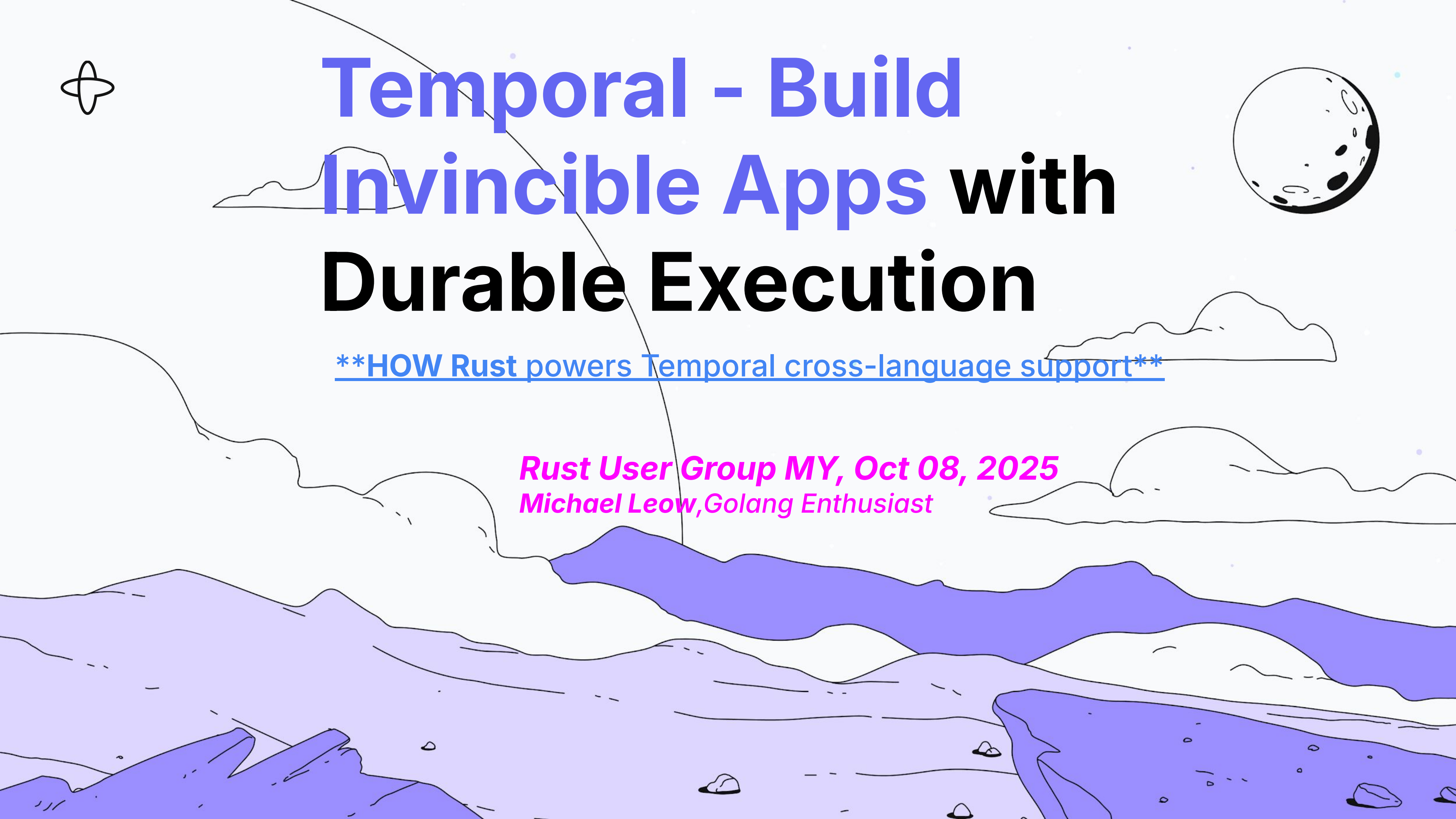
Temporal - Build Invincible Apps with Durable Execution

HOW Rust powers Temporal cross-language support

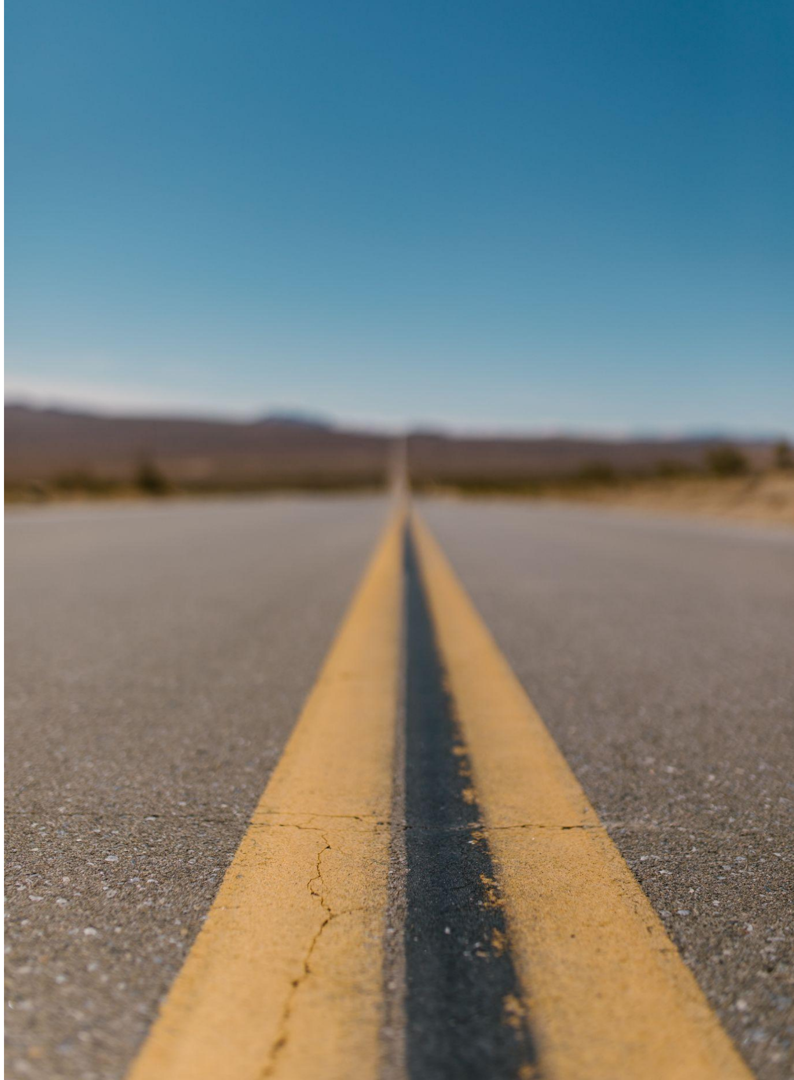


Rust User Group MY, Oct 08, 2025

***Michael Leow**,Golang Enthusiast*

TODAY WE'LL...

1. Review modern applications
2. Introduce Durable Execution and Temporal
3. Temporal "Hello World"
4. Rust in Temporal
5. More Realistic Temporal Demo



MODERNS APPS ARE DISTRIBUTED SYSTEMS

- Functionality split across microservices
- Accessed via the network
- Multiple instances of a single service
- Long-running batch jobs

What kinds of things go wrong in a distributed system?

WHAT GOES WRONG?

- External APIs go down
- Database goes offline or fails
- Queues back up
- Things time out
- EKS Cluster maintenance
- Network partitioning

OTHER CHALLENGES

- Sharing state
- Guarantee idempotent transactions
- Manage long running processes
- Manage unreliable + overlapping cronjob

MORE POWERFUL SYSTEMS, UNHAPPY DEVS

- Less productive and less focused
- Decreased feature velocity
- Systems are less reliable
- Difficult to get end-to-end visibility

WE HAVE THE TOOLS...

- Compensating Actions
- Load Balancing
- Batch Processing
- Queue (SQS), Logs (Kafka)
- but devs have to implement it...
- for every project...
- at every job...

WHAT IF THERE WAS AN EASIER WAY...

DURABLE EXECUTION

- **Durable Execution** is crash-proof execution.
- Development abstraction that:
  - Automatically maintains application state and recovers from failure
  - Ensures that your applications continues execution despite adverse conditions
  - Improves developer productivity by making applications easier to develop, scale, and support

WHAT DOES CRASH PROOF MEAN?

NORMAL VS DURABLE EXECUTION

TEMPORAL

Temporal is an **open-source Durable Execution Platform** that reduces your code’s complexity, makes your applications more **reliable**, and helps you **deliver more features faster.**

TEMPORAL BENEFITS

- Temporal abstracts away complexities around retries, rollbacks, queues, state machines, and timers.
- You write in **your choice of programming language** using Temporal SDKs that are **idiomatic to that language.**
- You deploy and run your code in your environment.
- Temporal provides a fault-tolerant durability layer so you can recover from failure.

TEMPORAL COMPONENTS

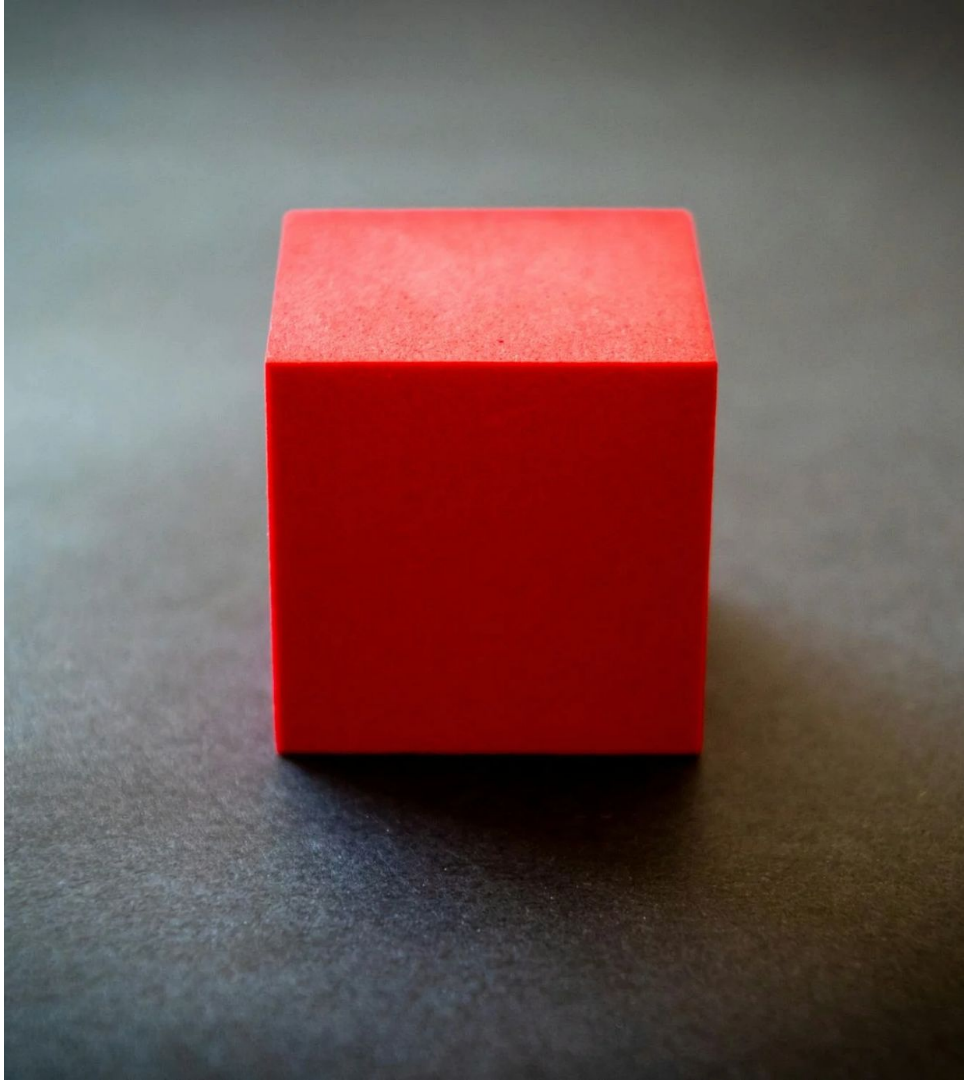
- Workflows
- Activities
- Workers
- Task Queues
- The Temporal Service

WORKFLOWS - THE STEPS OF YOUR PROCESS

- Can maintain state, even after a crash
- Can sleep and receive incoming messages
- Can run - and keep running - for years, even surviving infrastructure failures
- Written in code in a programming language of your choice

ACTIVITIES - THE UNITS OF WORK

- Calling remote services
- Processing files
- Sending an email
- Also written in your programming language of choice



TWO RULES FOR WORKFLOWS AND ACTIVITIES

- Workflow code must be **deterministic** so Temporal can replay it and rebuild state when needed.
- Activities should be **idempotent** in case they fail and are re-run.

WORKFLOWS HELP YOU BUILD DURABLE APPS

- **Workflow** code orchestrates the execution of **Activities**, persisting the results.
- If an Activity fails, the Workflow runs the Activity again ***by default***.
- If the application itself crashes, Temporal will automatically recreate its pre-failure state so it can continue right where it left off.

WORKERS

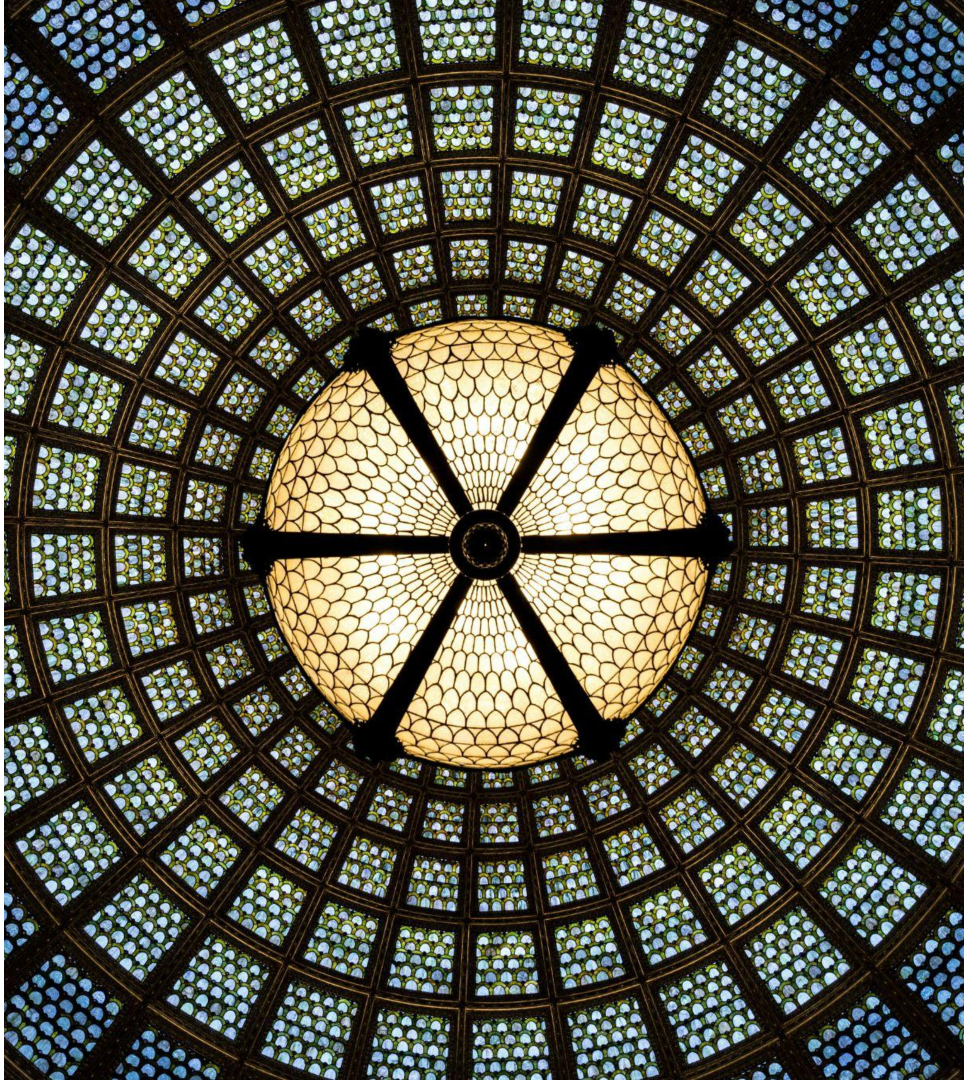
- Execute Workflow and Activity code
- Run on your own servers
- Deploy like any other app

WORKERS AND TASK QUEUES

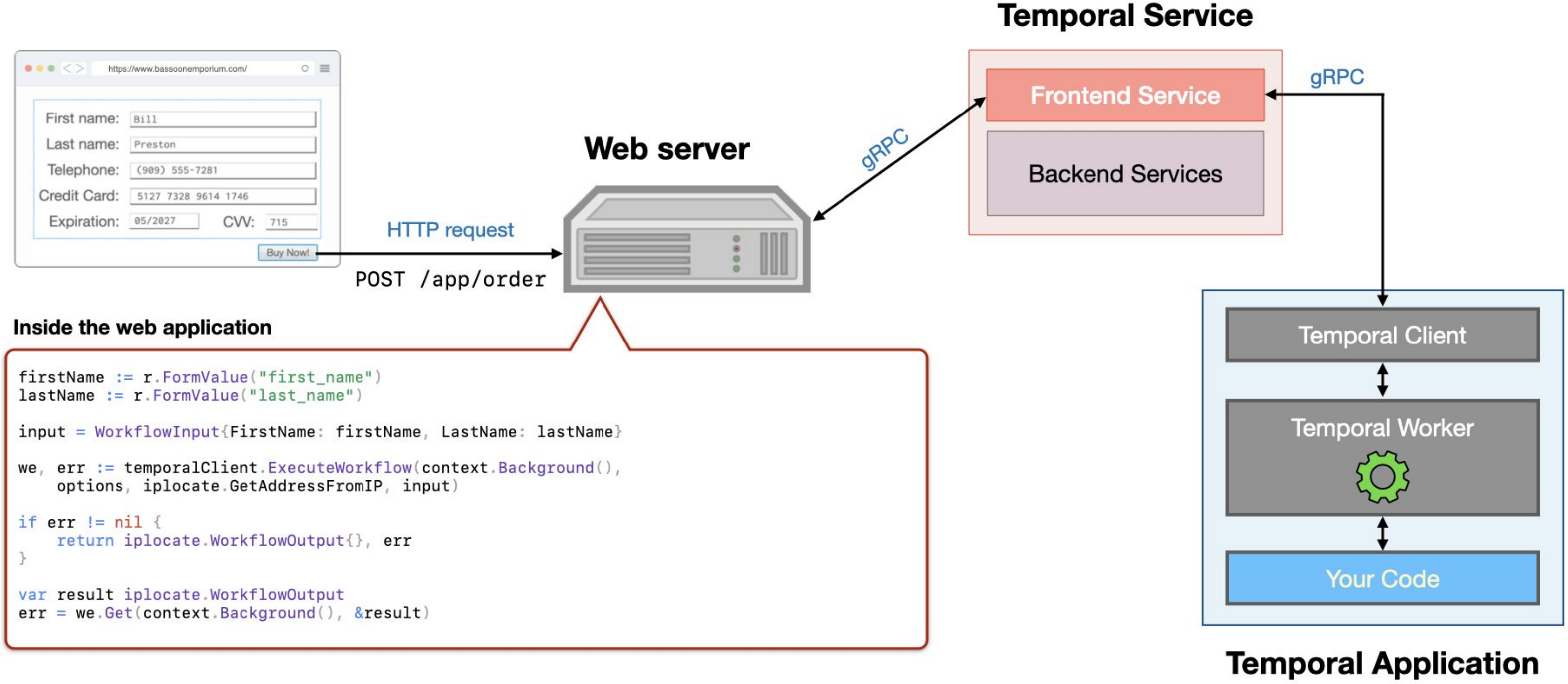
- A Temporal Client starts a Workflow by adding it to a Task Queue.
- Workers poll Task Queues for work to perform.
- As the Worker encounters Activities, it adds those to the Task Queue too.
- Multiple Workers can watch a single Task Queue.

