

Gaining Actionable Cross-Project Build Insights

From Experience to Implementation



Gradle

DPE Summit 2024



Who we are



Laurent Ploix

Senior Product Manager



Etienne Studer

SVP of Engineering &
Develocity Co-Founder

Agenda



Data-informed Development
Use cases

Collection and Revelation of Build Data
A journey

Data Informed Development

Knowledge

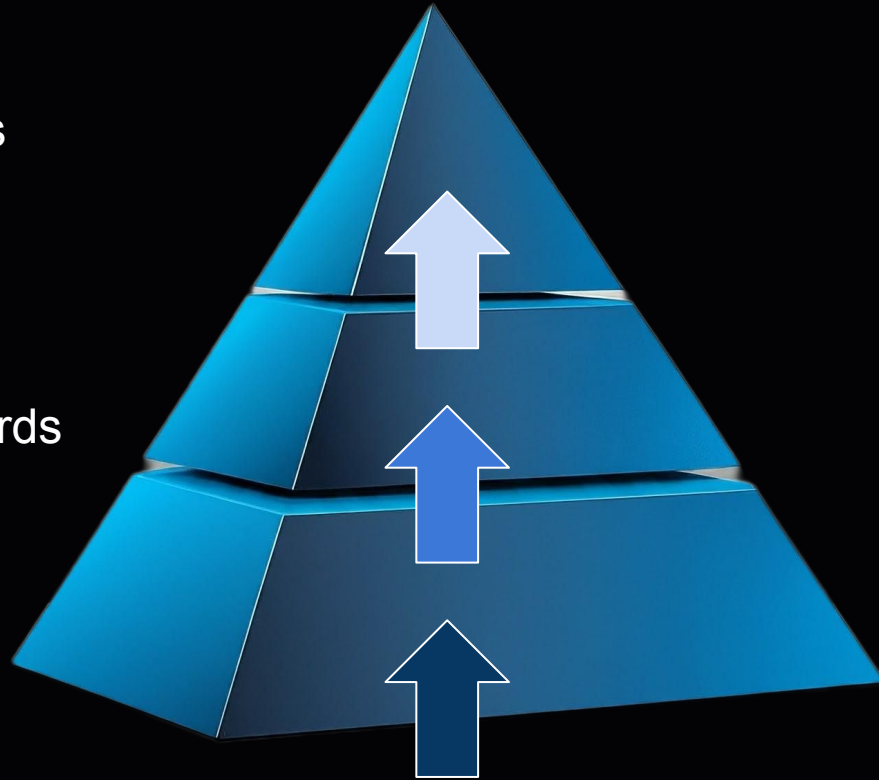
- Actionable Insights
- Reports

Information

- Metrics
- Charts & Dashboards

Data

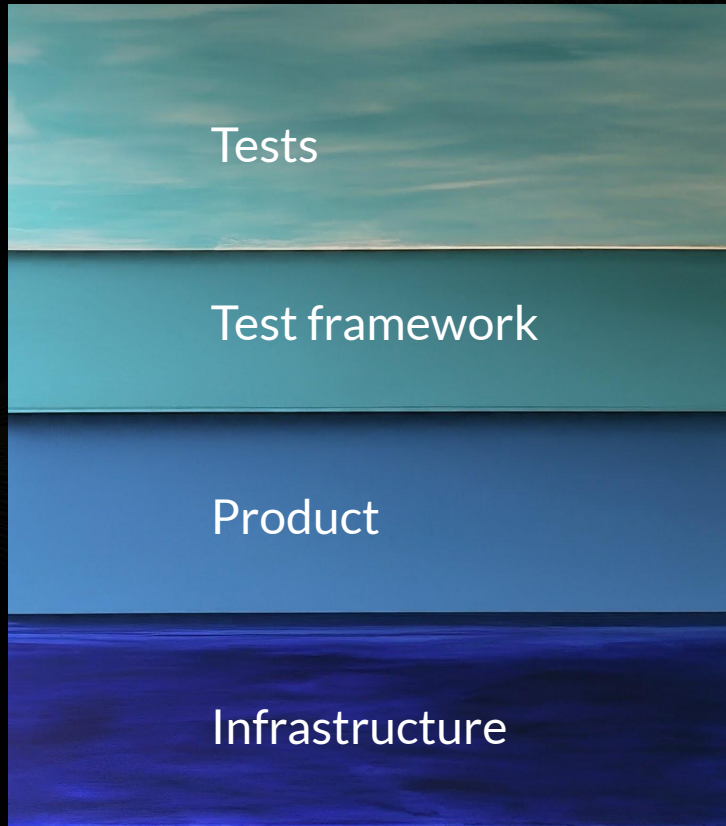
- Query-able
- Raw



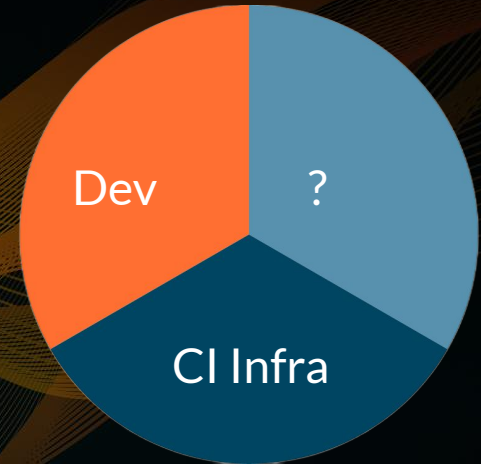
CI Failures

What to focus on?

Why does it fail?



Who gets to fix it?



Knowledge

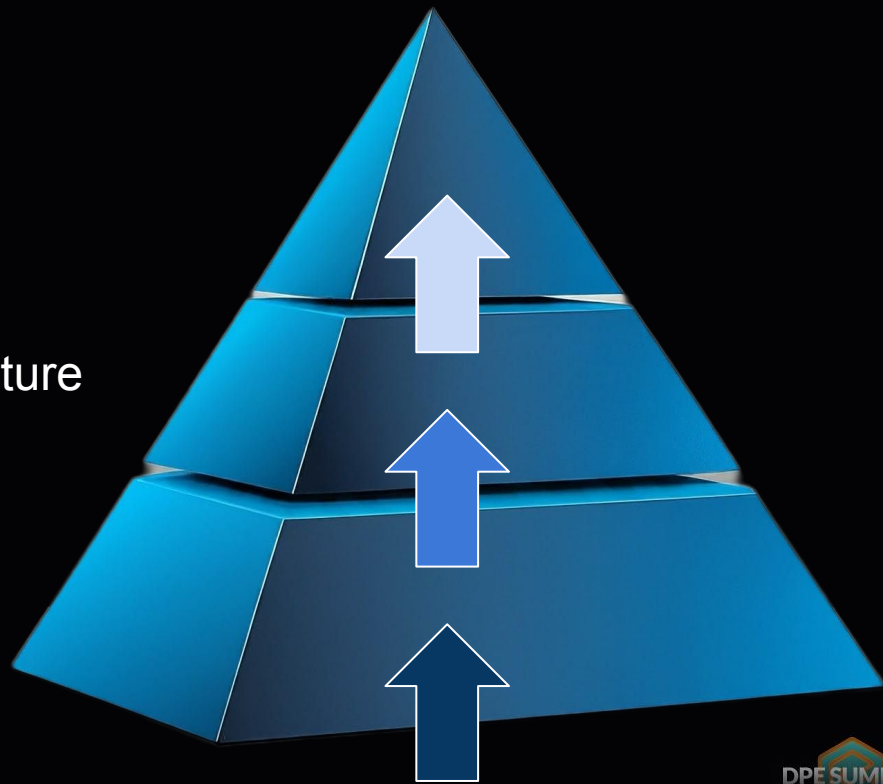
- What to fix, by which team

Information on Flaky Tests

- Test / Framework / Product / Infrastructure

Data:

- Test results



Faster machines What could possibly go wrong?

Because... humans

1. Long queues in CI
2. Double the speed of the CI machines
3. CI queues disappear
4. Wait 2 weeks
5. Get even longer CI queues
6. ...
7. ???



