DevOps for Defense

March 2020

Developer Productivity Engineering

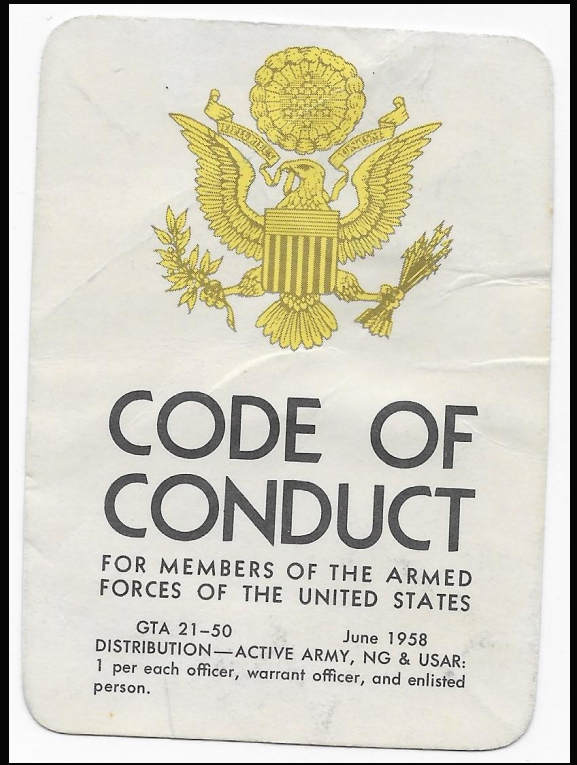
Gradle

Hans Dockter
Gradle Founder & CEO

https://devopsfordefense.org
https://www.meetup.com/DevOps-for-Defense/
https://github.com/jondavid-black/DevOpsForDefense
devopsfordefense@gmail.com
https://twitter.com/devops4defense

DevOps for Defense Meetup: Code of Conduct

- UNCLASSIFIED ONLY!!!!
- Treat each other with respect and professionalism.
- Do not talk about private, sensitive, or proprietary work.
- Do talk about your experiences, needs, desires to improve work in our domain.
- Do share your thoughts.
- Do learn from others.
- Do respect & tip your bartenders!



Local Event Announcements

https://devopsfordefense.org

DevOps for Defense Meetup

Presents

GitLab

GitLab is the first single application for the entire DevOps lifecycle. Only GitLab enables Concurrent DevOps, unlocking organizations from the constraints of today's toolchain.

GitLab provides unmatched visibility, radical new levels of efficiency and comprehensive governance to significantly compress the time between planning a change and monitoring its effect. This makes the software lifecycle 200% faster, radically improving the speed of business.

Thursday, April 2nd, 2020 at 6:00pm

Rocket Republic Brewing Co
289 Production Ave, Madison, AL

Join us on the first Thursday of every month!

Meetup Sponsored by:

TEKsystems
Own change


Gradle

Developer Productivity at Scale

Hans Dockter, Founder and CEO of Gradle

Gradle

Developer Productivity needs **Toolchain Effectiveness**

Potential

Developer Happiness

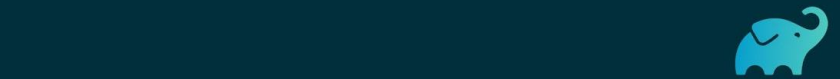
Actual Developer Unhappiness

Creative Flow

requires a dialogue


AUTO STOP
SAVE
RECORD
PLAY
REWIND
F.F.WD
STOP
EJECT
commodore

Enterprise Software Developers



Collaboration

requires a dialogue

Quality of Creative Flow

+

Collaborative Effectiveness

Team Productivity

Toolchain efficiency

enables

Creative Flow and Collaboration Effectiveness

LOC

Diversity of Tech Stack

No. of Devs

No. of Repos

No. of Deps

Let there be light

speed

Feedback cycle time affects creative flow

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

Builds that take less than 10 mins cause significant waiting time

	No of Devs	Local builds per week	Build Time	Build Time w. GE	Savings per year
Team 1	6	1010	1 mins	0.6 mins	44 days
Team 3	100	12000	9 mins	5 mins	5200 days

