How Temporal Simplified Checkr Workflows

This case study is based on an interview with Ben Jacobson who was a key player in Temporal adoption within Checkr*

Problem:

In the last decade, the growth of the gig economy has put background checks in the critical path of employers such as Uber, Lyft, Grubhub, etc. Traditionally, background checks have been discriminatory, tedious, and inaccurate. Checkr was founded to usher in a new era of automated background checks, bringing reliability and consistency to a historically unreliable and inconsistent process.

Running a background check is a multi-staged process which consists of both automated and manual steps. An automated step might be reaching out to a microservice to do a search in a database we maintain or sending a request to a court researcher at a court house. This might be followed by analyzing each record via another microservice and, depending on the result, escalating to a quality assurance expert to manually review a record before sending downstream. This whole process could take minutes, hours, or even days.

How was this solved before:

Although validation is a background process which can take days, it's often the blocker for slow background checks. This means validation is a critical process and Checkr needs a scalable method of managing a huge number of parallel validations. Furthermore, the moment any validation finishes, appropriate action needs to be taken based on the result.

Until switching to Temporal, Checkr solved this problem using an in-house solution powered by standalone databases and Kafka queues. As this was not a general purpose workflow system, Checkr developers were also required to implement a complex state machine to ensure the consistency of the validation

process. It didn't take long for the true cost of this in-house solution to become apparent:

- It's homegrown nature meant that there were no official mechanisms provided for updating the state machine running in the live system.
- New hires were immediately required to study and understand the Checkr specific state machine architecture, regardless of the team/area they were hired for.
- Even when things went as expected, the system was not as reliable as desired.

A Clear Need for Change:

Checkr was confronted with the challenge of finding a solution that would fit their arbitrary-length workflow needs and scale with their growth. A number of candidate workflow systems were considered:

- Zeebe
- AWS Step Functions
- Netflix Conductor
- Apache Airflow

And judged against key requirements:

- vendor lock-in
- scalability
- maturity
- DSL

Going into the process, there was a strong belief that a DSL (Domain Specific Language - such as SQL) based workflow solution was going to be the best bet. After Checkr engineers prototyped some workflows in code with Temporal, they started to reconsider the need for a DSL. The short experiment had convinced them that understanding and testing workflows written as code was fundamentally easier than with a DSL.

> After prototyping workflows in code using Temporal, we refocused our efforts towards that because of the easier understanding and testability of those workflows

Migrating to Temporal:

While Checkr was fairly confident in their choice to use Temporal, it wasn't practical to migrate all of their workflows at once. Instead, specific components and flows within the system were migrated one by one. Each migration meant refactoring a single section of their data pipeline to use Temporal workflows instead of the existing Kafka based solution. This enabled the team to migrate at their own pace, without affecting the live system.

In addition to migrating existing aspects of the pipeline, Checkr began encouraging engineers to build new features and flows with Temporal. Teams started to notice the increase in productivity and reliability that Temporal offers, and internal adoption grew rapidly. Today all new data sources used during background checks are implemented with Temporal. This is not a requirement but a choice by the Checkr developers.

> All new data sources incorporated into our background checks are now done via Temporal, and more specifically by choice of the engineering team working on it.

Looking back:

Since switching to Temporal, the way we think about the Checkr product has simplified and the happiness of developers has increased. Learning to model their problems as workflows and activities actually helped clarify the core product. Modeling things as workflows and activities makes inter-team sharing possible, meaning that code is continuously reused and not continuously reinvented.

> Developer happiness has increased. Thinking about our problems in terms of workflows and activities has clarified our product and now allow us to share workflow components with different teams.

The team has also noticed benefits from the switch that they never initially expected. Observability in both development and production has dramatically improved. Testing distributed parts of the system together not only became possible but enjoyable.

> Being able to see step by step what is happening, what path a workflow took, is very valuable.

Whats next:

Developers at Checkr who use Temporal love the technology. Usage has grown outside the core use case and new teams are evaluating it for future projects. It's possible that the entire background process might one day even be a Temporal workflow.

Working with Temporal:

> Great, the team has been very helpful when we have questions, the slack community is very active and we've even been able to contribute to the project with thoughtful feedback.