Knowledge

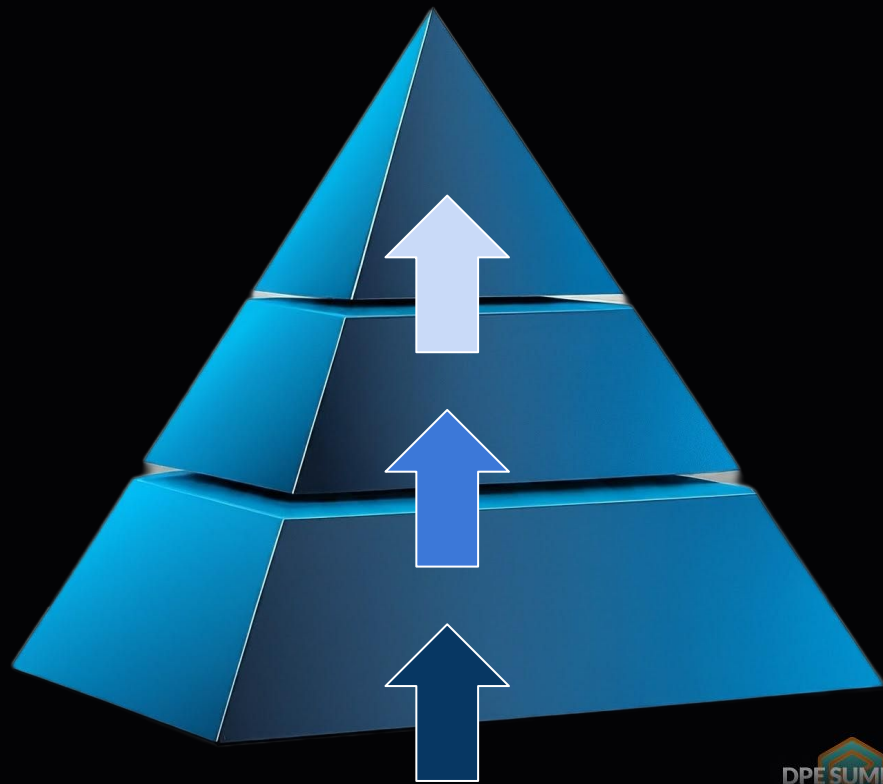
- Developers run their builds in CI because it's faster

Information

- More CI builds
- Fewer local builds

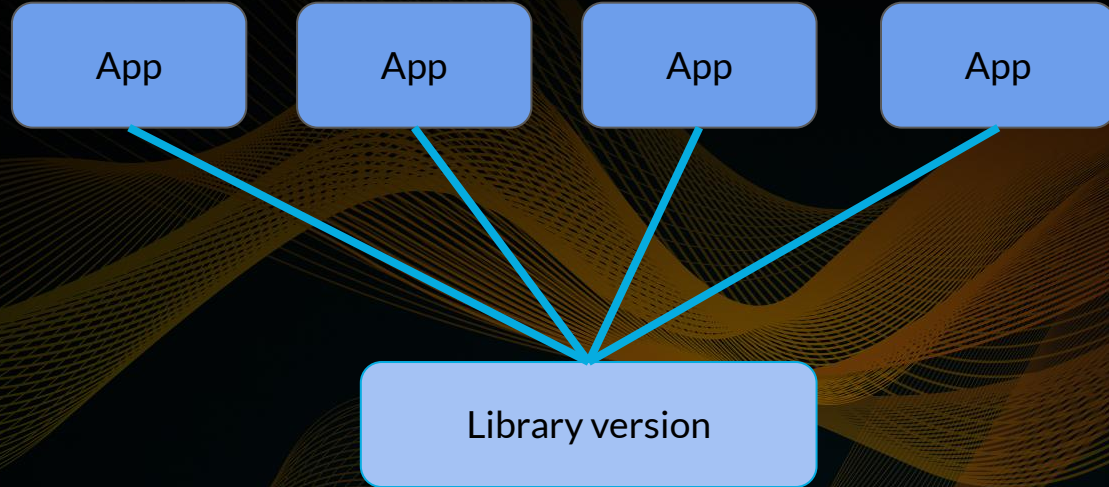
Data:

