Don’t Let Ephemeral CI Kill Your Developer Productivity

Louis Jacomet - Gradle

Gradle

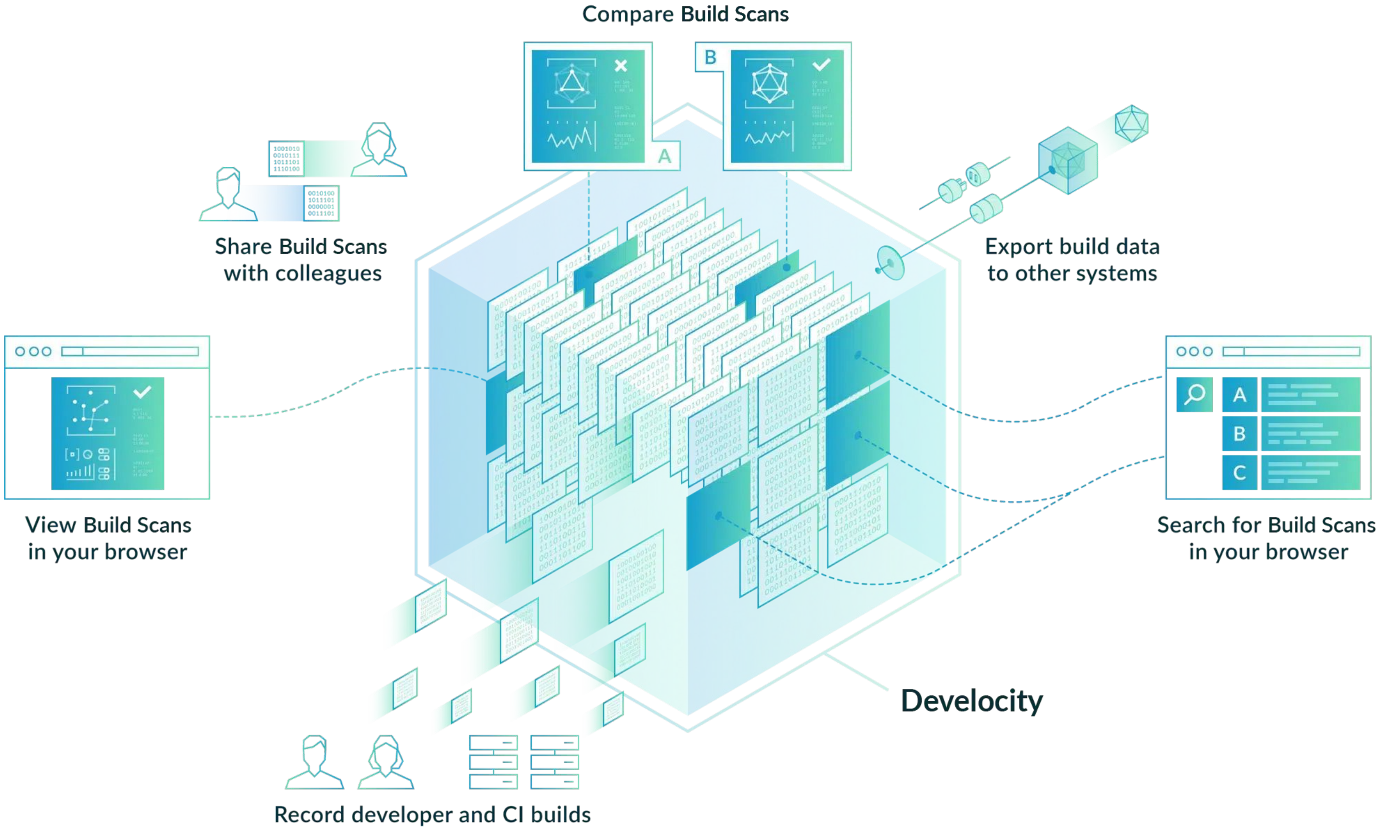
Louis Jacomet

```
speaker {
   company = "Gradle"
   joined = 2018
   position = "Support Team Lead and more …"
   previously = "Dependency Management, JVM plugins"
   past = listOf(
      "Terracotta / Ehcache" in 2013,
      "Devoxx Belgium Committee" in 2012,
      "Contractor" in 2002,
      "Java 'Hello, World!'" in 1997
   )
   failures = generateSequence(code) { bugs }
   social = listOf("@ljacomet@foojay.social”, “@ljacomet")
   github = "ljacomet"
   web = "https://jacomet.dev"
   extra = "Not fully figured out how to stay out of
management !?!"
}
```

Gradle Build Tool

- Apache 2.0 licensed build tool
- JVM based
- Kotlin and Groovy configuration DSLs
- 30+ millions downloads / month
- Extensive plugin ecosystem

