UNLOCKING
DEVELOPER
PRODUCTIVITY

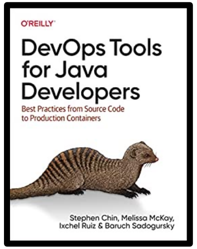
A DUAL PERSPECTIVE ON BEST PRACTICES & TESTING

DUAL PERSPECTIVE ON DEVELOPER PRODUCTIVITY?

DUAL PERSPECTIVE ON DEVELOPER PRODUCTIVITY?

BARUCH SADOGURSKY - @JBARUCH

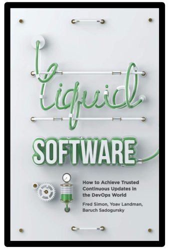
- Developer Productivity Advocate
- Gradle Inc
- Development -> DevOps -> #DPE


O'REILLY

DevOps Tools for Java Developers

Best Practices from Source Code to Production Containers

Stephen Chin, Melissa McKay, Ixchel Ruiz & Baruch Sadogursky


Liquid SOFTWARE

How to Achieve Trusted Continuous Updates in the DevOps World

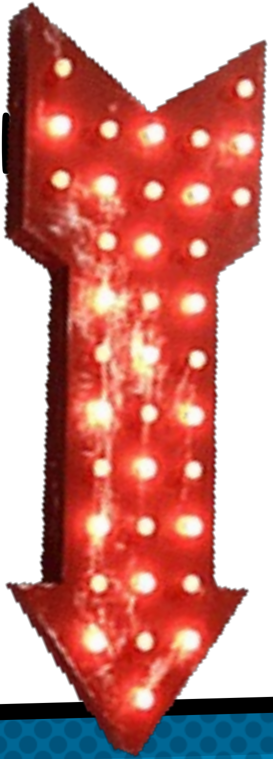
Fred Simon, Yoav Landman, Baruch Sadogursky

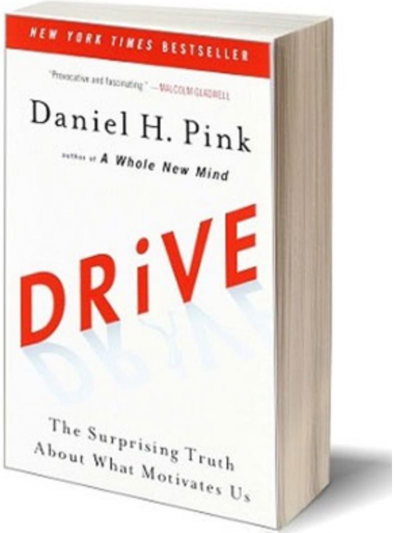
ELI ALEYNER - @EALEYNER

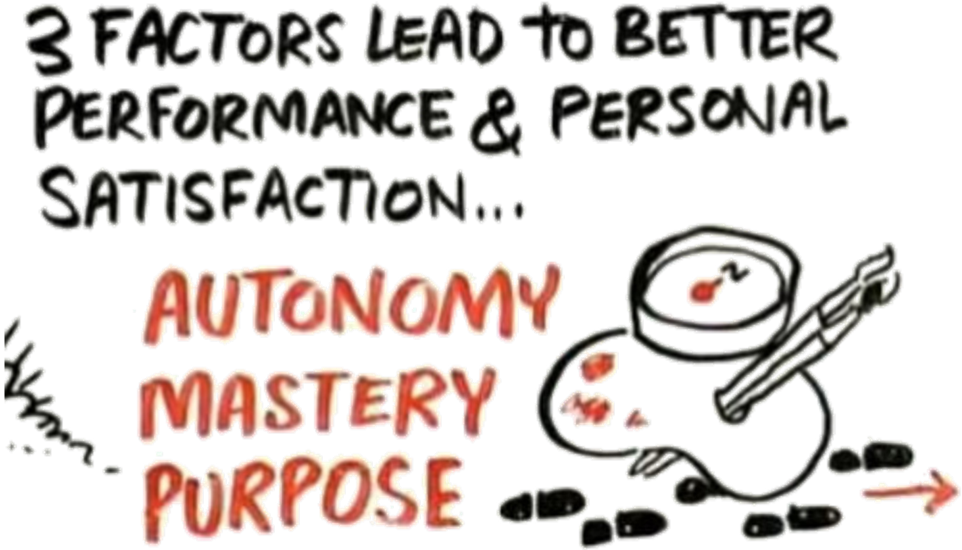
- Co-founder
- AtomicJar Inc
- Founding team @AWS
- Founding team @Bing.com

SHOWNOTES

- speaking.jbaru.c
- Slides
- Video
- All the links!

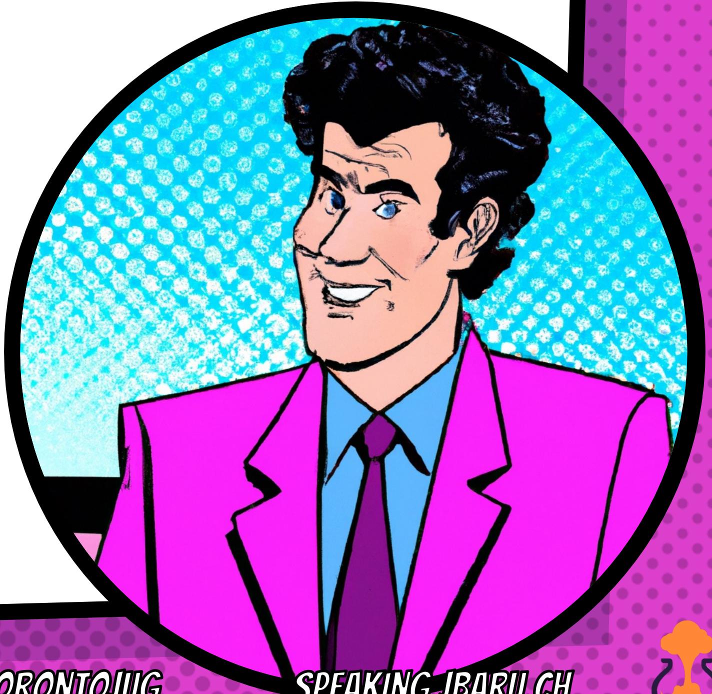



NEW YORK TIMES BESTSELLER
"Provocative and fascinating." —MALCOLM GLADWELL
Daniel H. Pink
author of A Whole New Mind
DRiVE
The Surprising Truth About What Motivates Us

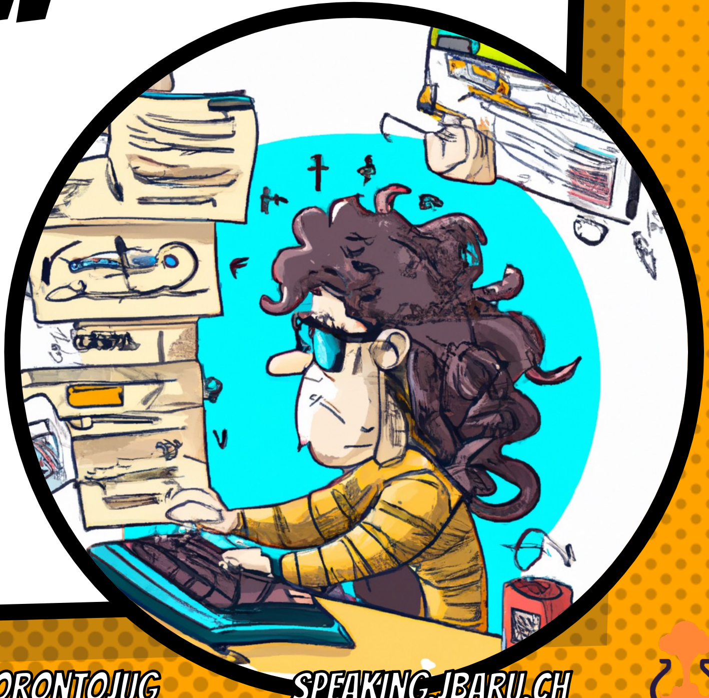


3 FACTORS LEAD TO BETTER PERFORMANCE & PERSONAL SATISFACTION...

AUTONOMY
MASTERY
PURPOSE



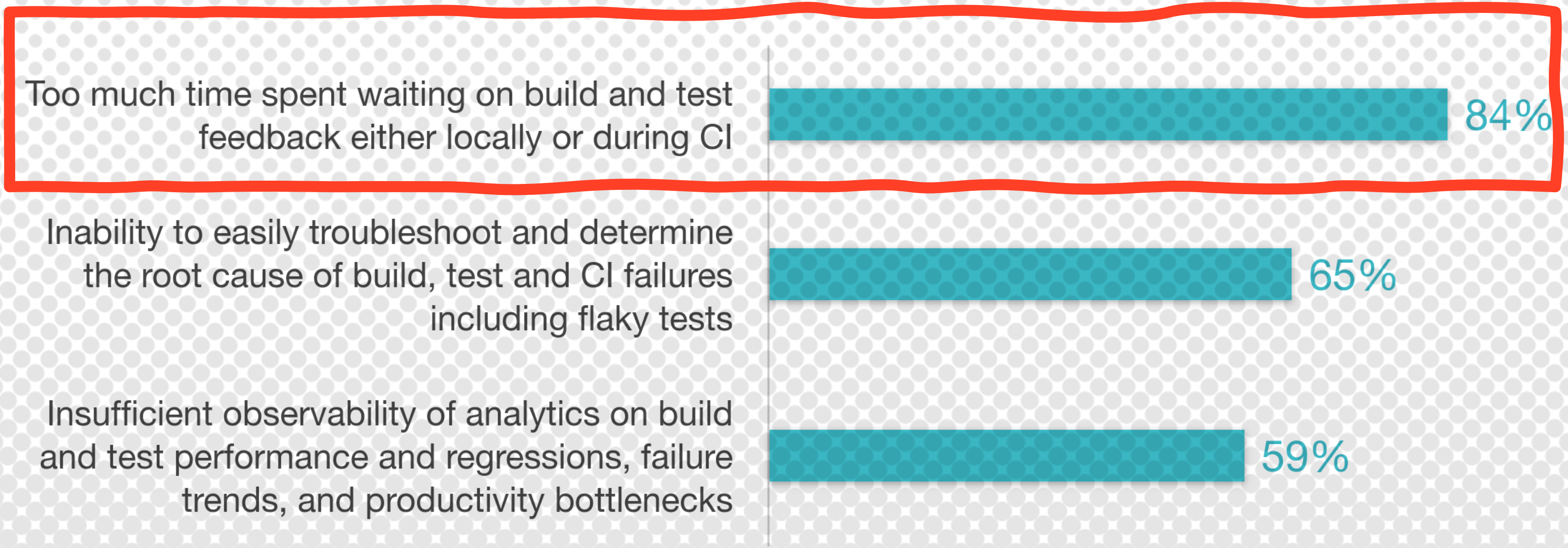
DON'T RUIN THE FLOW



"THE BUILD TAKES FOREVER, I AM DISTRACTED TO DO OTHER THINGS AND THE CONTEXT SWITCH IS TERRIBLE"

Development Pains are Widespread

Which of the following challenges or pain points did your organization experience prior to implementing Developer Productivity Engineering?

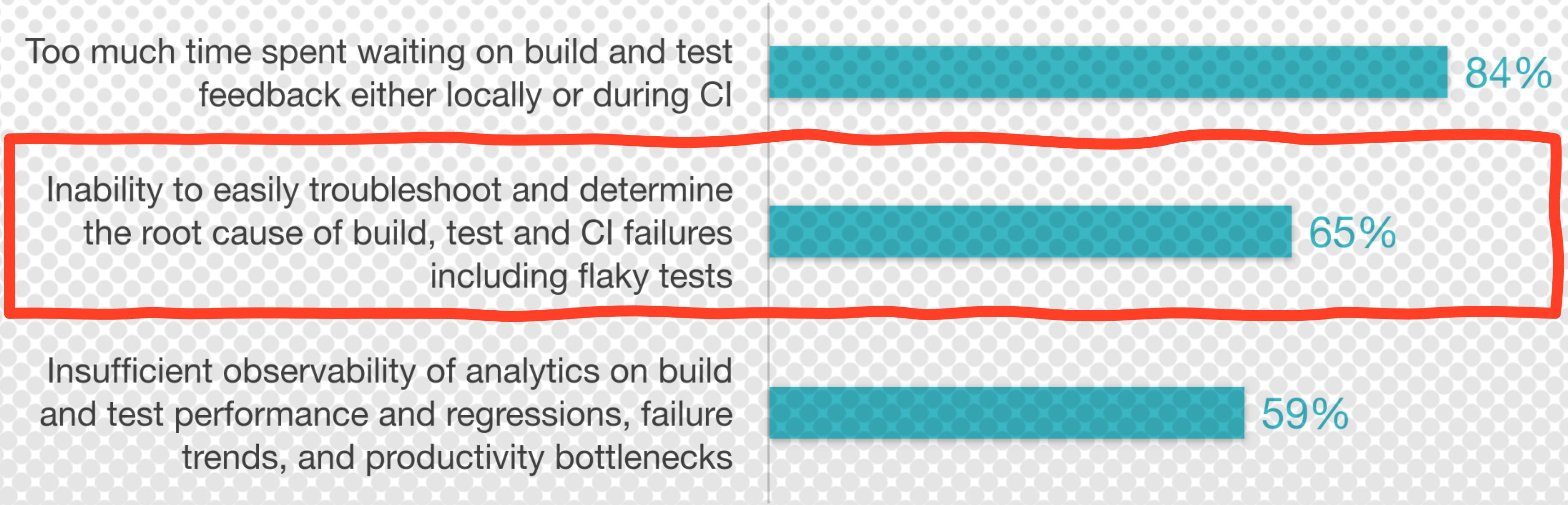


DON'T FRUSTRATE THE DEVELOPERS

"WE HAVE A FLANKY TEST, IT IS IRRELEVANT 99.5% OF THE TIME , BUT IT ALWAYS RUNS AND IT IS LAST IN THE SUITE"

Development Pains are Widespread

Which of the following challenges or pain points did your organization experience prior to implementing Developer Productivity Engineering?