TEMPORAL SERVICE - THE HEART OF IT ALL

- Manages the execution of your Workflows and Activities using Task Queues.
- As your app completes tasks, your app adds events to the Event History on the Service.
- Connections are **outbound only**. Temporal does not need access to your network.



TEMPORAL IN A WEB APPLICATION



https://www.bassoonemporium.com/

First name: Bill

Last name: Preston

Telephone: (909) 555-7281

Credit Card: 5127 7328 9614 1746

Expiration: 05/2027 CVV: 715

Buy Now!

Inside the web application

```
firstName := r.FormValue("first_name")
lastName := r.FormValue("last_name")

input = WorkflowInput{FirstName: firstName, LastName: lastName}

we, err := temporalClient.ExecuteWorkflow(context.Background(),
    options, iplocate.GetAddressFromIP, input)

if err != nil {
    return iplocate.WorkflowOutput{}, err
}

var result iplocate.WorkflowOutput
err = we.Get(context.Background(), &result)
```

WHAT HAPPENS WHEN YOU RUN A WORKFLOW

- A **Workflow** gets scheduled to run, and a **Worker** picks it up, creating a **Workflow Execution**.
- As the **Worker** executes the **Workflow**, it executes local code and **Activities**.
- Results of successful **Activity Executions** are stored in the **Workflow Execution's** Event History, and failed Activity Executions are retried until success or a retry limit is reached.

WHAT MAKES TEMPORAL APPLICATIONS DURABLE?

- In a crash, the Workflow is rerun, rebuilding state by replaying the Event History and rebuilding the Workflow's state.
- Previously completed Activities don't run again.
- When the state is rebuilt, the Workflow resumes execution.

INTERLUDE-1 - RUST WHERE?

- Is this a Golang Talk? Yes :P
- Temporal Server in Golang; CassandraDB Persistence
- SDKs - unified library - abstracts, ensure reliability + fault-tolerance; focus business logic
- Traditional SDK - Golang + Java - inconsistent
- **Rust enables consistent cross-language** SDK development: Python, TypeScript, Ruby, PHP, .NET
- Unofficial SDKs: Swift, Haskell, Scala, Clojure

LET’S BUILD A DURABLE APPLICATION

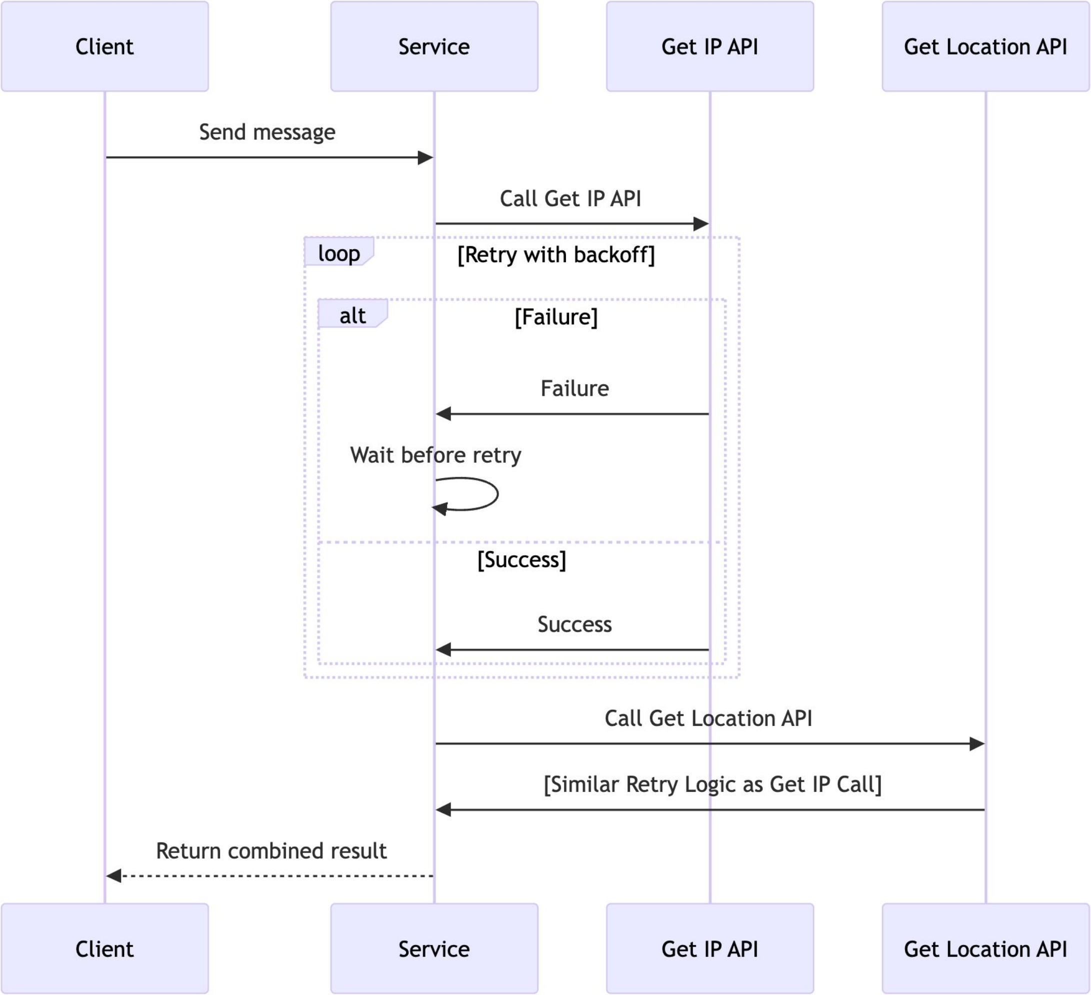
DISTRIBUTED HELLO WORLD

- A new take on an old classic.
- A web application running locally that takes a user’s name as input.
- Makes HTTP requests to two microservices
  - One to get the IP address of the user
  - One to geolocate the IP address

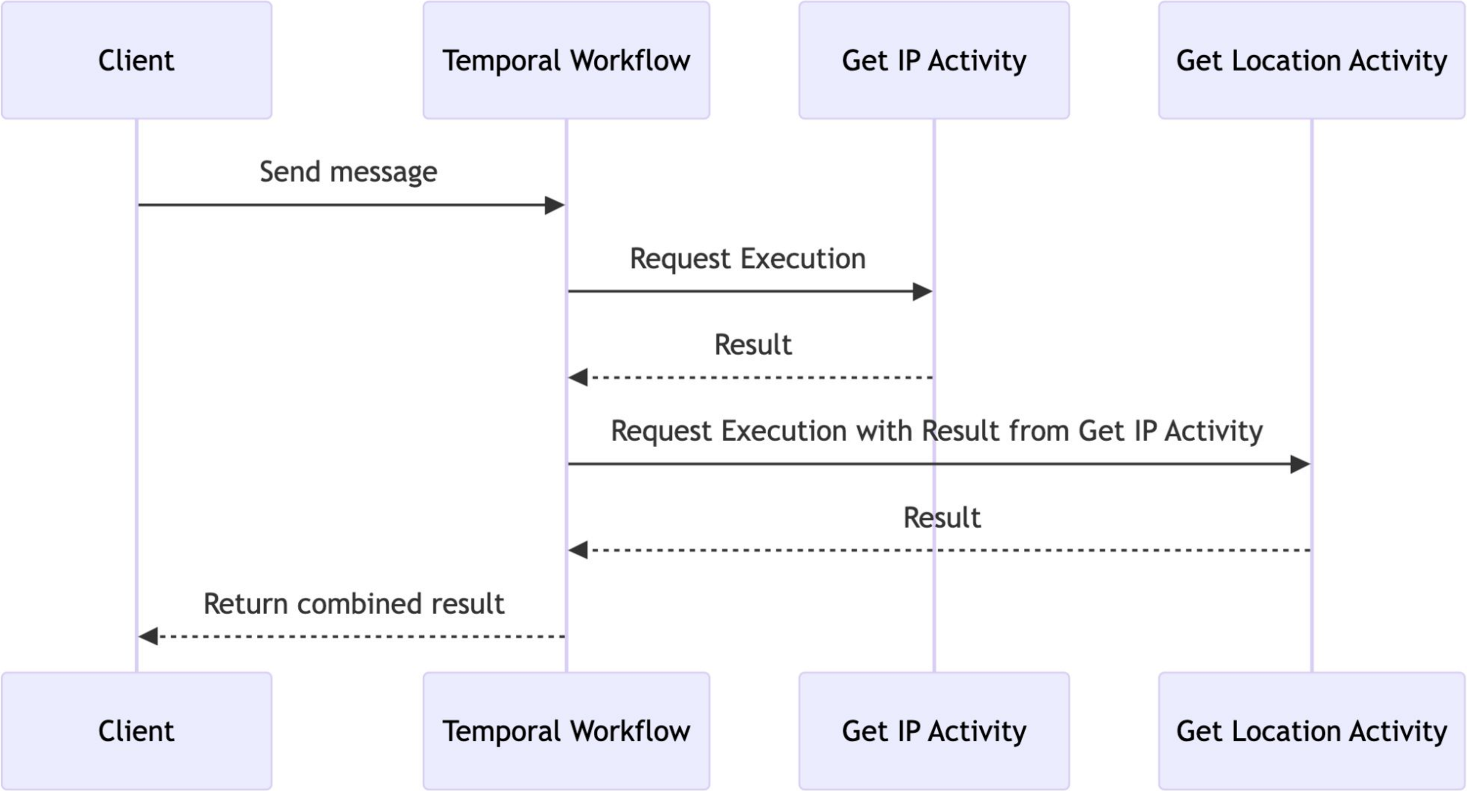
APP REQUIREMENTS

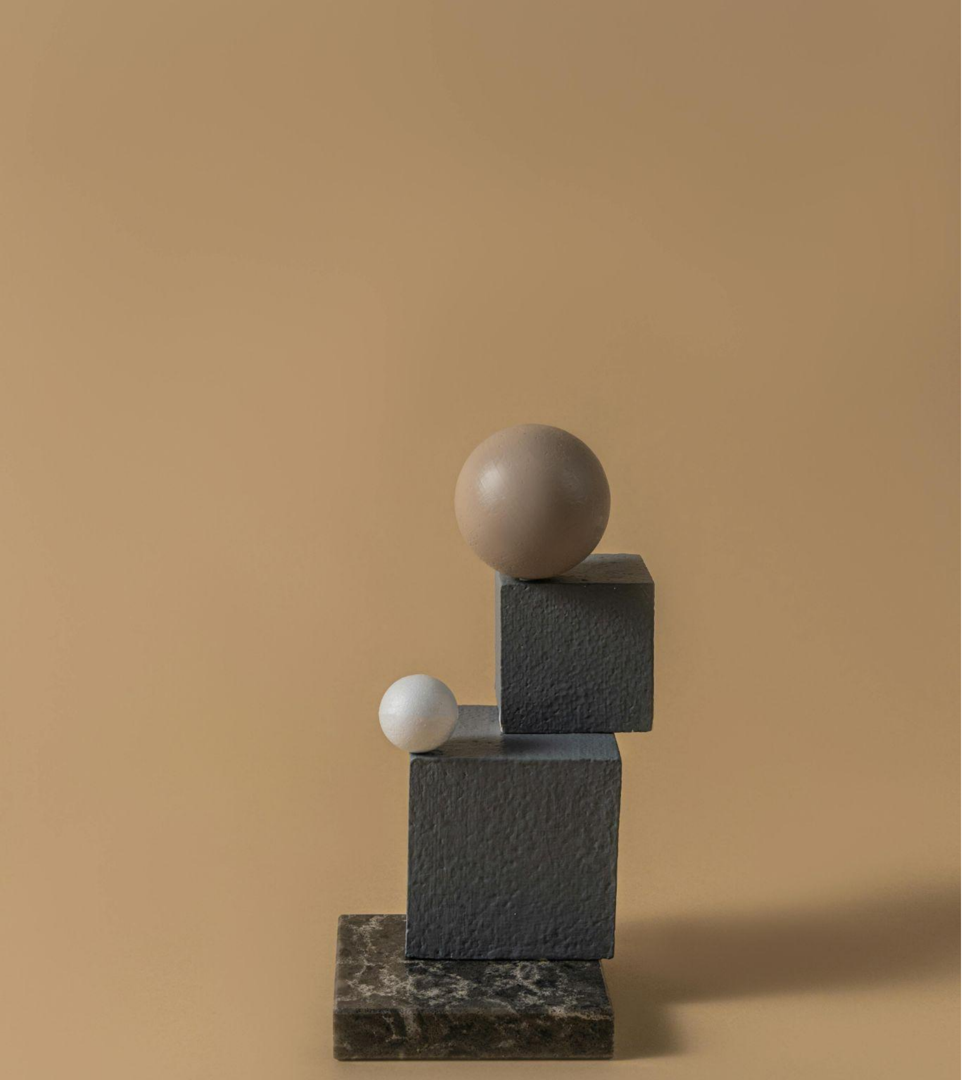
- Requests have to be made sequentially
- If a request to an API fails, retry until it works
- Need to retry without getting rate-limited
- Respond to the user when we get all responses

How would you build this?


