CD.FOUNDATION

cdevents

The Next Evolution in CI/CD Technology

A WHITEPAPER PUBLISHED BY THE CD FOUNDATION'S CDEVENTS PROJECT TEAM

This whitepaper describes the newest technology in CI/CD - CDEvents. It is intended for DevOps Engineers, Project Managers/Directors, CTOs, and Cloud Architects who are interested in evolving their DevOps pipelines to become more scalable, robust, measurable, and visible, using a technology- agnostic solution to provide interoperability.

The technology described is still in its very early stages, and the concepts in it could change quite substantially as we go along, so please join us to make sure this technology evolves into something we could all benefit from.

What is CDEvents?

Today's CI/CD systems often comprise of services that do not talk to each other in a standardized way. Such services include pipeline orchestrators, build/test tools, deployment tools, metrics collectors and visualizers. This leads to problems related to interoperability, notification of failure issues, and poor automation.

The Continuous Delivery Foundation's CDEvents project has been created to solve the interoperability problem. The mission of the CDEvents project is to define standards for an event-based CI/CD pipeline to support CI/CD systems with a decoupled architecture.

The CDEvents project focuses on both event-based CI/CD standards and best practices for event-driven CI/CD systems. The CDEvents project aims to define the common language of the CI/CD ecosystem events, so it provides a vocabulary, a specification as well as SDKs.

CDEvents Benefits

CDEvents delivers:

- Easy to scale pipelines
- Increased automation between workflows
- A simple way to enhance or modify workflows
- Standardized notifications for metrics collectors and visualizers

A decoupled CI/CD architecture is easy to scale and makes the CI/CD pipelines more resilient to failures, which is critical as the end-to-end software production and delivery pipelines grow more and more complex, not least in a microservices architecture with thousands of independent pipelines. Using CDEvents also increases automation when connecting workflows from different systems to each other, and as a result, empowers tracing/visualizing/auditing of the connected workflows through these events. Additionally, CDEvents make it super easy to switch between different CI/CD tooling to enhance or modify your workflows quickly.

The Goal of the CDEvents Project

The CDEvents project's mission is to standardize an event protocol specification that caters to technology-agnostic machine-to-machine communication in CI/CD systems. This specification will be published, reviewed, and agreed upon between relevant Linux Foundation projects/members. The CDEvents project aims to provide reference implementations such as event consumers/listeners and event producers/senders on top of for example CloudEvents.

History

Before we dive in further, a bit of history. The Continuous Delivery Foundation's Interoperability Special Interest Group(SIG) was created in early 2020 to discuss and research interoperability in the CD space. One of the workstreams of the SIG was focused on interoperability through 'events.' In early 2021 the workstream was transformed into a SIG of its own, and towards the end of that year, the CDEvents project was created. The project was proposed as a CDF incubating project and was accepted by the CDF Technology Oversight Committee in December 2021.

CDEvents Specification

To define CDEvents the contributors understood the need to define a common standard and vocabulary. Following is a description of the specification.

CDEvents Topics

Events provide interoperability between CI/CD tooling through topics. Part of the mission of the CDEvents project is to determine the best usage and process for CDEvents and to define a common standard. The CDEvents project team is working to define:

- When are events suited for triggers, audits, monitoring, and management?
- Common guidelines for at-least-once, at-most-once, exactly once, and ordering logic.
- When to apply particular strategies
- Events to be used by tools for orchestration/workflows
- Pipeline to pipeline communication via events
- Tracing/auditing/graphing/visualizing of the entire process, e.g., through events showing what has occurred.
- CDEvents Metrics, e.g., how many versions have been deployed, how many PRs (Pull Requests) have been raised, and how many events have been issued?
- How are events related and how are they ordered (links vs trace context)?

