Gradle & Android

Etienne Studer,
VP of Product Tooling

Gradle

Gradle

Motivation

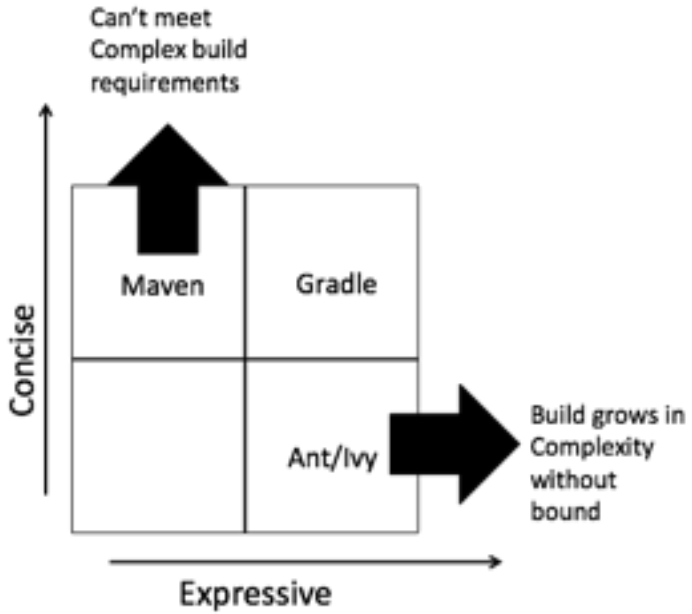
Gradle

Facts

- implicit complexity of the domain
- multi-language support
- resource/code generation
- platform diversity

Gradle

Complexity



Gradle

Conciseness

```
android {
    compileSdkVersion 20
    buildToolsVersion "20.0.0"

    flavorDimensions   "version", "abi"

    productFlavors {
        free {
            flavorDimension   "version"
        }
        paid {
            flavorDimension   "version"
        }

        arm {
            flavorDimension   "abi"
        }
        x86 {
            flavorDimension   "abi"
        }
    }
```

Polyglot

- multiple languages
- multiple teams
- coordinated releases

Gradle

Product delivery

- more than just building APKs
- documentation with tested code examples
- auto-provisioned dev environments
- automated release process
- …

