Pytech Summit
winter edition
All about Python 2024

Ruslan Korniichuk

Falling in Love with Temporal: Bulletproof Workflows

10 Grudnia 2024

FALCK

techevents

Falling in Love with Temporal: Bulletproof Workflows

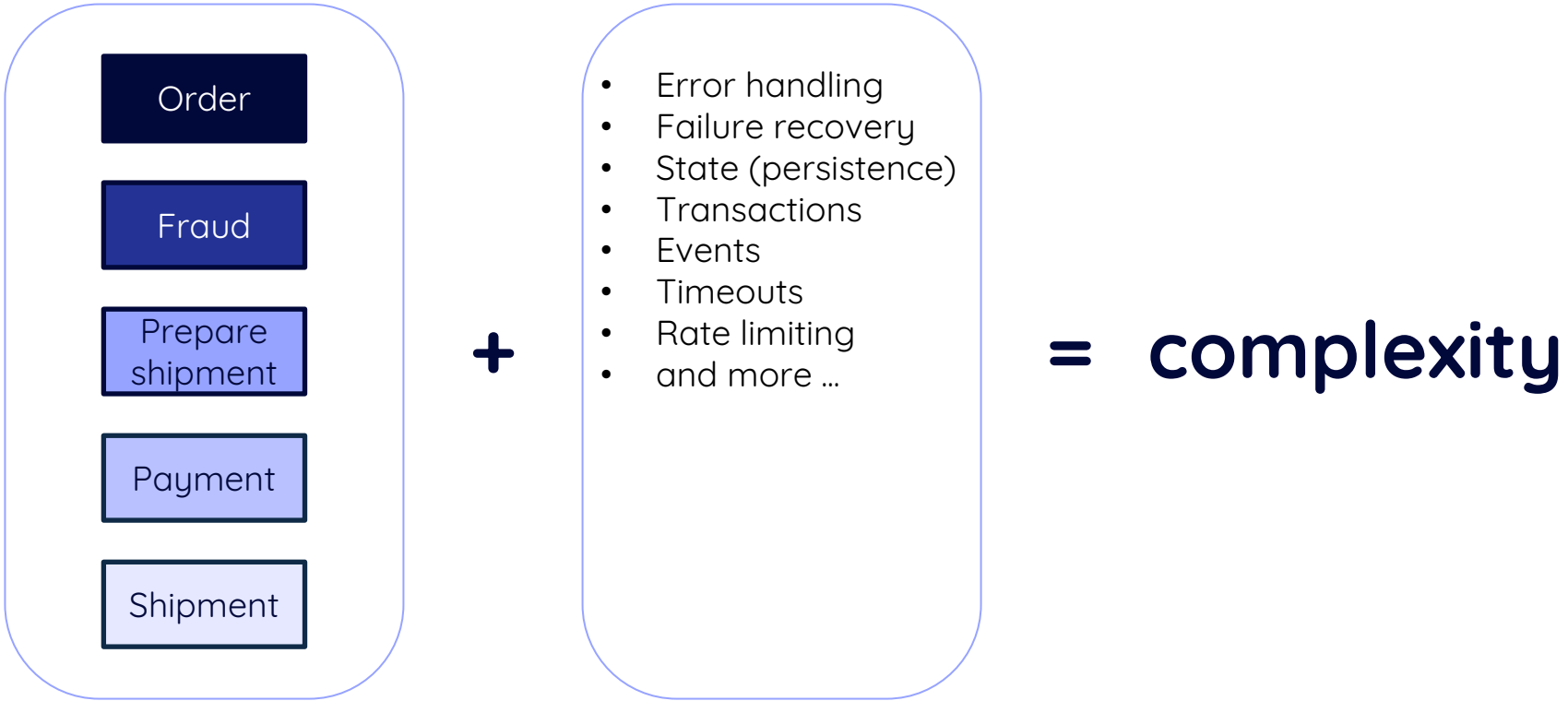
Ruslan Korniichuk | 10.12.2024

TABLE OF CONTENTS

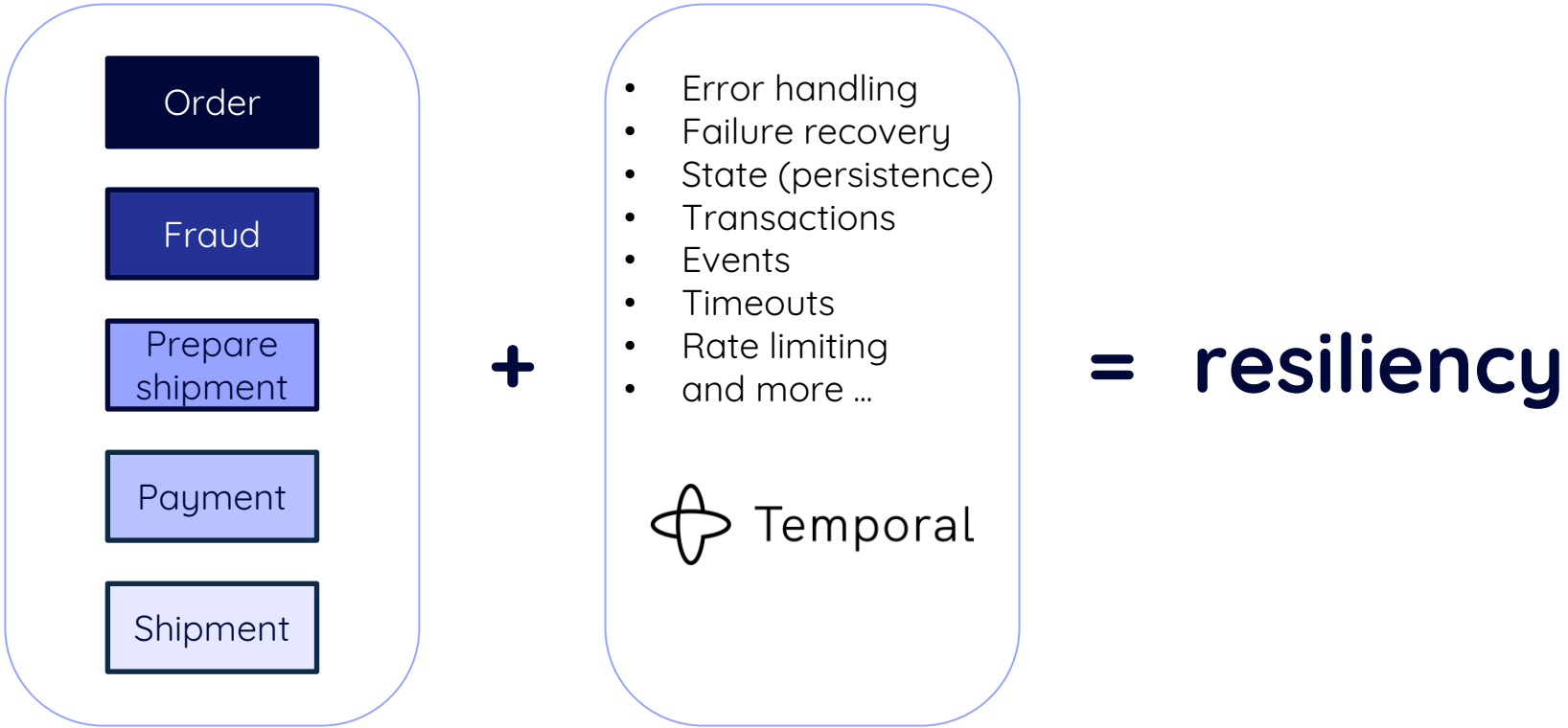
01

Problem statement

Problem statement



Problem statement



02

Introduction to Temporal

Introduction to Temporal

Temporal is an open source, distributed, and scalable workflow orchestration platform designed to execute mission-critical business logic with resilience.

Temporal:

- orchestration (not choreography)
- execution guarantee (aka Temporal Durable Execution)
