



## What is CI/CD observability and how to bring it to CI/CD pipelines

A reliable CI/CD pipeline is the backbone of every modern project, yet there's limited visibility into its processes, often requiring manual review and analysis of build outputs. By leveraging OpenTelemetry standards we want to bring observability to our pipelines, making the software delivery process fully observable. Join our journey redefining CI/CD observability, learn how you can start implementing the techniques we are using at Grafana Labs to ensure your pipelines are more reliable and stay performant over time, how to identify flakiness, bottlenecks, and how we envision a future where - no matter your system or your observability solution - we can effortlessly have full visibility over our software delivery process. You will learn what it takes to make CI/CD fully observable, how we think OpenTelemetry is going to play a major role in this, what obstacles we encountered, what are the challenges ahead and how anyone can help shape the future of CI/CD observability.

## Presenters



### **Giordano Ricci**

Senior Software Engineer  
@Grafana - Explore Squad

### **Dimitris Sotirakis**

Senior Software Engineer  
@Grafana - Platform Squad

## Agenda

- What is CI?
- Current issues with CI/CD systems
- Intro to OpenTelemetry
- Semantic Conventions
- Own your data
- Practical use cases for CI/CD observability
- What's next?



What is CI?



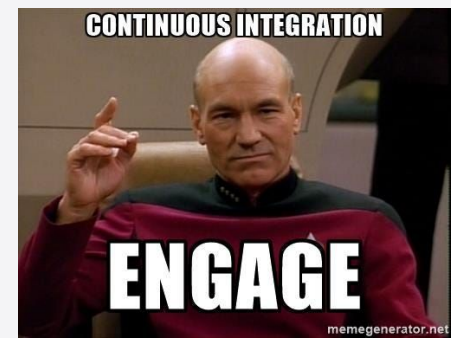
# Definition(s) of Continuous Integration

Continuous Integration: The continuous assembling and testing of complex and rapidly evolving systems.

Software Engineering at Google 📖

Continuous Integration: The process of combining code changes frequently, with each change verified on check-in.