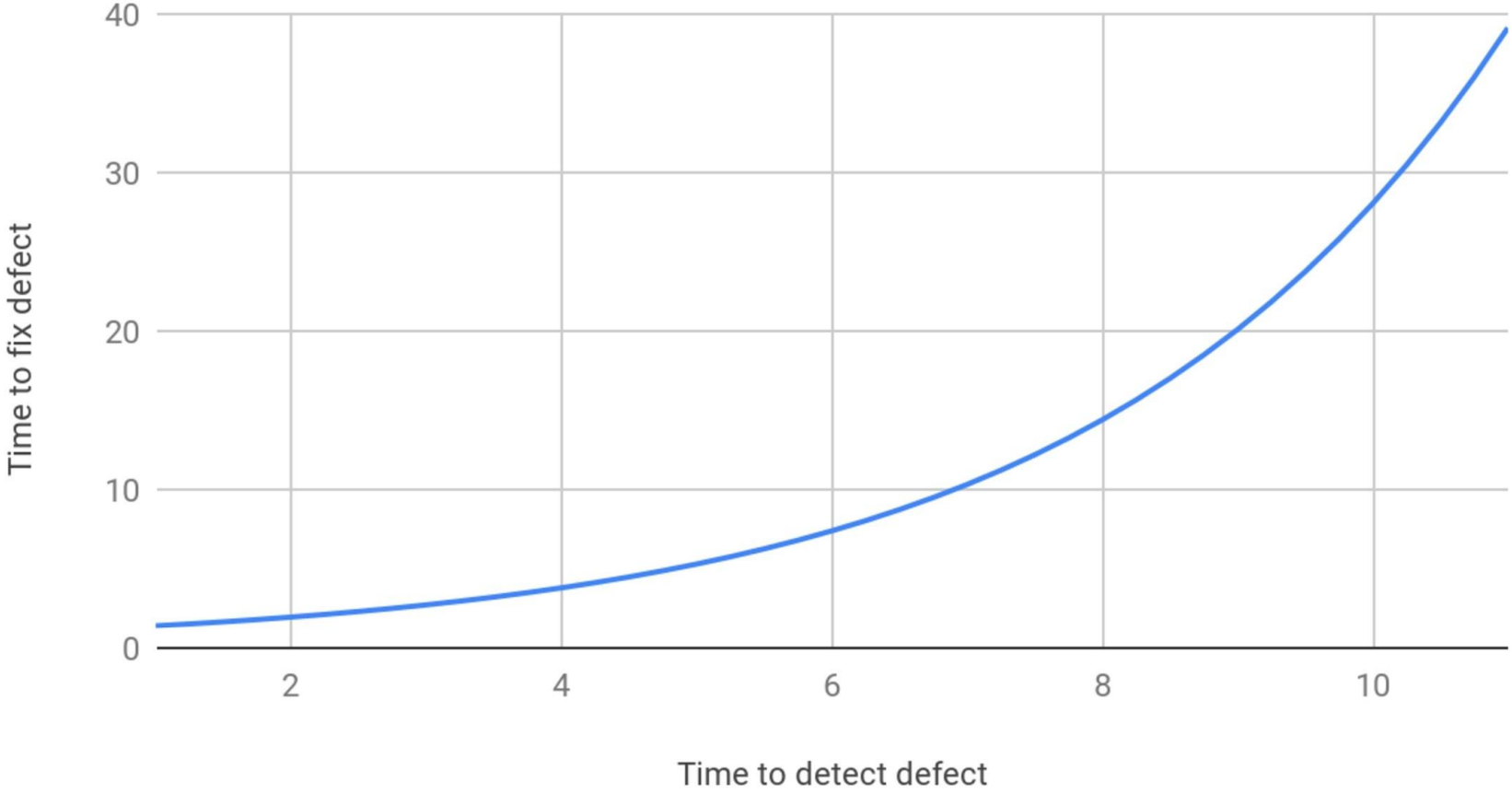
The longer the build the more context switching

- Whenever the build fails
- Whenever the build was necessary to provide intermediate feedback
- A unreliable toolchain substantially increases this cost