- long-term workflows rather than real-time
- Python SDK
- used by Netflix, Airbnb, Stripe, Snap, HashiCorp, Box
- community

History

2004

Simple Queue Service (Amazon SQS)

2009

Simple Workflow Service (Amazon SWF)

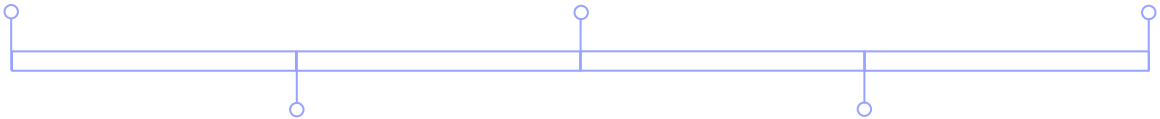
2015

Uber Cadence

Azure Durable Functions

2019

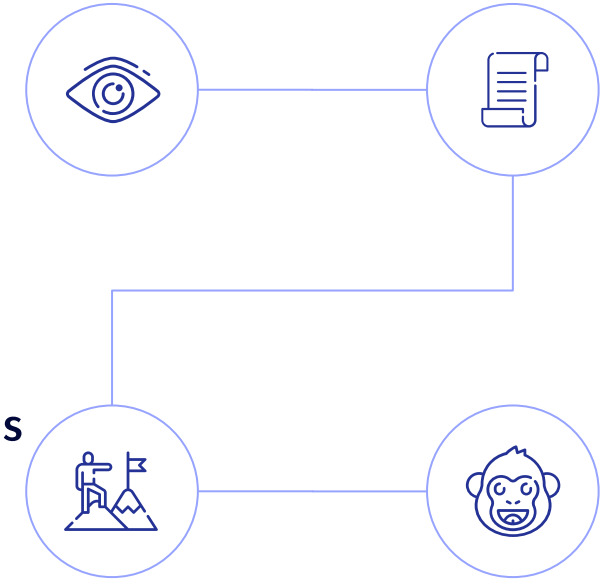
Temporal



Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code

- Windmill
- Pipedream
- Retool
- Superblocks

Version: 8.6

BPMN primer

Business Process Model and Notation 2.0 (BPMN) is an industry standard for process modeling and execution. A BPMN process is an XML document that has a visual representation. For example, here is a BPMN process:

