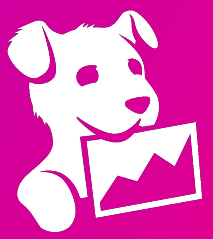
Workflow Automation And Temporal

October 2023



DATADOG

What is Workflow Automation?

DATADOG

Datadog

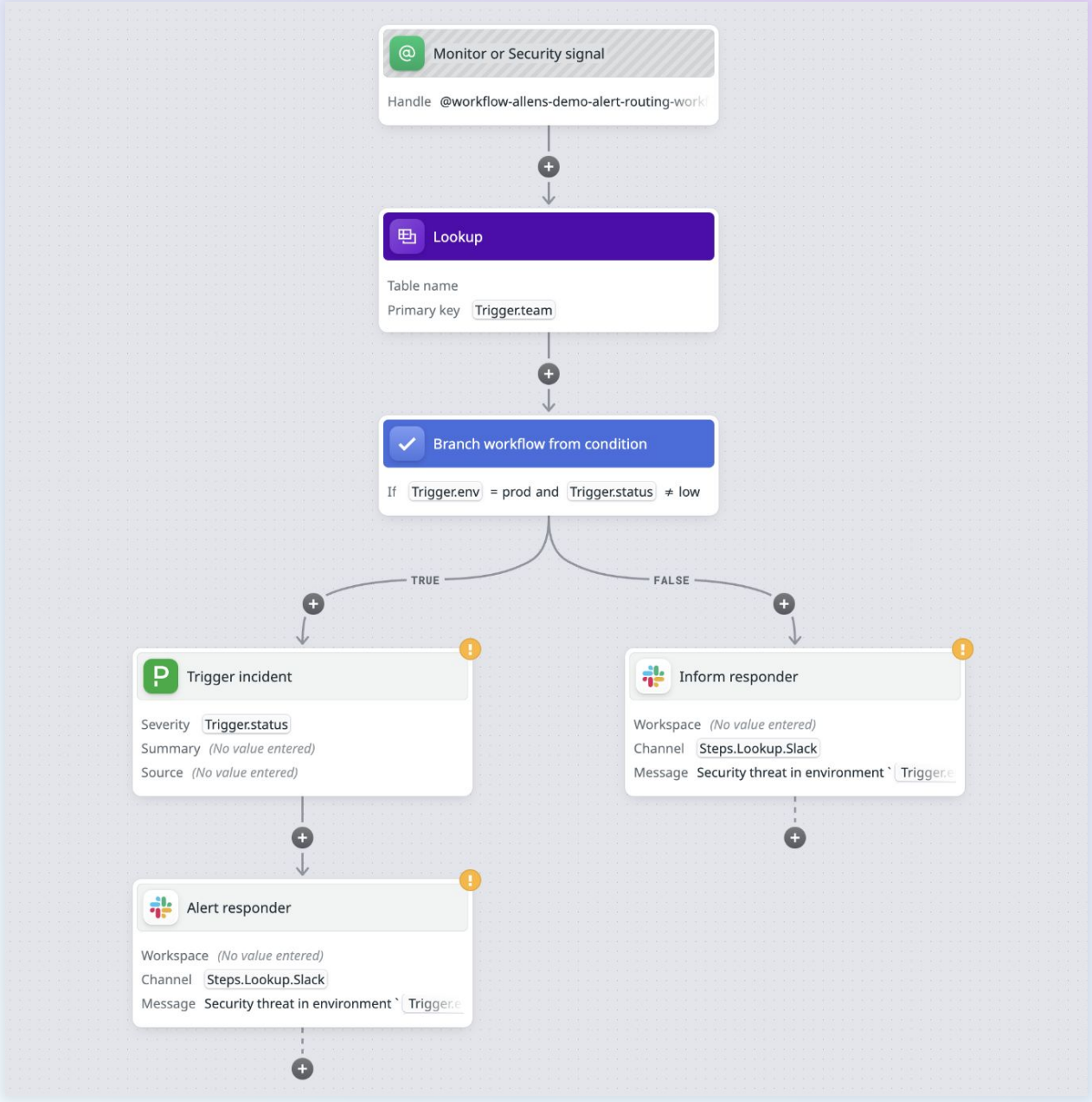
Response

Vision

Easily build execution flows that hook into signals and respond and remediate using Datadog and third-party integrations

Targeting

Initially, SRE and Security



Features

Arbitrary DAGs

Branching

Iteration

Split/Merge

Error-handling
(fallbacks, retry policies)

Integrations

300+ actions

AWS, Azure

Pagerduty

Slack, Teams

HTTP, Javascript

OpenAI

Github, Gitlab

JIRA

And more!