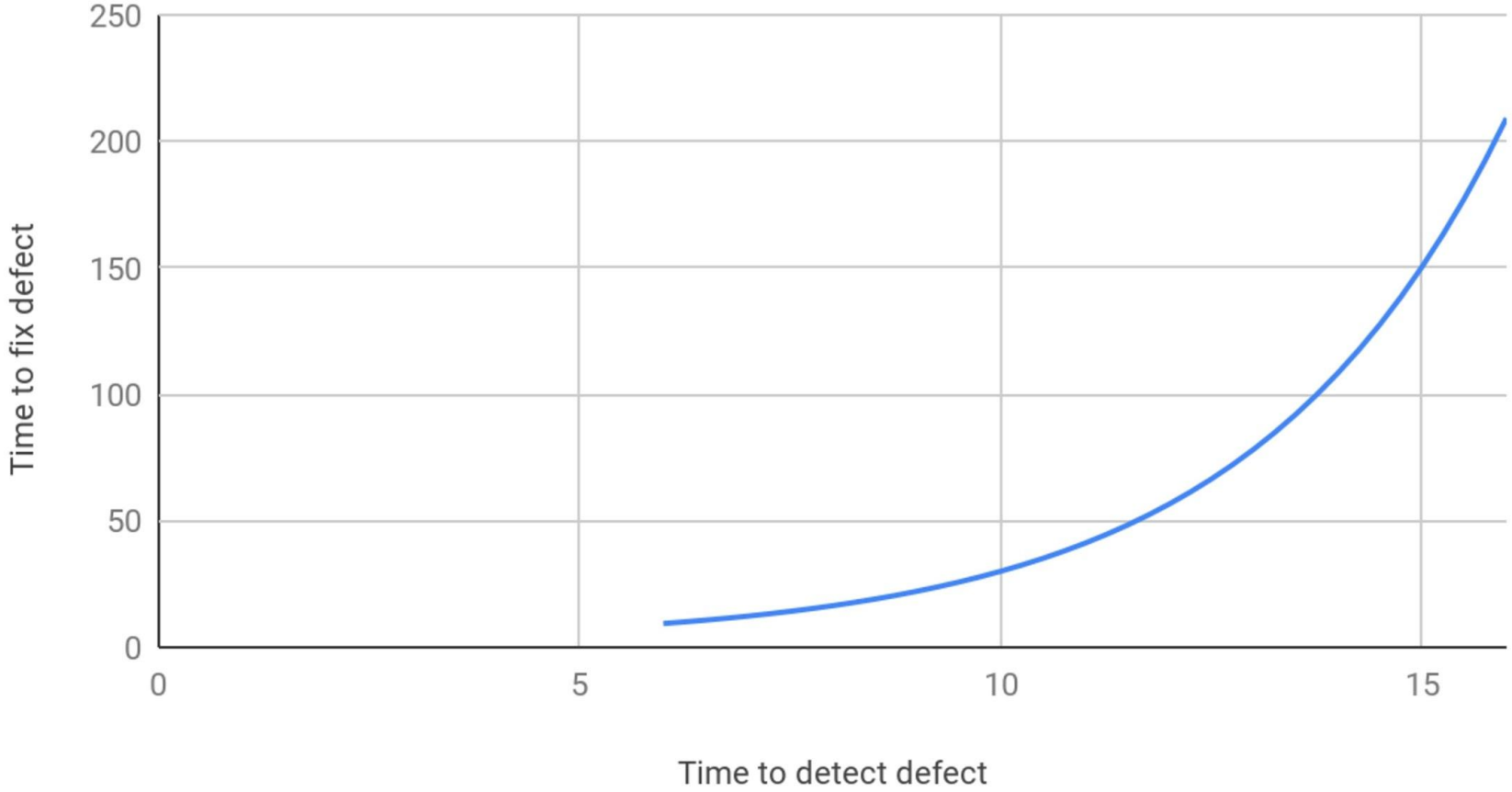
The longer the build the harder to debug

- The longer the build the more developers try to push in one build/PR/CI cycle
- The bigger the change-set, the harder to debug.

Fix time grows exponentially over detection time



Fix time grows exponentially over detection time



Let’s do less without doing less

Example Project

pom.xml

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

settings.gradle

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

Build Caching

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

Cache Key/Value Calculation

The *cacheKey* for Gradle Tasks/Maven Goals is based on the Inputs:

cacheKey(javaCompile) = hash(sourceFiles, jdk version, classpath, compiler args)

The *cacheEntry* contains the output:

cacheEntry[cacheKey(javaCompile)] = fileTree(classFiles)

