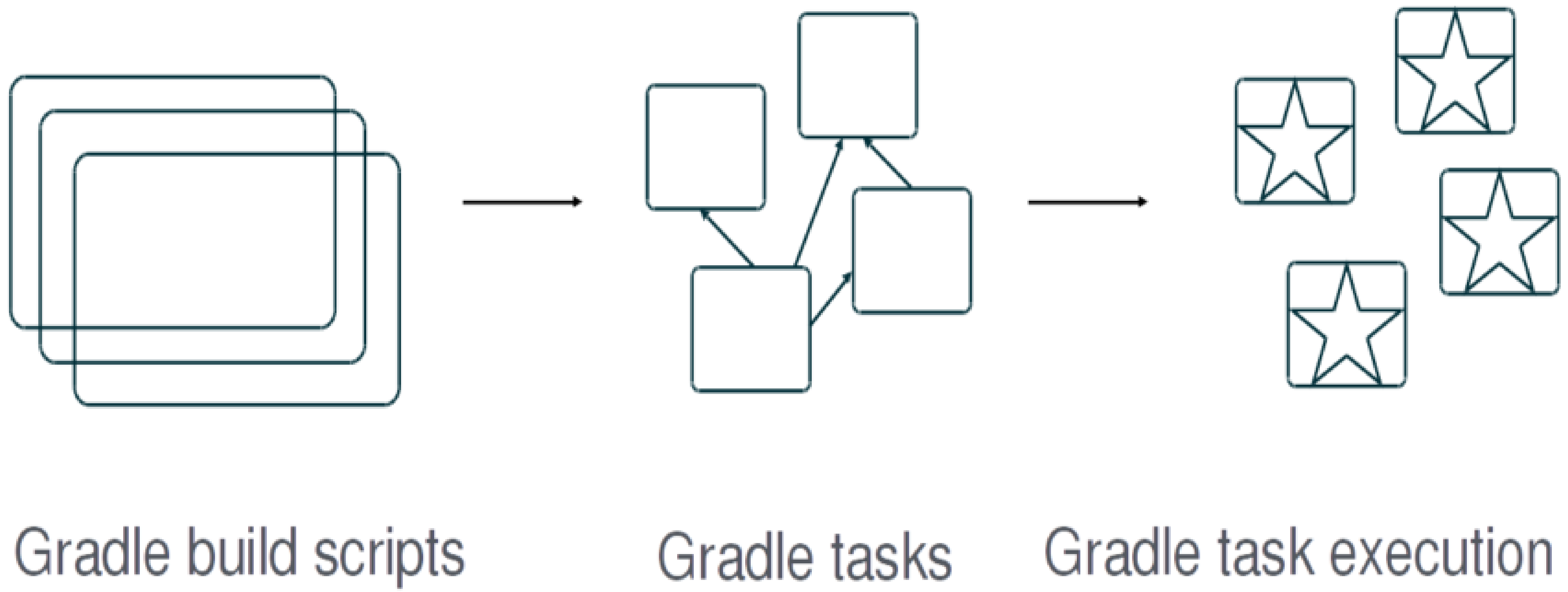
Gradle Plugins

- Core Plugins (`java`, `jacoco`, `maven-publish` ...)
- Community Plugins (`kotlin`, `android`, `golang`, `pygradle`, `asciidoctor` ...)



