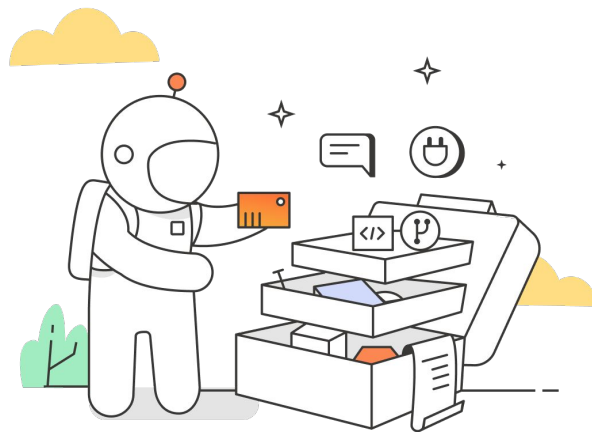




Supercharging Asynchronous workflows with Temporal.io



Samith Bharadwaj | @samithbharadwaj

Software Engineer, Postman

The screenshot displays the Postman interface within a workspace named "Samith's Test Space". The left sidebar shows a tree view of collections and environments. A tooltip is visible over the "Salesforce Platform APIs" collection, indicating it was forked from "Salesforce Platform APIs" by "Samith Bharadwaj's fork".

The main panel shows a pull request for the feature "Feature: Adding Rest endpoints for Platform Events". The source is "Salesforce Platform APIs" and the destination is "Salesforce Platform APIs".

Feature: Adding Rest endpoints for Platform Events OPEN

Source: Salesforce Platform APIs → **Destination:** Salesforce Platform APIs

Description

Endpoints

- Fetch Platform Event Schema by Event Name
- Fetch Platform Event Schema by Schema ID

Changes

3 entities added.

- Salesforce Platform APIs
 - Platform Events Copy added
 - Platform Event Schema by Event Name added
 - Platform Event Schema by Schema ID added