Client
Service
Get IP API
Get Location API
Send message
Call Get IP API
loop
[Retry with backoff]
alt
[Failure]
Failure
Wait before retry
[Success]
Success
Call Get Location API
[Similar Retry Logic as Get IP Call]
Return combined result
Client
Service
Get IP API
Get Location API

TEMPORAL REDUCES COMPLEXITY





THE APPLICATION

- Activities
  - get_ip
  - get_location_info
- Workflow - `GetAddressFromIP`
- Worker to execute the Workflow and Activities
- A user interface invoke the Workflow

ACTIVITIES - GET IP ADDRESS

```go
func GetIP(ctx context.Context) (string, error) {
    resp, err := http.Get("https://icanhazip.com")
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := io.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    ip := strings.TrimSpace(string(body))
    return ip, nil
}
```

ACTIVITIES - GET LOCATION INFO

```go
func GetLocationInfo(ctx context.Context, ip string) (string, error) {
    url := fmt.Sprintf("http://ip-api.com/json/%s", ip)
    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := io.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    // Struct and unmarshalling omitted for brevity

    return fmt.Sprintf("%s, %s, %s", data.City, data.RegionName, data.Country), nil
}
```

WORKFLOW - GET ADDRESS FROM IP

```go
func GetAddressFromIP(ctx workflow.Context, name string, seconds int) (string, error) {
    ao := workflow.ActivityOptions{
        StartToCloseTimeout: time.Minute,
    }
    ctx = workflow.WithActivityOptions(ctx, ao)

    var ip string
    err := workflow.ExecuteActivity(ctx, GetIP).Get(ctx, &ip)
    if err != nil {
        return "", fmt.Errorf("Failed to get IP: %s", err)
    }

    var location string
    err = workflow.ExecuteActivity(ctx, GetLocationInfo, ip).Get(ctx, &location)
    if err != nil {
        return "", fmt.Errorf("Failed to get location: %s", err)
    }
    return fmt.Sprintf("Hello, %s. Your IP is %s and your location is %s",
        name, ip, location), nil
}
```

WORKER CONFIGURATION

- Workers need to know
  - The Task Queue to poll
  - The Workflows and Activities to execute
- The Temporal SDK provides the Worker implementation. You configure it
- The Worker executes the Workflow and Activity code you wrote when it finds those tasks on the Task Queue

WORKER - EXECUTES WORKFLOW AND ACTIVITIES

```go
func main() {
    // Create the Temporal client
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create Temporal client", err)
    }
    defer c.Close()

    // Create the Temporal worker
    w := worker.New(c, iplocate.TaskQueueName, worker.Options{})

    // Register Workflow and Activities
    w.RegisterWorkflow(iplocate.GetAddressFromIP)
    w.RegisterActivity(iplocate.GetIP)
    w.RegisterActivity(iplocate.GetLocationInfo)

    // Start the Worker
    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start Temporal worker", err)
    }
}
```

STARTING A WORKFLOW

- With the Temporal CLI
- Through the SDK using a Temporal Client
- Using a gRPC request

CLIENT - CALL TO EXECUTE WORKFLOW

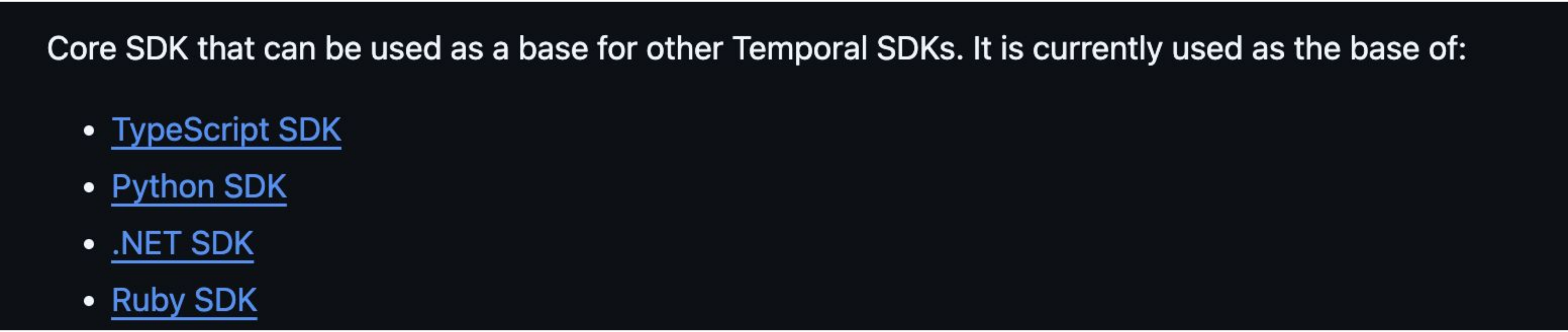
```go
workflowID := "getAddressFromIP-" + uuid.New().String()
    options := client.StartWorkflowOptions{
        ID:        workflowID,
        TaskQueue: iplocate.TaskQueueName,
    }

    we, err := temporalClient.ExecuteWorkflow(context.Background(),
        options, iplocate.GetAddressFromIP, name)
    if err != nil {
        return "", err
    }

    var result string
    err = we.Get(context.Background(), &result)
    return result, err
```

POWERING IT UP

- Start a Temporal Service
  - Temporal CLI dev server
  - Self-hosted
  - Temporal Cloud
- Start Workers, which host your Workflow and Activities
- Invoke a Workflow

INTERLUDE-2 - CODE IN RUST HOW?

- No Rust SDK - use core lib? → https://github.com/temporalio/sdk-core

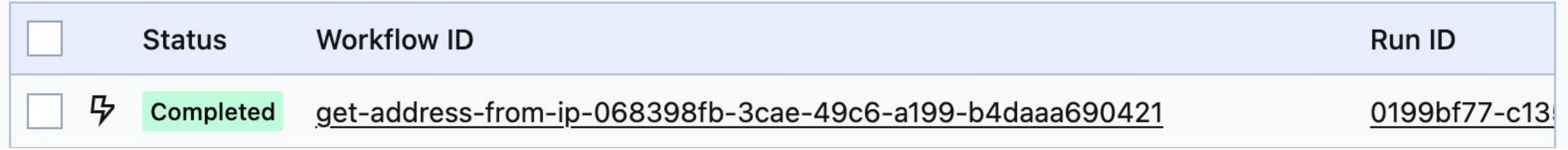


Core SDK that can be used as a base for other Temporal SDKs. It is currently used as the base of:

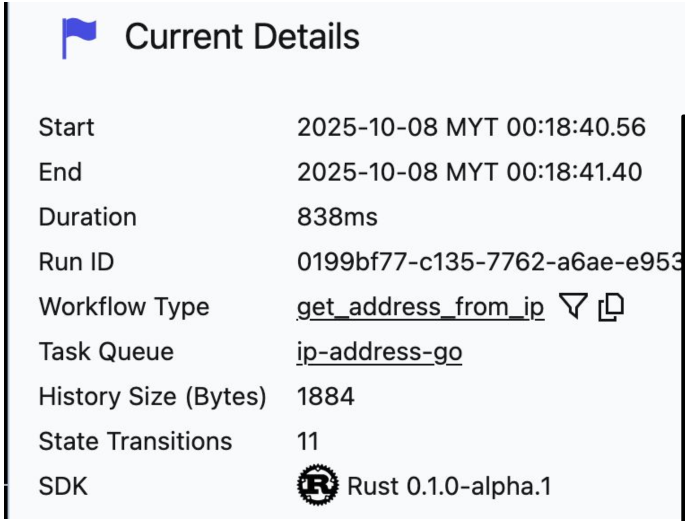
- TypeScript SDK
- Python SDK
- .NET SDK
- Ruby SDK

- Y’all are smart enough ;) - help contribute a port!
- Contribute pls → https://github.com/leowmjw/rust-temporal-demos

RUST-SDK - DEMO!

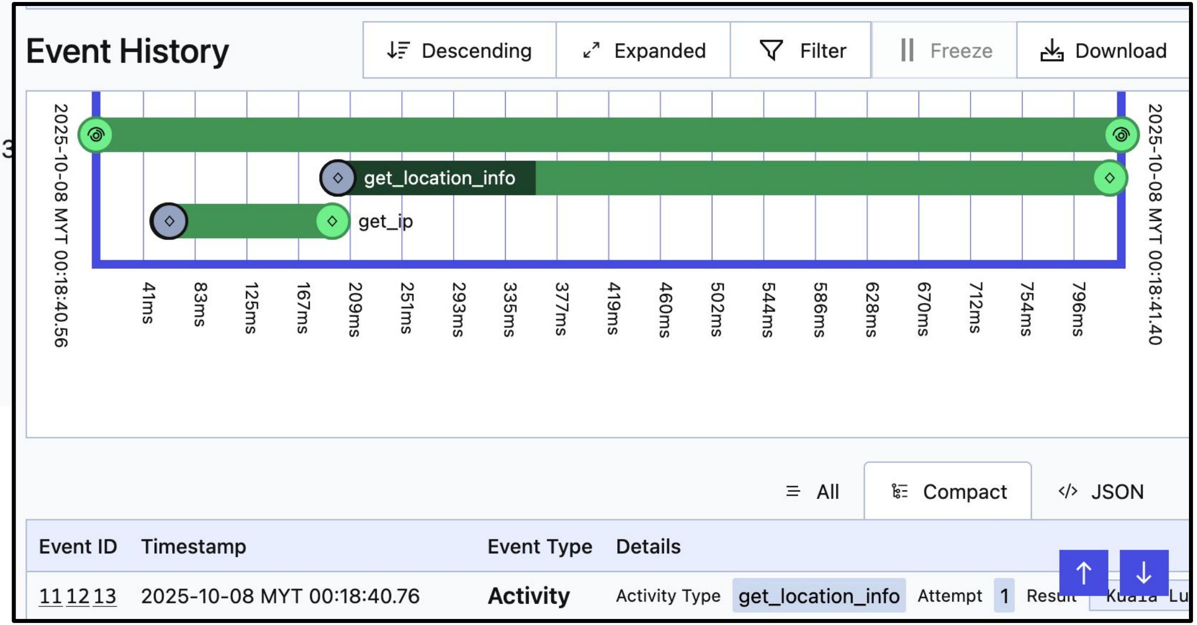


□	Status	Workflow ID	Run ID
□	Completed	get-address-from-ip-068398fb-3cae-49c6-a199-b4daaa690421	0199bf77-c13



Current Details

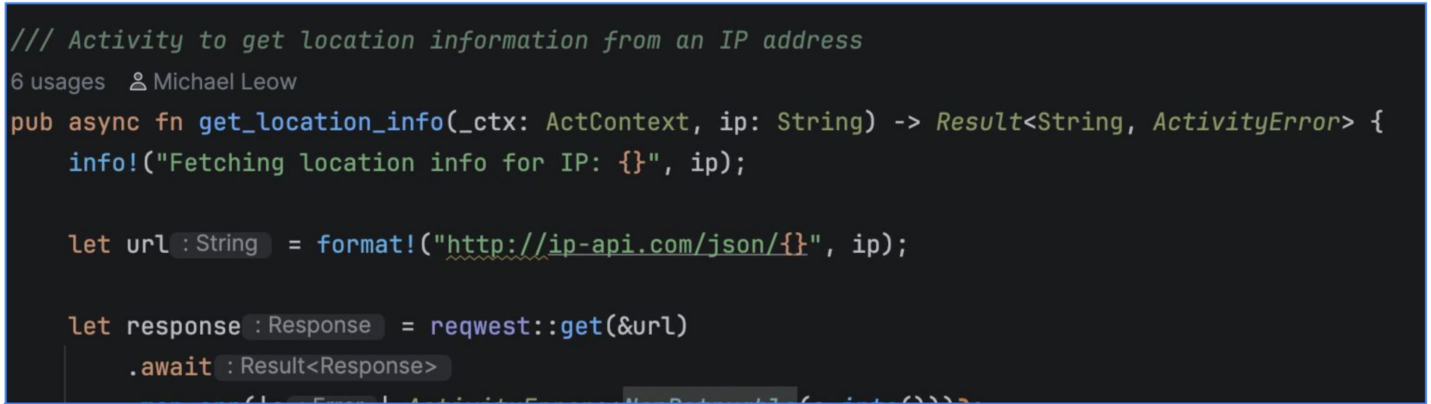
Start	2025-10-08 MYT 00:18:40.56
End	2025-10-08 MYT 00:18:41.40
Duration	838ms
Run ID	0199bf77-c135-7762-a6ae-e953
Workflow Type	get_address_from_ip
Task Queue	ip-address-go
History Size (Bytes)	1884
State Transitions	11
SDK	Rust 0.1.0-alpha.1



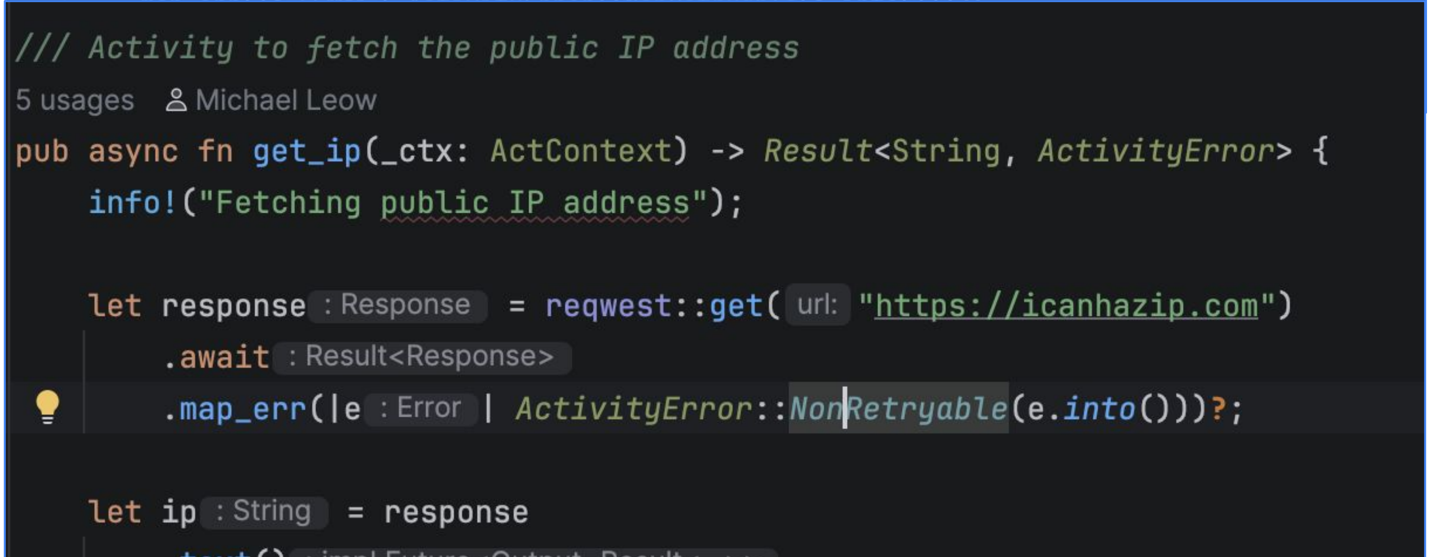
Event History

Event ID	Timestamp	Event Type	Details
11 12 13	2025-10-08 MYT 00:18:40.76	Activity	Activity Type get_location_info Attempt 1 Result Kuala Lu

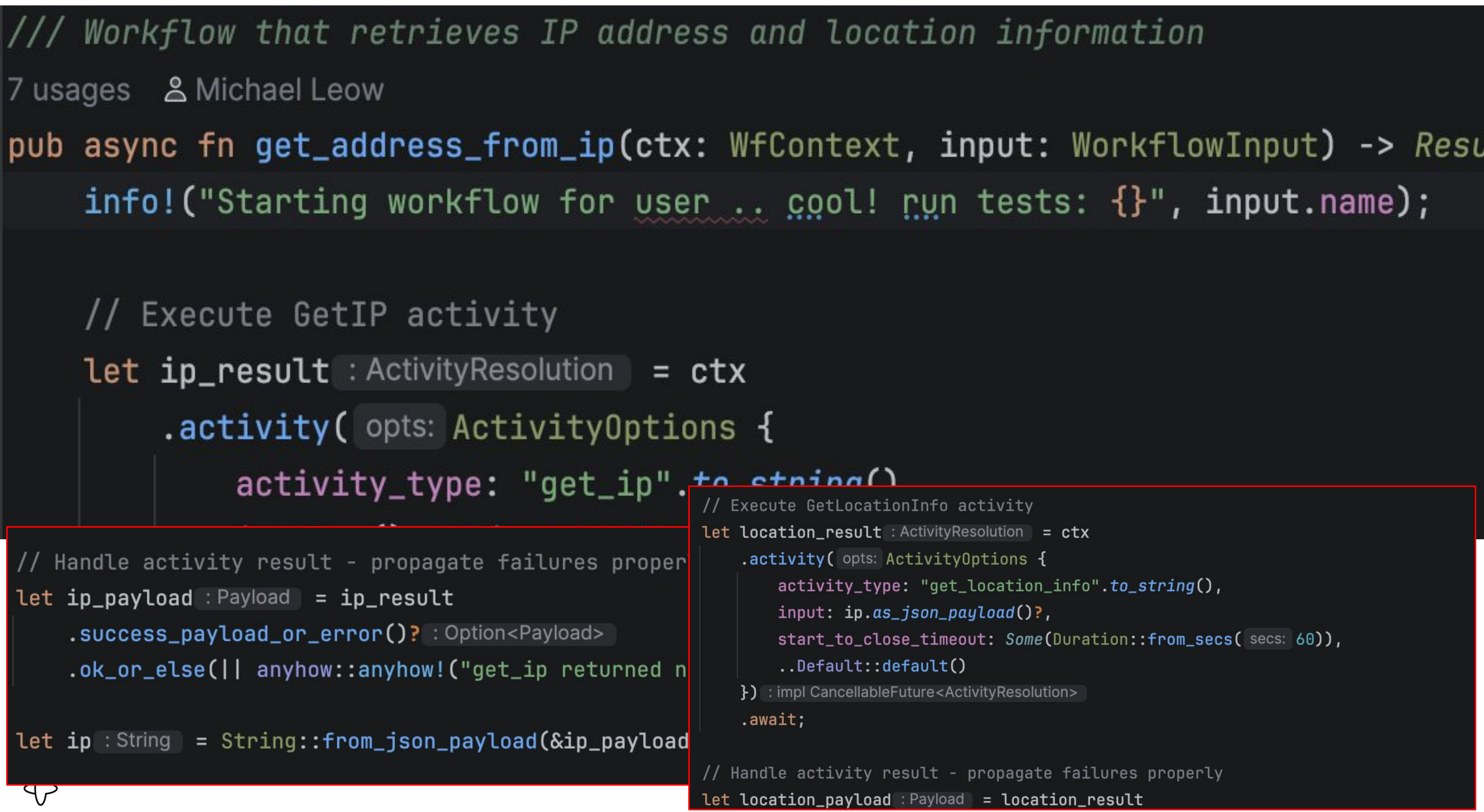
RUST-SDK - CODE!



```
/// Activity to get location information from an IP address
6 usages  Michael Leow
pub async fn get_location_info(_ctx: ActContext, ip: String) -> Result<String, ActivityError> {
    info!("Fetching location info for IP: {}", ip);

    let url : String = format!("http://ip-api.com/json/{}", ip);

    let response : Response = reqwest::get(&url)
        .await : Result<Response>
```



```
/// Activity to fetch the public IP address
5 usages  Michael Leow
pub async fn get_ip(_ctx: ActContext) -> Result<String, ActivityError> {
    info!("Fetching public IP address");

    let response : Response = reqwest::get( url: "https://icanhazip.com")
        .await : Result<Response>
        .map_err(|e : Error | ActivityError::NonRetryable(e.into()))?;

    let ip : String = response
```



