Developer Productivity Journey

Ankit Srivastava & Denis Cabasson

DPE Summit | Apple, Inc. | November 2022

Agenda

- Migration of the build from Maven to Gradle
- Lessons learned
- Troubleshooting local configuration
- User experience focus
- Code owners for mono repos

Migration from Maven to Gradle

Why a new build tool?

Landscape

- ~350 engineers - Mono repo - ~360 maven modules - Build times ~25-30 mins

Challenges with Maven

- Workarounds for building and packaging artifacts
- Working with XML
- 20 mins local builds
- Lack of good incremental build support
- Very opinionated

Gradle Migration journey

Build time comparison

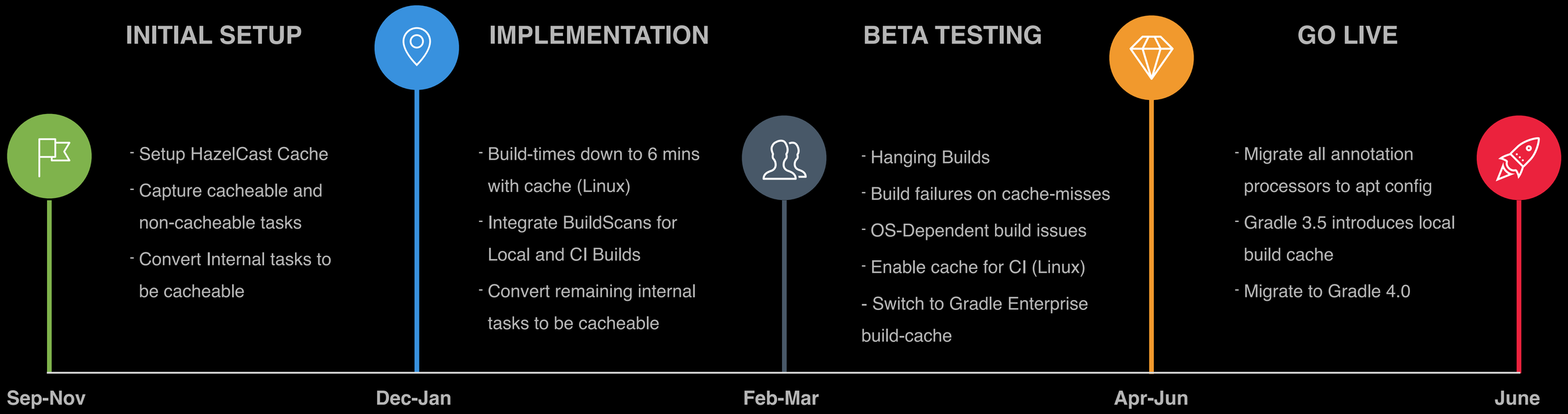
Tasks	Maven	Gradle (2.8)
Compile only	10m18s	7m41s
Compile + test	13m24s	8m57s
Compile + unit + integration	27m51s	20m28s

Lessons learned…

- Learn how to write Gradle plugins the right way
- Intermittent Unit/Int Test failures, deployment failures
- Gradle daemon used up a lot of memory
- Setup Gradle Builds in PRs early (avoid regression)
- Be ready for support after roll-out for ~3-4 months
- Plan for new modules and plugins during migration

Gradle Distributed cache

Gradle Distributed cache journey…



Gradle Build cache

Task	With Cache	Without cache
Clean assemble (CI agent)	1m20s	8m11s
Clean check (CI agent)	16m40s	27m5s
Clean assemble (Mac Pro)	3m54s	11m5s

Lessons learned…

- **Up-to-date and caching based on build inputs**
  - Too many inputs means task re-run when not needed
  - Not enough inputs means task re-use cache when it’s not correct

- **Up-to-date and caching based assume stable output**
  - Remove dates, git hashes, build numbers (or track them appropriately)
  - Generated code can be un-stable (ordering)

- Remove Symbols from JNI Libs
  - 0000000000000000 l df *ABS* 0000000000000000 /opt/teamcity-agent/buildAgent/work/ae8f77/assistant/nl/tokenizer/jni/native/src/main/native/NBEntry.c

Lessons learned…

- macOS is case-insensitive - Causes builds-cache misses

```
rm unSupportedOnWatch;
git checkout unsupportedOnWatch
```