For more information, see:

https://docs.gradle.org/current/userguide/build_cache.html

Other Cacheable Tasks/Goals Executions

Gradle Test/Maven Surefire inputs

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

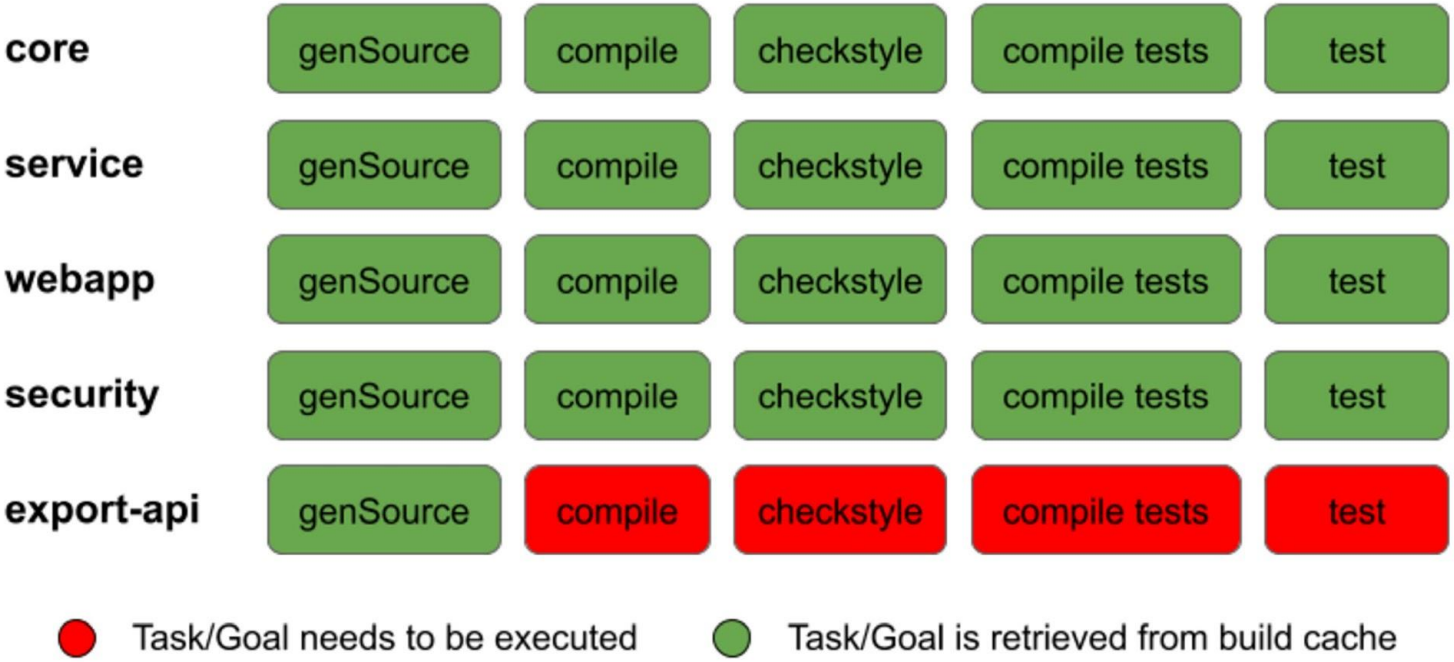
Checkstyle inputs

- Source Files
- Checkstyle version
- Checkstyle Config
- etc...

Caching is generic. It can apply to any task or goal meeting its requirements.
For IO-bound tasks/goals caching has no benefits (e.g. clean, copy).