CDEvents Vocabulary

Most CI/CD platforms define their abstractions, data model, and nomenclature. The interoperability SIG has already been collecting this level of data from various platforms. Many labels are shared across platforms, but sometimes the same label bears different meanings in different projects. To achieve interoperability through events, a nomenclature with shared semantics across platforms was seen to be essential. This nomenclature has its roots in the "Rosetta Stone" for CI/CD first initiated through the Interoperability SIG in CDF. The CDEvents vocabulary will continuously revise its vocabulary based on the evolution of that document and related publications, until the first official release of the CDEvents protocol specification is published.

To achieve shared semantics, the CDEvents project first created a vocabulary describing four 'buckets' to group the different but common CDEvents together.

- Core Events: this includes core events related to core activities and orchestration that need to exist to be able to deterministically and continuously be able to deliver software to users.
- Source Code Version Control Events: Events emitted by changes in source code or by the creation, modification, or deletion of new repositories that hold source code.
- Continuous Integration Pipelines Events: includes events related to building, testing, packaging, and releasing software artifacts, usually binaries.
- Continuous Deployment Pipelines Events: include events related to environments where the artifacts produced by the integration pipelines actually run. These are usually services running in a specific environment (dev, QA, production), or embedded software running in a specific hardware platform.

Within each 'phase,' a few abstractions have been defined. For instance, the 'Core Events' phase defines "Task Runs" and "Pipeline Runs". The 'Continuous Integration Pipeline Events' phase defines "Build", "Test Case", "Test Suite", and "Artifact".

These phases can also be considered as different profiles of the vocabulary that can be adopted independently. Also notice that the term 'pipeline' is used to denote a pipeline, workflow, and related concepts. We also use the term 'task' to denote a job/stage/step.

With the vocabulary defined, CDEvents can be easily assigned to each phase. Within each phase, abstractions can be assigned. For instance, the Core phase defines *"Task Runs"* and *"Pipeline Runs"*. A *Pipeline Run* can be:

- Queued
- Started
- Finished

While these four 'phases' define the most common CI/CD activities, they are not exhaustive. In the future, other activities may be included, for instance, monitoring and incident management.

CDEvents Format

CDEvents can be encapsulated in different message/stream/event envelopes, and the first such binding prepared by the CDEvents project uses CloudEvents with CDEvents-specific extensions and payload structure, which is based on the CDEvent's vocabulary.

CDEvents producers may use the payload to provide extra context to the event's consumer. The payload however is not meant to transport large amounts of data. Data such as logs or software artifacts should be linked from the event and not embedded into the events. CDEvents follows the CloudEvents recommendation on event size and size limits.

All CDEvents contain information such as the type of event, the source of the event, the time the event occurred and a unique identifier. Depending on its type it also contains multiple other attributes, of which some are mandatory and some are optional.

For more information about the CDEvents format, please visit the CDEvents Documentation site.

CDEvents Use Cases

Use cases are key to understanding CDEvents. When defining CDEvents and their attributes, we must know what minimal set of information is needed to satisfy a particular use case. There are two root use cases:

- The first use case is interoperability, making it possible for one CI/CD tool to consume events produced by another without the need for 'static' imperative definitions. This use case focuses on how to make CI/CD tools work together in a more automated, streamlined manner.

- The second one is observability and metrics. Essential to improving the CI/CD pipeline is the ability of the pipeline to collect events from different CI/CD tools. This collection is essential for the pipeline to correlate CDEvents and process them consistently, building an end-to-end view of the overall CI/CD workflow.

Use Case One: Interoperability

In most enterprise organizations, it is impossible to have one CI/CD setup to rule all development projects. Different languages, platforms, and tooling might be required for each. For this reason, most organizations want to let teams choose their own optimal CI/CD setup. Let's consider the following use case. In our fictional organization, many of the software development teams prefer to use Zuul for its dependency handling and scalability. But other teams prefer GoCD. Some teams started using GitLab and prefer a central platform for source code and the project tools. Other teams prefer Jenkins and rely on a wide variety of plugins. To further complicate things, our fictional company doesn't build all the software in-house. They instead use suppliers.