Platform Events Copy added

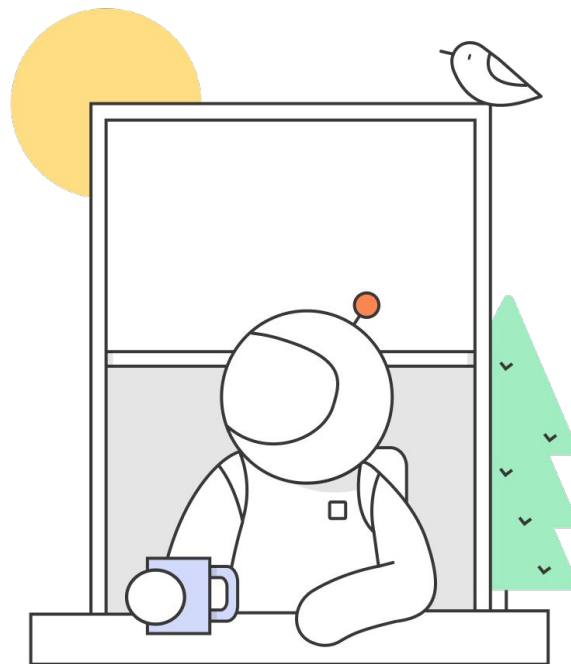
Name

- Platform Events Copy

Understanding Pull Requests in Postman

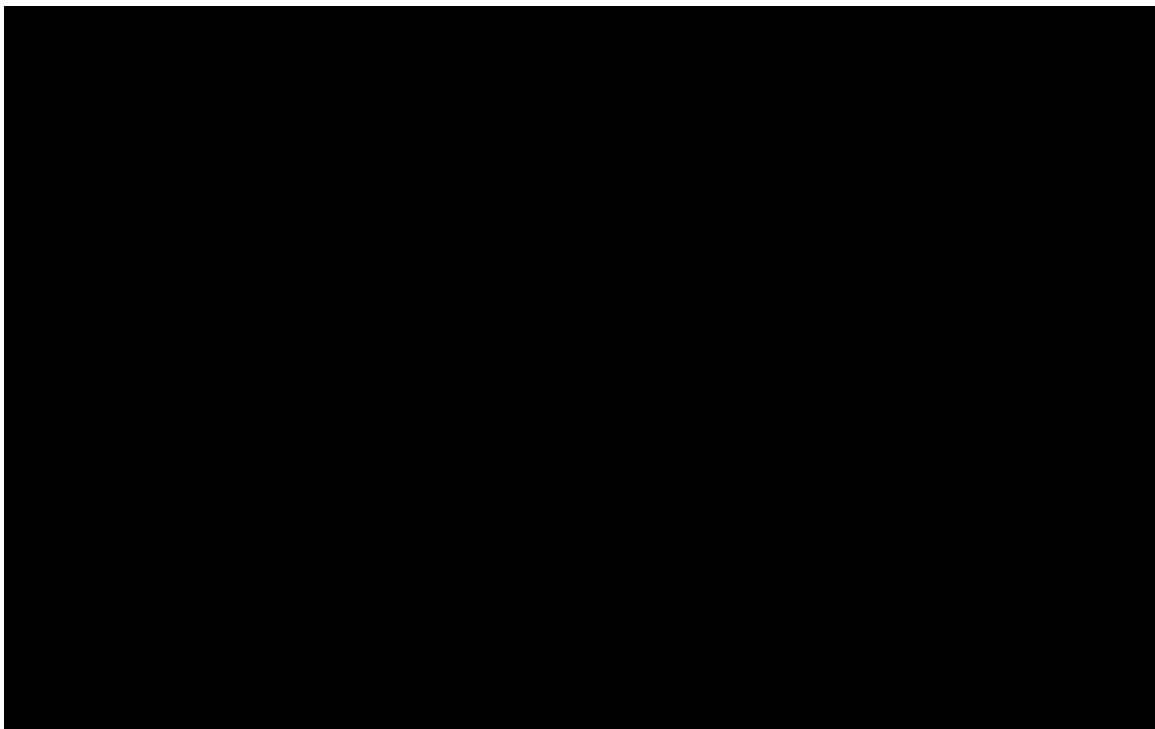


Unpacking the Problem





PR Merge Experience: Before Temporal.io





Merge Inefficiencies and problems

- **Product experience is time and UI blocking**

Merging large PRs can result in user frustration due to extended wait times before being able to use the product

- **Merge a PR is unreliable**

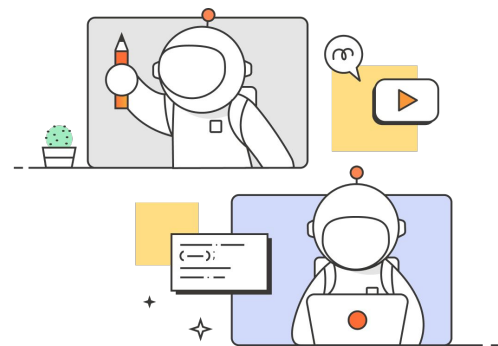
With merging being a complicated distributed transaction, it often lead to unpredictable failures

- **Downstream service errors result in the entire merge process returning a failure**

Lacked necessary mechanisms to recover from dependent service failures

- **Non deterministic end states reached due to low predictability**

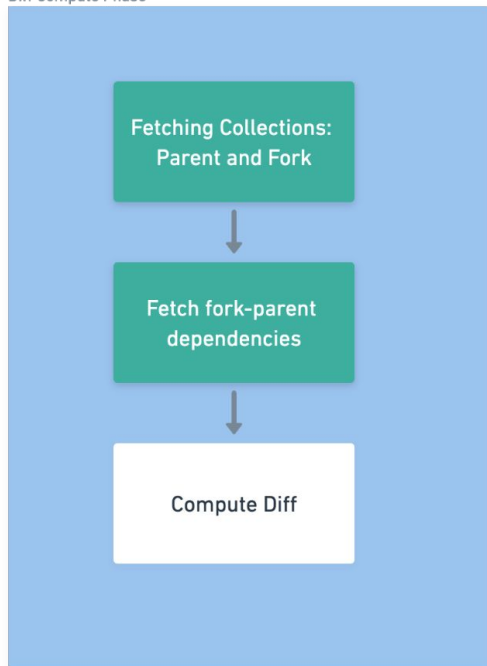
False positives and negative merge completion statuses confused the user



