The Path to Build Happiness

Jenn Strater, Developer Advocate at Gradle

@codeJENNerator

Gradle

Jenn Strater

- Developer Advocate at Gradle
- Previously a software engineer at various companies
- Berlin <-> Minneapolis, MN, USA

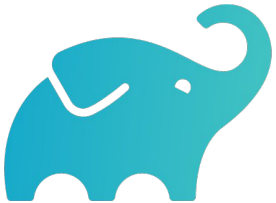
@codeJENNerator

Gradle Inc

Gradle Build Tool

Open Source

Build Automation



Gradle Enterprise

Commercial Product (on-premises)

Development Productivity

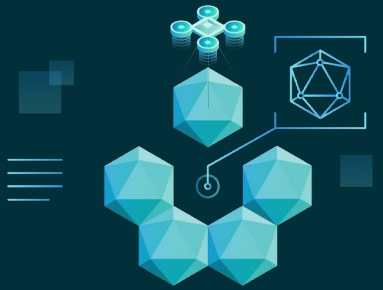
Outline

What is Developer Productivity Engineering?

Current state of Developer Productivity

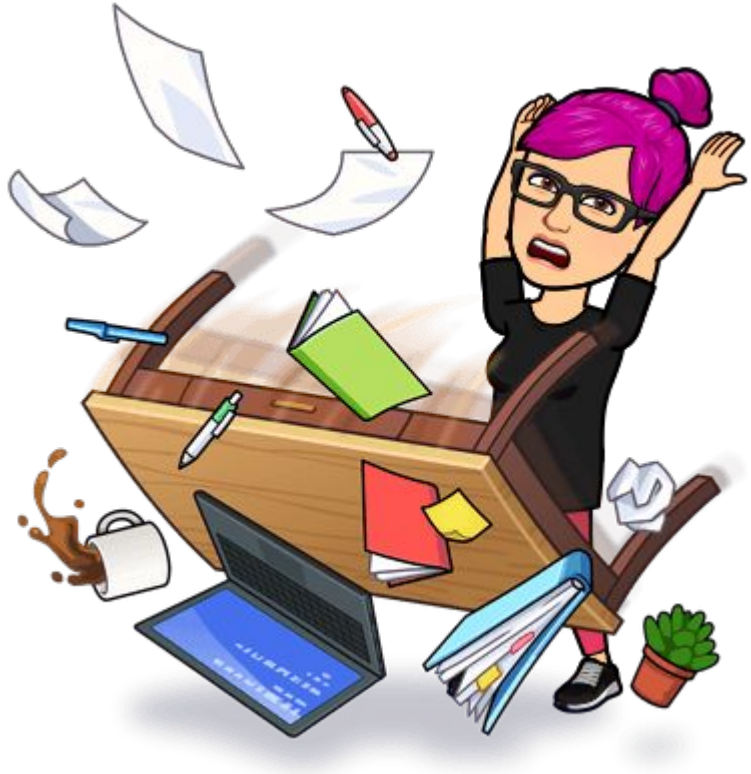
Achieving Build Happiness

- Measure
- Analyze
- Optimize
- Iterate



What is Developer Productivity Engineering?

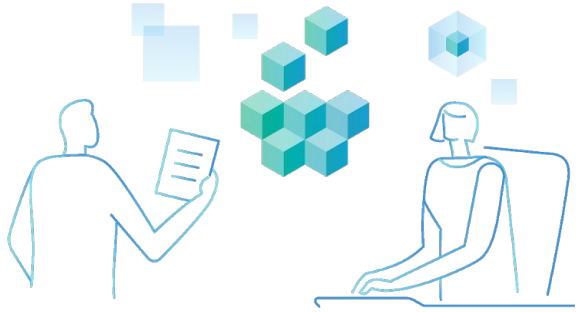
The Paradox of Success



Without intervention, as the

- Lines of Code
- No. of developers
- No. of repositories
- No. of dependencies
- No. of support tech stacks

increase, so does the frustration!



How do we get back to bliss?

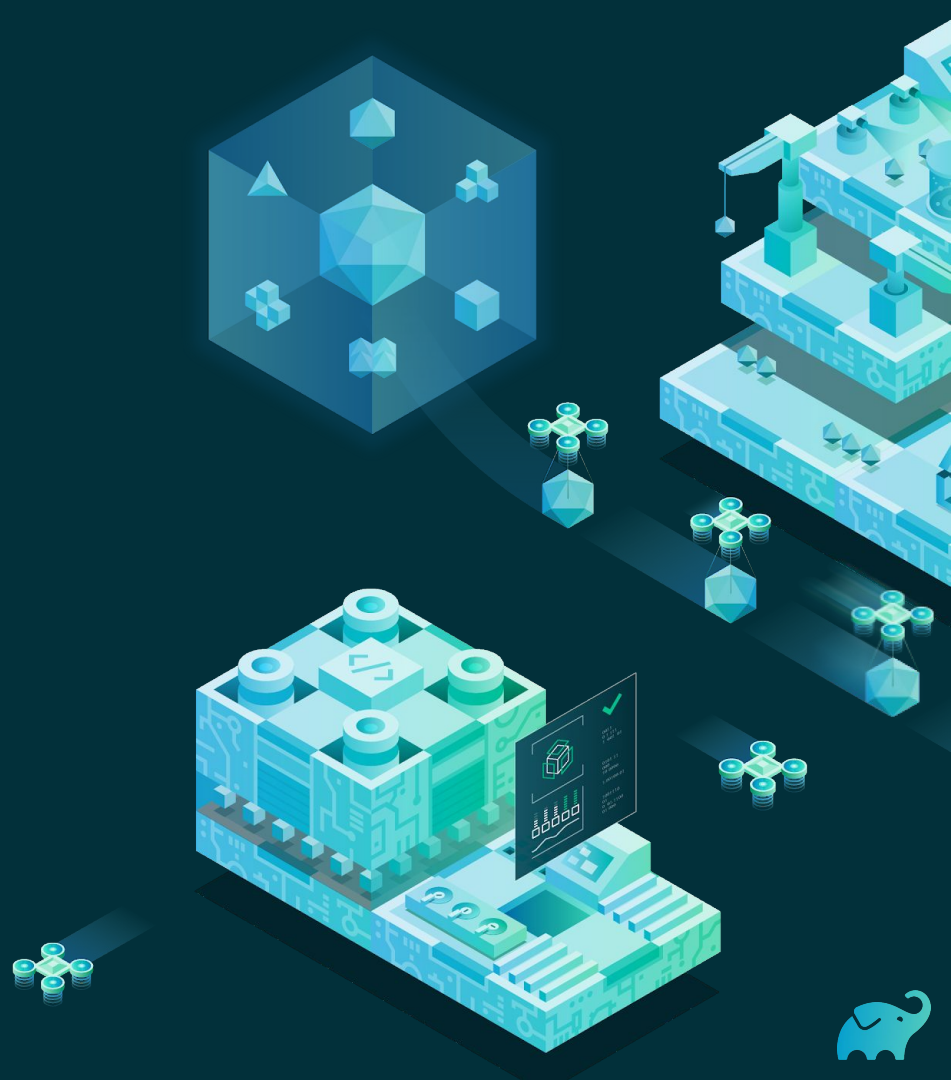
Creative Flow

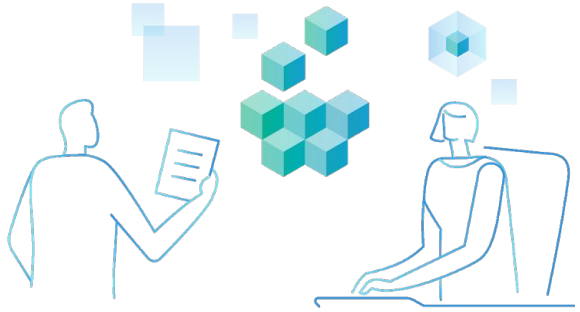
Collaborative Effectiveness

Image by #WOCinTech Chat
https://flic.kr/p/F81Nqd

Quality of Creative Flow
+ Collaborative Effectiveness

Team Productivity





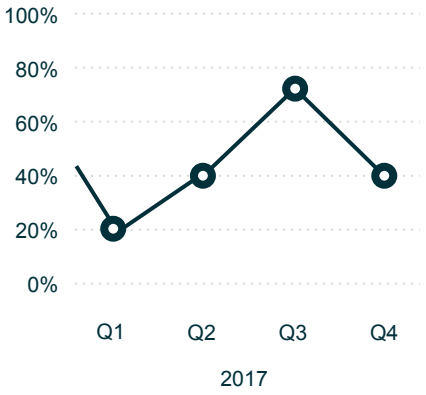
Developer Productivity Engineering is a culture where the whole organization commits to an effort to maximize developer productivity.

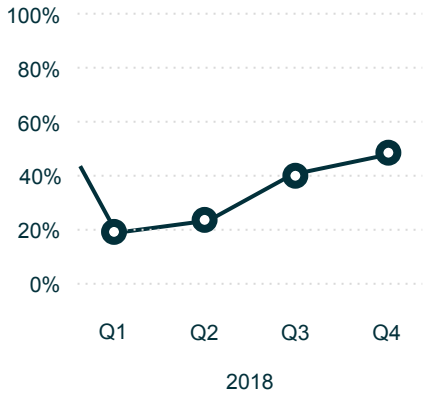
Developer Productivity Engineering KPIs

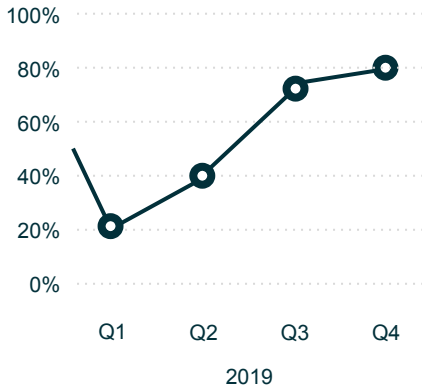
- Degree of automation
- Speed of feedback cycles
- Correctness of the feedback

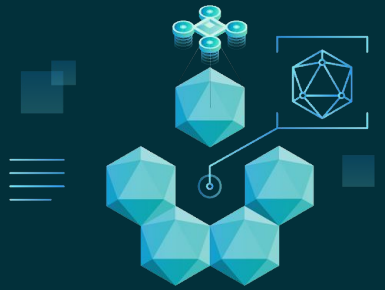
Developer Productivity Engineering

Priorities and success criteria primarily based on data that comes from a fully instrumented toolchain.