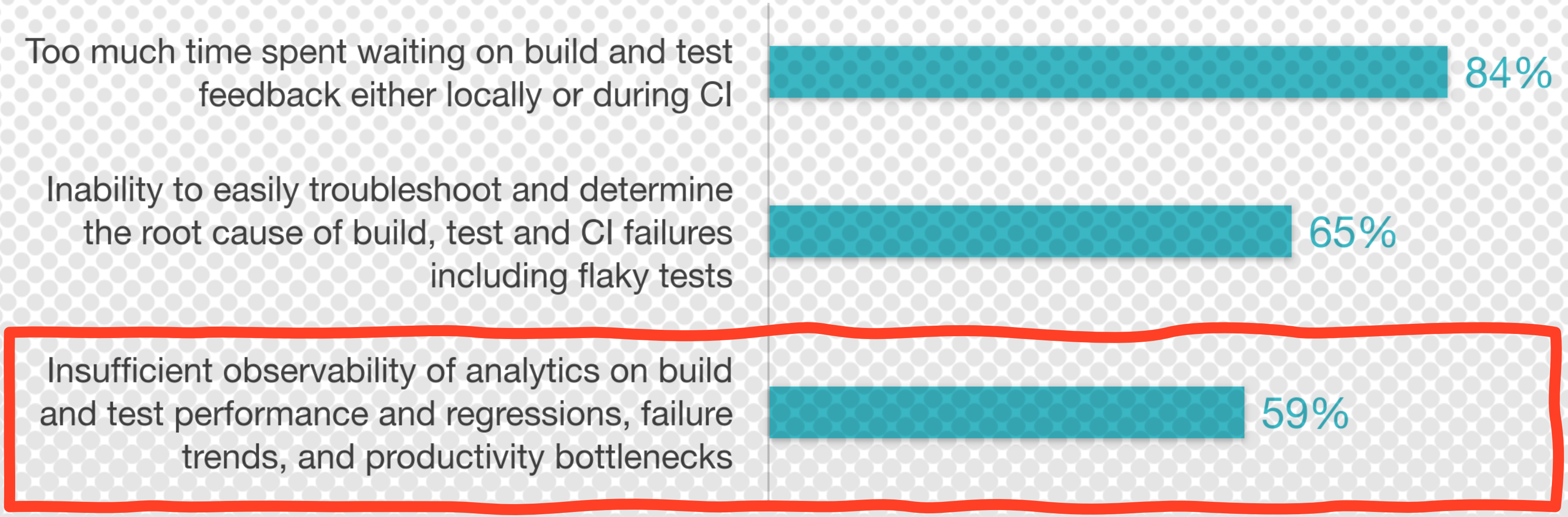


DON'T BOIL THE FROG

"I HAVE A FEELING THAT EVERYTHING IS SLOWER SOMEHOW…"

Development Pains are Widespread

Which of the following challenges or pain points did your organization experience prior to implementing Developer Productivity Engineering?



DEVELOPER PRODUCTIVITY :: A/M/P :: MOTIVATION

Autonomy	→	Tools and people aren't in my way
Mastery	→	Tools and processes help me to excel
Purpose	→	I want to be *productive*, i.e. create the product

DEVELOPER PRODUCTIVITY ENGINEERING!

DEVELOPER PRODUCTIVITY ENGINEERING

Foster Faster Feedback

Collaborate through Effective Tooling

Embrace Rigorous Observability for Proactive Improvement

Eliminate Toil for Developers

Prioritize Automation and Eliminate Bottlenecks

Dedicated Organizational Mindset

Outcomes Over Output

TALK IS CHEAP, SHOW ME THE GOODS!

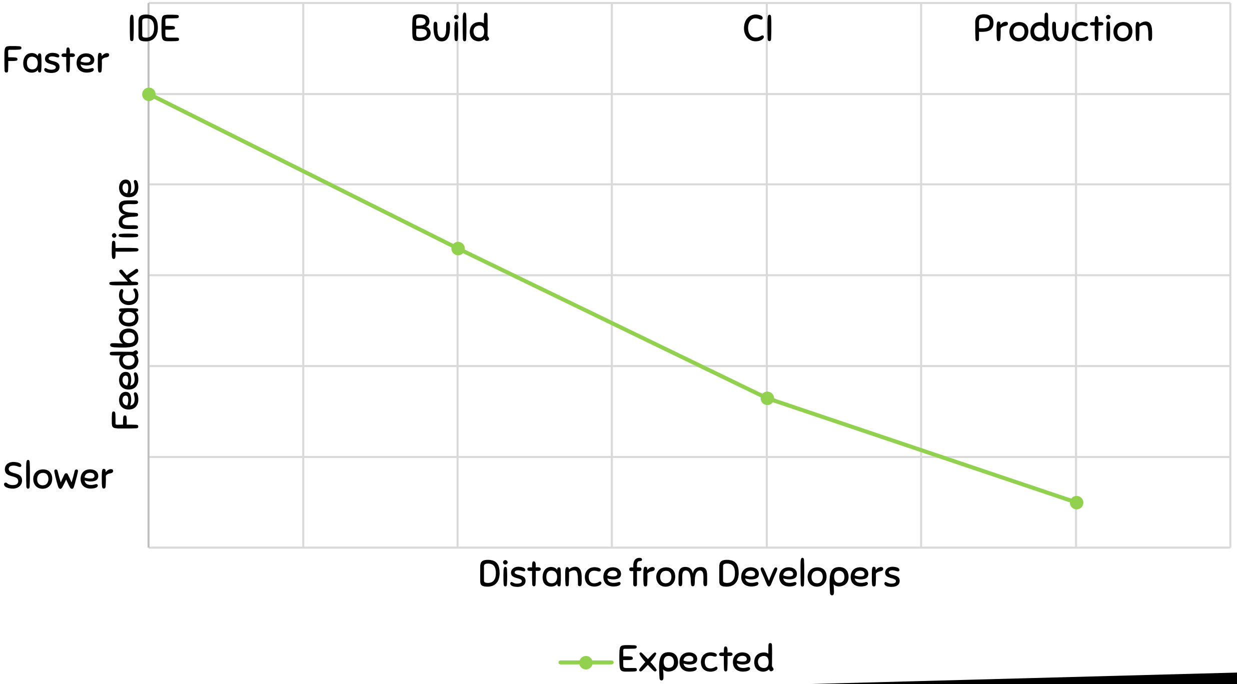
SMALL DPE IMPROVEMENTS MAKE A HUGE DIFFERENCE

- Generate code faster: Better IDE
- Test better: Testcontainers
- Enforce better code: Sonar
- Test more reliably: Flaky test detection
- Foster Faster Feedback:

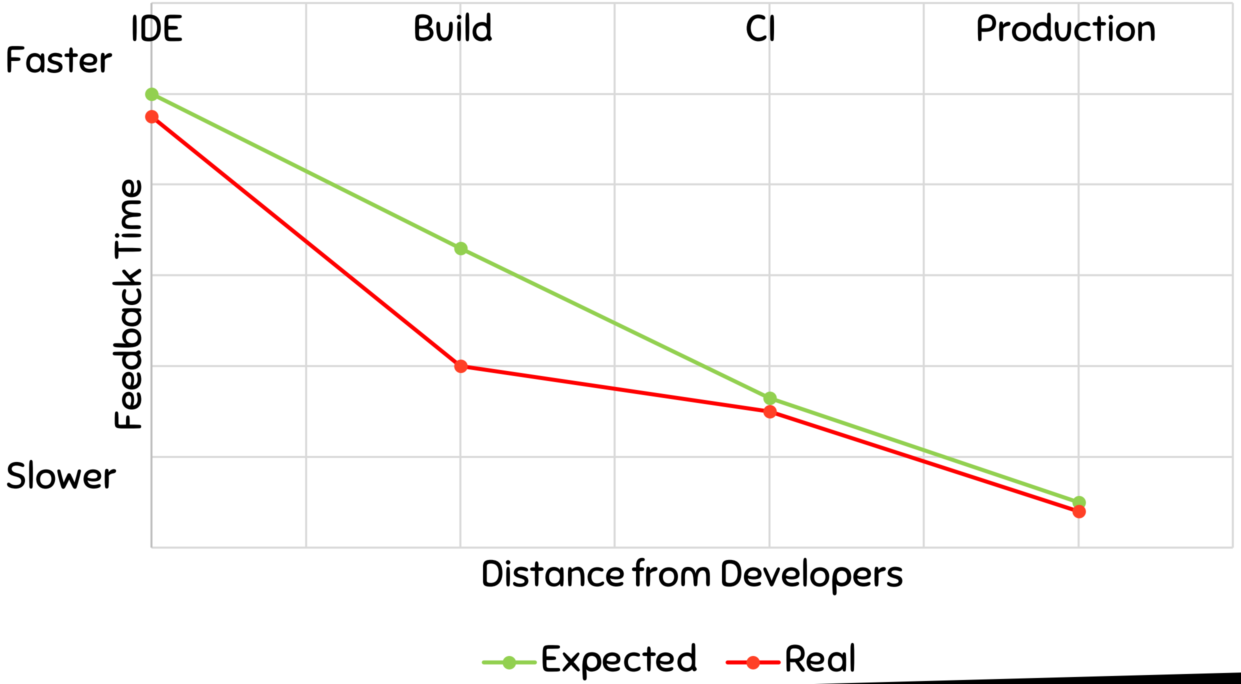
FEEDBACK EFFICIENCY

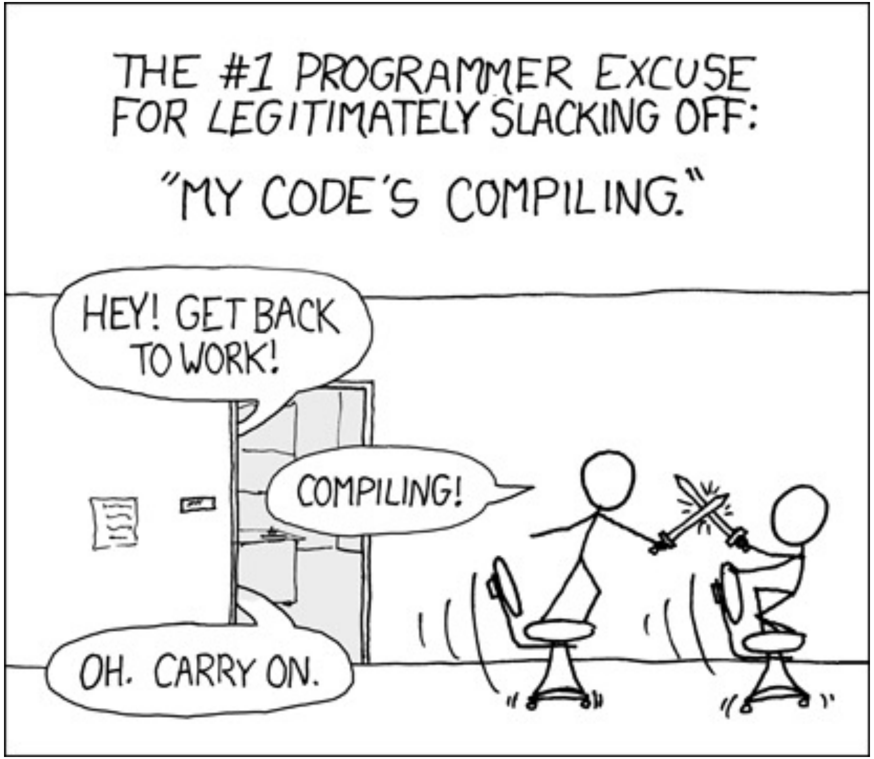
- IDE: Sub-seconds (I type, it marks it red)
- Build: Seconds
- CI: Minutes
- Production: Hours/Days

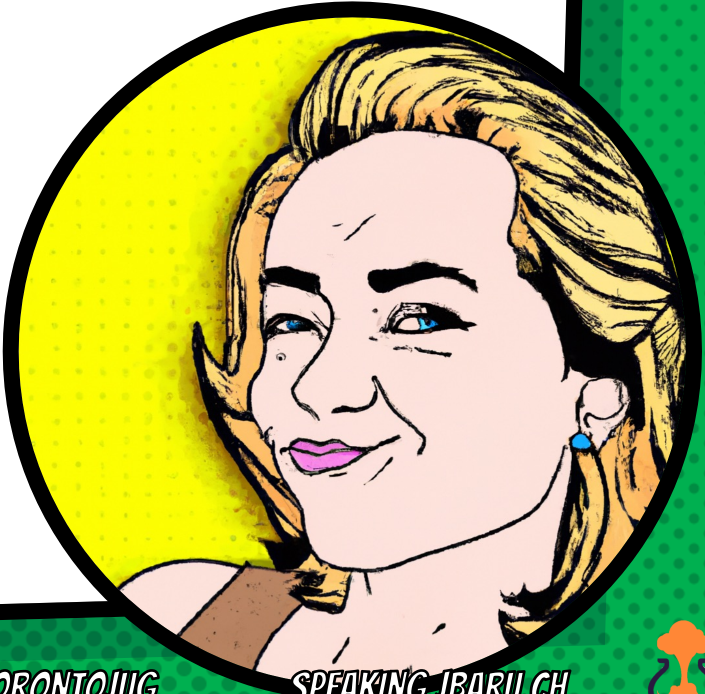
REVERSE DEPENDENCY ON DISTANCE FROM DEVELOPERS



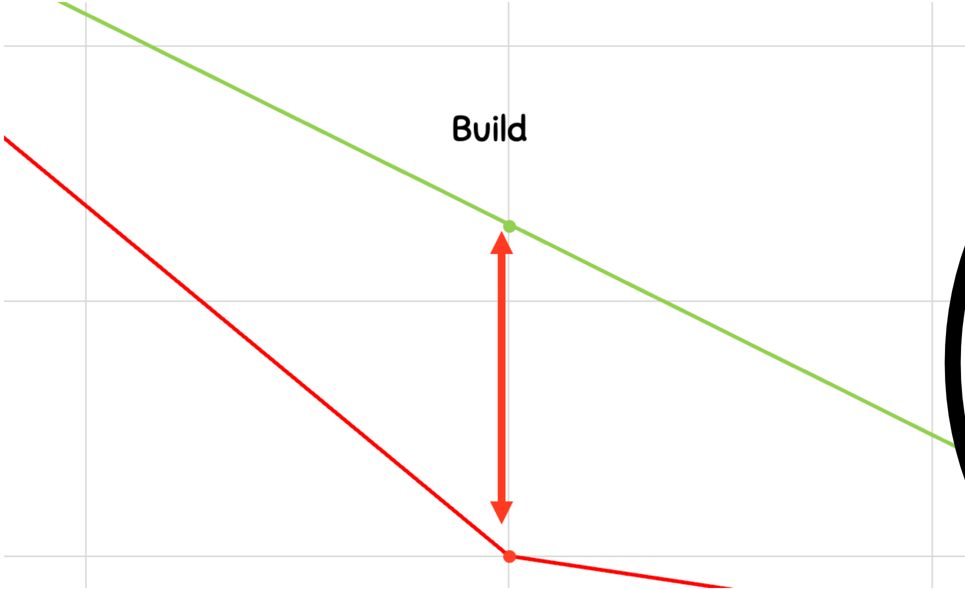
REVERSE DEPENDENCY ON DISTANCE FROM DEVELOPERS