Grokking Continuous Delivery 📖



What is CI?  
But like - for real

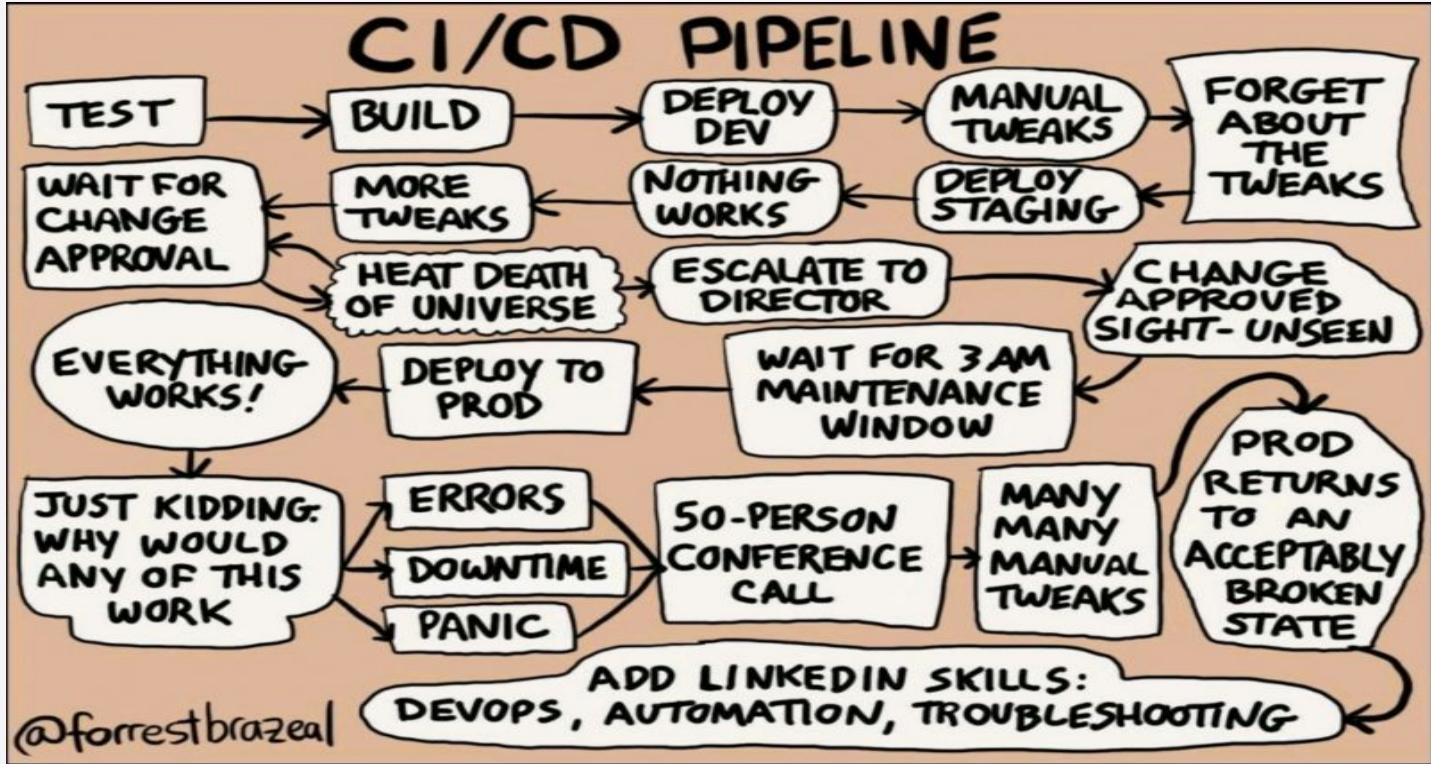


# Definition of Continuous Integration but for real

A mechanism to:

- Reduce risks
- Reduce repetitive manual processes
- Generate deployable software at any time and at any place
- Enable better project visibility
- Establish greeted confidence in the software product
- Find and resolve flaky tests/builds
- ...

...and of course (try to) prevent paging people at 3AM :-)



What is CI?  
But like - for **real**  
real





# Definition of Continuous Integration for **real real**

## CI and Alerting: A Common Purpose

- Proactive issue identification
- Continuous system monitoring
- Rapid response to anomalies/outliers

## CI as the "Left Shift" of Alerting

- Early detection in development
- Preemptive issue resolution
- Shifting focus to proactive monitoring



## Continuous Integration (CI)

- **Early Detection and Issue Catching in Development**
- **Build Health and System Maintenance by utilizing signals**
- **Continuous Monitoring for System Health**
  
- **Implementing Actionable Alerts and Tests**
- **Viewing CI and Alerting as Complementary Components**



## Continuous Integration (CI)

- Early Detection and Issue Catching in Development
- Build Health and System Maintenance by utilizing signals
- Continuous Monitoring for System Health
- Implementing Actionable Alerts and Tests
- Viewing CI and Alerting as Complementary Components

## Alerting

- Rapid Problem and Timely Issue Identification
- Service-Level Objective (SLO) Compliance using signal monitoring
- Well-Managed Alerting for System Health
- Dealing with Brittle Alerts and Tests
- Overcoming Siloed Perspectives

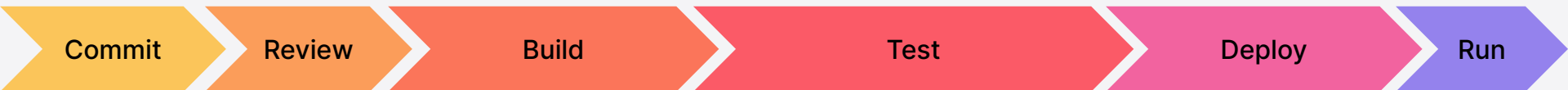


# Current issues with CI/CD systems



# Observability so far

- Local runs - `fmt.Println("here")` (*choose your fav language*)
- "Let's page the Platform team, they should know!"
- Create alerts after the fact
- Github → CI check webhook → Grafana \*