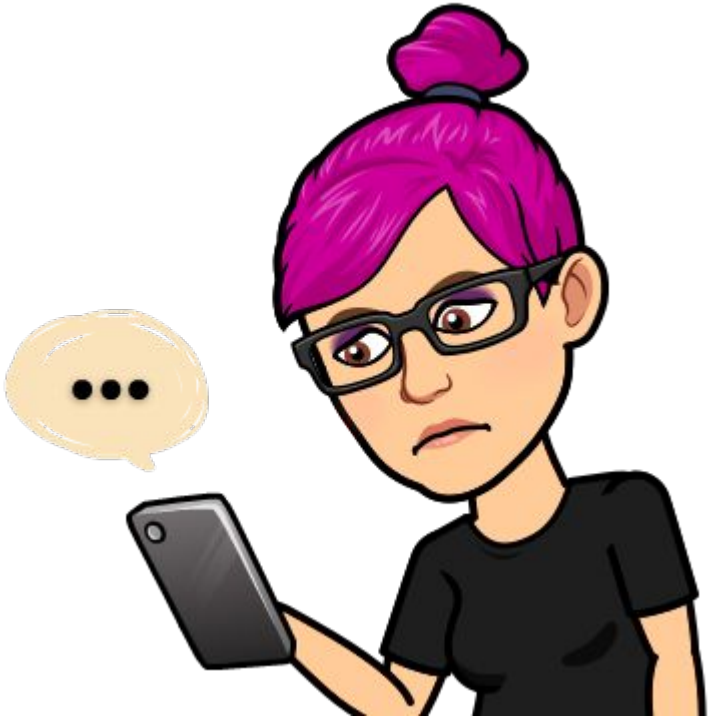


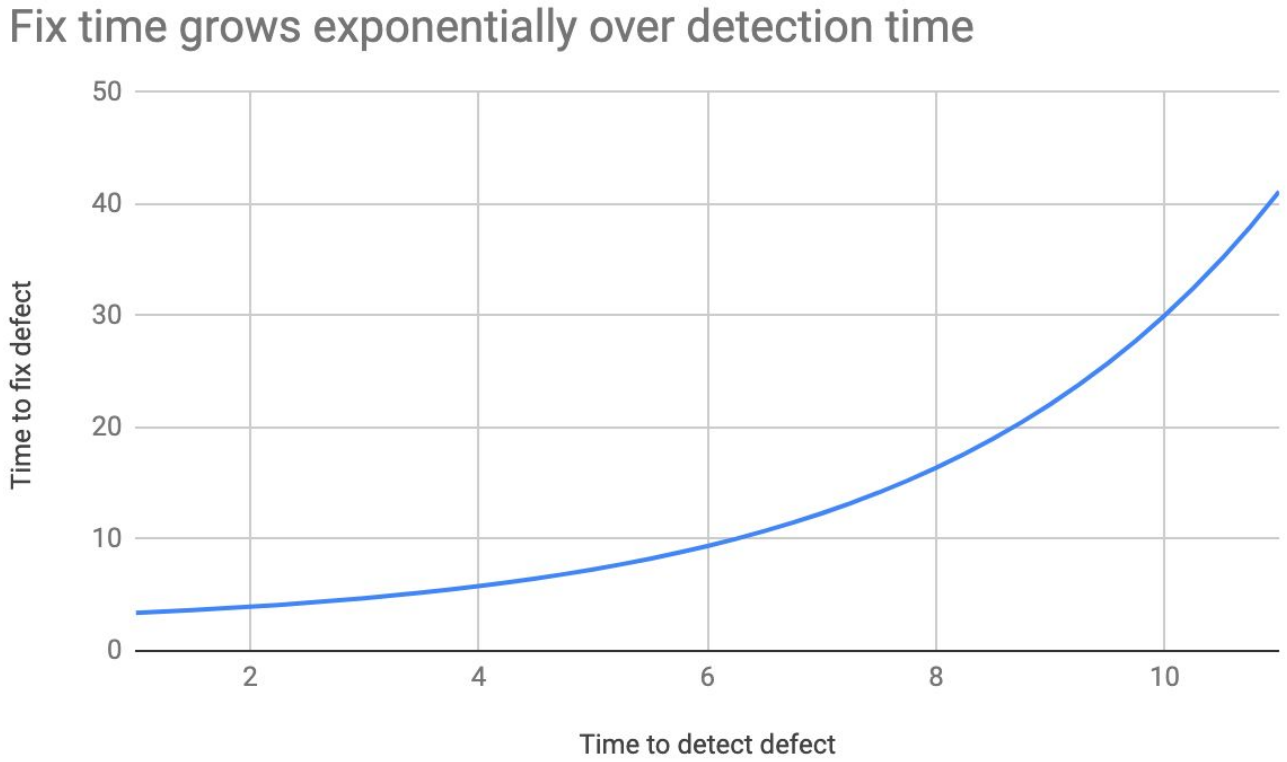


Current State of Developer Productivity Engineering

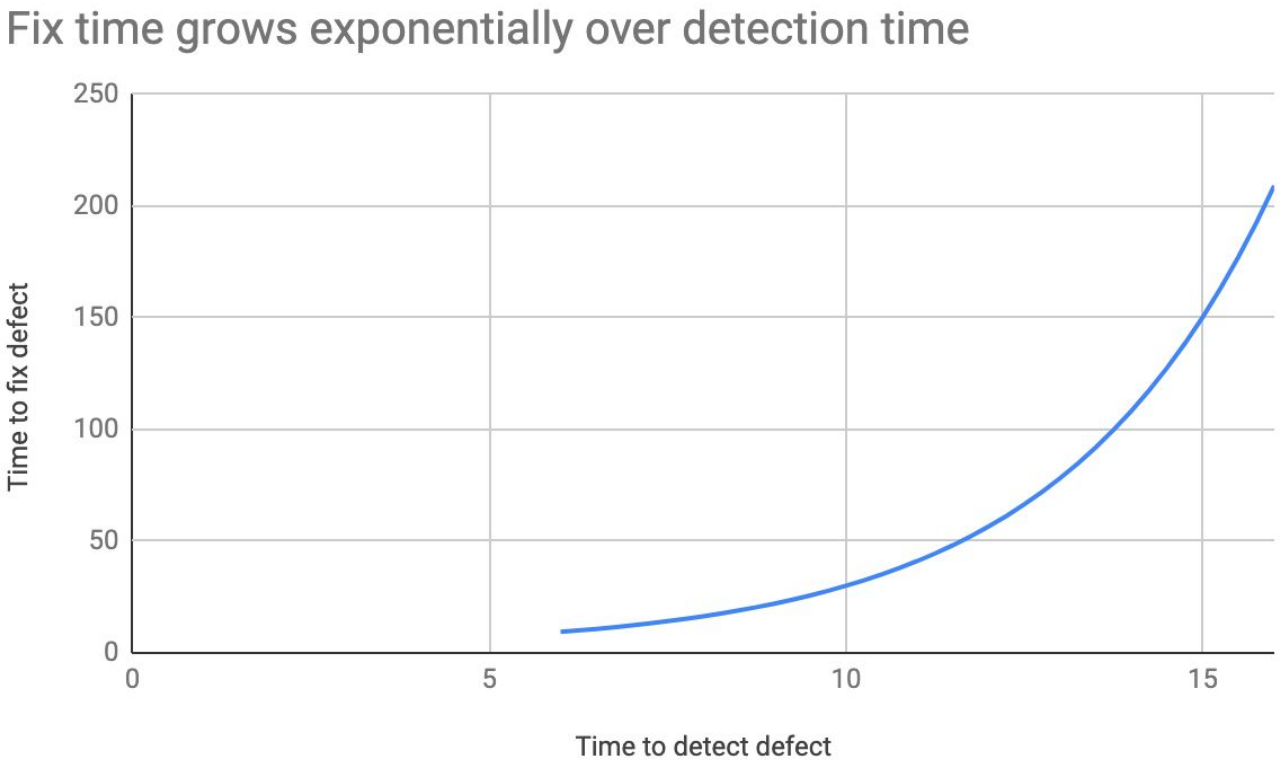
Context Switching

The longer the build the harder to debug





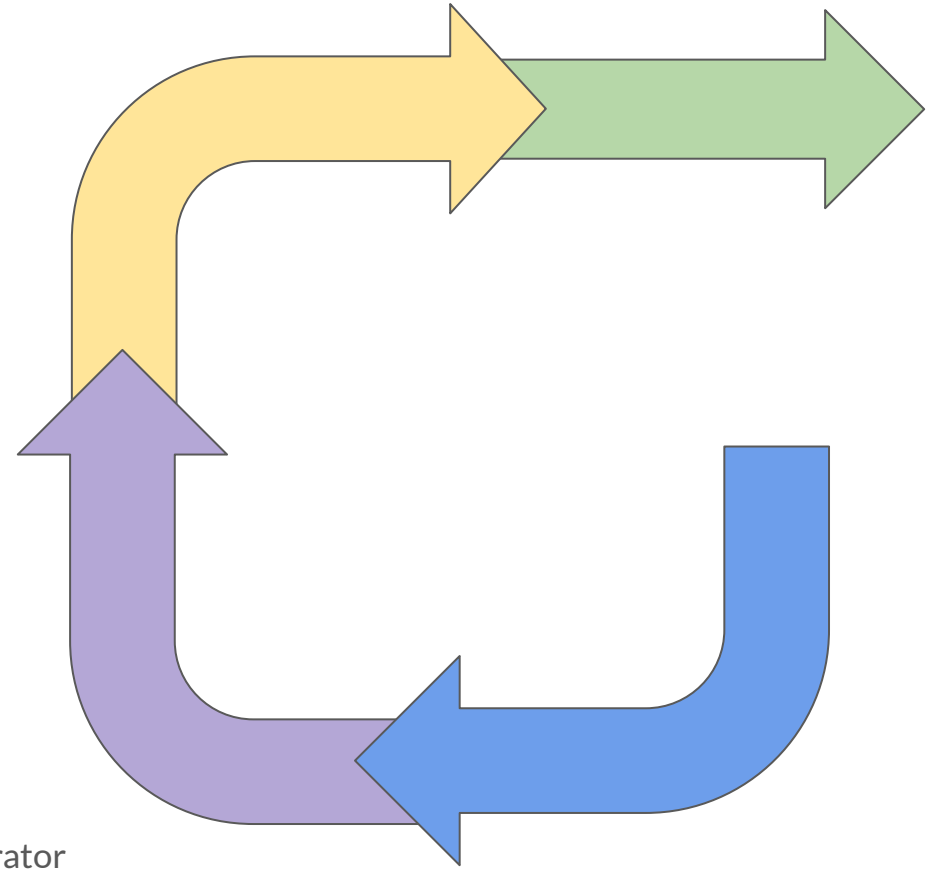
- In the case of a failure, the time fixing the failure is growing exponentially with the time it takes to detect it.



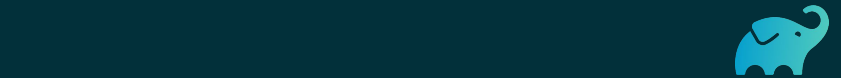
- Because of growing build times, test and builds are pushed to a later point in the life cycle.
- The exponential costs for debugging is increased by that.
- It also increases the change set size as it becomes inconvenient to get feedback.

Low Developer Productivity is **blocking innovation**

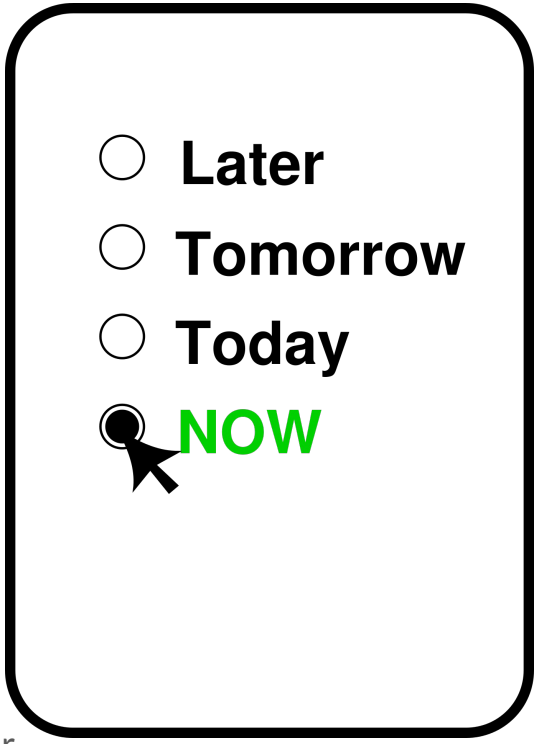
Breaking the cycle



Step 0. **Focus on Developer Productivity**



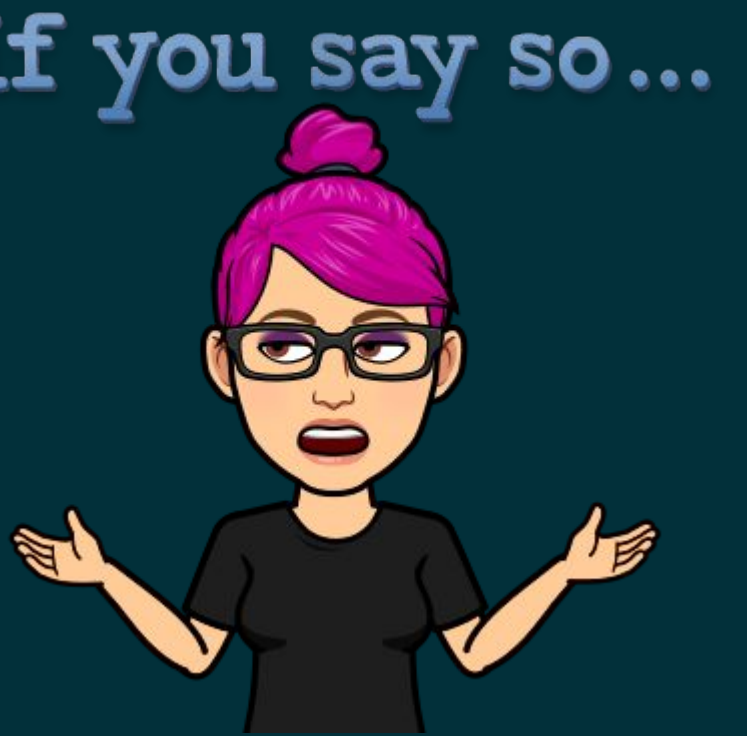
Prioritization without data



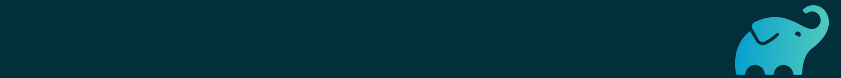
- Troubleshooting sessions begin with a game of 20 questions
- Person asking for help often doesn't know what context is important
- Helpers can burn out on helping
- Root cause analysis often impossible without the helper reproducing the problem
- Impossible to determine the impact of the issue

Results in anecdote-driven prioritization

- Non-verification failure masks as verification failure (flakey test)
- Verification failure masks as non-verification failure (snapshot dependency issue)
- Non-verification failure might be caused by bug in a plugin or user mis-configuration
- Many issues are flakey and hard to reproduce

