An introduction to

Temporal & Durable Execution

Temporal

20 years in the making

2004

Max is Tech Lead on **Simple Queue Service** (SQS) at Amazon

2009

Max & Samar lead launch of **Simple Workflow Service** (SWF) at AWS

2014

Samar leads Azure Service Bus, creates **Azure Durable Functions** at Microsoft

2015

Max & Samar reunite to create the **Cadence Project** at Uber

FOUNDED: 2019

Max & Samar fork Cadence and co-found **Temporal**

TODAY

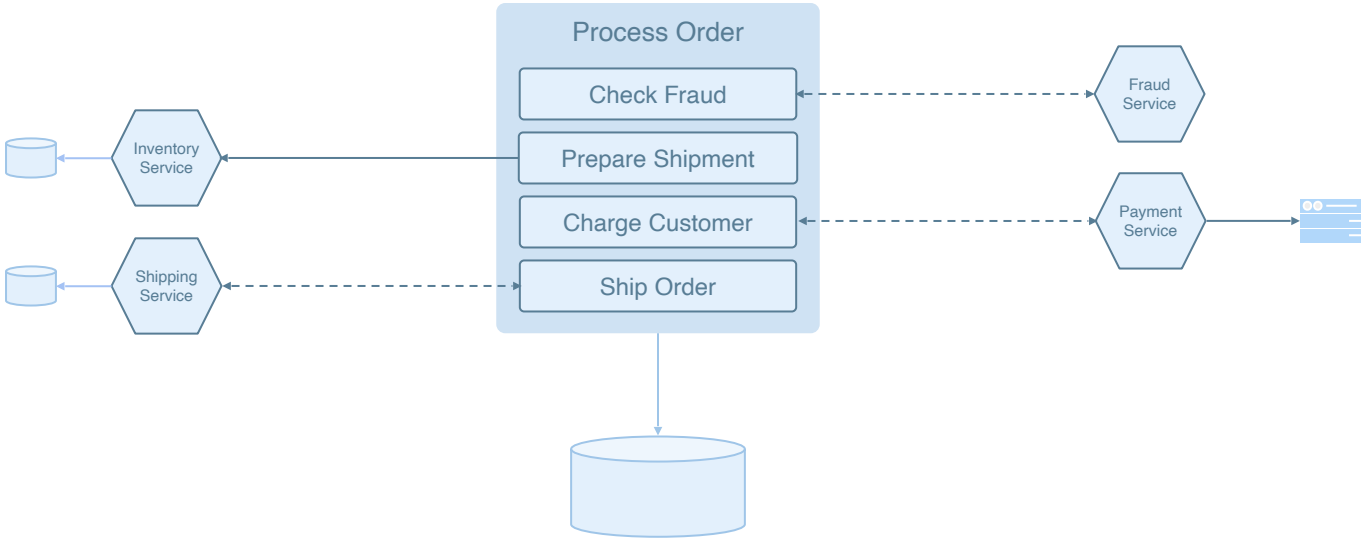
300,000+ open source users

1,500+ Temporal Cloud customers

Alaska, ADP, NETFLIX, Office Depot, COMCAST, SailPoint, salesforce, DATADOG, ANZ, American Eagle Outfitters, citrix, cruise, samsara, vodafone, POSTMAN, sunrun, LayerZero, twilio, cisco, VISA, Airbyte, Unity, PURESTORAGE, Grab, Yum!, SAP, Adobe, NVIDIA, EPIC GAMES, DELL, shopify, Retool, BLOCK, airbnb, Fanatics, HSBC, TURO, paloalto NETWORKS, instacart, coinbase, NORDSTROM, coupang, qualtrics XM, usbank, snyk, Walmart, SoFi, ROCKET Money, HashiCorp, Equipment Share

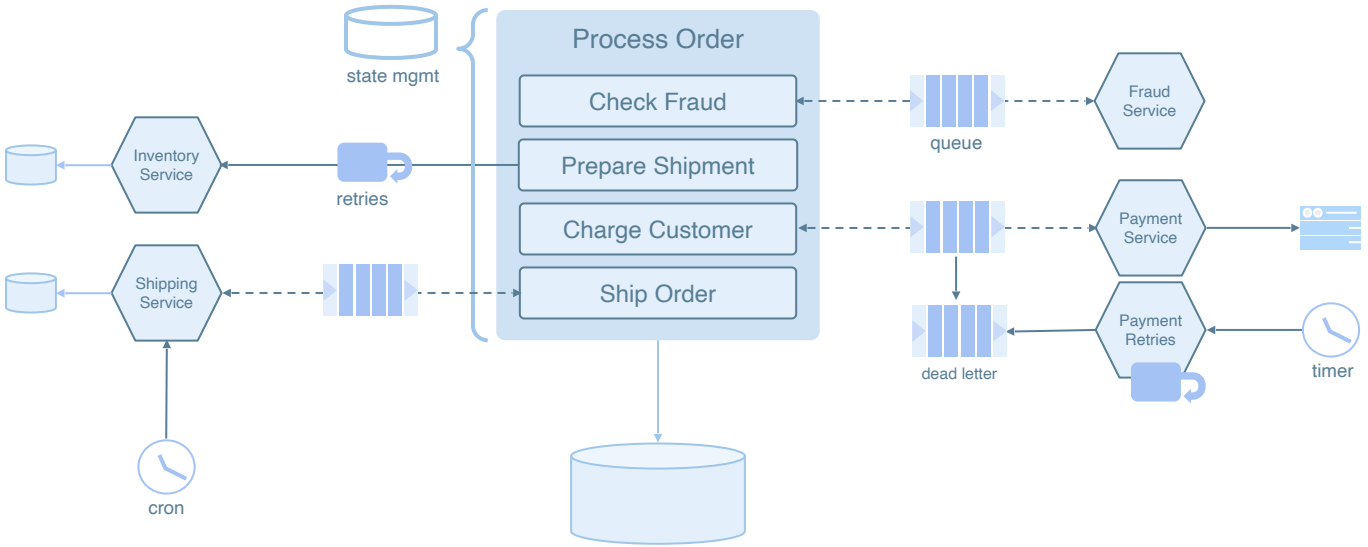
MODERN APPLICATION Architecture

If we just focus on business logic, our apps are seemingly simple



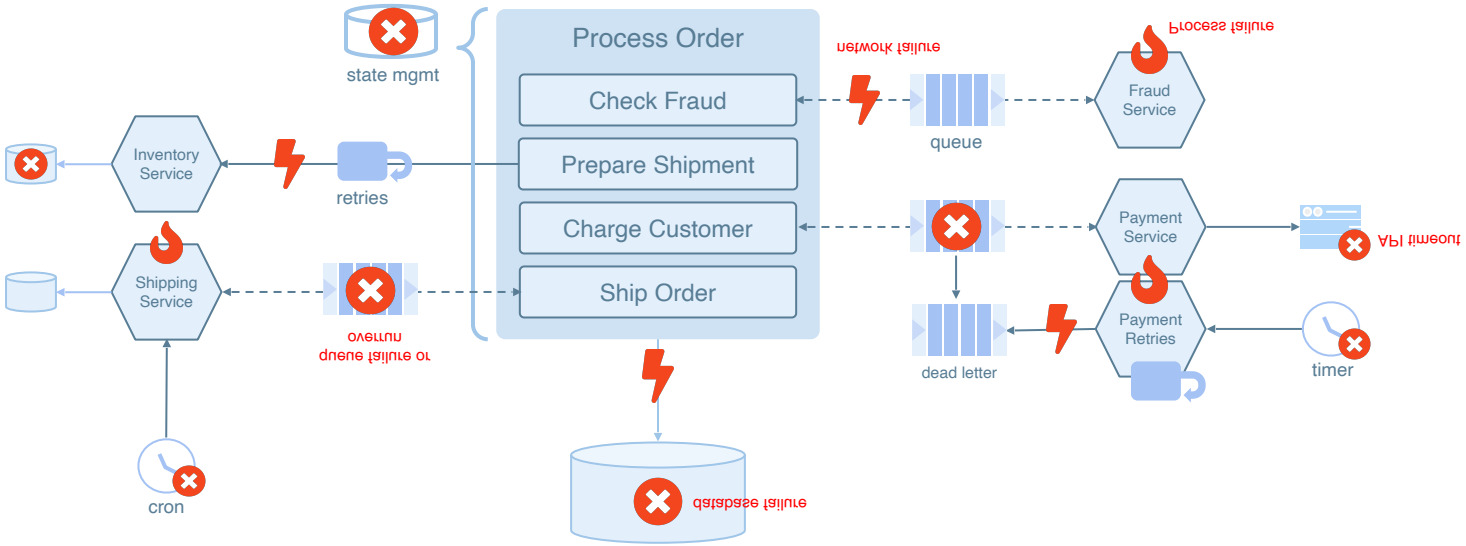
MODERN APPLICATIONS IN PRODUCTION

However, reliability, scale, and data integrity add deployment complexity



MODERN APPLICATIONS IN REALITY

Everything fails and we need to code for these added complexities

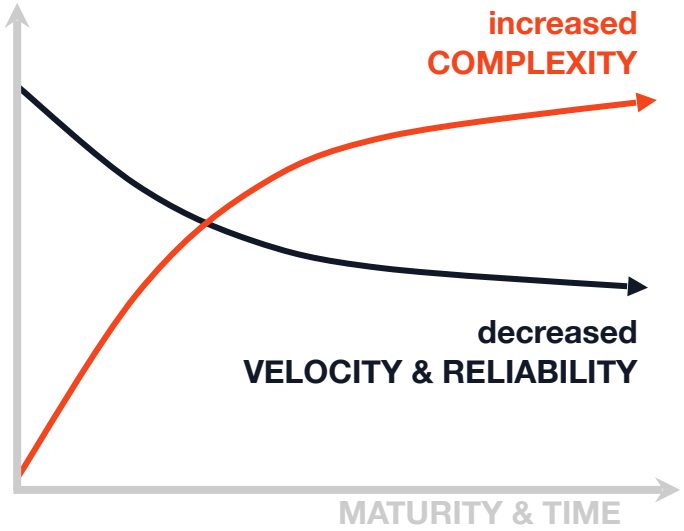


MODERN APPLICATIONS - MODERN PROBLEMs

As application complexity increases, velocity and reliability decrease

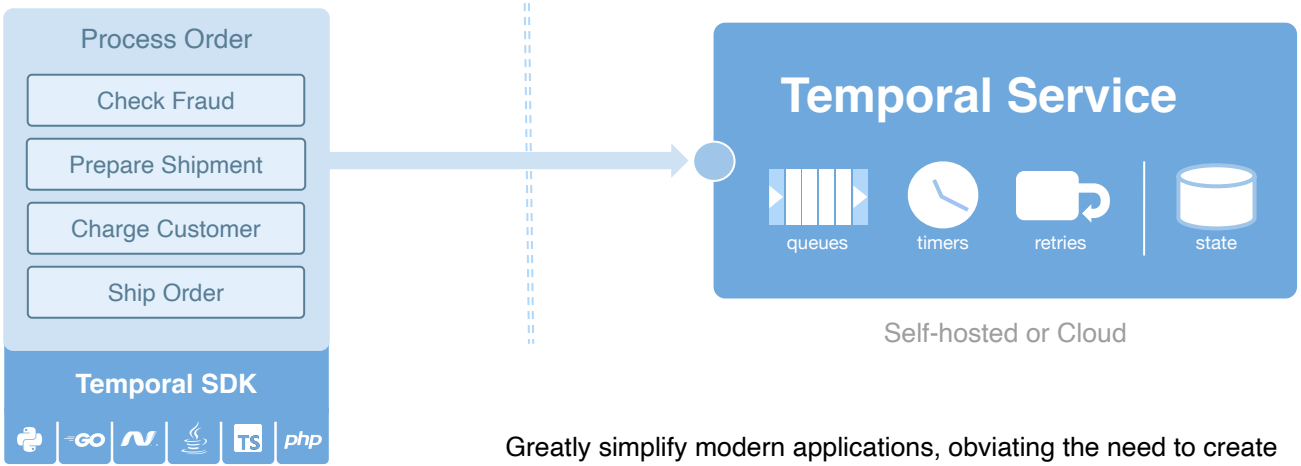
Developers spend an inordinate amount of time coding and testing for complexity and inevitable failure

- Developer Productivity ⇩
- Feature velocity ⇩
- Systems reliability ⇩
- End-to-end insight ⇩



ENTER Durable Execution

Define a workflow and then
maintain state and coordinate execution



Greatly simplify modern applications, obviating the need to create queues, manage timers, publish and consume events, implement retries and rollbacks, checkpoint state, and more.

What is Durable Execution?

```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)
```

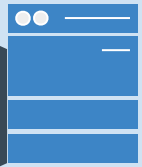
Durable Execution is a new abstraction.

It is a fault-oblivious development model that preserves full application state in the case of any host or software failure.

It ensures your apps and end-to-end services are correct and reliable.

What is Durable Execution?

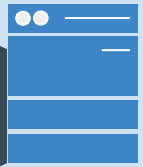
```python
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)

// Represented above is a simple order processing program
```



PROCESS: Active
SERVER: Running

What is Durable Execution?

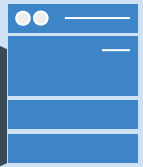
```python
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
```



PROCESS: Active
SERVER: Running

What is Durable Execution?

```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
// step two: prepare the shipment - executes without issue
```



PROCESS: Active
SERVER: Running

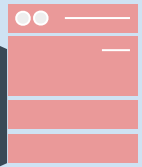
What is Durable Execution?

```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
// step two: prepare the shipment - executes without issue
// step three: confirm charge
//
// Charge request sent, function blocks waiting for request
```

PROCESS: Active
SERVER: Live

What is Durable Execution?

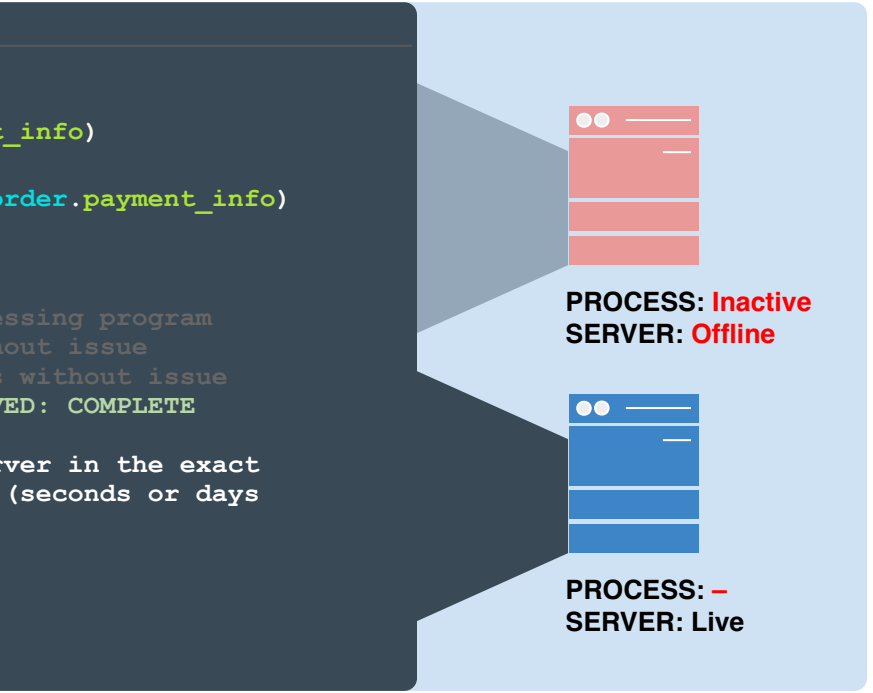
```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
// step two: prepare the shipment - executes without issue
// step three: confirm charge - PROCESS CRASH!
//
// Server CRASHES or loses power
```



PROCESS: Inactive
SERVER: Offline

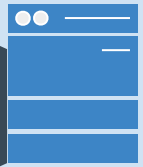
What is Durable Execution?

```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
// step two: prepare the shipment - executes without issue
// step three: confirm charge - ISSUE RESOLVED: COMPLETE
//
// Your program is rehydrated on another server in the exact
// state at time of failure. Response comes (seconds or days
// later) and the function returns
```



What is Durable Execution?

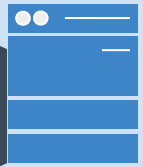
```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
// step two: prepare the shipment - executes without issue
// step three: confirm charge - ISSUE RESOLVED: COMPLETE
//
// With DURABLE EXECUTION...
// - you do not need to code for failure scenarios
// - you can run a function or wait for an API for a year
// - you don't need a db, because variables are durable
```



PROCESS: –
SERVER: Live

What is Durable Execution?