DEVELOCITY



Compare Build Scans

Share Build Scans with colleagues

Export build data to other systems

View Build Scans in your browser

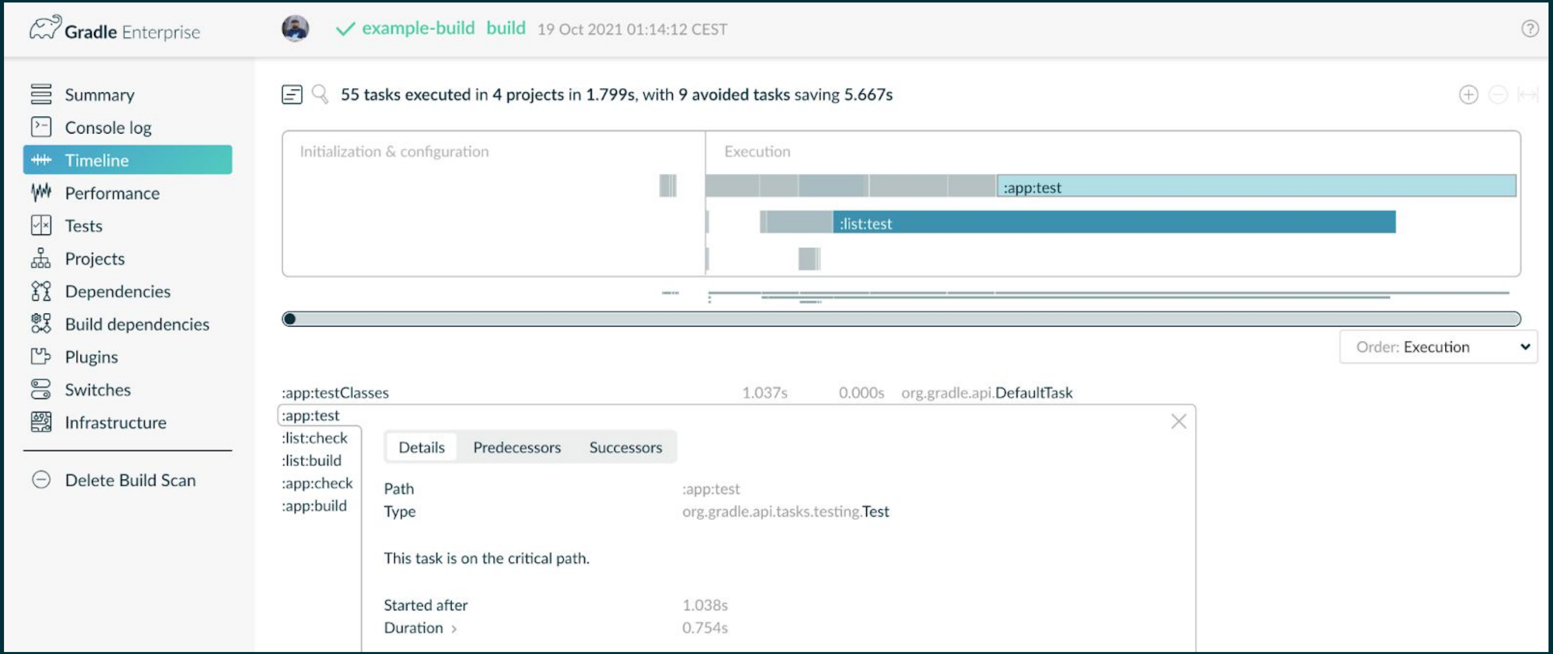
Search for Build Scans in your browser

Develocity

Record developer and CI builds

- Gradle
- Maven
- Bazel
- Sbt

Build scans



✓ example-build build 19 Oct 2021 01:14:12 CEST

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

Order: Execution

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

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

This task is on the critical path.

Started after 1.038s
Duration › 0.754s

Table of Contents

- Problem statement
- Gradle build profile
- Possible actions

Problem statement

- Gradle Build Tool performance

  vs.

- Ephemeral environments trend

Developer productivity and build performance?

- Only one aspect …

- But “Fast build” is not the goal

- “As fast as possible build” is the goal at Gradle

Gradle Build Tool performance

- Enable parallel execution
- Enable the Gradle daemon
- Enable the configuration cache
- Enable incremental build for custom tasks
- Enable the build cache
- Create build for specific developer workflows
- Increase the heap size
- Optimize configuration
- Optimize dependency resolution
- Optimize [Java|Android] projects

Ephemeral environments

- Industry trend
- Isolation → no state problems
- Short lived → no clean up jobs

Gradle Build Tool performance

- Enable parallel execution ✅
- Enable the Gradle daemon ❓
- Enable the configuration cache ❓
- Enable incremental build for custom tasks ❌
- Enable the build cache ✅
- Create build for specific developer workflows ✅
- Increase the heap size ✅
- Optimize configuration ✅
- Optimize dependency resolution ❌
- Optimize [Java|Android] projects ❓