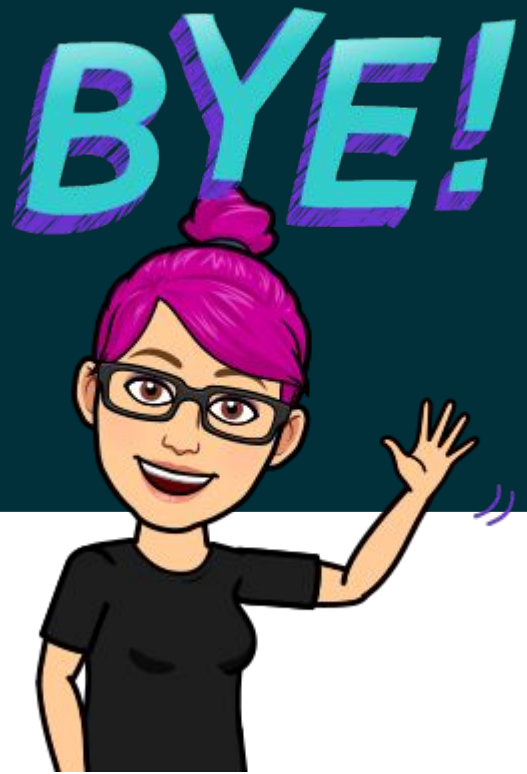


Reliability



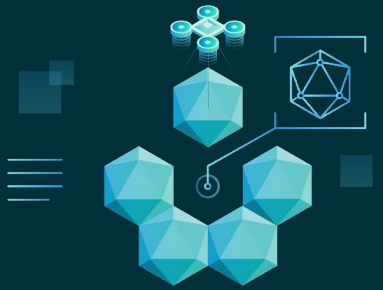
Flaky builds and tests are **maddening**

Frustrated Developers Leave

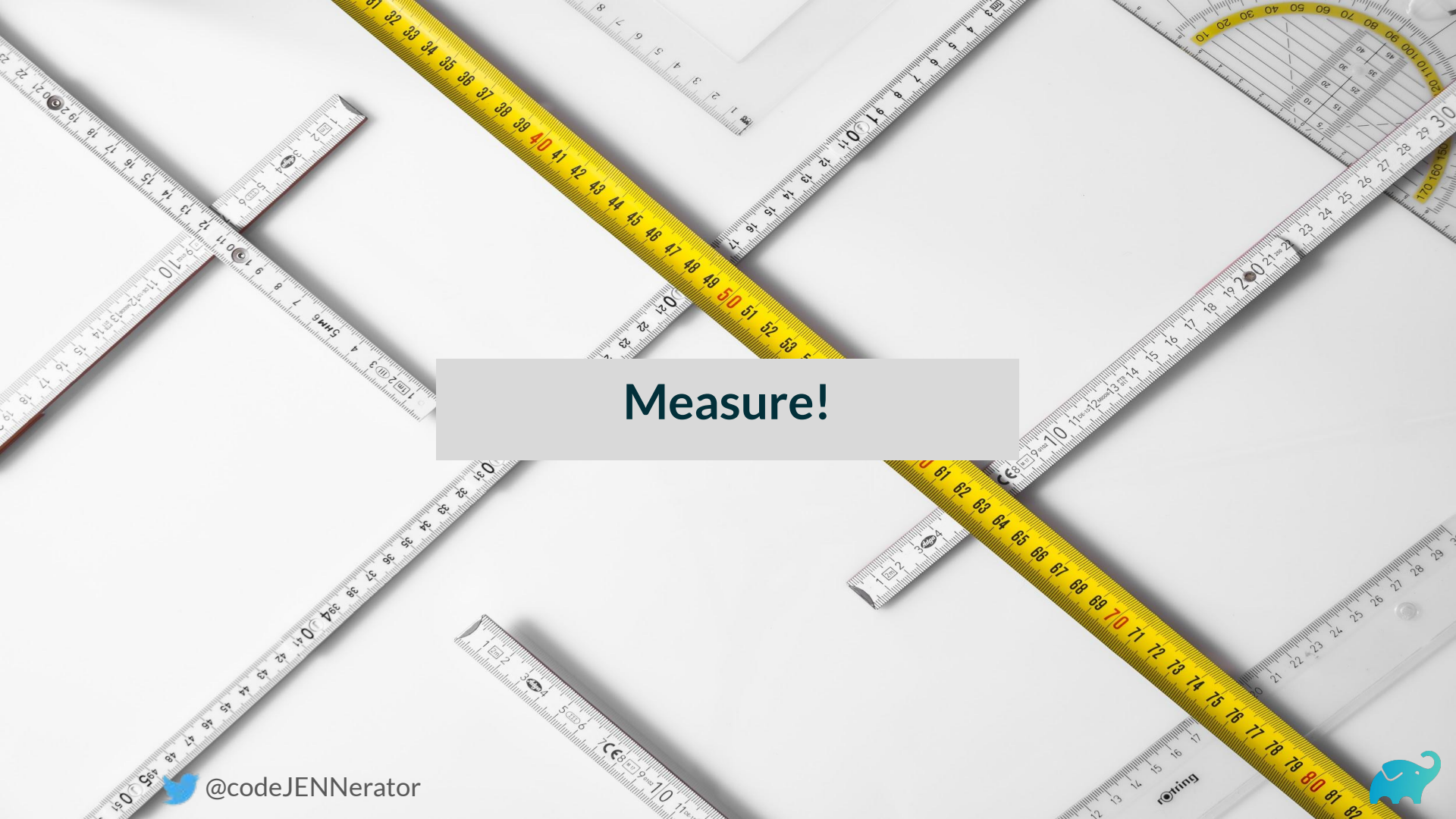


Most developers want to work in an environment that enables them to work at their full potential.

Organizations that can not provide such an environment will lose talent.



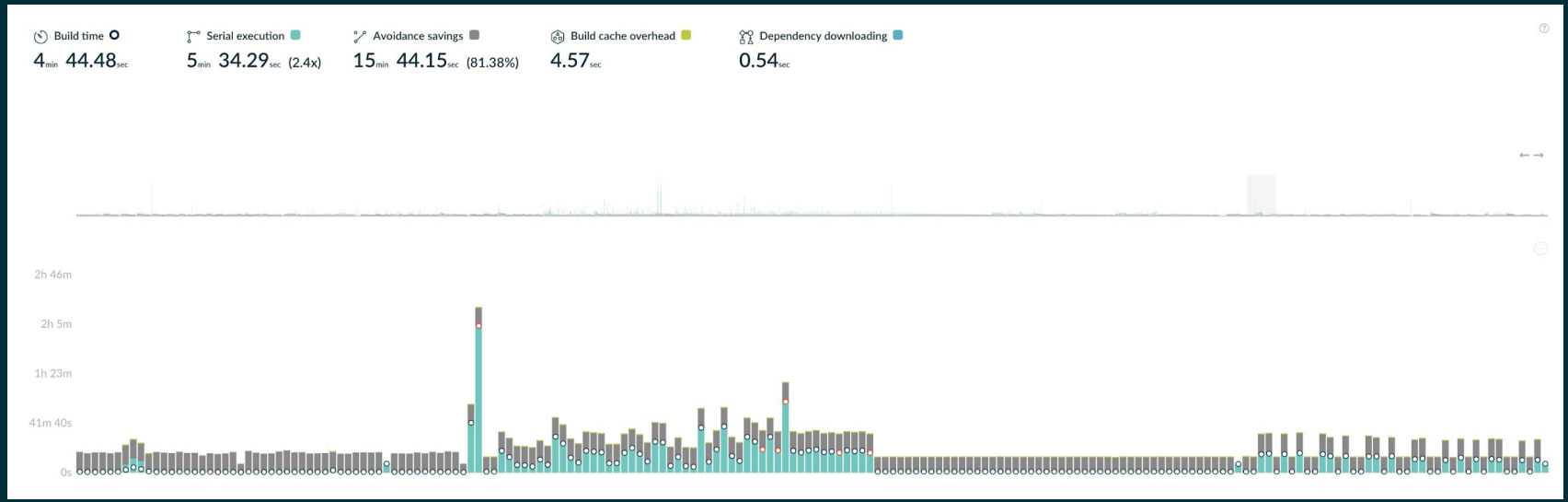
Achieving Build Happiness


Measure!

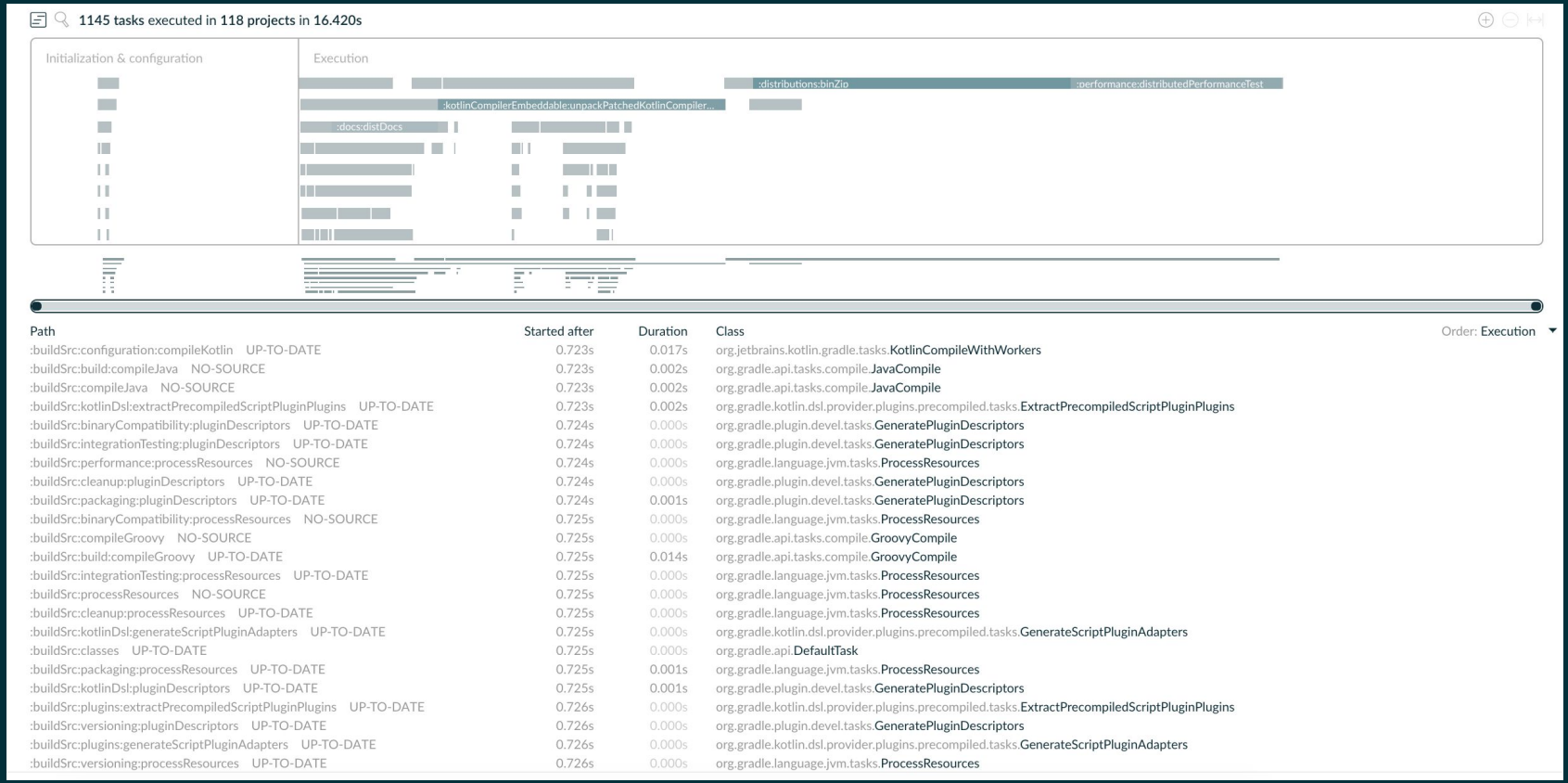
Faster builds improve the creative flow

	Team 1	Team 2
No. of Devs	11	6
Build Time	4 mins	1 mins
No. of local builds	850	1010

- The faster the feedback is, the more often devs ask for feedback
- The more often they ask for feedback, the more fine grained they can work.