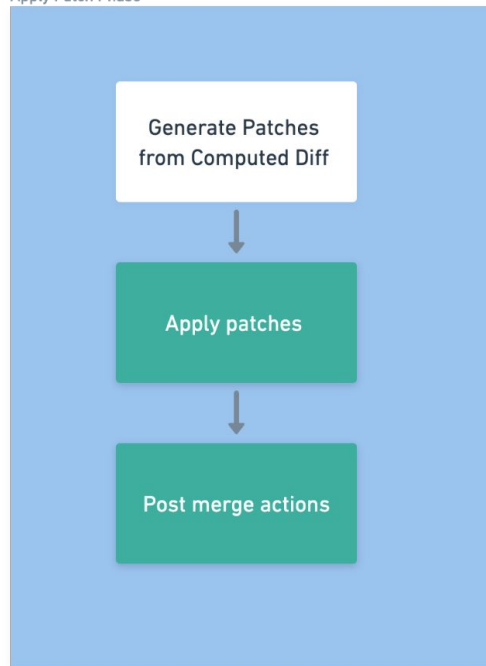


PR Merge Internals

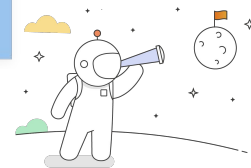
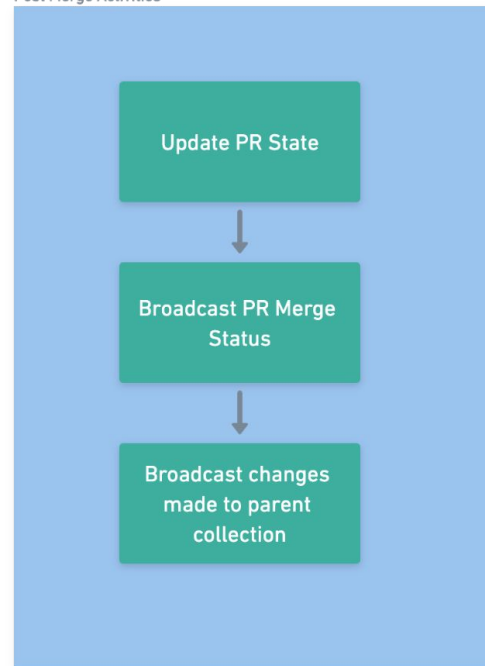
Diff Compute Phase