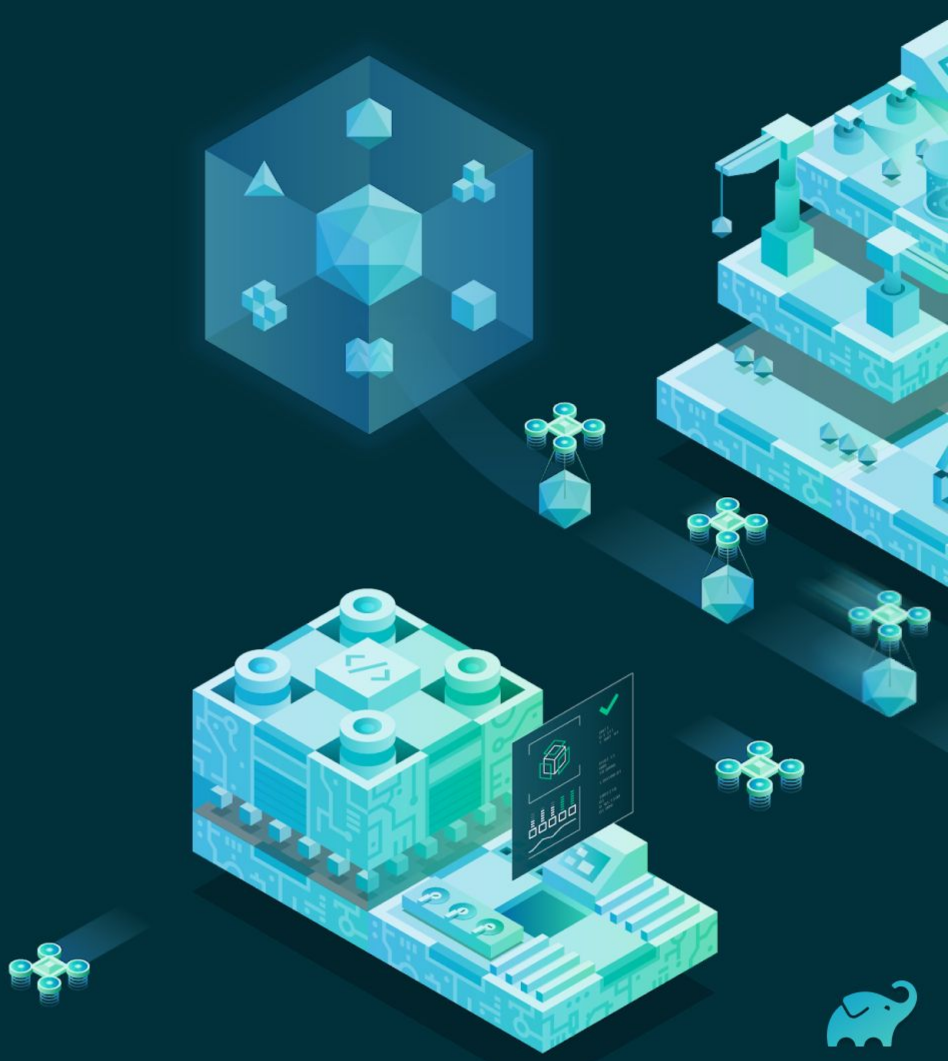
Key Gradle Build Tool performance elements

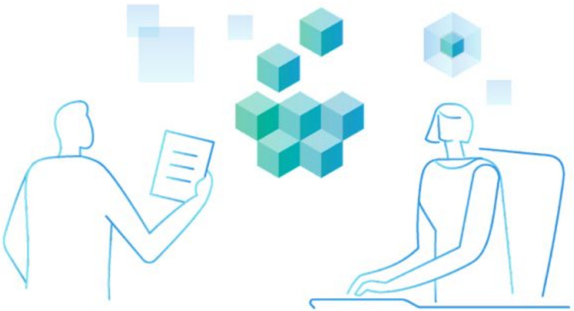
- Caches
  - Dependency cache
  - Task cache
  - …
- Incrementality
  - Execution history
- Parallelism
  - Tasks
  - Tests

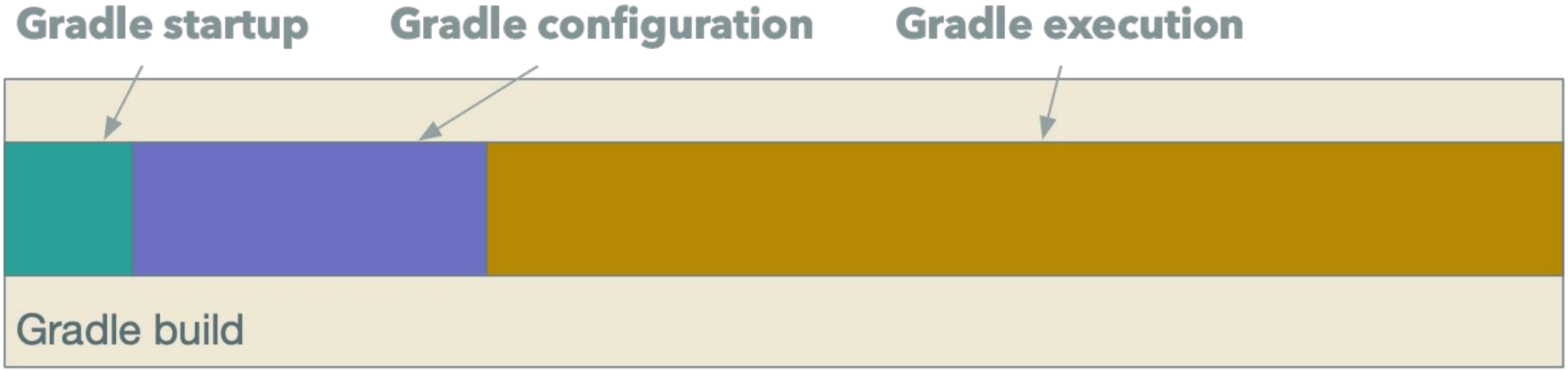
Key Gradle Build Tool performance elements

- Caches ❓
  - Dependency cache
  - Task cache
  - …
- Incrementality ❌
  - Execution history
- Parallelism ✅
  - Tasks
  - Tests