Full rebuild & test regardless of change

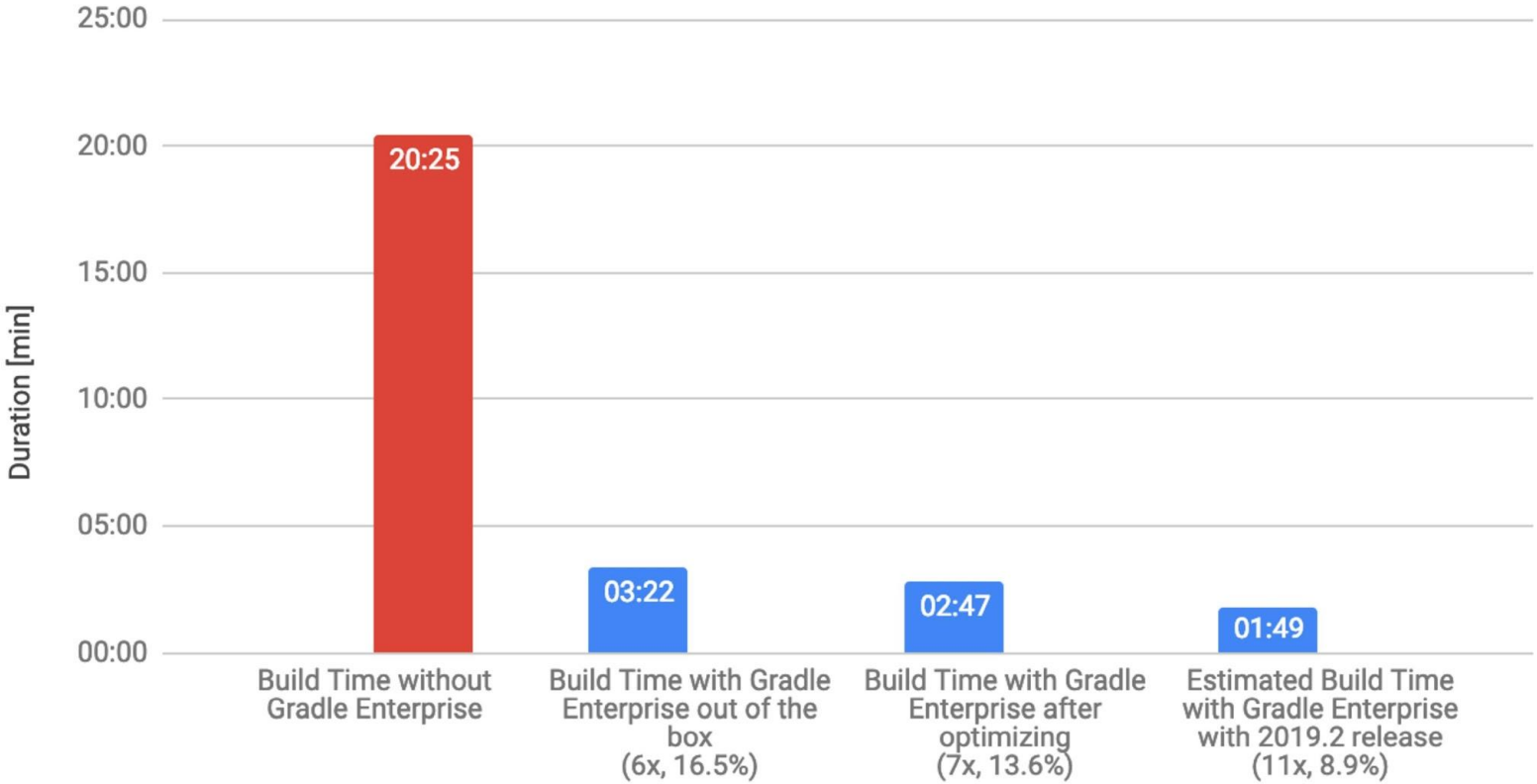
Changing a public method in the export-api module