@codeJENNerator

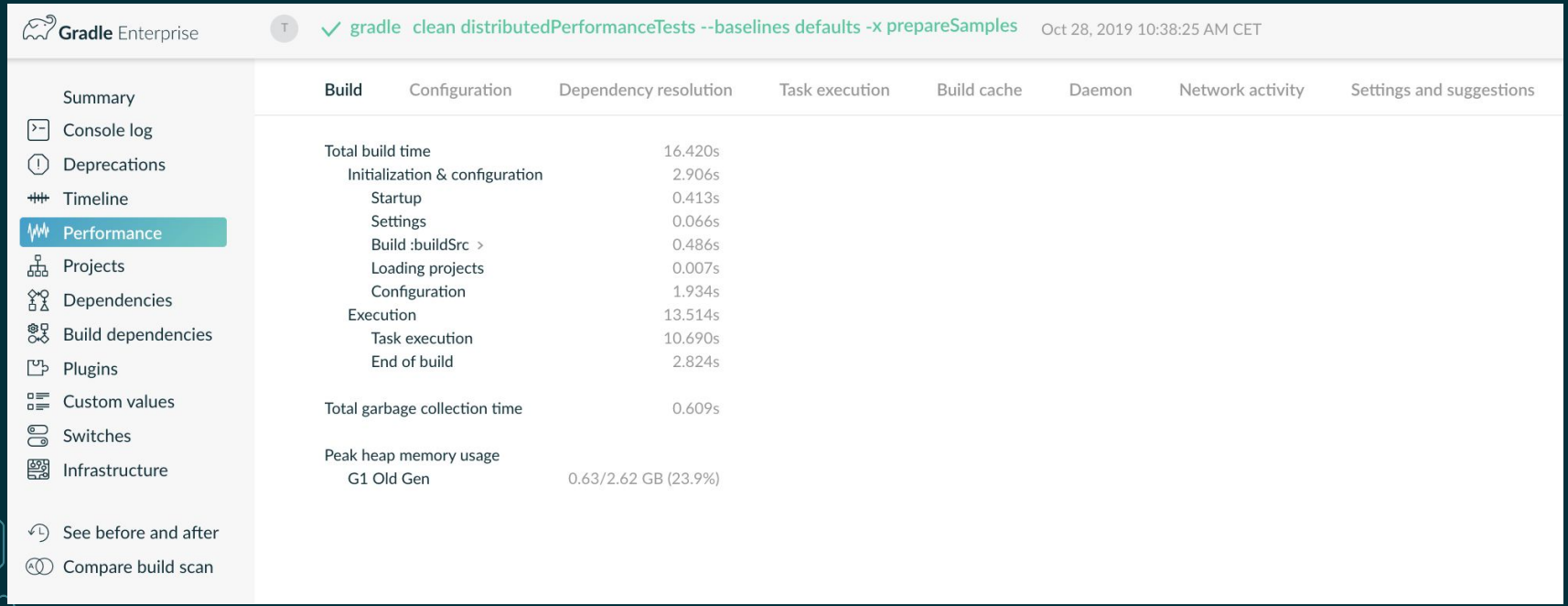


Path	Started after	Duration	Class
:buildSrc:configuration:compileKotlin UP-TO-DATE	0.723s	0.017s	org.jetbrains.kotlin.gradle.tasks.KotlinCompileWithWorkers
:buildSrc:build:compileJava NO-SOURCE	0.723s	0.002s	org.gradle.api.tasks.compile.JavaCompile
:buildSrc:compileJava NO-SOURCE	0.723s	0.002s	org.gradle.api.tasks.compile.JavaCompile
:buildSrc:kotlinDsl:extractPrecompiledScriptPluginPlugins UP-TO-DATE	0.723s	0.002s	org.gradle.kotlin.dsl.provider.plugins.precompiled.tasks.ExtractPrecompiledScriptPluginPlugins
:buildSrc:binaryCompatibility:pluginDescriptors UP-TO-DATE	0.724s	0.000s	org.gradle.plugin.devel.tasks.GeneratePluginDescriptors
:buildSrc:integrationTesting:pluginDescriptors UP-TO-DATE	0.724s	0.000s	org.gradle.plugin.devel.tasks.GeneratePluginDescriptors
:buildSrc:performance:processResources NO-SOURCE	0.724s	0.000s	org.gradle.language.jvm.tasks.ProcessResources
:buildSrc:cleanup:pluginDescriptors UP-TO-DATE	0.724s	0.000s	org.gradle.plugin.devel.tasks.GeneratePluginDescriptors
:buildSrc:packaging:pluginDescriptors UP-TO-DATE	0.724s	0.001s	org.gradle.plugin.devel.tasks.GeneratePluginDescriptors
:buildSrc:binaryCompatibility:processResources NO-SOURCE	0.725s	0.000s	org.gradle.language.jvm.tasks.ProcessResources
:buildSrc:compileGroovy NO-SOURCE	0.725s	0.000s	org.gradle.api.tasks.compile.GroovyCompile
:buildSrc:build:compileGroovy UP-TO-DATE	0.725s	0.014s	org.gradle.api.tasks.compile.GroovyCompile
:buildSrc:integrationTesting:processResources UP-TO-DATE	0.725s	0.000s	org.gradle.language.jvm.tasks.ProcessResources
:buildSrc:processResources NO-SOURCE	0.725s	0.000s	org.gradle.language.jvm.tasks.ProcessResources
:buildSrc:cleanup:processResources UP-TO-DATE	0.725s	0.000s	org.gradle.language.jvm.tasks.ProcessResources
:buildSrc:kotlinDsl:generateScriptPluginAdapters UP-TO-DATE	0.725s	0.000s	org.gradle.kotlin.dsl.provider.plugins.precompiled.tasks.GenerateScriptPluginAdapters
:buildSrc:classes UP-TO-DATE	0.725s	0.000s	org.gradle.api.DefaultTask
:buildSrc:packaging:processResources UP-TO-DATE	0.725s	0.001s	org.gradle.language.jvm.tasks.ProcessResources
:buildSrc:kotlinDsl:pluginDescriptors UP-TO-DATE	0.725s	0.001s	org.gradle.plugin.devel.tasks.GeneratePluginDescriptors
:buildSrc:plugins:extractPrecompiledScriptPluginPlugins UP-TO-DATE	0.726s	0.000s	org.gradle.kotlin.dsl.provider.plugins.precompiled.tasks.ExtractPrecompiledScriptPluginPlugins
:buildSrc:versioning:pluginDescriptors UP-TO-DATE	0.726s	0.000s	org.gradle.plugin.devel.tasks.GeneratePluginDescriptors
:buildSrc:plugins:generateScriptPluginAdapters UP-TO-DATE	0.726s	0.000s	org.gradle.kotlin.dsl.provider.plugins.precompiled.tasks.GenerateScriptPluginAdapters
:buildSrc:versioning:processResources UP-TO-DATE	0.726s	0.000s	org.gradle.language.jvm.tasks.ProcessResources

Order: Execution

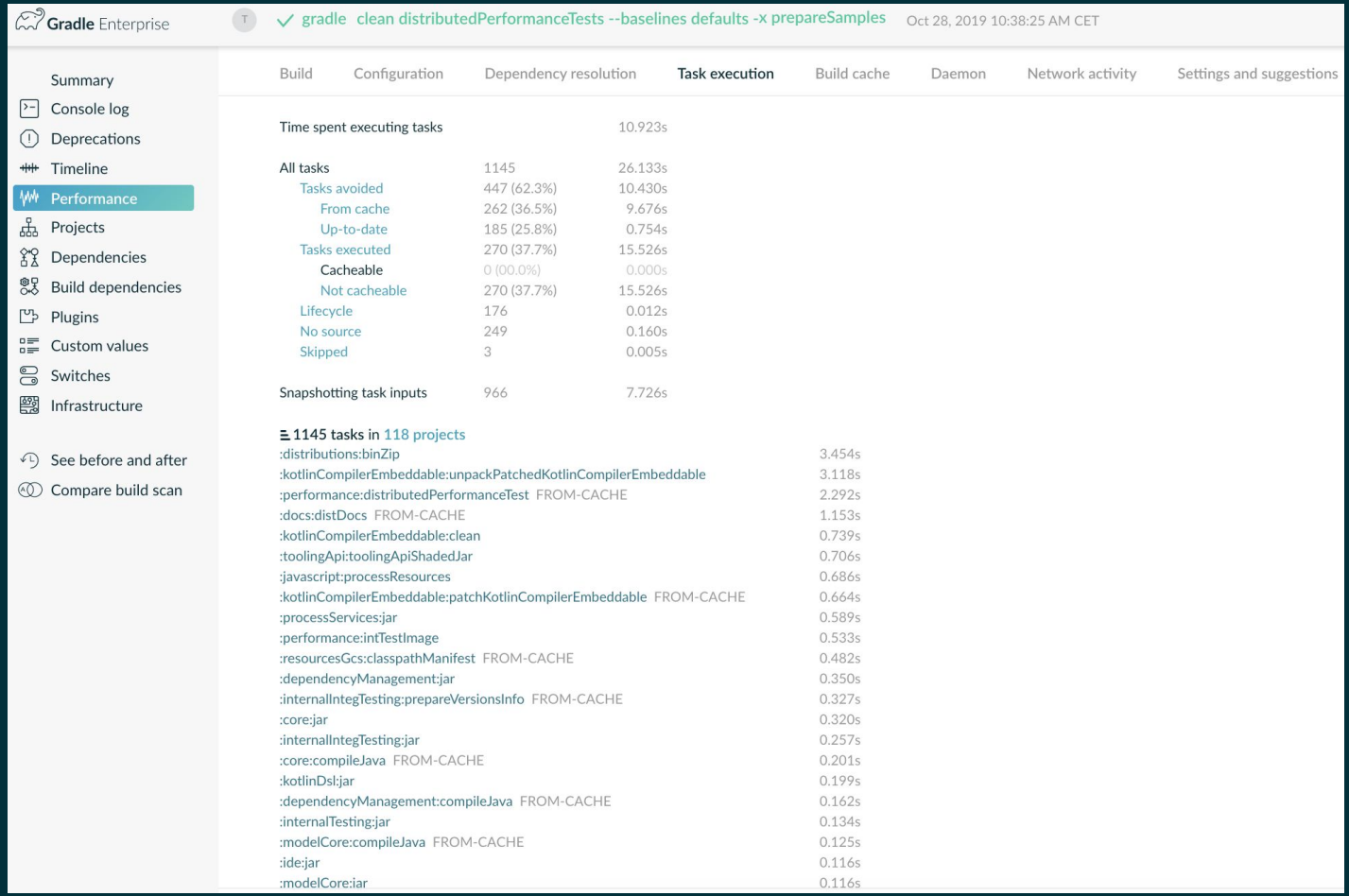
@codeJENNerator

https://e.grdev.net/s/vpui4db7vx6xo/timeline



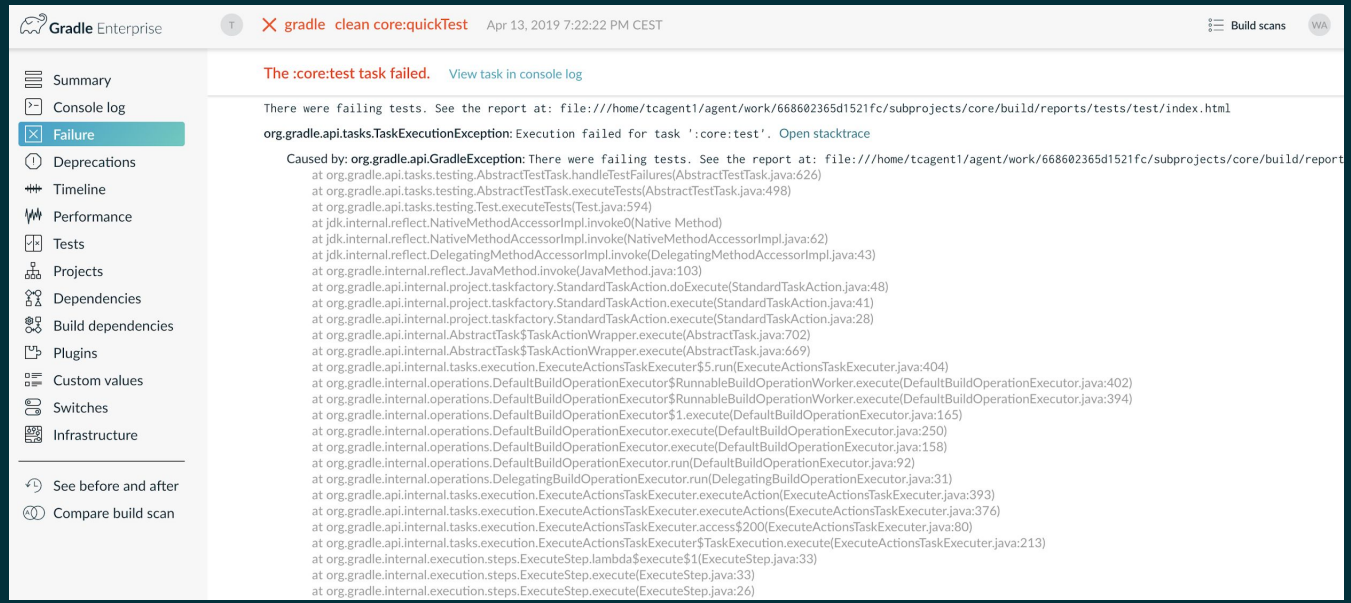
https://e.grdev.net/s/vpui4db7vx6xo/performance

