## WE’LL LOOK INTO

- Gradle Momentum
- My favourite latest features
- Forecast
- Q&A
- STICKERS!

## GRADLE (THE MANAGEMENT SUMMARY)

- Multi purpose software automation tool
- Build, automate and deliver better software, faster
- Cross-platform
- Language agnostic
- Apache v2 licensed
- A Build Tool + Cloud Services

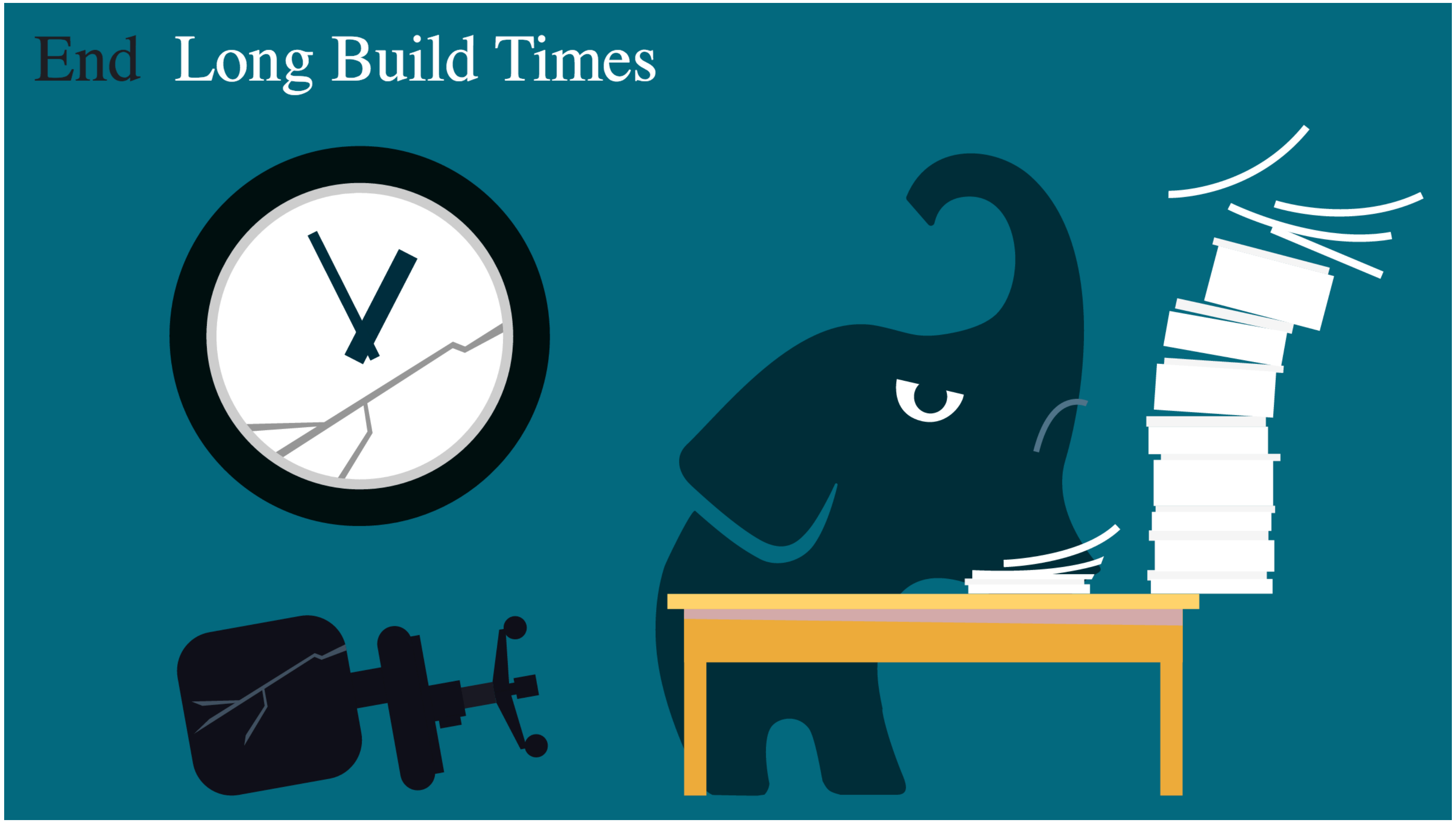
## MOMENTUM

- ~20 full time engineers on Gradle core
- 3.0 released on August 15th 2016
- 3.4 RC-2 released yesterday
- Releasing every 4 - 6 weeks.