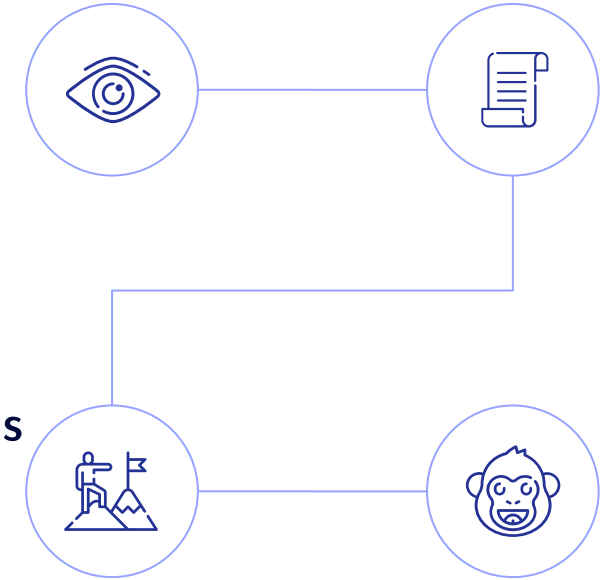
The corresponding XML

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:bpmndi="http:
  <bpmn:process id="Process_1" isExecutable="true">
    <bpmn:startEvent id="StartEvent_1" name="Order Placed">
      <bpmn:outgoing>SequenceFlow_1bq1azi</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:sequenceFlow id="SequenceFlow_1bq1azi" sourceRef="StartEvent_1" targetRef="Task_1f47
    <bpmn:sequenceFlow id="SequenceFlow_09hgjpg" sourceRef="Task_1f47b9v" targetRef="Task_1109
```

Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code

- Windmill
- Pipedream
- Retool
- Superblocks

README | Code of conduct | Apache-2.0 license

Hello World Application Using Conductor

In this section, we will create a simple "Hello World" application that executes a "greetings" workflow managed by Conductor.

Step 1: Create Workflow

Creating Workflows by Code

Create greetings_workflow.py with the following:

```python
from conductor.client.workflow.conductor_workflow import ConductorWorkflow
from conductor.client.workflow.executor.workflow_executor import WorkflowExecutor
from greetings_worker import greet

def greetings_workflow(workflow_executor: WorkflowExecutor) -> ConductorWorkflow:
    name = 'greetings'
    workflow = ConductorWorkflow(name=name, executor=workflow_executor)
    workflow.version = 1
    workflow >> greet(task_ref_name='greet_ref', name=workflow.input('name'))
    return workflow
```

(Alternatively) Creating Workflows in JSON

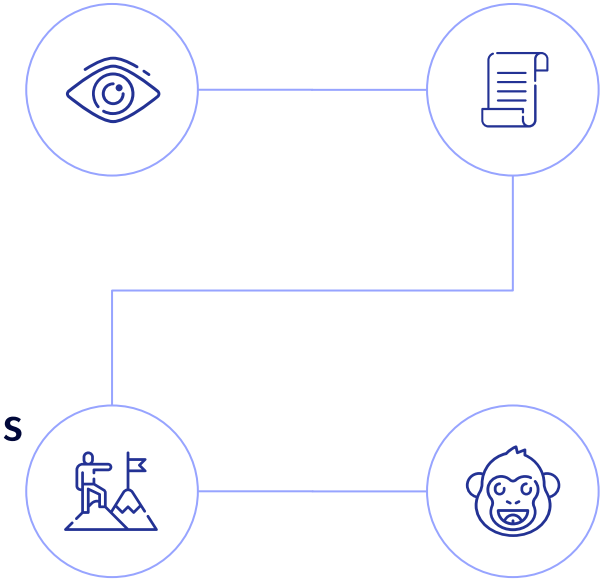
Create `greetings_workflow.json` with the following:

