CD.FOUNDATION

Scaling Up: Orchestrating CI/CD in a Large Organization

CHALLENGES
Interoperability, Automation

INDUSTRY
Analytics and AI

LOCATION
Cary, NC, USA

PUBLISHED
May 2025

AUTHORS
Andrew Larsen
Senior Software Developer, SAS

Brett Smith
Distinguished Software Developer, SAS

PROJECTS

cd events

- **CHALLENGE:** Improve the software release process to ship faster and more often to make the process less painful.
- **SOLUTION:** To create a provenance store that keeps track of events and would allow them to create, retrieve, and query events, event receivers, and groups of event receivers.
- **IMPACT:** The implementation resulted in improved automated testing, software promotions, security scanning and auditing, as well as, pipeline auditing.

BY THE NUMBERS

- 3000+ developers
- 185,500+ events sent per day
- Thousands of artifacts shipped per week

ABOUT SAS

SAS is a global leader in data and AI. With SAS software and industry-specific solutions, organizations transform data into trusted decisions. SAS gives you THE POWER TO KNOW®.

In the late 1960s, eight Southern United-States universities came together to develop a general-purpose, statistical software package to analyze agricultural data. North Carolina State had always been a leader in developing code for analyzing agricultural data, so it was a natural fit to house the project at the university's Cox Hall because the mainframe there could process enormous amounts of data.

The resulting program—the Statistical Analysis System—gave SAS both the basis for its name and its corporate beginnings. Since then, SAS has grown to become one of the largest privately held software companies in the world. Their focus is delivering value through their products and services while advancing education in data literacy and AI.

BACKGROUND

In 2019, SAS realized their R&D needed to move faster to stay competitive. Up until that point, the software release process consisted of two to three large-ship events per year and updates were slow and often painful.

To reverse that trend, the team set out to shorten the development cycle and deliver artifacts more quickly. The challenge was twofold:

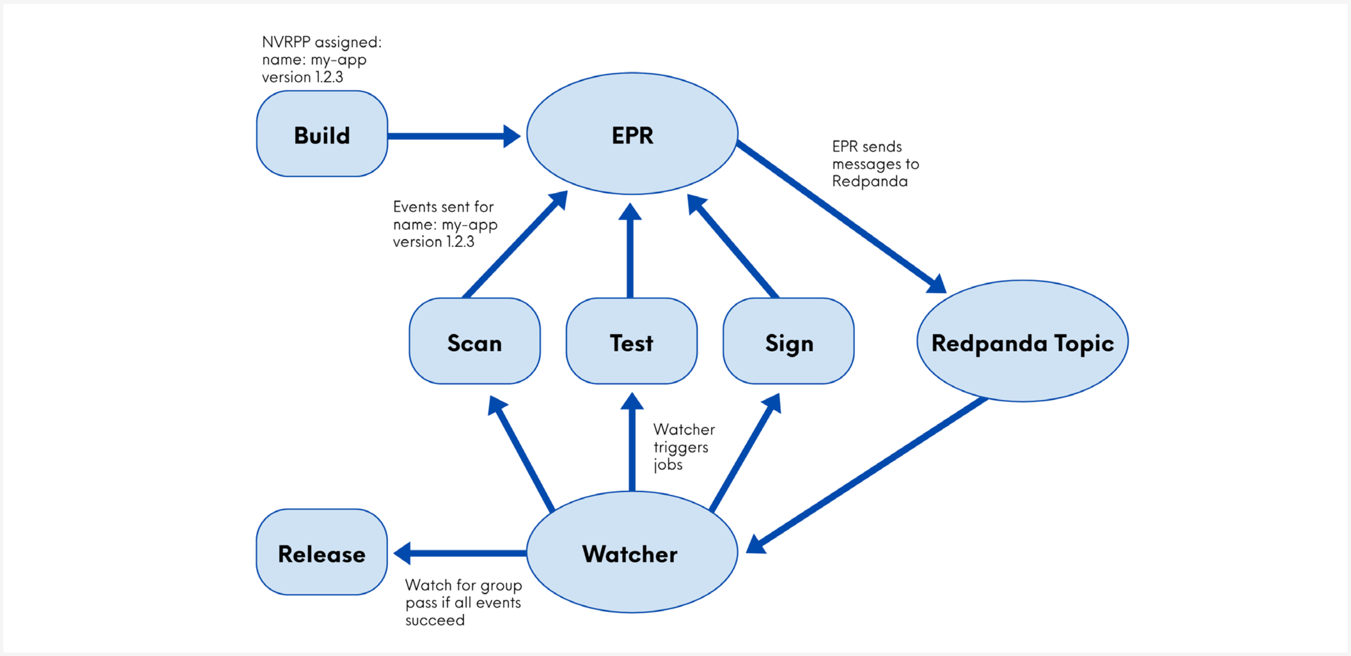
1. Create a system that facilitated disparate pieces of the pipeline to communicate and chain together.