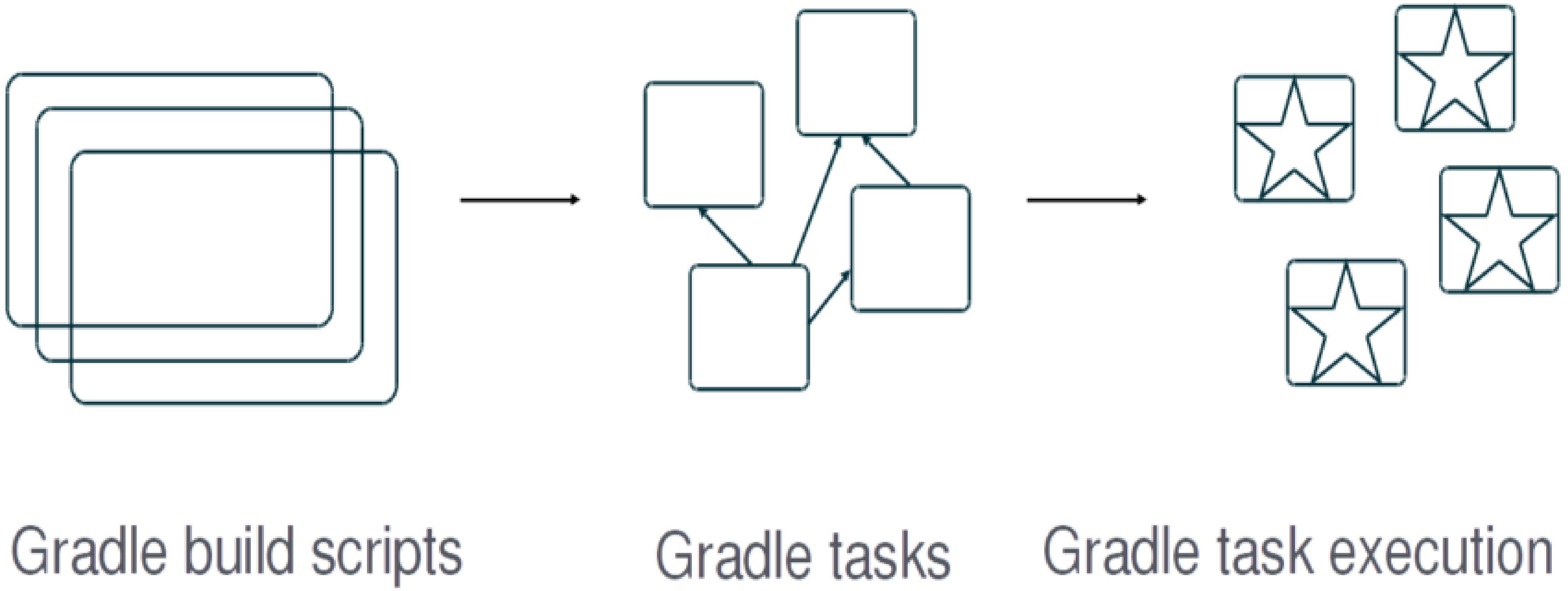
Gradle Plugins

- Gradle Plugins contribute



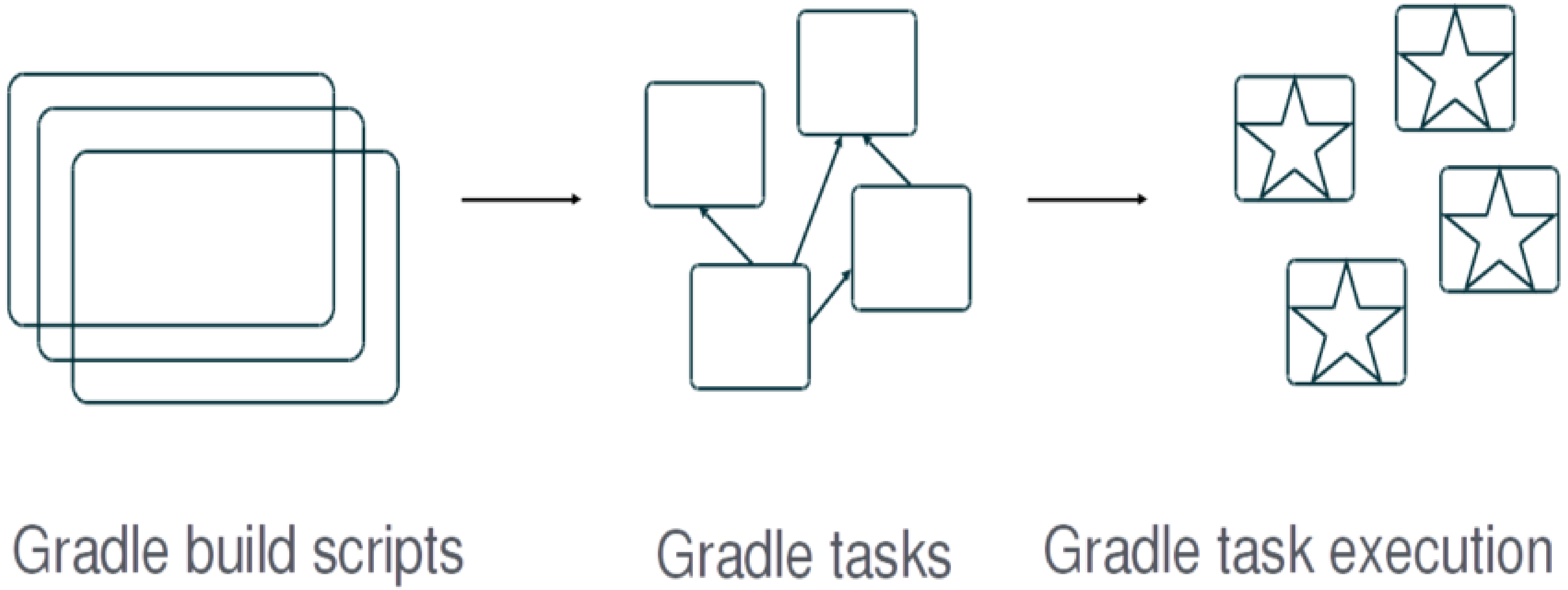
Gradle Plugins

- Gradle Plugins contribute