2,000
Active Gradle Using Developers

2,000
Software Components

300,000
Gradle Executions Per Week

1,000
Release Builds per Day

Gradle

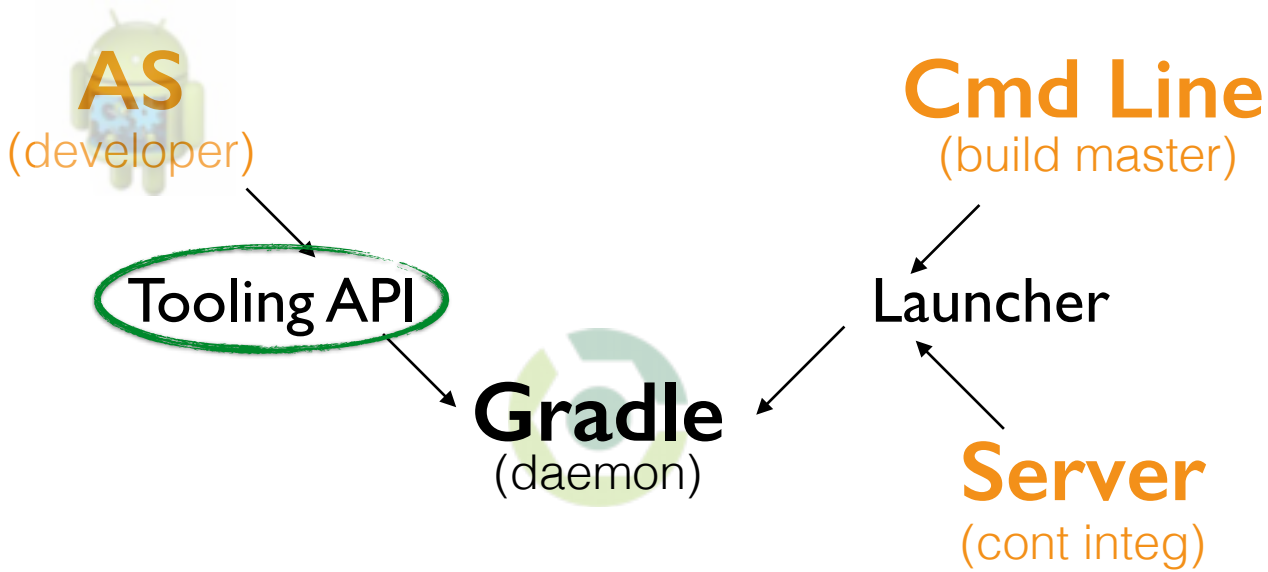
The Android Build System

Android Studio IDE +
Android Gradle Plugin +
Gradle Platform +
Android Tooling

Gradle

Have a single truth of build logic

Put all build logic into the build.
Derive all information from the build model.

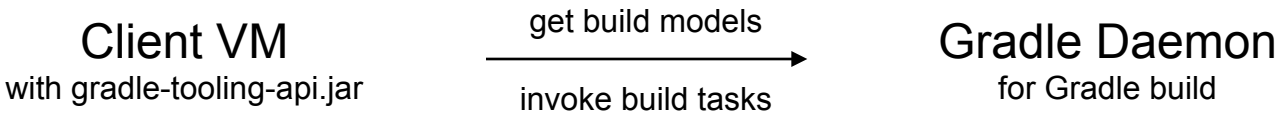


In a unified build,
Gradle is the single source of build logic.

Gradle

Tooling API

- Proxy for embedding Gradle
- Extension mechanism to provide custom models
- See DefaultAndroidProject model



Gradle

Tooling API

- Backward & forward compatible
  From 1.0 to 2.8
- Runtime Isolation
  Separate daemon process
- High Level Services
  Build Cancellation,Continuous Mode, Test Execution, etc.
- Build Event Model
  Life-cycle events, task events, test events

Gradle

Performance

Gradle

Goal

Minimize the build time while
using as little memory as needed.

Observation

Typically, not much changes in the build between consecutive invocations of the build.

When little changes in the build, little work should be done by the build.

Approach

Performance enhancements are achieved through evolutionary improvements and revolutionary changes.

Building with Gradle - today

2-phase build:

- Configuration phase
- Execution phase

Execution Phase

- Incremental build feature
  .Only run a task if its input or output has changed since the previous run

Inputs —> Task —> Outputs

Define inputs and outputs on your custom tasks.

Gradle

Execution Phase

```
class ConversionTask extends DefaultTask {


   @InputFiles
   def sourceFiles

   @OutputDirectory
   def targetDirectory

   @TaskAction
   def doSomeWork() {
       // consume the sourceFiles and write the result
       // to a file in the targetDirectory
   }


}


task foo(type: MyTask) {
  sourceFiles = files('input.txt');
  targetDirectory = file('build/result')
}
```

Execution Phase

- Continuous build feature
  .Keep the session running between build runs

```
gradlew test -t
```

```
Starting 3rd build in daemon [uptime: 26.37 secs, performance: 98%, memory: 8% of 3.8 GB]
Continuous build is an incubating feature.
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:compileTestJava UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE

BUILD SUCCESSFUL

Total time: 1.191 secs

Waiting for changes to input files of tasks... (ctrl-d to exit)
```

Gradle

Execution Phase

- Faster incremental builds
  .Improvements to the management of File checksums
  .Most notable for almost uptodate builds

- Faster compilation for continuous builds
  .Compiler daemon is kept alive for the entire ContB session

Evolutionary improvements with each release!

Gradle

Configuration Phase

- Imperative build logic (input)
  .Even when providing a declarative DSL
  .Lazy evaluation tricks
  .Lazy collection tricks
  .Project#afterEvaluate tricks

- Declarative build model (output)

*Know **all** things, build **some** things.*

Gradle

Configuration Phase

- Android
  .Full model construction at evaluation time
  .Full dependency resolution/processing at evaluation
  .Hard to know when things are ready, e.g flavors
  .Even when calling gradlew tasks