```json
{
  "name": "greetings",
  "description": "Sample greetings workflow",
  "version": 1,
  "tasks": [
    {
      "name": "greet",
      "taskReferenceName": "greet_ref",
      "type": "SIMPLE",
      "inputParameters": {
        "name": "${workflow.input.name}"
      }
    }
```

Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code

- Windmill
- Pipedream
- Retool
- Superblocks

Stars 11,136

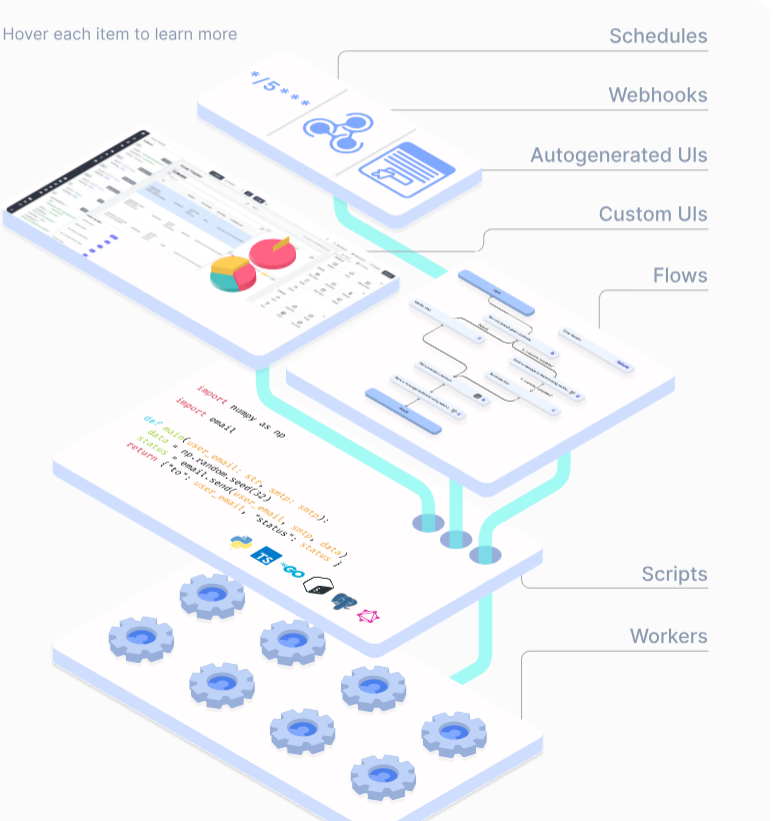
Open-source developer platform and workflow engine

Turn scripts into auto-generated UIs, APIs and cron jobs.
Compose them as workflows or data pipelines.
Build complex, data-intensive apps with ease.

Write and deploy software 10x faster, and run it with the highest reliability and observability on the fastest self-hostable job orchestrator.

Try Windmill cloud

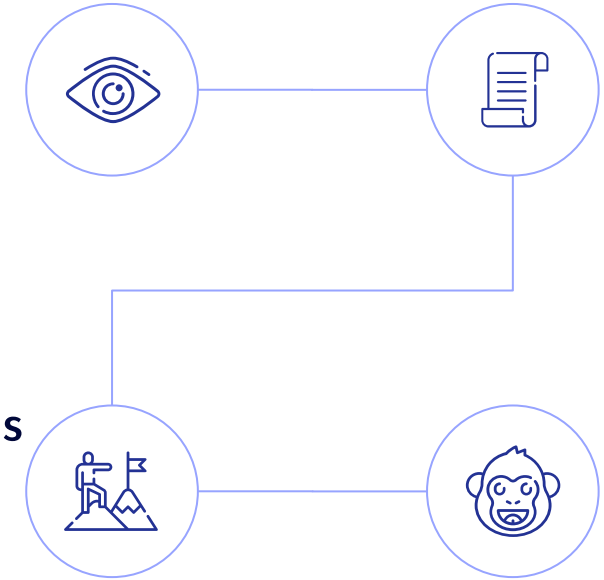
