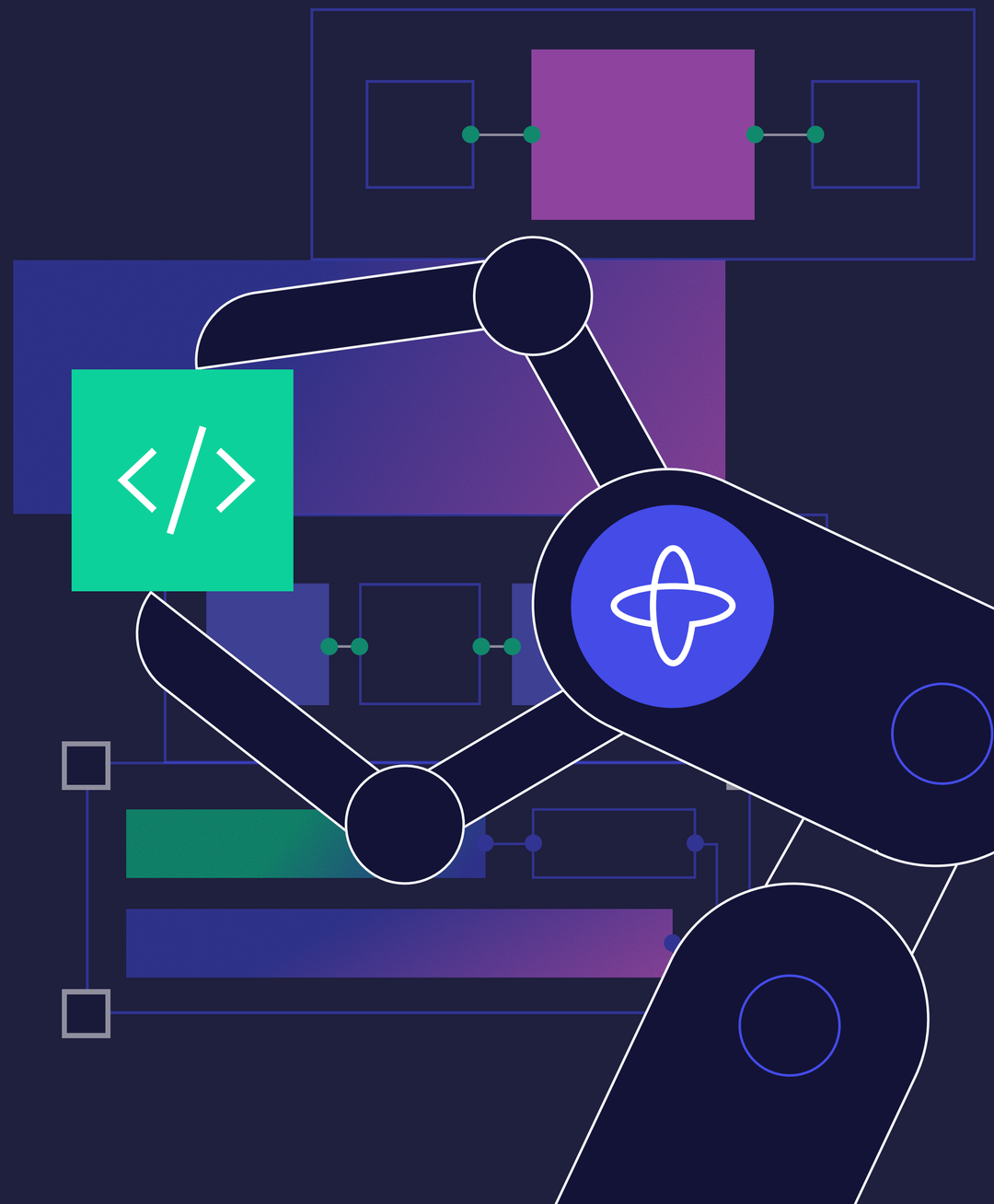
Building reliable platform automation with Temporal

How platform teams use Temporal to build automation that survives failures



Temporal

Table of contents

Summary		03
The Problem	IDP automation keeps breaking	04
	Why scripts, queues, and DIY fall short	05
The Temporal solution	Durable Execution for platform engineers	06
	Key benefits for platform engineering teams	07
	Temporal in the IDP architecture	08
	Temporal can improve everything else	09
Conclusion		10

Summary

When it comes to workflow automation, platform teams need to automate task execution, manage dependencies, and connect disparate applications to work together efficiently.