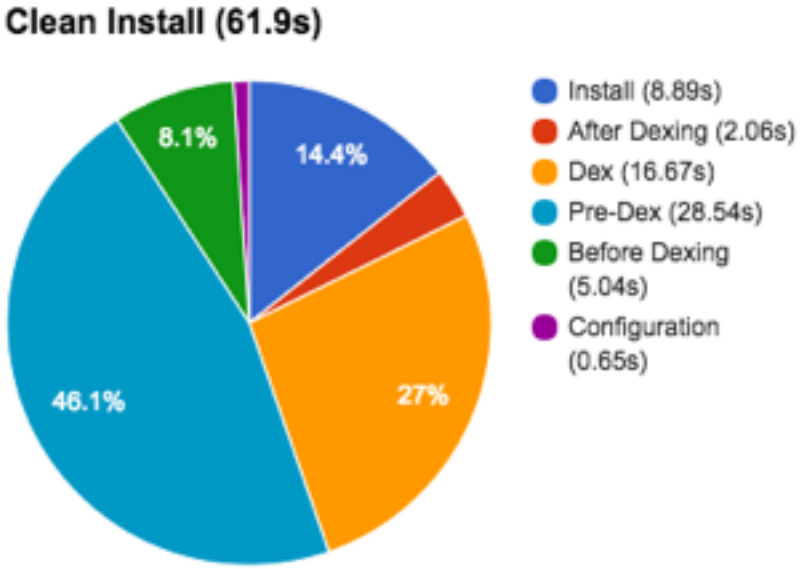
Configuration Phase

- Faster build script compilation
  .Caching the compiled form of the build scripts
  .Newer version of Groovy used for the DSL parsing

- Configure on demand
  .Attempt to only configure the projects relevant to the requested tasks

Evolutionary improvements with each release!

Android Build - Pre-dexing



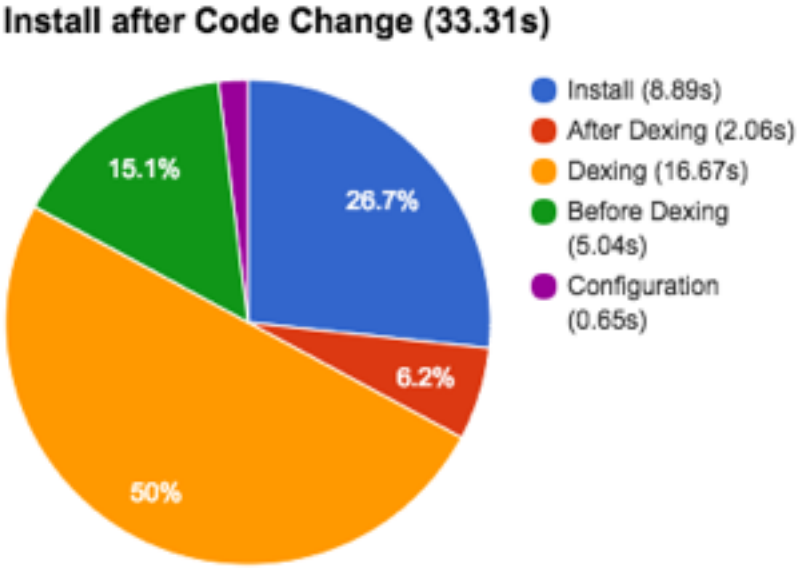
Gradle

Pre-Dexing Optimizations

- Clean Safe Cache
- Parallelization
- Distributed Cache

Gradle

Android Build - Dexing



Gradle

Dexing Optimizations

- Faster Dexing
- Incremental Dexing

Building with Gradle - experimental

The revolutionary new Configuration model

- From *know **all** things, build **some** things* to *know **some** things, build **some** things*

New Configuration Model

Apply the concepts already available in the Execution phase to the Configuration phase.

Describe what the model should look like and Gradle will provide the implementation.