- Do not share IntelliJ and Gradle output dirs
- Clang and Xcode version mismatch can cause cache misses
- To avoid cache misses all annotation processors should use apt classpath instead of compile
- 3rd party plugins tasks might not be cacheable (antlr, protobuf, JUnit 5)

Summary

- Went from 20 mins to 4 mins local builds for a mono-repo
- Better incremental build support
- Ability to manage the version of build-tool on very workstation
- Ability to detect differences between 2 different builds

- (11 mins * 350 engineers * 250 days) / 60 mins =
    - **16,500 developer hrs saved per year!**

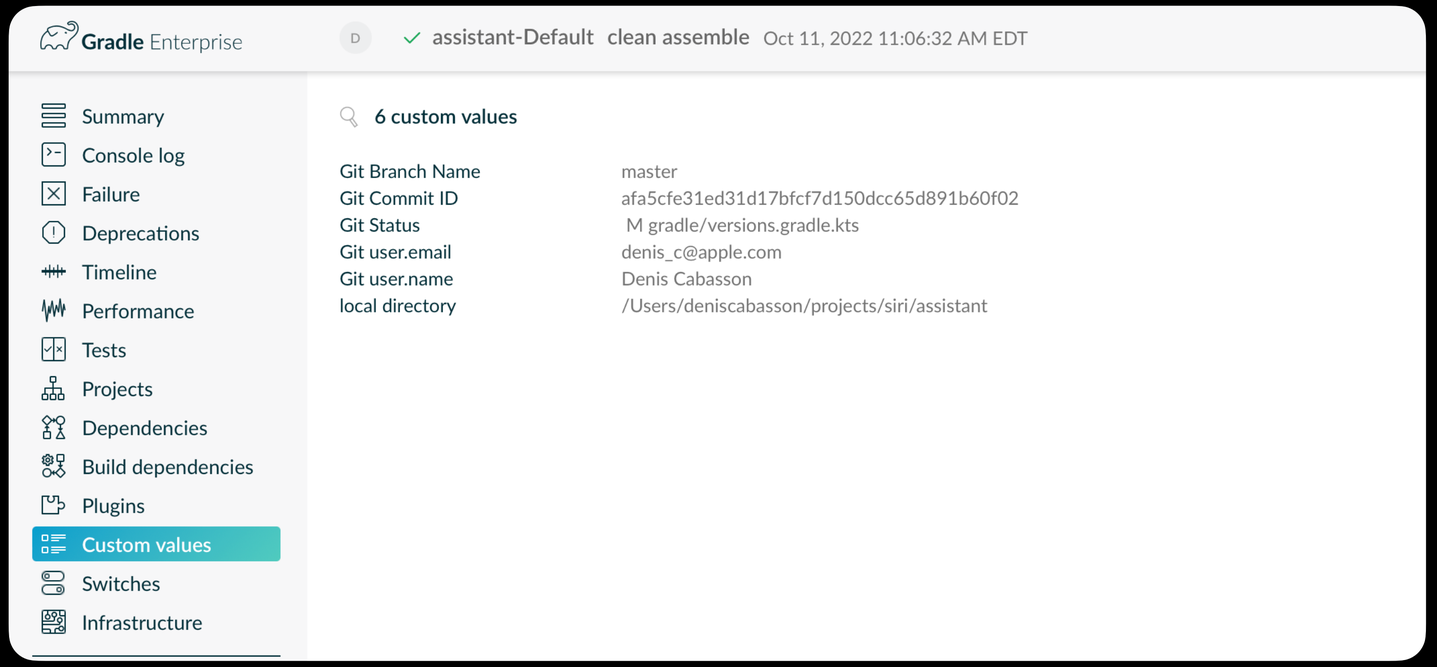
Troubleshooting local configuration

Troubleshooting local configuration

It doesn’t work on my machine

Troubleshooting local configuration

- Build scans enabled by default
- Include local information about the state



Troubleshooting local configuration

- Keep Gradle & Java up-to-date (currently Java 18/Gradle 7.5.1)
- Remove deprecated uses
- Move all configuration to Kotlin

User experience focus

User experience focus

Landscape

- Mono-repo with 20+ PR statuses
- 100k end-to-end tests with can be flaky at times
- ~1hr 20 mins per PR check run
- ~30 dependent sub-systems

Challenges

- flaky tests detection
- Identify root cause of a failure
- Reduce PR check runtime

User experience focus - What did we do?