Run from

Monitors

Security Signals

Cloud Cost Mgmt

Dashboards

Schedules

And more!

Security and Control

RBAC

Credentials Store

Decisions

Datadog Audit Trail

Features

Arbitrary DAGs

Branching

Iteration

Split/Merge

Error-handling
(fallbacks, retry policies)

Integrations

300+ actions

AWS, Azure

Pagerduty

Slack, Teams

HTTP, Javascript

OpenAI

Github, Gitlab

JIRA

And more!

Run from

Monitors

Security Signals

Cloud Cost Mgmt

Dashboards

Schedules

And more!

Security and Control

RBAC

Credentials Store

Decisions

Datadog Audit Trail

Features

Arbitrary DAGs

Branching

Iteration

Split/Merge

Error-handling
(fallbacks, retry policies)

Integrations

300+ actions

AWS, Azure

Pagerduty

Slack, Teams

HTTP, Javascript

OpenAI

Github, Gitlab

JIRA

And more!

Run from

Monitors

Security Signals

Cloud Cost Mgmt

Dashboards

Schedules

And more!

Security and Control

RBAC

Credentials Store

Decisions

Datadog Audit Trail

Features

Arbitrary DAGs

Branching

Iteration

Split/Merge

Error-handling
(fallbacks, retry policies)

Integrations

300+ actions

AWS, Azure

Pagerduty

Slack, Teams

HTTP, Javascript

OpenAI

Github, Gitlab

JIRA

And more!

Run from

Monitors

Security Signals

Cloud Cost Mgmt

Dashboards

Schedules

And more!

Security and Control

RBAC

Credentials Store

Decisions

Datadog Audit Trail

Features

Arbitrary DAGs

Branching

Iteration

Split/Merge

Error-handling (fallbacks, retry policies)

Integrations

300+ actions

AWS, Azure

Pagerduty

Slack, Teams

HTTP, Javascript

OpenAI

Github, Gitlab

JIRA

And more!

Run from

Monitors

Security Signals

Cloud Cost Mgmt

Dashboards

Schedules

And more!

Security and Control

RBAC

Credentials Store

Decisions

Datadog Audit Trail

Building Workflow Automation

DATADOG

So, you want to build a Workflows product…

Your customers:

Have arbitrary DAGs

Want to use on-prem and SaaS resources

Expect visibility into workflow executions

And you need:

An architecture that allows fast iterations

To quickly and easily develop integrations to external systems

Scalability and reliability

Multiple daily deployments without customer impact

So, you want to build a Workflows product…

Your customers:

Have arbitrary DAGs

Want to use on-prem and SaaS resources

Expect visibility into workflow executions

And you need:

An architecture that allows fast iterations

To quickly and easily develop integrations to external systems

Scalability and reliability

Multiple daily deployments without customer impact

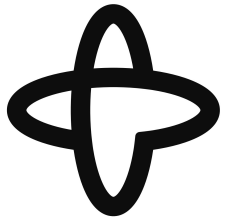
Use Temporal!

Tailor-made for this use-case

Allows us to iterate fast in the face of changing product requirements

Components

Datadog workflow ‘state-machine’