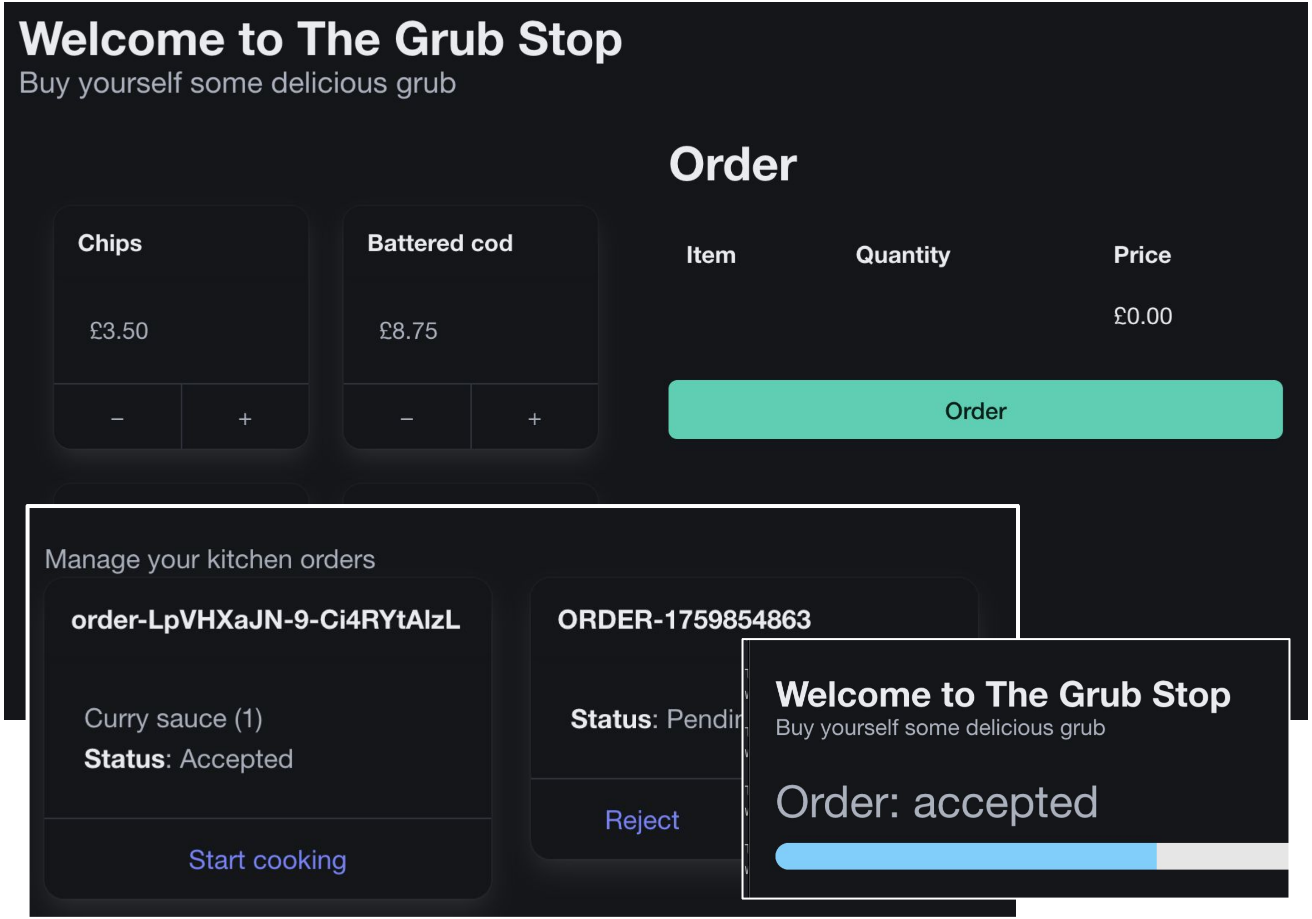
```
/// Workflow that retrieves IP address and location information
7 usages   Michael Leow
pub async fn get_address_from_ip(ctx: WfContext, input: WorkflowInput) -> Resu
    info!("Starting workflow for user .. cool! run tests: {}", input.name);

    // Execute GetIP activity
    let ip_result : ActivityResolution = ctx
        .activity( opts: ActivityOptions {
            activity_type: "get_ip".to_string()
```

```
// Handle activity result - propagate failures proper
let ip_payload : Payload = ip_result
    .success_payload_or_error()? : Option<Payload>
    .ok_or_else(|| anyhow::anyhow!("get_ip returned n

let ip : String = String::from_json_payload(&ip_payload
```

```
// Execute GetLocationInfo activity
let location_result : ActivityResolution = ctx
    .activity( opts: ActivityOptions {
        activity_type: "get_location_info".to_string(),
        input: ip.as_json_payload()?,
        start_to_close_timeout: Some(Duration::from_secs( secs: 60)),
        ..Default::default()
    }) : impl CancellableFuture<ActivityResolution>
    .await;

// Handle activity result - propagate failures properly
let location_payload : Payload = location_result
```

REAL APP: FOOD DELIVERY - 1

- Develop your own Grab Sandwich™ Durable App
- Worker hosted on old Rasberry-Pi; dial-up network
- Ensure no loss!



Welcome to The Grub Stop

Buy yourself some delicious grub

Chips

£3.50

Battered cod

£8.75

Order

Item	Quantity	Price
		£0.00

Manage your kitchen orders

order-LpVHXaJN-9-Ci4RYtAlzL

Curry sauce (1)

Status: Accepted

ORDER-1759854863

Status: Pendir

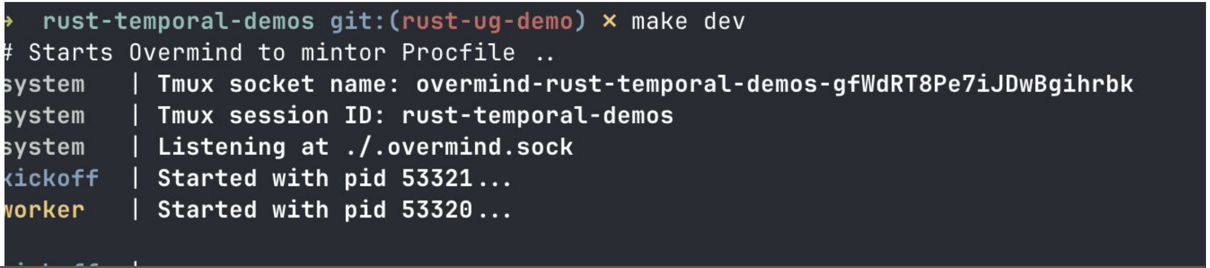
Welcome to The Grub Stop

Buy yourself some delicious grub

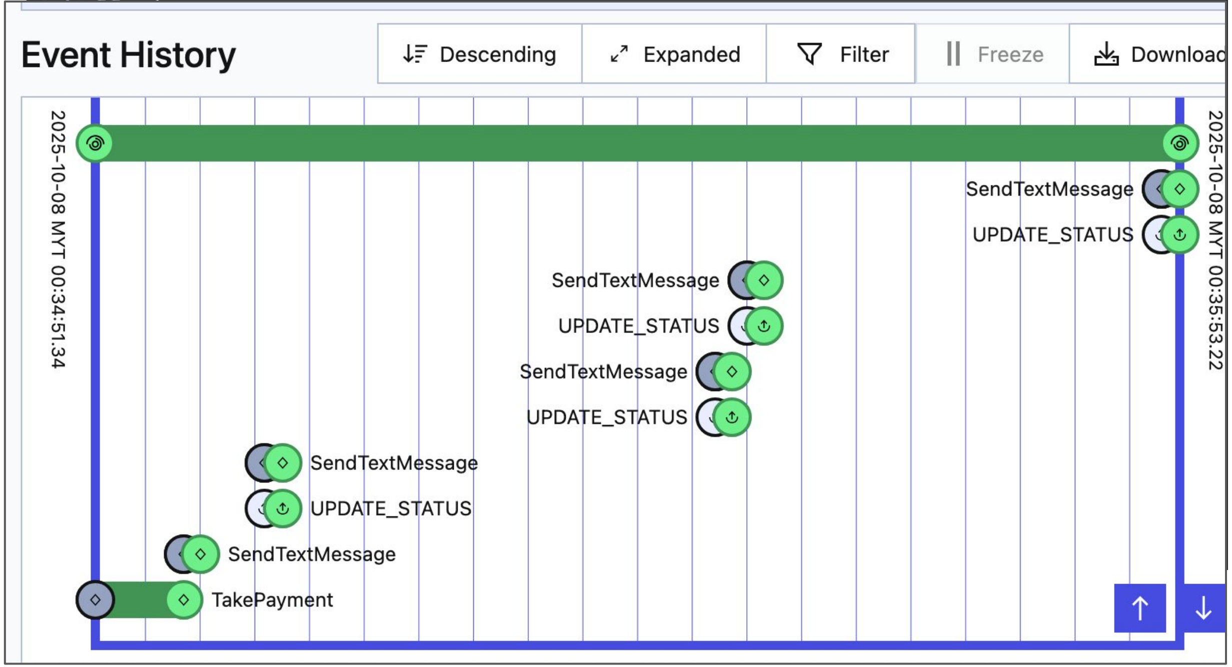
Order: accepted

REAL APP: FOOD DELIVERY - 2

- Auto start workflow
- Manual order
- Take Worker down before Payment - recovers?
- Use Schedule instead of CronJob
- Temporal Down?



```
→ rust-temporal-demos git:(rust-ug-demo) ✗ make dev
# Starts Overmind to mintor Procfile ..
system   | Tmux socket name: overmind-rust-temporal-demos-gfWdRT8Pe7iJDwBgihrbk
system   | Tmux session ID: rust-temporal-demos
system   | Listening at ./.overmind.sock
kickoff  | Started with pid 53321...
worker   | Started with pid 53320...
```



GOING TO PRODUCTION

- **Temporal does not run your code.**
- Deploy your Temporal Application as you would any other application
- Scale your Workers to meet the needs of your customers
- Ensure your network can make **outbound** connections over TCP 7233
- Codec to encrypt sensitive data

A PRODUCTION TEMPORAL SERVICE

- Self Host Open Source
  - MIT License
  - Scale as needed
  - Helm charts available
- Temporal Cloud
  - **Same as open source, just run by Temporal**
  - Managed w/ SLA

YOU'RE IN GOOD COMPANY

Financial Services

JPMorgan Chase & Co.

HSBC

Deutsche Bank

VISA

ANZ

standard chartered

usbank

coinbase

SoFi

PLAID

HDFC BANK

ROCKET Money

BLOCK

Transportation

TURO

TESLA

Alaska

cruise

Grab

lyft

Retail & Consumer

Walmart

Yum!

KFC

Pizza Hut

TACO BELL

the Habit BURGER GRILL

target

NORDSTROM

AMERICAN EAGLE OUTFITTERS

instacart

Office DEPOT

Fanatics

shopify

coupang

Chegg

Digital Media

NETFLIX

Disney

WB

COMCAST

Technology

salesforce

Adobe

Retool

snyk

POSTMAN

armory

SailPoint

Airbyte

LayerZero.

HashiCorp

twilio

qualtrics XM

NUTANIX

citrix

NVIDIA

EPIC GAMES

CISCO

DELL

vmware

PURESTORAGE

DATADOG

SAP

TEMPORAL ALTERNATIVES

Crowded market … who wins?
Who has the best DX?

Restate (Go,Java,Python,Typescript, Rust)

https://docs.restate.dev/

Cloudflare Workflows (Typescript)

https://cloudflare.com/

Inngest (JS)

https://www.inngest.com/

DBOS (Typescript, Python)

https://www.dbos.dev/

Golem Cloud (WASM)

https://www.golem.cloud/

Littlehorse (Go,Java,Python,Typescript,.NET)

https://littlehorse.io/

GETTING STARTED

Join the Temporal Community!

Read the docs! Go → https://learn.temporal.io/tutorials/go/

Follow a tutorial! HOT Topic - AI!! →

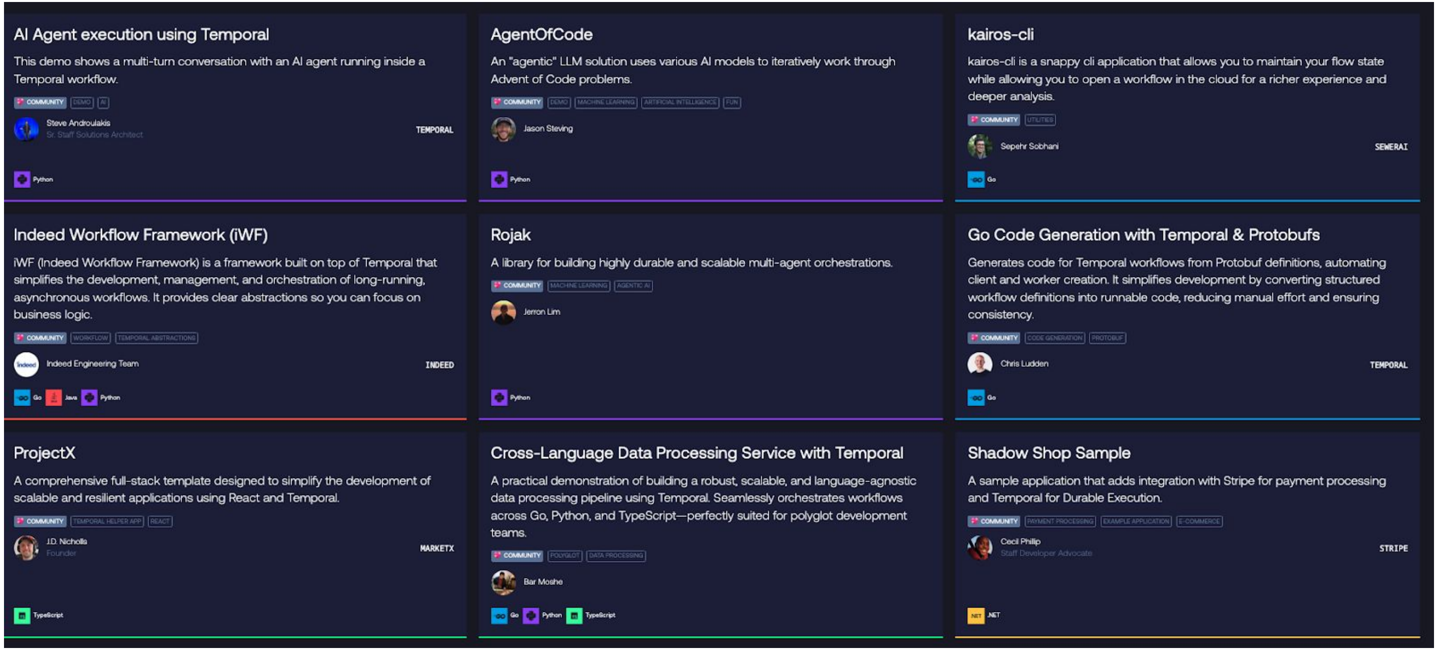
https://learn.temporal.io/tutorials/ai/durable-ai-agent/

Take a free online course! Complete 101 + 102 -

- https://learn.temporal.io/courses/temporal_101/go/
- https://learn.temporal.io/courses/temporal_102/go/

INTRODUCING: TEMPORAL CODE EXCHANGE

Marketplace of ideas to study + learn from. Open to submission!
https://temporal.io/code-exchange



AI Agent execution using Temporal

This demo shows a multi-turn conversation with an AI agent running inside a Temporal workflow.

COMMUNITY DEMO AI

Steve Androulakis
Sr. Staff Solutions Architect

TEMPORAL

Python

AgentOfCode

An "agentic" LLM solution uses various AI models to iteratively work through Advent of Code problems.

COMMUNITY DEMO MACHINE LEARNING ARTIFICIAL INTELLIGENCE FUN

Jason Steving

Python

kairos-cli

kairos-cli is a snappy cli application that allows you to maintain your flow state while allowing you to open a workflow in the cloud for a richer experience and deeper analysis.

COMMUNITY UTILITIES

Sepehr Sobhani

SEWERAI

Go

Indeed Workflow Framework (iWF)

iWF (Indeed Workflow Framework) is a framework built on top of Temporal that simplifies the development, management, and orchestration of long-running, asynchronous workflows. It provides clear abstractions so you can focus on business logic.

COMMUNITY WORKFLOW TEMPORAL ABSTRACTIONS

Indeed Engineering Team

INDEED

Go Java Python

Rojak

A library for building highly durable and scalable multi-agent orchestrations.

COMMUNITY MACHINE LEARNING AGENTIC AI

Jerron Lim

Python

Go Code Generation with Temporal & Protobufs

Generates code for Temporal workflows from Protobuf definitions, automating client and worker creation. It simplifies development by converting structured workflow definitions into runnable code, reducing manual effort and ensuring consistency.

COMMUNITY CODE GENERATION PROTOBUF

Chris Ludden

TEMPORAL

Go

ProjectX

A comprehensive full-stack template designed to simplify the development of scalable and resilient applications using React and Temporal.

COMMUNITY TEMPORAL HELPER APP REACT

J.D. Nicholls
Founder

MARKETX

TypeScript

Cross-Language Data Processing Service with Temporal

A practical demonstration of building a robust, scalable, and language-agnostic data processing pipeline using Temporal. Seamlessly orchestrates workflows across Go, Python, and TypeScript—perfectly suited for polyglot development teams.

COMMUNITY POLYGLOT DATA PROCESSING

Bar Moshe

Go Python TypeScript

Shadow Shop Sample

A sample application that adds integration with Stripe for payment processing and Temporal for Durable Execution.

COMMUNITY PAYMENT PROCESSING EXAMPLE APPLICATION E-COMMERCE

Cecil Phillip
Staff Developer Advocate

STRIPE

.NET

RESOURCES

- Hello-World Demo App (Go) - https://github.com/temporal-community/miab-build-invincible-apps-go
- Real-life Demo App (Go) - https://github.com/mrsimonemms/temporal-demos
- Real-life Demo App (Go + Rust) + Slides - https://github.com/leowmjw/rust-temporal-demos

Q&A + FAQ

- HOWTO handle Workflow version(s) → https://learn.temporal.io/courses/worker_versioning/
- Cron? KilCron! Use Temporal Schedule - https://docs.temporal.io/develop/go/schedules
- HOWTO Simplify Distributed Systems - avoid it! **Modulith**
- Workshop for Beginners - see GETTING STARTED slide
- Slides + Demo code - see RESOURCES slide
- Rust SDK when? Ask Maxim in Slack + Forum

- END -

Bonus Material Next Page - Invincible SRE Workflows with Temporal!