- Limited visibility during earlier stages
- Difficulty in root cause analysis
- Increased mean time to recovery (MTTR)
- Missed optimization opportunities



# Observability so far



# Observability so far

011y so far

Deploy

Run



# Getting data out of CI/CD



- Proactive Issue Resolution
- Streamlined Development Workflow
- Enhanced System Reliability
- Cost Reduction





# Getting data out of CI/CD

## FOCUS SHIFT



Source: <https://imgur.com/diZkDjj>



# OpenTelemetry 101 - What is OTel?

An **observability** framework designed to create and manage telemetry data such as **traces**, **metrics**, and **logs**.



# OTel & CI/CD - The interesting bits

**Semantic  
conventions**

**Standard naming** scheme for  
common telemetry data types

**Own the data  
that you  
generate**



# HOW STANDARDS PROLIFERATE: (SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION:  
THERE ARE  
14 COMPETING  
STANDARDS.

14?! RIDICULOUS!  
WE NEED TO DEVELOP  
ONE UNIVERSAL STANDARD  
THAT COVERS EVERYONE'S  
USE CASES.



SOON:

SITUATION:  
THERE ARE  
15 COMPETING  
STANDARDS.

<https://xkcd.com/927/>



# Semantic Conventions

## By Signal Type


- Events
- Logs
- Metrics
- Resource
- Trace

## By Area

- General
- Cloud Providers
- CloudEvents
- Database
- Exceptions
- ...
- **Continuous Integration & Continuous Delivery**



← Back to pull request #80688

✔ (Draft): Nested Content Outline #35389 

Re-run all jobs ...

Summary

Jobs

- ✔ codeowners-valid...

Run details

Usage

Workflow file

**codeowners-validator**  
succeeded 11 hours ago in 18s

- ✔ Set up job 1s
- ✔ Pull ghcr.io/mszostok/codeowners-validator:v0.7.4 2s
- ✔ Run actions/checkout@v4 5s
- ✔ GitHub CODEOWNERS Validator 7s
- ✔ Post Run actions/checkout@v4 0s
- ✔ Complete job 0s

## Don't delete the original objects for linked items


✔ Passed Martin Owens created pipeline for commit `ac65d80d` finished 9 hours


For `master`

latest 13 Jobs 158.02 161 minutes 56 seconds, queued for 2 seconds

Pipeline Needs **Jobs 13** Tests 0

Status	Job	Stage
✔ Passed	#5943405325: media-check	test
00:01:40	master ac65d80d	



← grafana 

❌ (Draft): Nested Content Outline | harisrozajac opened pull request

PIPELINE STAGES 3 stages

- ❌ pr-test-frontend 06:59

STEPS

- ✔ clone 00:31
- ✔ identify-runner 00:01

CONSOLE LOGS

```

1 Cloning with 3 retrie
2 Initialized empty Git
3 + git fetch origin +r
4 From https://github.c
5 * branch
6 * [new branch]
7 + git checkout main
8 Branch 'main' set up

```





Gio's totally reliable Go code





**Gio** 📅 1:04 PM

Yo! 🙌 I wrote some "Totally reliable Go code" ™ to get the data we needed out of Drone into Loki, Tempo and Mimir, can you check it out?



**ivana** 📅 1:05 PM

So cool! This is the most reliable code I've ever seen! 😄 Can you make it so it also sends Logs to [ElasticSearch](#) plz?



**PJ** 📅 1:05 PM

🎉 That's great! Can we get tracing data out of GitHub Actions?