  - reusable and configurable Gradle Tasks



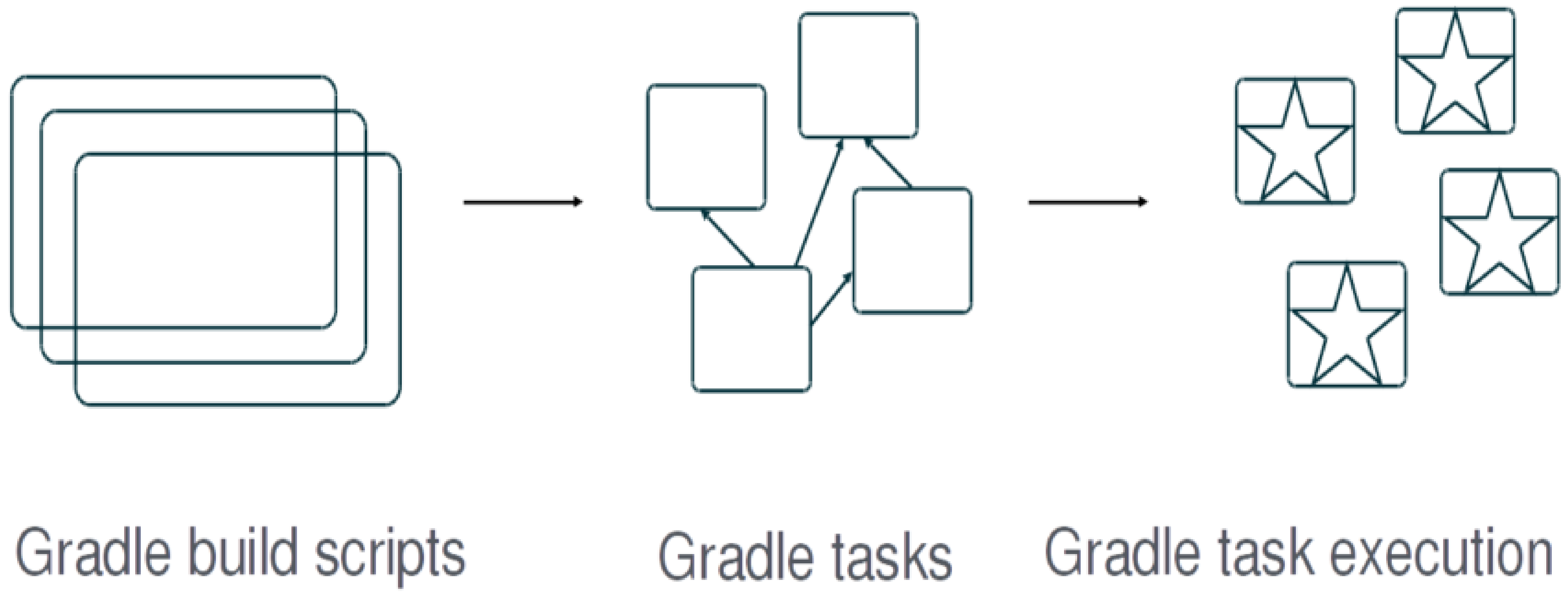
Gradle Plugins

- Gradle Plugins contribute
  - reusable and configurable Gradle Tasks
  - configurable Gradle Extensions