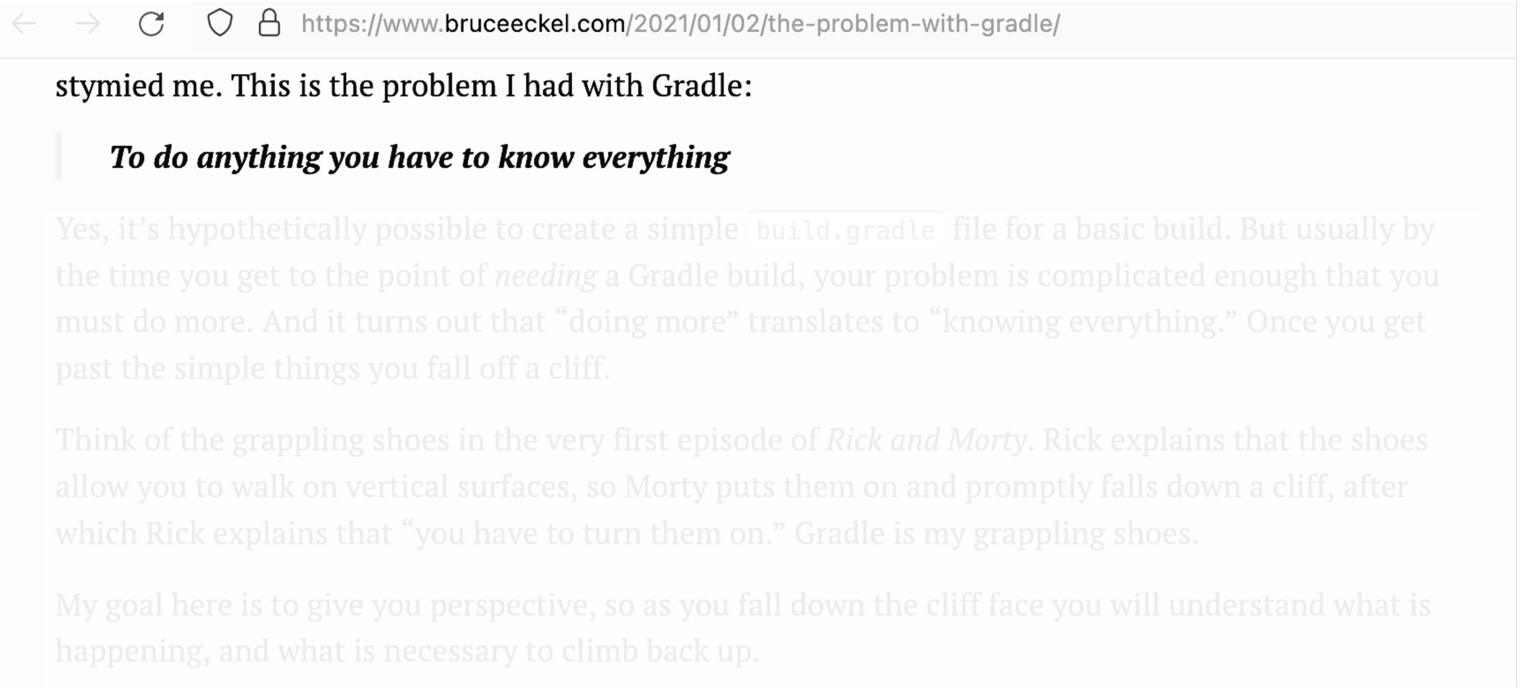



THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.



IT IS SLOW!





https://www.bruceeckel.com/2021/01/02/the-problem-with-gradle/

stymied me. This is the problem I had with Gradle:

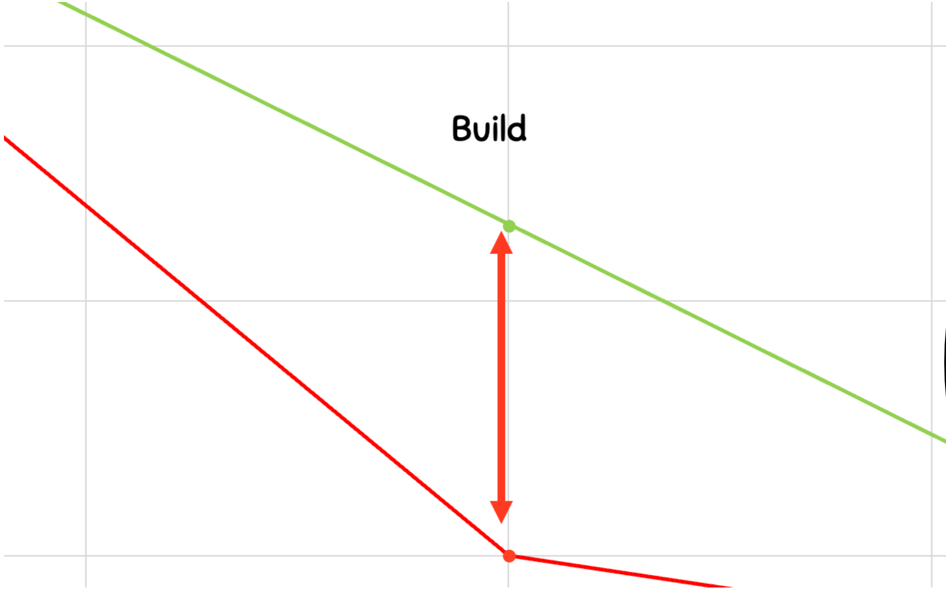
> ***To do anything you have to know everything***

Yes, it’s hypothetically possible to create a simple `build.gradle` file for a basic build. But usually by the time you get to the point of *needing* a Gradle build, your problem is complicated enough that you must do more. And it turns out that “doing more” translates to “knowing everything.” Once you get past the simple things you fall off a cliff.

Think of the grappling shoes in the very first episode of *Rick and Morty*. Rick explains that the shoes allow you to walk on vertical surfaces, so Morty puts them on and promptly falls down a cliff, after which Rick explains that “you have to turn them on.” Gradle is my grappling shoes.

My goal here is to give you perspective, so as you fall down the cliff face you will understand what is happening, and what is necessary to climb back up.

IT IS SLOW AND THE DEVELOPERS HAVE NO IDEA WHY!



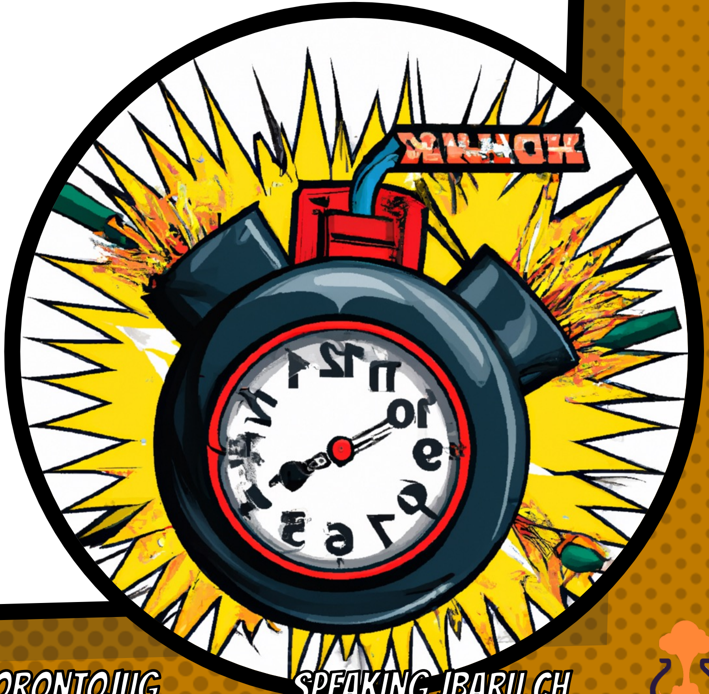
WHAT IS BUILD?

- Project setup
- Downloading the Internet
- Artifact generation: Compilation, packaging, etc
- Tests
- Artifact deployment

WHAT CAN GO WRONG?

- Project setup
- Downloading the Internet
- Artifact generation: Compilation, packaging, etc
- Tests
- Artifact deployment

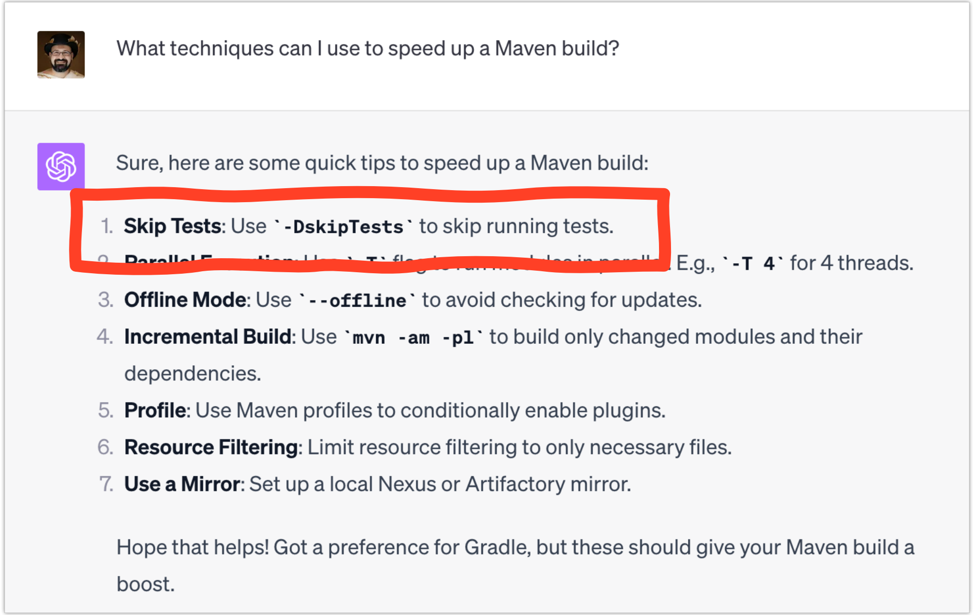
WHEN CAN IT GO WRONG?



AT ANY TIME.

THE BUILD FRUSTRATES THE DEVELOPERS

LET'S ASK CHATGPT



What techniques can I use to speed up a Maven build?

Sure, here are some quick tips to speed up a Maven build:

1. **Skip Tests**: Use `` `-DskipTests` `` to skip running tests.
2. **Parallel Execution**: Use `` `-T` `` flag to run modules in parallel. E.g., `` `-T 4` `` for 4 threads.
3. **Offline Mode**: Use `` `--offline` `` to avoid checking for updates.
4. **Incremental Build**: Use `` `mvn -am -pl` `` to build only changed modules and their dependencies.
5. **Profile**: Use Maven profiles to conditionally enable plugins.
6. **Resource Filtering**: Limit resource filtering to only necessary files.
7. **Use a Mirror**: Set up a local Nexus or Artifactory mirror.

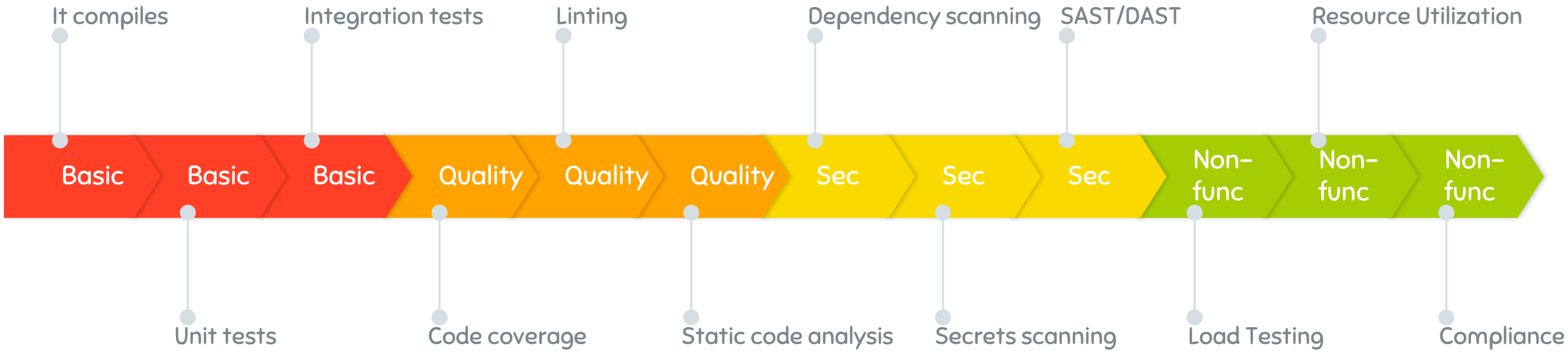
Hope that helps! Got a preference for Gradle, but these should give your Maven build a boost.

WHAT THE ACTUAL F*CK?!

- Skipping tests defeats the purpose of the build!
- How about skipping compilation?
- We want faster feedback, not less feedback

WHAT FEEDBACK DO WE WANT?