Enjoy!!

MoneyLion®

Invincible SRE Workflows with Temporal

Apr 08, 2025: SRE KL User Group

Michael Leow, Chee Lim Toh

Slide + Code: *https://github.com/cheelim1/go-temporal-sre*

Disclaimer: *Talks is opinion of speaker; does not reflect position of employer.*

Agenda

What is Temporal?

Getting Rid of Cron Forever for long-running jobs

Granting Superpowers to your humble scripts

Just In Time (JIT) Access Demo

Alternatives to Temporal

Q&A

01
What is Temporal

Durable Execution Platform: An abstraction for building simple, sophisticated, resilient applications

Code like it never fails

Write your business logic as code. Create Workflows that guarantee execution; idempotency guaranteed. Code Activities to handle and retry failure-prone logic. Support patterns: Event-Driven, Saga, Batch, Schedules, State-Machines

Testing + Observability

Comprehensive test suites; including time travel (workflows that takes days, months, years). Event Replays and audit logs with minimal effort. Metrics, tracing, logging available including search to troubleshoot and scaling.

Cross-Platform Support

Write business logic using native SDKs (major languages, communities). Inter-communicate + mix-match as needed. Strong access boundaries within namespace. Teams can securely communicate across namespaces via Nexus

Open Source + Commercial Managed

Full local-dev capabilities in OSS. Fully self-host with own controlled Cassandra cluster. Leverage Managed Temporal Cloud for 200ms SLA; scaling to millions++ of workflows and support

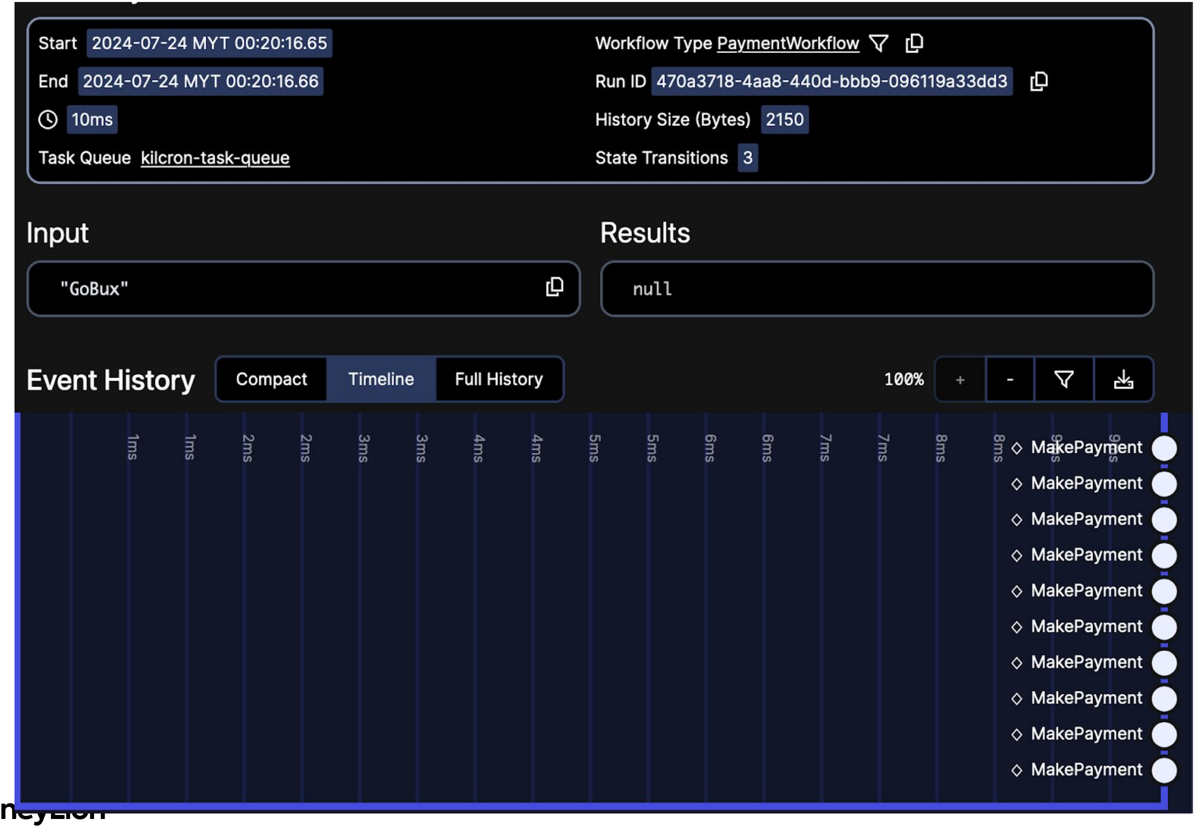
02
Trouble with Cron

Problems with Cron

- **Scene:** Startup getting traction
- Any long running day-to-day process: (e.g reports, payments, data processing)
- **Don’t:** Extend your web server timeout!
- Cron to the rescue!!
- Now got more problems; backfill failures

Cron - Wishful Thinking

- Cron jobs start immediately; no latency, no failure!



Start 2024-07-24 MYT 00:20:16.65

End 2024-07-24 MYT 00:20:16.66

10ms

Task Queue kilcron-task-queue

Workflow Type PaymentWorkflow

Run ID 470a3718-4aa8-440d-bbb9-096119a33dd3

History Size (Bytes) 2150

State Transitions 3

Input

```
"GoBux"
```

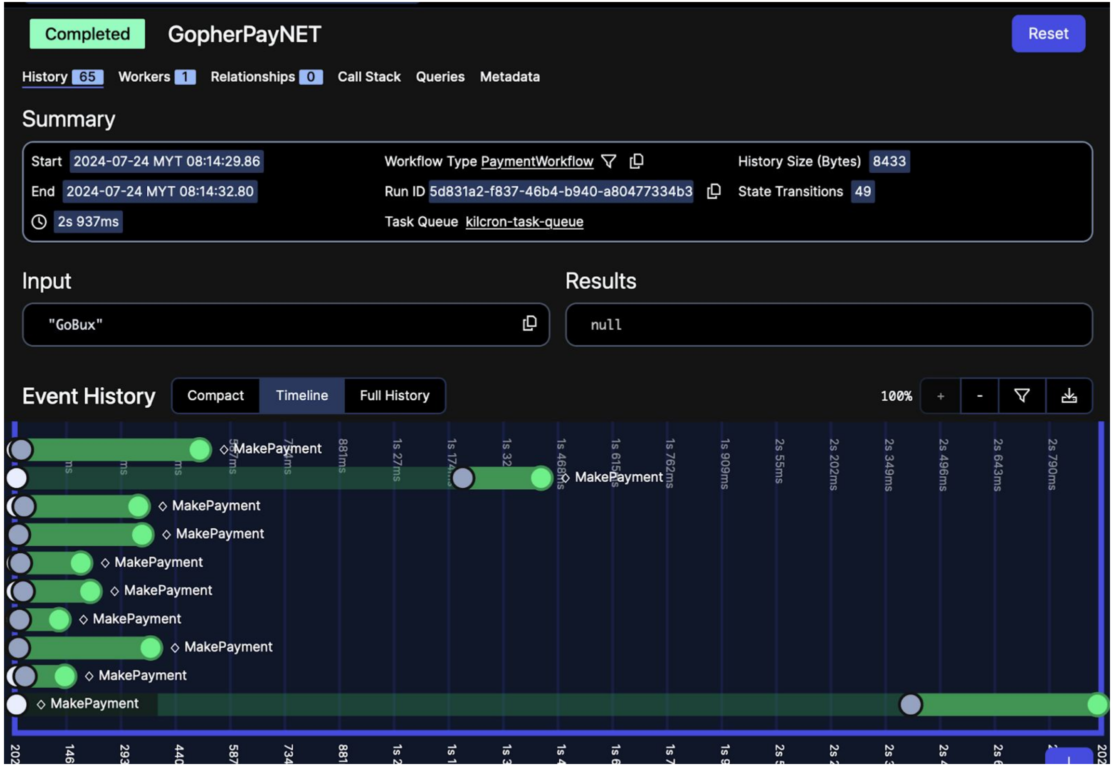
Results

```
null
```

Event History

Cron - Closer to Reality

- Cron jobs have variable latency; no failure!



Completed GopherPayNET

Reset

History 65 Workers 1 Relationships 0 Call Stack Queries Metadata

Summary

Start 2024-07-24 MYT 08:14:29.86

End 2024-07-24 MYT 08:14:32.80

2s 937ms

Workflow Type PaymentWorkflow

Run ID 5d831a2-f837-46b4-b940-a80477334b3

Task Queue kilcron-task-queue

History Size (Bytes) 8433

State Transitions 49

Input

```
"GoBux"
```

Results

```
null
```

Event History

Compact Timeline Full History

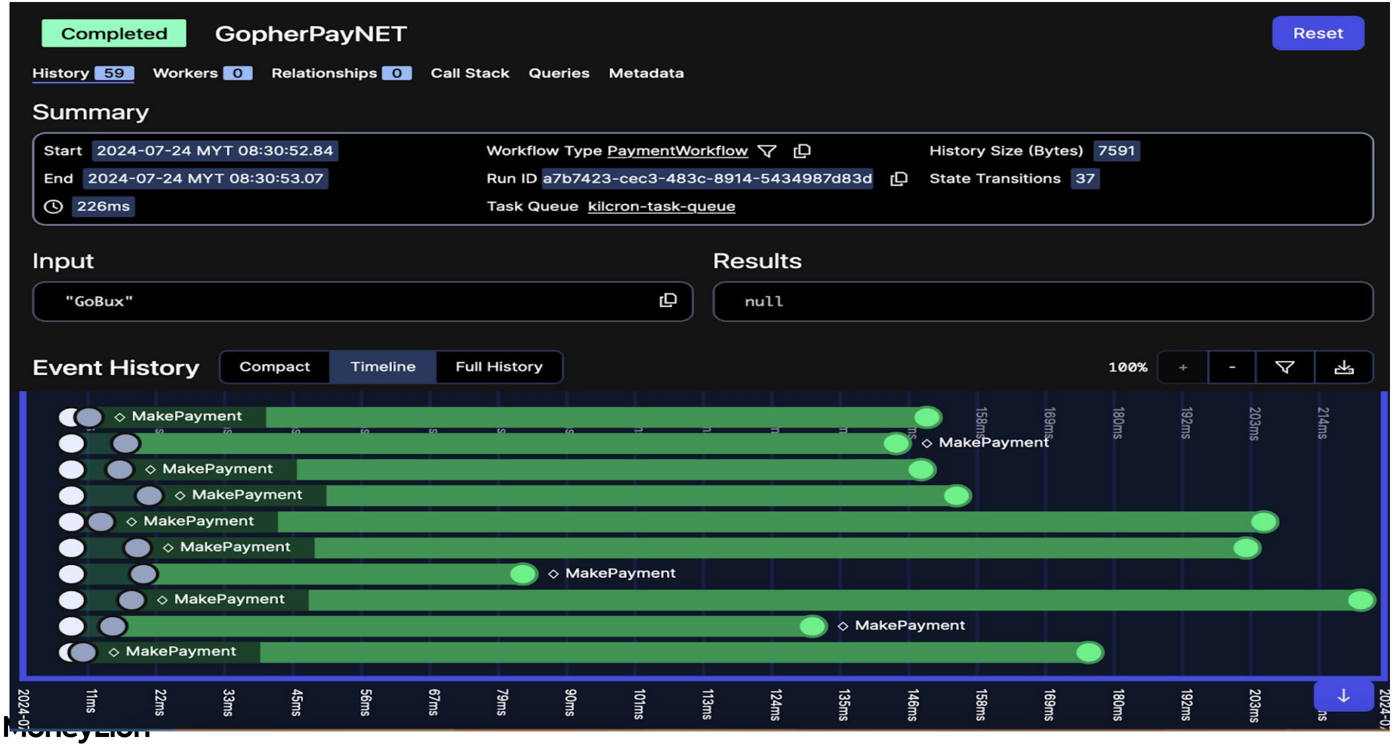
100%

KilCron
TEMPORAL
MODERN
ANTQURTED
CRON
CRON
CRON

KilCron
TEMPORAL
MODERN
ANTQURTED
CRON
CRON
CRON

Cron - Reality

- Use Temporal Schedules instead. Can start, pause or signal
- Rethink the whole flow; break it down to smaller parts (HOW?)



Completed GopherPayNET

Reset

History 59 Workers 0 Relationships 0 Call Stack Queries Metadata

Summary

Start 2024-07-24 MYT 08:30:52.84

End 2024-07-24 MYT 08:30:53.07

226ms

Workflow Type PaymentWorkflow

Run ID a7b7423-cec3-483c-8914-5434987d83d

Task Queue kilcron-task-queue

History Size (Bytes) 7591

State Transitions 37

Input

```
"GoBux"
```

Results

```
null
```

Event History

Compact Timeline Full History

100%

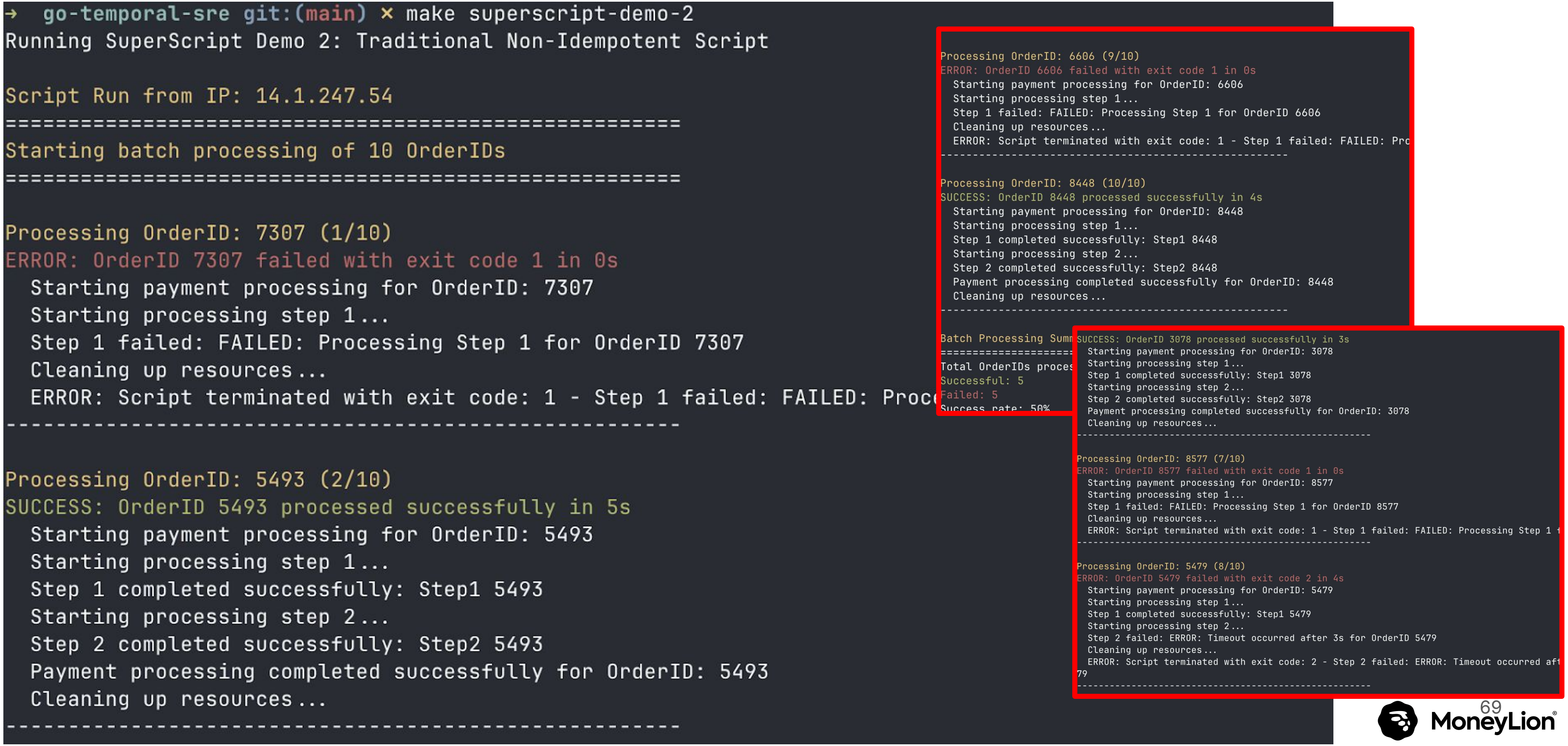
03
Granting Super Powers to your Humble Scripts

Real Life is Messy (as a SRE)

- Real life; unexpected events can happen! Not deterministic
- Bash or Python scripts used for automation are flaky
- Many dependencies out of control: DBs overloaded, network, vendors, cosmic-rays
- **Consequence:** Double billing of customers, Unnecessary cloud resources activated, Database upgrade left in unrecoverable state
- **Solution:** Idempotency allows safe retries. An operation that can be applied multiple times without changing the result
- Temporal to the rescue! (of course)