2. Help R&D shift gears from the old software development model to Continuous Integration/ Continuous Delivery (CI/CD).

In a happy, imaginary world somewhere, the team could have chained together GitHub actions, or equivalent, into a working pipeline without needing to develop something tailored, but the reality wasn't so simple.

SAS's source code was (and still is) spread across several different source management systems and only a few of them had modern CI/CD features. To further complicate matters, the company's build system is old, some parts older than others, which makes it tricky to modernize. SAS also delivers most of its software as an Independent Software Vendor (ISV) rather than hosting it as a service—so they have to support the latest version in addition to other supported versions previously shipped to customers. To confront the myriad of problems, they needed an ecosystem-agnostic solution that was simple enough to work anywhere.

Their solution was Event Provenance Registry (EPR), or rather, the precursor to it, which can be compared to duct-taping a Raspberry Pi to a tractor. EPR is the glue that enabled the rest of the pipeline to really take off.


NVRPP assigned:
name: my-app
version 1.2.3
Build
EPR
EPR sends messages to Redpanda
Events sent for name: my-app version 1.2.3
Scan
Test
Sign
Redpanda Topic
Watcher triggers jobs
Release
Watcher
Watch for group pass if all events succeed

EVENT PROVENANCE REGISTRY

Event Provenance Registry is a culmination of several years of SAS's effort to convert from large-ship events to CI/CD. They built the first version internally to facilitate CI/CD in a complex, aging build system. The result enables SAS to build, package, scan, promote, and ship thousands of artifacts daily.

Event Provenance Registry (EPR) is a provenance store. It keeps track of events. With EPR, you can use the API to create, retrieve, and query events, event receivers, and groups of event receivers.

The Event Provenance Registry (EPR) is a service that stores events and tracks event-receivers and event-receiver-groups. EPR provides an API that lets you create events, event-receivers, and event-receiver-groups. You can query the EPR using the GraphQL endpoint to get identifying information about events, event-receivers, and event-receiver-groups. EPR collects events from the supply chain to record the lifecycle of a unit in the SDLC. EPR validates event-receivers have events. EPR emits a message when an event is received, as well as when an event-receiver-groups is complete for a unit version.

CHALLENGES

EPR was, and is, a successful project internally. That doesn't mean SAS didn't encounter problems. Here are a few issues they ran into that you can hopefully avoid if you choose this route for your organization.

LACK OF ACCESS CONTROL FOR RECEIVERS AND GROUPS

The first problem was the lack of restrictions on who could use receivers and groups. This meant that anyone could post passing events to any receiver, which in turn, could trigger any associated groups.

Add some lazy message matching, and suddenly you find yourself releasing thousands of artifacts without intending to (yes, this truly happened). The team discussed adding serious Role Based Access Control (RBAC) to receivers and groups but decided not to in favor of development speed. Now that EPR is open sourced, they intend to implement a more robust solution soon.

DIFFICULT ADOPTION

Once the hard work of writing EPR was done, getting the rest of the company to adopt the fancy new tool should have been easy, in theory, but it was the next hurdle.

People don't like change and developers are no different. SAS discovered that developers especially don't like being handed a box of virtual LEGO bricks and told to "use these tools to integrate with EPR." Many developers prefer to live in a world where they don't need to worry about the intricacies of DevOps, in addition to their normal work.