CI/CD PIPELINE QUALITY GATES



Basic – It compiles

Basic – Unit tests

Basic – Integration tests

Quality – Code coverage

Quality – Linting

Quality – Static code analysis

Sec – Dependency scanning

Sec – Secrets scanning

Sec – SAST/DAST

Non-func – Load Testing

Non-func – Resource Utilization

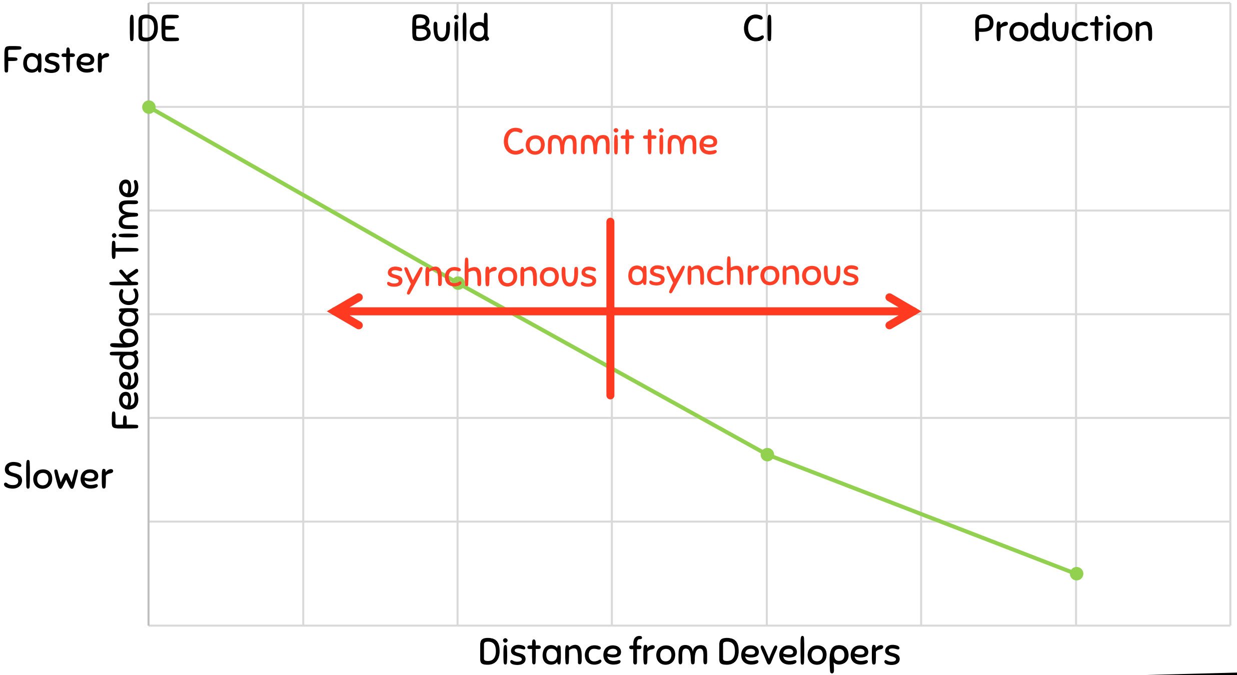
Non-func – Compliance

CI/CD? NOT GREAT, NOT TERRIBLE.

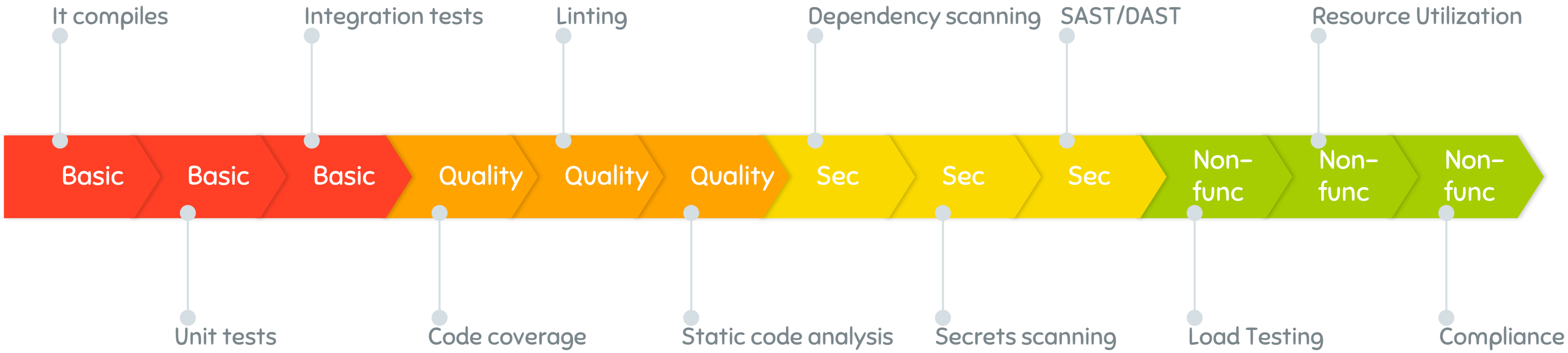
TWO TYPES OF FEEDBACK

ASYNCHRONOUS	× e.g., CI/CD × we never wait for it × results are distracting
SYNCHRONOUS	× e.g., build × we'll wait for it in the flow × we'll be pissed off when it's slow

REVERSE DEPENDENCY ON DISTANCE FROM DEVELOPERS



IDEAL BUILD TIME FEEDBACK



It compiles
Integration tests
Linting
Dependency scanning
SAST/DAST
Resource Utilization

Basic
Basic
Basic
Quality
Quality
Quality
Sec
Sec
Sec
Non-func
Non-func
Non-func

Unit tests
Code coverage
Static code analysis
Secrets scanning
Load Testing
Compliance

BUT WON'T IT SLOW DOWN THE BUILD?!

DELIGHTFUL BUILD (PICK TWO):

☑ PROVIDES MAX FEEDBACK
☑ FAST

SKIP WHAT CAN BE SKIPPED
(BUT NO MORE!)

AVOIDANCE: INCREMENTAL BUILD

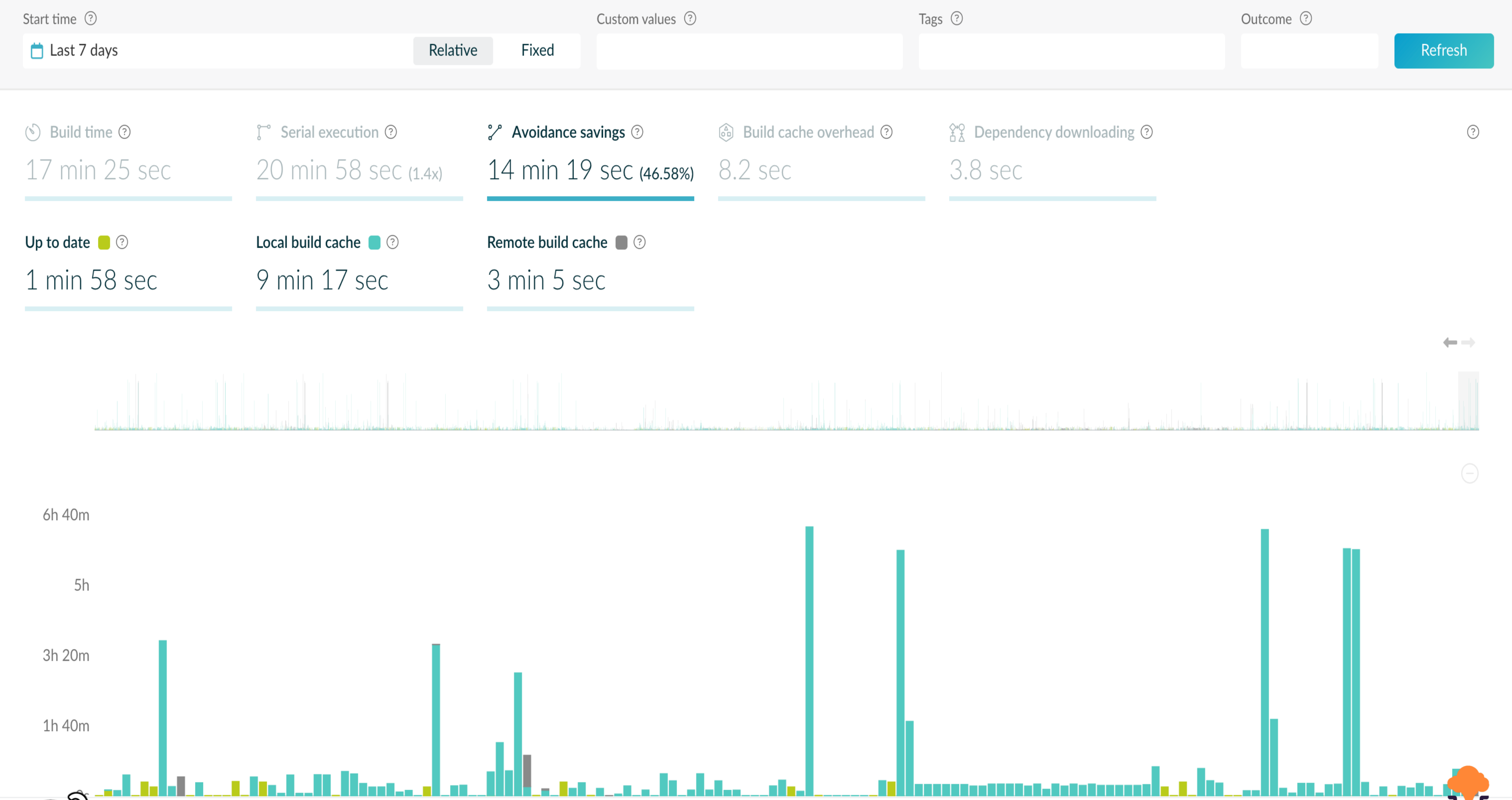
- Don't build what didn't changed
- Don't build what isn't affected

AVOIDANCE: INCREMENTAL BUILD SHORTCOMINGS

- Relies on produced artifacts
- Relies on architectural decisions

AVOIDANCE: CACHING

- Makes the build faster
- Makes the build faster for everybody
- Makes the build faster always
- Makes all parts of the build faster



Start time ? Last 7 days Relative Fixed

Custom values ?

Tags ?

Outcome ?

Refresh

Build time ?	Serial execution ?	Avoidance savings ?	Build cache overhead ?	Dependency downloading ?
17 min 25 sec	20 min 58 sec (1.4x)	14 min 19 sec (46.58%)	8.2 sec	3.8 sec

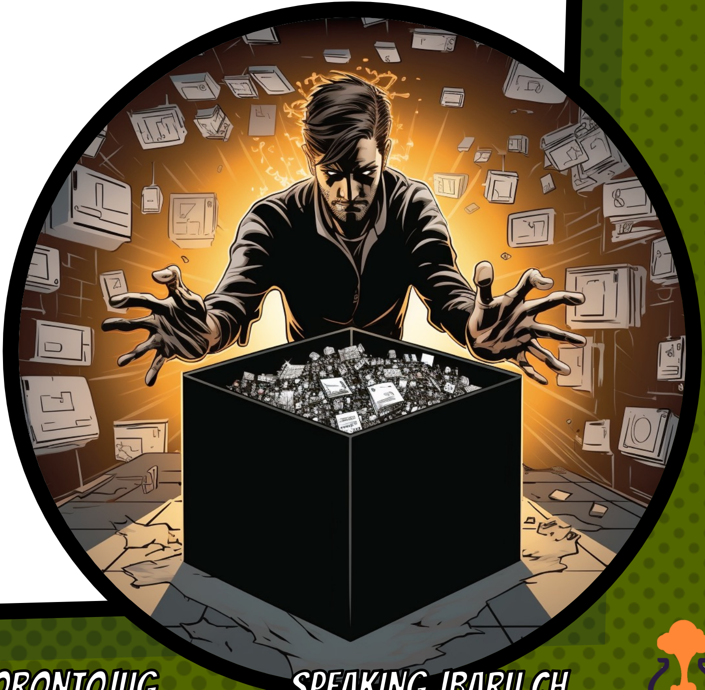
Up to date 🟩 ?	Local build cache 🟦 ?	Remote build cache ⬛ ?
1 min 58 sec	9 min 17 sec	3 min 5 sec

AVOIDANCE: PREDICTIVE TEST SELECTION

- Learns code changes effects de-facto
- Skips tests with
  high degree of confidence

HOW TEST PREDICTION WORKS

- Code changes and test results are thrown into learning model
- After a while, the model predicts which changes fail which tests