@codeJENNerator


Gradle Enterprise
gradle clean distributedPerformanceTests --baselines defaults -x prepareSamples
Oct 28, 2019 10:38:25 AM CET
Summary
Console log
Deprecations
Timeline
Performance
Projects
Dependencies
Build dependencies
Plugins
Custom values
Switches
Infrastructure
See before and after
Compare build scan
Build
Configuration
Dependency resolution
Task execution
Build cache
Daemon
Network activity
Settings and suggestions
Time spent executing tasks 10.923s
All tasks 1145 26.133s
Tasks avoided 447 (62.3%) 10.430s
From cache 262 (36.5%) 9.676s
Up-to-date 185 (25.8%) 0.754s
Tasks executed 270 (37.7%) 15.526s
Cacheable 0 (00.0%) 0.000s
Not cacheable 270 (37.7%) 15.526s
Lifecycle 176 0.012s
No source 249 0.160s
Skipped 3 0.005s
Snapshotting task inputs 966 7.726s
1145 tasks in 118 projects
:distributions:binZip 3.454s
:kotlinCompilerEmbeddable:unpackPatchedKotlinCompilerEmbeddable 3.118s
:performance:distributedPerformanceTest FROM-CACHE 2.292s
:docs:distDocs FROM-CACHE 1.153s
:kotlinCompilerEmbeddable:clean 0.739s
:toolingApi:toolingApiShadedJar 0.706s
:javascript:processResources 0.686s
:kotlinCompilerEmbeddable:patchKotlinCompilerEmbeddable FROM-CACHE 0.664s
:processServices:jar 0.589s
:performance:intTestImage 0.533s
:resourcesGcs:classpathManifest FROM-CACHE 0.482s
:dependencyManagement:jar 0.350s
:internalIntegTesting:prepareVersionsInfo FROM-CACHE 0.327s
:core:jar 0.320s
:internalIntegTesting:jar 0.257s
:core:compileJava FROM-CACHE 0.201s
:kotlinDsl:jar 0.199s
:dependencyManagement:compileJava FROM-CACHE 0.162s
:internalTesting:jar 0.134s
:modelCore:compileJava FROM-CACHE 0.125s
:ide:jar 0.116s
:modelCore:jar 0.116s

Capture data from every build run on **local AND CI**

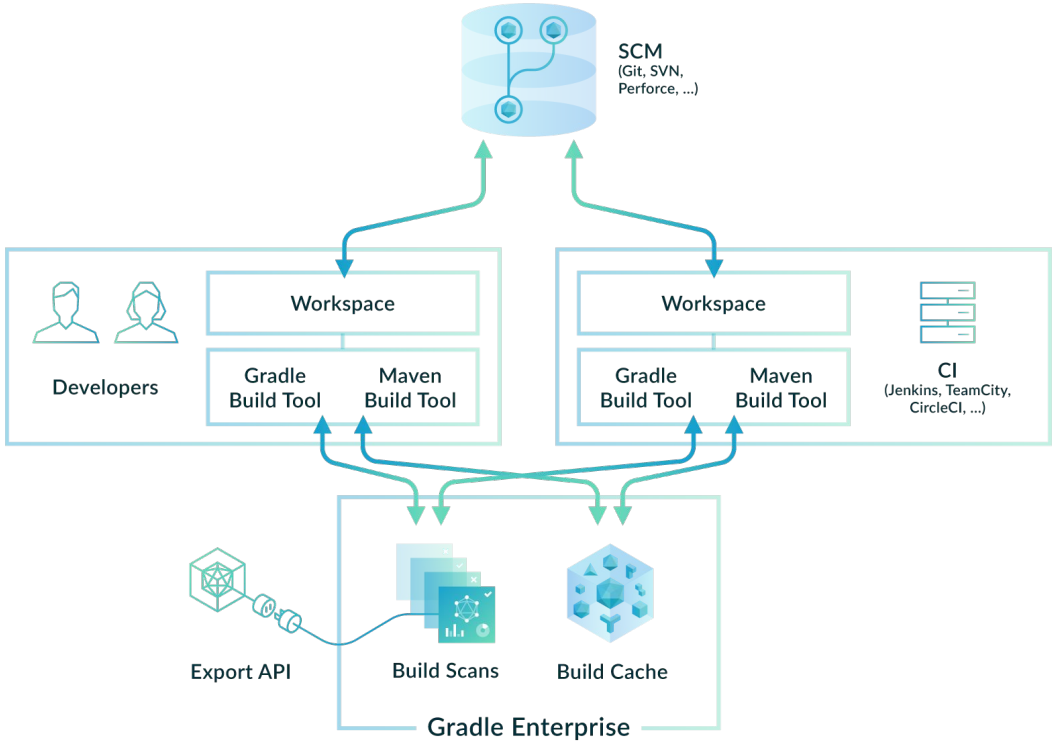
Comprehensive data allows for root cause analysis without reproducing locally



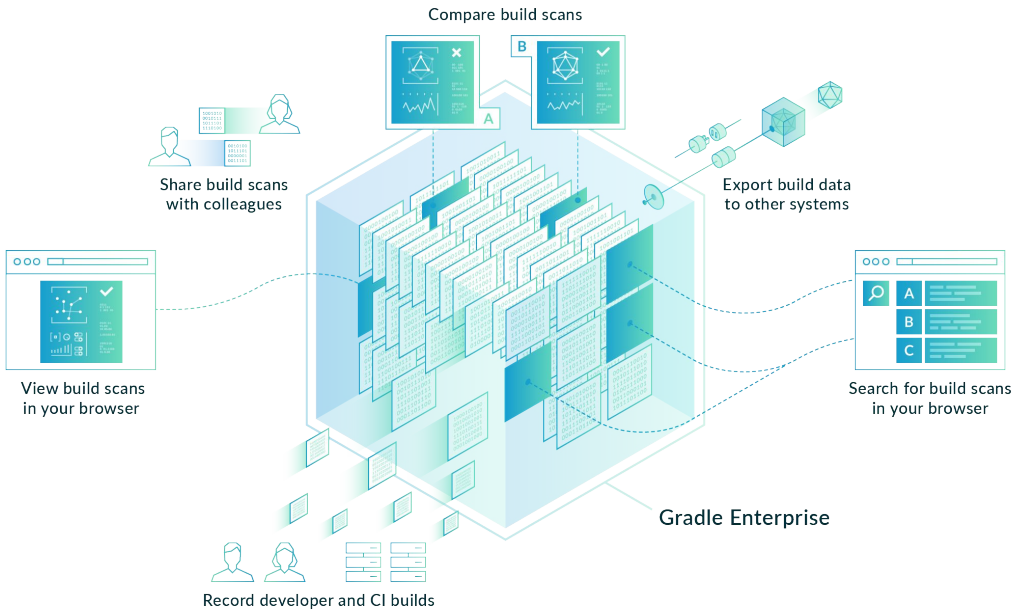

Business Chart - Visual
Space Usage (750 Mb)
Interactive User
1,505
18,321
Realtime Dashboard
Marketing Chart
BUSINESS STRATEGY
BUSINESS STRATEGY
10:24
Contract

Analyze!

What is Gradle Enterprise?



Gradle Enterprise is a data platform

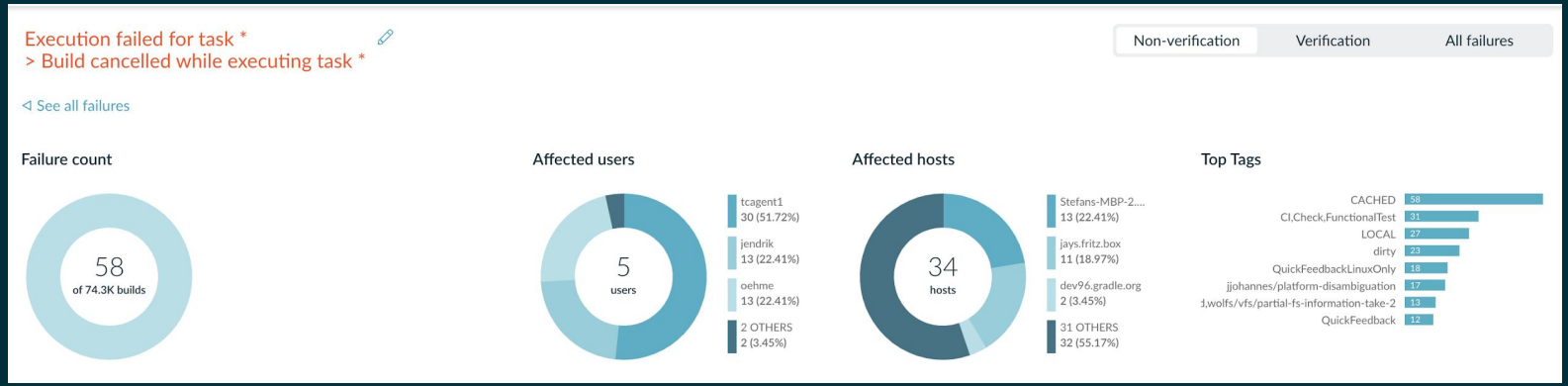


- **Collect team data** - all the data about every build across the team creates unique dataset and insights.
- **Build Performance Management** - only with representative, actionable data will builds get and stay fast and reliable.
- **Debugging acceleration** - only with comprehensive, deep data is it possible to quickly discover the root cause for build failures.

Failure Types

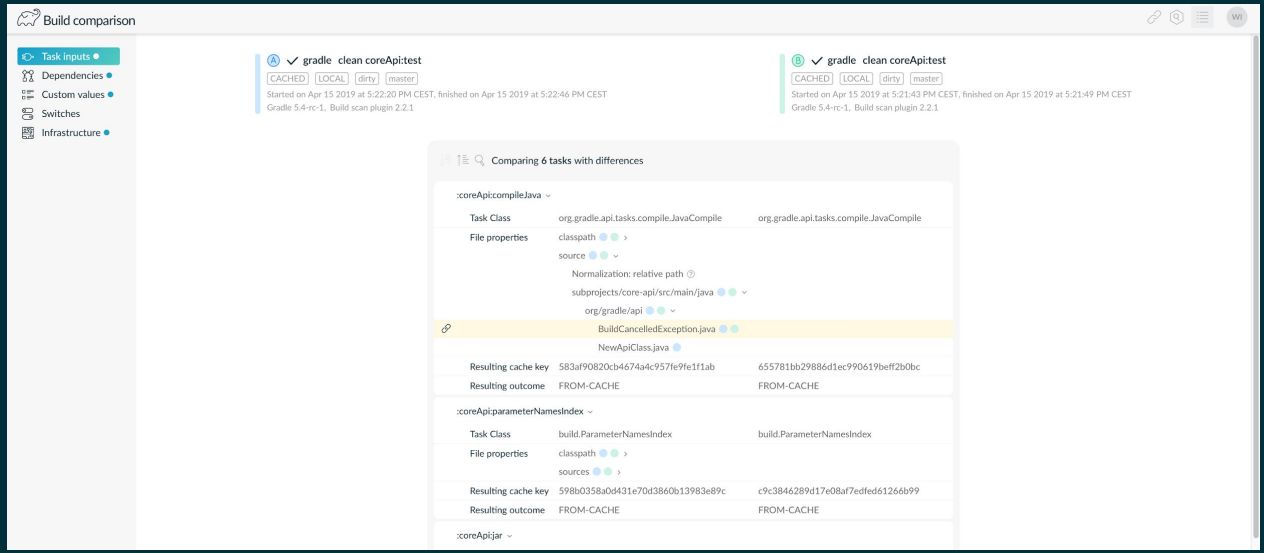
- Verification Failures
  - Syntax error detected by compilation
  - Code style violation detected by checkstyle
  - Misbehavior in the code detected by a JUnit test
- Non-Verification Failures
  - Flakey Test
  - Binary repository down
  - Out of memory exception while running the build
- Slow Builds