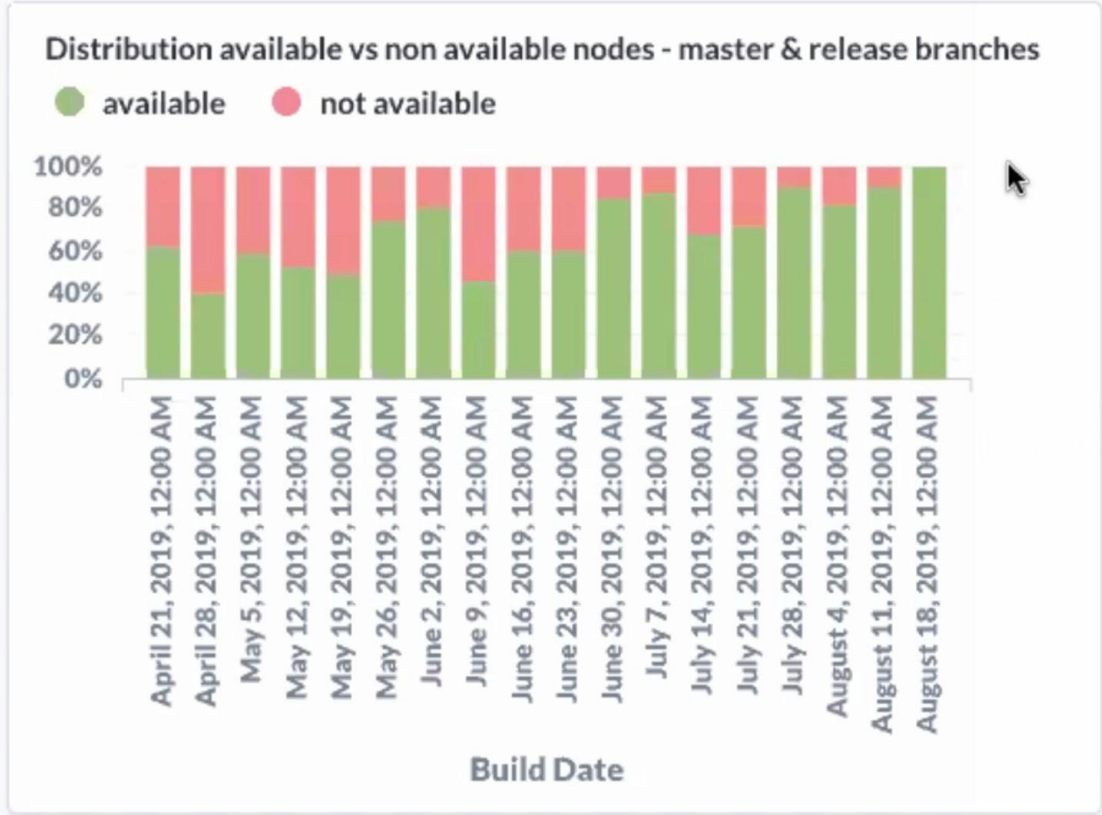
Changing the implementation of security

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

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



Reduce CI Queues



Modularization & Smaller API Surface

+

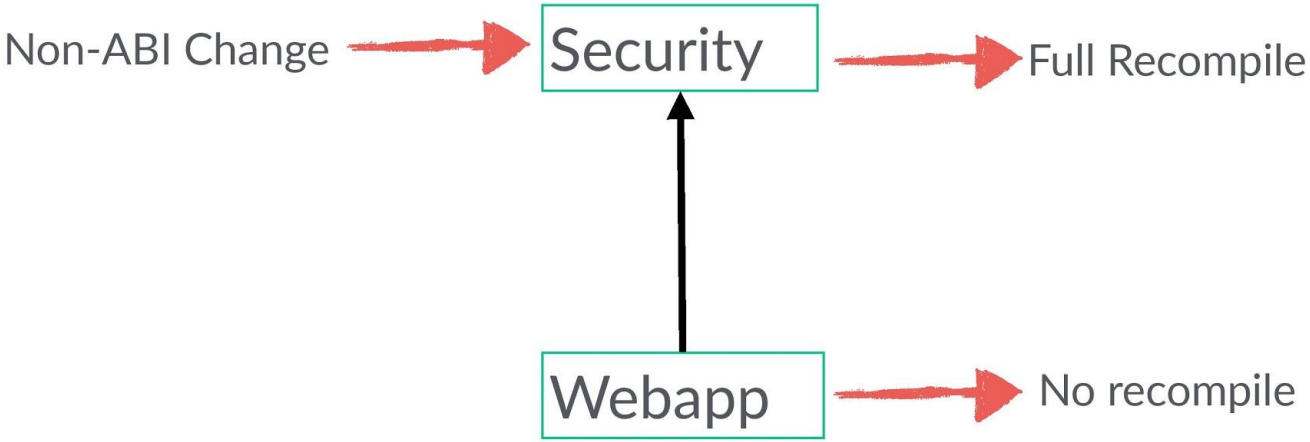
Acceleration Technologies

Much Faster Builds & Tests

ABI = Application Binary Interface

(Your code stripped of all implementation and private members/methods)

Compile Avoidance



Quantifiable Productivity Improvements

=

Strong Business Case for Modularization &
Reducing the monolith

Distributed Test Execution

- Besides caching, this is the other major acceleration technique

Data is essential to keep builds & tests fast

Who notices small regressions?

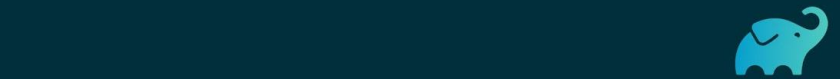
Who reports weird/flaky regressions?

Who can effectively determine the root cause of a regression?

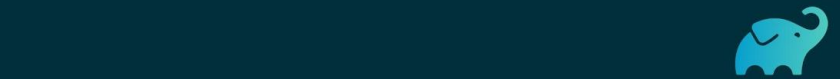
Who can determine the impact of a regression?