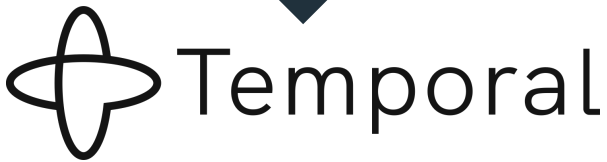
```
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)


// Represented above is a simple order processing program
// step one: check for fraud - executes without issue
// step two: prepare the shipment - executes without issue
// step four: ship the products!
//
// your program completes reliably
```



PROCESS: Active
SERVER: Live

Durable Execution

..is **an abstraction** that opens up an entirely new, and fundamentally better **development model**



TEMPORAL CAPABILITiES

A new set of primitives for durable execution and a more enjoyable **development experience**

Autosave for application state

Temporal captures the complete state of your functions (variables, threads, blocking calls) so you do not need complex state machine code

NATIVE RETRIES

Temporal allows you to define retry policies for Activities so you do not have to code for these

TIMEOUTS/TIMERS

Temporal allows you to manage timers outside your functions and allows you to signal and "wake up" them up without having to code for this

SCHEDULES

Temporal delivers an ability to schedule a workflow (much like a cron job) and then pause, start and stop them

HUMAN IN THE LOOP

Using signals, you can interact with or wake up a Workflow from an external source, even a human action

IDIOMATIC LANGUAGE SUPPORT

Temporal allows you to simply code for durable execution, using of the complete SDKs

TEmPORAL CAPABILITiES

All the key features required for even your most complex **enterprise applications**

EXECUTION VISIBILITY

Temporal captures complete state of your workflow and allows you to gain complete insight into ANY execution of it

LOCAL DEVELOPMENT MODEL

Temporal allows developers to code locally in a familiar environment and then deploy globally

FITS YOUR CURRENT DELIVERY SYSTEM

Because it is just code, with temporal, there is no need to modify your software delivery mechanisms, it fits within your current set of devops tools.

SECURITY

Data in motion and at rest are completely encrypted and with Data Converter your data is encrypted by you end to end. Further, it includes a complete set of auth mechanisms

Nexus

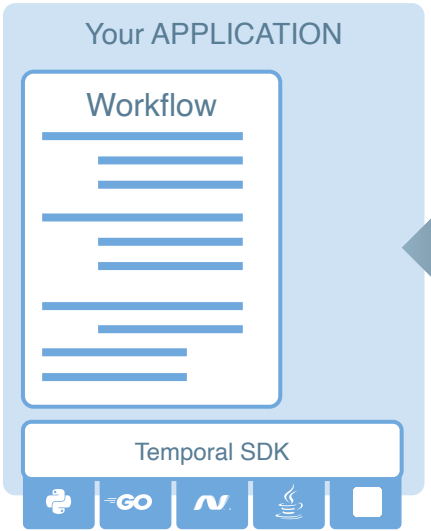
Connects durable executions across team, Namespace, region, and cloud boundaries. It promotes a more modular architecture for sharing a subset of your team's capabilities through well-defined service API contracts for other teams to use

OPEN SOURCE

Temporal is an open source project that you could self-host or use Temporal Cloud

MIT license. Developed in the open.

HOW DOES IT WORK?

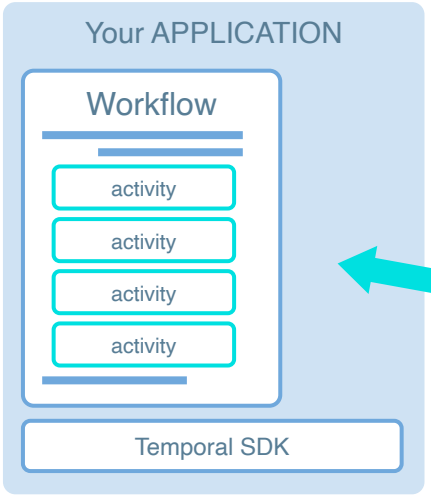


1 Write a Temporal Workflow in your language using a Temporal SDK

This is the business logic you want to execute

```python
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)
```

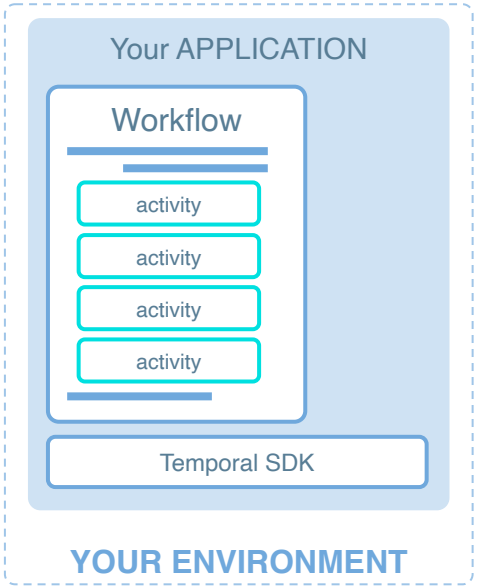
HOW DOES IT WORK?




```python
def process_order(order):
    check_fraud(order.order_id, order.payment_info)
    prepare_shipment(order)
    charge_confirm = charge(order.order_id, order.payment_info)
    shipment_confirmation = ship(order)
```

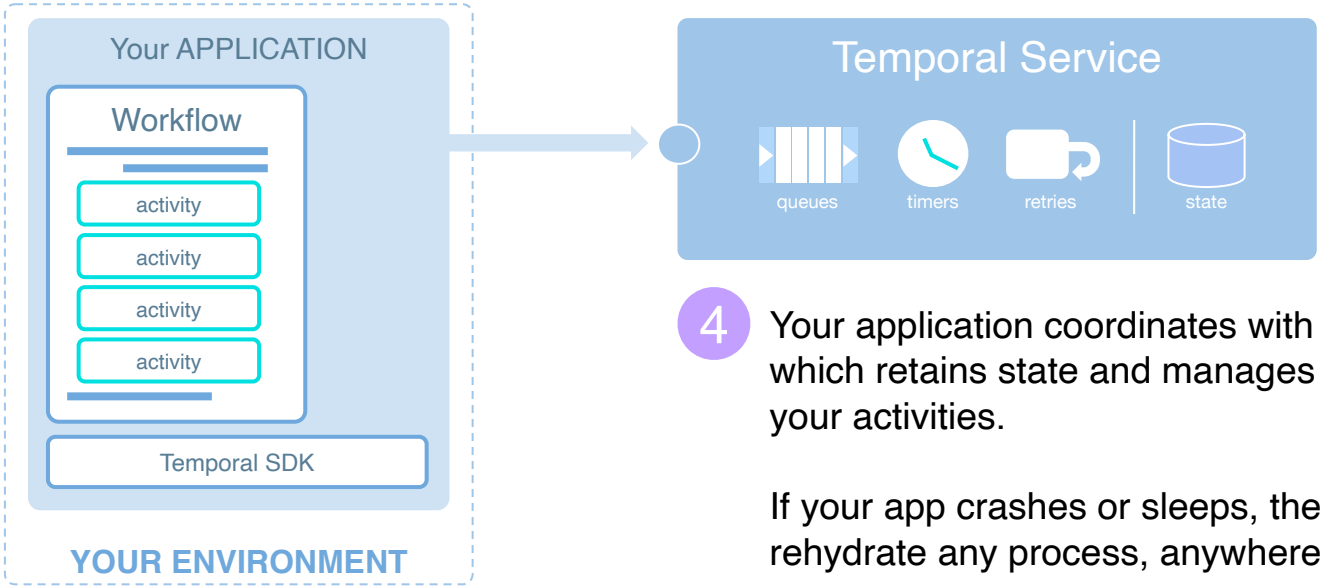
2. Write activities to hold the failure-prone parts of your code that can be **automatically retried** upon some failure

HOW DOES IT WORK?



3 Deploy your application on your infrastructure in your environment using all your standard deployment methods.

HOW DOES IT WORK?

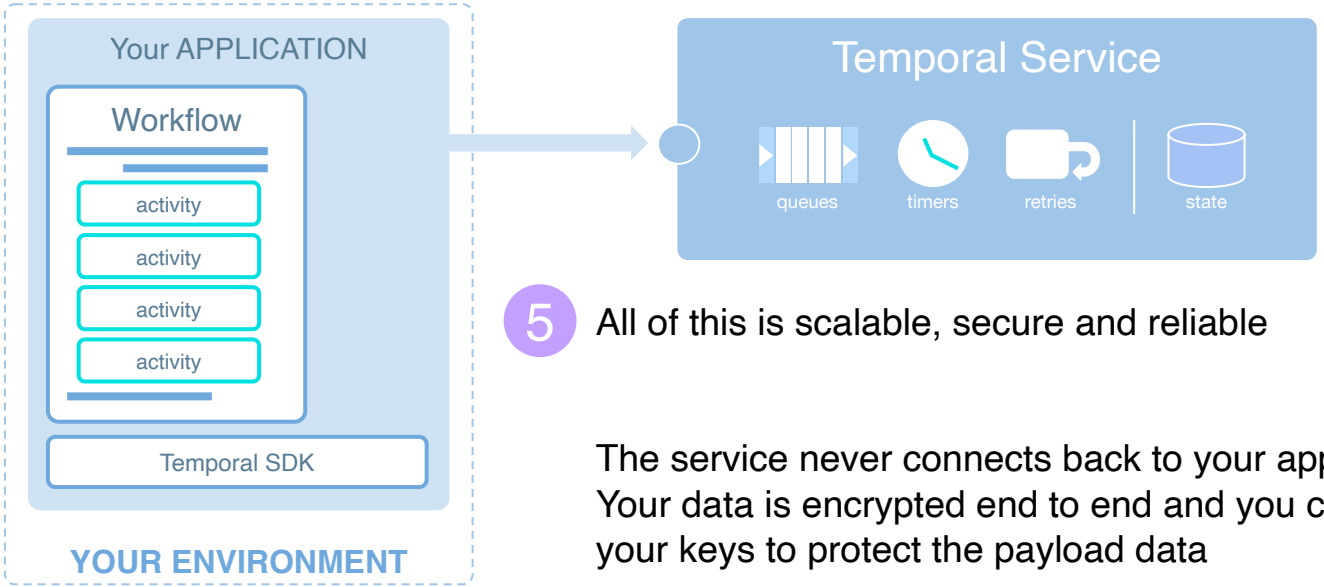


4 Your application coordinates with a Temporal Service which retains state and manages the completion of your activities.

If your app crashes or sleeps, the service can rehydrate any process, anywhere.

You can self-host or use **Temporal Cloud**

HOW DOES IT WORK?



5 All of this is scalable, secure and reliable

The service never connects back to your app.
Your data is encrypted end to end and you can use your keys to protect the payload data

Where is Temporal used?

A **Temporal Workflow** defines any process or series of events that you want to guarantee execute correctly and reliably.

Process Workflows

- Payment Processing
- Money Movement
- Order Management
- Bookings
- Payment Gateway
- Shopping Cart
- Logistics Management
- Customer Service
- Customer Notifications
- Marketing Campaigns
- Supply Chain

Lifecycle Workflows

- Inventory Management
- Menu/listing Versioning
- User Management
- Customer Lifecycle
- Subscription Lifecycle
- Inventory

Operational Workflows

- Infrastructure Services
- CI/CD Workflows
- Saas Provisioning
- Software Provisioning
- Server Provisioning
- Software Upgrades
- Compute Optimization
- Data Pipeline
- ETL

Temporal delivers Durable Execution

Velocity

Temporal allows developers to deliver more business logic, faster. It eliminates complex plumbing code, so that developers can spend more time building features.

Reliability

When failures happen, Temporal recovers processes where they left off or rolls them back correctly. The end result is fewer incidents and less downtime.

Insight

Temporal tracks application state for every execution, giving you insight into issues so you can more effectively debug code and understand application performance.

"The business was worried about getting the new capability done for Q4, but we did it in 1 day with **Temporal**."

"I estimate we saved two engineers by adopting **Temporal** for our team of seven, which is significant."

"There's no such thing as 100% reliability. But we're pretty close in terms of the two systems we have in flight with **Temporal**."

"**Temporal** provides deep understanding of what's happening, at given time, for any message, and for every customer."

Temporal CLOUD

temporal cloud

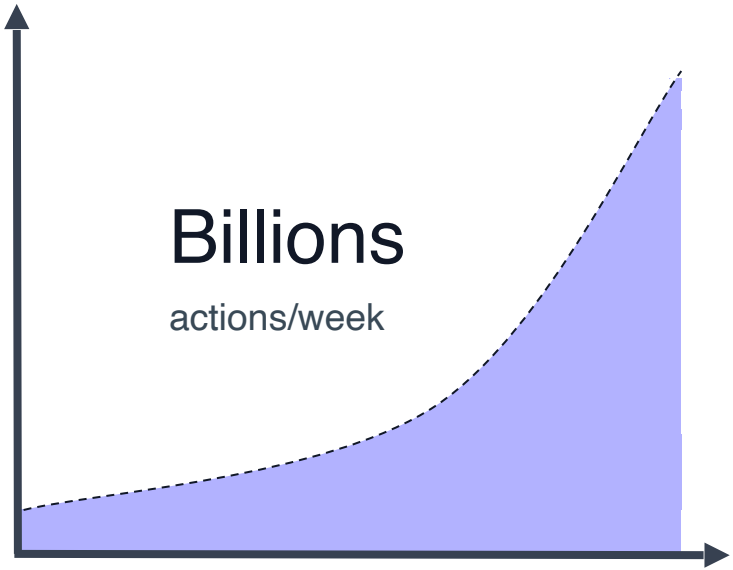
The high-performance, highly available, cost-effective way to run Temporal, trusted by the world's most innovative companies.

1.5K
customers

9.1K
namespaces

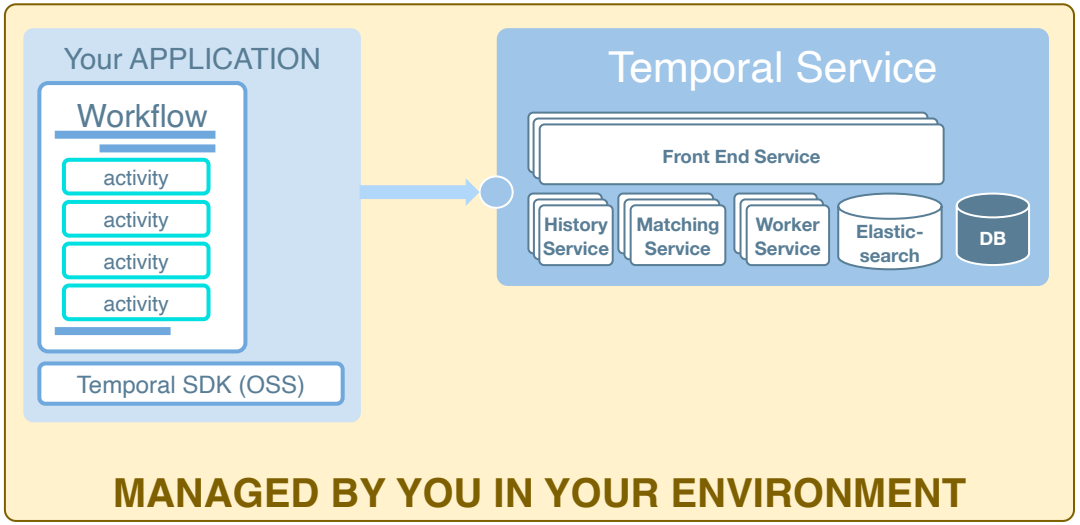
14
cloud regions

NETFLIX · Alaska · OpenAI · citrix · Brex · COMCAST · MACQUARIE · ANZ · qualtrics · flexport. · Yum! · NVIDIA · kotak · CLOUDFLARE · ROBLOX · HashiCorp · TURO · DOORDASH · Outreach · Remitly · SOUNDCLOUD · Airwallex · MAERSK · TRANSAMERICA



SELF-HOSTED Temporal: HOW DOES IT WORK?

Execute your Workflow code **and** manage the Temporal Service in your environment

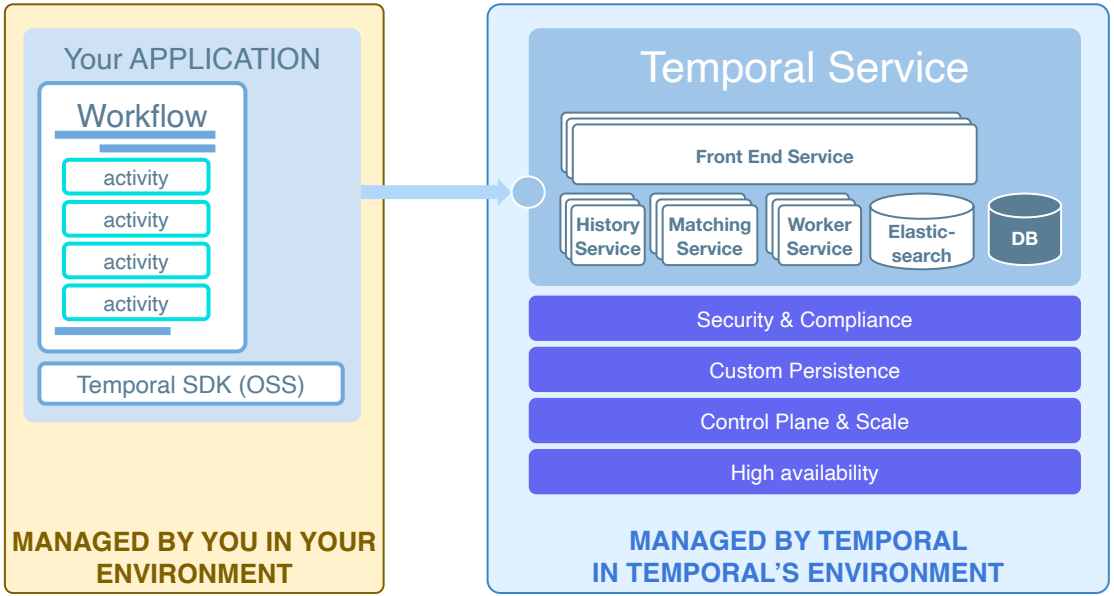


To take advantage of Temporal's durability, you must deliver a **reliable**, **highly available Temporal Service:**

- High availability SLA
- Manage, scale, and maintain availability of 5 independent services & database
- Support for critical issues to your developers
- Performance optimization

Temporal CLOUD: HOW DOES IT WORK?