Temporal

Event Store

Integration Platform

What we're building on

Worker Framework

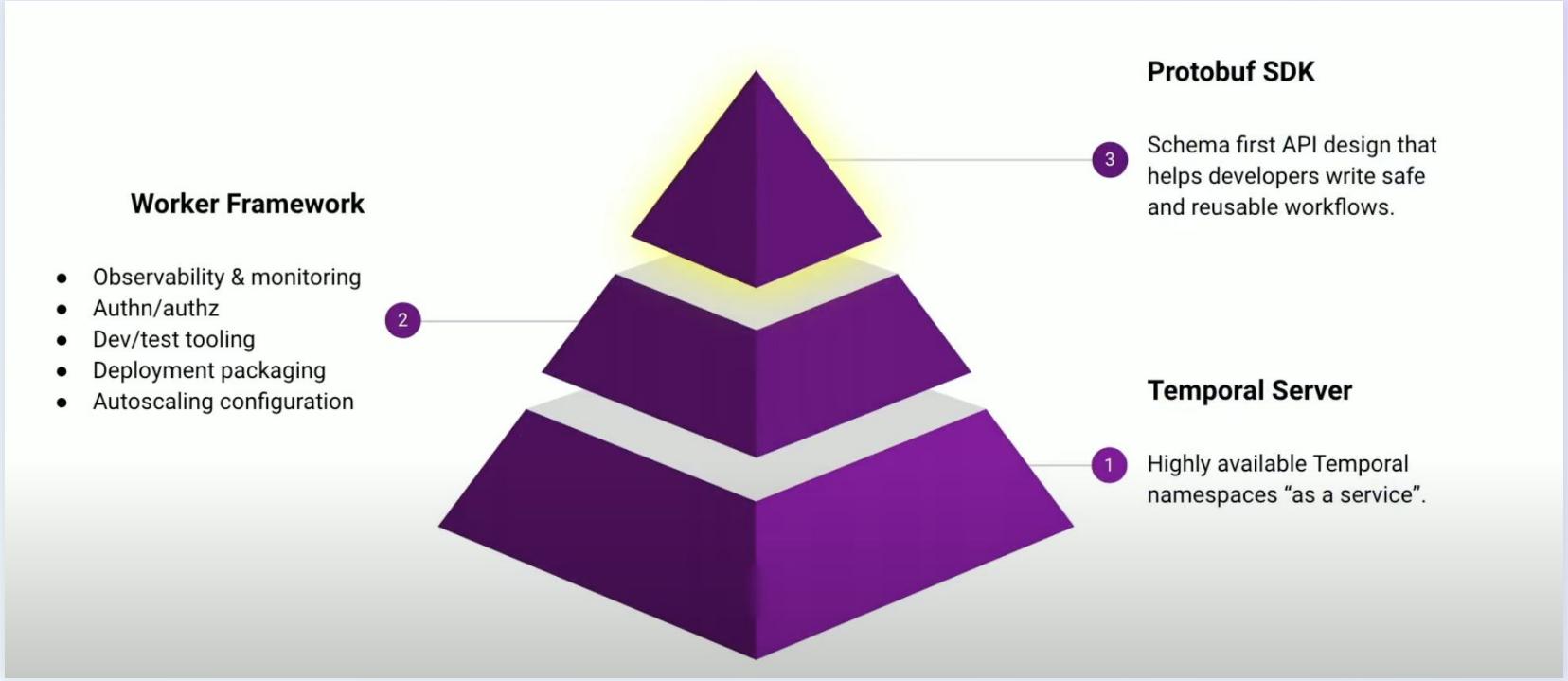
Abstracts away a lot of the operational minutiae of workers