But what happens if something fails silently, gets stuck in retry loops, or completes incorrectly? The platform team becomes the bottleneck they were hired to eliminate.

Temporal provides platform teams with reliable, durable execution for their IDP automation. Temporal Workflows are written as code and can survive infrastructure failures and maintenance operation disruptions automatically, while providing visibility into every step of the process.

THE RESULT

Golden Paths that actually hold up. Developers who can self-serve. A platform team that spends time building, not firefighting.

THE PROBLEM

IDP automation keeps breaking

Picture a Senior Platform Engineer, let's call her Maya. She's been building the team's IDP for two years and she's proud of it. Developers can now provision and scale new microservices without opening a ticket, deployments are mostly self-service, and the catalog UI actually works, all because of her hard work.

On paper, it looks exactly like what platform engineering is supposed to produce.

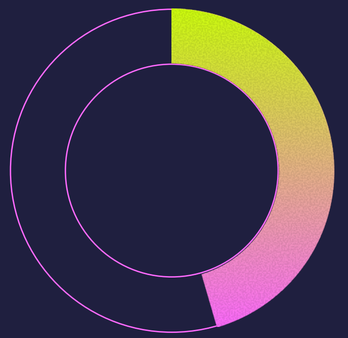
But failure is always a possibility. It can look like this:

- A microservice provisioning job fails halfway through, leaving orphaned resources behind
- A database upgrade stalls with no visibility into where it stopped or how to resume
- Certificates aren't replaced in time and cause service disruptions and security vulnerabilities
- A compliance approval workflow gets dropped because the service handling it restarted
- A script enters an infinite retry loop and requires someone to manually intervene

In this instance, let's say the problem is the provisioning job failure.

The domino effect happens at light speed. Now, the system is in an incomplete state: only some of the Terraform resources have been applied, and the service entry was created in the catalog, but the namespace never came up in Kubernetes. And, worst of all, our poor platform engineer has to spend the next four hours of precious engineering time reconstructing what happened from logs and dashboards spread across five systems. After that, she'll need to do the tedious work of manually cleaning up half-baked resources and then explain to her team why a workflow that was supposed to be "self-service" required two engineers to untangle in the end.

Even the best Golden Path is only as reliable as what runs underneath it.



45.5%

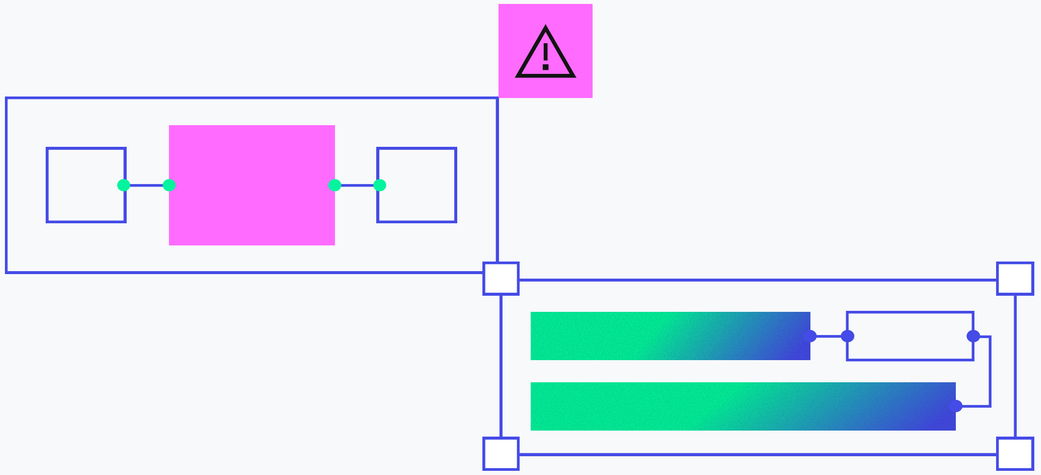
of platform teams with dedicated, budgeted staff still operate in a primarily reactive mode.

Source: 2025 State of Platform Engineering Report (Volume 4)

Why scripts, queues, and DIY fall short

Most teams' first instinct is to build a platform themselves. It’s easy to start with scripts, cron jobs, queues, or lightweight microservices, and they work well for simple automation tasks.