Gradle Build profile







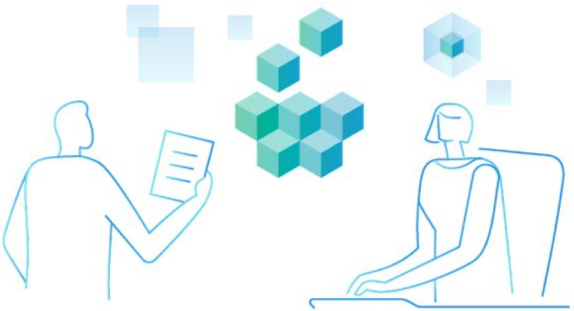
Gradle startup

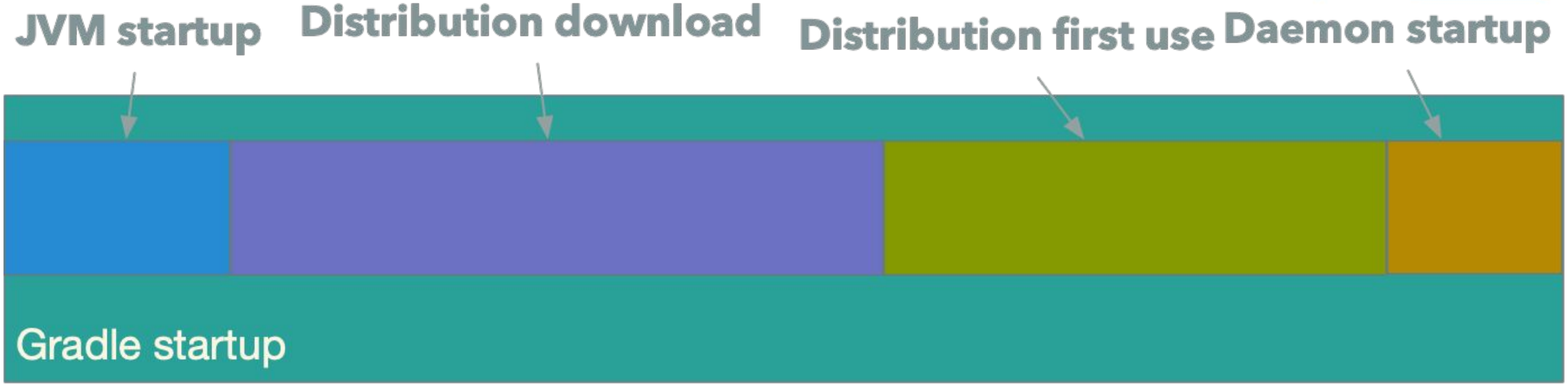
Gradle configuration

Gradle execution

Gradle build

Bars not at scale





JVM startup

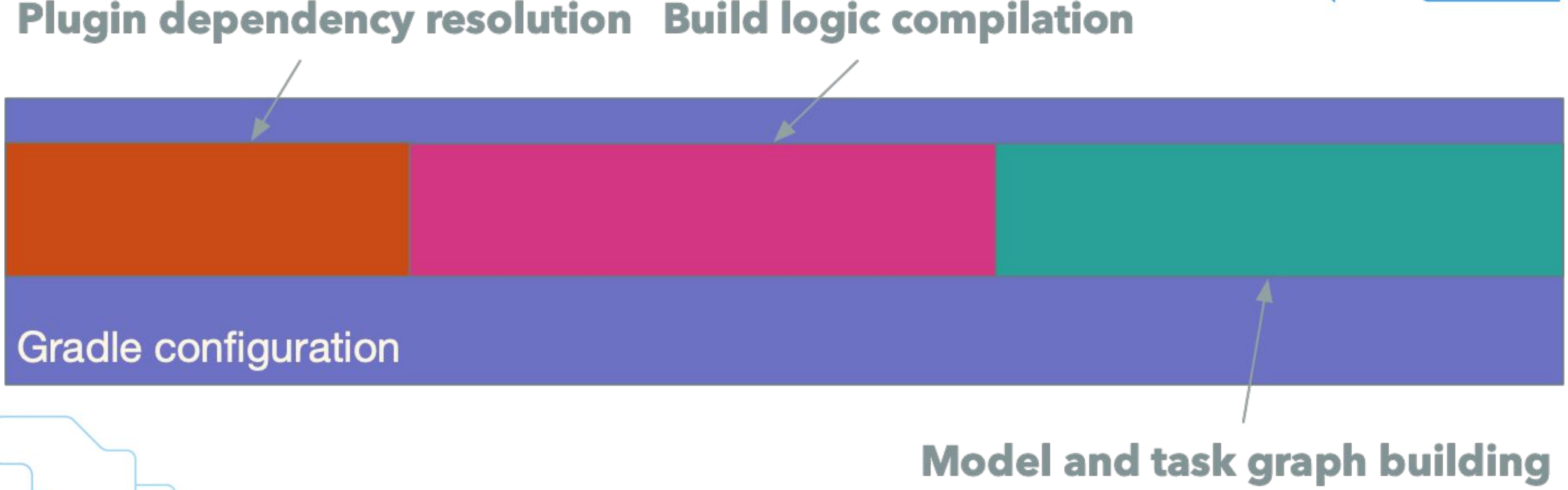
Distribution download

Distribution first use

Daemon startup

Gradle startup

Bars not at scale



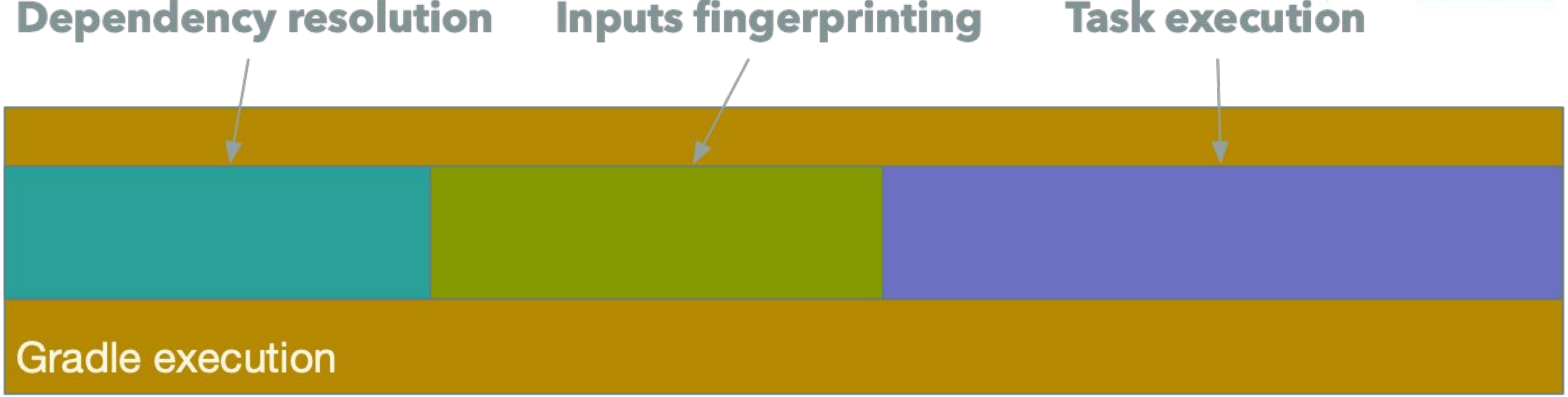
Plugin dependency resolution

Build logic compilation

Gradle configuration