Impact analysis

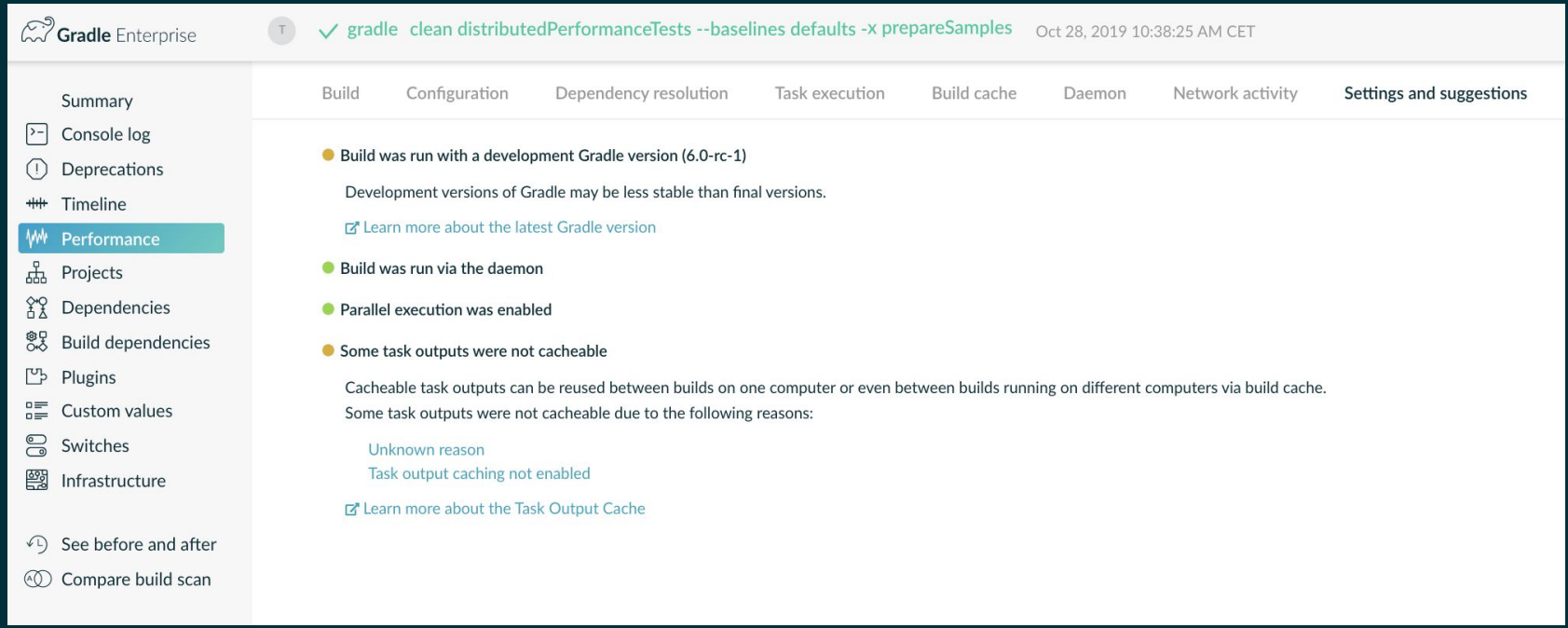


@codeJENNerator

Root Cause Analysis



Optimize!



https://e.grdev.net/s/vpuj4db7vx6xo/performance/suggestions

@codeJENNerator

Build Caching

When the inputs have not changed, the outputs can be reused from a previous run.

Cacheable Task/Goal Executions

Gradle Compile/Maven Compile

- Source Files
- Compile Classpath
- Java version
- Compiler configuration
- etc...

Gradle Test/Maven Surefire

- Test Source Files
- Runtime Classpath
- Java version
- System properties
- etc...

Caching is a generic feature and applies to all tasks/goals.
For IO-bound tasks/goals caching has no benefits (e.g. clean, copy).

Caching is effective for multi-module builds

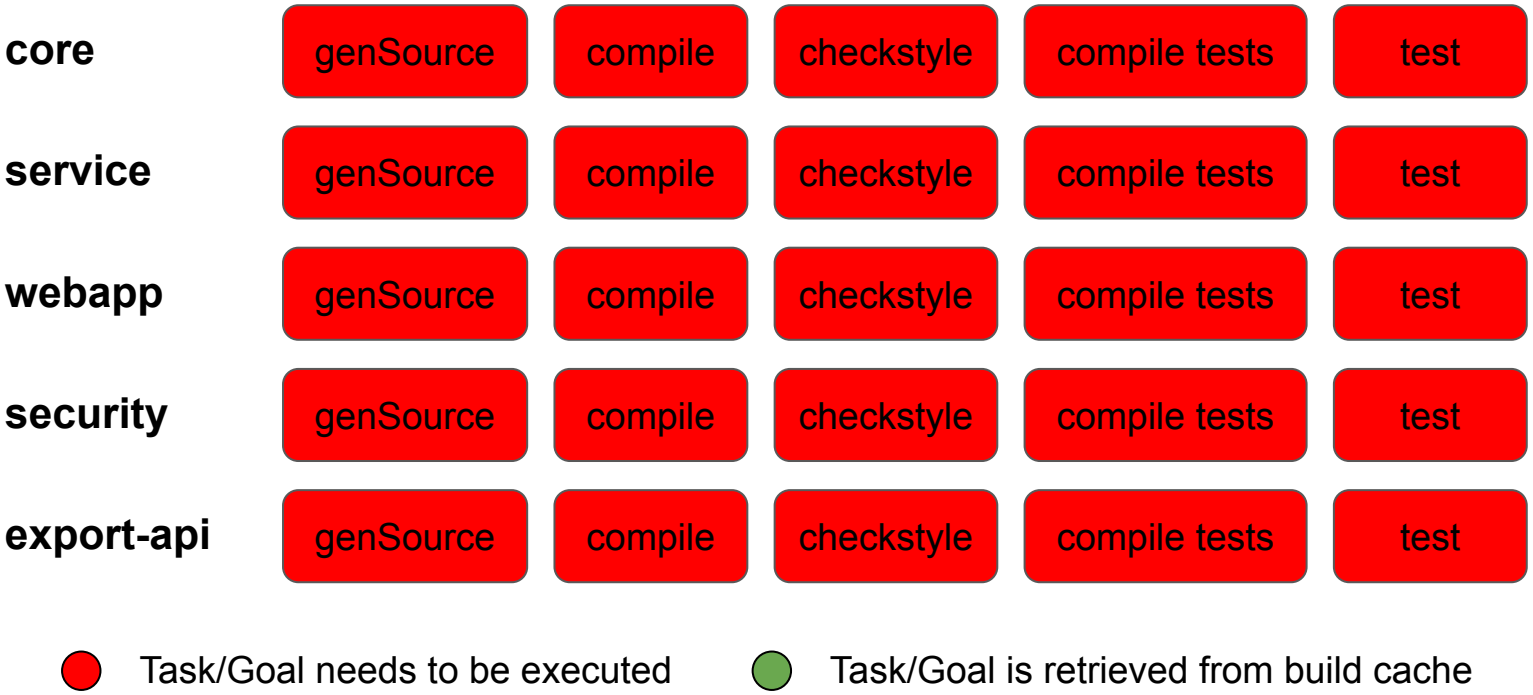
pom.xml

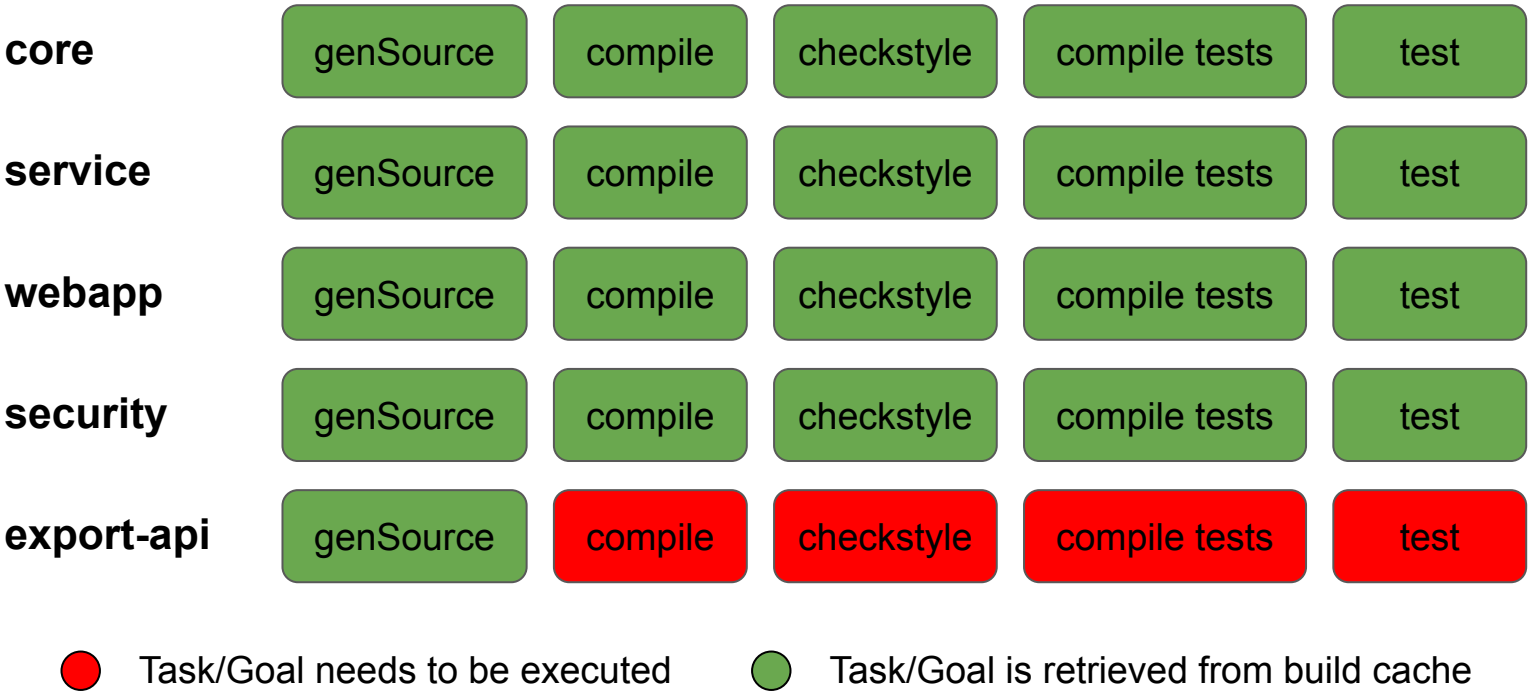
```
<modules>
  <module>core</module>
  <module>service</module>
  <module>webapp</module>
  <module>export-api</module>
  <module>security</module>
</modules>
```

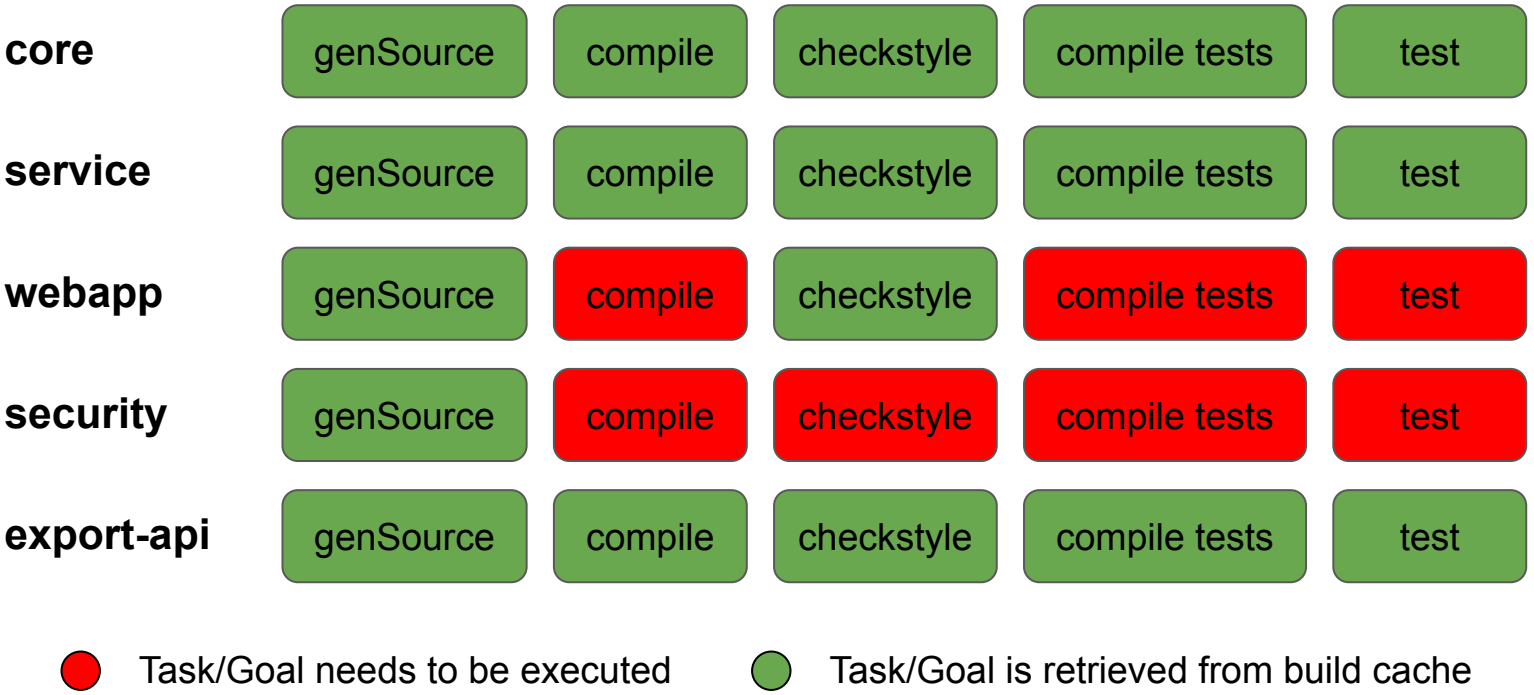
settings.gradle

```
include ‘core’
include ‘service’
Include ‘webapp’
Include ‘export-api’
Include ‘security’
```