Protobuf SDK

Define Workflow and Activity interfaces

See presentation by Jacob LeGrone @ Datadog

https://youtu.be/LxgkAoTSI8Q



Structuring workflows: Problem

Business problem →Temporal Workflow

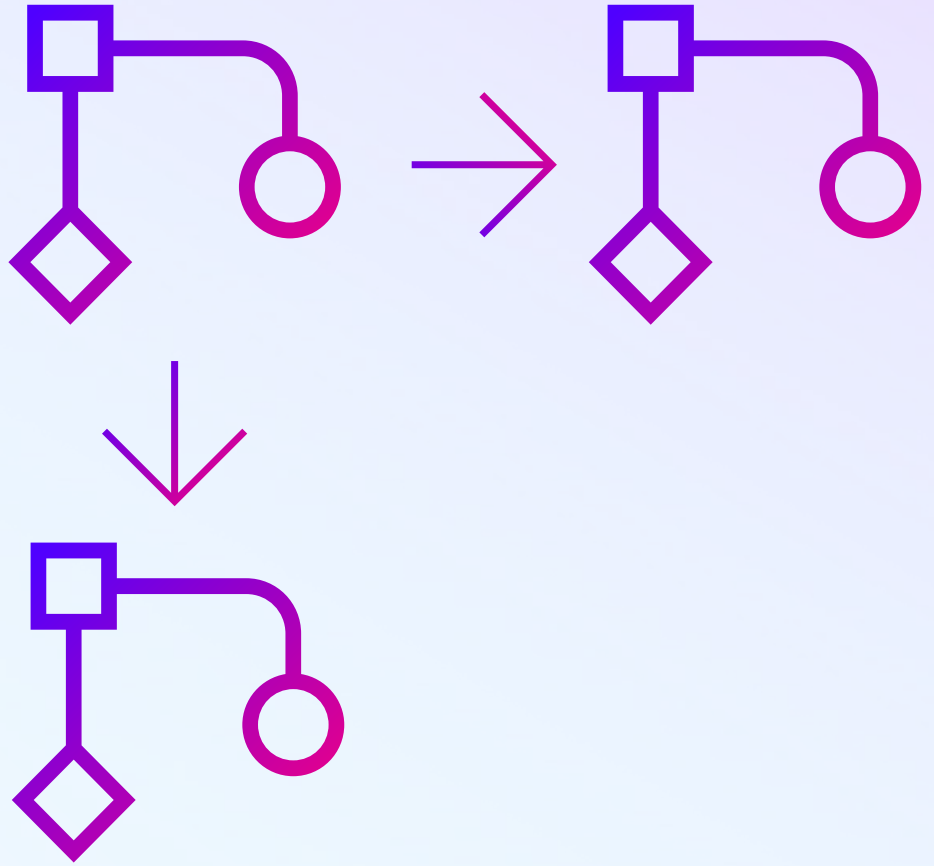
Customer workflows run in ‘state machine’

Ideally, a single Temporal Workflow

- 50K event history limit!
- Hard to reason from code → # of events
- Can’t support “Continue As New” for parent

Decompose into multiple workflows

- Local activities (⅓ the events)
- Limited parent lifetime
- Plus: separate operational failure domains



Structuring workflows: what we learned

Business problem →Temporal Workflow

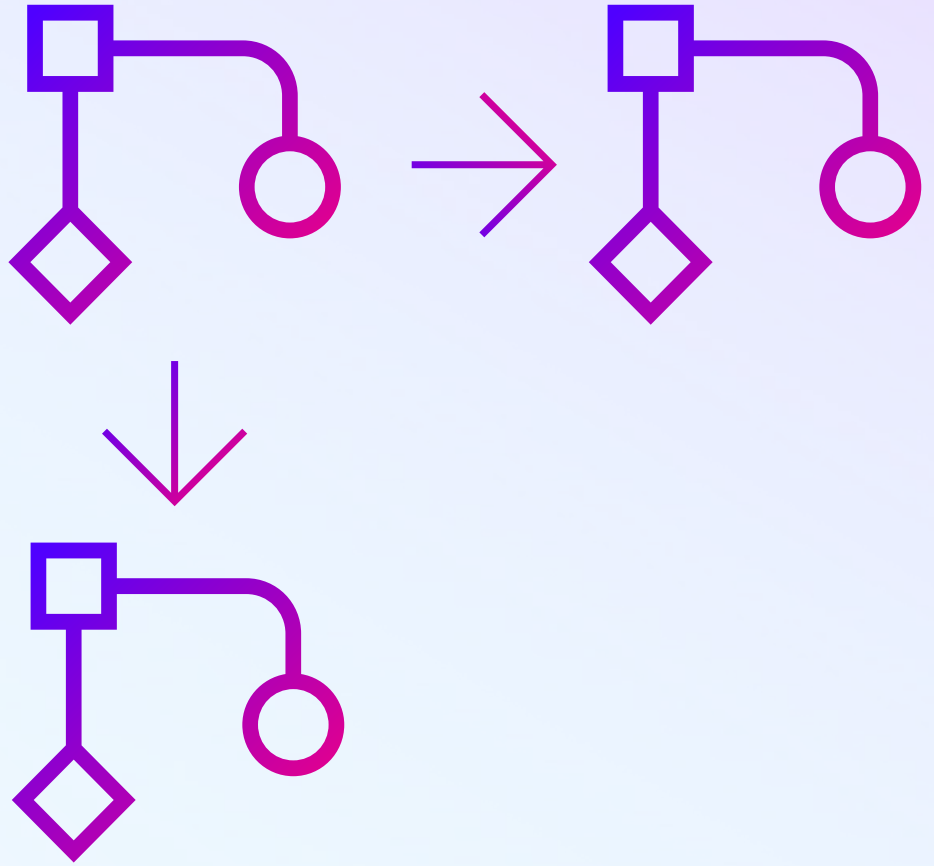
Customer workflows run in ‘state machine’

Ideally, a single Temporal Workflow

- 50K event history limit!
- Hard to reason from code → # of events
- Can’t support “Continue As New” for parent

Decompose into multiple workflows

- Local activities (⅓ the events)
- Limited parent lifetime
- Plus: separate operational failure domains



Activity timeouts and retries

Transient failures happen!