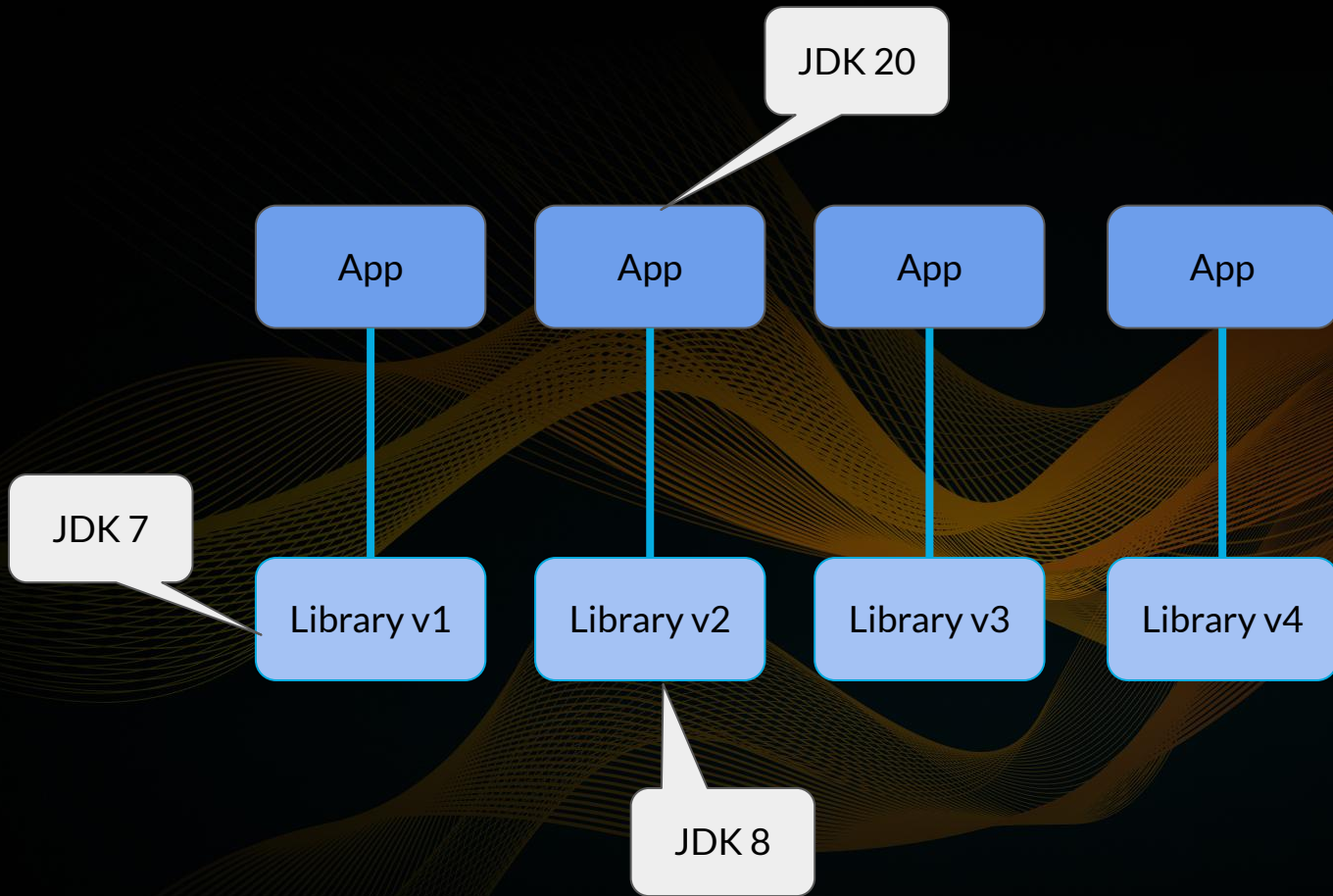
- CI builds
- Local builds

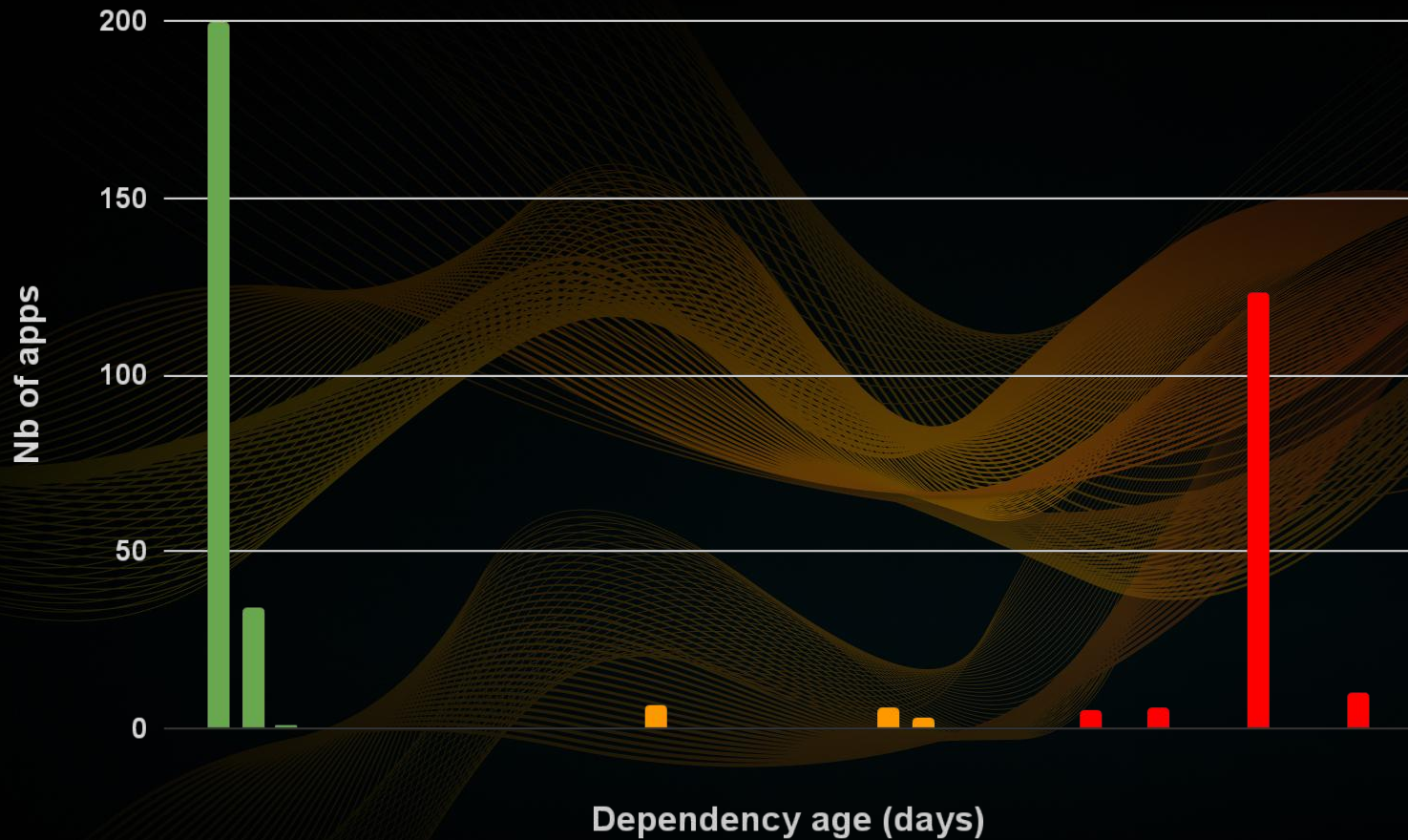


Large Migrations

Are we there yet?







Knowledge

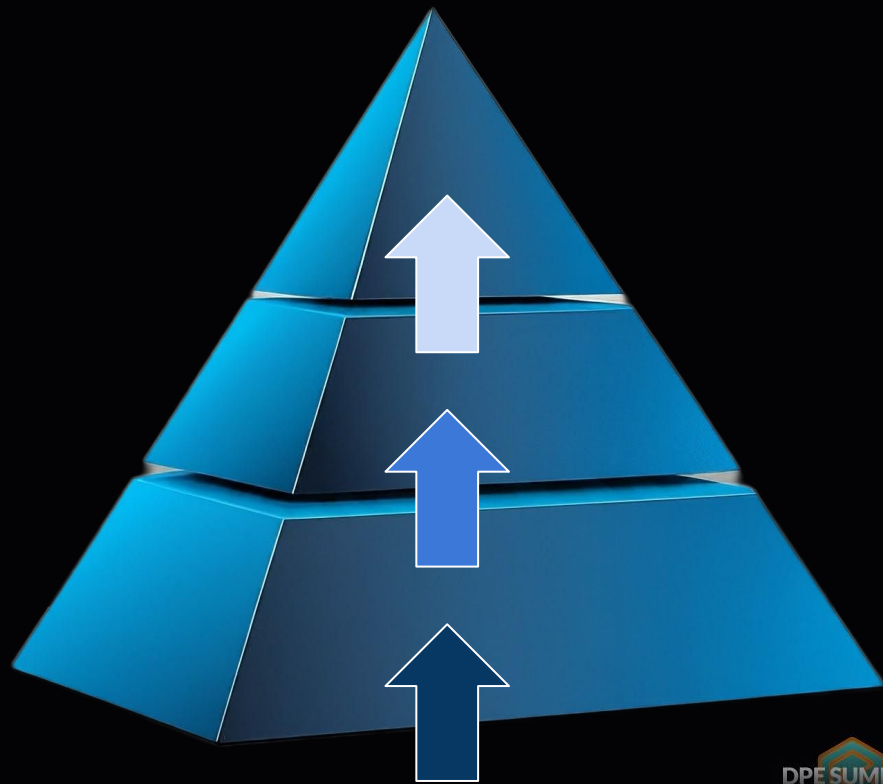
- Which dependency to update
- Where to update the Build Platform

Information

- All apps, all dependencies
- All apps, all Toolchain elements

Data:

- Source code
- Build configurations
- Build logs
- SBOM



Many repos to optimize

Navigating the Build jungle

2 repos x **30 minutes** x 1000 builds/day

=

2K repos x **3 minutes** x 10 builds/day

Toolchain Observability

Across all projects



Agenda

Data-informed Development
Use cases



Collection and Revelation of Build Data
A journey

**Comprehensive build data is
the foundation for developer
toolchain observability**

Develocity build plugins eavesdrop what happens while the build is running

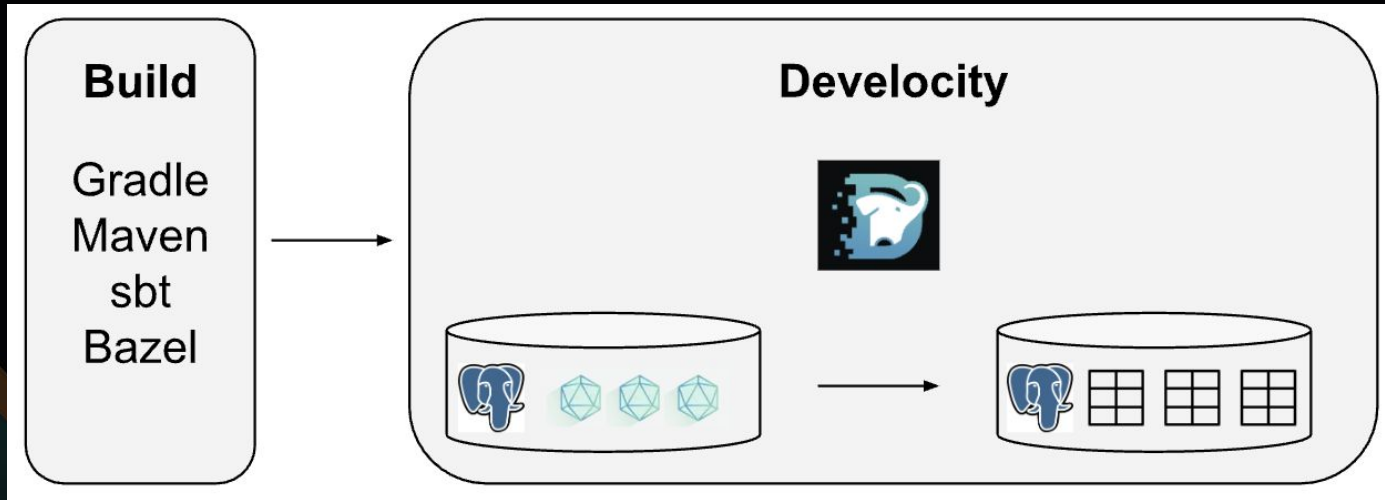
```
plugins {  
    id("com.gradle.develocity") version "3.18.1"  
}  
  
develocity {  
    server = "https://develocity.company.net"  
}
```

```
./gradlew build --init-script develocity-init.gradle
```

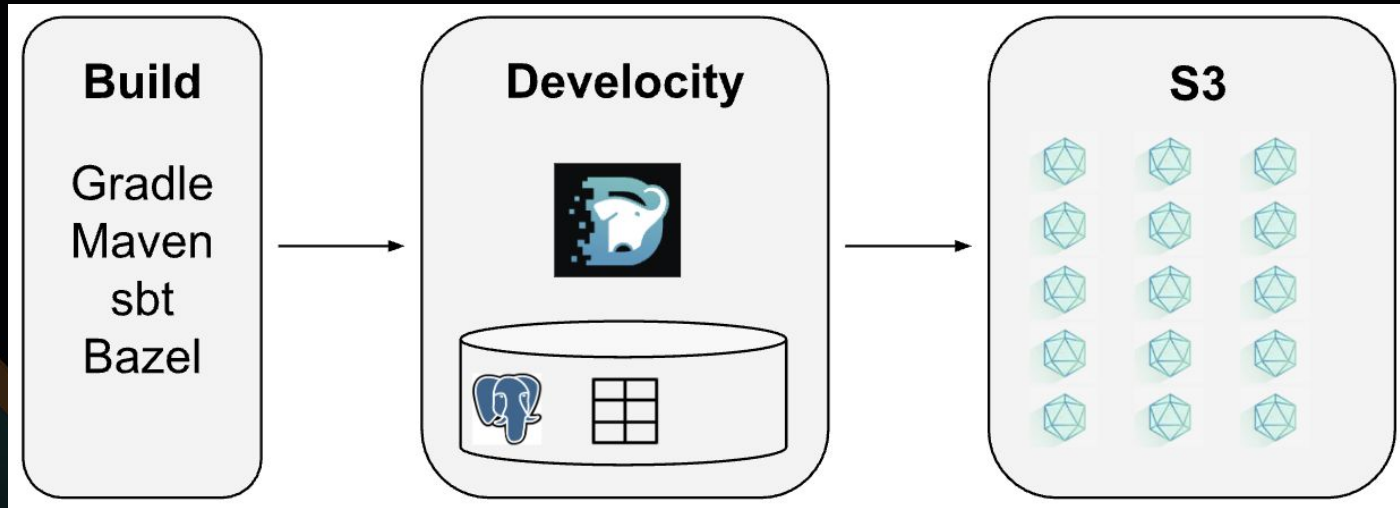
Build progress is tracked in the form of fine-grained build events

```
public class TaskStarted_1_4 extends TaskStarted_1_3 {  
  
    public final String buildPath;  
  
    public TaskStarted_1_4(long id, String path, String className,  
        @Nullable Long thread, String buildPath,  
    ) {  
        super(id, path, className, thread);  
        this.buildPath = buildPath;  
    }  
  
    ...  
}
```