Continue executing your Workflow code in your environment; our team manages the Service



Eliminate the need to manage a complex, highly available Service

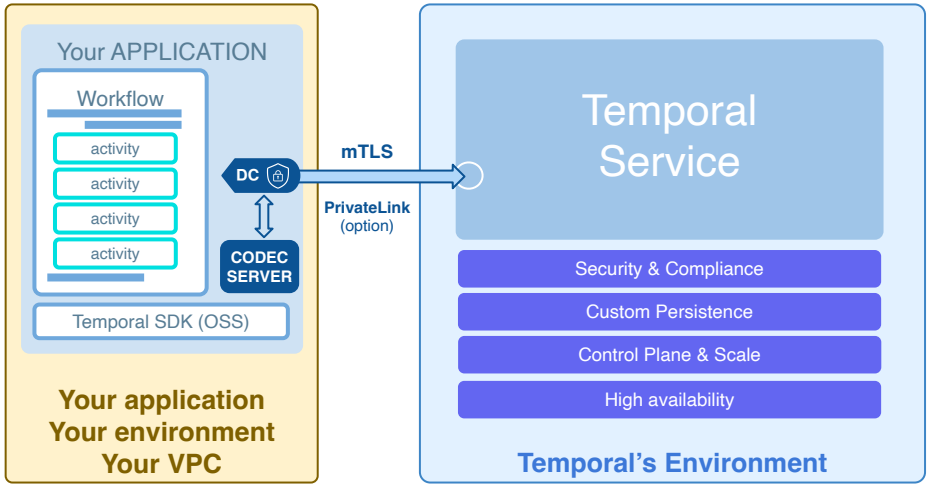
Temporal Cloud Service is optimized with:

- Security (RBAC, SSO with SAML, etc.)
- Compliance
- Control Plane
- Custom Persistence Layer
- Highly available out of the box

temporal cloud:

Maintain control of your data and your security

SECURITY



We never see your code
We only receive Workflow & Activity data

One-way, outbound connections from your environment to the Temporal Service

End-to-end encryption

- In motion & at rest
- Data Converter (DC) for high-value data
- Encryption owned and operated by customers

✔ AWS PrivateLink ✔ RBAC ✔ SSO ✔ Audit Logging ✔ Annual 3rd-party penetration test

SOC 2 TYPE II CERTIFIED

HIPAA COMPLIANT

temporal cloud:

Designed for high scale and low latency

PERFORMANCE

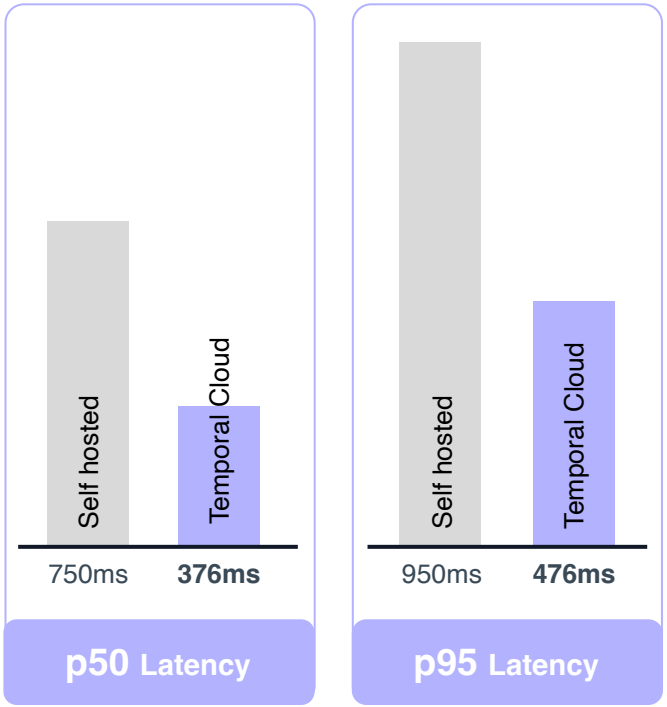
Take advantage of an architecture that maximizes performance and scale

Custom persistence layer - optimizes interactions between the Service and storage layer to scale transactional throughput

→ ***50% reduction*** *in latency compared to self-hosted*

Control Plane - delivers elastic scale; defaults to 400 actions per second and scales automatically based on load

→ *Tested to* ***300K actions per second***



End-to-end Workflow Latency

Public benchmark available on temporal.io

temporal cloud: HIGH AVAILABILITY

A fault-tolerant deployment with additional options for disaster recovery

SLA

99.9% contractual SLA by default

- Deployed across a minimum of 3 AZs
- Zero RTO/RPO for AZ failures
- Historical operational SLA of 4 9's

Credits-back SLAs: in the event of an SLA violation, we issue credits back based on the outage.

99.99% contractual SLA with Multi-Region Namespaces

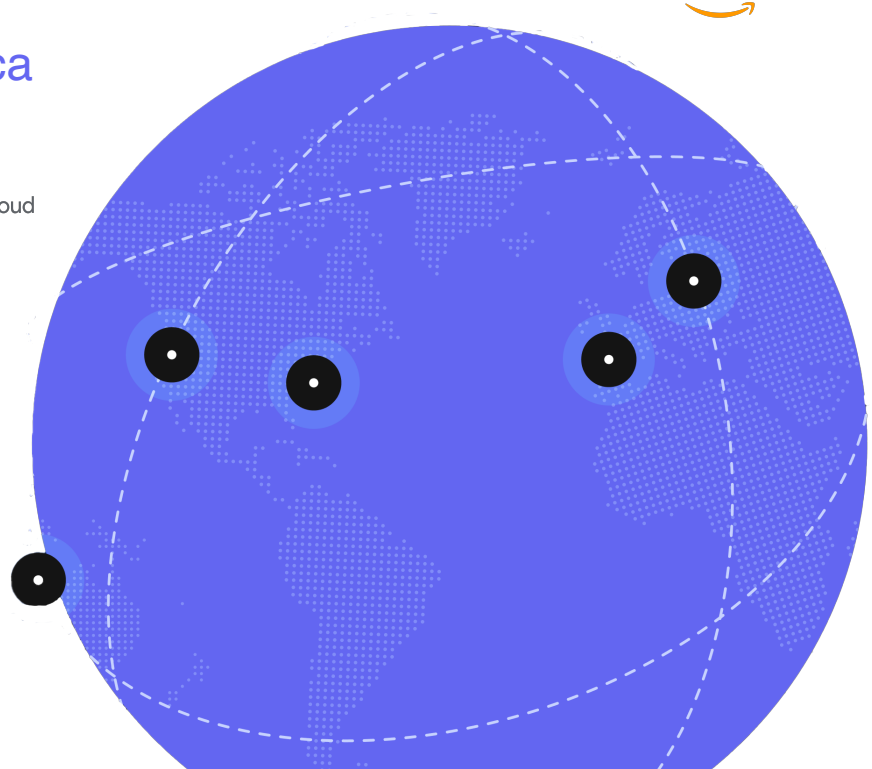
- Data is replicated to a secondary region
- Automatic failover during a region outage
- Up to 20 mins RTO / near-0 RPO for region failures

temporal cloud: GLOBAL AVAILABILITY

Push-button deployment to multiple regions around the world

Instantly spin up Temporal, wherever you are, and in multiple regions

- Dozens of regions globally in AWS and Google Cloud
- Multi-region deployment for extra fault-tolerance



temporal cloud: PAY FOR WHAT YOU VALUE

Pay only for what you use & Temporal Cloud scales on-demand to avoid wasting money on idle capacity

Available on aws marketplace

Three elements of Temporal Cloud pricing

ACTIONS PER MONTH

An Action is an Event that occurs as part of an execution of your Workflow.

Includes: Workflow, Activities, Signals, Timers, Heartbeats, Queries

STORAGE per GBh

Active Storage: Running Workflows

Retained Storage: Closed Workflows

SUPPORT PER MONTH

Basic & **Premium** options
Services included in the price of Support

- Onboarding**:** Setup, planning, SDK overview
- Development: Code and design review for optimized performance
- Production: Observability, metrics, and app design

Temporal Cloud: Support & Services

Temporal Cloud includes the necessary services and support to help you from onboarding through development, to running efficient production workloads

Temporal Cloud: EXPERT SERVICES

Accelerating your success with pragmatic, **expert services.**

Advisory

White-glove guidance including architectural recommendations, samples, and patterns for your org. A base of 40 expert hours that can be used to accelerate & optimize Temporal deployments.

$10,000 (varies based on number of hours and partner)

JUMPSTART

A **2-day in-person program** to accelerate implementation and adoption of Temporal on a specific use case, followed by a bucket of 40 hours to continue momentum.

$17,500 (varies based on scope and partner)

CO-DEVELOPMENT

Side-by-side engagement with an expert Temporal partner that enables you to deploy Temporal faster and more effectively, while accelerating enablement of your teams.

Tailored SOW for your specific needs

Services can be purchased directly from Temporal.
Have a preferred partner you already work with? Let us know.

BITOVI Deloitte. epam Globant KPMG MODERNIZE NETCONOMY platformadoru Prodapt

slalom SpiralScout takima techfabric VALCANN APARTMENT304 BUILDING ON THE CLOUD CRISPY BACON

temporal cloud: TWO PAYMENT MODELS

Pay-as-you-go MODEL

- Fixed-rate pricing, with no upfront costs
- Usage & Support fees invoiced monthly
- Cancel at anytime
- Pay-as-you-Go on AWS Marketplace

Actions	Cost
1M Actions	$25

For both payment models:

Support: 10% of monthly usage fees
(Minimum $200 Basic / $2,000 Premium)

Storage: Approx 5% of Actions cost
Further discounted for Credits Model
See Appendix for full details

CREDITS MODEL

- Discounted tiered pricing; minimum initial credits purchase of $2,500
- Usage & Support fees deducted from credits balance monthly
- Credits do not expire
- Credits are fully refundable
- Available via Private Offer on AWS Marketplace

Monthly Actions	Cost per 1M	Discount
Up to 300M Actions	$23.25	7%
After 300M up to 1.5B	$18.80	25%
After 1.5B up to 7.5B	$14.10	44%
After 7.5B up to 30B	$10.50	58%
After 30B up to 150B	$7.90	68%
After 150B Actions	$5.90	76%

temporal cloud: SUPPORT OPTIONS

	Basic	Premium
Business Hours	P0: 24x7 P1: 0500-1700 Mon-Fri PT	
Response times	**P0: 1 hour** P1: 4 business hours P2: 1 business days P3: 2 business days	P0: **30 minutes** P1: 1 business hour P2: 4 business hours P3: 1 business day
Channels	Community Slack, Email, Forum, Docs, Knowledge Base	**Private Slack Channel**, Email, Forum, Docs, Knowledge Base
Success	Expert services for successful development, deployment, and optimization, and guidance	
Price	Greater of: $200 or 10% of monthly usage	Greater of: $2,000 or 10% monthly usage

- Support from the inventors and contributors of the OSS project
- Global team of experts supports largest Temporal deployments in the world
- Fast response times and 24x7 coverage for critical issues

temporal cloud: INCIDENTS & RESPONSES

P0: Critical	**Production Impacted** Temporal Cloud service is unavailable or degraded with significant impact.
P1: High	**Production Issue** A service issue related to production workloads or a significant project block.
P3: Normal	**General Issues** General service issues without production impact or a workaround exists.
P4: Low	**General guidance** Questions or issues with \service that do not impact system availability or functionality.

Incident Management

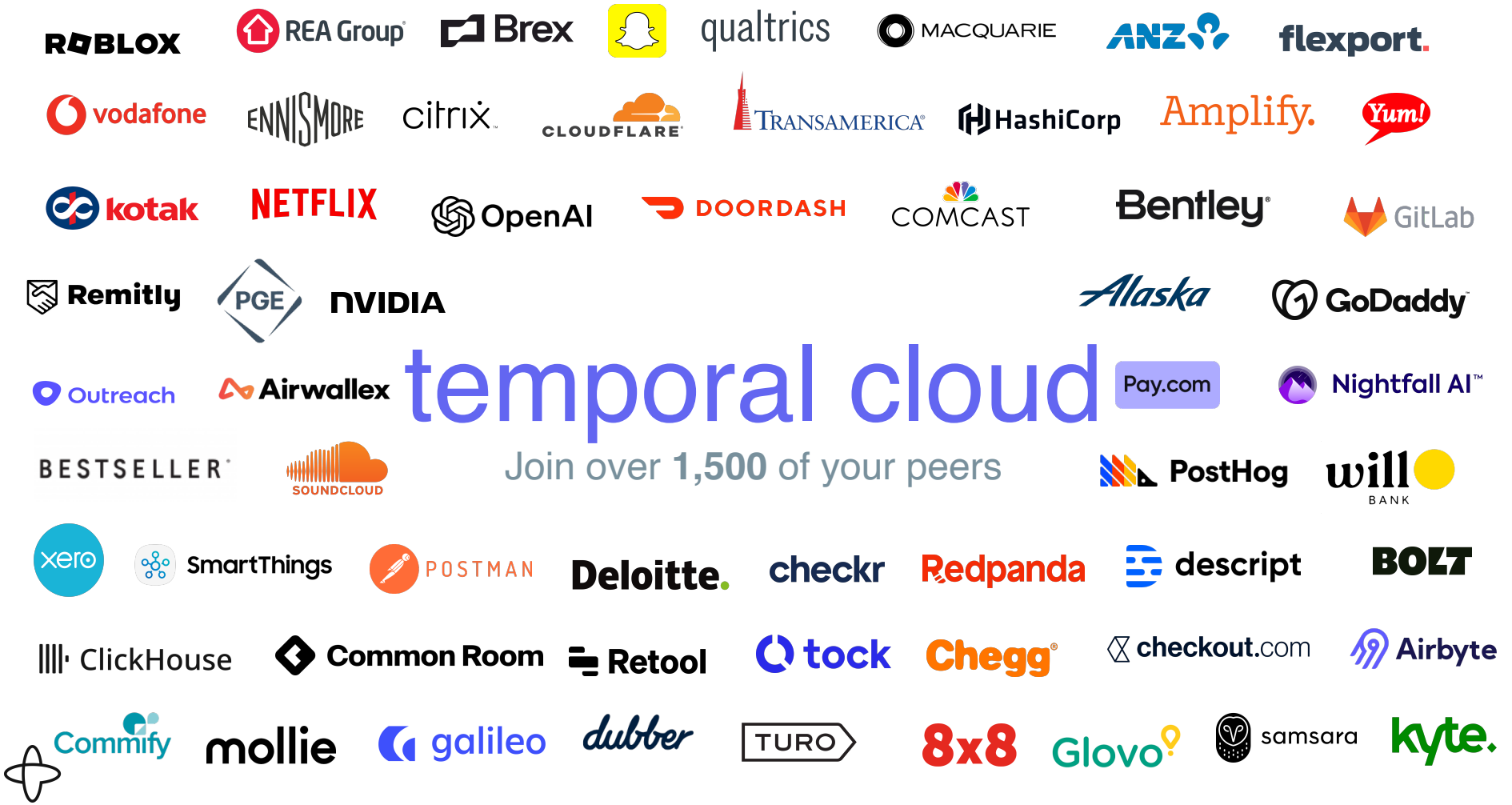
- Engineering team, Manager and support teams on call 24/7, 7x7.
- P0 - Critical incidents
  - Triggered via internal monitoring or inbound P0 report
  - Incident team assigned internal slack channel for remediation
  - External communications established via preferred channel for impacted customers
- Lifecycle of all incidents captured
  (root cause, remediation and corrective actions)
- Major incidents impacting uptime reported to public dashboard at status.temporal.io.

temporal cloud: COST-EFFECTIVENESS

Temporal Cloud wraps a production-grade, enterprise service into a consumption-based cost

Pay only for what you use, and get:

- ❏ Hardened security & compliance
- ❏ Lower latency & higher scale with custom architecture
- ❏ Management, scaling, upgrading, high availability (10+ engineers)
- ❏ Infrastructure provisioned to peak capacity
- ❏ Instant global deployment
- ❏ Availability SLA
  - ❏ 3 9’s with credits back for single-region
  - ❏ 4 9’s with credits back for multi-region
- ❏ 24/7 on-call support for 1 - 1,000+ developers
- ❏ Expert training on how to use & optimize Temporal


ROBLOX
REA Group
Brex
qualtrics
MACQUARIE
ANZ
flexport.
vodafone
ENNISMORE
citrix
CLOUDFLARE
TRANSAMERICA
HashiCorp
Amplify.
Yum!
kotak
NETFLIX
OpenAI
DOORDASH
COMCAST
Bentley
GitLab
Remitly
PGE
NVIDIA
Alaska
GoDaddy
Outreach
Airwallex
temporal cloud
Pay.com
Nightfall AI™
BESTSELLER
SOUNDCLOUD
Join over 1,500 of your peers
PostHog
will BANK
xero
SmartThings
POSTMAN
Deloitte.
checkr
Redpanda
descript
BOLT
ClickHouse
Common Room
Retool
tock
Chegg
checkout.com
Airbyte
Commify
mollie
galileo
dubber
TURO
8x8
Glovo
samsara
kyte.