Initially, we were “choose your own adventure”

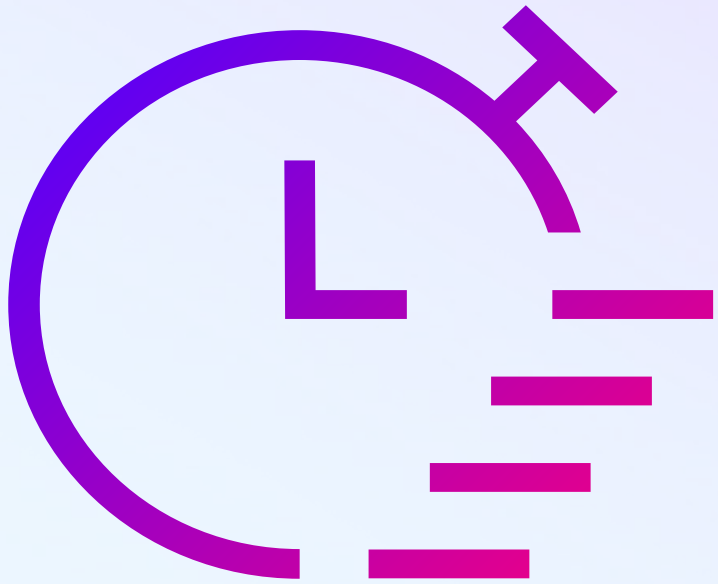
Now defined, named `ActivityOptions`

- Bucket classes of interactions with systems
- Forces thoughtfulness about the parameters we used, and how we went about adding to this set

Remember, retry policies are a tradeoff!

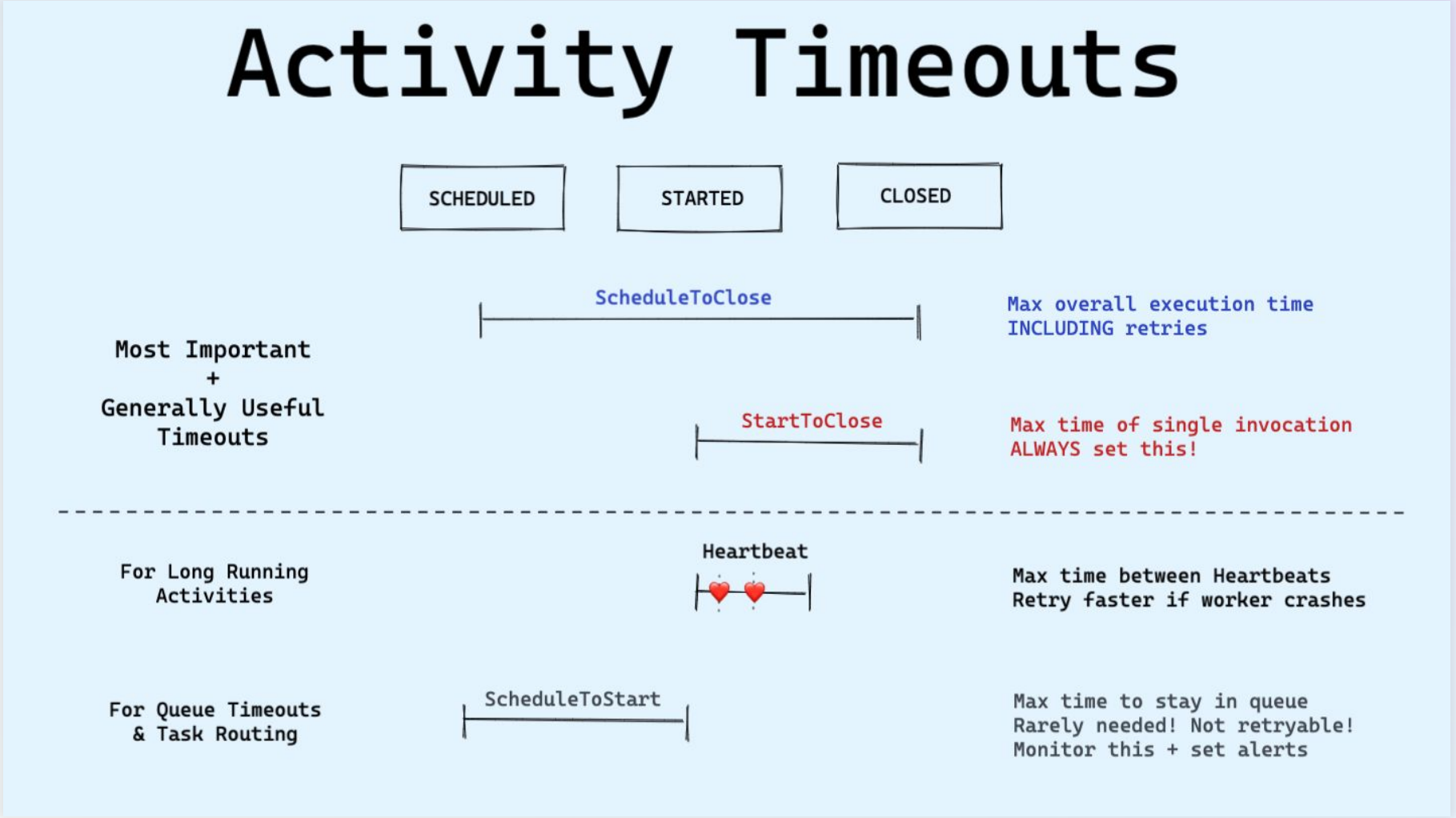
- How long to let an operation stall
- How long till we can mitigate the impact