## NEWEST FEATURES

## WHOSE BUILD IS TOO FAST?

## PERFORMANCE IS A FEATURE



## DEDICATED PERFORMANCE TEAM

- Faster test execution
- Faster IDE integration
- Faster dependency resolution
- Ongoing effort

## GRADLE DAEMON

- A long-lived background process
- Avoids costly jvm bootstrapping
- Benefits from warmed up hotspot compilation
- On by default since 3.0
- More communicative
- Leveraging daemon more in the future

## KOTLIN BASED GRADLE DSL

## KOTLIN (MANAGEMENT SUMMARY)

- Statically typed
- 1.0 released in 2016
- Driven by pragmatism
- Invented and maintained by Jetbrains
- Considerable uptake (particularly in the Android community).

## MOTIVATION

- Current DSL was not designed for
  - performance
  - tooling friendlyness
- Limitations on bringing patterns and techniques from application level to build level

## ENTICING OPPORTUNITIES

- Proper IDE support
  - Code completion
  - Refactoring
  - Documentation lookup
- Crafting DSLs with ease
  - While keeping build scripts clean and declarative

## CURRENT STATE

- Working closely with Jetbrains
- 1.1-M03 support in Gradle 3.3
- v0.7 in Gradle 3.4

## KOTLIN IN GRADLE

```
apply<ApplicationPlugin>()

configure<ApplicationPluginConvention> {
    mainClassName = "samples.HelloWorld"
}

configure<JavaPluginConvention> {
    setSourceCompatibility(1.7)
}

repositories {
    jcenter()
}

dependencies {
    testCompile("junit:junit:4.12")
}
```

## COMPOSITE BUILDS

## COMPOSITE BUILDS

Defined in a `settings.gradle` file:

```
// settings.gradle
rootProject.name='adhoc'

includeBuild '../my-app'
includeBuild '../my-utils'
```

Or passed via command line argument:

```
> gradle --include-build ../my-utils run
```

## COMPILE AVOIDANCE

## SO FAR

- Task up-to-date check been there forever
- Relies on tasks inputs/outputs model

## WE CAN DO BETTER

## BETTER COMPILE AVOIDANCE

- Gradle now detects **ABI** changes
- Dramatically improves incremental build performance

## MORE JAVA GOODNESS IN 3.4

- Better incremental java compiler
  - Working on making incremental compilation default
- **java-library plugin**
  - less classpath leakage
  - better poms than maven

```
apply plugin:'java-library'

dependencies {
   api 'org.apache.commons:commons-math3:3.6.1'
   implementation 'com.google.guava:guava:21.0'
}
```

## TALKING ABOUT UP-TO-DATE CHECKS...

## WE ARE REUSING RESULTS...

from **last time**
when we ran **this build**
on **this machine.**

## WE CAN DO BETTER

## WHY NOT...

from **anytime before**
when we ran **any build**
**anywhere**.

## BUILD CACHE (WIP)

```
> gradle clean logging:assemble
...
:native:classpathManifest
:native:compileJava FROM-CACHE
:native:compileGroovy UP-TO-DATE
:native:processResources UP-TO-DATE
:native:classes
:native:jar CACHED
:logging:compileJava FROM-CACHE
:logging:compileGroovy UP-TO-DATE
:logging:processResources UP-TO-DATE
:logging:classes
:logging:jar FROM-CACHE
:logging:assemble UP-TO-DATE

BUILD SUCCESSFUL
```