Modeling

- Richer modeling
- Cleaner modeling
- Collaborative modeling
- Comprehensible model

Gradle

Managed types

```
@Managed
interface Picture {

  String getName()
  void setName(String name)

  List<String> getTags()

}
```

Gradle

Plugin

```
class PicturesPlugin extends RuleSource {

  @Model
  void createPicture(Picture picture) {}

  @Mutate
  void configurePicture(Picture picture) {
    picture.name = 'mypic.jpg'
    picture.tags.addAll(['nature', 'night'])
  }

}
```

DSL

```
model {
  picture(Picture) {
    name = 'mypic.jpg'
    tags.addAll(['night', 'moon'])
  }
}
```

Report

```
+ createPicture
      | Type:            Picture
      | Creator:         PicturesPlugin#createPicture
      | Rules:
        ↳ PicturesPlugin#configurePicture
    + name
          | Type:        java.lang.String
          | Value:       mypic.jpg
          | Creator:     PicturesPlugin#createPicture
    + tags
          | Type:        java.util.List<java.lang.String>
          | Value:       [nature, night, moon]
          | Creator:     PicturesPlugin#createPicture
```

Gradle

Android (experimental)

```
model {
  android {
    compileSdkVersion = 21
  }
  android.buildTypes {
     debug {
     }
     create('qa')
  }
}
```

Gradle

New Configuration Model

- Configuration becomes parallelizable
- Managed types are externalizable
- Model becomes reusable between invocations

Revolutionary change!

Roadmap 6-12 months

- Radically improve performance

- Gentrify dependency management

No guarantees or commitments.

Gradle

Performance

Apply

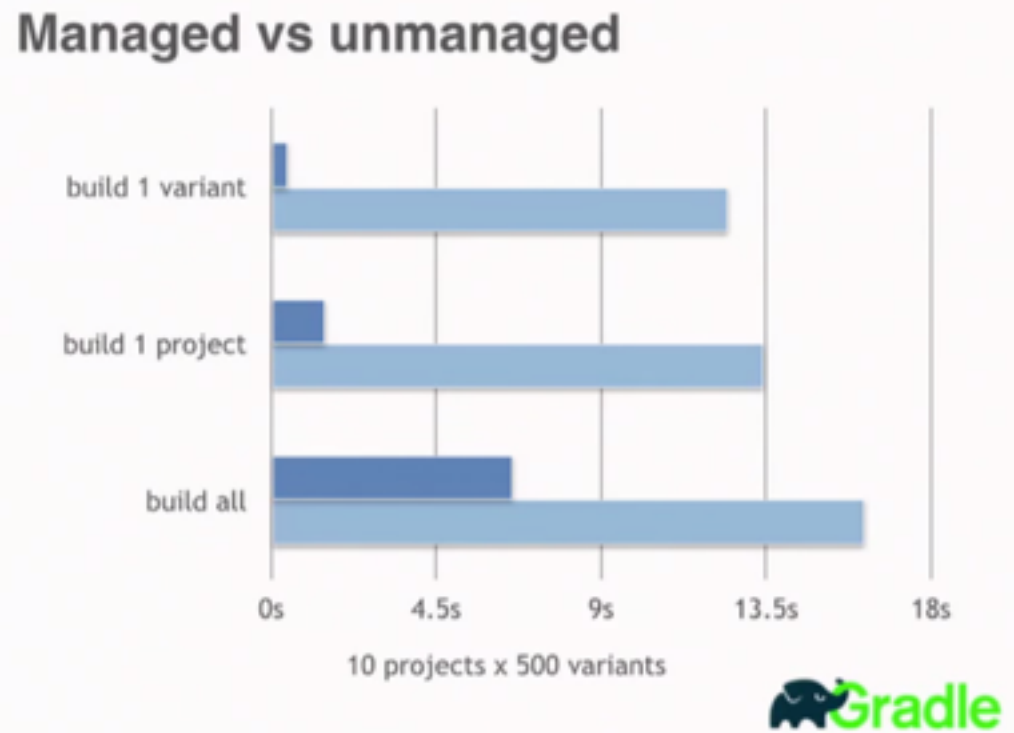
- Fix hotspots
- Cache and reuse
- Work in parallel
- Work in background