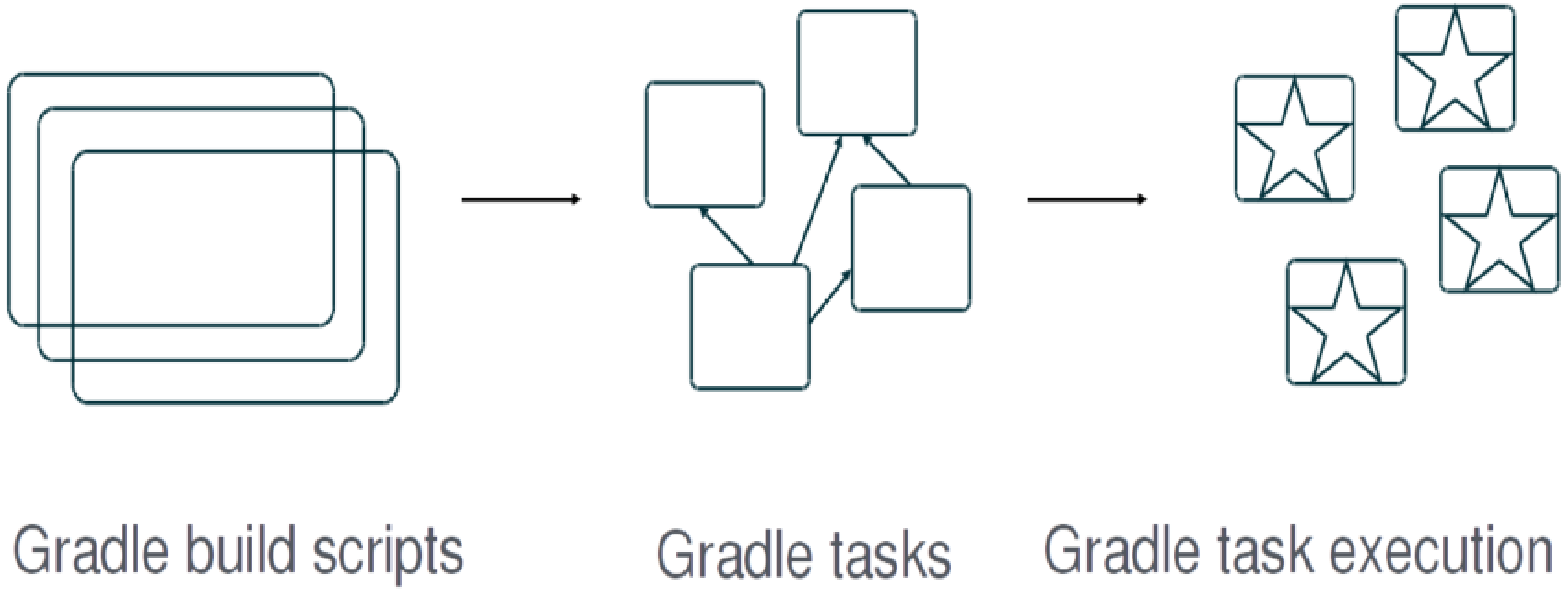
Gradle Plugins

- Gradle Plugins contribute a model to configure



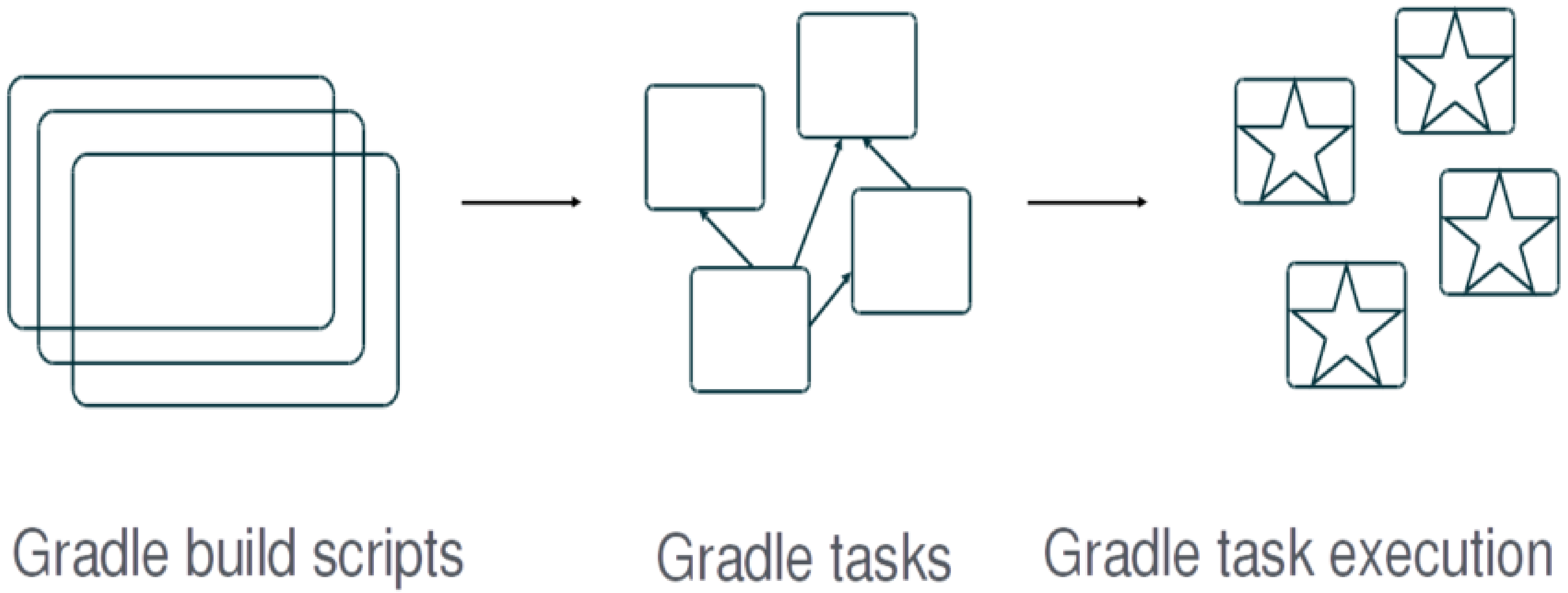
Gradle Plugins

- Gradle Plugins contribute a model to configure
    - in build scripts



Gradle Plugins

- Gradle Plugins contribute a model to configure
  - in build scripts
  - using a DSL



A Java library

```
plugins {
    `java-library`
}

dependencies {
   api("com.acme:foo:1.0")
   implementation("com.zoo:monkey:1.1")
}

tasks.withType<JavaCompile> {
    // ...
}
```

A native app

```
plugins {
    `cpp-application`
}

application {
    baseName = "my-app"
}

toolChains {
    // ...
}
```

Type-safe build logic

Type-safe build logic

DRAFT

Type-safe build logic

DRAFT

- Kotlin language

Type-safe build logic

DRAFT

- Kotlin language
- A type safety galore

Type-safe build logic

DRAFT

- Kotlin language
- A type safety galore
- Great IDE support

The Gradle Kotlin DSL

Uniting a dynamic configuration model and a statically typed language

Type-safe build logic

Demonstration

What did we just see?

What did we just see?

- A build whose logic is entirely written in Kotlin

What did we just see?

- A build whose logic is entirely written in Kotlin
- Type safety, null safety

What did we just see?

- A build whose logic is entirely written in Kotlin
- Type safety, null safety
- API and model discoverability

What did we just see?