Build events are uploaded to the Develocity server and persisted as a blob in S3-compatible storage



Build events are uploaded to the Develocity server and persisted as a blob in S3-compatible storage



A single build is visualized as a Build Scan by on-demand converting the build events into a Frontend build model

The screenshot displays the DEV ELOCITY build scan interface for a build of 'spring-boot-build' on Sep 24, 2024, at 14:16:59 PDT. The interface is divided into a left sidebar and a main content area. The sidebar contains navigation options: Summary, Console log, Failure, Deprecations, Timeline, Performance, Tests (highlighted), Projects, Dependencies, Build dependencies, Plugins, Custom values, Switches, and Infrastructure. Below these are 'See before and after' and 'Compare Build Scan' options. The main content area shows an overview of the build scan with the following summary: '17923 tests executed in 2430 test classes taking 2h 25m 45.402s serial time and 1h 19m 53.923s wall-clock time, 1 flaky execution'. Below this is a table of test results.

Test	Outcome	Total time
Showing 1-200 out of 17923 total items		
runCreatesConnectionDetailsThatCanBeUsedToAccessDatabase(JdbcConnectionDetails)	PASSED	1m 40.267s
runWithBitnamimagenameCreatesConnectionDetails(CassandraConnectionDetails)	PASSED	1m 31.865s
whenAotRunsFromClassProxyClassesAreCopiedToTargetClasses(MavenBuild)[1]	PASSED	1m 22.957s
runCreatesConnectionDetailsThatCanBeUsedToAccessDatabase(JdbcConnectionDetails)	PASSED	1m 9.165s
runCreatesConnectionDetails(PulsarConnectionDetails)	PASSED	1m 1.057s
launchWithSingleCommandLineArgument(String, String)[1]	PASSED	57.886s
typicalPluginsAppliesExceptedPlugins[2]	PASSED	57.492s
zip64JarThatExceedsZipSizeLimitCanBeRead()	PASSED	54.886s
runWithBitnamimagenameCreatesConnectionDetailsThatCanAccessNeo4j(Neo4jConnectionDetails)	PASSED	53.473s

A single build is visualized as a Build Scan by on-demand converting the build events into a Frontend build model

DEV ELOCITY ✓ spring-boot-build build Sep 24 2024 14:18:05 PDT

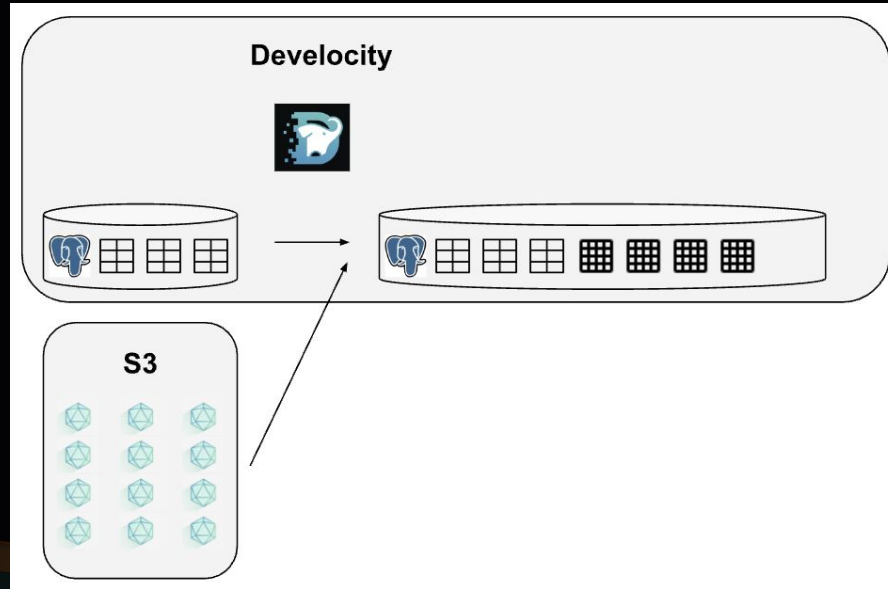
Summary
Console log
Failure
Deprecations
Timeline
Performance
Tests
Projects
Dependencies
Build dependencies
Plugins
Custom values
Switches
Infrastructure

See before and after
Compare Build Scan

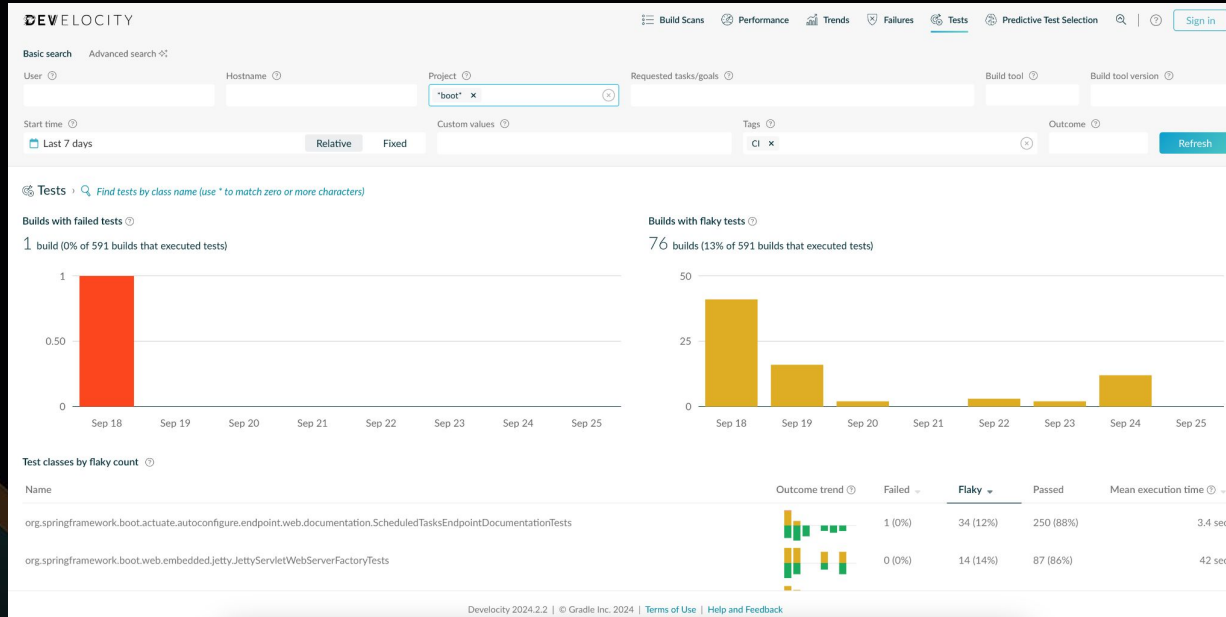
1253 dependencies resolved in 187 projects across 1359 configurations from 5 repositories

```
.buildSrc >
spring-boot-projects-spring-boot >
spring-boot-projects-spring-boot-actuator >
spring-boot-projects-spring-boot-actuator-autoconfigure >
spring-boot-projects-spring-boot-autoconfigure >
annotationProcessor > 0.022s
checkstyle > 0.021s
compileClasspath > 0.171s
dockerTestAnnotationProcessor 0.000s
dockerTestCompileClasspath > 0.157s
dockerTestRuntimeClasspath > 0.104s
runtimeClasspath > 0.548s
testAnnotationProcessor 0.001s
testCompileClasspath > 0.181s
spring-boot-projects-spring-boot >
spring-boot-projects-spring-boot-parent [enforcedApiElements] > platform
spring-boot-projects-spring-boot-test >
spring-boot-projects-spring-boot-tools:spring-boot-test-support >
ch.qos.logback:logback-classic: → 1.5.8 >
co.elastic.clients:elasticsearch-java: → 8.15.1 >
com.fasterxml.jackson.core:jackson-databind: → 2.17.2 >
com.fasterxml.jackson.dataformat:jackson-dataformat-cbor: → 2.17.2 >
com.fasterxml.jackson.dataformat:jackson-dataformat-xml: → 2.17.2 >
com.fasterxml.jackson.datatype:jackson-datatype-jsr310: → 2.17.2 >
com.fasterxml.jackson.module:jackson-module-jakarta-xmlbind-annotations: → 2.17.2 >
com.fasterxml.jackson.module:jackson-module-parameter-names: → 2.17.2 >
com.github.ben-manes.caffeine:caffeine: → 3.1.8 >
com.github.h-thurrow:simple-jndi: → 0.23.0 >
com.github.mxab.thymeleaf-extras:thymeleaf-extras-data-attribute: → 2.0.1 >
com.google.code.gson:gson: → 2.11.0 >
com.h2database:h2: → 2.3.232 >
com.hazelcast:hazelcast-spring: → 5.5.0 >
com.hazelcast:hazelcast: → 5.5.0 >
com.ibm.db2jcc: → 11.5.9.0 >
com.jayway.jsonpath:json-path: → 2.9.0 >
com.mysql:mysql-connector-j: → 9.0.0 >
com.nimbusds:oauth2-oidc-sdk: → 9.43.4 >
com.oracle.database.jdbc:jdbc-dt: → 23.5.0.24.07 >
```