Appendix

Temporal CLOUD

temporal cloud: TWO PAYMENT MODELS

Pay-as-you-go MODEL

- Fixed-rate pricing, with no upfront costs
- Usage & Support fees invoiced monthly
- Cancel at anytime

Element	Cost per Consumption
Actions	$25 per million Actions
Running Storage	$0.042 per GBh
Retained Storage	$0.00042 per GBh

For both payment models:

Support: 10% of monthly usage fees (Minimum $200 Basic / $2,000 Premium)

CREDITS MODEL

- Discounted tiered pricing; minimum initial credits purchase of $2,500
- Usage & Support fees deducted from credits balance monthly
- Credits do not expire
- Credits are fully refundable

Monthly Actions	Cost per 1M	Discount
Up to 300M Actions	$23.25	7%
After 300M up to 1.5B	$18.80	25%
After 1.5B up to 7.5B	$14.10	44%
After 7.5B up to 30B	$10.50	58%
After 30B up to 150B	$7.90	68%
After 150B Actions	$5.90	76%

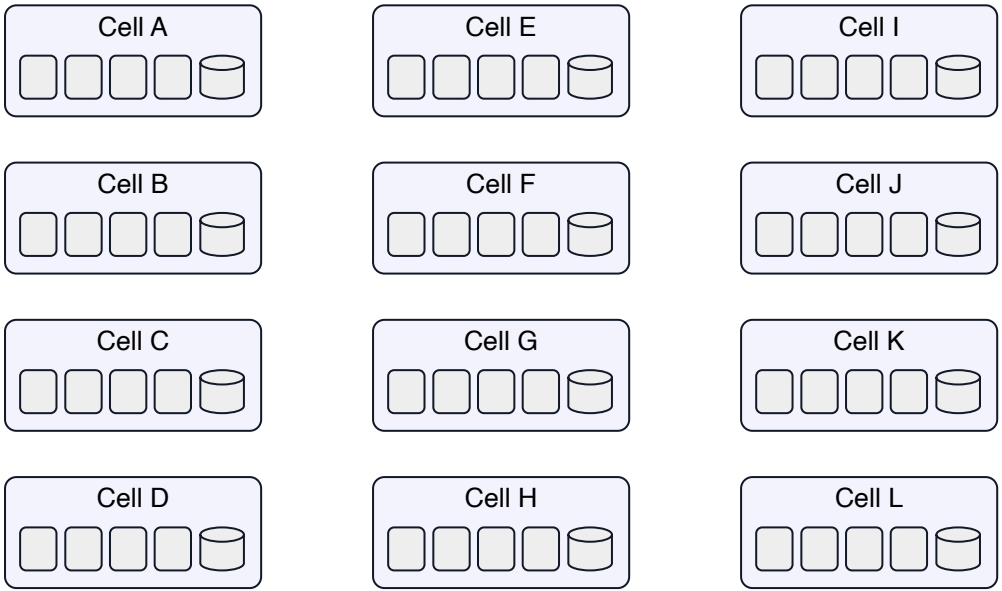
Monthly Running Storage	Cost per GBh
Up to 7,000 GBh	$0.039
After 7,000 up to 30,000	$0.031
After 30,000 up to 90,000	$0.023
After 90,000 up to 400,000	$0.018
After 400,000 up to 1,500,000	$0.013
After 1,500,000 GBh	$0.010

Architecture & Operations

Temporal CLOUD

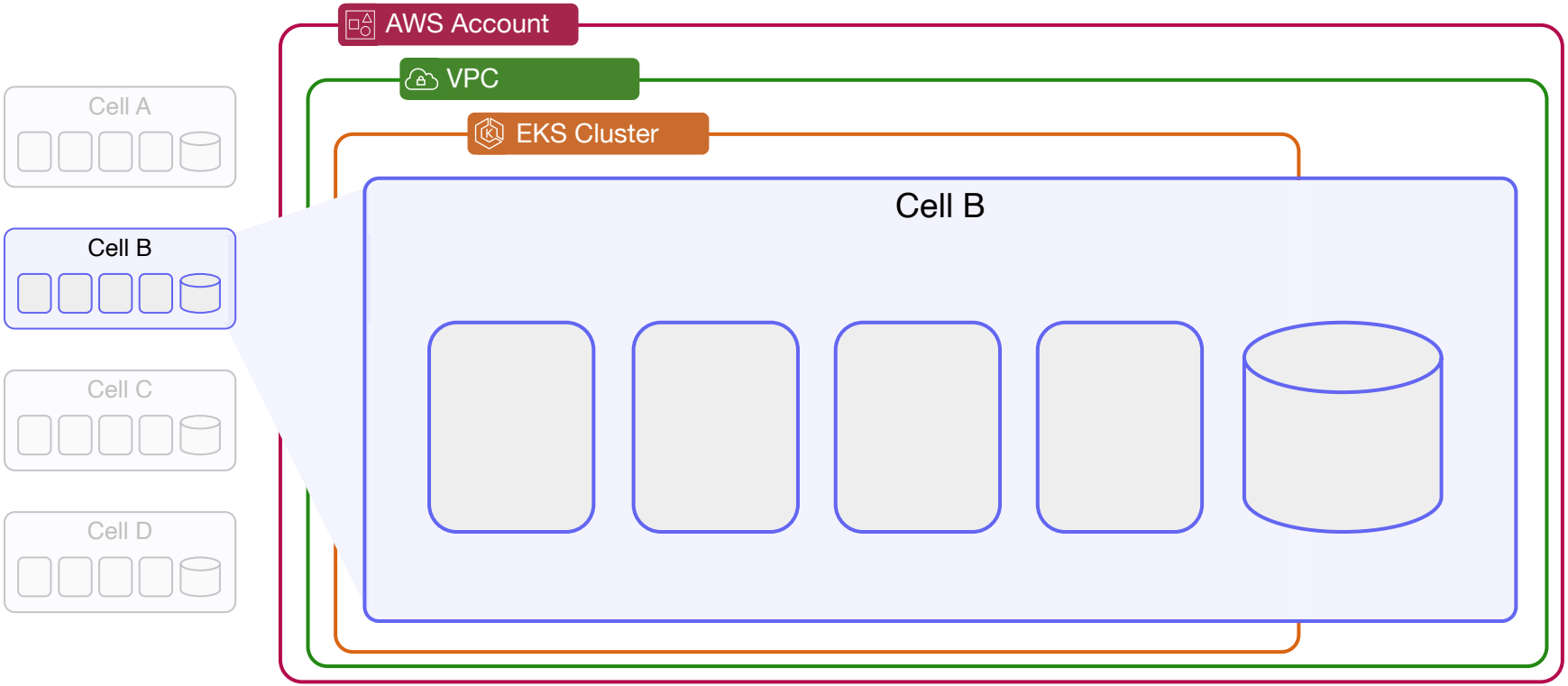
Temporal Cloud: A "Cell" Architecture

A series of containers that isolate failure and improve performance

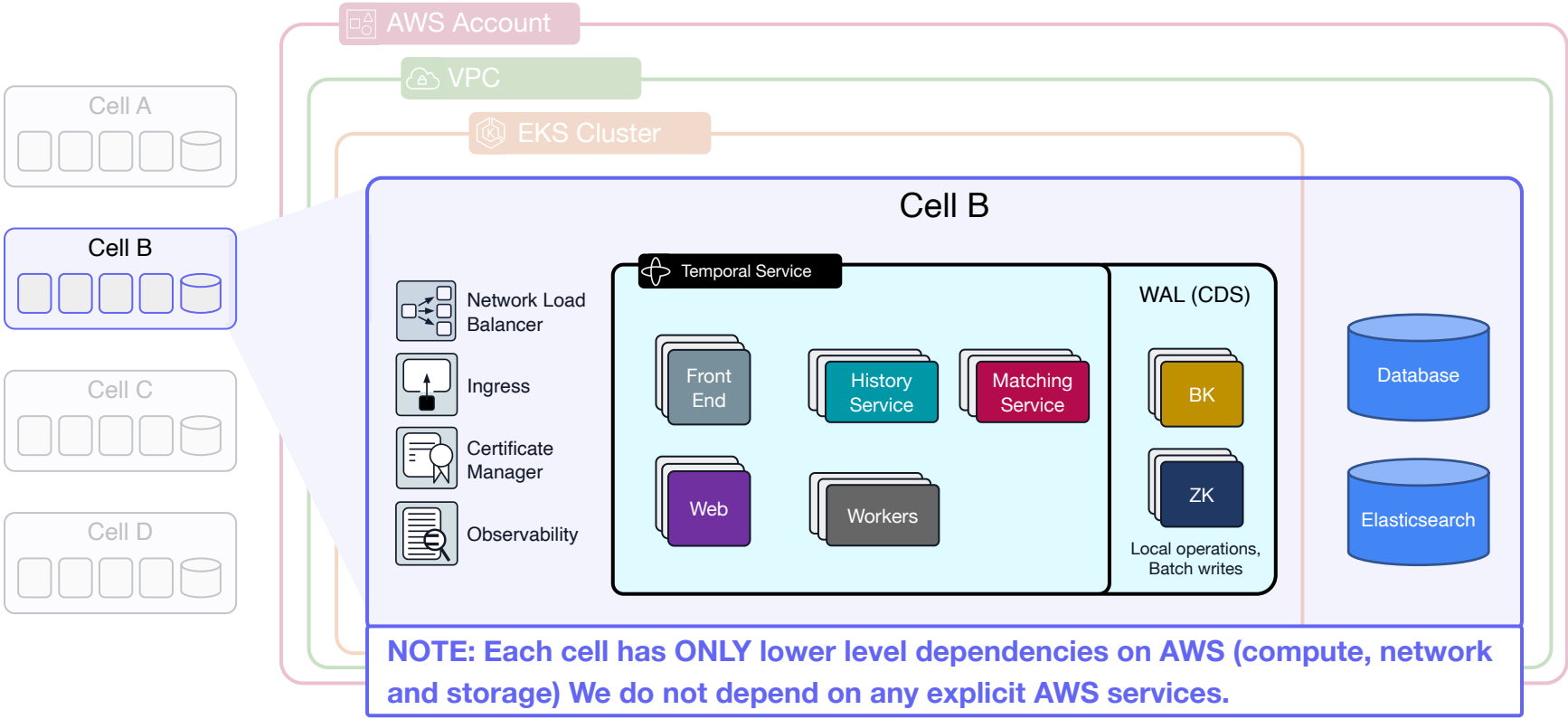


- Cells are different sizes
- Cells can contain multiple namespaces
- Multiple namespaces are spread across multiple cells
- We distribute load across cells to improve service

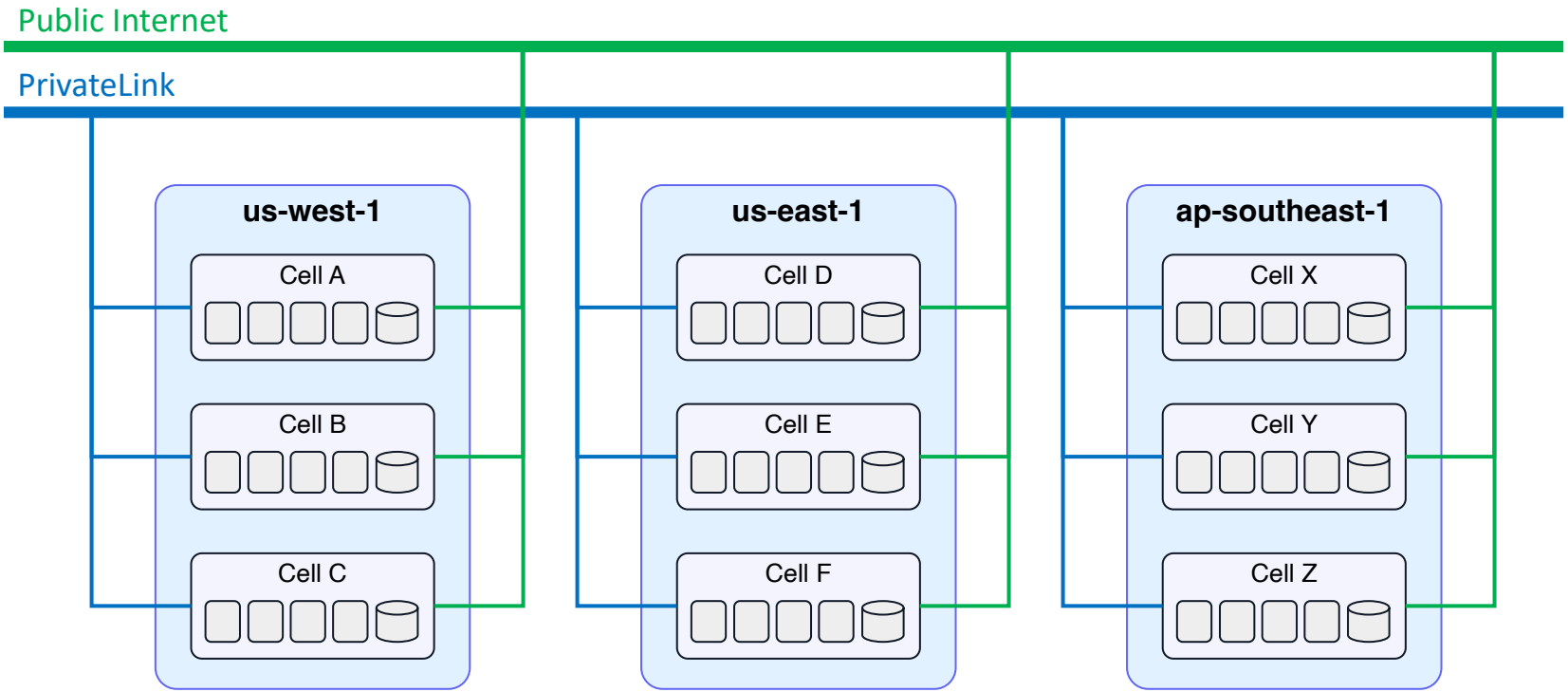
Each cell runs in AWS behind a VPC



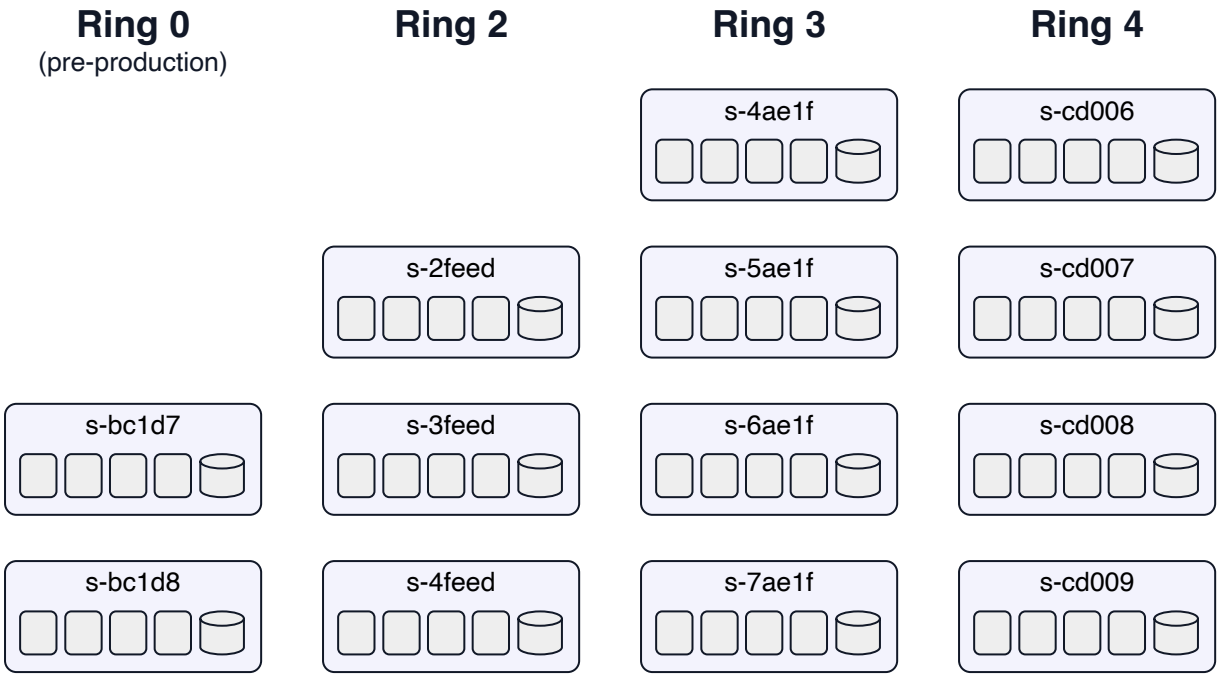
Within each cell, a complex service is deployed



Regional isolation of cells



Deployment Rings optimize rollouts



- Temporal workflow applies changes in batches on each ring
- Reduces probability of failure
- Default time is ten days

Temporal Services: default prioritization

Upon degradation in performance, Temporal Service naturally prioritizes critical calls so that retriable calls are throttled and business impact is minimized

SLOs defined and priorities set across four levels of calls:

1. **External events** - start workflow, signal, etc. → highest priority
2. **Workflow progress** - task completed, activity execution
3. **Visibility API** - operational tasks that provide insight
4. **Cloud operations**- create namespace, etc. - zero impact active workloads

What happens upon incident?