Capture comprehensive data of every build and test run

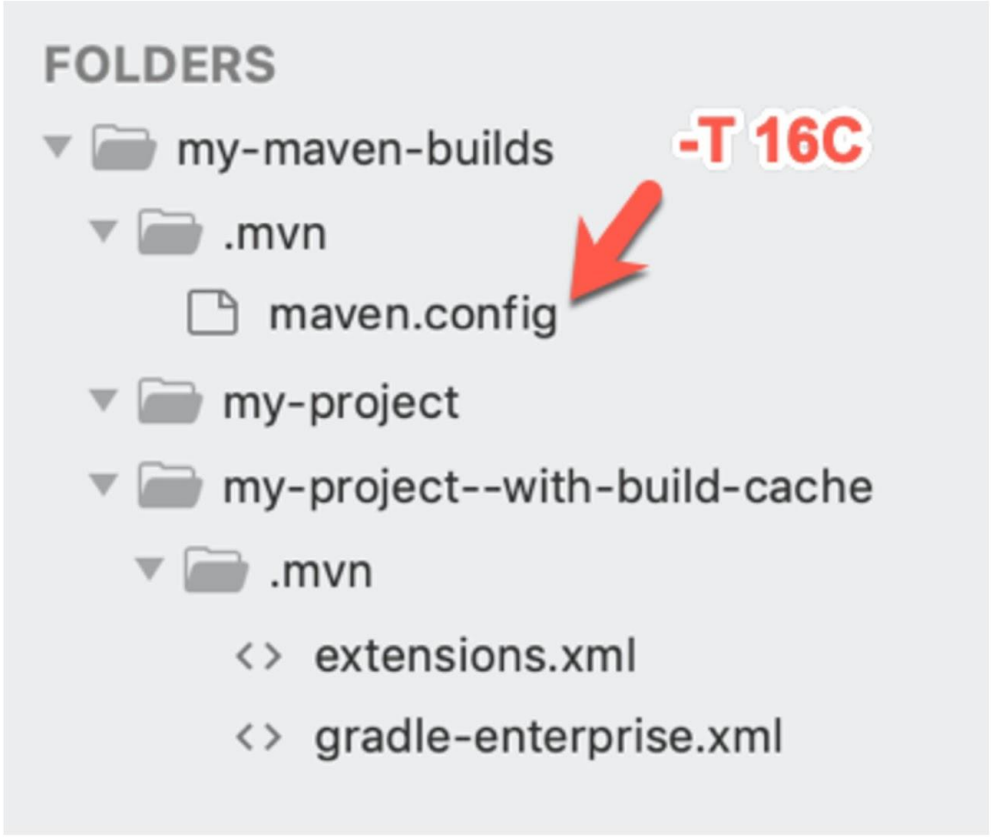
Demo Performance Insights



Troubleshooting

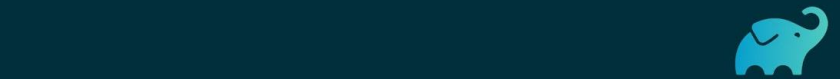


Comparing 13 infrastructure items, with 1 difference		
Operating system	Linux 4.15.0-46-generic	Linux 4.15.0-46-generic
CPU cores	16 cores	16 cores
Max Threads	128 threads	1 thread
Java runtime	Oracle Java(TM) SE Runtime Environment 1.8.0_201-b09	Oracle Java(TM) SE Runtime Environment 1.8.0_201-b09
Java VM	Oracle Java HotSpot(TM) 64-Bit Server VM 25.201-b09 (mixed mode)	Oracle Java HotSpot(TM) 64-Bit Server VM 25.201-b09 (mixed mode)
Max JVM memory heap size	7477 MB	7477 MB
Locale	English (United States)	English (United States)
Default charset	UTF-8	UTF-8
Username	xxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxx
Public hostname	xxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxx
Local hostname	xxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxx
Public IP address	xxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxx
Local IP addresses	xxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxx


FOLDERS
my-maven-builds
-T 16C
.mvn
maven.config
my-project
my-project--with-build-cache
.mvn
extensions.xml
gradle-enterprise.xml

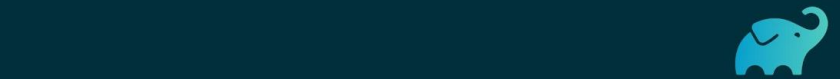
Demo Performance Troubleshooting

Pro-actively reduce incidents



I'm so tired of CI. Can I have a f... build which gives me errors that I am responsible of?