The issue is that IDP automation isn’t simple, nor is it stateless. The gap between what these tools can handle and what production actually demands is where platform engineering hours silently disappear.



What DIY may not give you

Durability
When a script fails, we don’t know how far it got. Progress state is lost.

Intelligent retries
Custom retry logic is fragile. Unfortunately, infinite loops, silent failures and, worst of all, inadvertent DDoS events are the norm, not the exception.

Observability
When a job fails, knowing exactly which step failed, why it failed, and what state the system was in requires reconstructing from scattered logs and system interfaces across multiple systems.

Long-running operations
Scripts are not designed to run for days or weeks. Compliance workflows, lease management, and staged rollouts require something fundamentally different.

The bitter truth is that most platform teams spend significant engineering time building reliability infrastructure (retries, state management, failure recovery) that has nothing to do with their core product. Temporal handles that layer so they do not have to.

THE TEMPORAL SOLUTION

Durable Execution for platform engineers

Let’s check back in with Maya now.

The provisioning failure she had looks completely different now.

- The workflow doesn’t leave any orphaned resources
- If the Kubernetes namespace step fails, Temporal retries it automatically (with full context intact)
- If the workflow process goes down mid-execution, the Workflow picks up where it left off when it comes back

Now, there’s no need to spend four hours in the logs. When she opens the Workflow in the UI, she’ll see exactly which Activity failed and know when. If necessary, she can make fixes and allow the workflow to complete its processes.

This is Durable Execution in motion.

Temporal is a Durable Execution platform. It acts as a stateful control plane for your IDP, handling retries, persistence, failure recovery, and timeouts automatically. Platform engineers write the business logic. Temporal handles everything else.

Workflows

Define the complete, reliable logic for any platform operation: service provisioning, teardown, compliance checks, upgrade sequences. Workflows are code, version-controlled, and execute durably from start to finish regardless of infrastructure failures.

Activities

Execute the everyday actions within a Workflow: calling Terraform, running kubectl, hitting cloud APIs. Activities are individually retried on failure without restarting the whole Workflow.

Visibility and the Web UI

Give you a complete event history for every Workflow execution. See exactly what ran, what failed, what retried, and why, in real time. Debugging a broken provisioning job goes from "check the logs across five systems" to "open the Workflow in the UI."

Signals and Queries

Let you interact with running Workflows externally. Trigger state transitions, query current state, and handle asynchronous requests from the IDP catalog or CLI without polling.

Key benefits for platform engineering teams



Reliable automation, without the reliability code

Temporal guarantees Workflow execution through failures. Platform engineers write the automation logic, not the retry logic, state management, or failure handling underneath it.



Long-running operations, natively

Workflows can span minutes, days, or months. Compliance audits, canary deployments, staged rollouts: no custom persistence layers required.



Real developer self-service

Reliable Golden Paths mean developers can provision, deploy, and manage services without opening a platform team ticket. The platform team stops being the bottleneck.