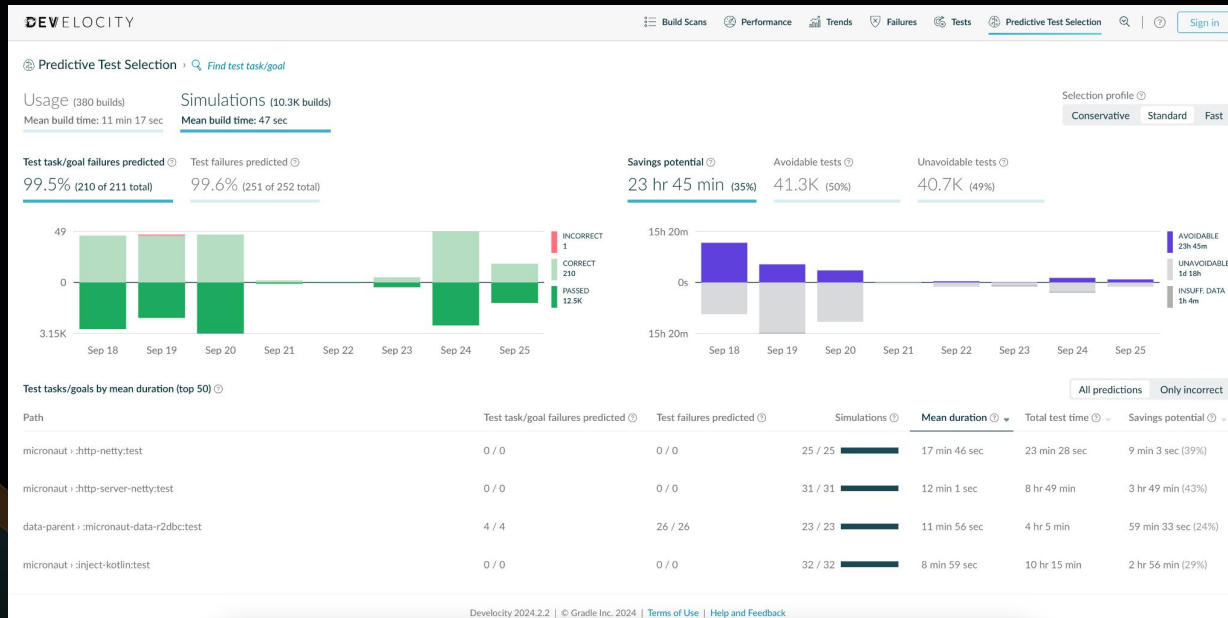
Highly specialized Develocity cross-build dashboards are backed by projection tables in the DB



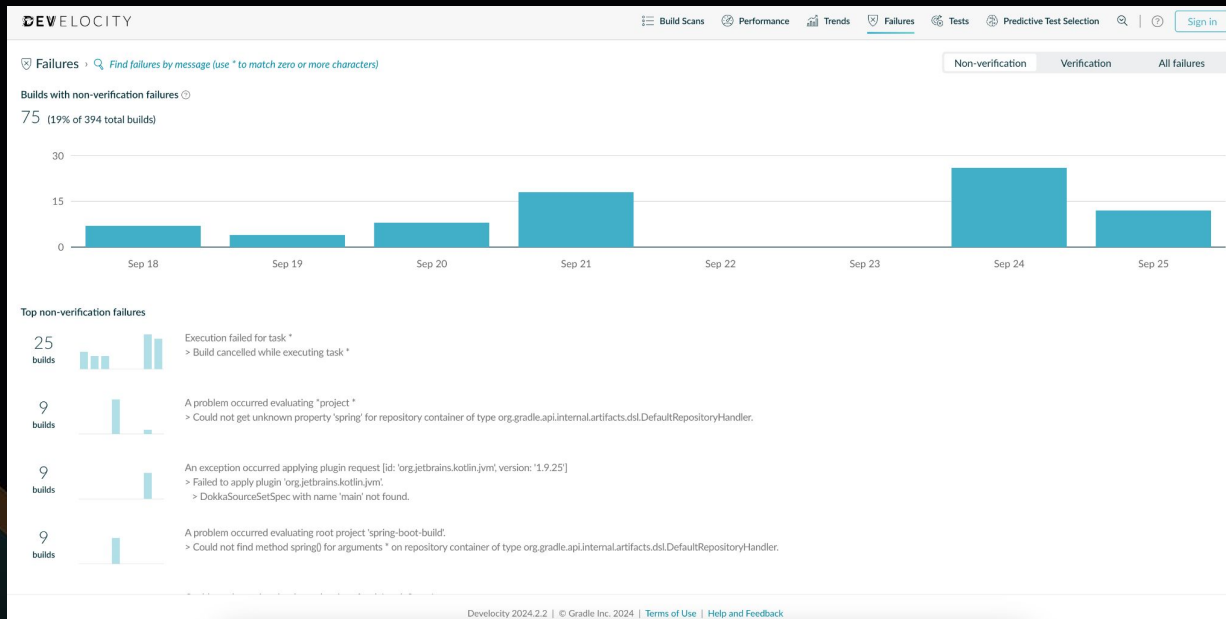
Highly specialized Develocity cross-build dashboards are backed by projection tables in the DB



Highly specialized Develocity cross-build dashboards are backed by projection tables in the DB



Highly specialized Develocity cross-build dashboards are backed by projection tables in the DB



**More questions can be asked
about the build data than can be
provided with built-in dashboards**

The fine-grained build events can be consumed via API in their raw format

/build-export/v2/build/dcetsxpvueltk/events

```
id: dcetsxpvueltk
event: Build
data:
{"toolType":"maven","agentVersion":"1.22.1","toolVersion":"3.9.9","buildId":"dcetsxpvueltk","timestamp":1727267831018}

id: 0
event: BuildEvent
data:
{"timestamp":1727267724835,"type":{"majorVersion":1,"minorVersion":0,"eventType":"MvnBuildStarted"},"data":{}}

id: 1
event: BuildEvent
data:
{"timestamp":1727267724835,"type":{"majorVersion":1,"minorVersion":0,"eventType":"MvnHardware"},"data":{"numProcessors":8}}
```

The build data can be consumed via API as higher-level build models that provide specific views on the data

`/api/builds/uywqwqs7bhhqc/gradle-attributes`

```
{
  "id": "uywqwqs7bhhqc",
  "buildStartTime": 1727268163248,
  "buildDuration": 27285,
  "gradleVersion": "8.10.2",
  "pluginVersion": "3.18.2-rc-1",
  "rootProjectName": "dv",
  "requestedTasks": [
    ":lib-common:verify"
  ],
  "hasFailed": false,
  "tags": [
    "CI",
    "base_release",
    "Linux",
    "Release"],
  ...
}
```