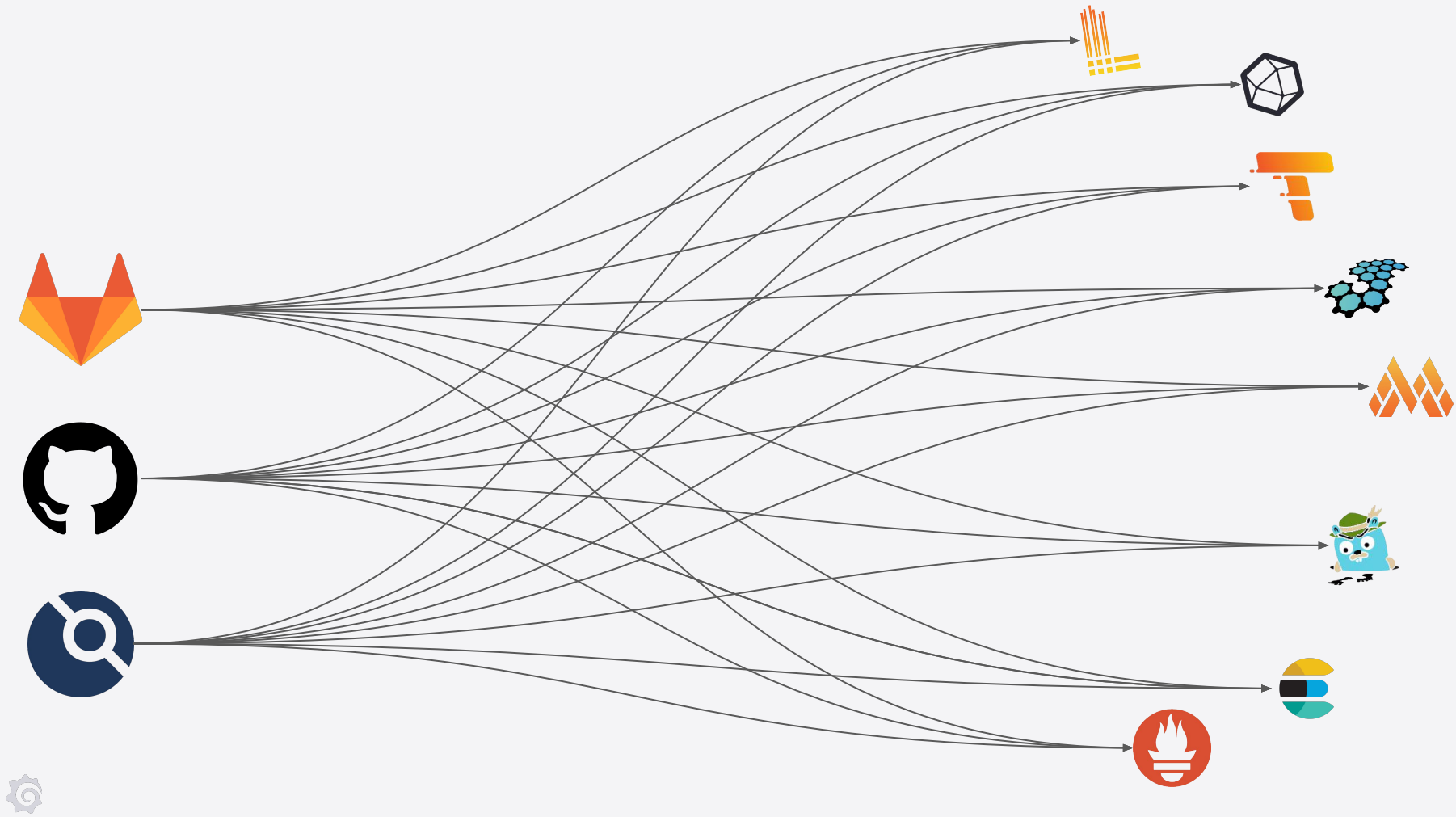
**Gio** 📅 1:06 PM

kill me (666 kB) ▾



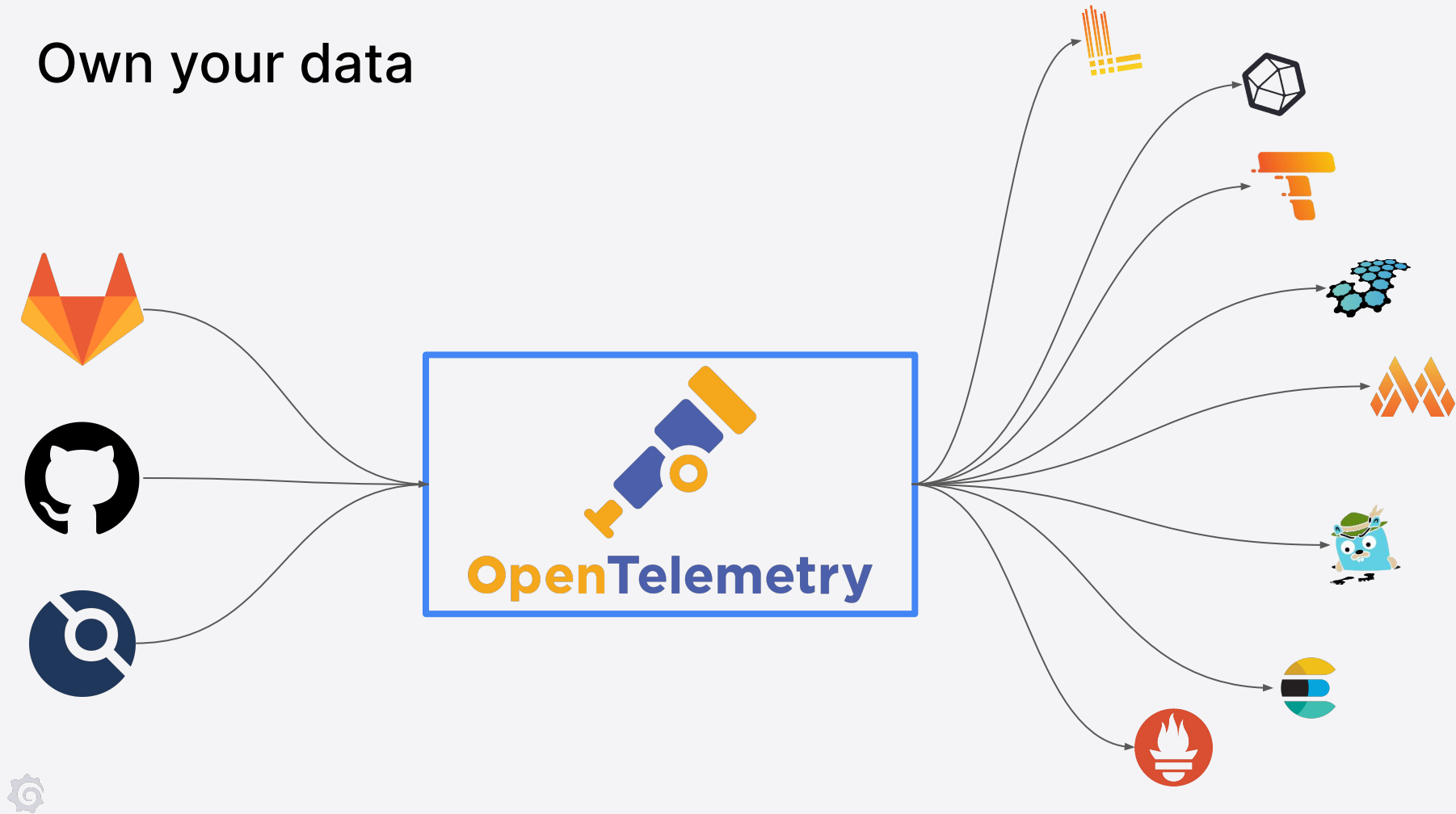




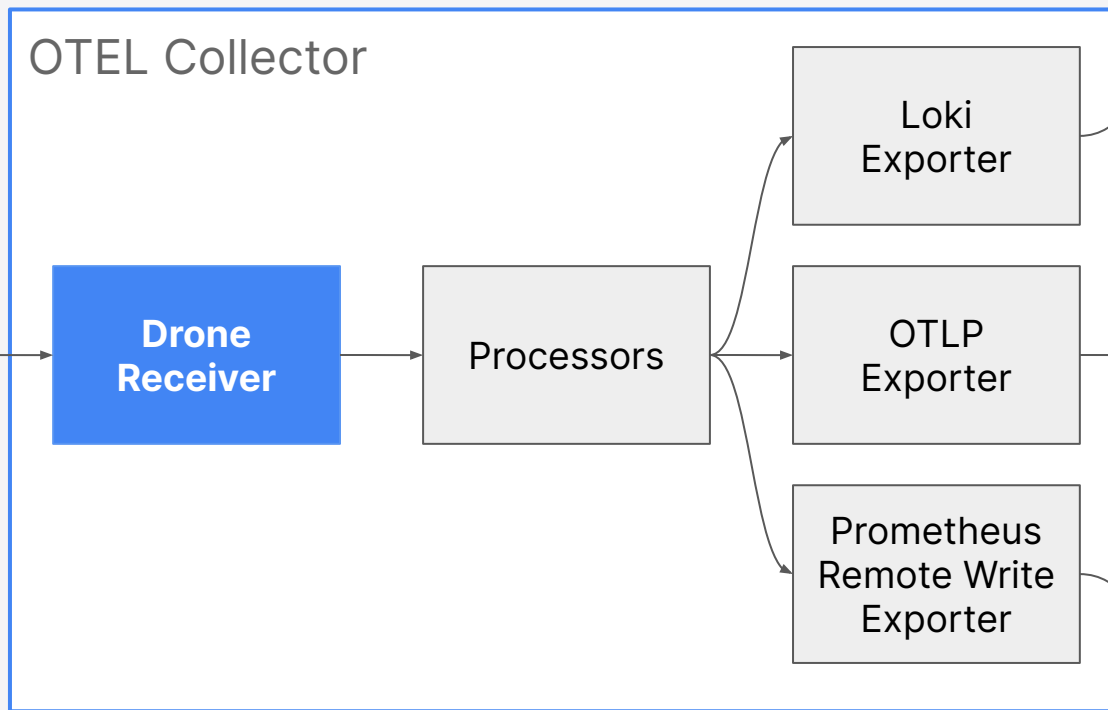




# Own your data



# grafana-ci-otel-collector



# Practical examples

- <https://plugins.jenkins.io/opentelemetry/>
  - Jenkins plugins that exports data via OTLP protocol
- <https://github.com/krzko/run-with-telemetry>
  - Run commands and export results as trace data
- <https://github.com/grafana/grafana-ci-otel-collector>
  - Generate Logs, Metrics, and traces from Drone Pipelines