## BUILD CACHE IN ACTION

Demo

## TALKING ABOUT BUILDS

## WOULDN'T IT BE NICE IF WE COULD...

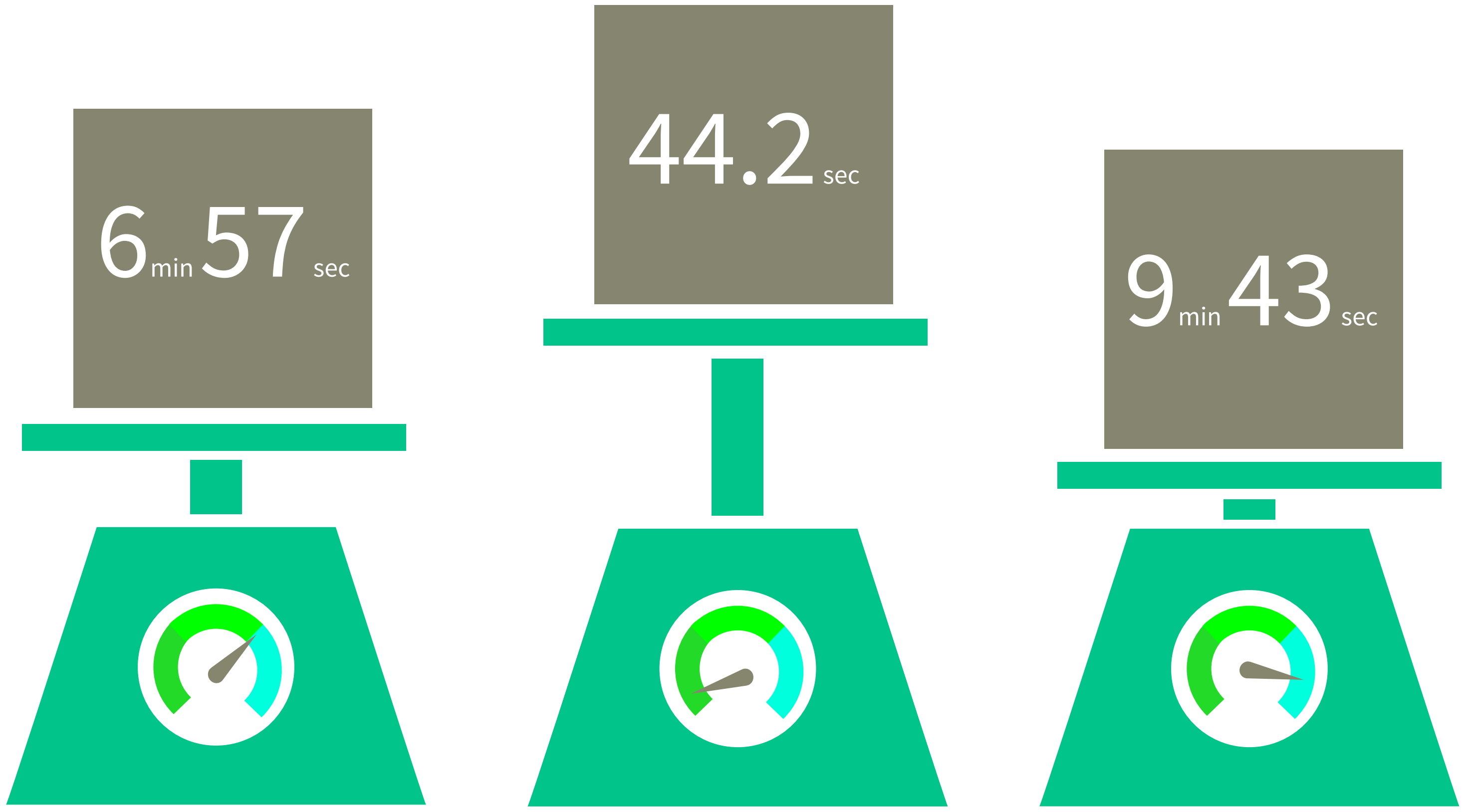
## COLLABORATE

...easily share builds to debug issues together?

## OPTIMIZE BUILD PERFORMANCE

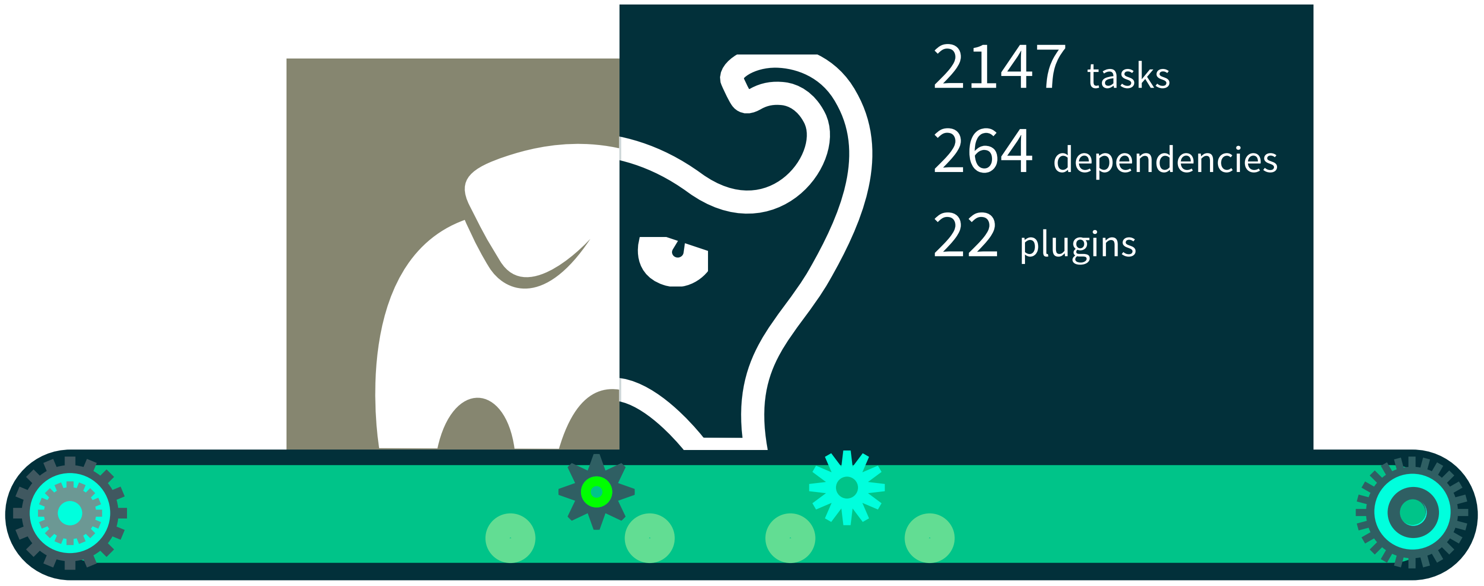
...easily understand where our build time is going and make our builds faster?