TEST PREDICTION

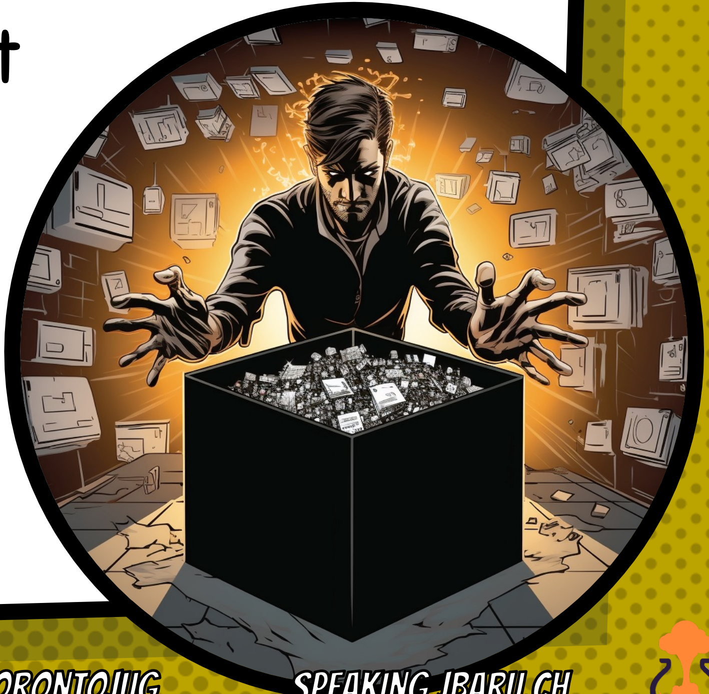
What changed
Where it changed

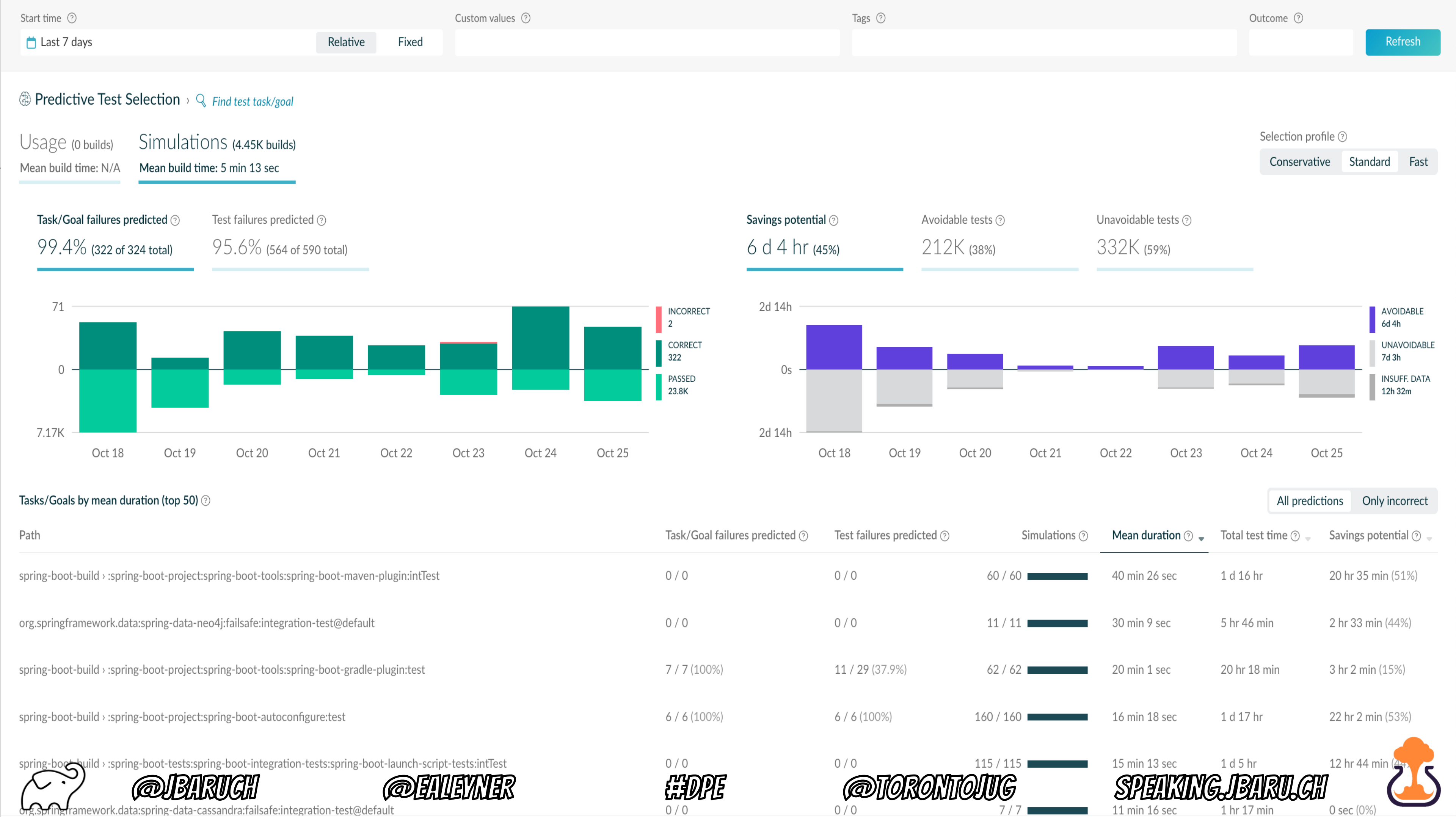
Correlate with observed test failures

Predictions which changes will fail which tests

BLACK MAGIC IN ACTION

- The more tests a project has, the less they break
- Refactorings in Java break tests less than in JavaScript





Predictive Test Selection

Usage (0 builds)
Mean build time: N/A

Simulations (4.45K builds)
Mean build time: 5 min 13 sec

Selection profile

Task/Goal failures predicted
99.4% (322 of 324 total)

Test failures predicted
95.6% (564 of 590 total)

Savings potential
6 d 4 hr (45%)

Avoidable tests
212K (38%)

Unavoidable tests
332K (59%)

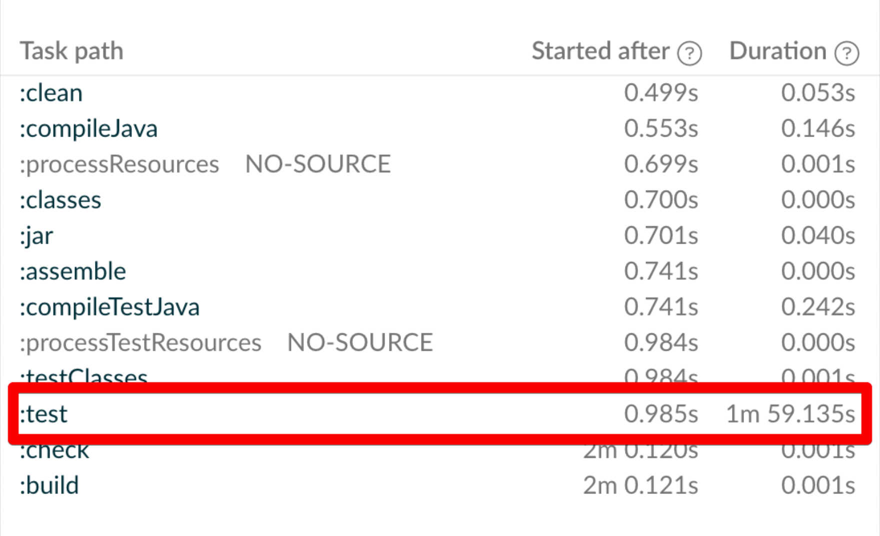
Tasks/Goals by mean duration (top 50)

Path	Task/Goal failures predicted	Test failures predicted	Simulations	Mean duration	Total test time	Savings potential
spring-boot-build › :spring-boot-project:spring-boot-tools:spring-boot-maven-plugin:intTest	0 / 0	0 / 0	60 / 60	40 min 26 sec	1 d 16 hr	20 hr 35 min (51%)
org.springframework.data:spring-data-neo4j:failsafe:integration-test@default	0 / 0	0 / 0	11 / 11	30 min 9 sec	5 hr 46 min	2 hr 33 min (44%)
spring-boot-build › :spring-boot-project:spring-boot-tools:spring-boot-gradle-plugin:test	7 / 7 (100%)	11 / 29 (37.9%)	62 / 62	20 min 1 sec	20 hr 18 min	3 hr 2 min (15%)
spring-boot-build › :spring-boot-project:spring-boot-autoconfigure:test	6 / 6 (100%)	6 / 6 (100%)	160 / 160	16 min 18 sec	1 d 17 hr	22 hr 2 min (53%)
spring-boo build › :spring-boot-tests:spring-boot-integration-tests:spring-boot-launch-script-tests:intTest	0 / 0	0 / 0	115 / 115	15 min 13 sec	1 d 5 hr	12 hr 44 min (4
org.springframework.data:spring-data-cassandra:failsafe:integration-test@default	0 / 0	0 / 0	7 / 7	11 min 16 sec	1 hr 17 min	0 sec (0%)

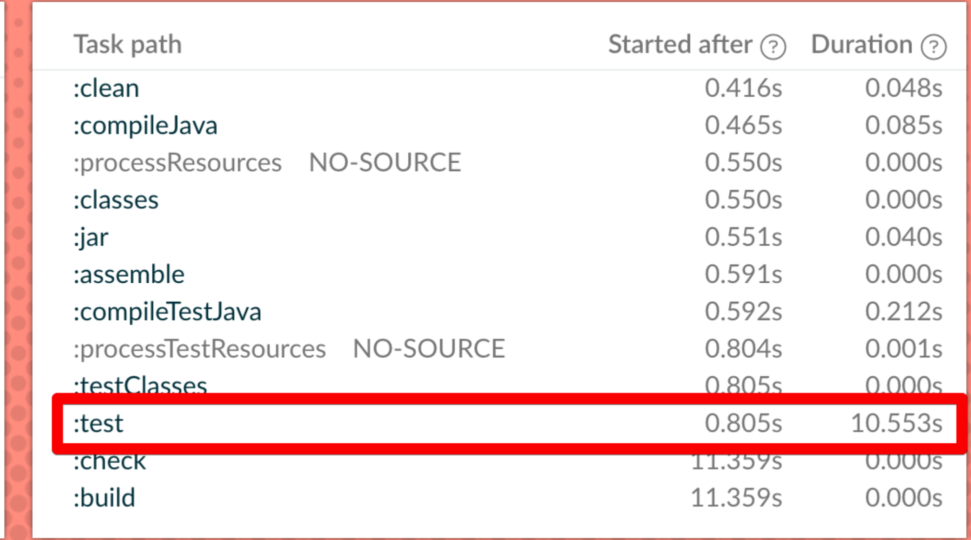
SPEED UP WHAT CAN'T BE SKIPPED

TEST PARALLELIZATION

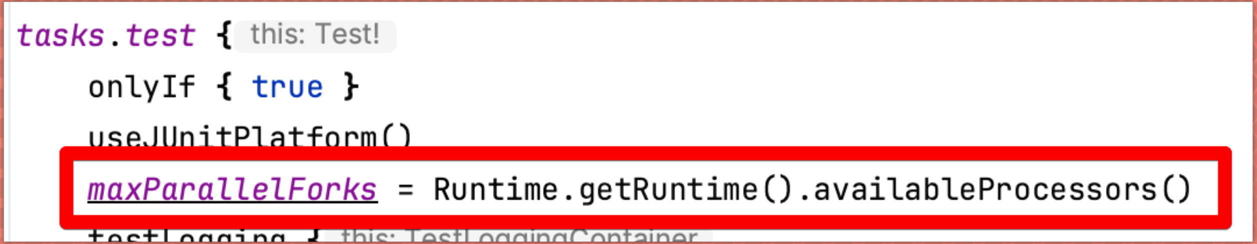
- Use max power of local machine
- (Yes, your boss should buy you the bleeding edge)



Task path		Started after ?	Duration ?
:clean		0.499s	0.053s
:compileJava		0.553s	0.146s
:processResources	NO-SOURCE	0.699s	0.001s
:classes		0.700s	0.000s
:jar		0.701s	0.040s
:assemble		0.741s	0.000s
:compileTestJava		0.741s	0.242s
:processTestResources	NO-SOURCE	0.984s	0.000s
:testClasses		0.984s	0.001s
:test		0.985s	1m 59.135s
:check		2m 0.120s	0.001s
:build		2m 0.121s	0.001s



Task path		Started after ?	Duration ?
:clean		0.416s	0.048s
:compileJava		0.465s	0.085s
:processResources	NO-SOURCE	0.550s	0.000s
:classes		0.550s	0.000s
:jar		0.551s	0.040s
:assemble		0.591s	0.000s
:compileTestJava		0.592s	0.212s
:processTestResources	NO-SOURCE	0.804s	0.001s
:testClasses		0.805s	0.000s
:test		0.805s	10.553s
:check		11.359s	0.000s
:build		11.359s	0.000s



```
tasks.test { this: Test!
    onlyIf { true }
    useJUnitPlatform()
    maxParallelForks = Runtime.getRuntime().availableProcessors()
    testLogging { this: TestLoggingContainer
```

TEST DISTRIBUTION

- CI uses fan-out to speed-up tests
- Shouldn’t you enjoy it for local tests?
- Use the cloud to distribute test load
- RUN ALL THE TESTS!

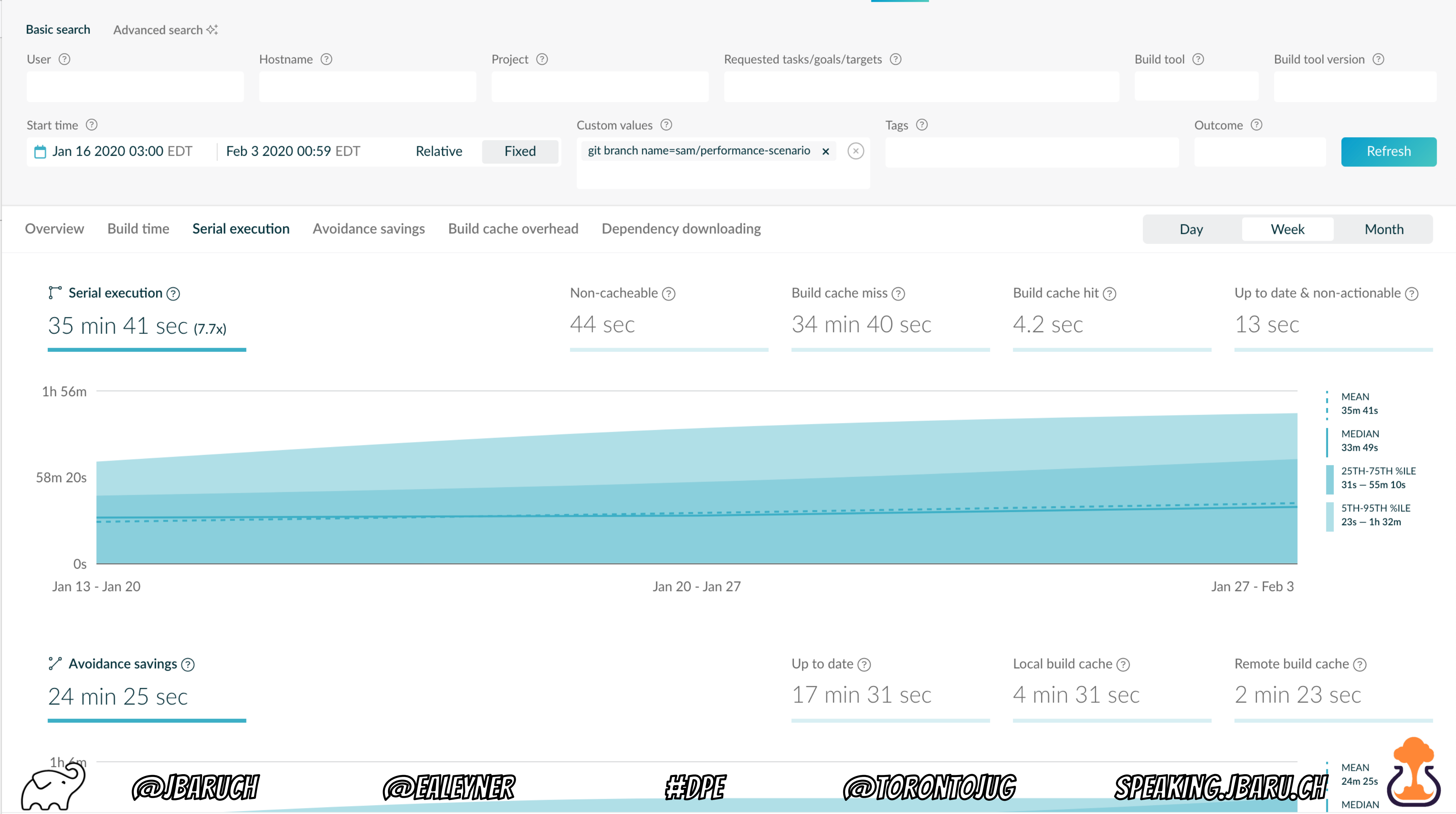
WHY NOT JUST USING CI FAN-OUT?