Demo Failure Analytics



Demo Test Analytics

Correctness

Build Scan

Developer Productivity Engineering

How to improve your developer experience and key software development processes

Improve and fix your developer experience from builds and tests to CI/CD

Developer Productivity Engineering is a discipline of using data and acceleration techniques to improve essential software development processes for greater automation, fast feedback cycles, and reliable feedback.

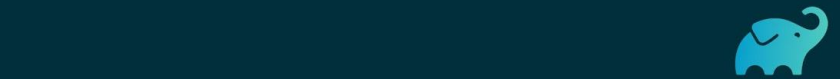
Download the Book

First Name *

Last Name *

Free download to early access edition: https://gradle.com/developer-productivity-engineering/

More Info at gradle.com



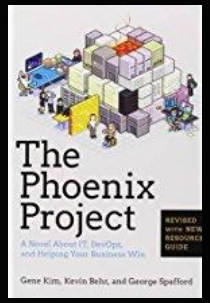
Thank you!

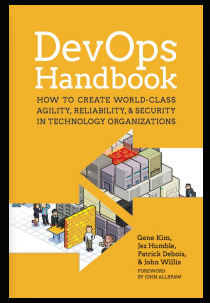
DevOps Resources

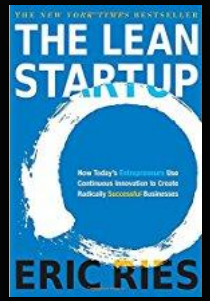
https://devopsfordefense.org/resources/

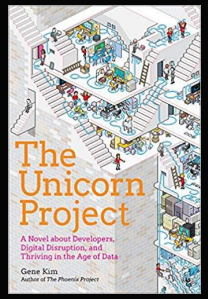
https://www.meetup.com/DevOps-for-Defense/
https://github.com/jondavid-black/DevOpsForDefense
devopsfordefense@gmail.com

Books / Publications:





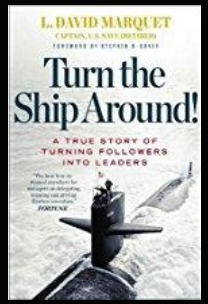


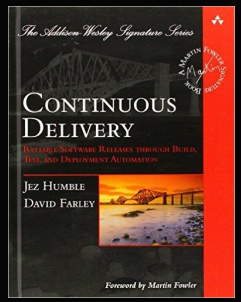


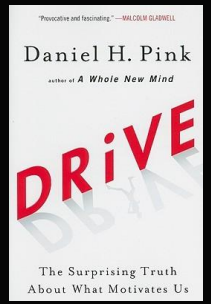
Conference Presentations (YouTube):

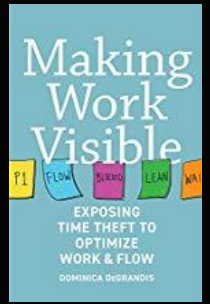
- DevOps Enterprise Summit (DOES)
- IT Revolution
- Velocity
- GoTo

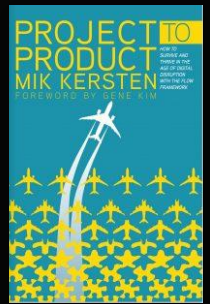
YouTube

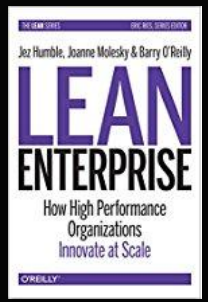


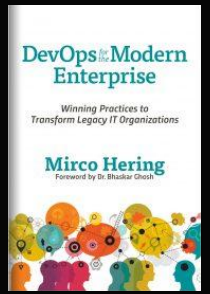


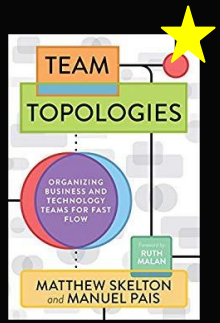












Group Exercise: Lean Coffee

1. Each table has a facilitator.
2. The facilitator has a short introduction.
3. Everyone write down questions or topics for discussion on the subject. Place them in the middle of the table.
4. The group votes on each question or topic by placing a dot on the card. 3 votes per person.
5. Cards with most dots goes first. Set a timer for 5 minutes and discuss.
6. After 5 minutes, either vote (thumbs up/down) to keep going or move on to the next card.

Suggested Topic: “Toolchain Interactions”