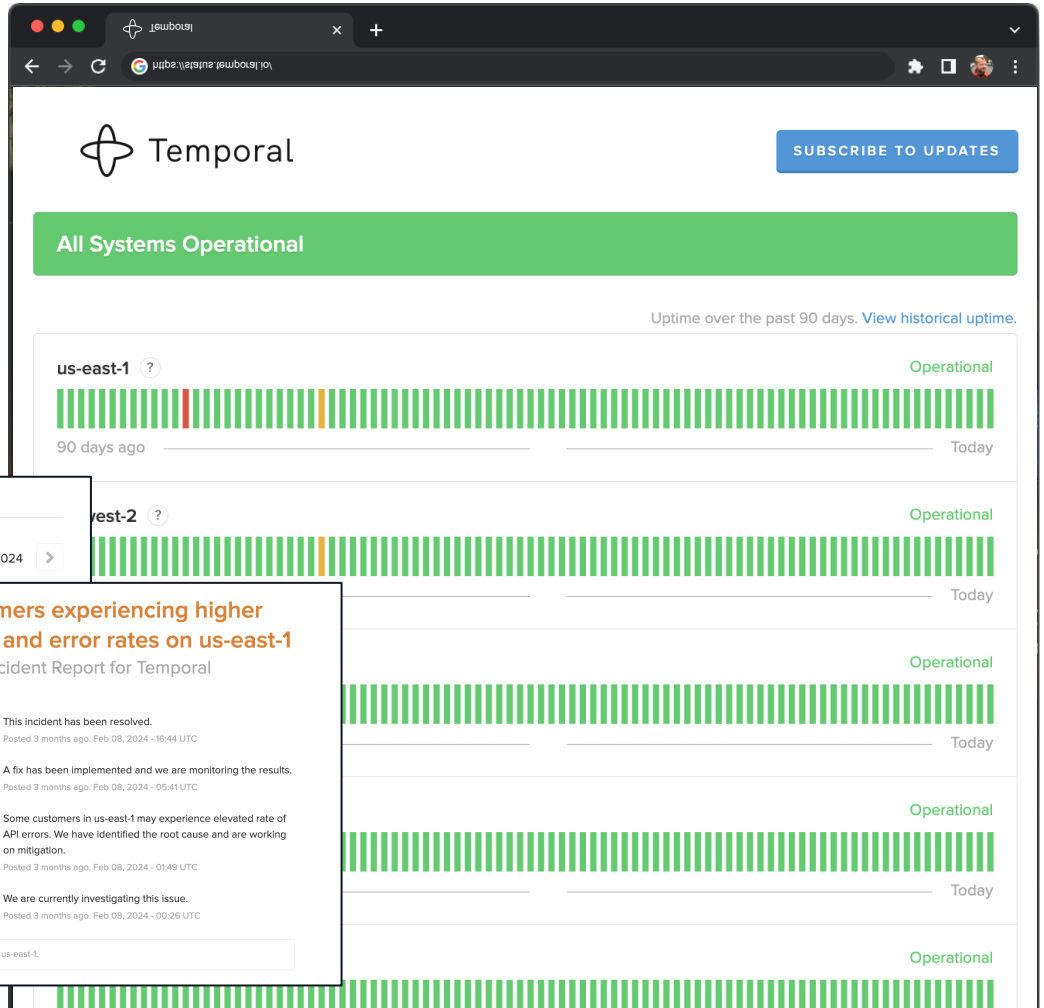
Engineering team, manager and support organization on call 24/7

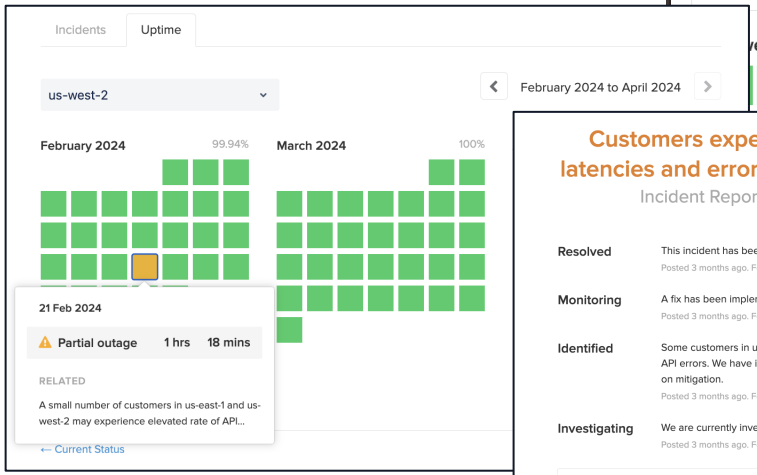
1. Workflow prioritization insulates damage for service degradation
2. Internal team (engineering, manager & support) paged w/in 5 minutes of "event"
    - Event can be P0 incident reported by service owner (customer)
    - Event can be identified via internal monitoring
3. Internal Slack channel opened for triage
4. Customer preferred communication channel opened
5. Incident lifecycle tracked and documented
   (root cause, remediation and corrective actions captured and reported)
6. status.temporal.io continuously updated

status.temporal.io

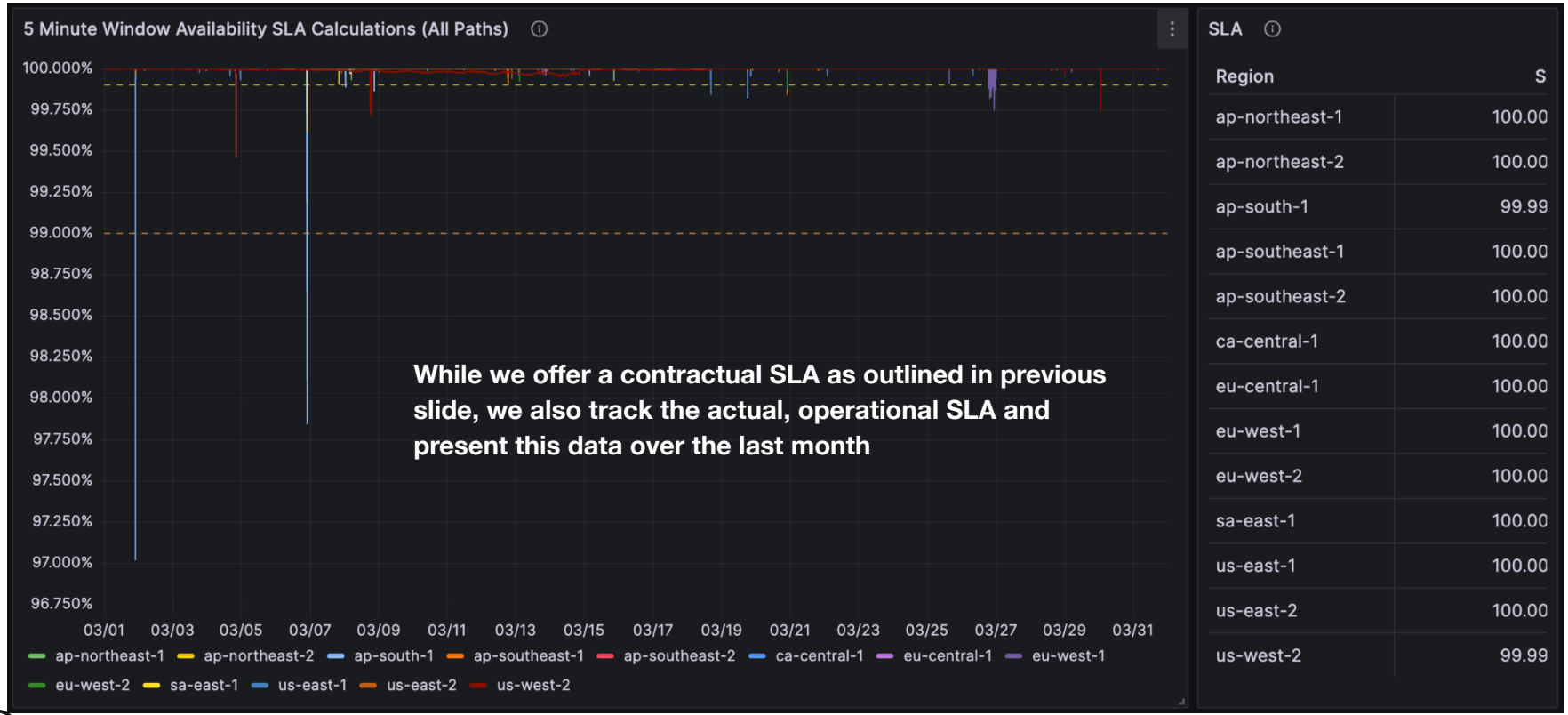
Fully transparent communication mechanism to identify/determine issues with Temporal Service

Monitoring: issue has been fully mitigated (**TTR**); Temporal then completes a series of follow-up and monitoring events until the Incident is marked as Resolved.





Temporal Cloud: Operational SLA



TEMPORAL USE CASES

What applications benefit from Temporal?

Temporal is a general-purpose platform, valuable for any process or series of events that must execute correctly and reliably.

The most common Temporal use cases fall into these categories:

- Business process
- Operations
- AI & ML
- Platform

Temporal Use Cases | Business Processes

Temporal is commonly used to orchestrate reliable, scalable business processes.

Orders & Bookings
Inventory & Logistics
Payment Gateway

Customer Onboarding
Customer Lifecycle
Subscription Lifecycle

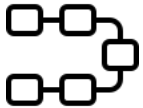
Payment Processing
Risk & Fraud Analysis
Identity Verification

Temporal Use Cases | Ops, Data & Platform

Temporal is commonly used to orchestrate infrastructure management, data pipelines, and more.

Operations & Data
CI/CD
Infra management
Infras provisioning

AI & ML
AI Inferences
Data pipeline & ETL
Model Training



Platform
Control Plane
Workflows (DSL)

Orders & Bookings

Orders & Bookings | Overview

Order management applications ensure successful completion of an order, like an **e-commerce order** or **flight booking**, and often span the end-to-end order lifecycle:

- Promotions & coupons
- Payment validation
- Payment Installments
- Inventory control & validation
- Customer notifications
- Shipment
- Delivery
- Returns & cancellations

Orders & Bookings I Challenges

Challenge	Impact
Poor reliability & validation rates	✗ Lost revenue
Complex, legacy architectures	✗ Stifled innovation & slow dev velocity
Seasonal traffic spikes & promos	✗ Scale issues and bottlenecks

Orders & Bookings I Why Temporal

Temporal Value	Impact
10x+ improved reliability	✓ Saved revenue
Simplify the coding, testing, and debugging of multi-step workflows	✓ 10x feature velocity & innovation
Gracefully scale up Workers to handle more traffic	✓ Be more agile & increase revenue

Yum! boosted developer productivity by 10x

Challenges

New project to centralize web & in-store orders from 53,000 global stores onto a single OMS

- V1 new OMS built with services, cron jobs, & queues
- Reliability issues & code complexity reduced dev velocity

Solution

Complete rebuild of OMS to replace that state machines & EDA with Temporal Workflows

- Lower latency & higher availability order processing
- Can add new features in hours instead of months

"The business was worried about getting [the new feature] done for Q4, but we did it in one day with Temporal."

Matt McDole
VP of Engineering

10x
Reduction in code

10x
More reliable OMS

Maersk deploys changes to prod 9x faster

Challenges
Goal of transforming 20-year old monolith to modern, highly resilient, reactive OMS. Monolith riddled with pain points:

- 60 to 80 days required to deploy changes to prod
- Stalled orders & poor customer satisfaction

Solution
Orchestrate end-to-end OMS using Temporal Workflows

- Only 5 to 10 days required to deploy changes to prod
- Quick, real-time observability for shipments
- Easy human-in-the-loop & long-running order flows

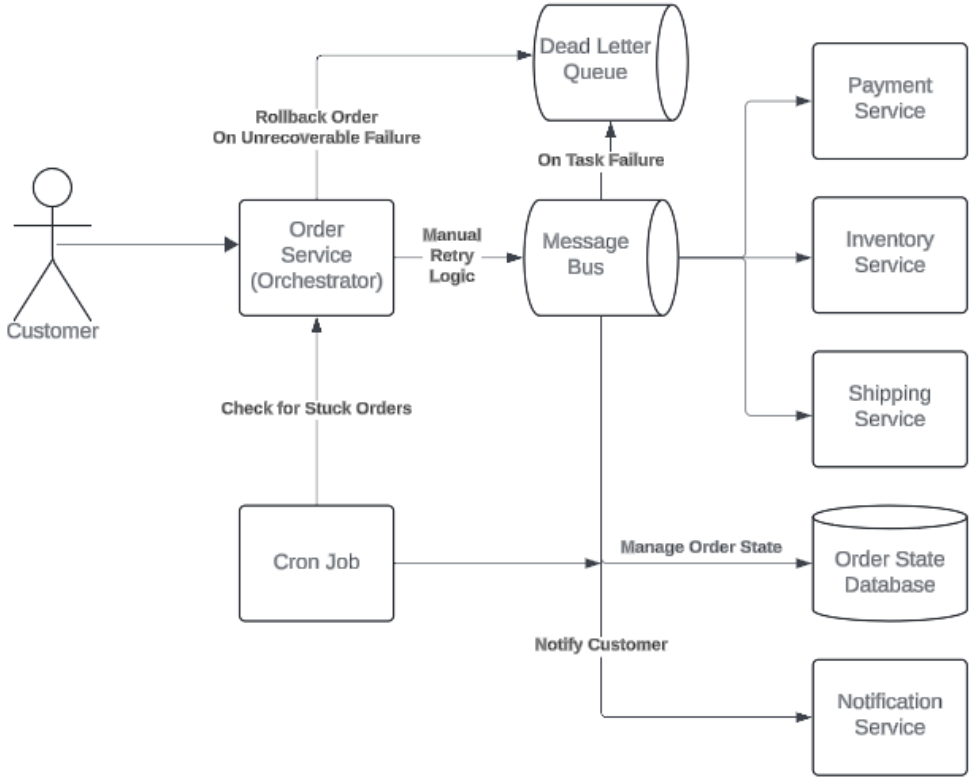
"Is time travel possible? We're giving time back compared to our old selves, we're giving time to our customers, we're giving time to our engineers to do better things with their lives… And it is all possible due to Temporal."