- Relying on shared CI infrastructure
- CI infrastructure is not optimized for real-time feedback!
- Are the agents as fast as they can be?

DON'T LET IT SLIDE

OBSERVE AND IMPROVE

- Measure local build times across time and environments
- Detect downfacing trends
- Find root causes and improve



Start time: Jan 16 2020 03:00 EDT – Feb 3 2020 00:59 EDT, Fixed

Custom values: git branch name=sam/performance-scenario

Serial execution

Serial execution: 35 min 41 sec (7.7x)

Non-cacheable: 44 sec

Build cache miss: 34 min 40 sec

Build cache hit: 4.2 sec

Up to date & non-actionable: 13 sec

Avoidance savings: 24 min 25 sec

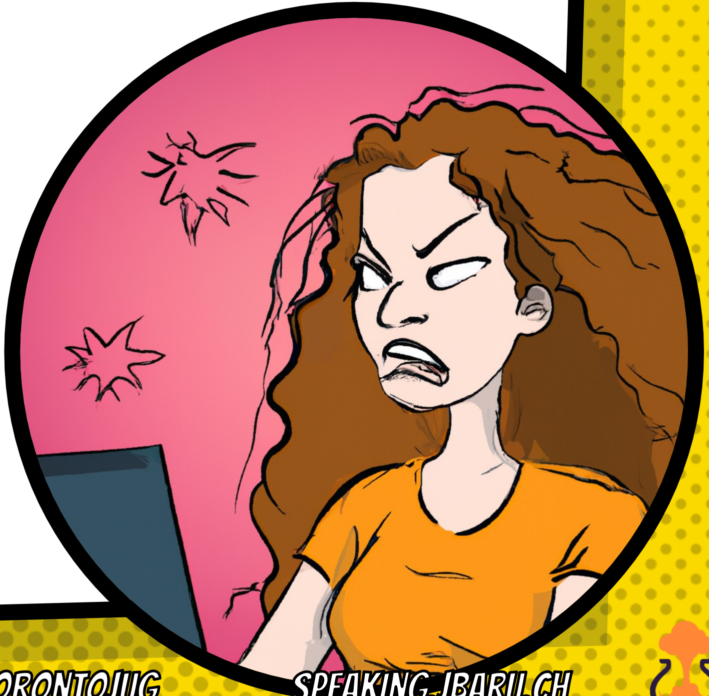
Up to date: 17 min 31 sec

Local build cache: 4 min 31 sec

Remote build cache: 2 min 23 sec

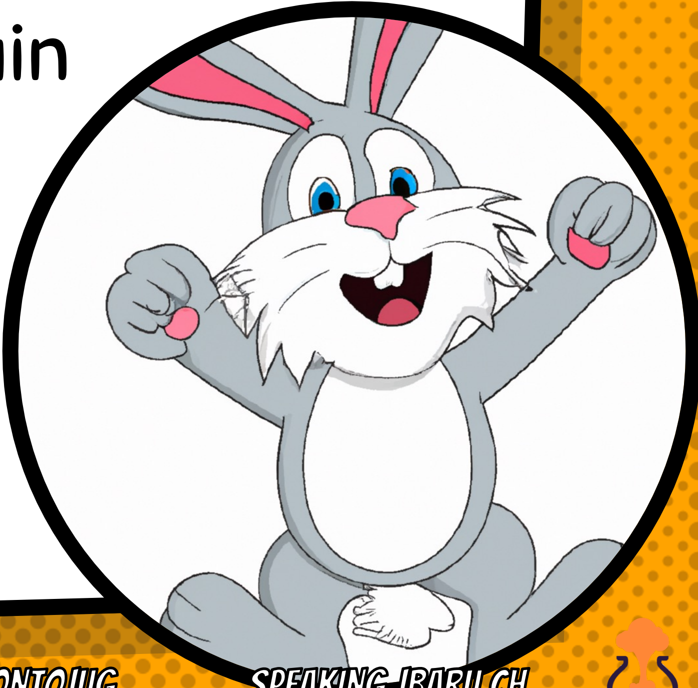
SAME TRADEOFF FOR TESTS

EXPENSIVE, SLOW, BUT PROD-LIKE
OR
NAÏVE, USELESS, BUT FAST



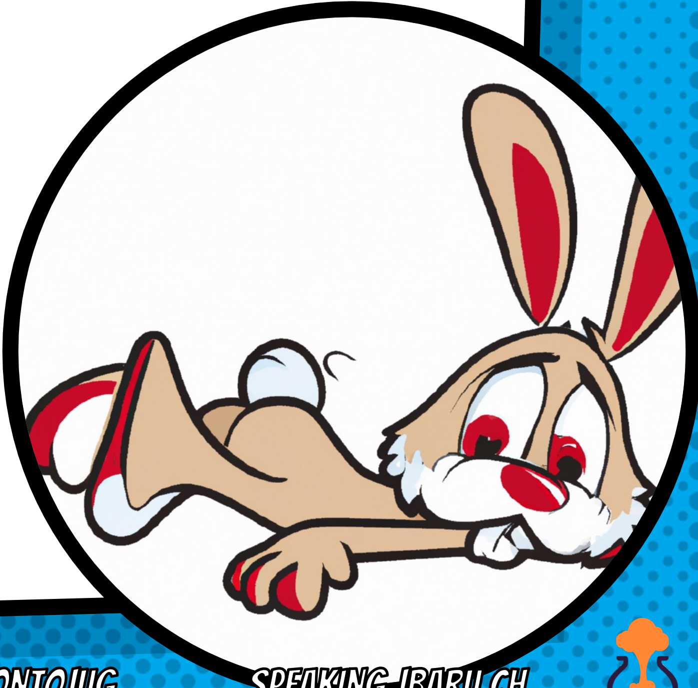
AWS S3 - BUNNY!

- Integration Tests – as main quality gate
- Creative, Relatable test status
- Sad and depressed outcome over time



WHY THE DEPRESSED BUNNY?

- Flakiness due to:
  - Data, Deployments, Environment
  - Don't care syndrome
  - Self-fulfilling prophecies



DELIGHTFUL TEST (PICK TWO):

- ☑ PROD-LIKE
- ☑ LOCAL

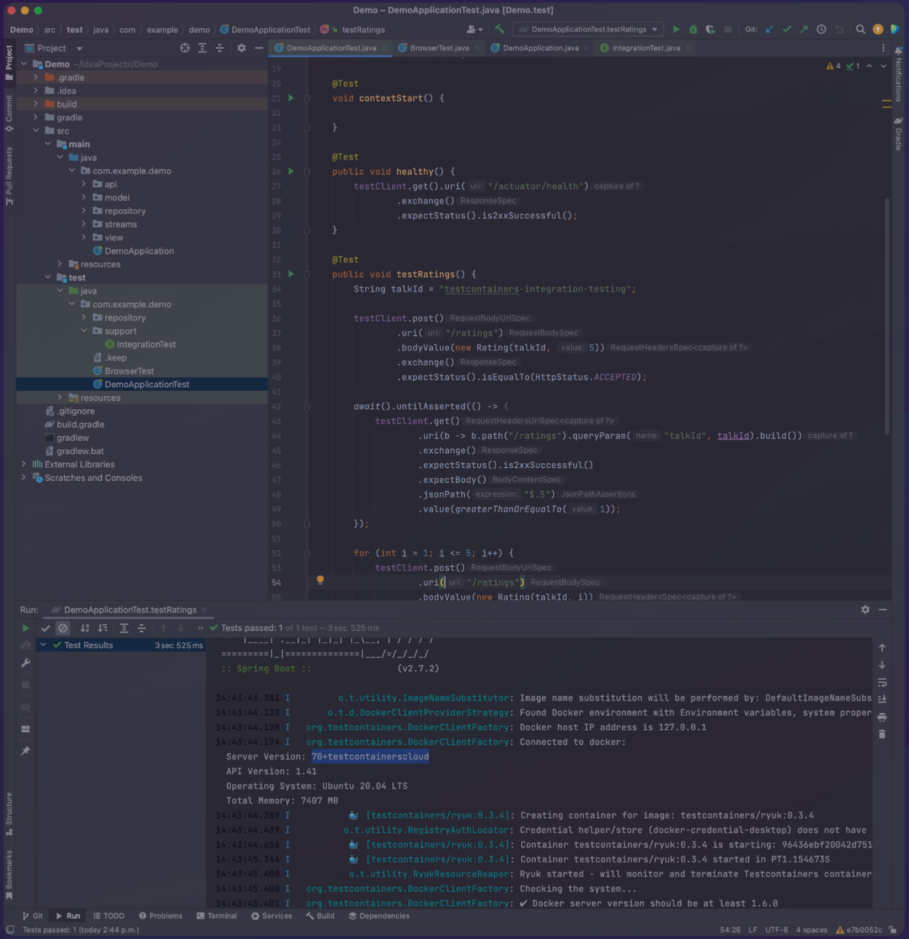
SELF CONTAINED TESTS ARE THE ANSWER

- Real tests (not just mocks)
- No dependency on centralized resources
- No interdependency of tests
- Treated as code! Same life-cycle
- Allow for clone-and-run

EHE.. BUT HOW?

TESTCONTAINERS IS THE ANSWER

DEMO



```java
    @Test
    void contextStart() {

    }

    @Test
    public void healthy() {
        testClient.get().uri("/actuator/health")
                .exchange()
                .expectStatus().is2xxSuccessful();
    }

    @Test
    public void testRatings() {
        String talkId = "testcontainers-integration-testing";

        testClient.post()
                .uri("/ratings")
                .bodyValue(new Rating(talkId, 5))
                .exchange()
                .expectStatus().isEqualTo(HttpStatus.ACCEPTED);

        await().untilAsserted(() -> {
            testClient.get()
                    .uri(b -> b.path("/ratings").queryParam("talkId", talkId).build())
                    .exchange()
                    .expectStatus().is2xxSuccessful()
                    .expectBody()
                    .jsonPath("$.5")
                    .value(greaterThanOrEqualTo(1));
        });

        for (int i = 1; i <= 5; i++) {
            testClient.post()
                    .uri("/ratings")
                    .bodyValue(new Rating(talkId, i))
```

Tests passed: 1 of 1 test – 3 sec 525 ms

```
=========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

14:43:43.381 I        o.t.utility.ImageNameSubstitutor: Image name substitution will be performed by: DefaultImageNameSubs
14:43:44.123 I      o.t.d.DockerClientProviderStrategy: Found Docker environment with Environment variables, system proper
14:43:44.128 I  org.testcontainers.DockerClientFactory: Docker host IP address is 127.0.0.1
14:43:44.174 I  org.testcontainers.DockerClientFactory: Connected to docker:
  Server Version: 70+testcontainerscloud
  API Version: 1.41
  Operating System: Ubuntu 20.04 LTS
  Total Memory: 7407 MB
14:43:44.289 I         🐳 [testcontainers/ryuk:0.3.4]: Creating container for image: testcontainers/ryuk:0.3.4
14:43:44.439 I         o.t.utility.RegistryAuthLocator: Credential helper/store (docker-credential-desktop) does not have
14:43:44.658 I         🐳 [testcontainers/ryuk:0.3.4]: Container testcontainers/ryuk:0.3.4 is starting: 96436ebf20042d751
14:43:45.344 I         🐳 [testcontainers/ryuk:0.3.4]: Container testcontainers/ryuk:0.3.4 started in PT1.154673S
14:43:45.400 I         o.t.utility.RyukResourceReaper: Ryuk started - will monitor and terminate Testcontainers container
14:43:45.400 I  org.testcontainers.DockerClientFactory: Checking the system...
14:43:45.401 I  org.testcontainers.DockerClientFactory: ✔ Docker server version should be at least 1.6.0
```

Tests passed: 1 (today 2:44 p.m.)