As our fictional company needs to understand and receive software modules built using different tooling. The problem is that Zuul artifacts are a bit different from GoCD artifacts, which is a bit different from GitLab artifacts, or artifacts produced by a custom Jenkins build. The solution is to write custom translation or "glue code" to be able to understand and receive all these diverse built software modules.

And this diversity does n'ot apply only to building artifacts. I can apply to many steps in the pipeline including:

- Source changes
- Build activities
- Test runs
- Failures
- Compositions (multiple artifacts)
- Announcements

In a CDEvent-based system, it is unnecessary to develop custom 'glue code' for each activity. To allow our fictional company to announce new artifacts in a standardized format, CDEvents has predefined the format taking care of the interoperability between the diverse build systems.

Use Case Two: Visualization and Metrics

Consider the following CI/CD setup. Code is written and maintained on GitHub. When changes are made, they go through different tests, maintained by different teams, which use different technologies. Some tests are running in GitHub directly as GitHub Actions. Some others are executed in Jenkins and others as Tekton pipelines. Releases are managed through Tekton as well, while deployments are managed with Argo. Keptn is used to manage remediation strategies on production clusters.

If all these systems supported events in some predefined format, they could be easily collected. When they are not unified, teams must build event collectors that support multiple ways of collecting payloads. For example, both Tekton and Keptn use CloudEvents, but there is no shared semantics for interacting between them.

The goal is to have all platforms share the same format for events allowing a standard event collector across all tools and platforms managing CI/CD pipelines. For example, to visualize the flow of a change from when it's written, through the test, release, deploy, and possibly rollback, there needs to be enough information in the events to be able to correlate the data across all tools. CDEvents addresses unifying the data through a standard event collector.

Tracking metrics across the CI/CD Pipelines is critical to improving development processes by answering the question 'How effective is the DevOps setup.' To answer that question metrics need to be commonly defined, collected, and visualized. CDEvents collect data from heterogeneous sources, making it possible to store and process it consistently.

CDEvents Proof of Concept

The CDEvents contributors completed a Proof of Concept using Tekton and Keptn. The PoC shows a combined effort between Keptn and Tekton. In the PoC Tekton played the role of the pipeline executor doing the heavy lifting with building and deploying whereas Keptn handled the business decision. More information about the PoC can be found on the PoC GitHub page.

Your Next Steps (Call to Action)

Get involved in the CDEvents project at the Continuous Delivery Foundation. CDEvents will be critical as we move away from traditional monolithic development models to cloud-native models where decoupled applications require thousands of CI/CD workflows. Building a standard CDEvents protocol specification that can be easily supported by all CI/CD tooling is required. Contributing to the CDEvents team is a way for you to get involved in solving this critical piece of the CI/CD puzzle.

Get involved by going to https://cdevents.dev/community. You will find community information which is the easiest way to get started.

Conclusion

CDEvents is the next evolution of CI/CD pipeline orchestration and visualization. Every DevOps Engineer has understood the challenges of building 'plugins,' 'glue code,' and one-off scripts to build a single CI/CD pipeline. CDEvents will revolutionize the way pipelines are coordinated and unified providing the end-to-end CI/CD pipeline visibility and data collection needed for both processing and tracking key workflow metrics. And most critically, as we move into a cloud-native architecture with microservices, scaling the end-to-end CI/CD pipeline to thousands of workflows is already happening. CDEvents will allow your pipeline to scale, and make it easy to stand up a new workflow as often as needed.

Learn more at cdevents.dev

CD.FOUNDATION

About the Continuous Delivery Foundation

The Continuous Delivery Foundation (CDF) serves as the vendor-neutral home of many of the fastest-growing projects for continuous integration/ continuous delivery (CI/CD). It fosters vendor-neutral collaboration between the industry's top developers, end-users, and vendors to further CI/CD best practices and industry specifications. Its mission is to grow and sustain projects that are part of the broad and growing continuous delivery ecosystem.