Model and task graph building

Bars not at scale



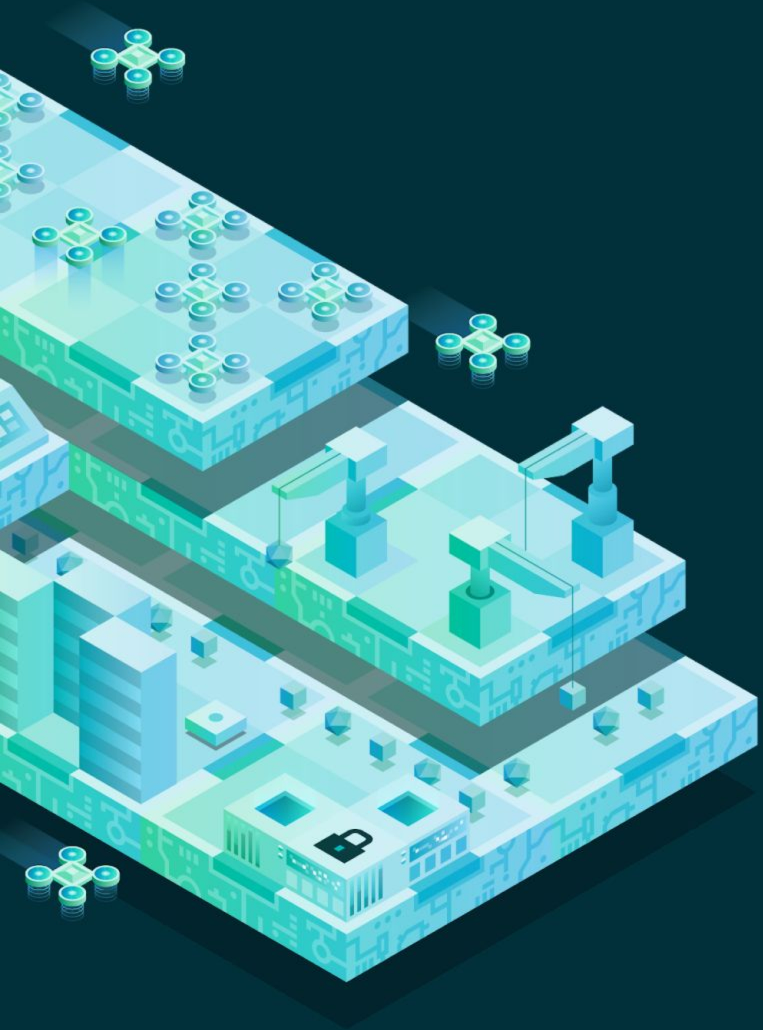
Dependency resolution

Inputs fingerprinting

Task execution

Gradle execution

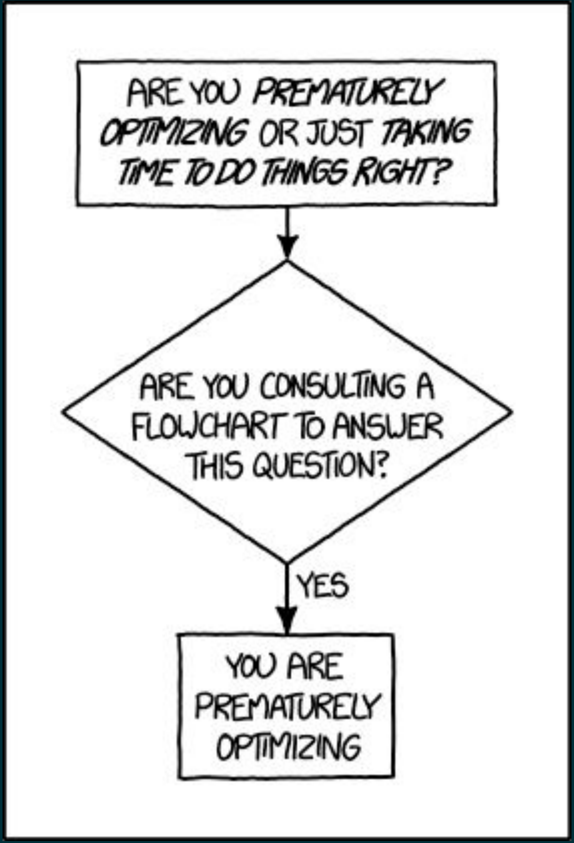
Bars not at scale



Possible actions

Dealing with performance

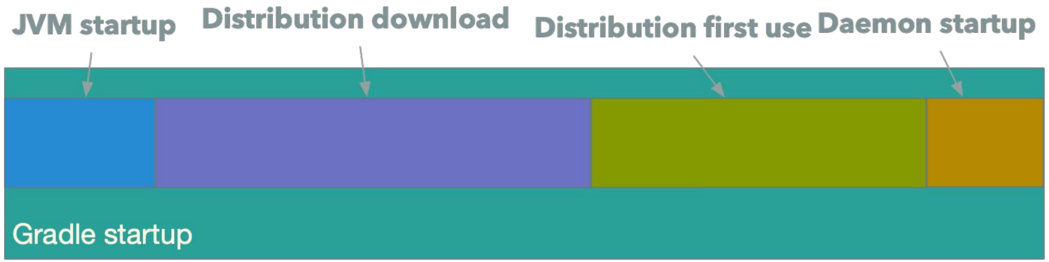
- Measure
- Change
- Measure
- Compare


ARE YOU PREMATURELY OPTIMIZING OR JUST TAKING TIME TO DO THINGS RIGHT?
ARE YOU CONSULTING A FLOWCHART TO ANSWER THIS QUESTION?
YES