TESTCONTAINERS IS USED BY

Spotify

Uber

NETFLIX

JET
BRAINS

neo4j

Skyscanner

WISE

elastic

Capital One

INSTANA
an IBM Company

zalando

Playtika

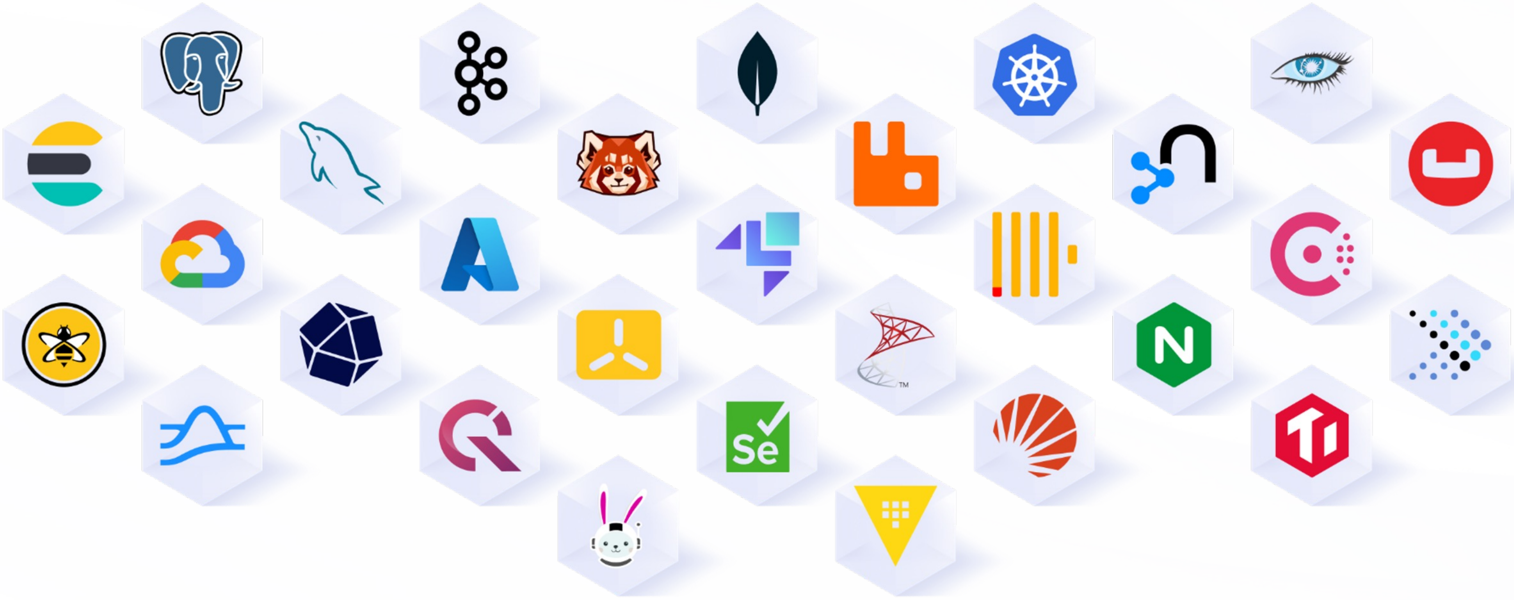
Testcontainers

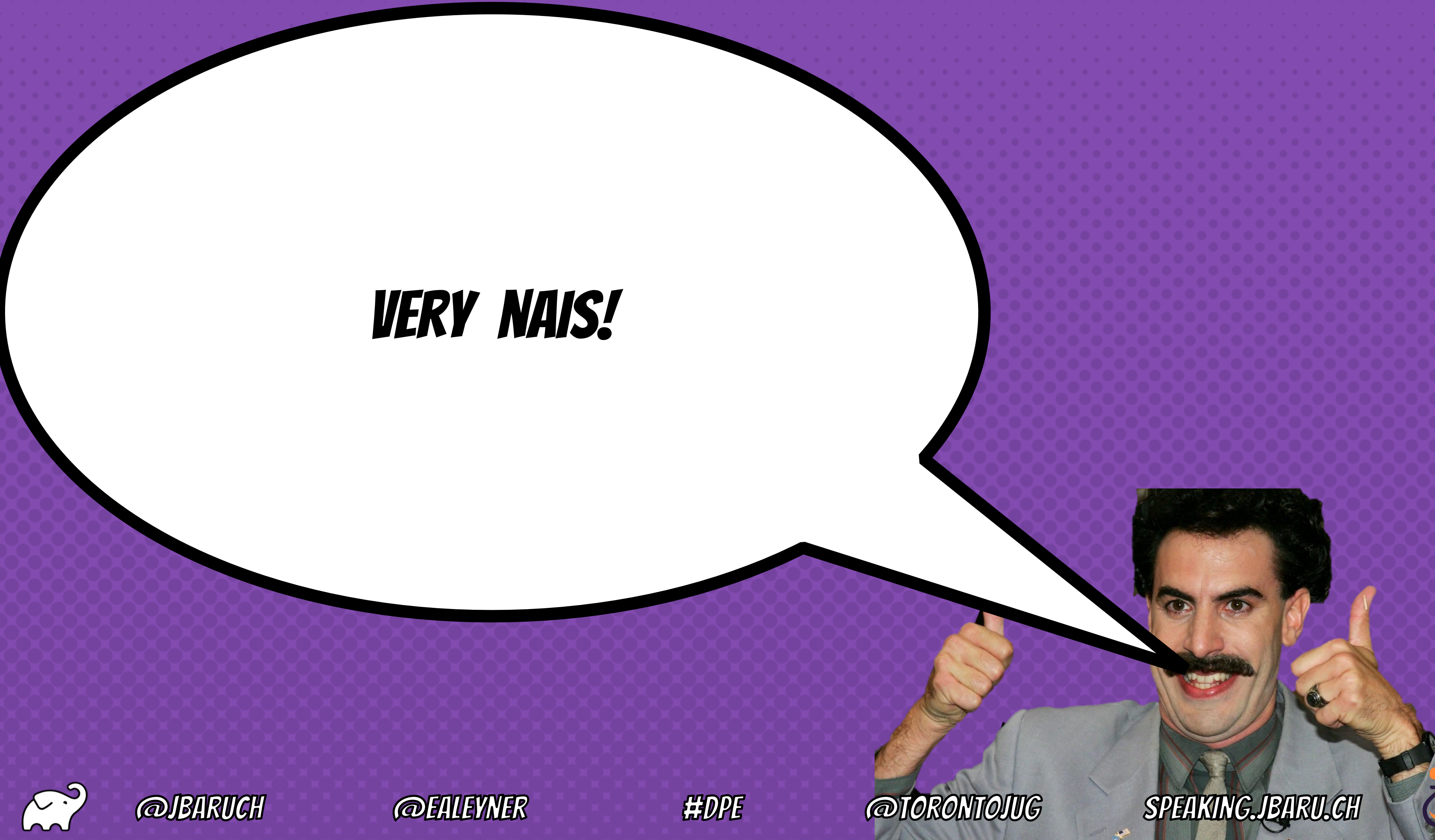
WE ❤ OPEN SOURCE

& EMPLOY MAINTAINERS

FOR YOUR ENTIRE STACK

Test against any database, message broker, browser...
or just about anything that runs in a Docker container!




VERY NAIS!

BUT WE CAN GO FURTHER!

Foster Faster Feedback

Collaborate through Effective Tooling

Embrace Rigorous Observability for Proactive Improvement

Eliminate Toil for Developers

Prioritize Automation and Eliminate Bottlenecks

Dedicated Organizational Mindset

Outcomes Over Output

TEST OBSERVABILITY & DEBUGGING

Eliminate Toil for Developers

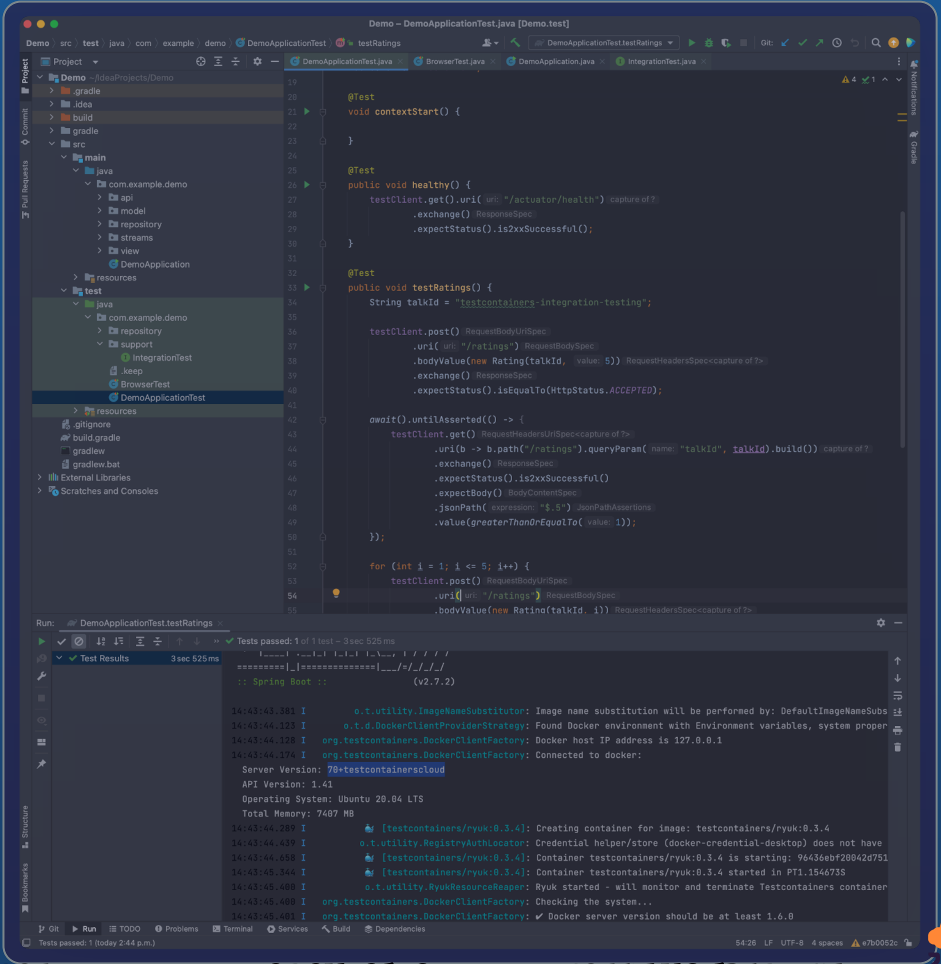
Embrace Rigorous Observability for Proactive Improvement

Outcomes Over Output

TESTCONTAINERS DESKTOP: BATTERIES INCLUDED

- Freeze container shutdown
- Service configuration
- Enables re-usable containers
- Bonus: Embedded Runtime

DEMO



```java
    @Test
    void contextStart() {

    }

    @Test
    public void healthy() {
        testClient.get().uri("/actuator/health")
                .exchange()
                .expectStatus().is2xxSuccessful();
    }

    @Test
    public void testRatings() {
        String talkId = "testcontainers-integration-testing";

        testClient.post()
                .uri("/ratings")
                .bodyValue(new Rating(talkId, 5))
                .exchange()
                .expectStatus().isEqualTo(HttpStatus.ACCEPTED);

        await().untilAsserted(() -> {
            testClient.get()
                    .uri(b -> b.path("/ratings").queryParam("talkId", talkId).build())
                    .exchange()
                    .expectStatus().is2xxSuccessful()
                    .expectBody()
                    .jsonPath("$.5")
                    .value(greaterThanOrEqualTo(1));
        });

        for (int i = 1; i <= 5; i++) {
            testClient.post()
                    .uri("/ratings")
                    .bodyValue(new Rating(talkId, i))
```

```
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

14:43:43.381 I        o.t.utility.ImageNameSubstitutor: Image name substitution will be performed by: DefaultImageNameSubs
14:43:44.123 I      o.t.d.DockerClientProviderStrategy: Found Docker environment with Environment variables, system proper
14:43:44.128 I  org.testcontainers.DockerClientFactory: Docker host IP address is 127.0.0.1
14:43:44.174 I  org.testcontainers.DockerClientFactory: Connected to docker:
  Server Version: 70+testcontainerscloud
  API Version: 1.41
  Operating System: Ubuntu 20.04 LTS
  Total Memory: 7407 MB
14:43:44.289 I         🐳 [testcontainers/ryuk:0.3.4]: Creating container for image: testcontainers/ryuk:0.3.4
14:43:44.439 I        o.t.utility.RegistryAuthLocator: Credential helper/store (docker-credential-desktop) does not have
14:43:44.658 I         🐳 [testcontainers/ryuk:0.3.4]: Container testcontainers/ryuk:0.3.4 is starting: 96436ebf20042d751
14:43:45.344 I         🐳 [testcontainers/ryuk:0.3.4]: Container testcontainers/ryuk:0.3.4 started in PT1.154673S
14:43:45.400 I         o.t.utility.RyukResourceReaper: Ryuk started - will monitor and terminate Testcontainers container
14:43:45.400 I  org.testcontainers.DockerClientFactory: Checking the system...
14:43:45.401 I  org.testcontainers.DockerClientFactory: ✔ Docker server version should be at least 1.6.0
```

TEST CONSISTENCY, REPEATABILITY & SCALE

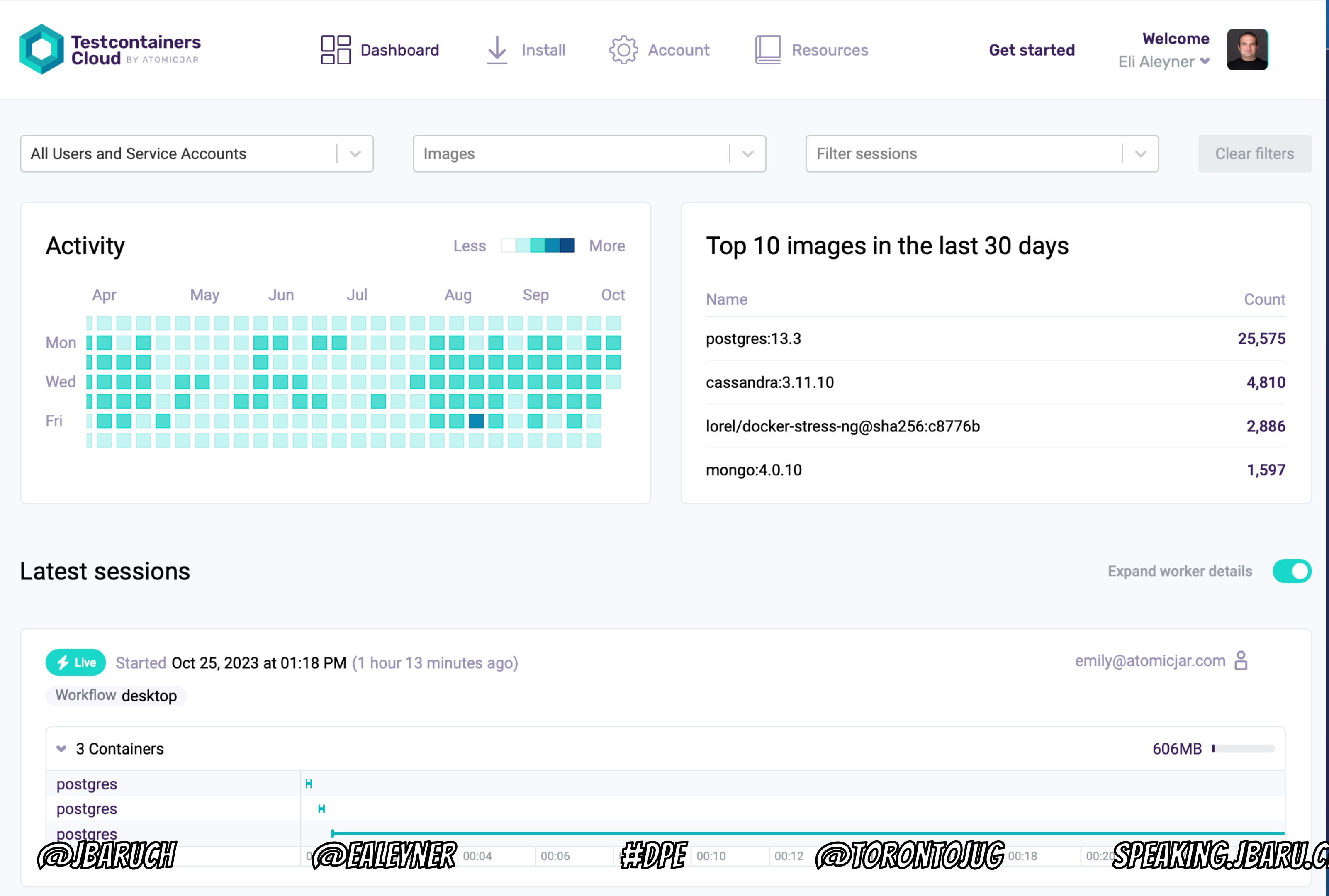
Dedicated Organizational Mindset

Prioritize Automation and Eliminate Bottlenecks

Outcomes Over Output

TESTCONTAINERS CLOUD: CONSISTENT EXPERIENCE FOR ALL

- Apple-silicon? Windows? – No problem!
- Docker-in-Docker in CI? – No problem!
- Turbo mode – for parallelization
- Team and enterprise-wide visibility for your Testcontainers usage