Flaky test detection

- Record and replay 2nd party response
- Implemented automated snooze of flaky tests
- Automated re-evaluation of test results

User experience focus - What did we do?

Identify root cause failures

DrWatson which allows querying:

- Splunk and graphite for the lifespan of a test run
- Detect 3rd party API failures, dependent system failures

User experience focus - What did we do?

Reduce PR check runtime

- Implemented selective testing
- Parallel test execution for unit and end-to-end tests
- Mono repo but not mono build
  - components for slow moving modules (including jni modules)
  - Rest of the repo have binary dependencies onto components

User experience focus - What did we do?

- Feedback to developers - Github, in context
- Github statuses -> Github Checks
- Rich, timely information but keep it relevant

KPIs to track

Optional subtitle (use sentence case)

- Last commit to merge time
- Number of triggers required after the last commit
- Empty PR success rate

Optional footnote

Code owners for mono repos

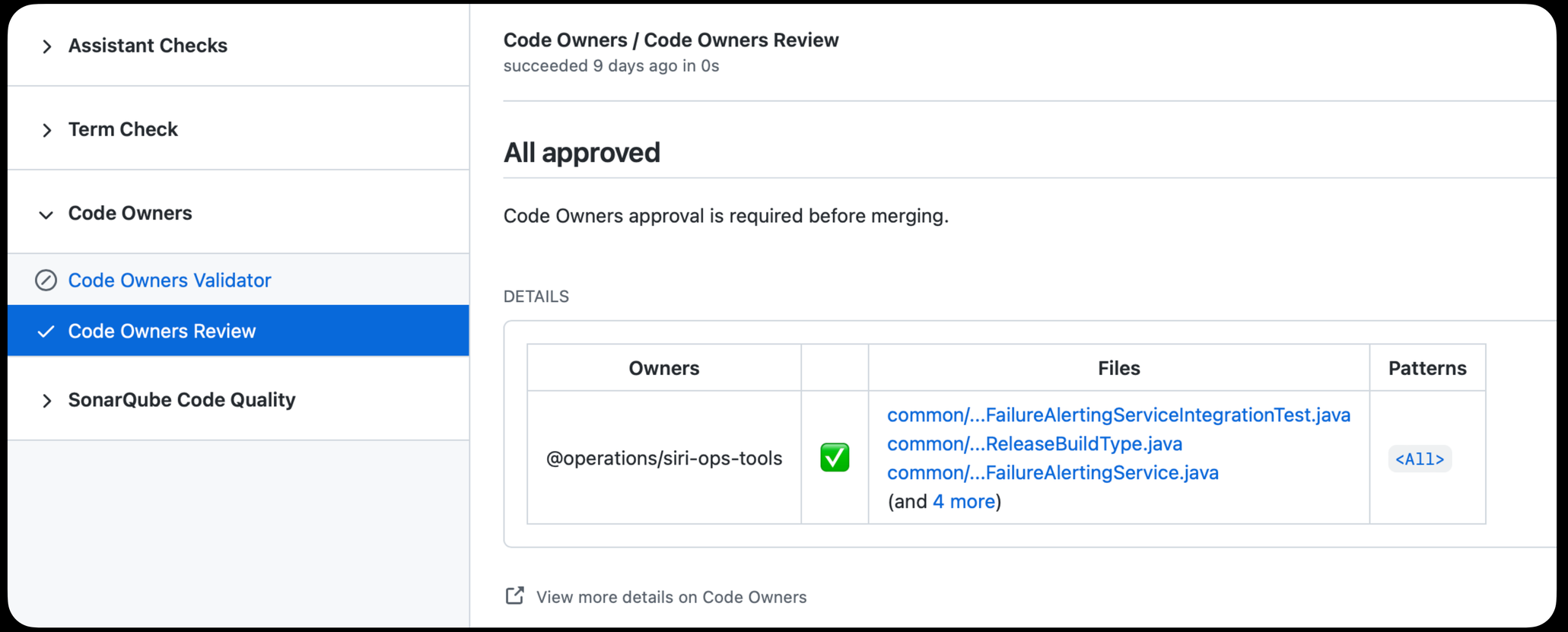
Code owners for mono repos

Optional subtitle (use sentence case)

- Alert the right people based on changed code
- Clear information to reviewers about what needs review
- Validate owners information

Optional footnote

Code owners for mono repos



Q&A