## COMPARE



...compare builds within our entire organization?

## DISCOVER

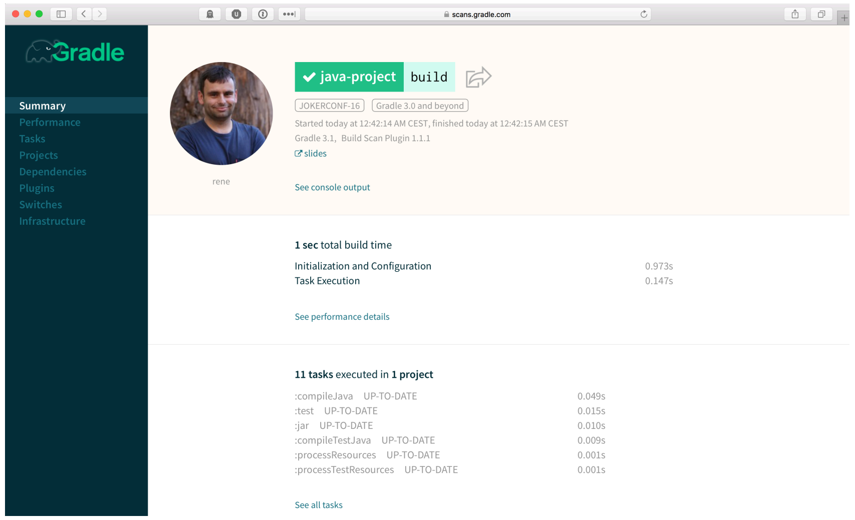


...discover

- how our software is actually being built within our entire organization?
- where our build time is going and make our builds faster?

# INTRODUCING GRADLE BUILD SCANS

- Insights into your build
- View and share via URL
- Debug, optimize and refine
- Communicate via builds
- Analyze all of your builds



scans.gradle.com

Summary
Performance
Tasks
Projects
Dependencies
Plugins
Switches
Infrastructure

rene

✓ java-project build

JOKERCONF-16 Gradle 3.0 and beyond

Started today at 12:42:14 AM CEST, finished today at 12:42:15 AM CEST

Gradle 3.1, Build Scan Plugin 1.1.1

slides

See console output

**1 sec** total build time

```
Initialization and Configuration        0.973s
Task Execution                          0.147s
```

See performance details

**11 tasks** executed in **1 project**

```
:compileJava   UP-TO-DATE                    0.049s
:test   UP-TO-DATE                           0.015s
:jar   UP-TO-DATE                            0.010s
:compileTestJava   UP-TO-DATE                0.009s
:processResources   UP-TO-DATE               0.001s
:processTestResources   UP-TO-DATE           0.001s
```

See all tasks

## GRADLE INC

**Motto**: Build Happiness

**Mission**: To revolutionize the way software is built and shipped.

**We're Hiring**: Gradle is hiring front-end, back-end, and core software engineers. Visit gradle.org/jobs to apply.

## THANK YOU!

- Gradle-Script-Kotlin project: https://github.com/gradle/gradle-script-kotlin
- Composite Builds at LinkedIn: https://www.youtube.com/watch?v=krv317ZOWGg
- Slides and code : https://github.com/breskeby/talks/tree/master/07022017-jfokus-stockholm/
- Gradle Build Scans : https://gradle.com