Andrey Dubnik
Principal Engineer

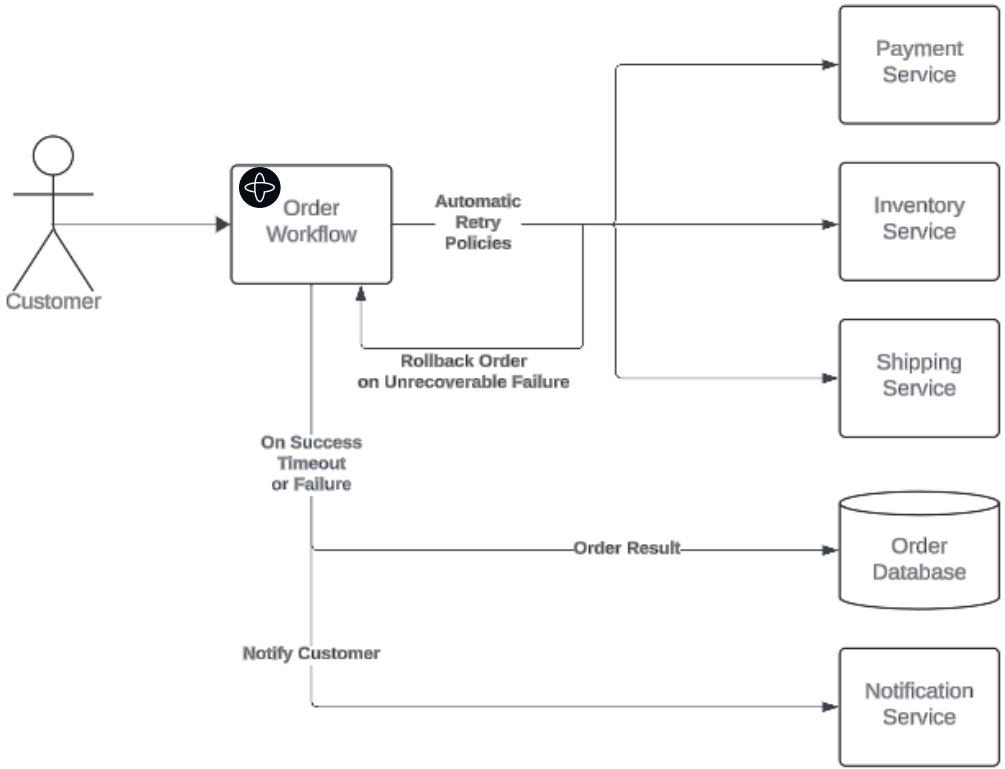
Years
Estimated timeline for transformation *without* Temporal

Months
Actual timeline for transformation *with* Temporal

Orders & Bookings I Architecture before Temporal



Orders & Bookings | Architecture after Temporal



Payment Processing

Payment Processing I Overview

Payment processing applications span a variety of purposes including:

- Money movement
- Bank payments (deposit/withdrawal)
- Recurring payments
- Reconciliation

Payment Processing | Challenges

Challenge	Impact
Brittle legacy apps, state machines, & orchestration frameworks	✗ Stifled innovation & new features
Millions of transactions per day	✗ Extensive designing & tuning required
Stringent regulations	✗ Wasted dev time & stifled innovation
Difficult to meet internal SLAs	✗ Poor reliability & lost revenue

Payment Processing I Why Temporal

Temporal Value	Impact
Simpler path to modernization	✓ Simpler architecture, more innovation, & increased developer velocity
Horizontal scale for concurrent payments, even for monoliths	✓ Less optimization time required
Enhanced visibility into systems	✓ Easier compliance
Improved availability & reliability	✓ Better success rates & saved revenue

"It currently takes a team of 3 experienced developers roughly 20 weeks to build a new feature in our legacy systems and only 2 weeks using Temporal.

That's akin to a team of 8 developers (using Temporal) being as productive as a team of 80."

Engineering Manager
Leading Investment Management Firm

Big 5 Global Bank

Top payment processor

Big 5 global bank increased payment reliability from 99.4% to 99.99+%

Challenges

Project to modernize payment monolith, which was a bloated codebase with complex, manual logic

- Sub-optimal payment completion rate of 99.4%
- Difficult to troubleshoot across many services

Solution

Streamlined system and developer experience using Temporal Workflows and the Java SDK

- Reliability reached 99.99+% with Multi-Region Namespaces
- All boilerplate code handled by Temporal, allowing devs to purely focus on writing business logic

"You can improve the reliability of the system by introducing Temporal and making sure every single payment has succeeded – and if it's failed, we know what's happening."

Executive Director - Software Engineering

Hours

spent combing through logs to find lost payments

Minutes

spent looking at the Event History to find lost payments

Leading payment processor significantly improved reliability

Challenges

- *Poor reliability & observability*: EDA difficult to debug, leading to lost payments and poor customer satisfaction
- *Poor scale*: monoliths, not well documented, stifled growth
- *Poor feature velocity and innovation:* hampered by brittle state machines and orchestration frameworks

Solution

- *Improved reliability*: few failures, better customer satisfaction
- *Horizontal scale:* fewer bottlenecks, less work for Architects
- *Faster innovation*: out-of-the-box state machine that enables easier coding, new features, and innovation

"Reliability is reputation, and reputation is money in the banking industry."

Engineering Manager

10s of millions

of account statements generated through Temporal

Green Got runs applications with 10x less work

Green Got

Private European bank that allows consumers to invest in green companies.

Challenges

Replace an unreliable, event-driven, payment scheduling app

- Struggled to maintain state across multi-step processes that ran as concurrent or parallel cron jobs
- Failure resolution required manual engineering intervention

Solution

Rebuilt the app using Temporal and the Go SDK

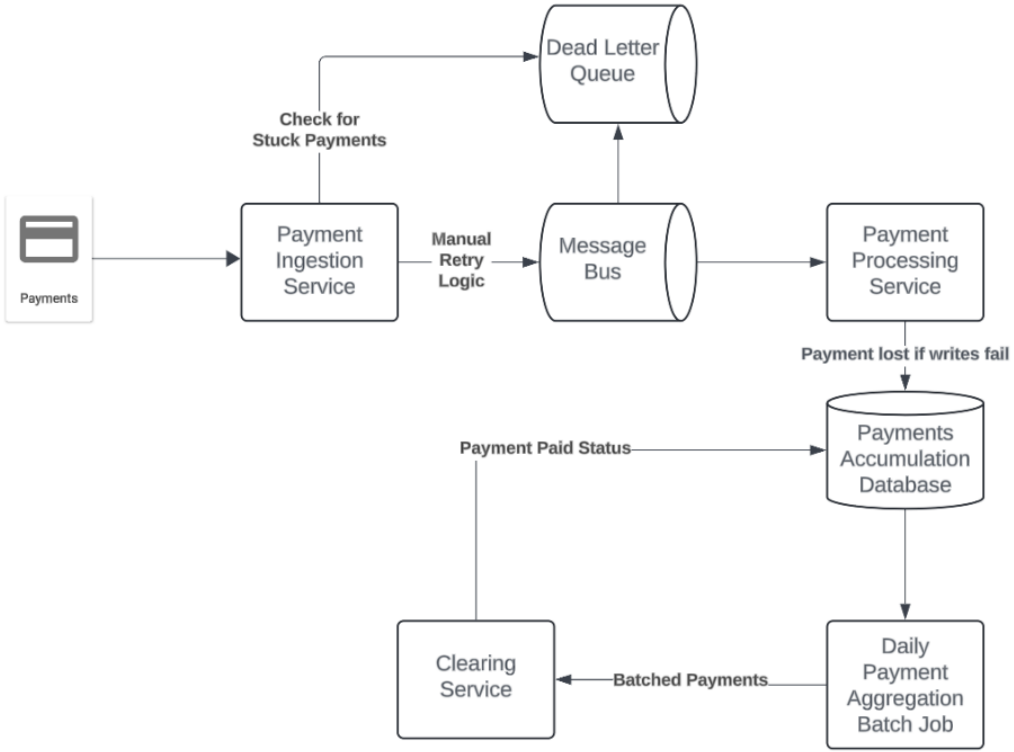
- Massive reliability gains with removal of unreliable cron jobs
- Dramatically improved its visibility into systems
- Code is easy to write, reason about, and unit test

"1/10 the work to run your application than conventional microservices architectures"

"Temporal is extremely important for us because it ensures nothing fails. When something fails, Temporal retries and fixes the issue for us."

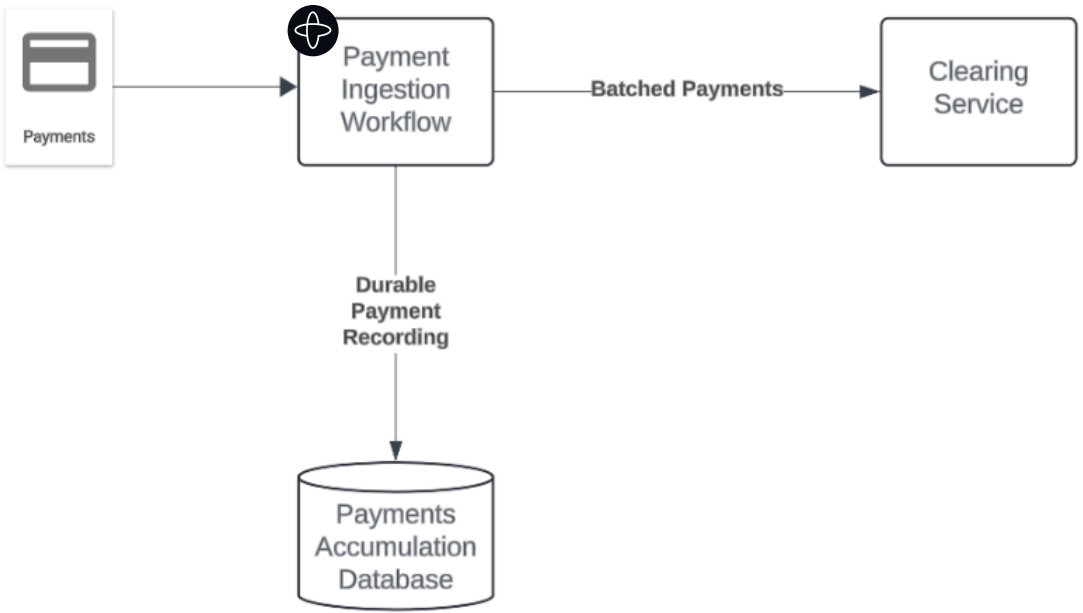
Fabien Huet
CTO

Example: Payment Accumulations



Payment Processing I Architecture after Temporal

Example: Payment Accumulations



Customer Onboarding

Customer Onboarding | Overview

Customer onboarding apps manage the set up and activation of accounts for new customers. These systems have multiple steps, often **asynchronous**, and **touch many services and integrations**. Steps include:

- Record user information
- Verify identity
- Provision resources
- Set up payment integration
- Authorize to different systems
- Send re-engagement emails to activate users

Customer Onboarding | Challenges

Challenge	Impact
Relies on failure-prone external integrations	✗ Manual retries and on-call intervention take up dev time
Long-running processes with time-based notifications	✗ Manual coding of timers; failures may prevent notifications
Multi-step, asynchronous workflows requiring auth to many systems	✗ Complexity inhibits dev productivity

Customer Onboarding | Why Temporal

Temporal Value	Impact
Automatic retries of integrations	✓ Improved reliability & decreased paging of on-call devs
Easily scheduled & reliable long-running processes	✓ Improved reliability & dev productivity
Orchestrates multi-step flows	✓ Simpler dev experience & enhanced productivity

checkr

Top HR Software Company

Leading digital experience company

Top digital experience company modernized onboarding platform

Challenges

One of the world's largest digital experience companies needed to streamlining an onboarding platform that has over 200 teams building and release components of it.

- Lack of visibility made it challenging to coordinate
- Disparate frameworks, terms, and dev patterns

Solution

Adopted Temporal to streamline the platform and create a unified developer experience that can span 200 teams.

- Visibility into every step of every execution
- Single shared framework across teams

"Temporal gives teams a common language, deeper visibility into progress and failures, and the ability to develop and release loosely coupled services without complex error handling logic and code boundaries."

Senior Software Engineer
Provisioning Team

$20B
Annual Revenue

200
Teams using the platform

Leading HR software company streamlined complex system

Challenges

Goal of simplifying complex, homegrown task management system composed of multiple systems stitched together

- Manual processes (email & spreadsheets)
- Poor employee experience and protracted time-to-value
- Unreliable and difficult to debug and reason about

Solution

Temporal harmonized the complex process into a single stream of synchronous and asynchronous steps

- Reliable integrations
- Full traceability of Workflow executions, live and complete

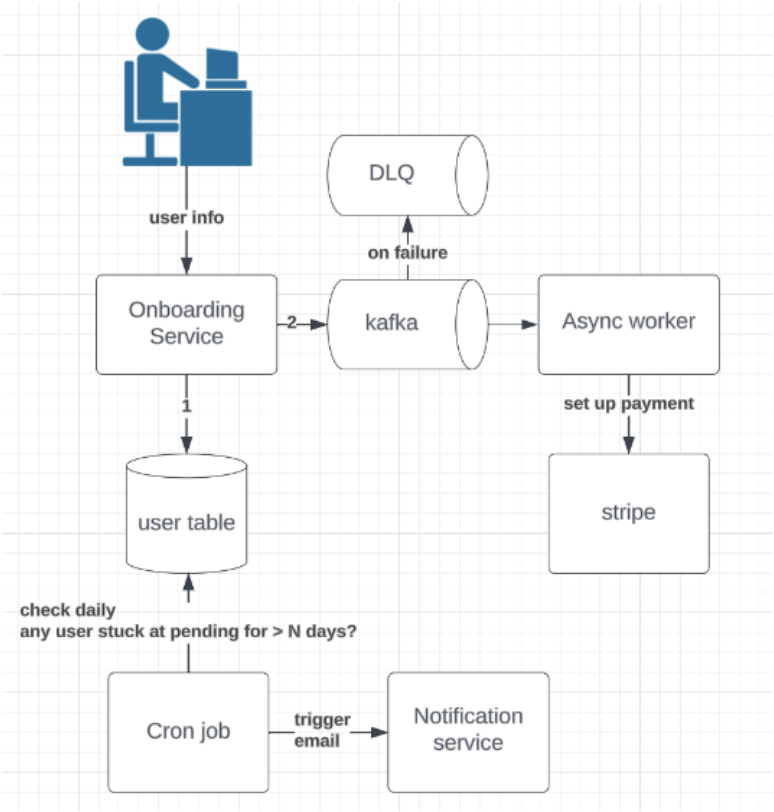
Many teams now approach the Platform Team with existing problems and

ask if they can use Temporal

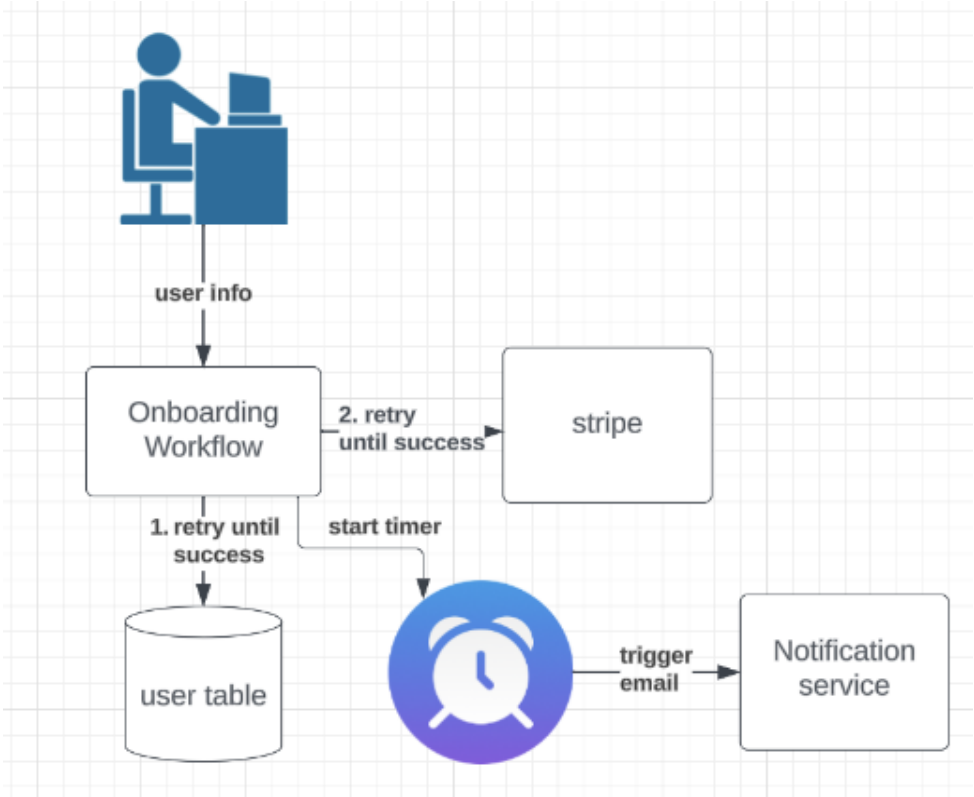
"With Temporal Workflows, suddenly communication becomes easier. Traceability, conceptualization, debugging – it all becomes easier."