Self-host in 3 mins →



Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code

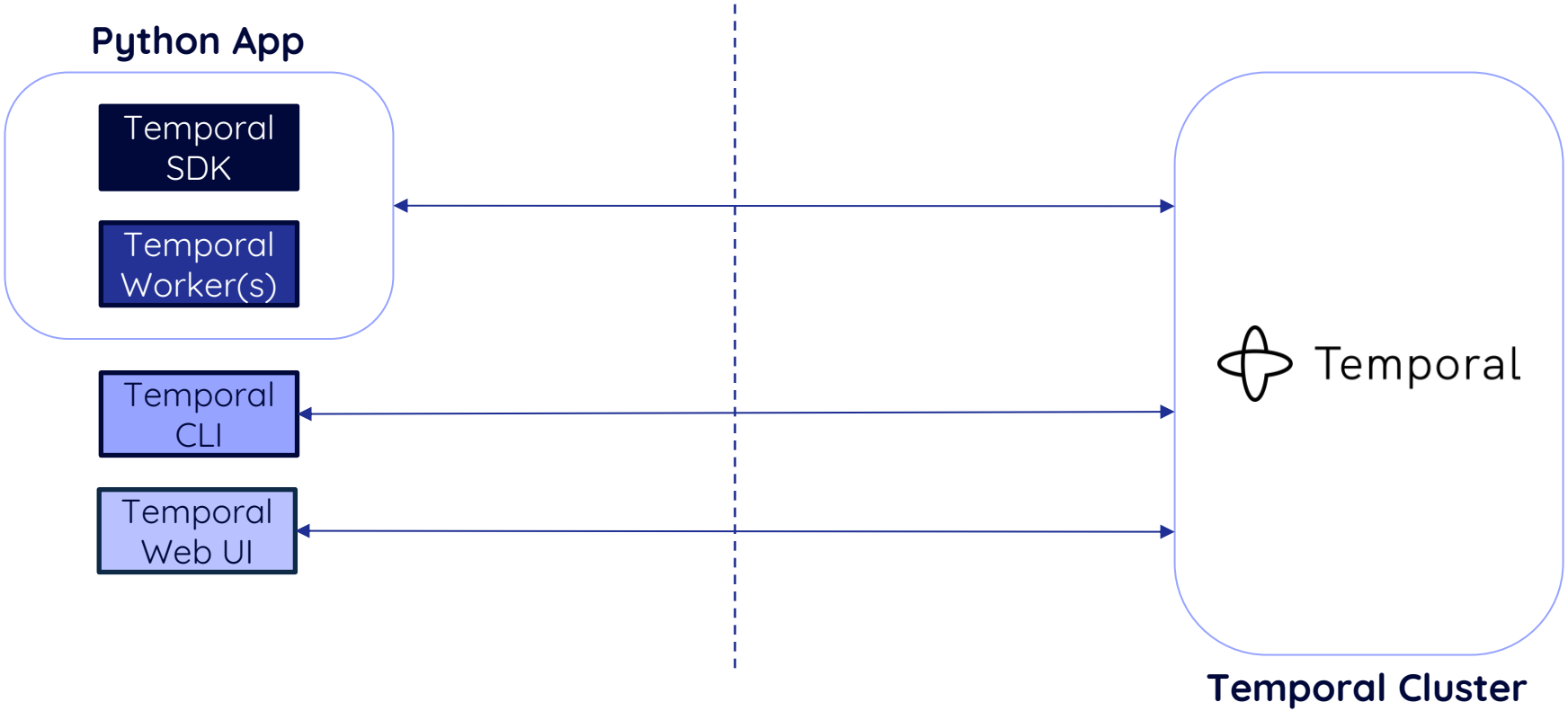
- Windmill
- Pipedream
- Retool
- Superblocks

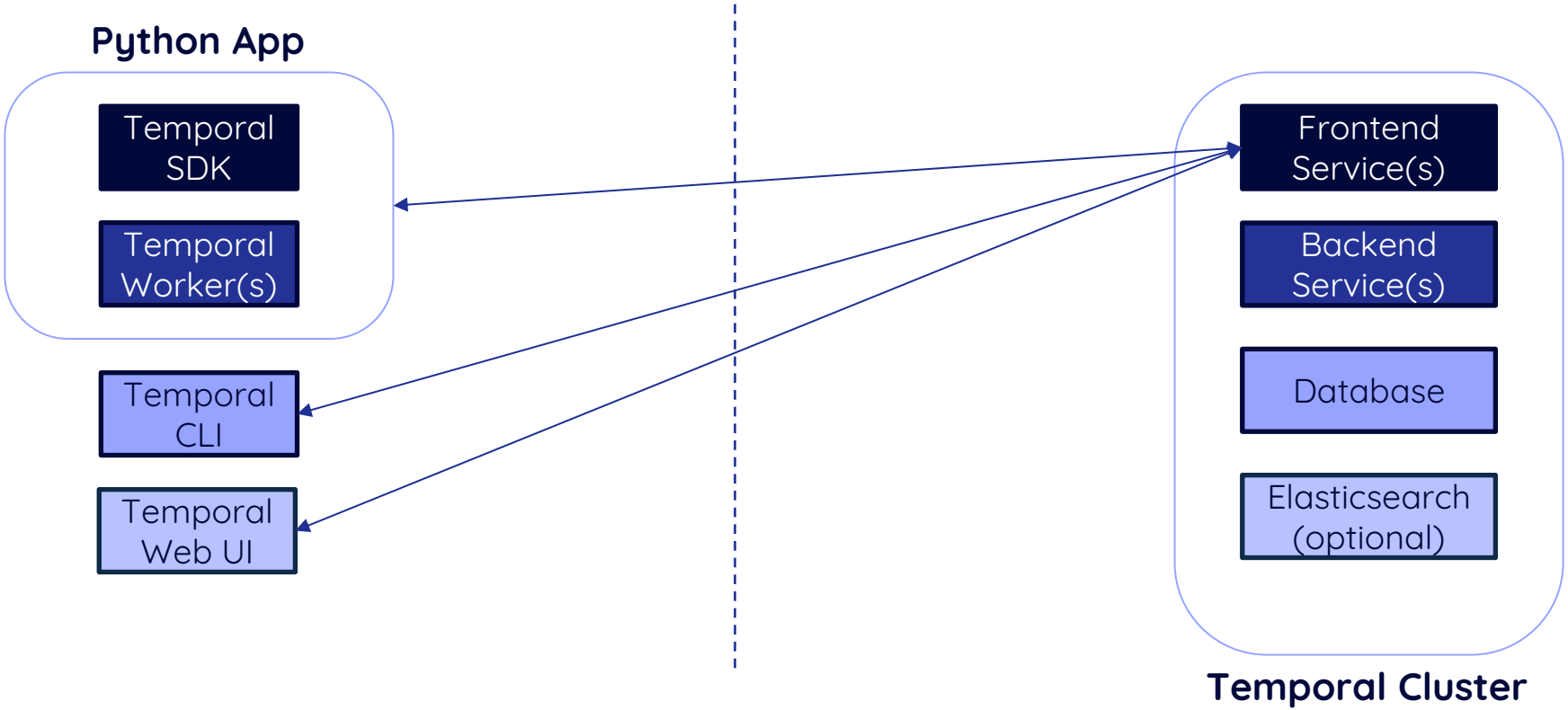
Temporal use cases

- Business process automation (e.g., client lifecycle, order processing, payments)
- Microservice orchestration
- Infrastructure provisioning
- CI/CD, data and ML pipelines
- Engine for low-code/no-code

03

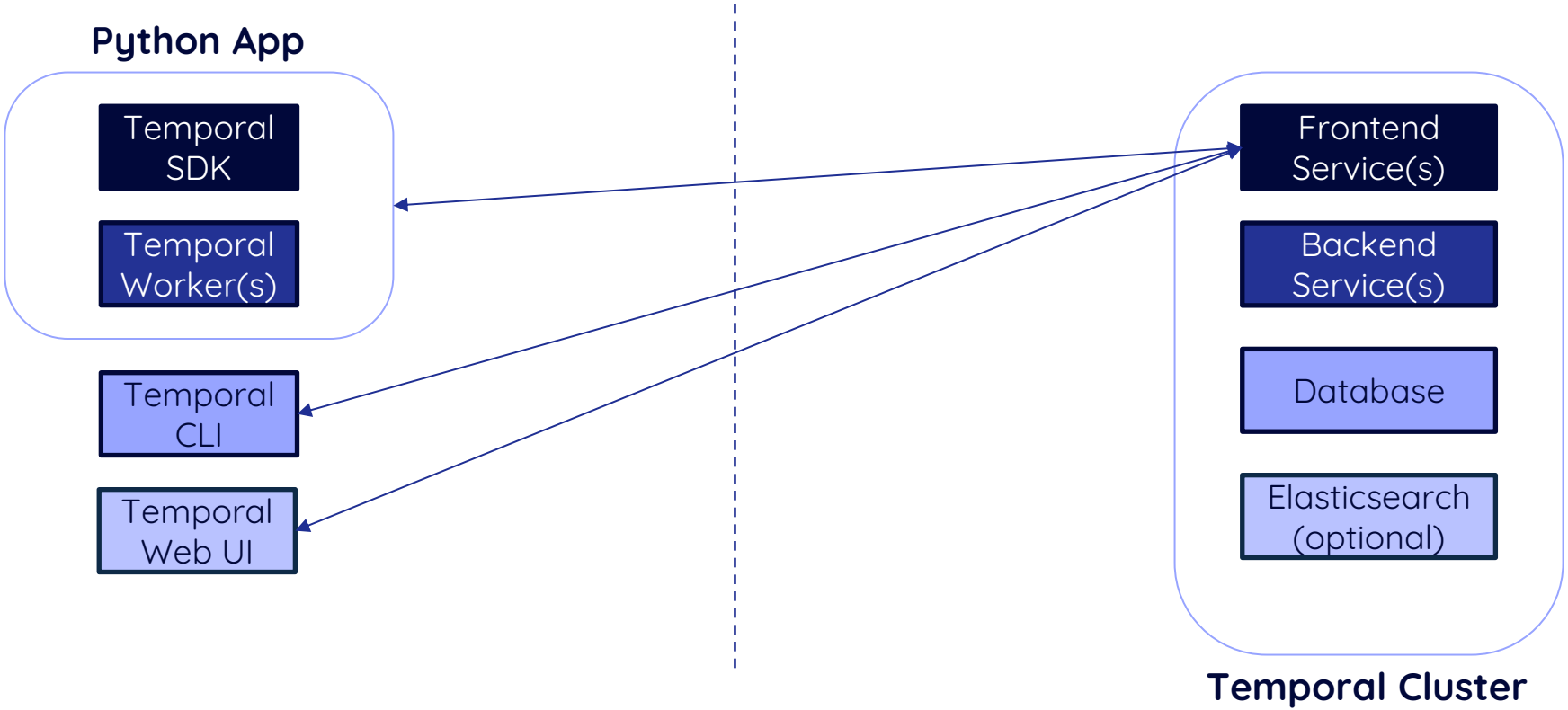
Temporal architecture


Python App
Temporal SDK
Temporal Worker(s)
Temporal CLI
Temporal Web UI
Temporal
Temporal Cluster


Python App
Temporal SDK
Temporal Worker(s)
Temporal CLI
Temporal Web UI
Frontend Service(s)
Backend Service(s)
Database
Elasticsearch (optional)
Temporal Cluster

The role of Temporal Cluster

