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

1

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

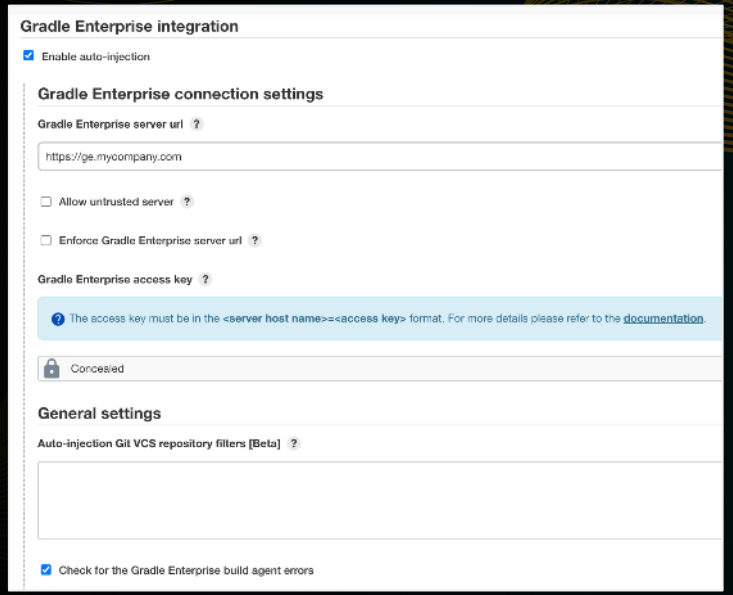
Gitlab templates
https://github.com/gradle/gradle-enterprise-gitlab-templates

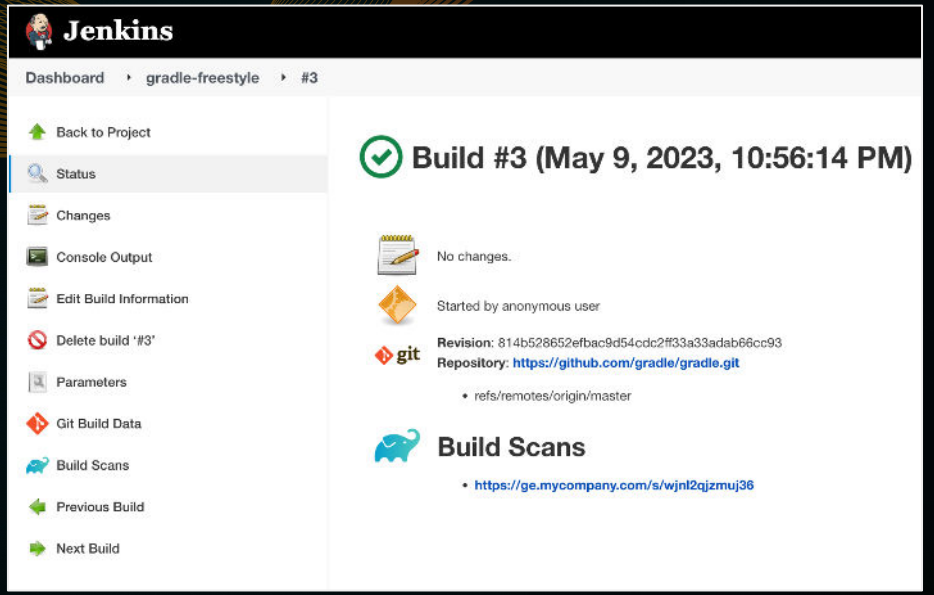
GitHub Gradle Build Action
https://github.com/gradle/gradle-build-action

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





Config-injection for Gradle and Maven

Gradle:

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

Maven:

```
mvn package -Dmaven.ext.class.path=develocity-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 / Maven extension

2

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 Develocity API.

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

docs.gradle.com/enterprise/api-manual/ref/2023.3.html

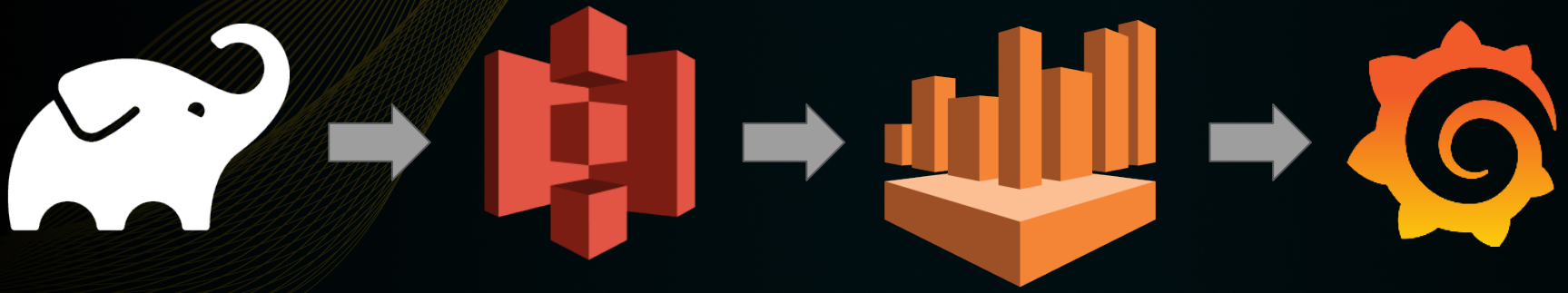
Manually retrieve build models via API client

```
curl -s -H "$AUTH" "https://
develocity.develocity.mycompany.com/api/builds
?fromInstant=1694097000000&maxBuilds=3
&query=tag:ci" | jq
```

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



Develocity

AWS S3

AWS Athena

Grafana

Develocity – build model export

Develocity can be configured to export the build models to S

Exported models are automatically updated on new version

Newly available models are automatically exported.


