Orchestrating Mission-Critical Workflows with Temporal and Python

Ruslan Korniichuk

WARSAW IT DAYS

A lecture selected by a Program Council consisting of recognized leaders in the IT and Data Science field.

Warsaw,
19.03.2026 - 20.03.2026

OFFICIAL LECTURE OF THE WARSAW IT DAYS

Event organized by: Academic Partners Foundation

ANTONOV AN-124
Ruslan

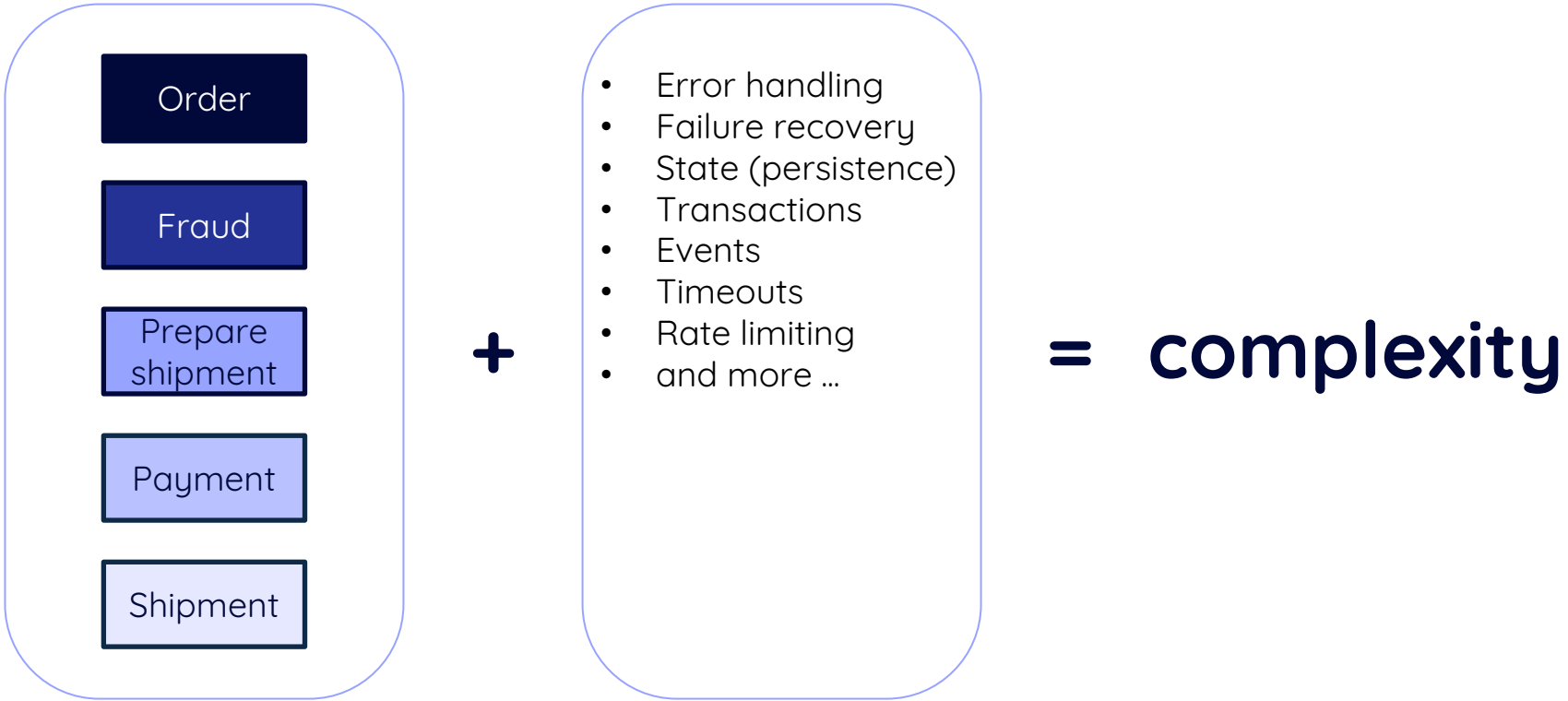
Source: https://tinyurl.com/y8pfmm67

TABLE OF CONTENTS

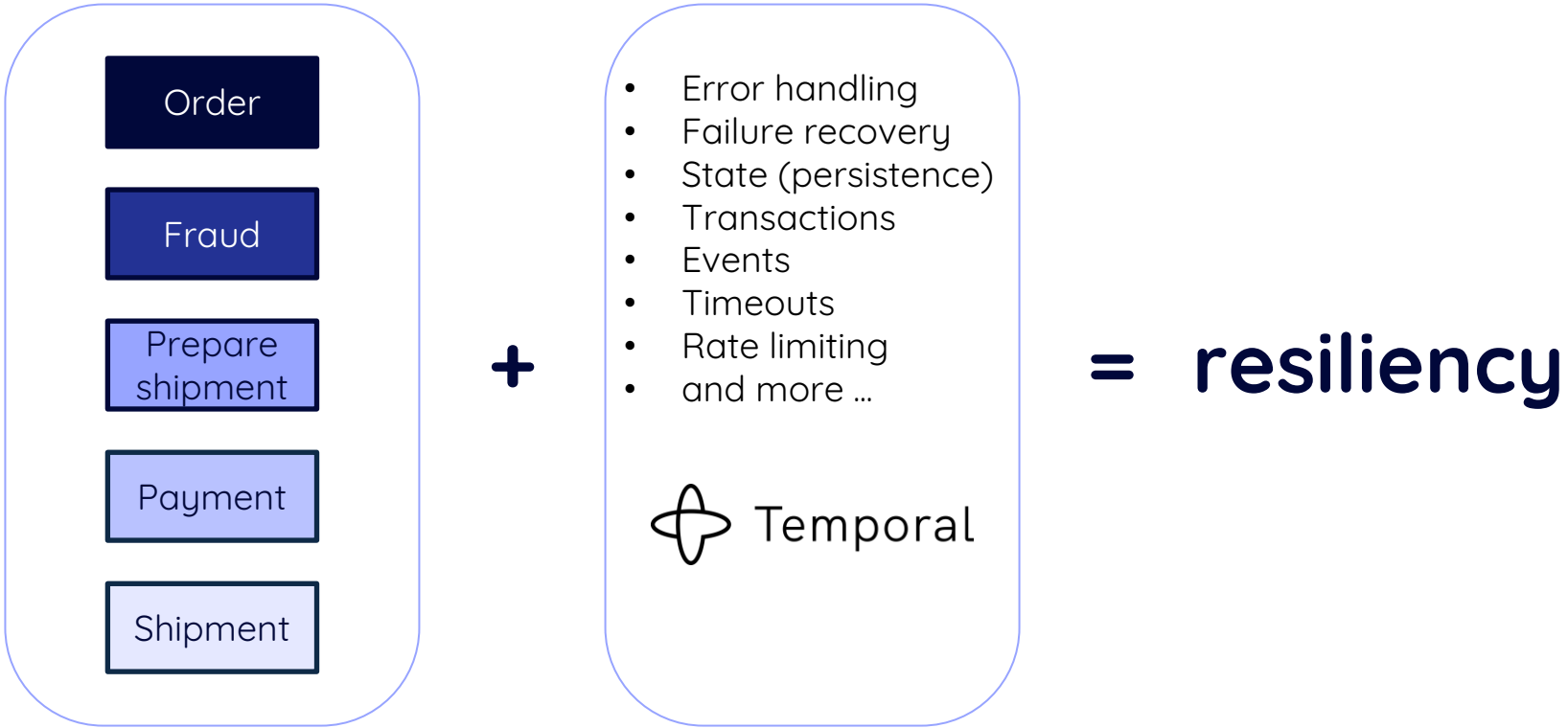
01

Problem statement

Problem statement



Problem statement