# CI/CD Observability in practice

The image displays two side-by-side dashboards from Grafana, illustrating CI/CD observability. The left dashboard is a Loki query editor for the 'Loki-Gracie' data source. It shows a query for logs related to a failed test step in a CI pipeline. Below the query editor, the 'Logs volume' section is visible, showing a log entry for a failed test.

```
{exporter="OTLP"} | json | attributes_ci_drone_step_name = 'test-frontend' | = 'FAIL public/app/plugins/datasource/loki/configuration/DerivedFields.test.tsx' | line_format '{{.body}}'
```

Options: Type: Range Line limit: 1000 This query will process approximately 10.8 GiB.

Common labels: test-frontend OTLP Line limit: 1000 (18 returned) Total bytes processed: 12.5 GiB

Fields

Field	Value
attributes_ci_drone_build_number	154282
attributes_ci_drone_stage_name	pr-test-frontend
attributes_ci_drone_step_name	test-frontend
body	FAIL public/app/plugins/datasource/loki/configuration/DerivedFields.test.tsx (5.373 s)
exporter	OTLP
spanid	98a4aba1d28d61a9

The right dashboard is a Jaeger trace viewer for a 'drone' service. It shows a detailed view of a trace with 29 spans. The trace starts with a 'clone' operation (29s) and ends with a 'pr-lint-frontend' operation (4m 37s). The trace is filtered to show spans for the 'drone' service.