```
{
  "id": "tzbbyguougvos",
  "type": "gradle",
  "buildToolVersion": "7.3.3",
  "buildAgentVersion": "3.8.1",
  "modelVersion": {
    "string": "2023.1.6",
    "year": 2023,
    "release": 1,
    "patch": 6
  },
  "gradleAttributes": {
    "id": "tzbbyguougvos",
    "buildStartTime": 1655007064459,
    "buildDuration": 13521,
    "gradleVersion": "7.3.3",
    "pluginVersion": "3.8.1",
    "rootProjectName": "object-storage-parent",
    "requestedTasks": [
      "tasks",
      "--all"
    ],
    "hasFailed": false,
    "tags": [
      "CI",
      "HEAD",
      "github:action",
      "Linux"
    ],
```

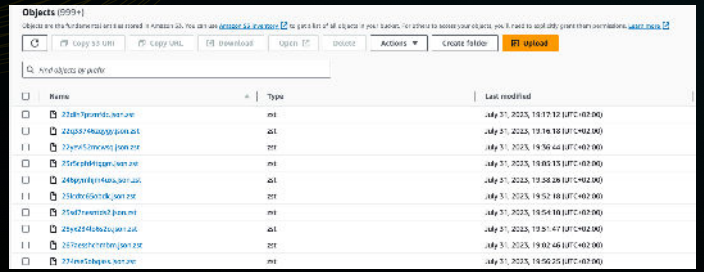
AWS S3 – build model intermediate storage

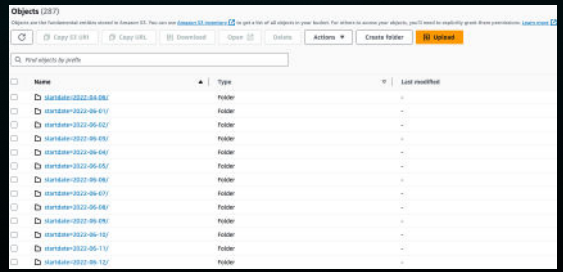
Build models conform to the same OpenAPI-based schema as the Develocity API.

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.



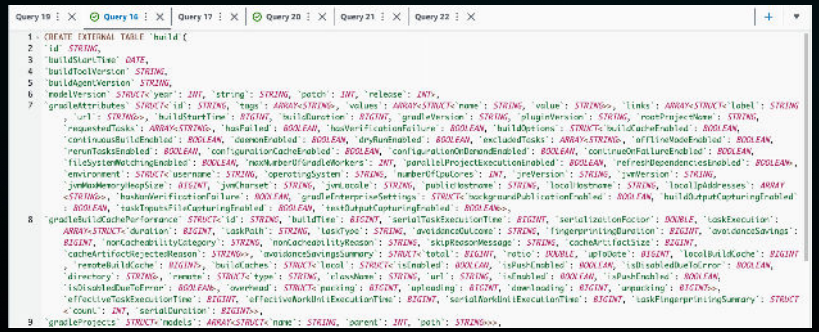


AWS Athena – build model query

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.

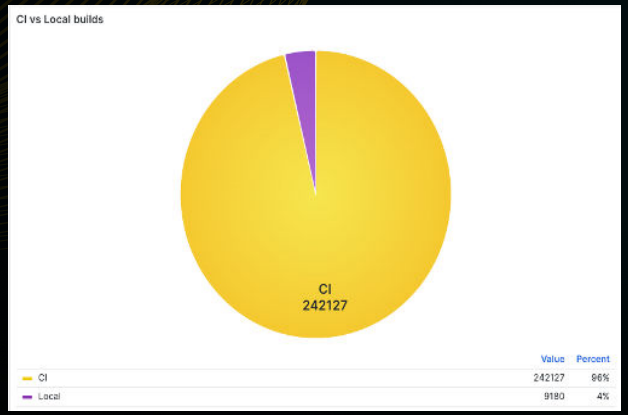
SQL-like queries are run against tables and views and may include joins across build models.

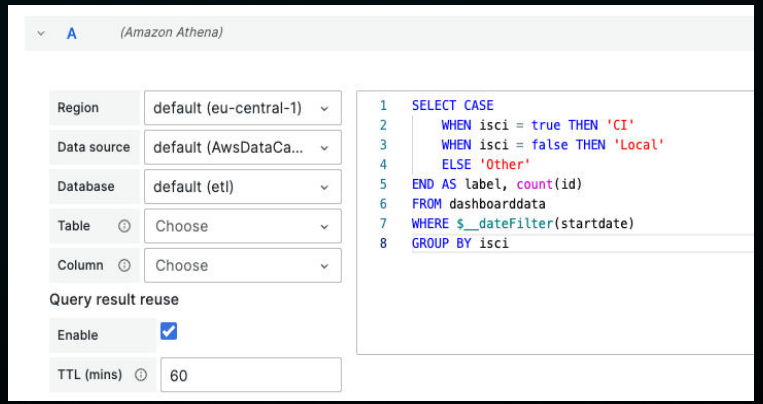


Grafana – build data visualization

Athena datasource is available for Grafana.

Charts are backed by queries into Athena tables and views.





IoC

Using Terraform to provision Athena resources.

Construct Grafana dashboard from JSON definitions.

Planned: Automatic creation and updates of Athena table definitions by Develocity

3

Surfacing insights from build data

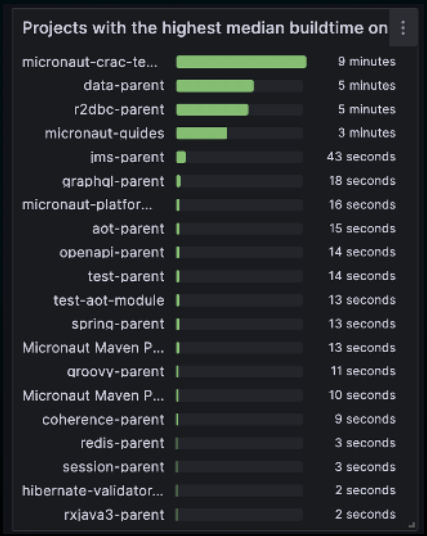
Use the captured build data to identify usage and failure patterns, flakiness, build acceleration potential, resource waste, etc.

Overview

- How many projects, builds, build tools?

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