To

1. Build configuration
2. Dependency resolution
3. (Task Execution)

Gradle

Performance



Gradle

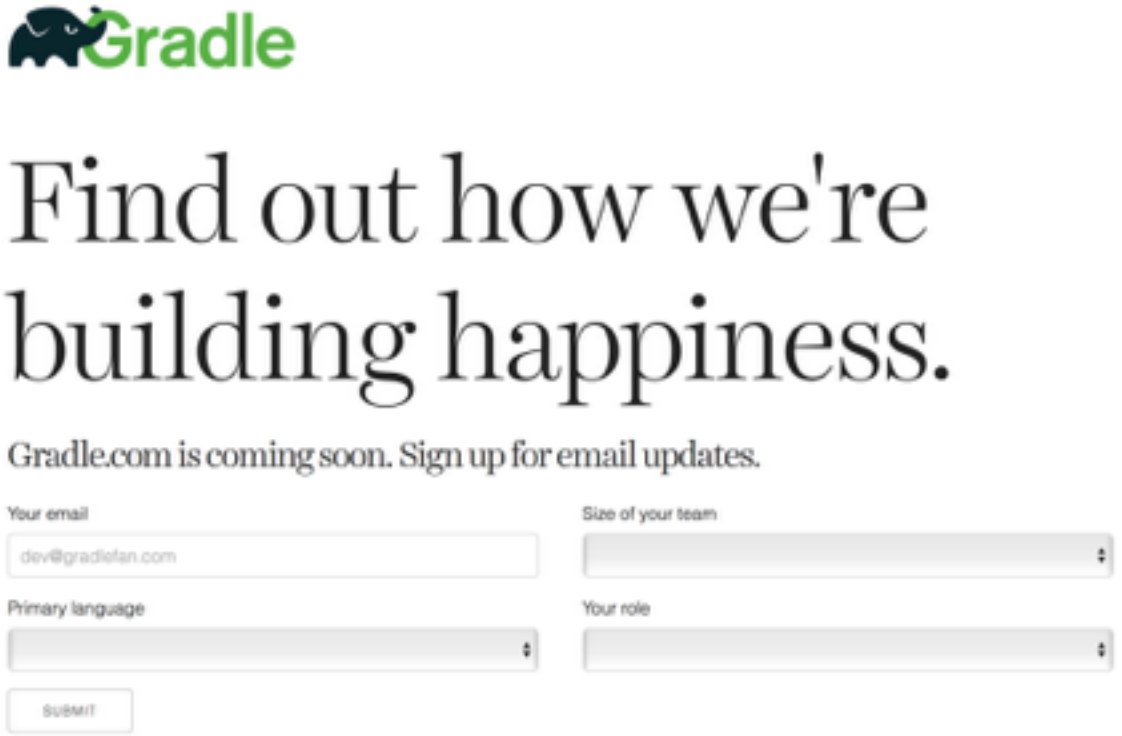
Dependency Management

- dependency management for all the variants of the project

- variant-aware dependency management for the external dependencies

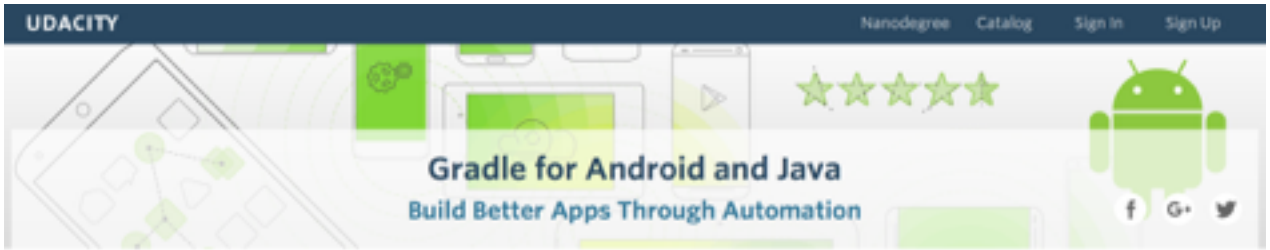
Gradle

Gradle.com



Gradle

MOOC



Advanced

Built by gradle Google

Approx. 6 weeks

Assumes 6hr/wk (work at your own pace)

Join 9,704 Students

Course Summary

This course explores how the Gradle build tool compiles and packages apps, and you'll learn to customize the build process. The first half of this course is for anyone interested in Gradle, build automation, and continuous delivery of software.

The latter half of the course reveals the magic that happens after you hit the "Run" button in Android Studio. You'll also explore advanced Android topics, learning to configure free vs paid app flavors, create and integrate Android libraries, test your app, and prepare your app for the Play Store.

Start Free Course

Start free course

Free

You get

- Instructor videos
- Learn by doing exercises

Gradle

Gradle is hiring

http://gradle.org/gradle-jobs

Gradle & Android

Etienne Studer,
VP of Product Tooling

Gradle