Staff Software Engineer

Customer Onboarding I Architecture before Temporal



Customer Onboarding I Architecture after Temporal



Infrastructure Management

Infrastructure Management | Overview

Orchestrate the provisioning and maintenance of database, server, software, or other infrastructure components. These systems perform functions like:

- Provisioning
- Maintenance
- CI/CD

Infrastructure Management | Challenges

Challenge	Impact
Dozens of components with different failure conditions	✗ Manual retries and on-call intervention take up dev time
Lack of visibility	✗ Hours wasted trying to understand what went wrong
Difficult to test updates	✗ Deploy without testing, leading to potential prod issues

Infrastructure Management | Why Temporal

Temporal Value	Impact
Exponentially improved reliability and success rate	✓ Better uptime and less on-call paging
Full Workflow visibility to identify and fix problems	✓ Faster time-to-resolution
Test all possible pathways and integrate with many testing suites	✓ Better production reliability

NETFLIX Snap Inc. Brex MAERSK nuon

Netflix improved reliability of CI/CD platform

NETFLIX

Challenges
Goal of improving multi-cloud CI/CD platform and providing layer of abstraction so devs can focus on innovation

- Transient failures could derail long running workflows
- Difficult to test workflows

Solution
Rebuilt control plane with Temporal

- Improved reliability, with indefinite retries by default
- Easy, through testing. Temporal's JUnit integration for Java allows verification of every branching code path

Deployment failure rate dropped from

4% → 0.0001%

"The Temporal team is fantastic so our support burden is quite low, since we lean on them with Temporal Cloud."

Rob Zienert
Senior Software Engineer

Nuon increased developer productivity by 36x

nuon

SaaS applications delivery platform

Challenges
Goal of improving infrastructure platform for builds, deploys, provisions, and health checks.

- Built with home-grown choreography pattern
- Slow to develop and buggy
- Minimal visibility

Solution
Rebuilt platform with Temporal in one month

- Faster to develop and build new business logic
- Improved visibility

"[With Temporal], you're able to focus on the real problem at hand. Usually, distributed systems are not your real problem, they're a side effect."

Jon Morehouse
CEO

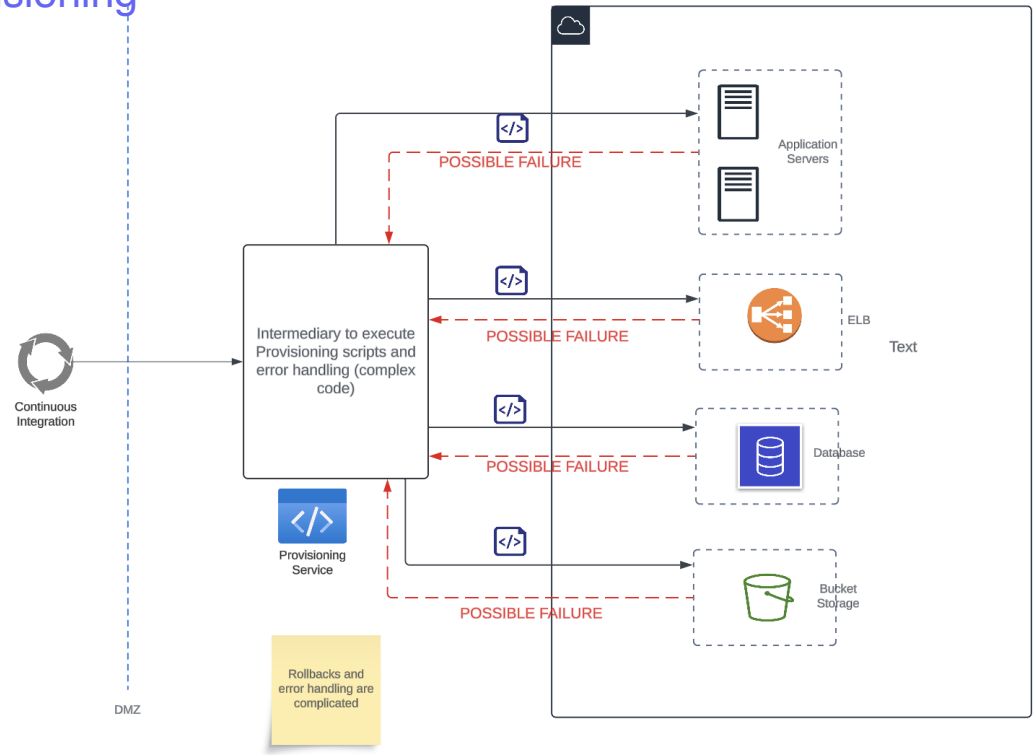
60%
Time saved for new features

50%
Reduction in codebase

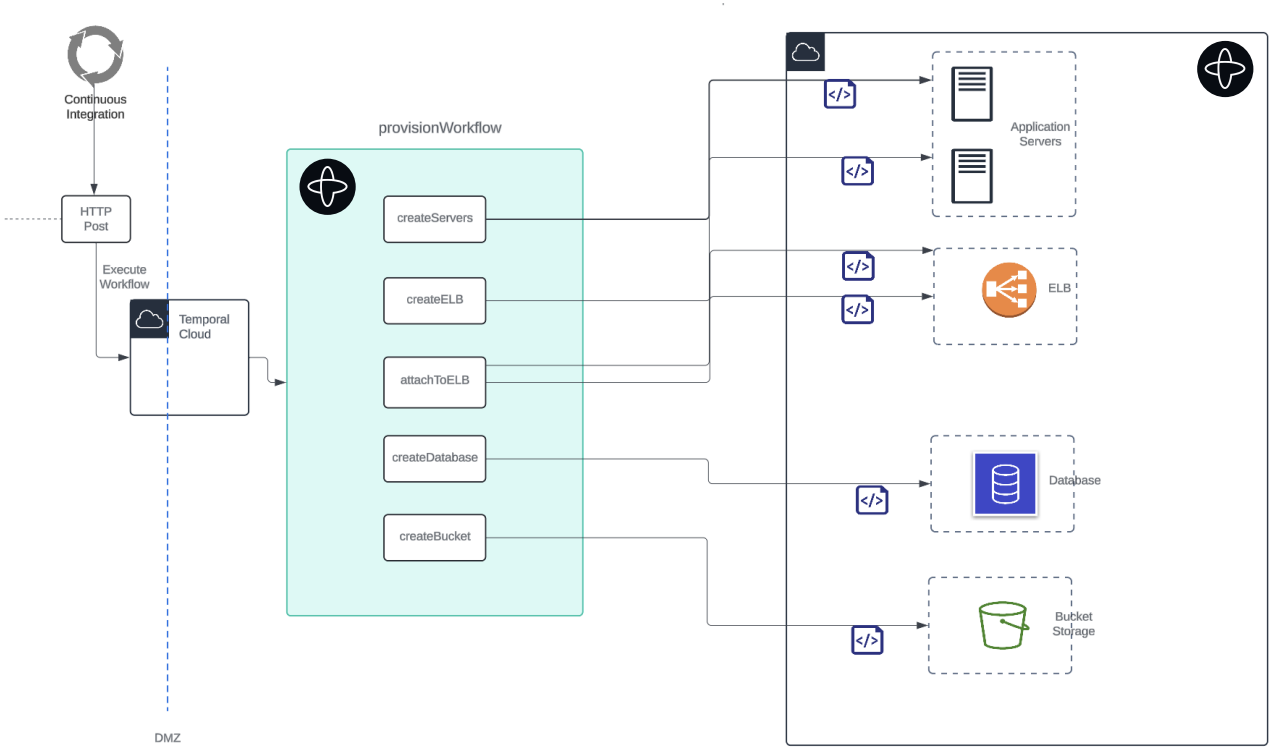
2x
Faster time-to-market

Infrastructure Management | Architecture before Temporal

Example: Provisioning



Example: Provisioning



AI & ML Applications

AI & ML Applications | Overview

Temporal is used for a variety of AI and ML applications that span the full data lifecycle:

- Model training
- Data engineering - multi-step workflows and data pipelines
- AI inferences

AI & ML Applications | Challenges

Challenge	Impact
High volumes of data to process	✗ Potential for bottlenecks
Multiple steps, any of which might fail	✗ Downtime, complex retry logic, and time wasted on jobs that must re-run
Difficult traceability and debugging	✗ Hours wasted trying to identify issues

AI & ML Applications I Why Temporal

Temporal Value	Impact
Horizontal scalability of EDA without the complexity	✓ Better performance and simpler dev experience
Automatic recovery and restart from a failure	✓ More reliable pipelines
Quick visibility into the entire flow	✓ Faster resolution of issues

bugcrowd Crowdsourced Security

MESSARI

dubber

descript

Bugcrowd reduced downtime by 50%

bugcrowd
Crowdsourced Security

Security platform that connects hackers with companies to find vulnerabilities

Challenges
Goal of automating platform that connects hackers to businesses, by automating data pipeline and applying ML

- Existing process required human-in-the-loop
- Monolith was at maximum scale capacity

Solution
Redesigned architecture as Temporal Workflows

- Pass information to ML model through an Activity
- Humans no longer required to match hackers to businesses
- System more efficient, allowing business to thrive

"Temporal's adaptability, resilience, and easy learning curve played a huge role in reshaping our workflow orchestration strategy."

Han Mace
Software Engineer

400%
Scaled business due to Temporal

3x
Faster creation of opportunities

15
Human hours per week saved

Messari increased developer velocity by 4-6x

Challenges
New project to improve pipeline that takes data from social media, enriches it with AI, and distributes it across APIs

- Wanted to avoid audit logs & dead letter queues
- Developer velocity was a top priority

Solution
Built new pipeline using Temporal

- Observability of every workflow and activity
- Reliability and eventual consistency even if APIs fail

MESSARI

Cryptocurrency research, reports, and news summaries

"Temporal makes my life easier. I don't have to do stuff like I did at my previous jobs."

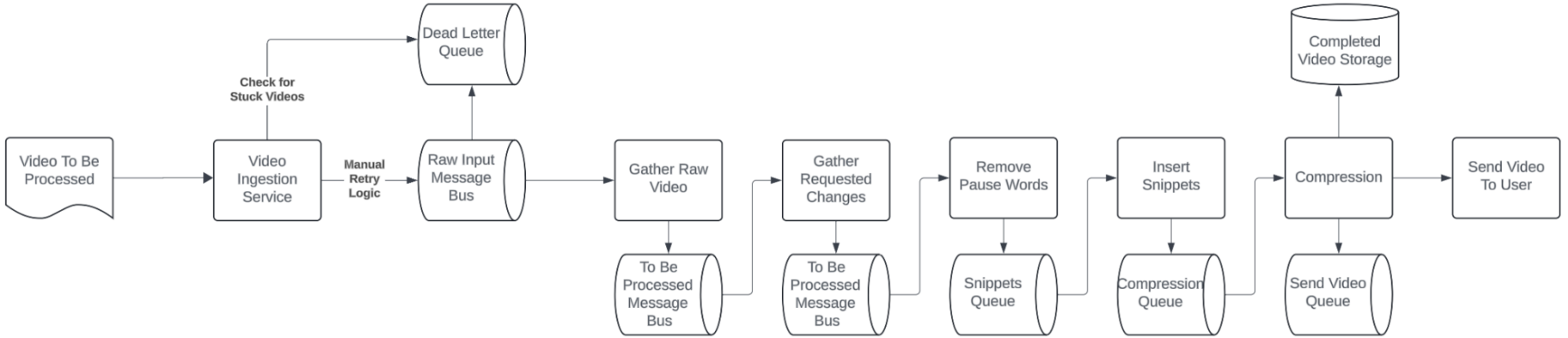
Sohom Bhattacharya
Senior Software Engineer

2x
Fewer engineers required

Faster
Onboarding of new engineers

AI & ML Applications | Architecture before Temporal

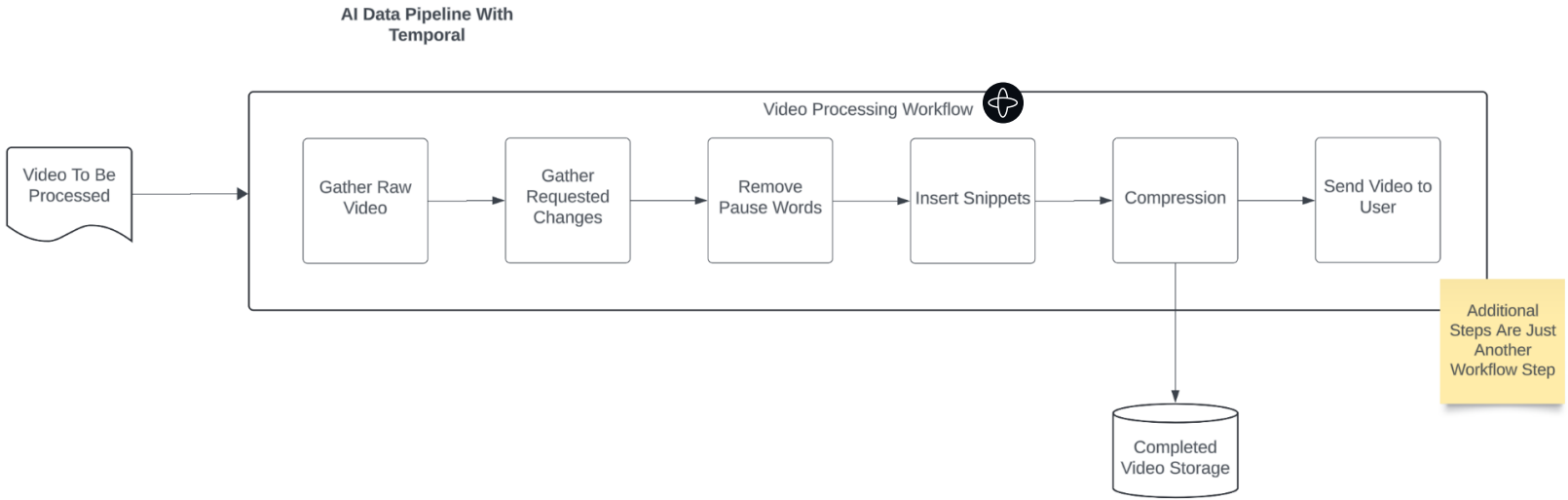
Example: Video Editing with ML



Additional Steps Require More Queues, and Dead Letter Queues

AI & ML Applications | Architecture after Temporal

Example: Video Editing with ML



MORE CUSTOMER STORIES

Snap increased new project velocity by 6x

Challenges
Previously relied on custom workflows built using disparate services, queues, and databases, leading to:

- Slow dev velocity to build and maintain custom workflows
- Difficulty debugging and troubleshooting

Solution
Adopted Temporal for content processing, GenAI and ML workflows, CI/CD pipelines, and other complex logic

- Workflows that took **3-4 days** to build now take **a few hours**
- Debugging, maintenance, and reliability have improved

"It's very hard to build a highly reliable system with good testability and debuggability. Temporal met those requirements."

Jing Huan
Infrastructure Team

800M
Users worldwide

6+
Temporal use cases

DigitalOcean doubled engineering productivity

Challenges
Difficulty managing distributed transactions across multiple data storage locations, resulting in desychnoized data

- Intricate state machines to manage distributed transactions
- Error-prone, resource intensive, and difficult to onboard

Solution
Adopted Temporal to orchestrate distributed transactions and ensure all data is consistent across all storage systems

- 2x fewer developers required to build new storage systems
- Improved reliability and speed of onboarding new engineers

10
engineers to build systems

→

5
engineers to build systems

"Temporal has allowed me to deliver projects that would otherwise be much more complex, and much more difficult, on time and ahead of schedule, and we no longer have this ongoing maintenance burden."

Mike Steger
Staff Engineer, Storage Team

SAP Concur orchestrated a migration to the cloud

SAP Concur

Challenges

Needed a tool to migrate 70+ subsystems from co-located infrastructure to AWS cloud environment

- Many different DBs, including DynamoDB and SQL Server
- Evaluated tools like SWS, Step Functions, and Conductor

Solution

Adopted Temporal to build a migration tool - orchestrated data movement, schema updates, and configuration changes

- Temporal significantly reduced the complexity
- Automatic failure-handling increased efficiency and success, and reduced costs

"Using Temporal, our developers got more comfortable with failure, so that when things did fail, they learned to trust the system."

Sean Donovan
Software Engineer

70+
subsystems to migrate

90,000
databases per subsystem

Turo Enhances Scalability and Developer Velocity

TURO

Challenges

Needed to transition from a complex, legacy monolith to scalable microservices.

- Legacy Java monolith with millions of lines of code
- Slow feature development and scalability bottlenecks

Solution

Adopted Temporal to break down the monolith and streamline microservices.

- Adopted Temporal for reliable transaction processing
- Used workflows-as-code for easier implementation

"Temporal has significantly improved our development velocity and maintainability."

Josh Wickham
Principal Software Engineer

30%
faster feature development

25%
increase in system reliability

Dapper Labs Optimizes Blockchain Payments

Dapper Labs

Challenges

Scaling and maintaining payment workflows in a microservices architecture.

- Complex microservice interactions causing delays
- Proprietary event-sourcing system struggled with scalability

Solution

Adopted Temporal for reliable, scalable payment processing.

- Used Temporal for payment systems and asynchronous workflows
- Leveraged Temporal Cloud for high uptime and easy management

"Temporal has significantly improved our developer velocity by 50 to 60 percent."

Ian Pun
Senior Software Engineer,

50%
increase in developer velocity

40%
reduction in code complexity

Will Bank Streamlines Banking Transfers

Challenges
Improving payment processing reliability and efficiency.