YOU ARE PREMATURELY OPTIMIZING

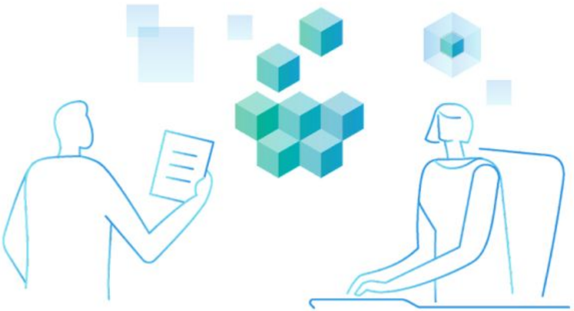
https://xkcd.com/1691/

Optimize Gradle startup

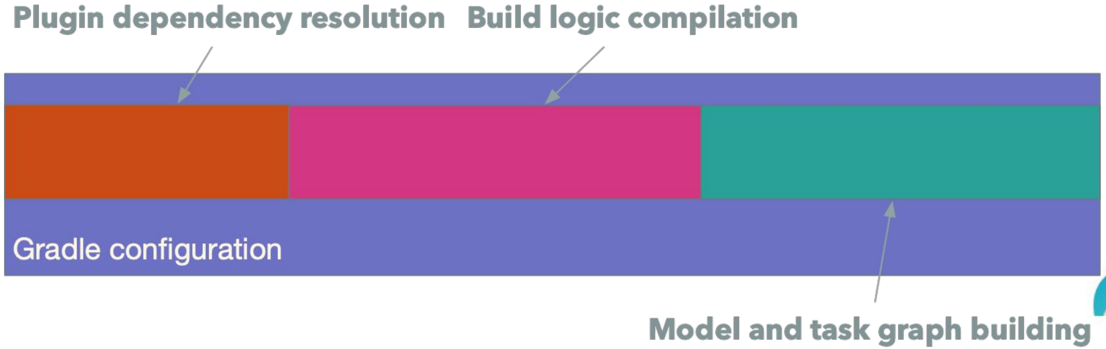
- Distribution availability
  - Always use the `-bin` one
  - Already in the image / docker file / …
  - OR Downloaded from a closer location
  - OR Save and restore `<GUH>/wrapper/dists`
- Prime distribution
  - Run it once to have the first use elements
  - OR Save and restore `<GUH>/caches/<version>/generated-gradle-jars`