Builds with a single module will only moderately benefit from the cache


core
genSource
compile
checkstyle
compile tests
test
service
genSource
compile
checkstyle
compile tests
test
webapp
genSource
compile
checkstyle
compile tests
test
security
genSource
compile
checkstyle
compile tests
test
export-api
genSource
compile
checkstyle
compile tests
test
Task/Goal needs to be executed
Task/Goal is retrieved from build cache

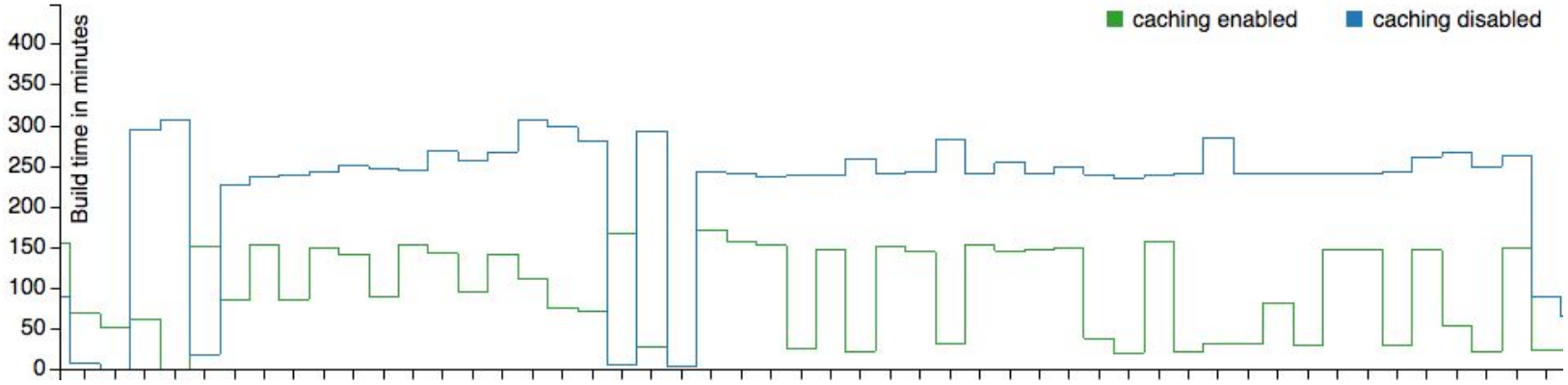

core
genSource
compile
checkstyle
compile tests
test
service
genSource
compile
checkstyle
compile tests
test
webapp
genSource
compile
checkstyle
compile tests
test
security
genSource
compile
checkstyle
compile tests
test
export-api
genSource
compile
checkstyle
compile tests
test
Task/Goal needs to be executed
Task/Goal is retrieved from build cache


core
genSource
compile
checkstyle
compile tests
test
service
genSource
compile
checkstyle
compile tests
test
webapp
genSource
compile
checkstyle
compile tests
test
security
genSource
compile
checkstyle
compile tests
test
export-api
genSource
compile
checkstyle
compile tests
test
Task/Goal needs to be executed
Task/Goal is retrieved from build cache

Cache effectiveness

- Even with only a few modules a cache significantly reduces build and test times
- For larger multi-module builds often 50% of modules are leaf modules
  - Build times is reduced by approximately 1/n with n being the number of modules
- Checking the inputs and downloading & unpacking items of the cache introduces overhead.
- Overhead is often very small compared to benefits
- Overhead should be measured and monitored too

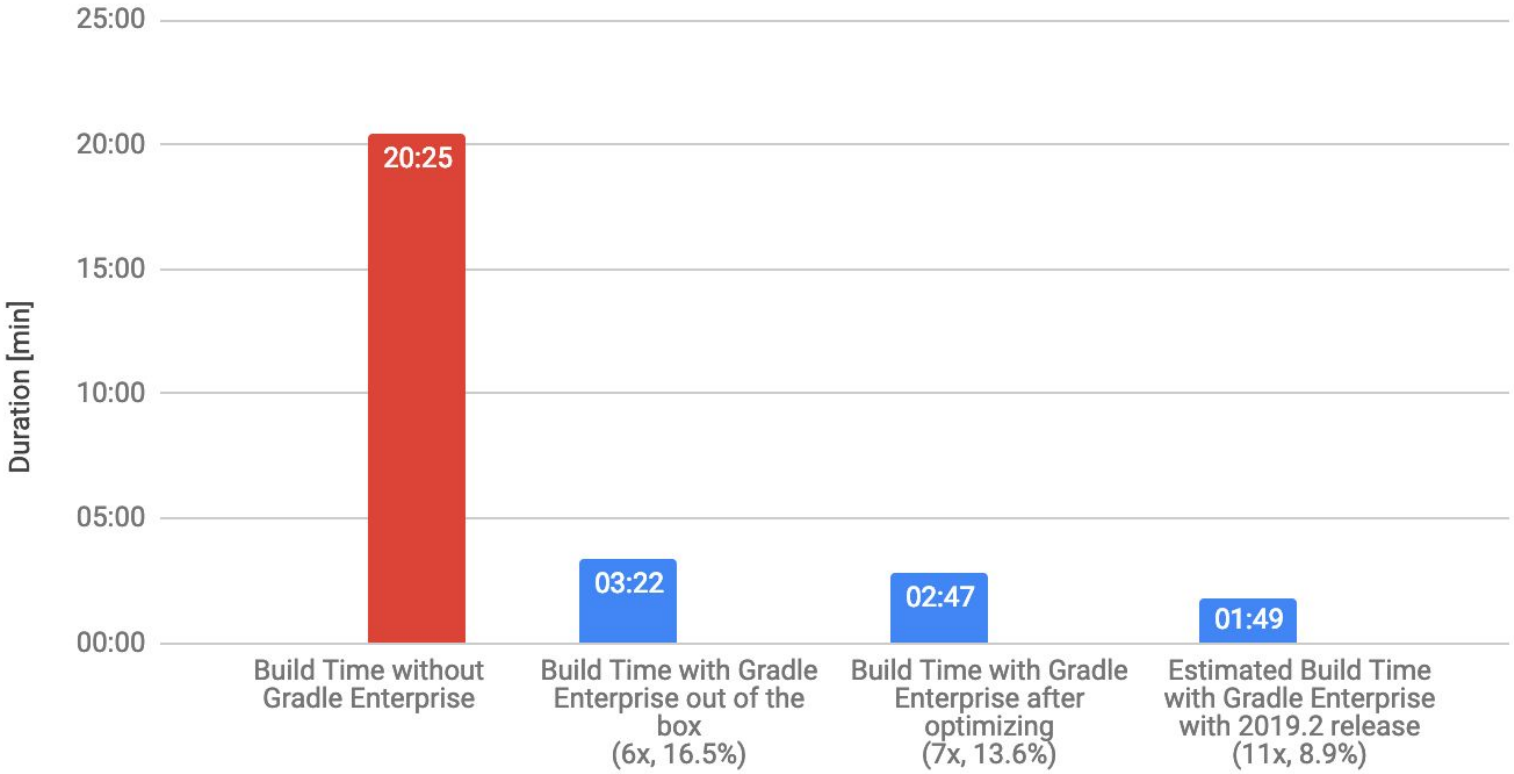
Gradle CI Builds



Build times are **>80%** faster for Gradle Core
Dramatically better caching results due to build scans.

Spring Boot build time for compile and unit tests (fully cached)

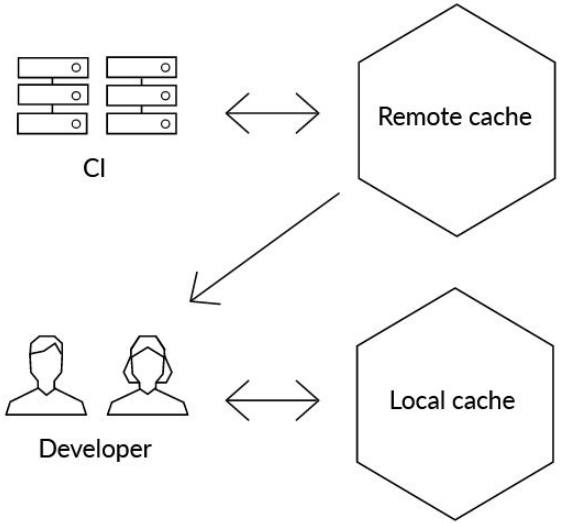
https://spring.io/projects/spring-boot



Local Build Cache

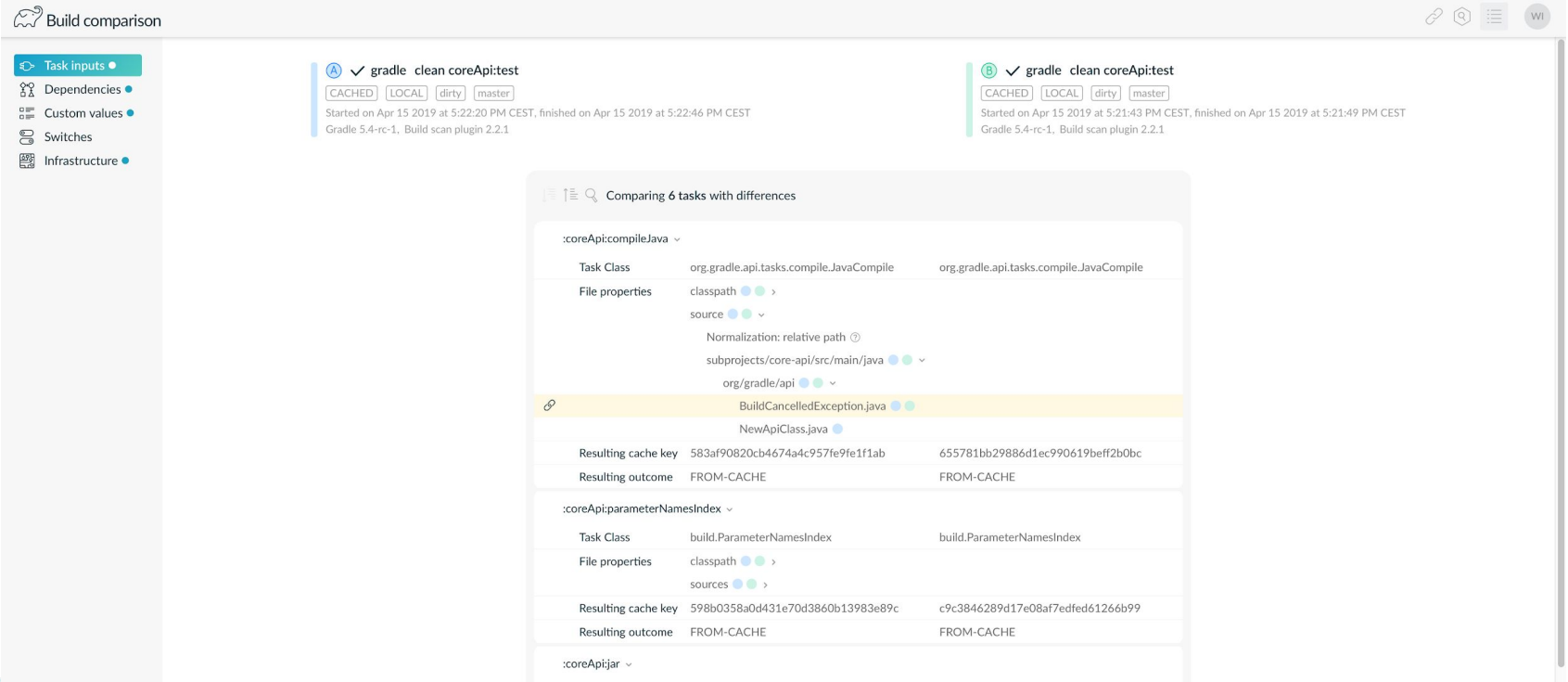
- Uses a cache directory on your local machine
- Speeds up development for single developer or build agent
- Reuses build results when switching branches locally

Remote Build Cache



- Shared among different machines
- Speeds up development for the whole team
- Reuses build results among CI agents/jobs and individual developers

Debugging cache misses



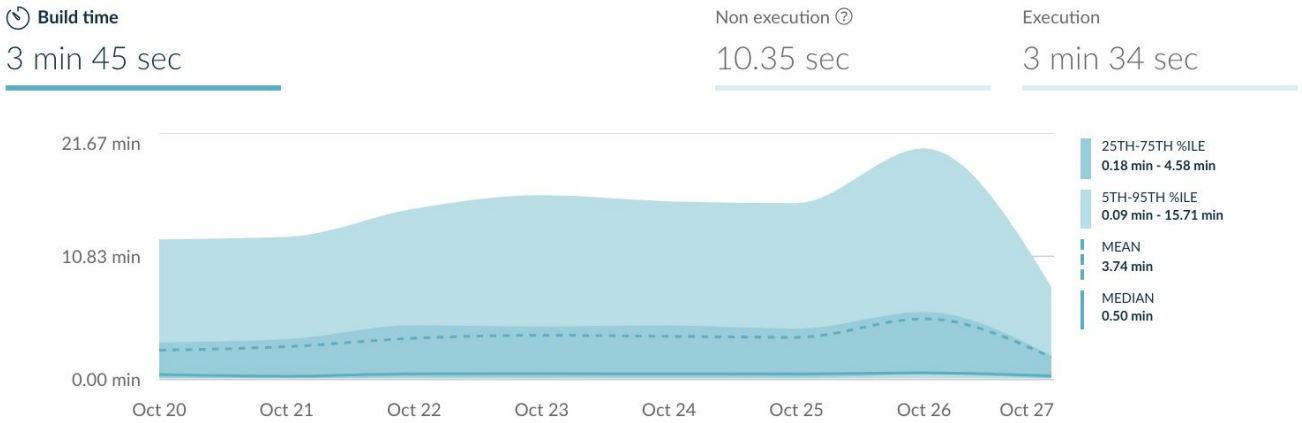
LinkedIn: [Productivity at scale: How we improved build time with Gradle build cache]
SoundCloud: [Solving Remote Build Cache Misses by Annoying Your Colleagues]

@codeJENNerator


YAY!