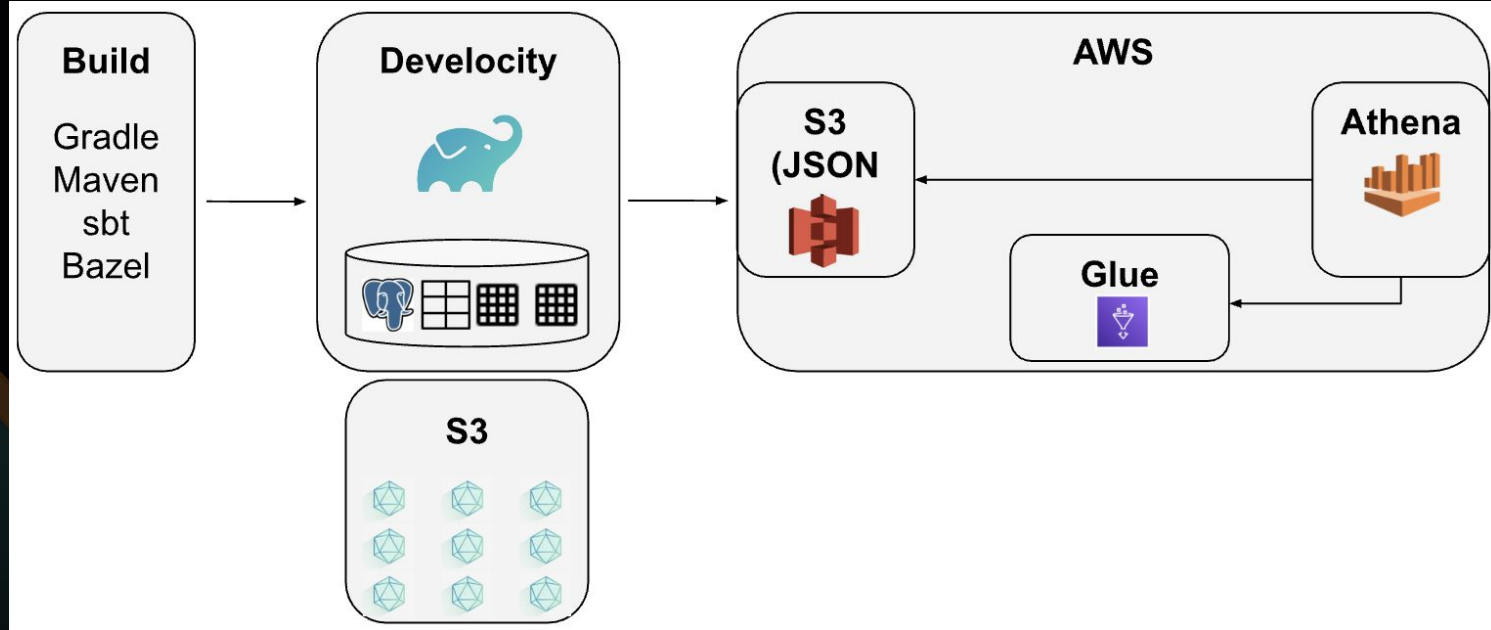
```
"buildOptions": {
  "buildCacheEnabled": true,
  "configurationCacheEnabled": false,
  "configurationOnDemandEnabled": true,
  "continuousBuildEnabled": false,
  "continueOnFailureEnabled": true,
  "daemonEnabled": true,
  "dryRunEnabled": false,
  "excludedTasks": [],
  "fileSystemWatchingEnabled": true,
  "isolatedProjectsEnabled": false,
  "maxNumberOfGradleWorkers": 4,
  "offlineModeEnabled": false,
  "parallelProjectExecutionEnabled": true,
  "refreshDependenciesEnabled": false,
  "rerunTasksEnabled": false
},
```

The build data can be consumed via API as higher-level build models that provide specific views on the data

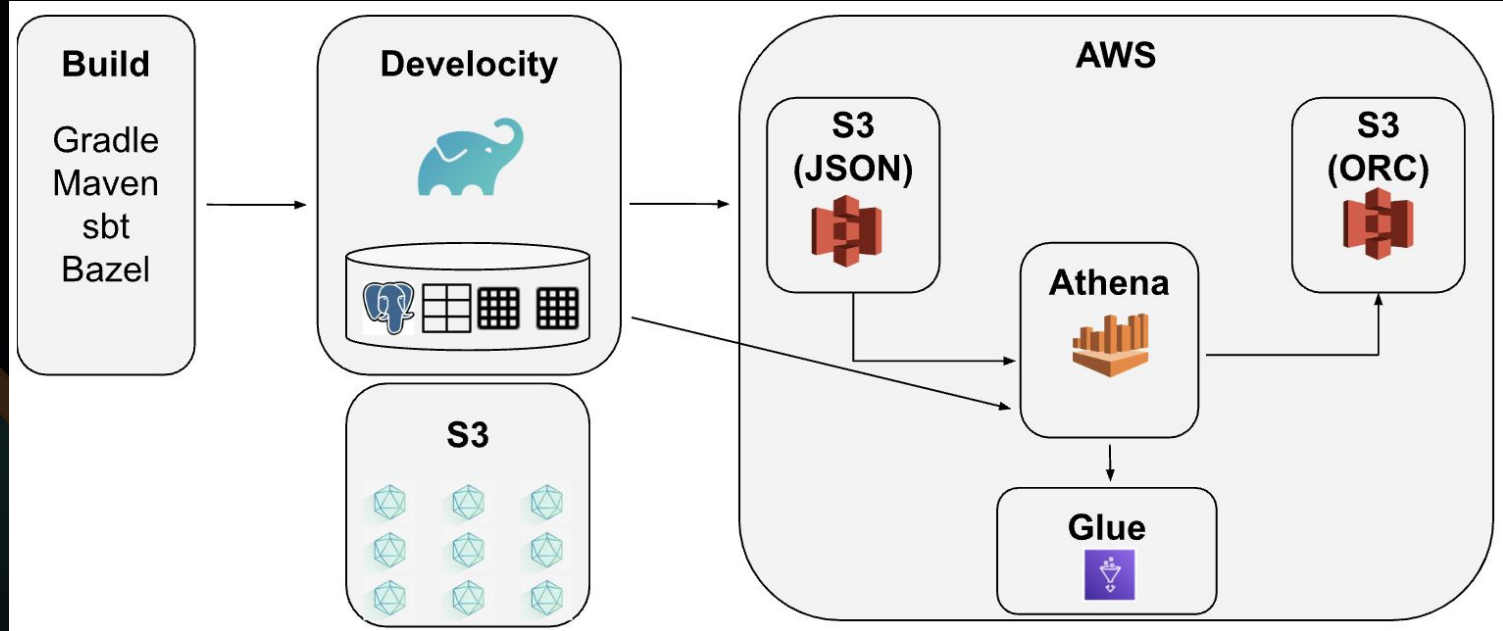
- General attributes of each build
- Multi-module/project structure
- Build cache performance characteristics
- Network activity
- Resource usage
- Applied plugins
- etc.

**Customers must be able to
instantly ask their own questions
about the build data**

Build models can be queried via big query services available from common cloud providers



Build models can be queried via big query services available from common cloud providers



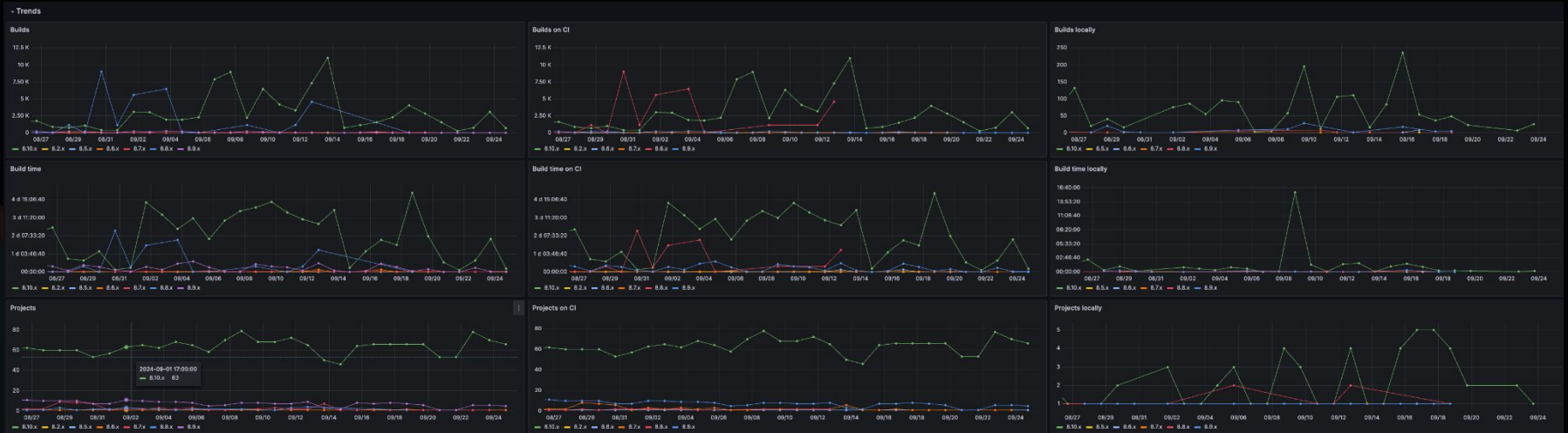
Build models can be queried via big query services available from common cloud providers

- Using ORC to store build data is faster to query and requires less disk space
- Storing multiple builds per ORC file avoids AWS S3 rate limits
- Using tables to mimic materialized views avoids slow UNNEST queries