Traditional non-Idempotent

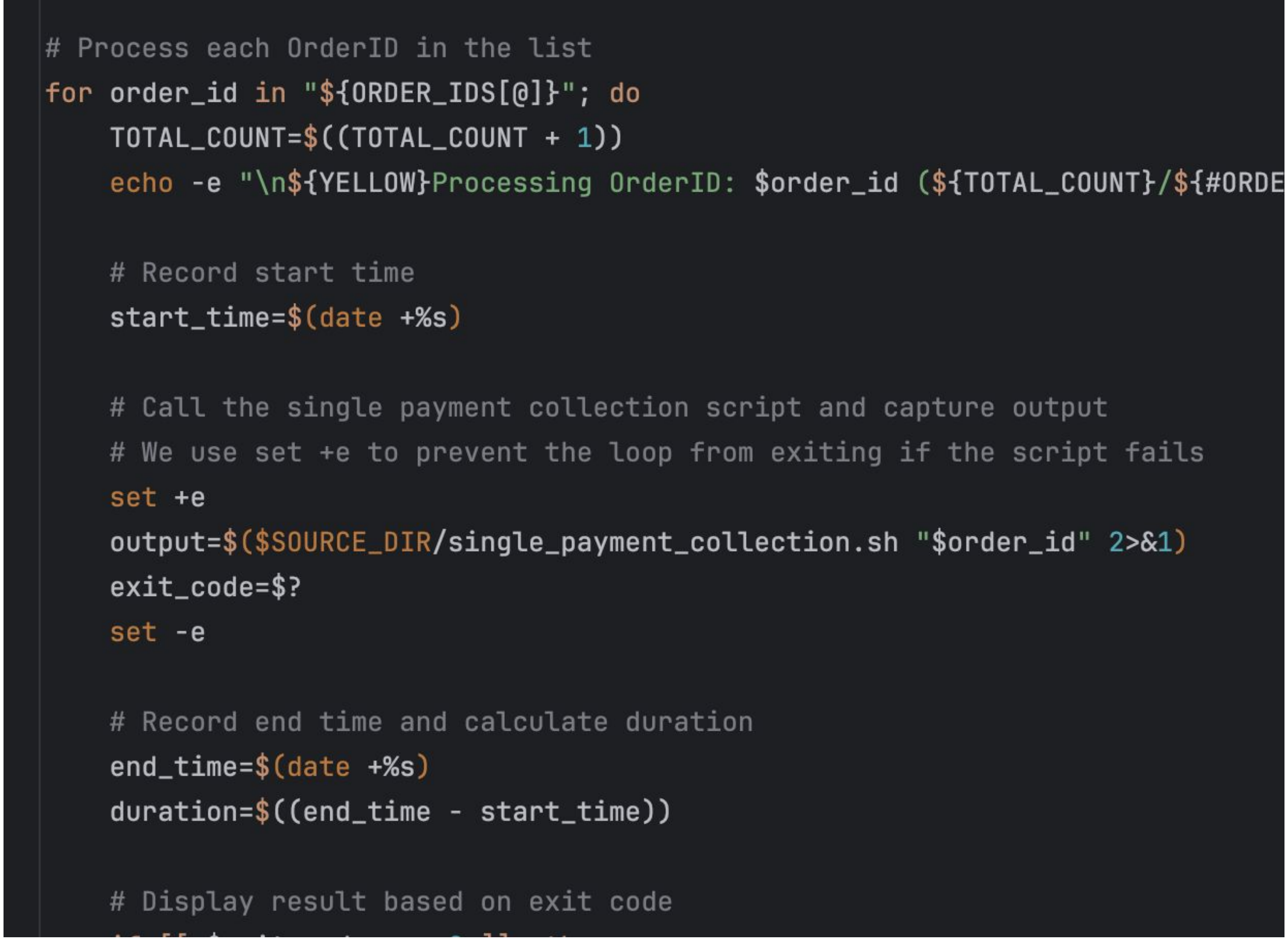
- Each time the script runs it is different! Not deterministic



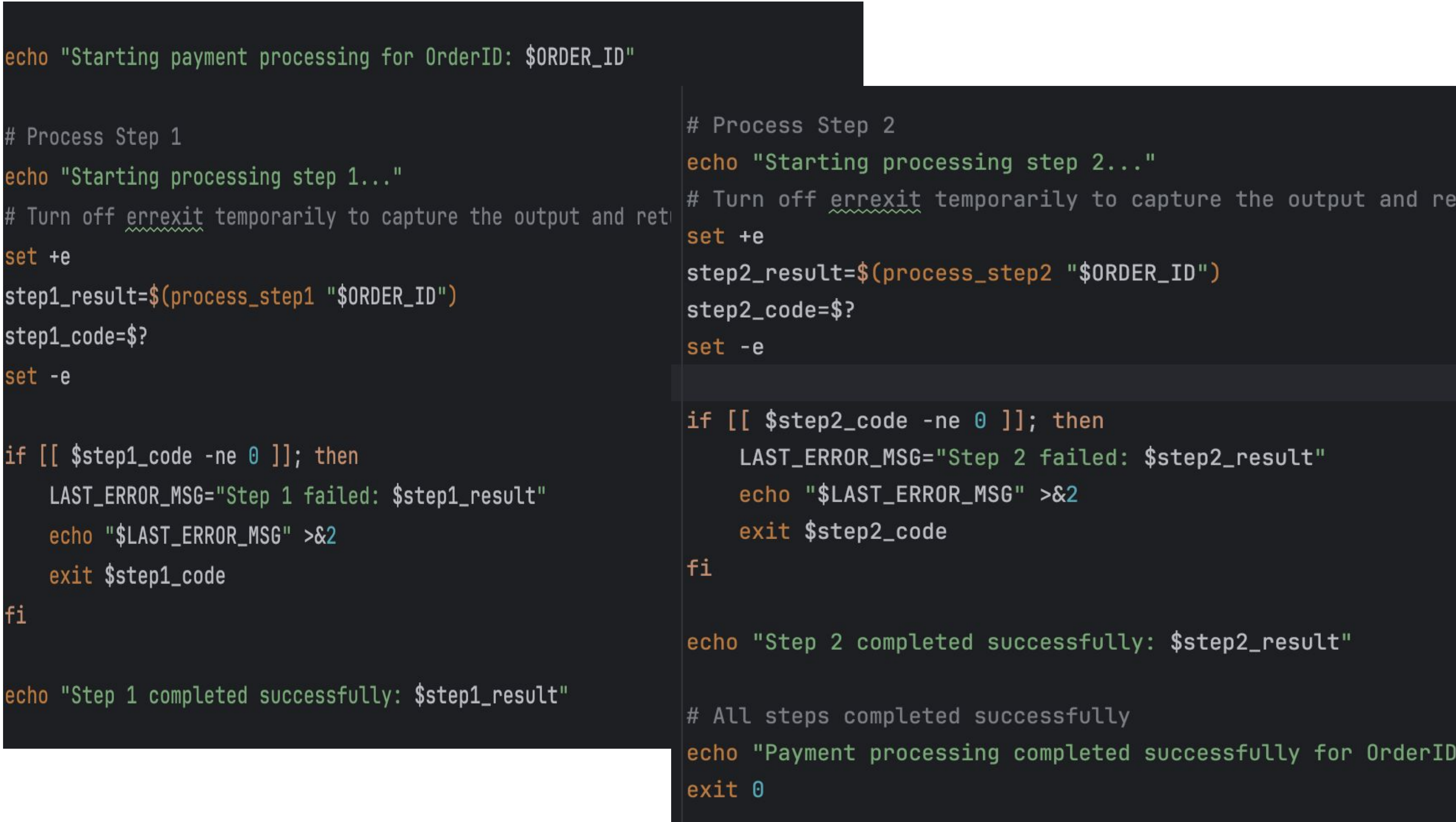
```
➜  go-temporal-sre git:(main) ✗ make superscript-demo-2
Running SuperScript Demo 2: Traditional Non-Idempotent Script

Script Run from IP: 14.1.247.54
======================================================
Starting batch processing of 10 OrderIDs
======================================================

Processing OrderID: 7307 (1/10)
ERROR: OrderID 7307 failed with exit code 1 in 0s
  Starting payment processing for OrderID: 7307
  Starting processing step 1...
  Step 1 failed: FAILED: Processing Step 1 for OrderID 7307
  Cleaning up resources...
  ERROR: Script terminated with exit code: 1 - Step 1 failed: FAILED: Proce
------------------------------------------------------

Processing OrderID: 5493 (2/10)
SUCCESS: OrderID 5493 processed successfully in 5s
  Starting payment processing for OrderID: 5493
  Starting processing step 1...
  Step 1 completed successfully: Step1 5493
  Starting processing step 2...
  Step 2 completed successfully: Step2 5493
  Payment processing completed successfully for OrderID: 5493
  Cleaning up resources...
------------------------------------------------------
```

```
Processing OrderID: 6606 (9/10)
ERROR: OrderID 6606 failed with exit code 1 in 0s
  Starting payment processing for OrderID: 6606
  Starting processing step 1...
  Step 1 failed: FAILED: Processing Step 1 for OrderID 6606
  Cleaning up resources...
  ERROR: Script terminated with exit code: 1 - Step 1 failed: FAILED: Pro
------------------------------------------------------

Processing OrderID: 8448 (10/10)
SUCCESS: OrderID 8448 processed successfully in 4s
  Starting payment processing for OrderID: 8448
  Starting processing step 1...
  Step 1 completed successfully: Step1 8448
  Starting processing step 2...
  Step 2 completed successfully: Step2 8448
  Payment processing completed successfully for OrderID: 8448
  Cleaning up resources...
------------------------------------------------------

Batch Processing Summ
=====================
Total OrderIDs proces
Successful: 5
Failed: 5
Success rate: 50%
```

```
SUCCESS: OrderID 3078 processed successfully in 3s
  Starting payment processing for OrderID: 3078
  Starting processing step 1...
  Step 1 completed successfully: Step1 3078
  Starting processing step 2...
  Step 2 completed successfully: Step2 3078
  Payment processing completed successfully for OrderID: 3078
  Cleaning up resources...
------------------------------------------------------

Processing OrderID: 8577 (7/10)
ERROR: OrderID 8577 failed with exit code 1 in 0s
  Starting payment processing for OrderID: 8577
  Starting processing step 1...
  Step 1 failed: FAILED: Processing Step 1 for OrderID 8577
  Cleaning up resources...
  ERROR: Script terminated with exit code: 1 - Step 1 failed: FAILED: Processing Step 1 f
------------------------------------------------------

Processing OrderID: 5479 (8/10)
ERROR: OrderID 5479 failed with exit code 2 in 4s
  Starting payment processing for OrderID: 5479
  Starting processing step 1...
  Step 1 completed successfully: Step1 5479
  Starting processing step 2...
  Step 2 failed: ERROR: Timeout occurred after 3s for OrderID 5479
  Cleaning up resources...
  ERROR: Script terminated with exit code: 2 - Step 2 failed: ERROR: Timeout occurred aft
79
------------------------------------------------------
```



```
# Process each OrderID in the list
for order_id in "${ORDER_IDS[@]}"; do
    TOTAL_COUNT=$((TOTAL_COUNT + 1))
    echo -e "\n${YELLOW}Processing OrderID: $order_id (${TOTAL_COUNT}/${#ORDE

    # Record start time
    start_time=$(date +%s)

    # Call the single payment collection script and capture output
    # We use set +e to prevent the loop from exiting if the script fails
    set +e
    output=$($SOURCE_DIR/single_payment_collection.sh "$order_id" 2>&1)
    exit_code=$?
    set -e

    # Record end time and calculate duration
    end_time=$(date +%s)
    duration=$((end_time - start_time))

    # Display result based on exit code
```

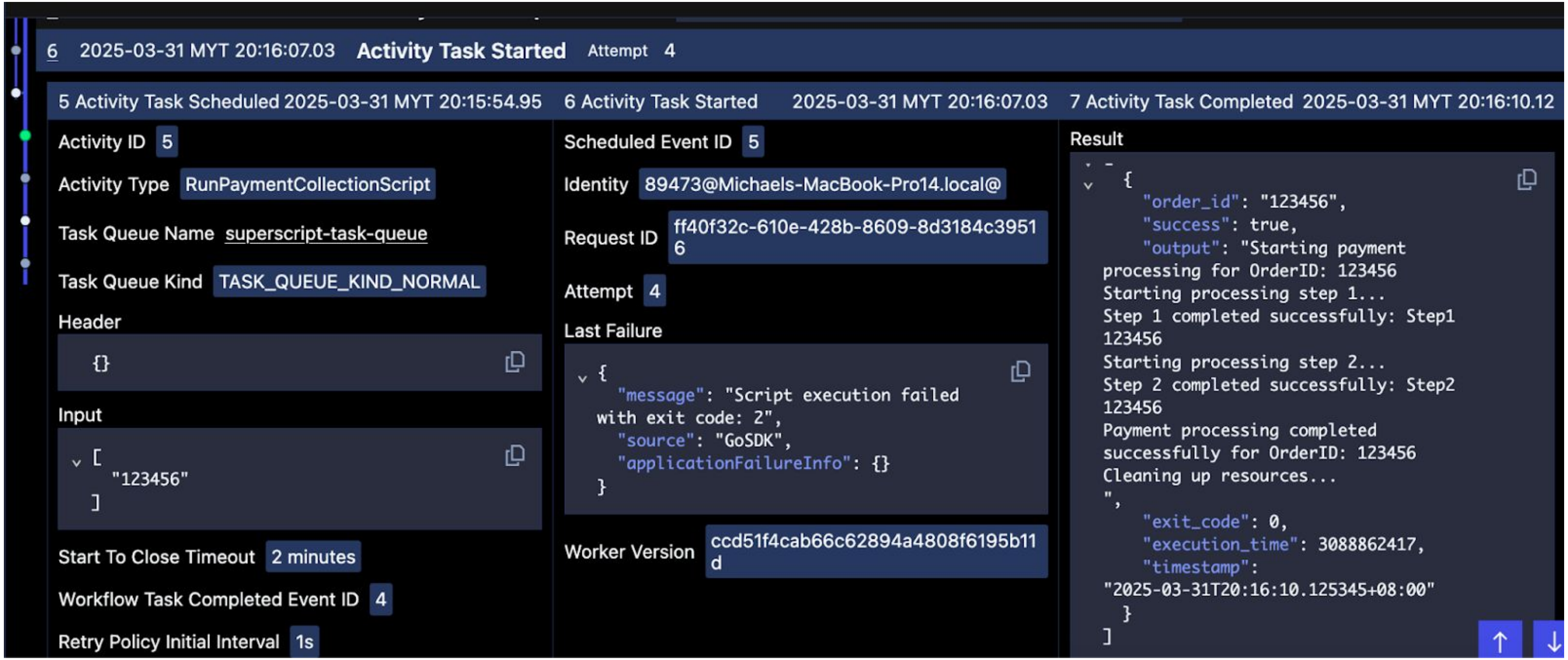


```bash
echo "Starting payment processing for OrderID: $ORDER_ID"

# Process Step 1
echo "Starting processing step 1..."
# Turn off errexit temporarily to capture the output and ret
set +e
step1_result=$(process_step1 "$ORDER_ID")
step1_code=$?
set -e

if [[ $step1_code -ne 0 ]]; then
    LAST_ERROR_MSG="Step 1 failed: $step1_result"
    echo "$LAST_ERROR_MSG" >&2
    exit $step1_code
fi

echo "Step 1 completed successfully: $step1_result"
```

```bash
# Process Step 2
echo "Starting processing step 2..."
# Turn off errexit temporarily to capture the output and re
set +e
step2_result=$(process_step2 "$ORDER_ID")
step2_code=$?
set -e

if [[ $step2_code -ne 0 ]]; then
    LAST_ERROR_MSG="Step 2 failed: $step2_result"
    echo "$LAST_ERROR_MSG" >&2
    exit $step2_code
fi

echo "Step 2 completed successfully: $step2_result"

# All steps completed successfully
echo "Payment processing completed successfully for OrderID
exit 0
```

Single Workflow made Deterministic

- From chaos to order; now idempotent
- Ensure WorkflowID no reuse; retry for free



6 2025-03-31 MYT 20:16:07.03 **Activity Task Started** Attempt 4

5 Activity Task Scheduled 2025-03-31 MYT 20:15:54.95

Activity ID 5

Activity Type RunPaymentCollectionScript

Task Queue Name superscript-task-queue

Task Queue Kind TASK_QUEUE_KIND_NORMAL

Header

```
{}
```

Input

```
[
  "123456"
]
```

Start To Close Timeout 2 minutes

Workflow Task Completed Event ID 4

Retry Policy Initial Interval 1s

6 Activity Task Started 2025-03-31 MYT 20:16:07.03

Scheduled Event ID 5

Identity 89473@Michaels-MacBook-Pro14.local@

Request ID ff40f32c-610e-428b-8609-8d3184c3951
6

Attempt 4

Last Failure

```
{
  "message": "Script execution failed
with exit code: 2",
  "source": "GoSDK",
  "applicationFailureInfo": {}
}
```