Activity

Top 10 images in the last 30 days

Name	Count
postgres:13.3	25,575
cassandra:3.11.10	4,810
lorel/docker-stress-ng@sha256:c8776b	2,886
mongo:4.0.10	1,597

Latest sessions

Started Oct 25, 2023 at 01:18 PM (1 hour 13 minutes ago)

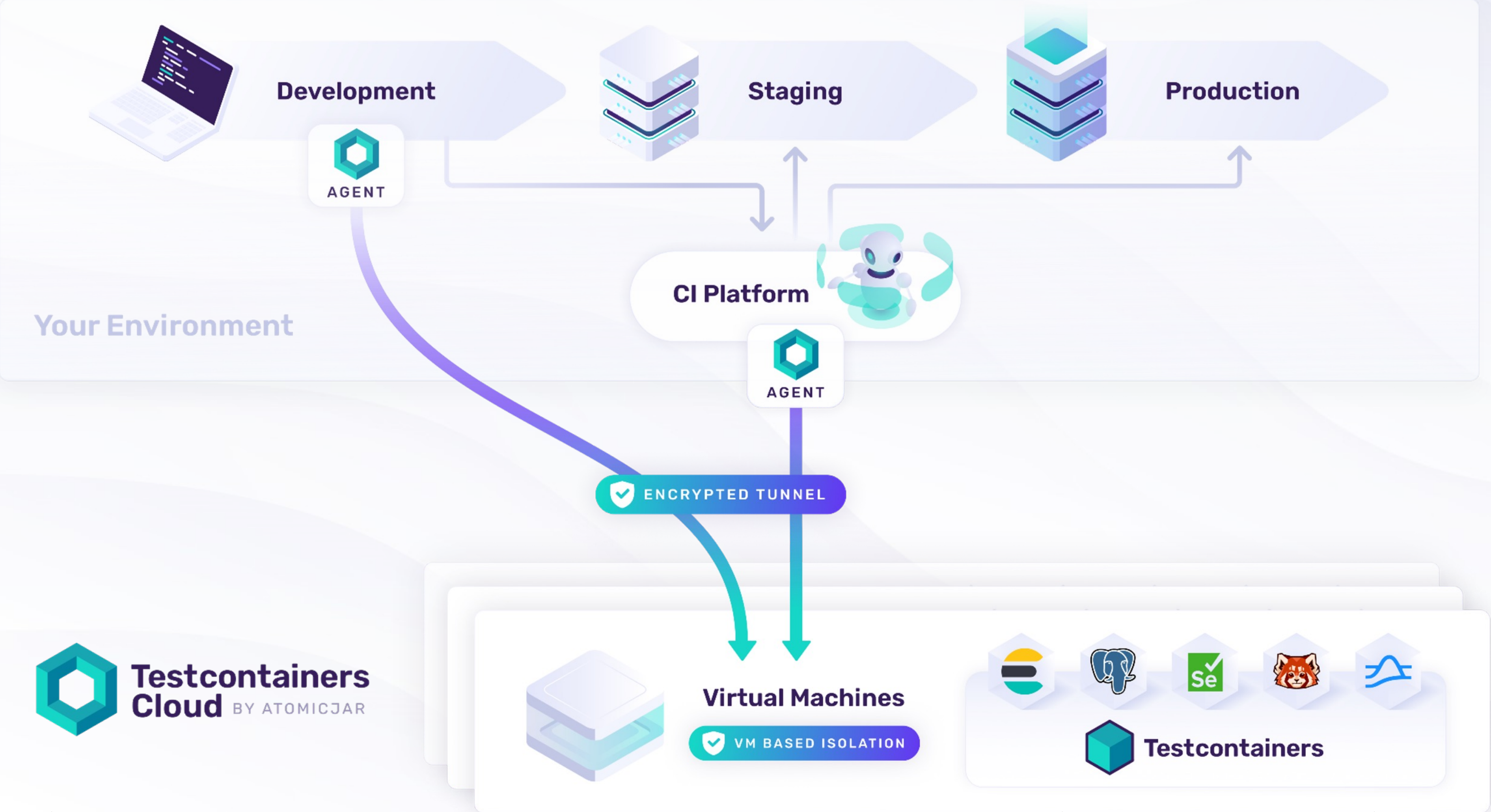
emily@atomicjar.com

Workflow desktop

3 Containers 606MB

postgres
postgres
postgres

HOW IT WORKS



Testcontainers Cloud BY ATOMICJAR

TESTCONTAINERS IS DPE:

Foster Faster Feedback

Collaborate through Effective Tooling

Embrace Rigorous Observability for Proactive Improvement

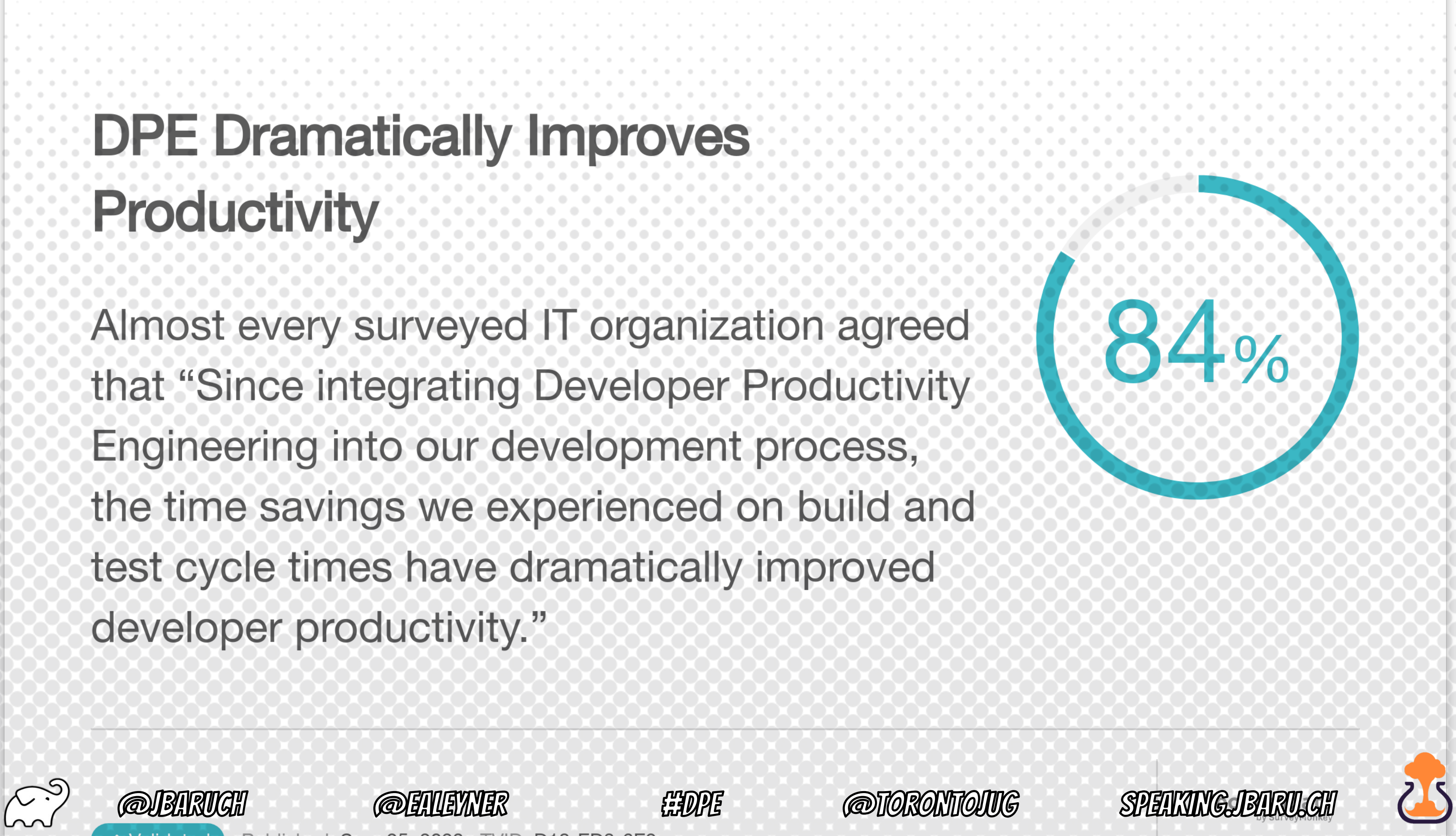
Eliminate Toil for Developers

Prioritize Automation and Eliminate Bottlenecks

Dedicated Organizational Mindset

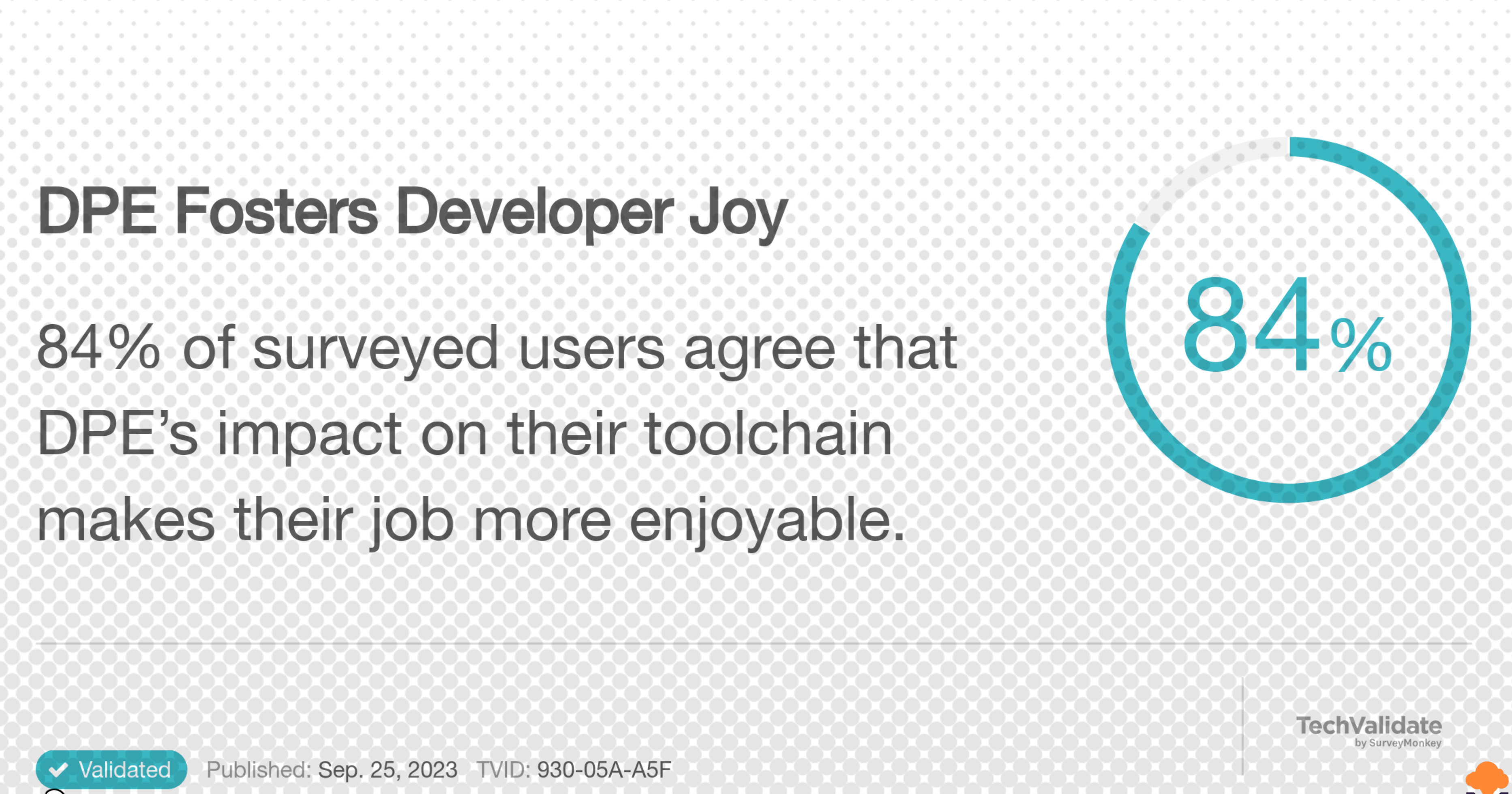
Outcomes Over Output

THE GAINS ARE REAL!



DPE Dramatically Improves Productivity

Almost every surveyed IT organization agreed that “Since integrating Developer Productivity Engineering into our development process, the time savings we experienced on build and test cycle times have dramatically improved developer productivity.”



DPE Fosters Developer Joy

84% of surveyed users agree that DPE’s impact on their toolchain makes their job more enjoyable.

✓ Validated Published: Sep. 25, 2023 TVID: 930-05A-A5F

TechValidate
by SurveyMonkey

LEARN MORE AND TRY IT TODAY!

- Take the Gradle/Maven Speed Challenge!
- Be DPE Agent of Change!
- Read the DPE Handbook!
- Watch the DPE Summit videos!

Scan me

SPEAKING.JBARU.CH

Q&A AND SOCIAL ADS

- @JBARUCH
- @EALYENER
- @TORONTOJUG
- #DPE
- SPEAKING.JBARU.CH