02

Introduction to Temporal

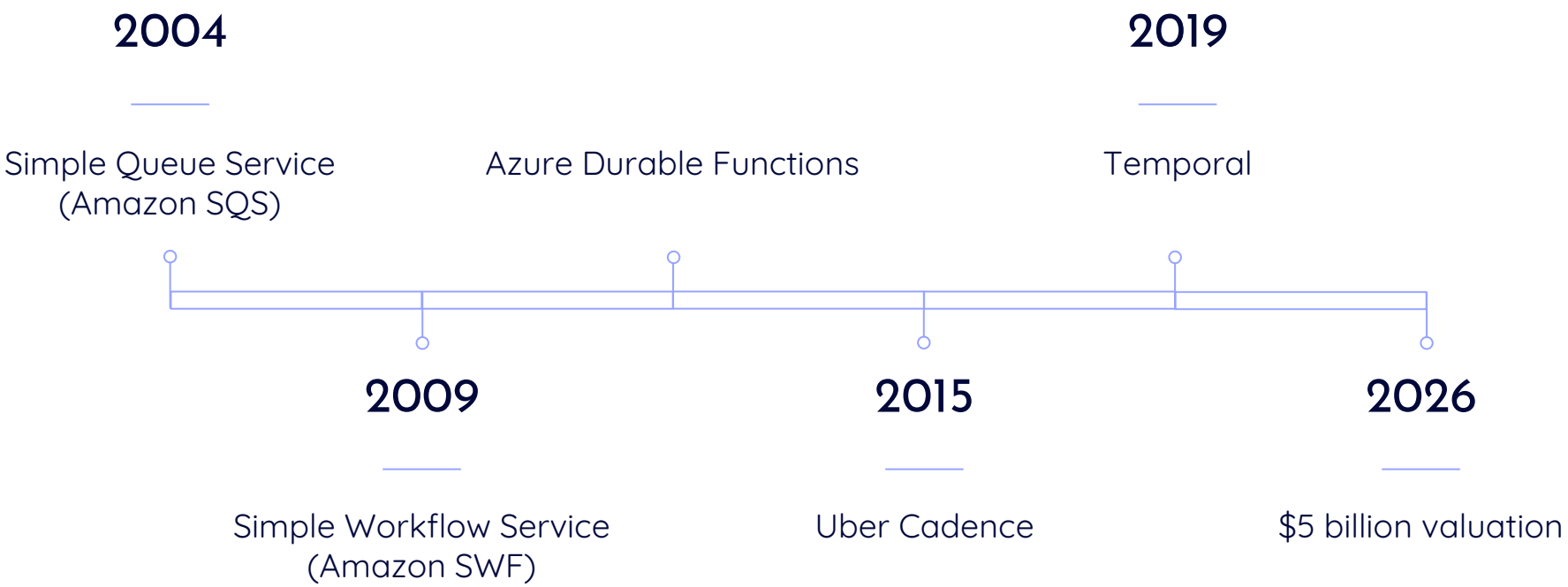
Introduction to Temporal

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 OpenAI, Netflix, Airbnb, Stripe, Snap, HashiCorp, Box
- community

History


2004
Simple Queue Service (Amazon SQS)
2009
Simple Workflow Service (Amazon SWF)
Azure Durable Functions
2015
Uber Cadence
2019
Temporal
2026
$5 billion valuation

Press Release 2/17/2026

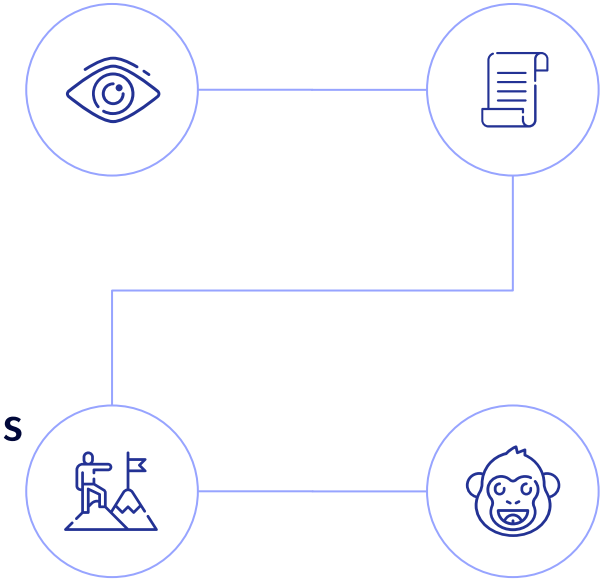
Temporal Raises $300M Series D to Make Agentic AI Real for Companies