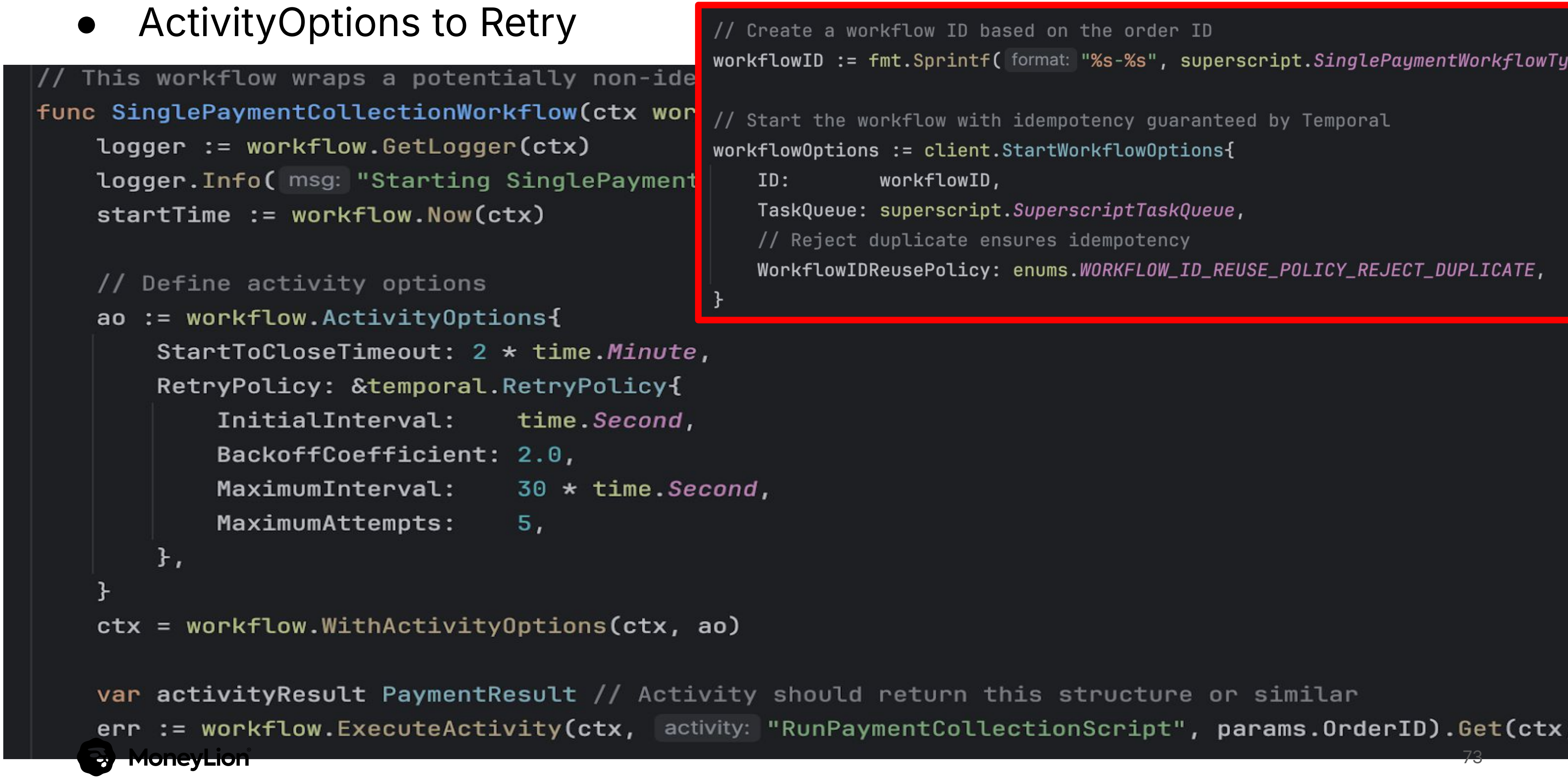
Worker Version ccd51f4cab66c62894a4808f6195b11
d

7 Activity Task Completed 2025-03-31 MYT 20:16:10.12

Result

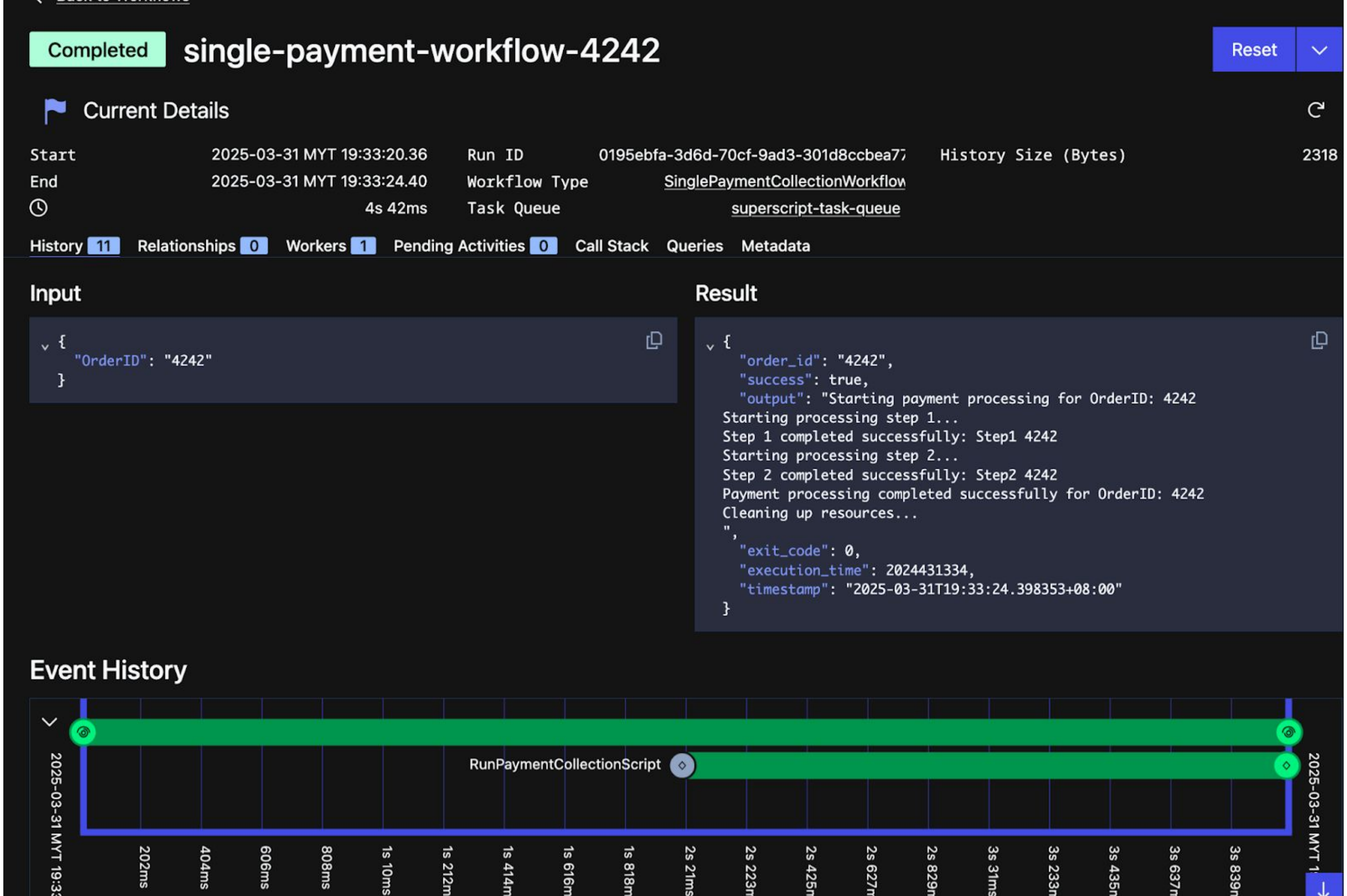
```
[
  {
    "order_id": "123456",
    "success": true,
    "output": "Starting payment
processing for OrderID: 123456
Starting processing step 1...
Step 1 completed successfully: Step1
123456
Starting processing step 2...
Step 2 completed successfully: Step2
123456
Payment processing completed
successfully for OrderID: 123456
Cleaning up resources...
",
    "exit_code": 0,
    "execution_time": 3088862417,
    "timestamp":
"2025-03-31T20:16:10.125345+08:00"
  }
]
```

- Reuse Policy: **WORKFLOW_ID_REUSE_POLICY_REJECT_DUPLICATE**
- ActivityOptions to Retry



```
// Create a workflow ID based on the order ID
workflowID := fmt.Sprintf( format: "%s-%s", superscript.SinglePaymentWorkflowTy

// Start the workflow with idempotency guaranteed by Temporal
workflowOptions := client.StartWorkflowOptions{
    ID:         workflowID,
    TaskQueue: superscript.SuperscriptTaskQueue,
    // Reject duplicate ensures idempotency
    WorkflowIDReusePolicy: enums.WORKFLOW_ID_REUSE_POLICY_REJECT_DUPLICATE,
}
```

```
// This workflow wraps a potentially non-ide
func SinglePaymentCollectionWorkflow(ctx wor
    logger := workflow.GetLogger(ctx)
    logger.Info( msg: "Starting SinglePayment
    startTime := workflow.Now(ctx)

    // Define activity options
    ao := workflow.ActivityOptions{
        StartToCloseTimeout: 2 * time.Minute,
        RetryPolicy: &temporal.RetryPolicy{
            InitialInterval:    time.Second,
            BackoffCoefficient: 2.0,
            MaximumInterval:    30 * time.Second,
            MaximumAttempts:    5,
        },
    }
    ctx = workflow.WithActivityOptions(ctx, ao)

    var activityResult PaymentResult // Activity should return this structure or similar
    err := workflow.ExecuteActivity(ctx, activity: "RunPaymentCollectionScript", params.OrderID).Get(ctx
```



single-payment-workflow-4242

Start 2025-03-31 MYT 19:33:20.36
End 2025-03-31 MYT 19:33:24.40
4s 42ms

Run ID 0195ebfa-3d6d-70cf-9ad3-301d8ccbea77
Workflow Type SinglePaymentCollectionWorkflow
Task Queue superscript-task-queue

History Size (Bytes) 2318

Input

```
{
  "OrderID": "4242"
}
```

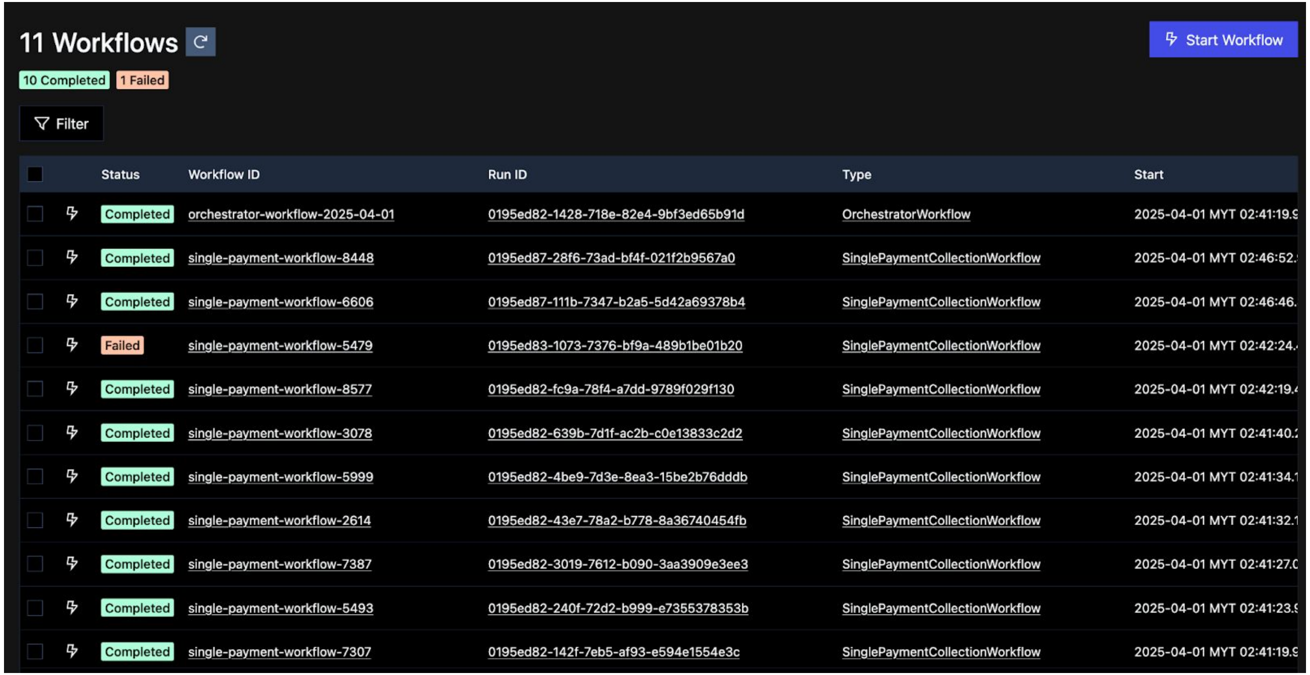
Result

```
{
  "order_id": "4242",
  "success": true,
  "output": "Starting payment processing for OrderID: 4242
Starting processing step 1...
Step 1 completed successfully: Step1 4242
Starting processing step 2...
Step 2 completed successfully: Step2 4242
Payment processing completed successfully for OrderID: 4242
Cleaning up resources...
",
  "exit_code": 0,
  "execution_time": 2024431334,
  "timestamp": "2025-03-31T19:33:24.398353+08:00"
}
```

Event History

Superscript Demo

- Real world is messy; but now under control - idempotent + auto-retry
- It may take time but run to completion successfully



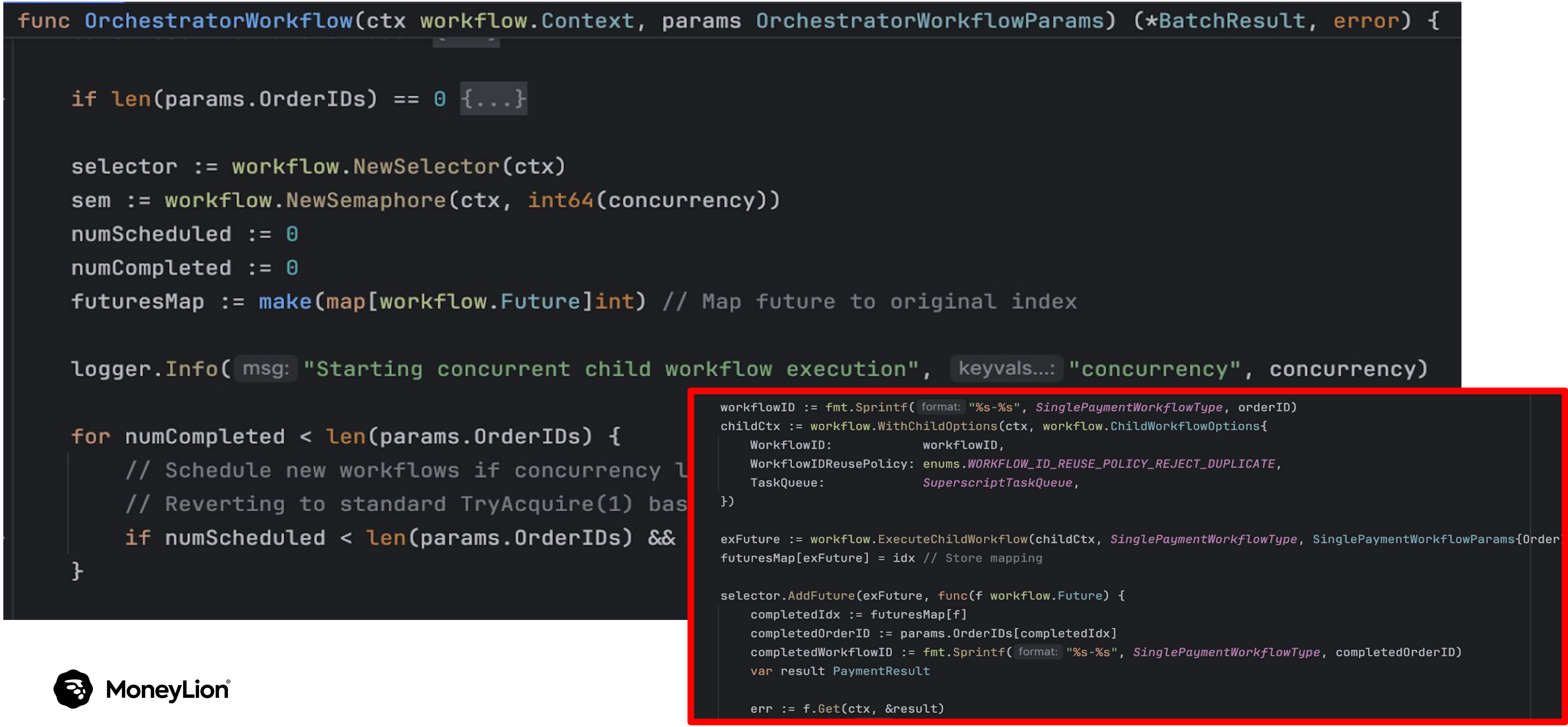
11 Workflows

10 Completed 1 Failed

□		Status	Workflow ID	Run ID	Type	Start
□	ϟ	Completed	orchestrator-workflow-2025-04-01	0195ed82-1428-718e-82e4-9bf3ed65b91d	OrchestratorWorkflow	2025-04-01 MYT 02:41:19.9
□	ϟ	Completed	single-payment-workflow-8448	0195ed87-28f6-73ad-bf4f-021f2b9567a0	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:46:52.
□	ϟ	Completed	single-payment-workflow-6606	0195ed87-111b-7347-b2a5-5d42a69378b4	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:46:46.
□	ϟ	Failed	single-payment-workflow-5479	0195ed83-1073-7376-bf9a-489b1be01b20	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:42:24.
□	ϟ	Completed	single-payment-workflow-8577	0195ed82-fc9a-78f4-a7dd-9789f029f130	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:42:19.
□	ϟ	Completed	single-payment-workflow-3078	0195ed82-639b-7d1f-ac2b-c0e13833c2d2	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:41:40.
□	ϟ	Completed	single-payment-workflow-5999	0195ed82-4be9-7d3e-8ea3-15be2b76dddb	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:41:34.1
□	ϟ	Completed	single-payment-workflow-2614	0195ed82-43e7-78a2-b778-8a36740454fb	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:41:32.1
□	ϟ	Completed	single-payment-workflow-7387	0195ed82-3019-7612-b090-3aa3909e3ee3	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:41:27.0
□	ϟ	Completed	single-payment-workflow-5493	0195ed82-240f-72d2-b999-e7355378353b	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:41:23.
□	ϟ	Completed	single-payment-workflow-7307	0195ed82-142f-7eb5-af93-e594e1554e3c	SinglePaymentCollectionWorkflow	2025-04-01 MYT 02:41:19.9