Apply Patch Phase

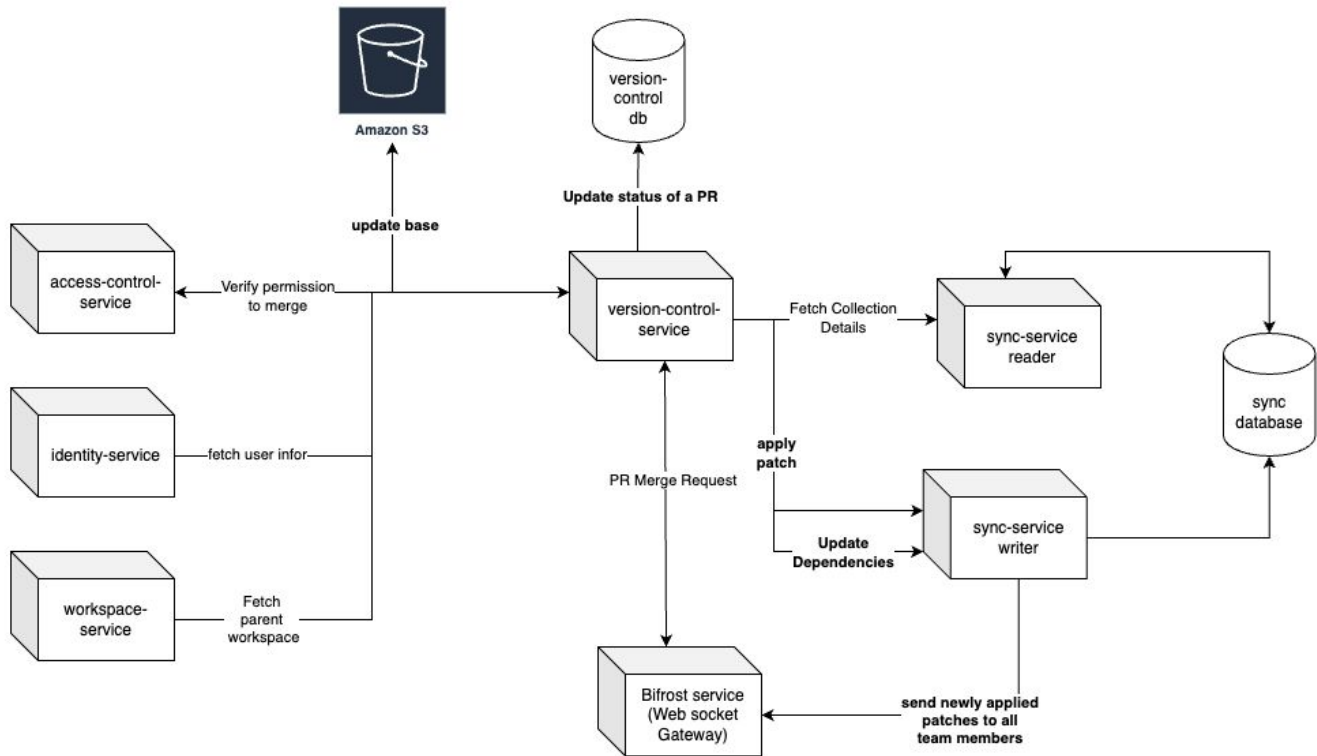


Post Merge Activities





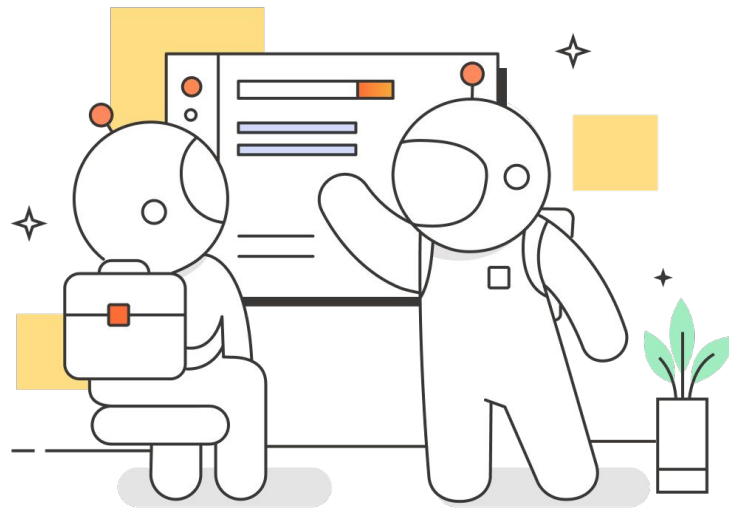
PR Merge Internals





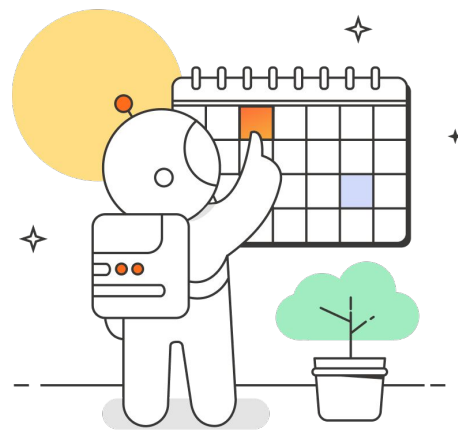
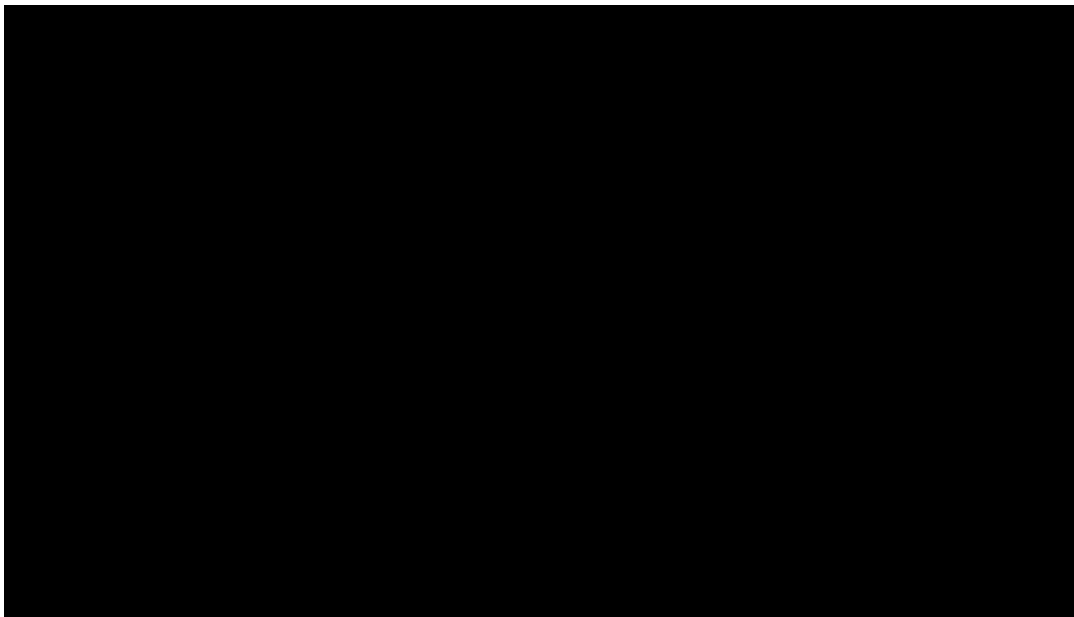
Crafting an effective solution