Strongly suggest gamedays!



Activity Timeouts

Temporal blog post: https://temporal.io/blog/activity-timeouts



Feature Flags

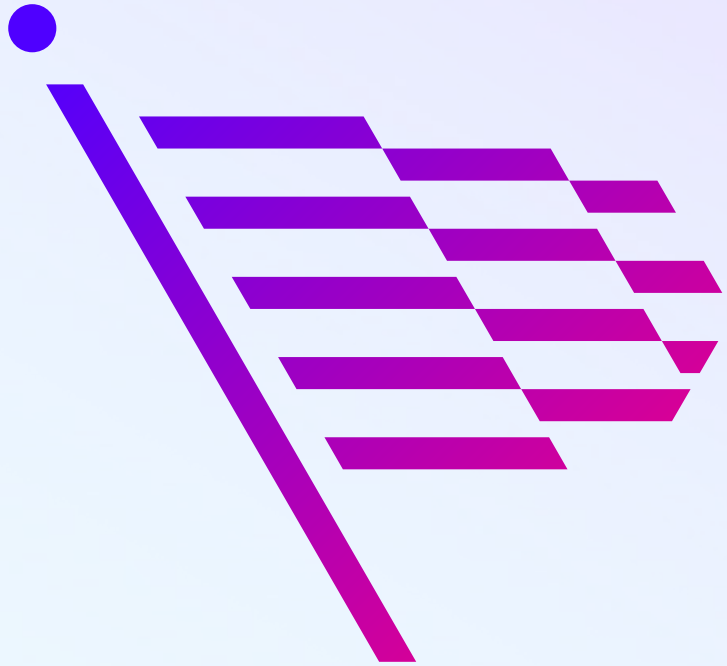
Incremental rollouts, per-customer options…

Default approach would be inline resolution

- Activity
- Protos
- Retry policies
- Versioning
- Feels like a lot of ceremony…

Resolve upfront *prior* to Temporal submission

- Include in Temporal Workflow input
- Trivially replayable



Versioning

Opinionated runtime change management

Use multiple processes to avoid errors

- Extensive code reviews
- Staged rollouts, monitor gates
- Currently implementing replay tests

Surprising sources of non-determinism

- Generated code (ex: validation)
- Interceptors
- Utility functions

Can be *very* hard to track down error cause, especially in complex workflows!

Need replay tests

Temporal: 👍👍