Superscript - Code / Flow

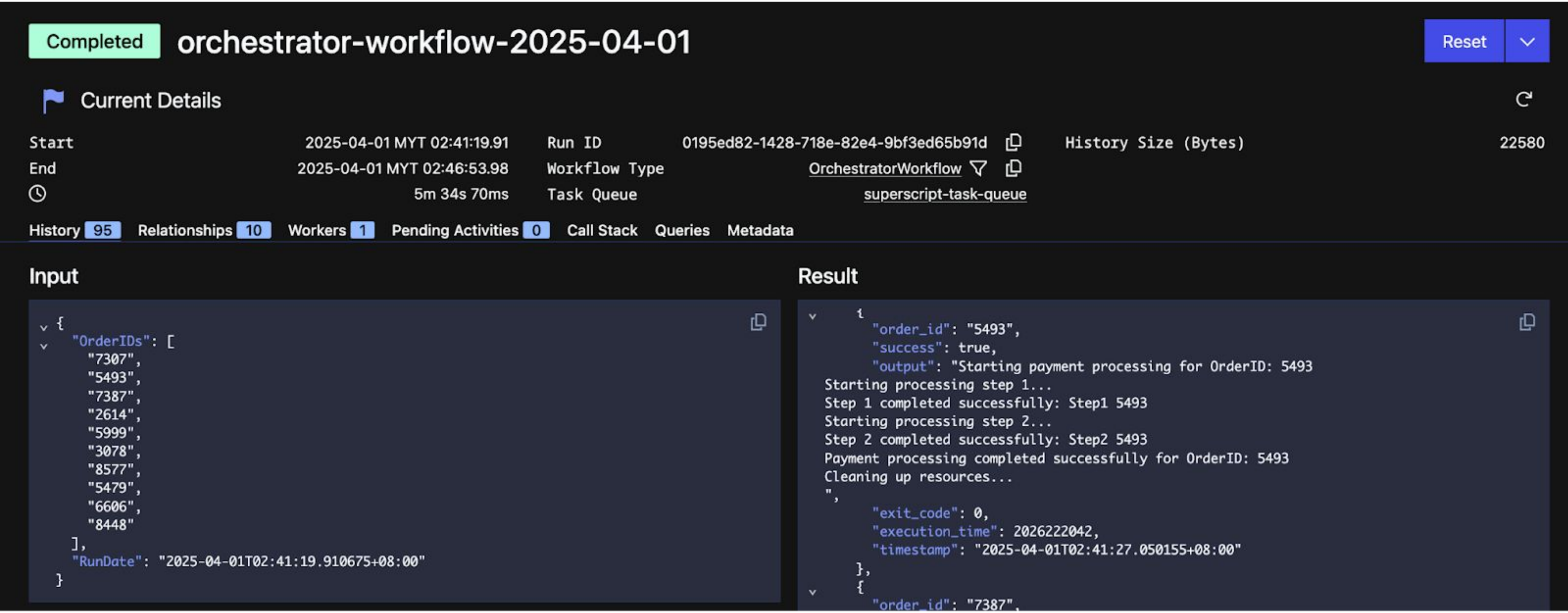
- Straight-forward, composable; calls earlier SinglePaymentWorkflow



```go
func OrchestratorWorkflow(ctx workflow.Context, params OrchestratorWorkflowParams) (*BatchResult, error) {

    if len(params.OrderIDs) == 0 {...}

    selector := workflow.NewSelector(ctx)
    sem := workflow.NewSemaphore(ctx, int64(concurrency))
    numScheduled := 0
    numCompleted := 0
    futuresMap := make(map[workflow.Future]int) // Map future to original index

    logger.Info( msg: "Starting concurrent child workflow execution", keyvals...: "concurrency", concurrency)

    for numCompleted < len(params.OrderIDs) {
        // Schedule new workflows if concurrency l
        // Reverting to standard TryAcquire(1) bas
        if numScheduled < len(params.OrderIDs) &&
    }
```

```go
workflowID := fmt.Sprintf( format: "%s-%s", SinglePaymentWorkflowType, orderID)
childCtx := workflow.WithChildOptions(ctx, workflow.ChildWorkflowOptions{
    WorkflowID:            workflowID,
    WorkflowIDReusePolicy: enums.WORKFLOW_ID_REUSE_POLICY_REJECT_DUPLICATE,
    TaskQueue:             SuperscriptTaskQueue,
})

exFuture := workflow.ExecuteChildWorkflow(childCtx, SinglePaymentWorkflowType, SinglePaymentWorkflowParams{Order
futuresMap[exFuture] = idx // Store mapping

selector.AddFuture(exFuture, func(f workflow.Future) {
    completedIdx := futuresMap[f]
    completedOrderID := params.OrderIDs[completedIdx]
    completedWorkflowID := fmt.Sprintf( format: "%s-%s", SinglePaymentWorkflowType, completedOrderID)
    var result PaymentResult

    err := f.Get(ctx, &result)
```

MoneyLion®



Completed **orchestrator-workflow-2025-04-01**

Current Details

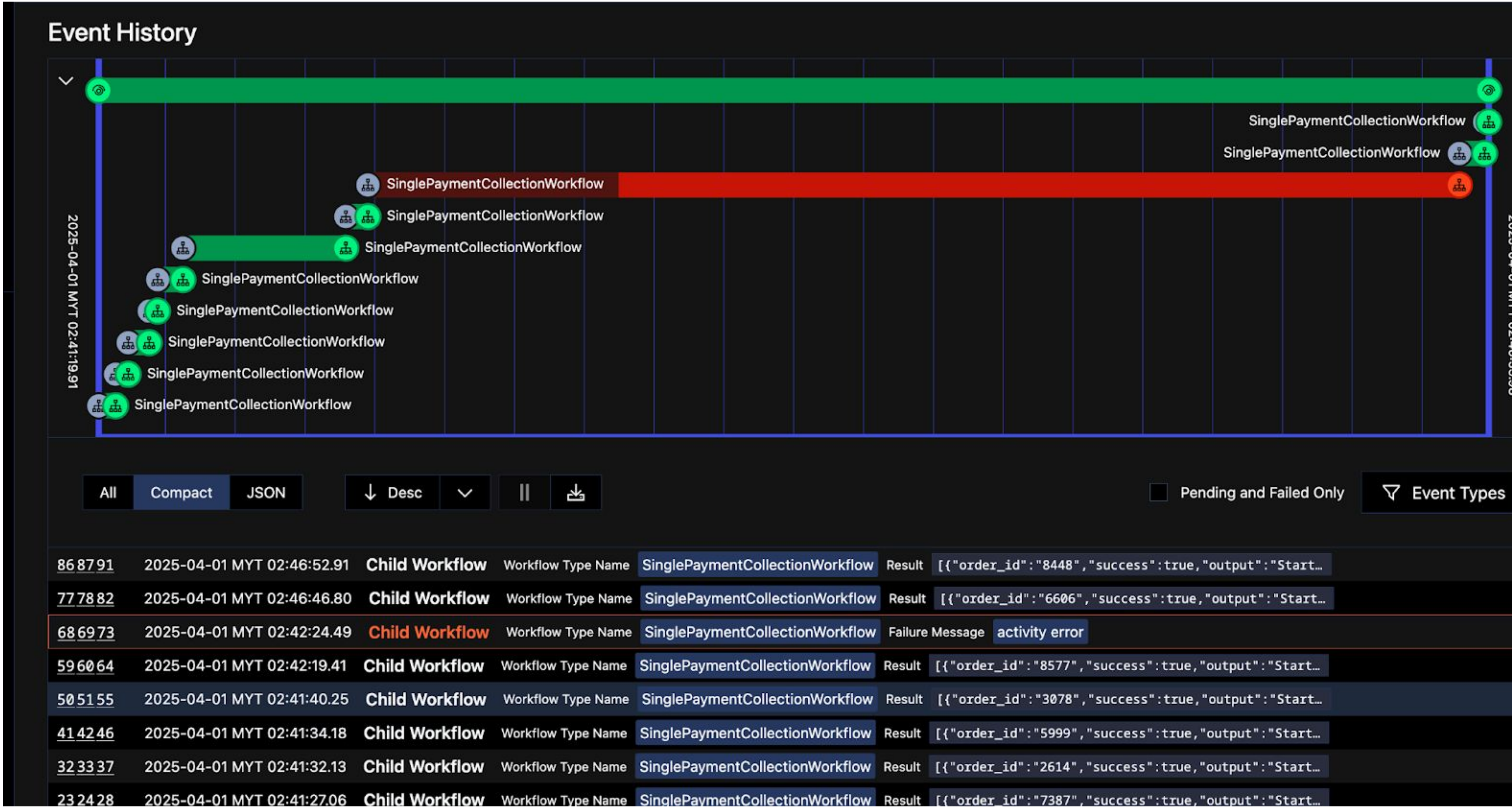
```
Start                2025-04-01 MYT 02:41:19.91    Run ID           0195ed82-1428-718e-82e4-9bf3ed65b91d    History Size (Bytes)    22580
End                  2025-04-01 MYT 02:46:53.98    Workflow Type    OrchestratorWorkflow
                     5m 34s 70ms                   Task Queue       superscript-task-queue
```

Input

```
{
  "OrderIDs": [
    "7307",
    "5493",
    "7387",
    "2614",
    "5999",
    "3078",
    "8577",
    "5479",
    "6606",
    "8448"
  ],
  "RunDate": "2025-04-01T02:41:19.910675+08:00"
}
```

Result

```
    {
      "order_id": "5493",
      "success": true,
      "output": "Starting payment processing for OrderID: 5493
Starting processing step 1...
Step 1 completed successfully: Step1 5493
Starting processing step 2...
Step 2 completed successfully: Step2 5493
Payment processing completed successfully for OrderID: 5493
Cleaning up resources...
",
      "exit_code": 0,
      "execution_time": 2026222042,
      "timestamp": "2025-04-01T02:41:27.050155+08:00"
    },
    {
      "order_id": "7387",
```



Event History
2025-04-01 MYT 02:41:19.91
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
SinglePaymentCollectionWorkflow
All
Compact
JSON
Desc
□ Pending and Failed Only
Event Types

86 87 91	2025-04-01 MYT 02:46:52.91	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"8448","success":true,"output":"Start…`
77 78 82	2025-04-01 MYT 02:46:46.80	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"6606","success":true,"output":"Start…`
68 69 73	2025-04-01 MYT 02:42:24.49	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Failure Message	activity error
59 60 64	2025-04-01 MYT 02:42:19.41	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"8577","success":true,"output":"Start…`
50 51 55	2025-04-01 MYT 02:41:40.25	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"3078","success":true,"output":"Start…`
41 42 46	2025-04-01 MYT 02:41:34.18	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"5999","success":true,"output":"Start…`
32 33 37	2025-04-01 MYT 02:41:32.13	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"2614","success":true,"output":"Start…`
23 24 28	2025-04-01 MYT 02:41:27.06	Child Workflow	Workflow Type Name	SinglePaymentCollectionWorkflow	Result	`[{"order_id":"7387","success":true,"output":"Start…`

04
Just In Time (JIT) Access + Demo

What is JIT?

Is it the same as break glass?

JIT vs Break glass

Use Cases of Just In Time(JIT)

Temporary AWS IAM Access

Gaining a temporary elevated role to perform a certain access on a Resource in AWS.

Temporary K8s Access

Temporary access to access k8s using IAM to perform elevated troubleshooting in the production environment cluster.

Temporary Access to approve Github Deployments

Temporary access to approve deployments when no one in the team is available to review and approve.

Temporary Database Access

Temporary access to a certain database (most likely production) to perform a certain change while being audited.

*Every JIT request must be audited & comply to the audit requirements.*

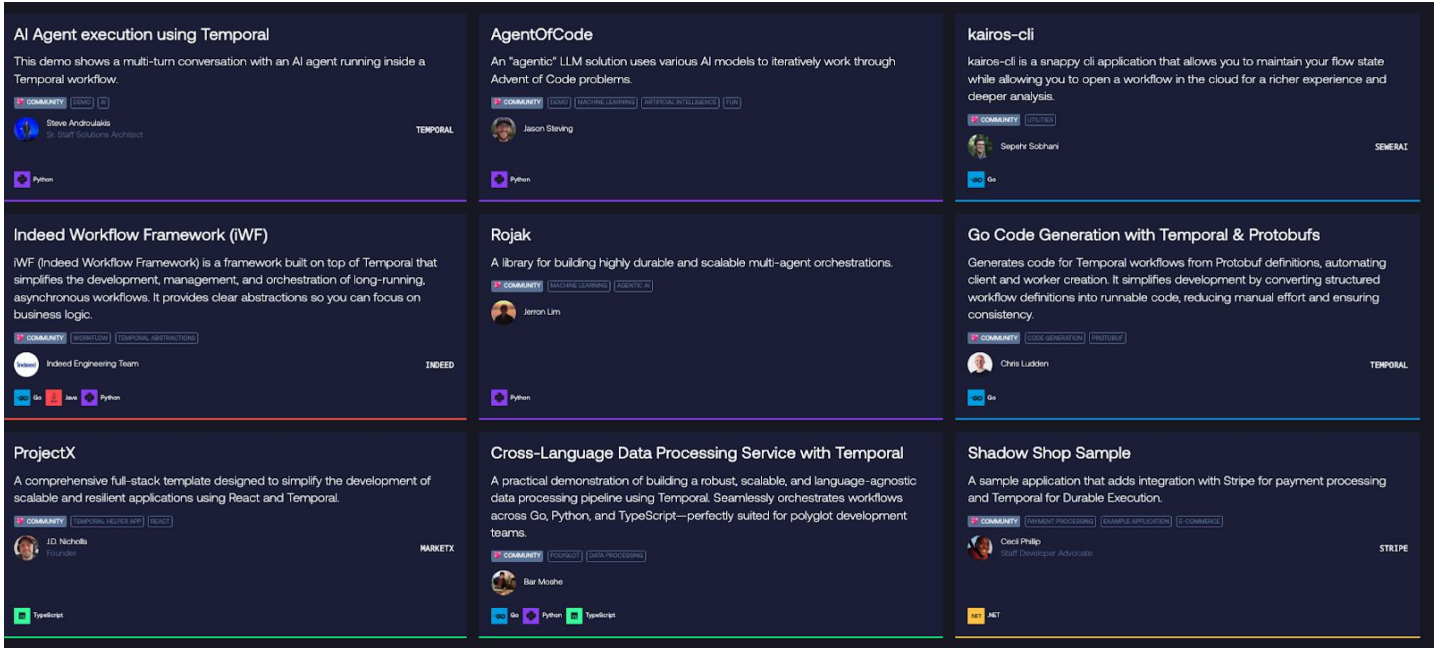
1. _Ticket Tracked_
2. _Required Approvers to approve requests_
3. _Audit trail_
4. _Access is automatically revoked after specific period of time._

DEMO

05
Temporal Universe Expanded

Introducing: Temporal Code Exchange

Marketplace of ideas to study + learn from. Open to submission!
https://temporal.io/code-exchange



AI Agent execution using Temporal

This demo shows a multi-turn conversation with an AI agent running inside a Temporal workflow.

COMMUNITY DEMO AI

Steve Androulakis
Sr. Staff Solutions Architect

TEMPORAL

Python

AgentOfCode

An "agentic" LLM solution uses various AI models to iteratively work through Advent of Code problems.

COMMUNITY DEMO MACHINE LEARNING ARTIFICIAL INTELLIGENCE FUN

Jason Steving

Python

kairos-cli

kairos-cli is a snappy cli application that allows you to maintain your flow state while allowing you to open a workflow in the cloud for a richer experience and deeper analysis.

COMMUNITY UTILITIES

Sepehr Sobhani

SEWERAI

Go

Indeed Workflow Framework (iWF)

iWF (Indeed Workflow Framework) is a framework built on top of Temporal that simplifies the development, management, and orchestration of long-running, asynchronous workflows. It provides clear abstractions so you can focus on business logic.

COMMUNITY WORKFLOW TEMPORAL ABSTRACTIONS

Indeed Engineering Team

INDEED

Go Java Python

Rojak

A library for building highly durable and scalable multi-agent orchestrations.

COMMUNITY MACHINE LEARNING AGENTIC AI

Jerron Lim

Python

Go Code Generation with Temporal & Protobufs

Generates code for Temporal workflows from Protobuf definitions, automating client and worker creation. It simplifies development by converting structured workflow definitions into runnable code, reducing manual effort and ensuring consistency.

COMMUNITY CODE GENERATION PROTOBUF

Chris Ludden

TEMPORAL

Go

ProjectX

A comprehensive full-stack template designed to simplify the development of scalable and resilient applications using React and Temporal.

COMMUNITY TEMPORAL HELPER APP REACT

J.D. Nicholls
Founder

MARKETX

TypeScript

Cross-Language Data Processing Service with Temporal

A practical demonstration of building a robust, scalable, and language-agnostic data processing pipeline using Temporal. Seamlessly orchestrates workflows across Go, Python, and TypeScript—perfectly suited for polyglot development teams.

COMMUNITY POLYGLOT DATA PROCESSING

Bar Moshe

Go Python TypeScript

Shadow Shop Sample

A sample application that adds integration with Stripe for payment processing and Temporal for Durable Execution.

COMMUNITY PAYMENT PROCESSING EXAMPLE APPLICATION E-COMMERCE

Cecil Phillip
Staff Developer Advocate

STRIPE

.NET

06
Alternatives to Temporal

Temporal Alternatives

Crowded market ... who wins? Who has the best DX?

Restate (Go,Java,Python,Typescript, Rust)

https://docs.restate.dev/

DBOS (Typescript, Python)

https://www.dbos.dev/

Cloudflare Workflows (Typescript)

https://cloudflare.com/

Golem Cloud (WASM)

https://www.golem.cloud/

Inngest (JS)

https://www.inngest.com/

Littlehorse (Go,Java,Python,Typescript,.NET)

https://littlehorse.io/

07
Q&A

Slide + Code: *https://github.com/cheelim1/go-temporal-sre*

Thank you

MoneyLion®