drone: 6m 55s 2024-01-12 12:51:26.000

Span Filters: 29 spans

Timeline: 0µs, 1m, 3m, 5m

Spans:

- drone (6m 55s)
- pr-build-e2e (3m 38s)
- clone (29s)
- trigger-enterprise (2s)
- grabpl (1s)
- verify-gen-jsc (1s)
- yarn-install (18s)
- compile-build (2s)
- identify-runner (1s)
- verify-gen-cu (23s)
- build-frontend (20s)
- rgm-build (2m 50s)
- rgm-pack (1m 9s)
- wire-install (10s)
- e2e-tests (1m 2s)
- pr-lint-frontend (4m 37s)
- clone (30s)
- identify-runner (1s)
- yarn-install (18s)
- clone-enterprise (5s)



# CI/CD Observability in practice

Home > Apps > CI/CD Observability > Repositories > grafana/grafana > Builds > #155502

## #155502

grafana/grafana

Build Trace

- pr-test-frontend 07:23
  - clone 00:30
  - identify-runner 00:00
  - yarn-install 00:40
  - betterer-frontend 01:01
  - clone-enterprise 00:04
  - test-frontend 06:13
- pr-lint-frontend 05:47
  - clone 00:33
  - clone-enterprise 00:06

```
1 Cloning with 3 retries
2 Initialized empty Git repository in /dr
3 + git fetch origin +refs/heads/main:
4 From https://github.com/grafana/grafana
5 * branch      main      ->
6 * [new branch] main      ->
7 + git checkout main
8 Branch 'main' set up to track remote bra
9 Switched to a new branch 'main'
10 + git fetch origin refs/pull/80884/head:
11 From https://github.com/grafana/grafana
12 * branch      refs/pull/808
13 + git merge 53c2c67cb26e8edff6ab5a73ba25
14 Auto-merging .betterer.results
15 Merge made by the 'recursive' strategy.
16 .betterer.results
17 .../dashboard-scene/panel-edit/PanelEdi
18 .../panel-edit/PanelOptionsPane.tsx
19 .../panel-edit/PanelVizTypePicker.tsx
20 .../components/PanelEditor/OptionsPaneC
```

Home > Apps > CI/CD Observability > Repositories

## Repositories

Repository grafana/grafana Branch All x x Last 7 days UTC

### Status timeline

Timeline showing build status for various branches and versions from 01/11 00:00 to 01/17 00:00. Legend: Success (green), Failure (red), Killed (orange), Blocked (yellow), Declined (purple), Pending (grey), Waiting on Dependencies (light blue).

### Builds by status

Status	Count
Success	73728
Running	36864
Failure	18432
Error	9216
Killed	4608
Blocked	2304
Declined	1152
Pending	576
Waiting on Dependencies	288
Other	144
Other	72
Other	36
Other	18
Other	0

### Recent builds

Build #	Start time	Status	Duration
154907	2024-01-17 04:46:41	Success	00:11:16
154906	2024-01-17 04:32:09	Failure	00:13:23





# What it unlocks

RED-like metrics (Rate, Errors, Duration)

DORA metrics

Code coverage stats over time

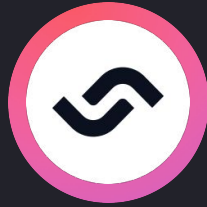
Flakiness detection

CI performance regression

Security

...A lot more





# What's next

- An OpenTelemetry working group focusing on defining CI/CD semantic conventions
- More receivers / components or tools to get data out of CI/CD systems

## Join the discussion

- <https://cloud-native.slack.com/archives/C0598R66XAP>
  - CI/CD Observability channel on CNCF Slack
- <https://github.com/open-telemetry/oteps/pull/223>
  - OpenTelemetry enhancement proposal





# Thank you

Want to chat?

We'll be at the Grafana booth  
k8 - level 1 ~15:40