- Manages and tracks tasks related to Temporal Workflows and Temporal Activities
- Maintains and durably persists the Event History for each execution
- Does NOT run Python app code
- Does NOT have access to Python app code


Python App
Temporal SDK
Temporal Worker(s)
Temporal CLI
Temporal Web UI
Frontend Service(s)
Backend Service(s)
Database
Elasticsearch (optional)
Temporal Cluster

The role of Temporal Worker

- Responsible for executing Temporal Workflow and Temporal Activity code
- Polls for tasks and communicates status via network with the Temporal Cluster
- Is provided by the Temporal Python SDK and configured by the engineer

One platform, many options for using it

- **Option 0:** Local Temporal Server
  - for dev purpose only
  - in-memory database
  - `$ temporal server start-dev`

- **Option 1:** Self-hosted Temporal Cluster
  - Cassandra, MySQL, PostgreSQL, SQLite
  - Docker and Docker Compose, or Kubernetes
  - Elasticsearch (optional)

- **Option 2:** Temporal Cloud
  - Fully-managed cloud service

04

Temporal Workflow and Activity

Temporal Workflow and Activity

Temporal Workflow vs Activity

Workflow

Workflow must be deterministic. You can view determinism as a requirement that each execution of a given Workflow must produce the same output, given the same input.

Workflow does not retry by default.

Workflow can start another Workflow.

Activity

Any operation that introduces the possibility of failure, such as calling microservice, querying database, writing to file, should be done as part of an **Activity**. Activity retries if they fail.

Activity is orchestrated by Workflow.

Activity must be idempotent.

Temporal Workflow determinism