- A build whose logic is entirely written in Kotlin
- Type safety, null safety
- API and model discoverability
- Documentation and navigation to sources

What did we just see?

- A build whose logic is entirely written in Kotlin
- Type safety, null safety
- API and model discoverability
- Documentation and navigation to sources
- Refactorings

What makes all this possible?

The Gradle build model

DRAFT

- The build model
  - Static API
  - Static Kotlin view over dynamic model contributed by plugins

`.kt` VS `.kts` VS `.gradle.kts`

DRAFT

- kt plain Kotlin code
- kts Kotlin code assumed to be executed - kotlin scripting support
- .gradle.kts is Kotlin code assumed to be hosted by Gradle
  - Implicit imports
  - Gradle Kotlin DSL in the classpath
  - Script compilation dependencies coming from Gradle
  - Custom IDE script editor support

The Gradle Kotlin DSL sugar

DRAFT

- Kotlin friendly extensions of the Gradle API (KClass, reified)
- Statically compiled
- Dynamically generated for model elements contributed by plugins
  - how and when they are available
  - what to do when they aren't (explain a bit, link to docs)
- Configuration avoidance by default

Migrating from Groovy

Migrating from Groovy

Migrating from Groovy

- Migrate a build from Groovy DSL to Kotlin DSL

Migrating from Groovy

- Migrate a build from Groovy DSL to Kotlin DSL
- Look at some other migration use cases

Migrating from Groovy

Demonstration

What benefits?