- **Making the PR merging experience asynchronous**
With Temporal, we can establish a queue and worker setup to make the process async.
- **Reliable distributed transactions**
Retries and Timeouts at an activity level provided by temporal help with increasing reliability across the entire process
- **Observability and Event Sourcing**
With temporal, observability is provided out of the box along with effective state tracking for workflows
- **Code/Service reusability**
Business logic from existing services should be reusable in the async workflows.



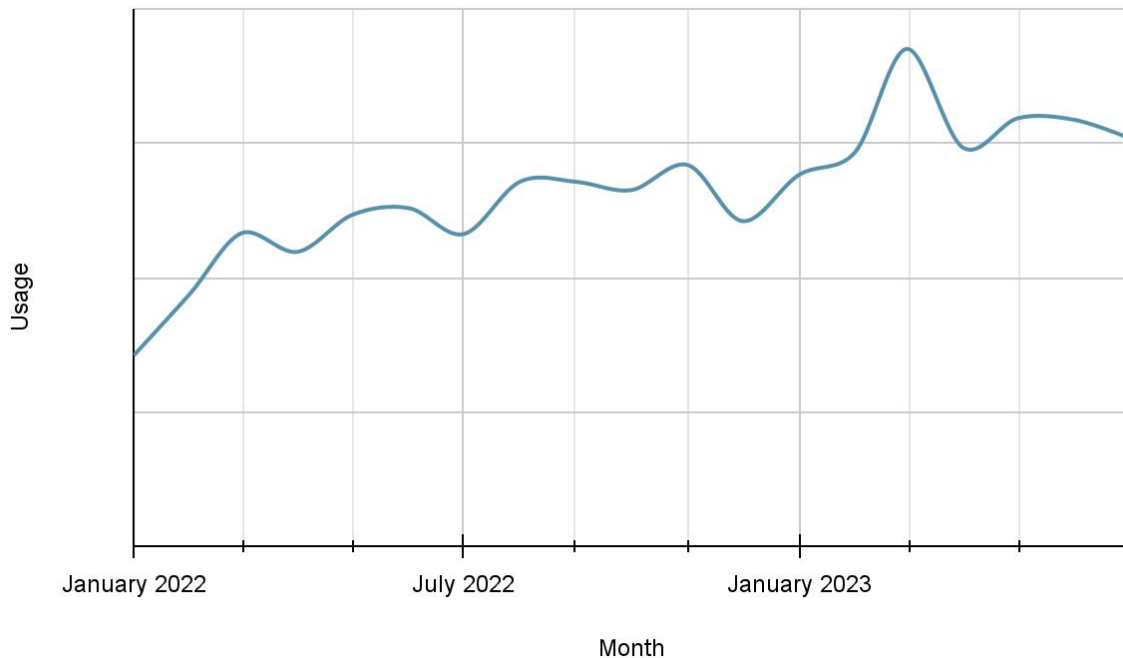


Merge experience after using Temporal





Post Temporal Integration: Pull Request Usage



- **Increased adoption of pull requests**
PR adoption rate increased by **30.71%** after the introduction of the new merging experience
- **Decrease in merging failures**
With Temporal, retries and rollbacks of transactions have helped bring down the error rate by **43%**





Future of Temporal @ Postman

- **Internal SDK to use temporal across teams and services**

With increased adoption of temporal within Postman, the sdk helps with easy onboarding, standardising security standards, data exchange formats, etc.

- **Increased adoption for various use cases**

Higher throughput and critical workflows are now using temporal within Postman. Some of them are:

- Deleting, Replicating a workspace (~200k workflows and ~1M activities p.m)
- Publishing, Deleting an API(~50k workflows and ~1M activities p.m)





Thank you