Optimize Gradle configuration

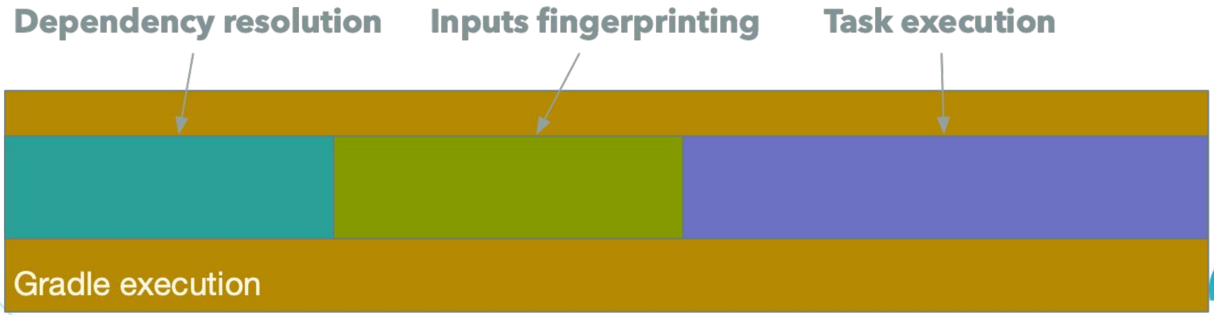


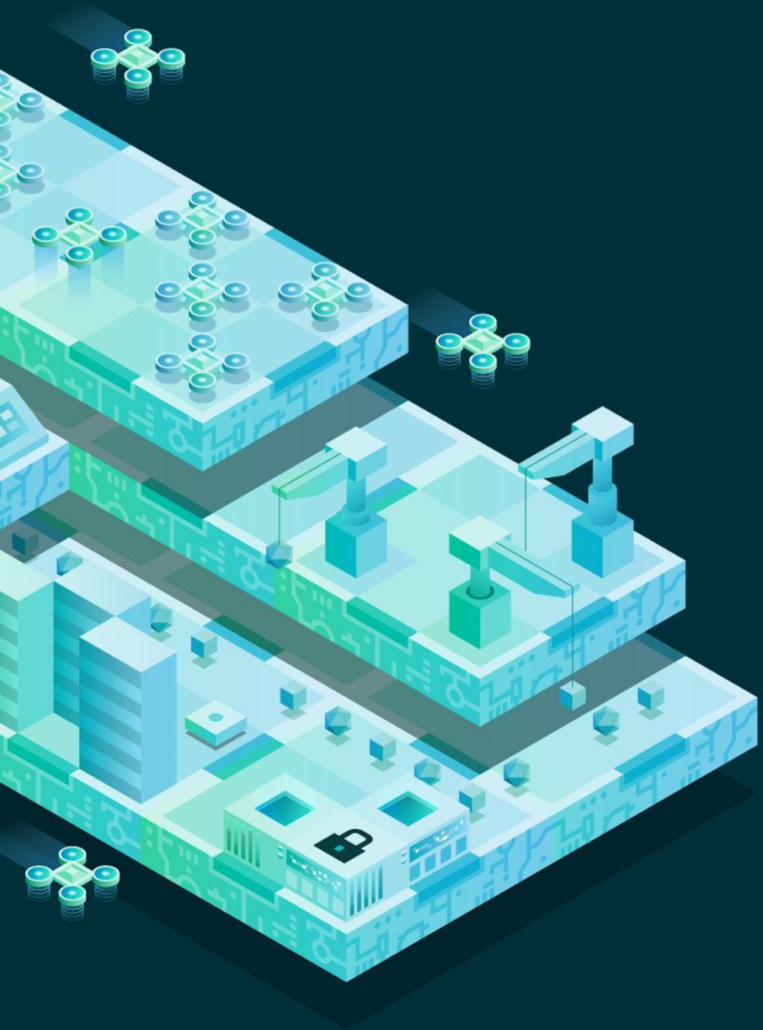
- Dependency cache
  - Read-only cache feature
  - OR save and restore `<GUH>/caches/modules-2`
- Script compilation cache
  - Remote build cache
  - OR save and restore `<GUH>/caches/<version>/kotlin-dsl`, `<GUH>/caches/<version>/scripts` and `<GUH>/caches/jars-9`