- Complex state management led to errors
- High maintenance overhead with custom solutions

Solution
Adopted Temporal for reliable payment workflows and faster deployment.

- Temporal powers debit authorizer and banking transfers
- Used Temporal Cloud to reduce operational costs

"Temporal enables us to say 'yes' more often to new ideas, allowing us to build and ship faster."

Bruno Panuto
Senior Staff Software Engineer

98%
reduction in transaction errors

50%
faster deployment times

Instacart Simplifies Complex Workflows

instacart

Challenges

Managing complex workflows involving multiple interdependent steps.

- Limited visibility into workflow execution
- Struggled to ensure the reliability and consistency of workflow executions

Solution

Adopted Temporal to manage state, retries, and observability in processes

- Leveraged Temporals ability to accelerate development cycles

10 million+

Workflows per day

"Temporal has impacted our success greatly because the less time we have to think about…how to do the things that Temporal does, the more time we can spend thinking about what kind of business logic we want."

Yarislov Taben

Senior Software Engineer,

Linus Health Boosts Innovation

Challenges

Faced significant challenges with their custom-built, in-house workflow solution, leading to:

- Platform support hindered development of core product
- Issues during Quality Assurance and User Acceptance Testing Stages

Solution

Adopted Temporal

- Can now offload cluster management, focusing only on self-hosted Workers
- Defining workflows as code with Temporal offers greater flexibility for the team

"Another alternative we considered before Temporal was Cadence, but it only offered Java and Go SDKs. We preferred Temporal for its support of Python, which our team already used extensively."

Diane Nguyen
Software Architect

Logikcull Achieves 5x Faster Development

Challenges

Wanted to break out from their monolith architecture into micro services:

- Lacked a high-level overview of their entire pipeline
- Needed a consistent way to build new services to enhance team efficiency

Solution

Adopted Temporal to transform the backend architecture and improve development efficiency:

- Reliable workflow for their audio file redactions needs
- Updated their document processing pipeline, moving a logical step earlier in the process

"Temporal Workflows have been integral to our feature development."

Ehlana Gray
Backend Processing Team

AutoKitteh Increases Development Efficiency

autokitteh

Challenges

Wanted to ensuring that their software was durable and reliable.

- Maintaining program durability and reliability was a struggle
- Built their own state management and event-driven system, causing extra maintenance work

Solution

Adopted Temporal to enhance durability and reliability of their programs while simplifying development processes.

- Developers allocate less time to distributed system development, focusing more on innovation
- Intervening when errors happen has drastically decreased

"We knew we were going to use Temporal because it gave us a huge advantage over other solutions out there."

Itay Donanhirsh
CTO & Founder

Netflix Increases Developer Productivity with Temporal Cloud

NETFLIX

Challenges

Maintaining a complex and unwieldy workflow orchestration system.

- Netflix teams using disparate approaches to managing cloud infrastructure
- Homegrown solutions were error-prone and lacked stability

Solution

Adopted Temporal because it allows code-first workflow development.

- Used Temporal for enhanced testability without running on runtime execution
- Leveraged Temporal Cloud to reduce cognitive load

"What I love about Temporal is that it's productive for both the users and... me as a platform provider. I don't really need to manage it. It's a system that just works. Even when it fails, things will just pick up right back where they started or left off."

Rob Zienert
Senior Software Engineer,

Thank you!

Come JOiN Us...

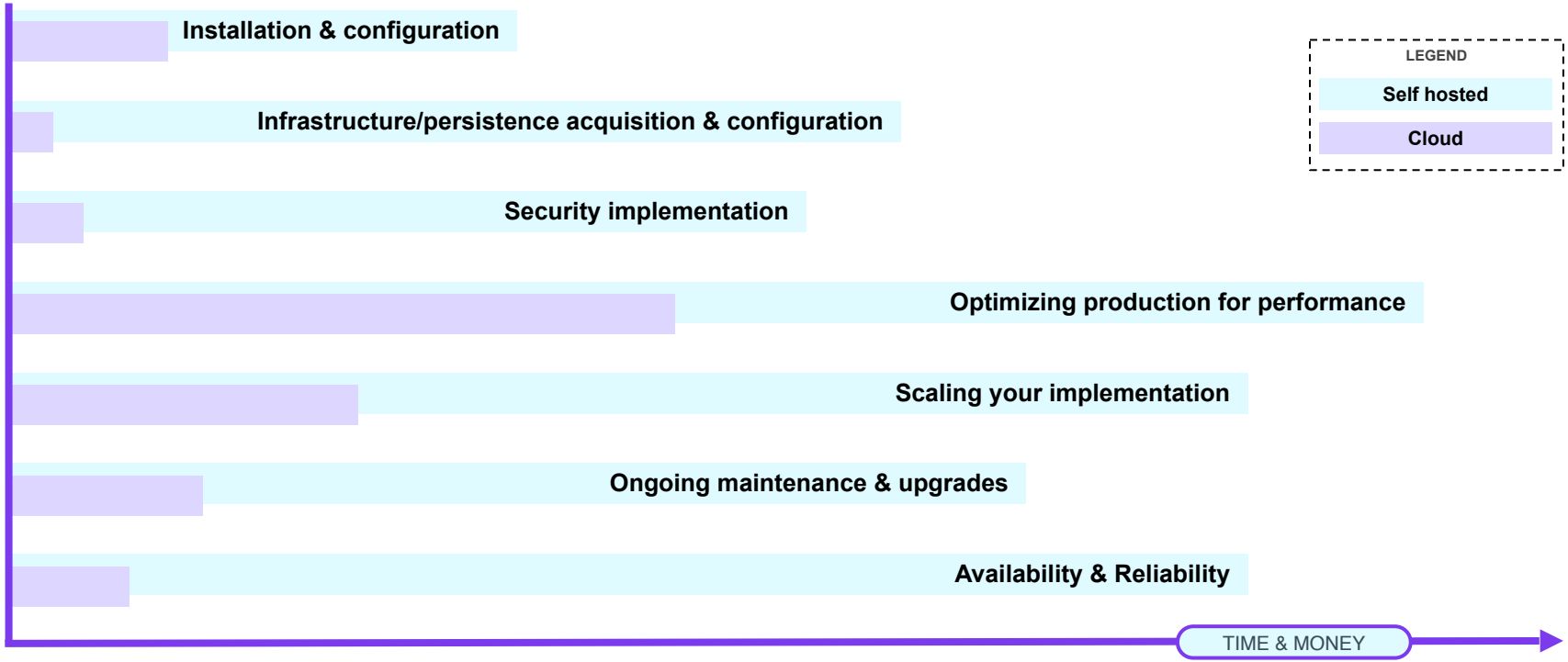
"Not only is the product great, but the Slack community and support are some of the best I've encountered.

an Geslen - Software Engineer

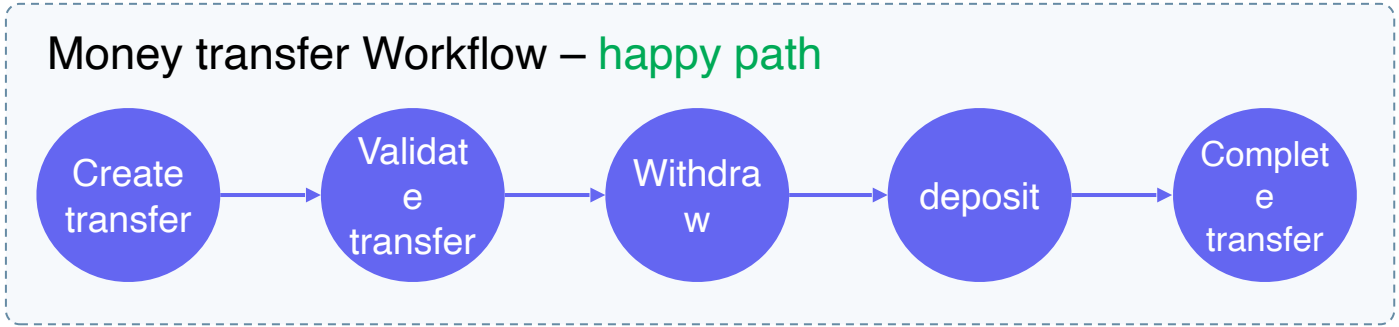
Estimating Costs by workload type

	Cost (monthly)	Characteristics	Actions	Typical use cases
Small	< $25.00	Modest / transient throughput	< 1M / month *< 0.38 actions per second*	Development & testing General automation / orchestration Human dependent processes
Medium	< $1K	Steady or burst throughput	< 40M / month *< 15 actions per second*	Transaction & order systems Infrastructure automation & CICD Batch processes
Large	< $10K	Sustained throughput or multiple use cases	< 400M / month *< 150 actions per second*	Data processing / synchronization National retail order system KYC & fraud detection
Extra Large	$20K+	“Web scale” and / or numerous use cases	1B+ / month *400+ actions per second*	Global enterprise, broad usage Social media application SaaS application service

Temporal Service: operational costs



Sample Temporal Workflow



Start workflow

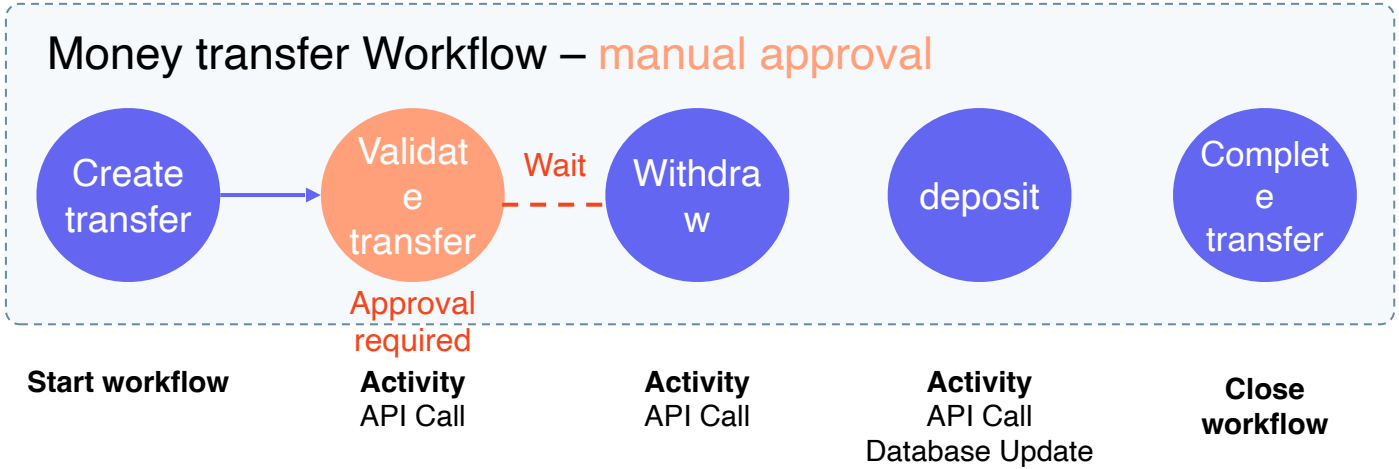
Activity
API Call

Activity
API Call

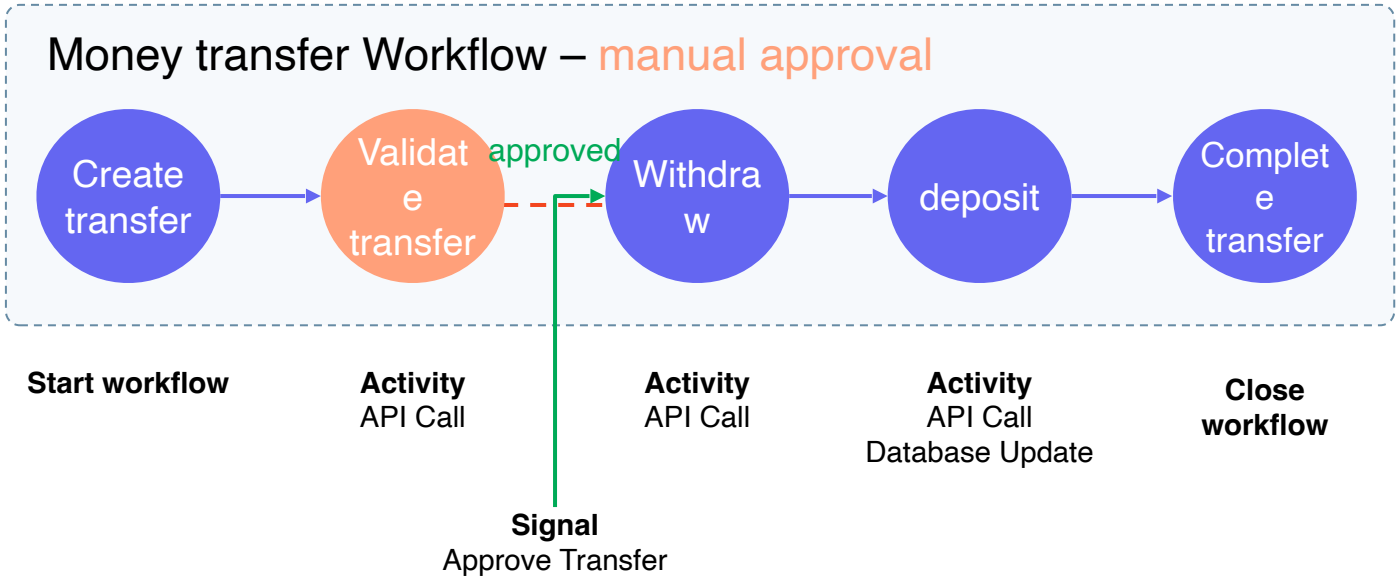
Activity
API Call
Database Update

Close workflow

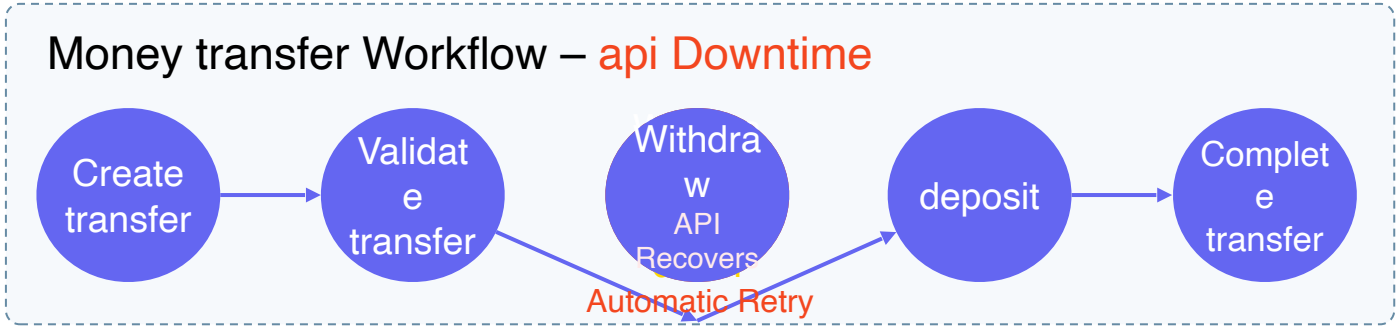
Sample Temporal Workflow



Sample Temporal Workflow



Sample Temporal Workflow

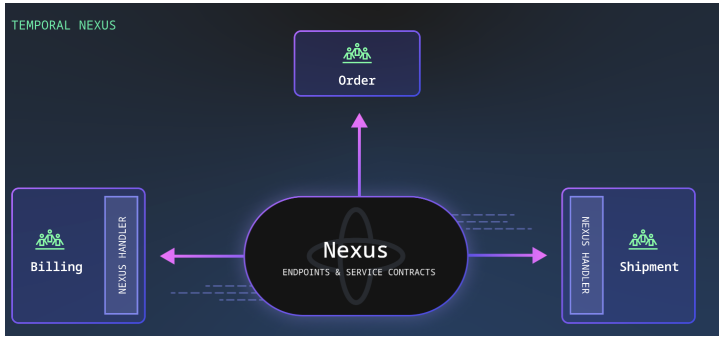


Temporal

Temporal Nexus connects durable executions across team, namespace, region, and cloud boundaries.

What is Nexus?

Nexus provides a fully integrated Temporal experience for connecting durable executions. Use the Temporal SDK to add Nexus services to your existing workers with clean API contracts for others to use. Nexus endpoints are API proxies for Nexus services that abstract Temporal primitives like Workflows for seamless integration and collaboration across teams.



Temporal.io/nexus

Benefits to developers

Familiar API programming model that support both sync and async patterns for connecting durable executions using existing queue-based Workers.

Modular architecture that allows teams to independently develop, scale, and maintain parts of an application with clear API contracts backed by robust Nexus machinery.

Enhanced security, troubleshooting, and fault isolation that reduces the risk of misbehaving workers, since each team can run their workflows in separate namespaces.

> ***Nexus goes a step further by enabling runtime integration of products with an even lighter, more efficient connection***

Bard Wood, Architect, Qualtrics