The average developer has no idea how EPR works, and they prefer to keep it that way—which is the mindset the EPR team had to combat. To get them to adopt it, they had to make it as minimally intrusive as possible. Forcing people to learn new technology tends to make them complain, which leads to management pushback.

For a smooth transition, make sure you have management backing you and make it easy for people to adopt your technology. The battle is as much political as technical.

LAZY RECEIVER SCHEMAS

In the interest of speed, the team formed the habit of filling out their receivers with empty JSON schemas. While perfectly valid by EPR standards, this type of lazy schema validation set them up for some nasty problems later. There were many cases where they started running analytics on events sent to a particular receiver—a good example is processing security scan results stored in the event. The problem is that without a schema to validate the event contents, nothing prevents a user from posting complete garbage to your receiver. Empty schemas are fine for development, but not so great in production.

INCONSISTENT EVENT TYPES

When the development for EPR started, there were, and still are, no standard event types. Users are largely left to determine their own event types. This can lead to confusion where events for two separate groups mean the same thing but are named differently. CDEvents would have been useful in solving this problem had the team known about it at the time. Consistent messaging makes it much easier to tell what's going on, especially when disparate groups produce millions of events.

CDEVENTS

Receivers can have multiple events that correspond with them. Any events associated with a receiver must have a payload that complies with the schema defined on the receiver. This allows some guarantees about what kind of data you can expect of events going to any given receiver.

Enter the CDEvents spec. This spec provides a standard on which to build interoperability between CI/CD systems. CDEvents promote a system-agnostic approach, where any system speaking the normalized CDEvents vocabulary can respond to events from any other system without prior knowledge about the originating system. This facilitates interoperability and flexibility.

Integrating CDEvents with EPR allows efficient event exchange and processing. It is straightforward to integrate when the target technology already understands CDEvents. Messaging systems can handle event propagation, while Channel Adapters and Message Translators help legacy systems share and consume CDEvents without built-in support.

BENEFITS

It wasn't all doom and gloom. Most of the challenges were overcome and SAS saw massive improvements in several key areas.

GREATLY IMPROVED AUTOMATED TESTING

One of the biggest improvements EPR allowed was the mass automation of the integration test suite. SAS ships dozens of microservices that need to be tested together. The test automation team leveraged EPR to control microservice deployments and test result collection. Using event receiver groups, they gate the promotion of artifacts based on test results.

AUTOMATED SOFTWARE PROMOTIONS

Automated promotions go hand in hand with testing. Like many other companies, SAS divides its artifacts into different promotion levels (dev/test/prod) based on their ship readiness.

They wrote a watcher to handle artifact promotion based on event receiver group completion. That way, they can ensure that artifacts are only promoted if their criteria pass, which could include passing integration tests, clean security scans, stakeholder signoff, etc. If you're willing to write the automation, there's no limit to what you can do.

AUTOMATED SECURITY SCANNING AND AUDITING

EPR messages are used to trigger several types of security scans. Combined with special test containers that examine the test results, SAS can prevent artifacts from shipping if they don't pass their scans. Watchers can then create work tickets for the artifact owners. Not only that, but they also use EPR messages to coordinate the delivery of scan results for further analysis with other tools. This has dramatically reduced SAS's remediation time and allows us to locate any CVE (Common Vulnerabilities and Exposures) in their codebase within minutes.

PIPELINE AUDITING

In the same vein as security scanning, EPR provides an immutable record of how an artifact traveled through the pipeline. Failed pipeline tasks are tracked by NVRPP and the team can see exactly what event failed. From an auditing standpoint, the team can prove they did their due diligence with security scan results and signoffs. Every pipeline action is tracked.

LOOKING FORWARD

The EPR project has now been open sourced under the Apache 2.0 license. We still have features to add like an RBAC, OIDC, Event signing, and a Rust SDK. There is an open source Python SDK and a workshop for learning EPR. Contributions are welcome, and we hope to get things rolling again soon.

CD.FOUNDATION