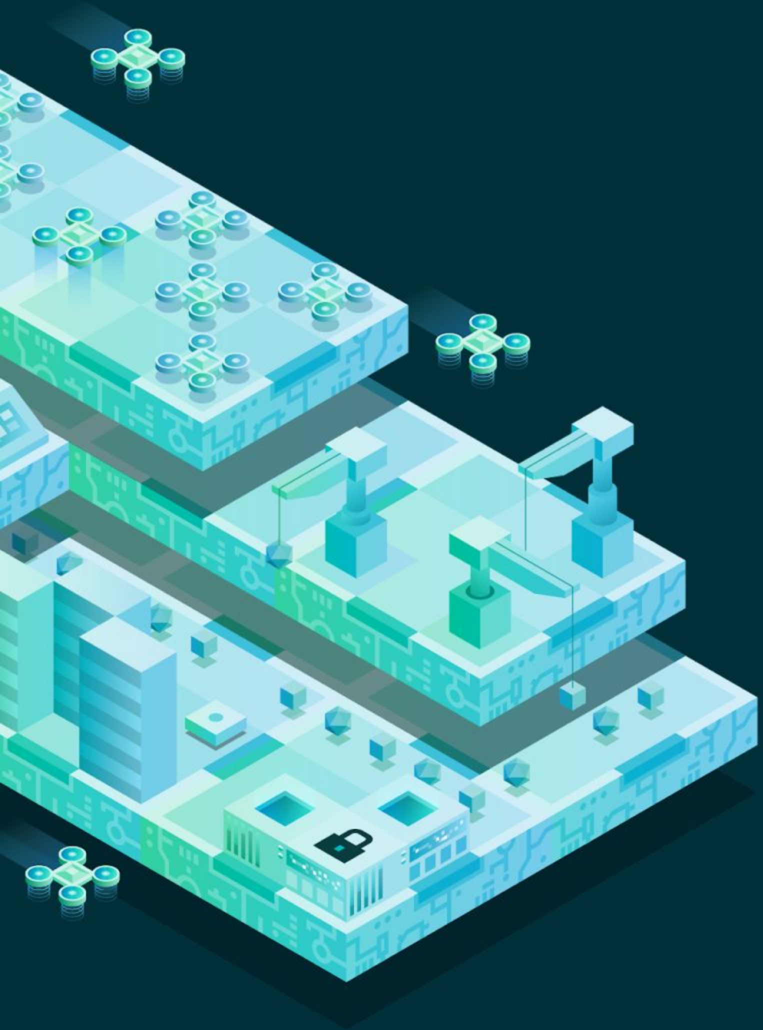
Optimize Gradle execution

- *Dependency cache*
- Task execution cache
    - Remote build cache
    - OR save and restore `<GUH>/caches/build-cache-1`
    - (Android mostly) save and restore `<GUH>/caches/transforms-3`
- Provisioned toolchains cache
    - Save and restore `<GUH>/jdks`

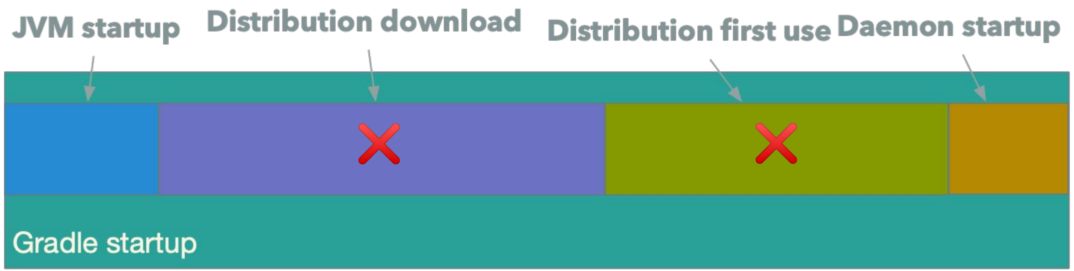




In practice



Conclusion



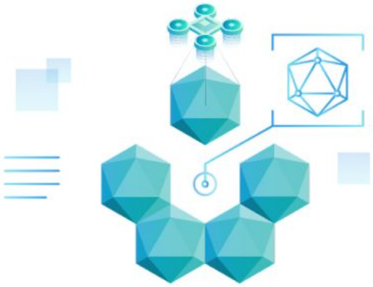
JVM startup

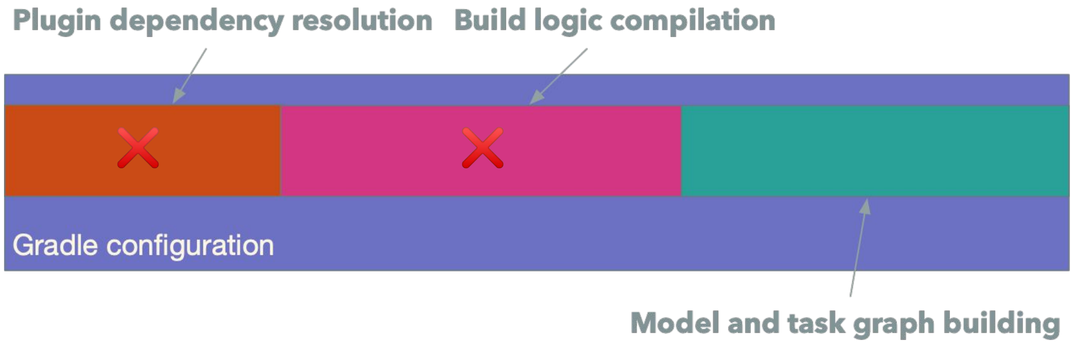
Distribution download

Distribution first use

Daemon startup

Gradle startup



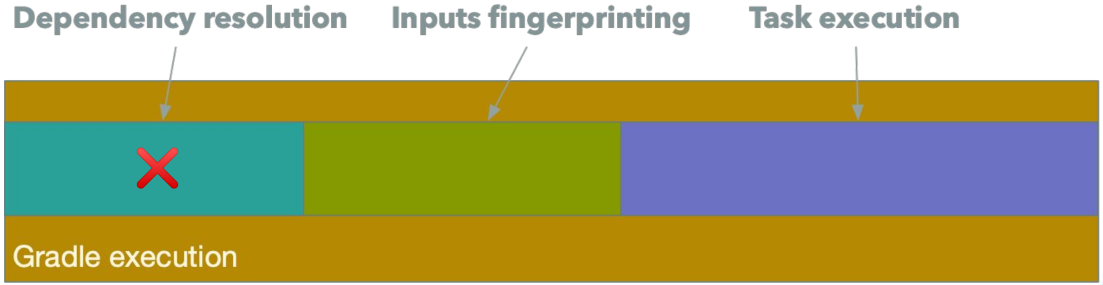


Plugin dependency resolution

Build logic compilation

Gradle configuration

Model and task graph building



Dependency resolution

Inputs fingerprinting

Task execution

Gradle execution

Thank you!