Full observability, out of the box

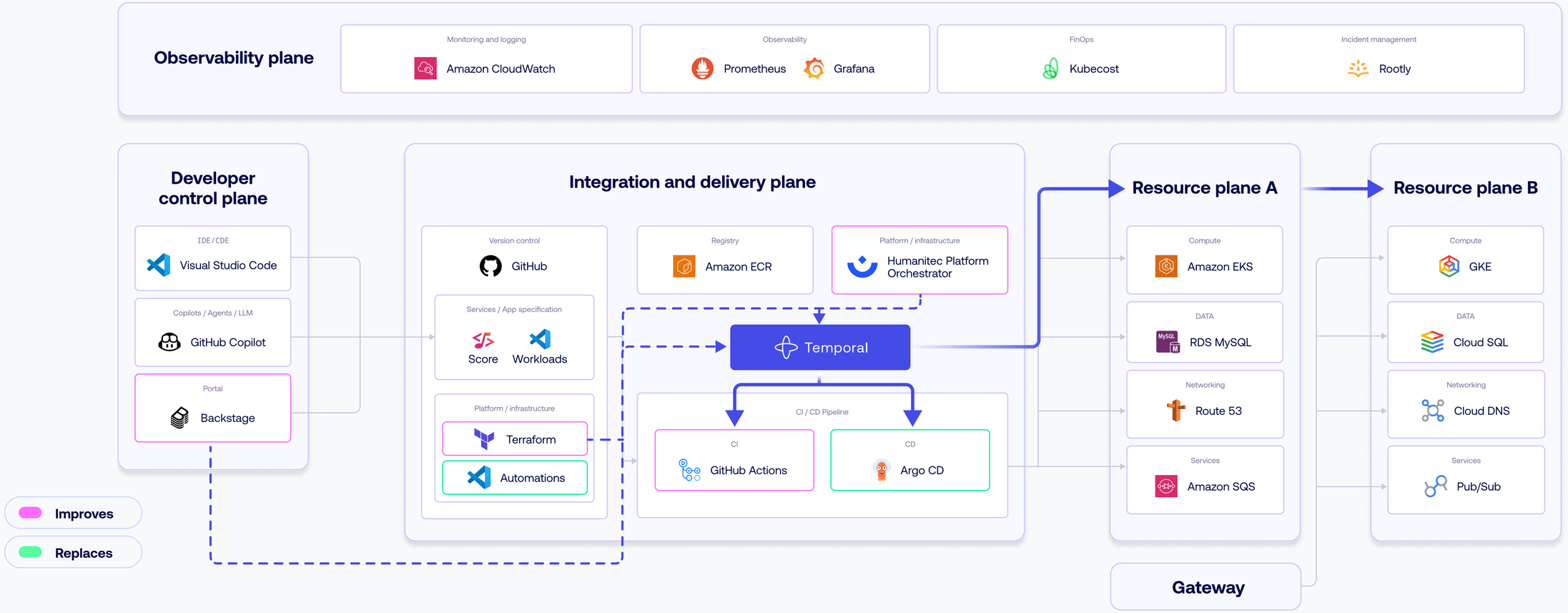
Every execution is logged with a complete event history. Instant audit trail for compliance, and a clear view of exactly what happened when something goes wrong.



One layer to orchestrate everything

Chain Terraform, Helm, cloud APIs, and internal tools into a single, cohesive Workflow. Consolidate the mess of scripts and queues into a reliable execution layer.

Temporal in the IDP architecture



Temporal can become your durable platform orchestrator

Temporal replaces your platform orchestrator with durable, code-defined workflows, so canary rollouts, approval gates, and cross-cluster deployments become programmable logic instead of stitched-together tooling.

Temporal can improve everything else

IaC and Automation

Temporal wraps your infrastructure-as-code workflows with durable orchestration, so multi-environment applies, post-provision validation, and drift remediation run reliably without custom scripting holding it together.

Developer Portal

Temporal powers the workflows behind your developer portal, so service provisioning, access requests, and onboarding flows run durably from trigger to completion without dropping state.

Incident Management

Temporal drives your incident lifecycle, so auto-remediation, escalation timers, and runbook execution run as durable workflows with a full audit trail.

Identity and Policy Control

Temporal orchestrates your access and compliance workflows, so time-bound permission grants, certificate rotation, and policy exception approvals run on schedule with automatic expiry and a built-in audit trail.

Implementation roadmap

01

Phase 1: Start with your most brittle Workflow

Pick the one automation that breaks most often and requires the most manual intervention. Rewrite it as a Temporal Workflow. This becomes your proof of concept and internal case study (and, honestly, your best recruiting tool for getting the rest of the team on board).

02

Phase 2: Build workflows behind your service catalog

Use Temporal's Signal and Query primitives to power asynchronous request handling from your IDP catalog. Developers submit requests; Temporal tracks and executes them durably.

03

Phase 3: Migrate long-running operations

Onboard compliance workflows, security patching, upgrade sequences, and any automation that currently requires manual intervention or custom state management.

04

Temporal Cloud vs. self-hosted

Teams can start with the open-source, self-hosted version. Temporal Cloud removes the operational burden entirely: managed upgrades, 99.99% SLA, and zero infrastructure to run.

You’re building a product that the rest of engineering depends on. That product deserves a foundation that holds.

Like Maya, developers want to spend their time building things that make an impact, not fixing broken systems or fighting fires. The difference between shipping a platform your developers trust and one you spend days firefighting comes down to the tooling.

Temporal turns complex, distributed automation into predictable, durable code.

Platform teams that run on Temporal stop absorbing the failure modes of brittle scripts and queues, and start shipping the platform their developers actually need.

See how Temporal simplifies your most complex Golden Path

TALK TO AN ENGINEER →

Or explore on your own

START A FREE TRIAL →