DRAFT

- type-safety
- discoverability
- documentation and navigation
- refactorings

Shared declarations

e.g. dependencies

TODO

Shared functions

TODO

Script Plugins

DRAFT

- to script plugins
- to precompiled script plugins

Interoperability

When things go south

DRAFT

- Dokka
  - requires `closureOf<T> {}` and other tricks
- this will be fixed in 0.9.18
  - https://github.com/Kotlin/dokka/pull/358
  - show how it'll look like then

Resources for migration

DRAFT

- Migration guide
  - https://guides.gradle.org/migrating-build-logic-from-groovy-to-kotlin/
- Gradle user manual
  - both Groovy/Kotlin snippets
  - best place to learn how to do what with each DSL, and compare
- TODO animated gif showing groovy/kotlin samples

Migration strategies

DRAFT

- Kotlin and Groovy build logic can coexist
  - mechanical step by step migration possible
  - migrating doesn't block your team
- to get the most benefit
  - from the outer to the inner
  - it's easier when kotlin build logic drives groovy build logic than the other way around
- make it easier by preparing your build first
  - by applying Gradle fundamentals and best practices (`buildSrc`, `plugins {}`)

Taking a step back

Organize build logic

Organize build logic

- `buildSrc`

Organize build logic

- `buildSrc`
- Gradle Plugins, Gradle Plugins, Gradle Plugins

Organize build logic

- `buildSrc`
- Gradle Plugins, Gradle Plugins, Gradle Plugins
- `plugins {},plugins {},plugins {}`

Organize build logic

- `buildSrc`
- Gradle Plugins, Gradle Plugins, Gradle Plugins
- `plugins {},plugins {},plugins {}`
- Basically, apply Gradle fundamentals and best practices
  - docs.gradle.org/current/userguide/userguide.html#best-practices

Organize build logic

- `buildSrc`
- Gradle Plugins, Gradle Plugins, Gradle Plugins
- `plugins {},plugins {},plugins {}`
- Basically, apply Gradle fundamentals and best practices
  - docs.gradle.org/current/userguide/userguide.html#best-practices
- Profit

Authoring type-safe plugins

Authoring type-safe plugins

- Plugins contribute to the DSL

Authoring type-safe plugins

- Plugins contribute to the DSL
- They should do so in a type-safe manner

Authoring type-safe plugins

- Plugins contribute to the DSL
- They should do so in a type-safe manner
  - Don’t expose `groovy.lang.Closure<*>` taking methods

Authoring type-safe plugins

- Plugins contribute to the DSL
- They should do so in a type-safe manner
  - Don’t expose `groovy.lang.Closure<*>` taking methods
  - Use strong types instead, like `Action<T>`

Authoring type-safe plugins

- Plugins contribute to the DSL
- They should do so in a type-safe manner
  - Don’t expose `groovy.lang.Closure<*>` taking methods
  - Use strong types instead, like `Action<T>`
  - If written in Kotlin, prefer `Action<T>` over Kotlin lambdas

Authoring type-safe plugins

- Plugins contribute to the DSL
- They should do so in a type-safe manner
  - Don't expose `groovy.lang.Closure<*>` taking methods
  - Use strong types instead, like `Action<T>`
  - If written in Kotlin, prefer `Action<T>` over Kotlin lambdas
- Again, apply Gradle fundamentals and best practices
  - gradle.org/guides/?q=Plugin%20Development

About performance

DRAFT

- kotlin-dsl numbers, good and bad
- compilation is the biggest bottleneck
  - you can reuse compilation results for scripts already compiled on CI via remote build cache
  - build cache helps you also with the rest of your build
  - Gradle Enterprise provides enterprise ready cache backend with replication, monitoring, node management etc
- general Gradle performance advices → performance guide
- plans

Wrapping up

Ready for general use

Gradle 5.0 is when the Gradle Kotlin DSL is ready for general use!

Please give it a try with Gradle 5.0-M1!

gradle.org/release-candidate

Gradle Kotlin DSL Team

- Chris Beams @cbeams
- Rodrigo B. de Oliveira @rodrigobamboo
- myself
- contributors from other teams at Gradle
- even from some Groovy commiters ツ

Gradle Kotlin DSL Community

We wouldn’t be here without the community!

- Very friendly and active Kotlin community
- Bug reports, of course
- But also pull-requests, code reviews, **documentation**, support to others

Thank you!

Join us at gradle.org/slack

Questions

Gradle 5.0-M1
    gradle.org/release-candidate

Slides
    eskatos.github.io/kotlinconf2018-type-safe-build-logic

Documentation
    docs.gradle.org

Issue tracker
    github.com/gradle/kotlin-dsl/issues

Slack
    gradle.org/slack

We're hiring!
    gradle.com/careers

Thank you