```
def process_order(order):
    if random.choice([True, False]):
        check_fraud(...)
    ...
```

```
@workflow.run
async def process_order(self, order):
    if workflow.random().choice([True, False]):
        await workflow.execute_activity(check_fraud, ...)
    ...
```

Temporal Workflow determinism

```
def process_order(order):
    if order.time > datetime.now() - timedelta(days=1):
        check_fraud(...)
    ...
```

```
@workflow.run
async def process_order(self, order):
    if order.time > workflow.now() - timedelta(days=1):
        await workflow.execute_activity(check_fraud, ...)
    ...
```

Temporal Workflow vs Activity

Workflow

Workflow must be deterministic. You can view determinism as a requirement that each execution of a given Workflow must produce the same output, given the same input.

Workflow does not retry by default.

Workflow can start another Workflow.

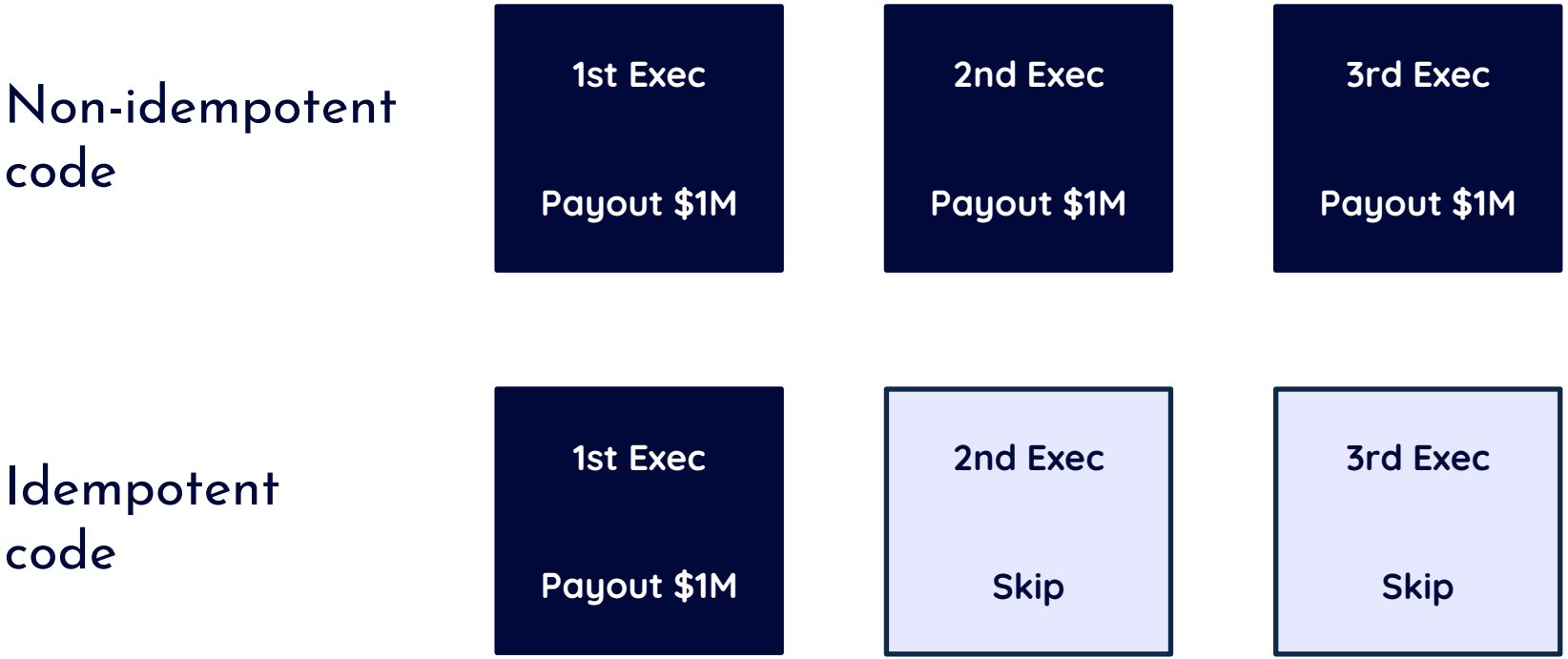
Activity

Any operation that introduces the possibility of failure, such as calling microservice, querying database, writing to file, should be done as part of an **Activity**. Activity retries if it fails.

Activity is orchestrated by Workflow.

Activity must be idempotent.

Code accidentally runs 3 times



Temporal 101 with Python

Last updated on ***Mar 11, 2024***

Tags: courses Python

Estimated time: ~ 2 hours, self-paced.

Cost: Free

Ask AI

Description

Temporal 102: Exploring Durable Execution with Python

Last updated on ***Mar 7, 2024***

Tags: courses Python

Estimated time: ~ 4 hours, self-paced.

Cost: Free

05

Temporal demo

NARODOWY BANK POLSKI

NBP Web API

wersja polska

Currency exchange rates and gold prices in the XML and JSON formats

The **api.nbp.pl** service operates a public Web API enabling HTTP clients to make enquiries on the following datasets published by the NBP.PL service:

1. current and historic exchange rates of foreign currencies:
    - table A of middle exchange rates of foreign currencies,
    - table B of middle exchange rates of foreign currencies,
    - table C of buy and sell prices of foreign currencies;
2. current and historic prices of gold calculated at NBP.

Communication with the service based on parametrized `HTTP GET` requests send to the **https://api.nbp.pl/api/** address.

User manual

Important API Update: Transition to HTTPS Only

temporal-pytech-summit-2024

Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file t Add file Code

korniichuk Update readme 05ee4bb · 10 minutes ago 8 Commits

img	Add Temporal Activity to Workflow	18 hours ago
.gitignore	Initial commit	yesterday
LICENSE	Initial commit	yesterday
README.md	Update readme	10 minutes ago
app.py	Add Temporal Activity to Workflow	18 hours ago
example_workflow.py	Add Retry Policy	15 hours ago
nbp.py	Fix get_usd_rate()	18 hours ago
requirements.txt	Add Temporal Activity to Workflow	18 hours ago
worker.py	Add Temporal Activity to Workflow	18 hours ago

README Unlicense license

Falling in Love with Temporal: Bulletproof Workflows

Step 1/4: Installation

About

Falling in Love with Temporal: Bulletproof Workflows

Readme
Unlicense license
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Suggested workflows

Based on your tech stack

RUSLAN KORNIICHUK

Software and Platform Engineer in Poland

Hire me

Solid experience of 9+ years in IT. Former Software Engineer and Architect at Fortune 500 companies. Core areas: Cloud Computing, and Software Engineering. Especially interested in Platform Engineering. Click the button above to hire me.

DONATE TO UKRAINE ARMY FUNDRAISER

Select the destination of your assistance

ARMY ASSISTANCE | SUPPORT THE FUND

All funds raised to help the army will be spent exclusively on purchases for the Armed Forces of Ukraine and other components of the Defense Forces.

Payment method

PAYMENT BY CARD | SWIFT TRANSFERS | CRYPTOCURRENCY

Regularity

ONE TIME | SUBSCRIPTION EVEN BETTER

Payment currency

UAH, ₴ | USD, $ | EUR, €

Amount of donation*

€ 10

Email*

example@example.com

☐ I consent to the processing of my email address to receive email newsletters (twice a month), invitations and receipts.

PAY

Payment secured

tinyurl.com/PYTECH2024

06

Q&A session