DevProd for CI maintainers

DPE Summit 2023

Etienne Studer, SVP of Engineering, Develocity
Collect deep build data to understand the state of the developer toolchain on CI and make informed decisions what to improve.
Agenda

- Capturing build data on CI
- Processing the captured build data
- Surfacing insights from the processed build data
Capturing build data on CI
Collecting data about every CI build allows to understand the state of the developer toolchain on CI, including performance, reliability, and resource usage.
CI build data can be collected by configuring all projects’ builds to capture and publish their build data to the server.

This requires the buy-in of the project owners and build modifications.

When there are many projects, data is required to prioritize which projects to reach out to first.
Alternatively, the CI build data can be collected by having the CI runner inject the configuration to capture and publish the build data into the invoked build.

This does not require touching the projects’ builds.
CI plugins for Develocity

Jenkins Plugin
https://github.com/jenkinsci/gradle-plugin

Gitlab templates
https://github.com/gradle/gradle-enterprise-gitlab-templates

GitHub Gradle Build Action
https://github.com/gradle/gradle-build-action

TeamCity Plugin
https://github.com/etiennestuder/teamcity-build-scan-plugin

Bamboo Plugin
https://github.com/gradle/gradle-enterprise-bamboo-plugin
Example: Config-injection on Jenkins

Grade Enterprise integration

- Gradle auto-injection

Gradle Enterprise connection settings

- Gradle Enterprise server url
  - [https://mycompany.com](https://mycompany.com)
- Allow untrusted server
- Enforce Gradle Enterprise server url
- Gradle Enterprise access key

The access key must be in the `server host name`-`access key` format. For more details please refer to the documentation.

General settings

- Auto-injection Git VCS repository filters (Beta)

- Check for the Gradle Enterprise build agent server

Jenkins

Dashboard ➔ gradle-freestyle ➔ #3

Build #3 (May 9, 2023, 10:56:14 PM)

- No changes...
- Started by anonymous user
- Revision: 814b3b8652adca09fd64b0d933a2840adbab6ec58
- Repository: [https://github.com/gradle/gradle.git](https://github.com/gradle/gradle.git)
  - [https://ge.mycompany.com/w/ujj92/jirrmaj936](https://ge.mycompany.com/w/ujj92/jirrmaj936)

Build Scans

- [https://ge.mycompany.com/w/ujj92/jirrmaj936](https://ge.mycompany.com/w/ujj92/jirrmaj936)
Config-injection for Gradle and Maven

Gradle:
./gradlew build --init-script developecity-init.gradle

Maven:
mvn package -Dmaven.ext.class.path=developecity-extension.jar
All configuration related to capturing and publishing the build data to the server can be consolidated into a versioned convention plugin / extension.

This unifies the configuration of the data capturing, build caching, test acceleration, custom values, etc.

Example: [Gradle plugin](https) / [Maven extension](https)
Processing build data
Exporting build data into a big data store allows asking specific toolchain questions at scale.
Build models exposed by Develocity are described via OpenAPI specification. They currently cover build attributes, build cache performance, and project structure.

Build models are consumption-oriented.

Build model versions are backward-compatible.
Build models can be manually retrieved by writing a client that gets the data from the Develco API.

This requires retries, resuming, paging, parallelizing, etc. and either ad-hoc analysis or saving in another store.

Manually retrieve build models via API client

Alternatively, build models can be automatically exported by Develocity and made available to a big data engine. This allows to immediately ask specific toolchain questions.
Build model export, query & visualization

Develocit y ➔ AWS S3 ➔ AWS Athena ➔ Grafana
Develocity – build model export

Develocity can be configured to export the build models to S3.

Exported models are automatically updated on new versions.

Newly available models are automatically exported.
AWS S3 – build model intermediate storage

Build models conform to the same OpenAPI-based schema as the Develocity API.

All build models of a build are in a single JSON stored in compressed format.

Stored build models are partitioned by build start time, one prefix per hour.

Designed for consumption by any Apache Hive-based engine.
Athena is a serverless big data query engine, with some caching to save query results.

The schema of the table is defined by the JSON schema of the build models, with each build model contained in a separate column.

SQL-like queries are run against tables and views and may include joins across build models.
Athena datasource is available for Grafana.

Charts are backed by queries into Athena tables and views.
Using Terraform to provision Athena resources.

Construct Grafana dashboard from JSON definitions.

Planned: Automatic creation and updates of Athena table definitions by Develocity
Surfacing insights from build data
Use the captured build data to identify usage and failure patterns, flakiness, build acceleration potential, resource waste, etc.
Overview

- How many projects, builds, build tools?

<table>
<thead>
<tr>
<th>Total projects</th>
<th>Total users</th>
<th>Total builds</th>
<th>Total build time</th>
<th>Total build cache usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>16</td>
<td>251k</td>
<td>134 d 08:40:13</td>
<td>61 d 02:40:44</td>
</tr>
</tbody>
</table>

Projects per Build Tool:

- Value 1 (80%)
- Value 2 (11%)
- Value 3 (9%)

Users per Build Tool:

- Value 4 (100%)

Builds per Build Tool:

- Value 5 (100%)

Build times per Build Tool:

- Value 6 (100%)

Build Cache Savings:

- Value 7 (100%)

Legend:

- Online
- Offline

Percentages:

- Value 8 (80%)
- Value 9 (11%)
- Value 10 (9%)

Value:

- Value 1 (80)
- Value 2 (11)
- Value 3 (9)
- Value 4 (100)
- Value 5 (100)
- Value 6 (100)
- Value 7 (100)
- Value 8 (80)
- Value 9 (11)
- Value 10 (9)
Contributors to build volume

- Projects with highest build count, highest total build duration, highest median build time?
Failure impact

- Projects with highest build count of failed builds, total build duration, and median build time?
- Projects with highest total build duration of builds failing due to non-verification failures?
- Build agents with highest build count of failed builds, total build duration, and median build time?
- ...
Build acceleration improvements

- Projects that do not have build caching enabled with highest total execution and median execution time of generate & compile & test?
- Projects that have build caching enabled with highest total execution and median execution time of cacheable goals/tasks
- Projects that have build cache errors
- Projects that have long dependency download times
- Multi-module Maven or Gradle projects not built in parallel
- Multi-module Maven or Gradle projects built in parallel with only 1 worker
- Maven projects run with different Maven versions
- …
Prioritize actions for improvements based on quantitative impact analysis and surfacing of the top-10 offenders.
THANKS

etienne@gradle.com
gradle.com