Pre-built Grafana dashboards can be connected to big query services to gain actionable build insights across projects



Pre-built Grafana dashboards can be connected to big query services to gain actionable build insights across projects



Pre-built Grafana dashboards can be connected to big query services to gain actionable build insights across projects

Details per project and requested tasks/goals

All builds																	
Project	Requested tasks/goals	Build tool	Environments	Builds	Build time	Failed builds	Failure rate	Failed build time	Avg failed build time	Non-Verif. failures	Non-Verif. failure rate	Non-Verif. failure build tir	Avg Non-Verif. failure build	Verif. failures	Verif. failure rate	Verif. failure build time	Avg Verif. failure build tm
micronaut-starter	test-features:test	Gradle	CI	95	5 d 12:33:09	29	30.5%	1 d 07:48:08	01:05:43	1	1.1%	00:00:35	00:00:35	28	29.5%	1 d 07:45:31	01:08:03
micronaut	check	Gradle	CI	307	4 d 12:17:05	41	13.4%	20:30:09	00:30:00	0	0.0%	00:00:00	00:00:00	41	13.4%	20:30:09	00:30:00
java-parent	test	Gradle	Local	30	15:29:50	26	86.7%	15:29:32	00:35:45	14	46.7%	00:18:51	00:01:20	12	40.0%	15:10:40	01:15:53
data-parent	check	Gradle	CI	232	3 d 18:51:15	24	10.3%	13:07:26	00:32:48	3	1.3%	03:32:15	01:19:45	21	9.1%	09:35:10	00:27:23
micronaut-gradle-pl...	check	Gradle	CI	31	15:08:45	9	29.0%	05:47:26	00:38:36	0	0.0%	00:00:00	00:00:00	9	29.0%	05:47:26	00:38:36
foo	test	Gradle	CI	65.3 K	6 d 14:19:15	997	1.5%	04:57:43	00:00:17	871	1.3%	00:28:14	00:00:01	128	0.2%	04:29:28	00:02:08
CI builds																	
Project	Requested tasks/goals	Build tool	Environments	Builds	Build time	Failed builds	Failure rate	Failed build time	Avg failed build time	Non-Verif. failures	Non-Verif. failure rate	Non-Verif. failure build tir	Avg Non-Verif. failure build	Verif. failures	Verif. failure rate	Verif. failure build time	Avg Verif. failure build time
micronaut-starter	test-features:test	Gradle	CI	95	5 d 12:33:09	29	30.5%	1 d 07:48:08	01:05:43	1	1.1%	00:00:35	00:00:35	28	29.5%	1 d 07:45:31	01:08:03
micronaut	check	Gradle	CI	307	4 d 12:17:05	41	13.4%	20:30:09	00:30:00	0	0.0%	00:00:00	00:00:00	41	13.4%	20:30:09	00:30:00
data-parent	check	Gradle	CI	232	3 d 18:51:15	24	10.3%	13:07:26	00:32:48	3	1.3%	03:32:15	01:19:45	21	9.1%	09:35:10	00:27:23
micronaut-gradle-pl...	check	Gradle	CI	31	15:08:45	9	29.0%	05:47:26	00:38:36	0	0.0%	00:00:00	00:00:00	9	29.0%	05:47:26	00:38:36
foo	test	Gradle	CI	65.3 K	6 d 14:19:15	997	1.5%	04:57:43	00:00:17	871	1.3%	00:28:14	00:00:01	128	0.2%	04:29:28	00:02:08
oracle-cloud	check nativeTest	Gradle	CI	48	10:32:03	18	37.5%	04:41:13	00:15:37	0	0.0%	00:00:00	00:00:00	18	37.5%	04:41:13	00:15:37
Local builds																	
Project	Requested tasks/goals	Build tool	Environments	Builds	Build time	Failed builds	Failure rate	Failed build time	Avg failed build time	Non-Verif. failures	Non-Verif. failure rate	Non-Verif. failure build tir	Avg Non-Verif. failure build	Verif. failures	Verif. failure rate	Verif. failure build time	Avg Verif. failure build time
java-parent	test	Gradle	Local	30	15:29:50	26	86.7%	15:29:32	00:35:45	14	46.7%	00:18:51	00:01:20	12	40.0%	15:10:40	01:15:53
data-parent	micronaut-data-jdbc:stest	Gradle	Local	280	02:44:09	231	82.5%	02:28:09	00:00:37	60	21.4%	01:58:36	00:01:38	171	61.1%	00:49:32	00:00:17
micronaut	!http-server-netty:test	Gradle	Local	155	02:24:29	126	81.3%	02:17:24	00:01:05	79	51.0%	02:06:25	00:01:36	47	30.3%	00:10:59	00:00:14
data-parent	micronaut-doc-exampl...	Gradle	Local	121	01:23:00	93	76.9%	01:20:22	00:00:51	28	23.1%	01:11:38	00:02:33	65	53.7%	00:08:44	00:00:08
Micronaut Maven Pl...	install	Maven	Local	5	01:16:14	3	60.0%	01:15:34	00:25:11	3	60.0%	01:15:34	00:25:11	0	0.0%	00:00:00	00:00:00
data-parent	micronaut-doc-exampl...	Gradle	Local	58	01:09:22	47	81.0%	01:07:17	00:01:25	12	20.7%	01:01:32	00:05:07	35	60.3%	00:05:44	00:00:09

Pre-built Grafana dashboards can be connected to big query services to gain actionable build insights across projects

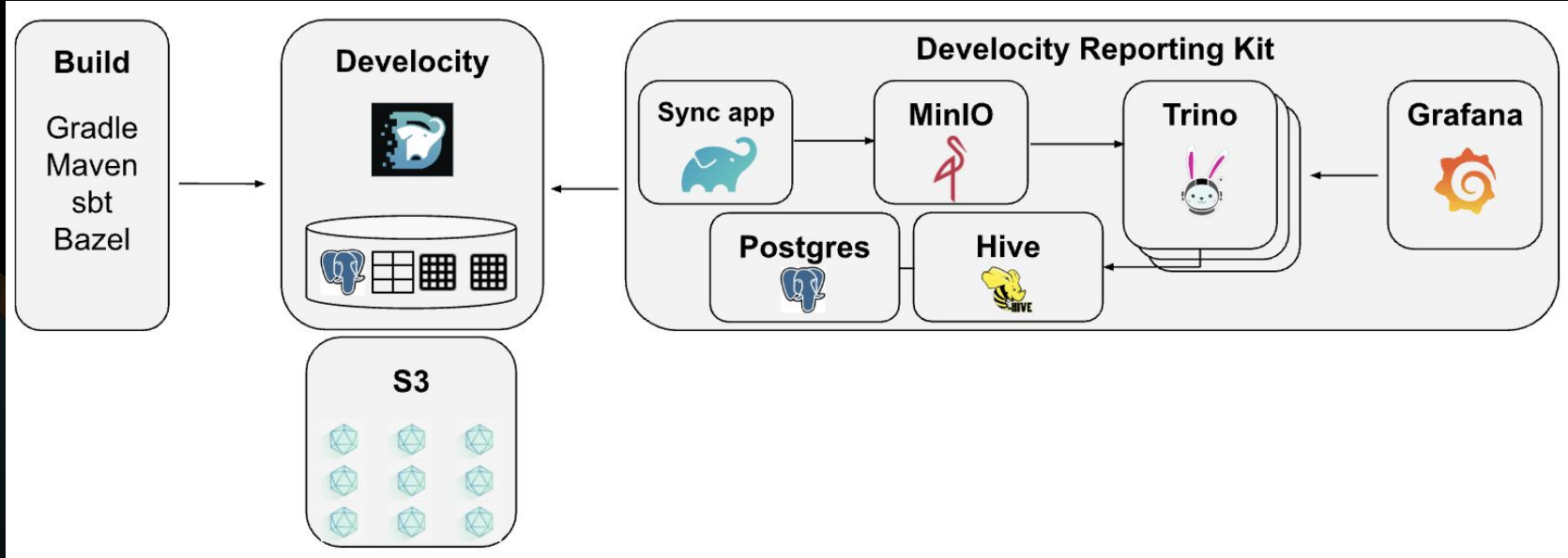
- Build volume
- Environment
- Build Failures
- Build Deprecations
- Plugins
- Networking
- Parallelization and resource usage
- Build Caching
- Settings

More build models and pre-built dashboards coming with every Develocity release.