Iterate!

Performance regressions are easily introduced

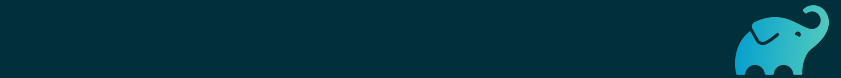


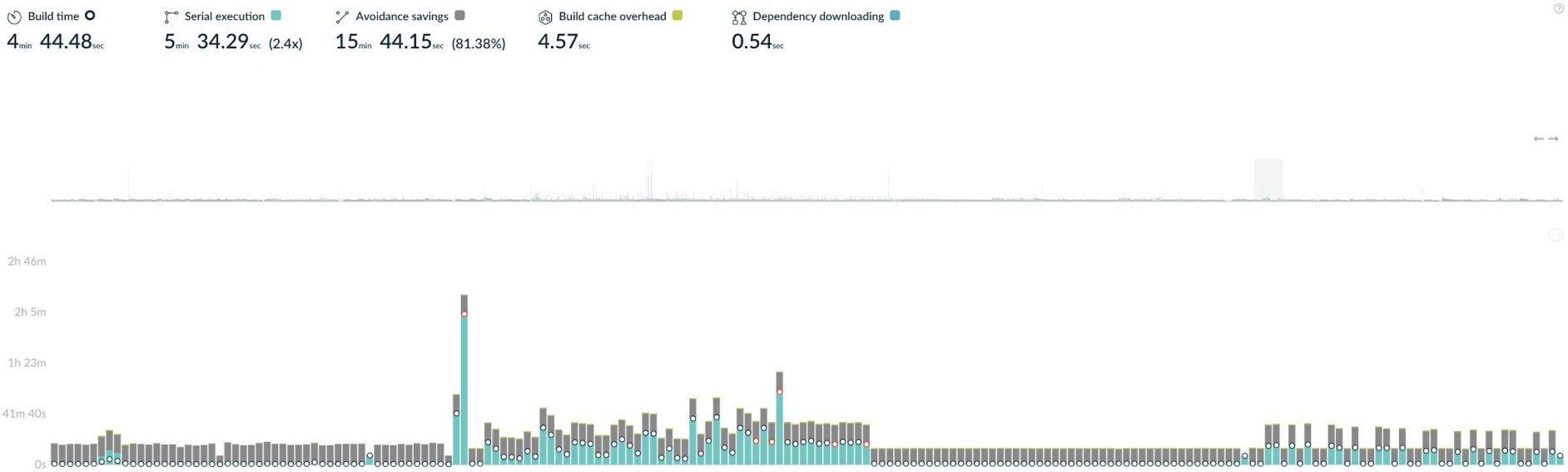
- Infrastructure changes
- New annotation processors or versions of annotation processors
- Build logic configurations settings
- Code refactoring

What happens today with most regressions

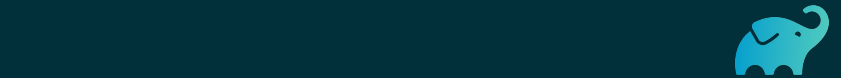
- Unnoticed
- Noticed but unreported
- Reported but not addressed
  - Root cause is hard to detect (especially with flakey issues)
  - Overall impact and priority can not be determined
- Escalated after they have caused a lot of pain
  - Problem gets fixed after it has wasted a lot of time and caused a lot of frustration amongst developers.
- **Result**: The average build time is much higher than necessary and continuously increasing.

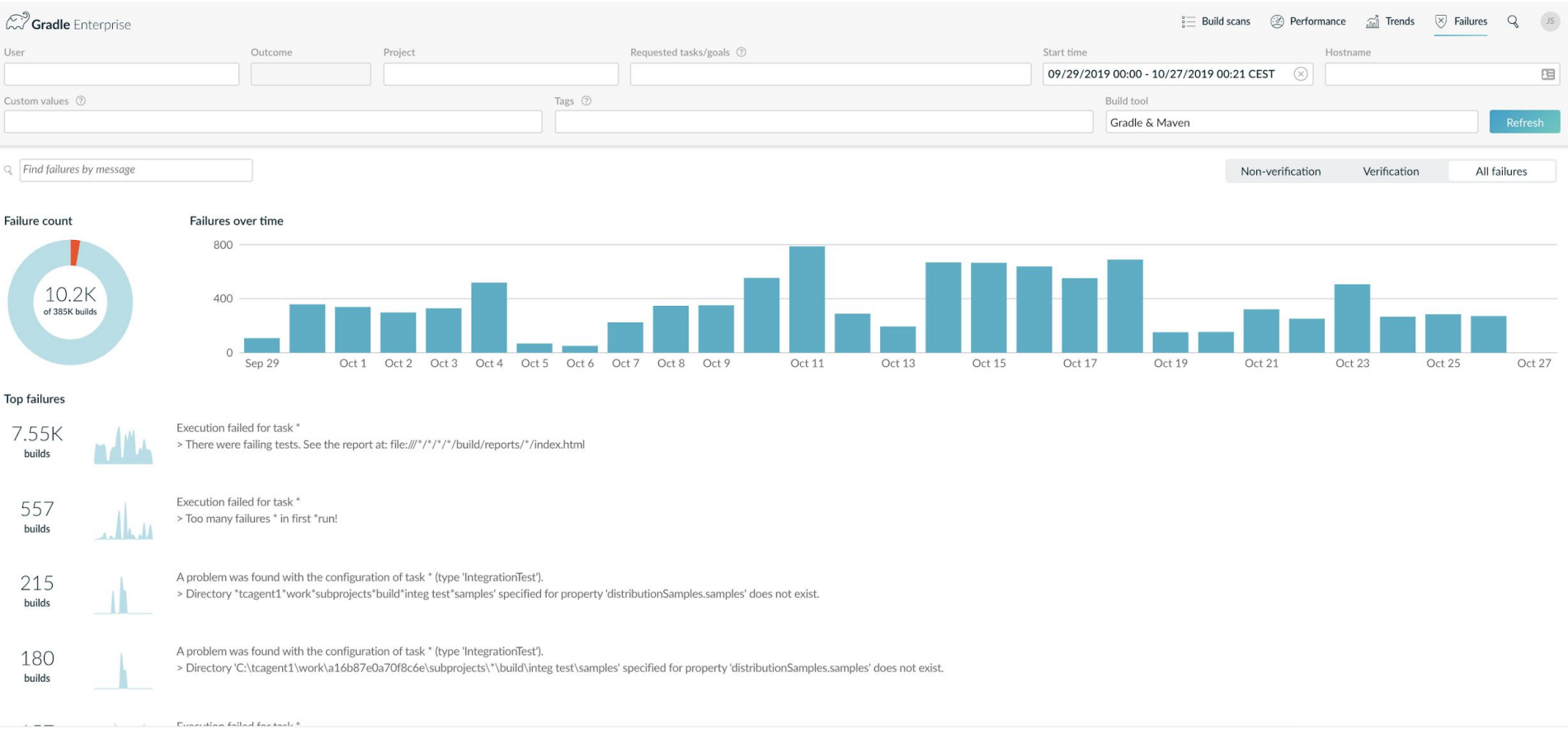
Performance Analytics



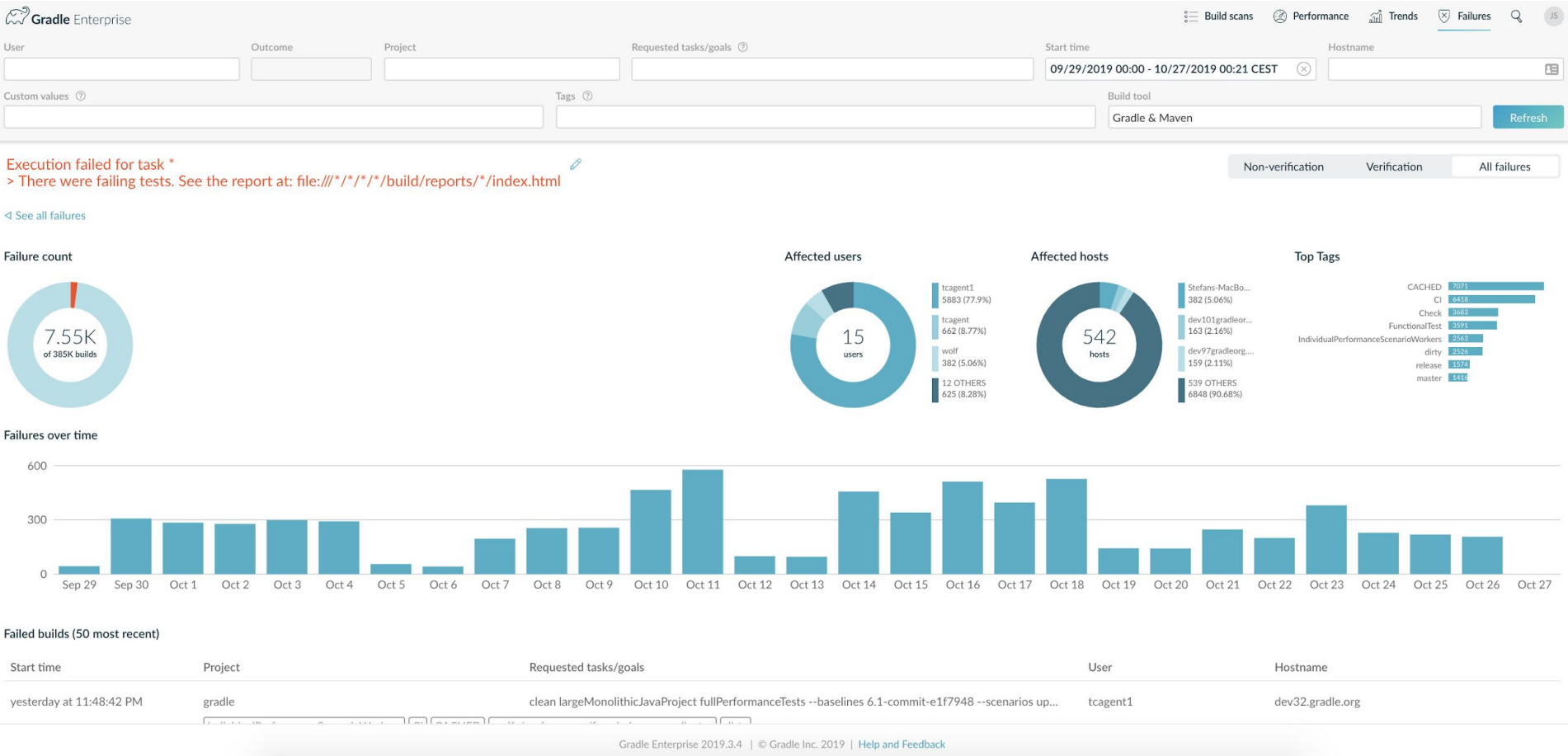

Build time
4min 44.48sec
Serial execution
5min 34.29sec (2.4x)
Avoidance savings
15min 44.15sec (81.38%)
Build cache overhead
4.57sec
Dependency downloading
0.54sec
2h 46m
2h 5m
1h 23m
41m 40s
0s

Failures Dashboard





@codeJENNerator



@codeJENNerator

Flaky Tests

Flaky tests are a problem for everyone

We’re currently working on a solution

Plan to release this in Q4/2019

Interested in your ideas & thoughts

Conclusion

- Don’t suffer in silence.
- The Path to Build Happiness is through Developer Productivity.
- Measure, Analyze, Optimize, and Iterate to achieve and maintain build happiness.

Resources

- Early Access Book: https://gradle.com/developer-productivity-engineering
- Try out build scans for Maven and Gradle for free: https://scans.gradle.com
- Gradle Enterprise docs and tutorials: https://docs.gradle.com

Thank you!

@codeJENNerator