Alerts can be implemented via Grafana Alerting or by combining different cloud provider services

- Example:
 - Event Bridge Scheduler
 - Lambda function
 - Simple notification service (SNS)

All-in-one reporting kit can be used to fetch, store, query, and visualize build data



Customers may prefer different reporting outputs to surface the build insights

Reports combining explanations and numeric results are another way to surface actionable build insights

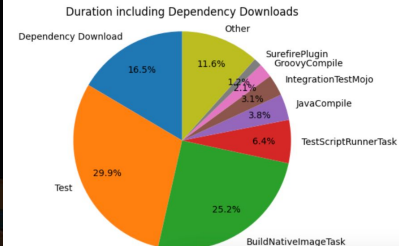
Analysis of Executed Maven Goals and Gradle Tasks

- duration: The cumulated build wall clock time of just running this goal.
- frequency: How many build projects run this goal or task as part of their build execution.
- Frequency Percentage: The percent of build projects running this goal or task as part of their build execution.
- Goal Duration Percentage: The duration of this goal relative to the total time of running goals. They will add up to 100%.
- Total Duration Percentage: The duration of this goal relative to the cumulated overall build time (which for example includes dependency download time). They will not add up to 100%.

All those variables are relative to the context in which they are evaluate. When a goal is solely executed on CI, this goal has a higher duration percentage in the CITotal context than in the Total context.

Goals with more than 1% total duration for Total

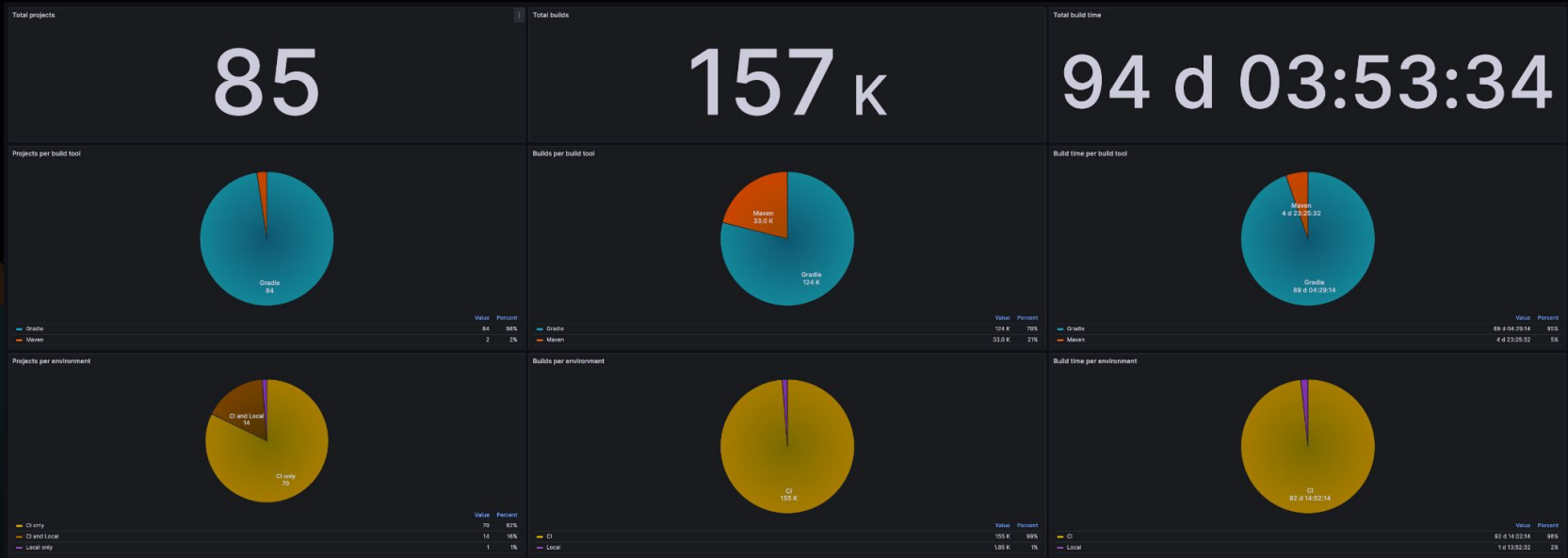
Type	Duration	Frequency	Build Cache Savings	Frequency Percentage	Build Cache Savings Percentage	Goal Duration Percentage	Total Duration Percentage
org.gradle.api.tasks.testing.Test	36,038	349	13,647	3.66%	37.87%	35.88%	29.95%
org.graalvm.buildtools.gradle.tasks.BuildNativeImageTask	30,334	113	0	1.18%	0.00%	30.18%	25.19%
io.micronaut.guides.tasks.TestScriptRunnerTask	7,714	21	9,003	0.22%	116.72%	7.67%	6.41%
org.gradle.api.tasks.compile.JavaCompile	4,625	726	9,921	7.60%	214.51%	4.60%	3.84%
org.apache.maven.plugins.invoker.IntegrationTestMojo	3,788	3	0	0.03%	0.00%	3.77%	3.15%
org.gradle.api.tasks.compile.GroovyCompile	2,575	328	2,037	3.44%	79.11%	2.56%	2.14%
org.apache.maven.plugin.surefire.SurefirePlugin	1,433	14	0	0.15%	0.02%	1.43%	1.19%
Other	14,012	nan	7,333	nan%	52.34%	13.94%	11.64%



Use cases

Inactive project causing high CI build queues

NPM & Python build insights coming in 2025



High dependency download rate overloading the infrastructure

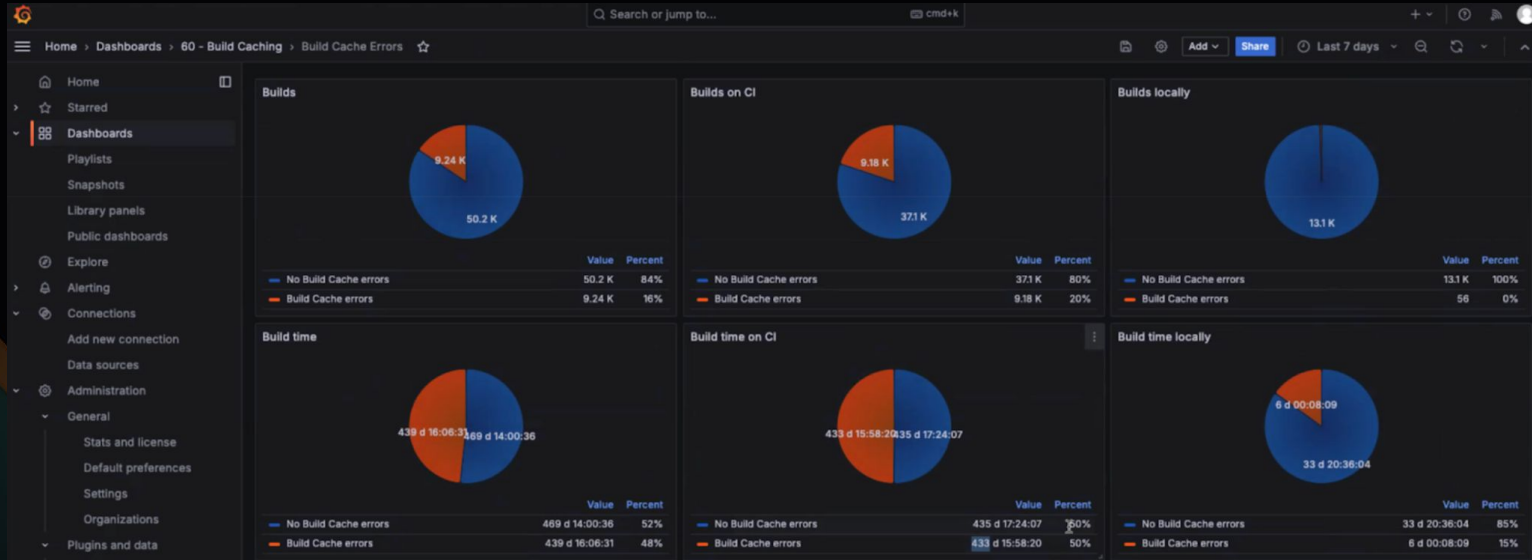


High dependency download rate overloading the infrastructure

Ephemeral build slow-down mitigation coming in 2025



Network instabilities causing half the build volume to run without build caching



Network instabilities causing half the build volume to run without build caching



JDK usages not in compliance with company policies

CI builds			
Version		Projects	Builds
Oracle Java HotSpot 17.0.12		74	30.0 K
Eclipse OpenJDK 17.0.12		77	122 K
Oracle Java HotSpot 21.0.4		72	2.12 K
GraalVM OpenJDK 17.0.9		3	459
Eclipse OpenJDK 21.0.4		1	161
Amazon OpenJDK 17.0.12		1	46

Local builds			
Version		Projects	Builds
Homebrew OpenJDK 21.0.2		2	11
Homebrew OpenJDK 17.0.12		1	21
Oracle Java HotSpot 21		1	7
GraalVM OpenJDK 17.0.4		2	14
GraalVM OpenJDK 22.0.2		1	2
Amazon OpenJDK 17.0.7		1	3

**Develocity provides comprehensive build data
and enables instant build data exploration
to improve the developer toolchain**

VISIT US

Want to learn more about Develocity, or
want to work on Develocity?

Visit us at the booth in the lounge!

THANKS

lploix@gradle.com

etienne@gradle.com

gradle.com/careers