SAN FRANCISCO — February 17, 2026 — Temporal, the open-source platform

Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code

- n8n
- Retool
- Pipedream
- Superblocks
- Windmill

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:

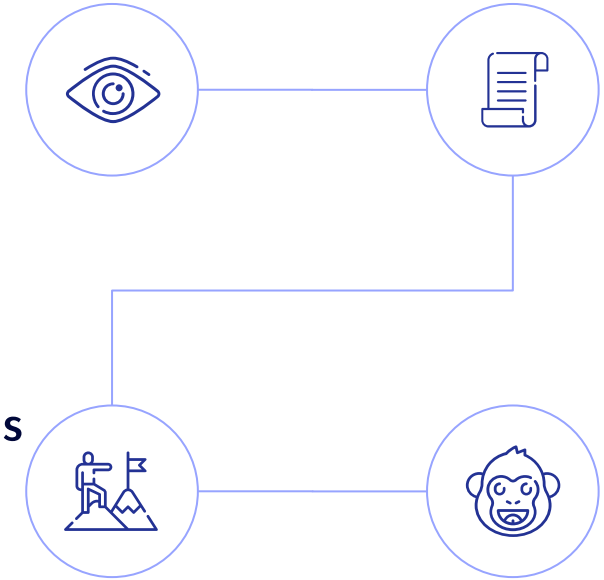
The corresponding XML

```
<?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_1f4
    <bpmn:sequenceFlow id="SequenceFlow_09hqjpg" sourceRef="Task_1f47b9v" targetRef="Task_110
    <bpmn:sequenceFlow id="SequenceFlow_1ea1mpb" sourceRef="Task_1109v9g" targetRef="Task_00m
```

Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code

- n8n
- Retool
- Pipedream
- Superblocks
- Windmill

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

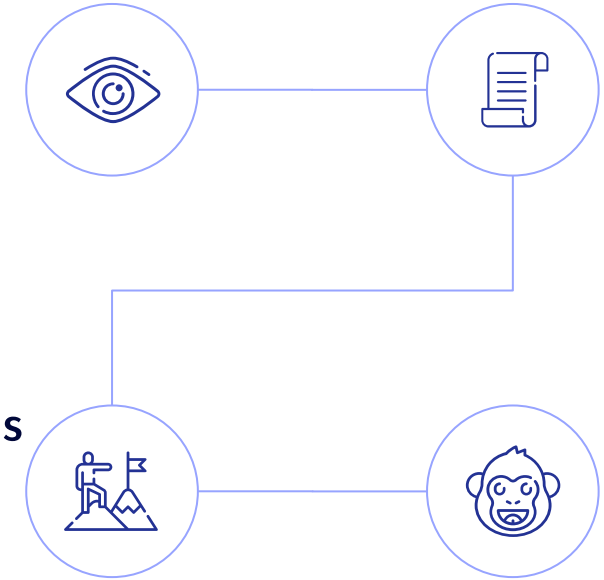
Create `greetings_workflow.json` with the following:

```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

- n8n
- Retool
- Pipedream
- Superblocks
- Windmill

OSS 15,962

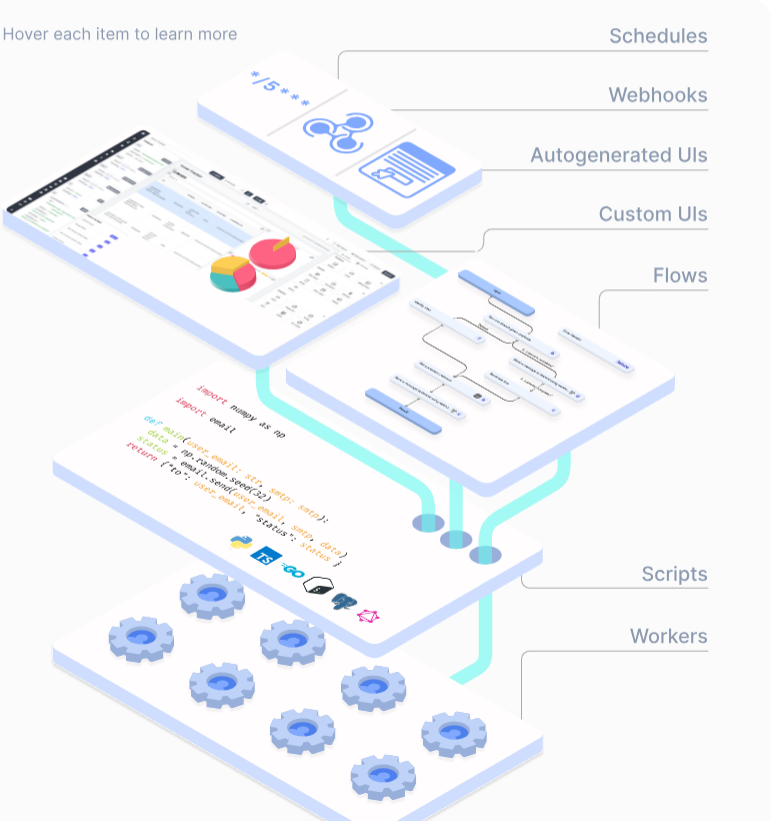
Build, deploy and monitor internal software at scale

For developers who need real code capabilities without platform engineering overhead.

- ✓ Open-source and easy to self-host
- ✓ Workflows, internal tools and data pipelines in one platform
- ✓ Full flexibility of code with Git-based collaboration
- ✓ Zero-ops infra powered by the fastest job orchestrator and workflow engine

Try Windmill cloud

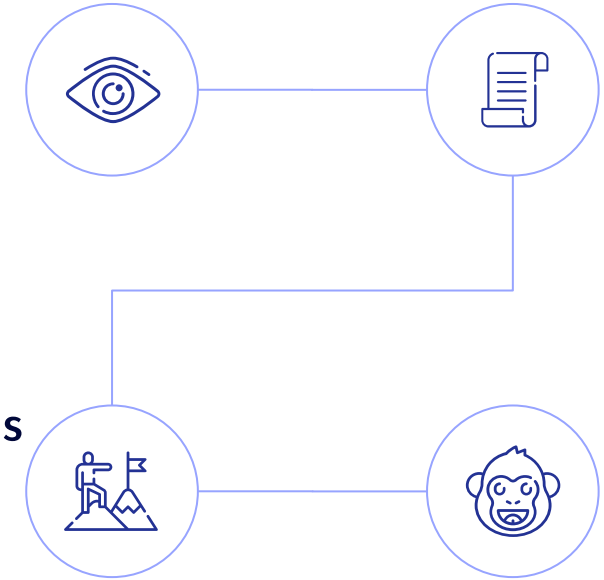
Self-host in 3 mins →



Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

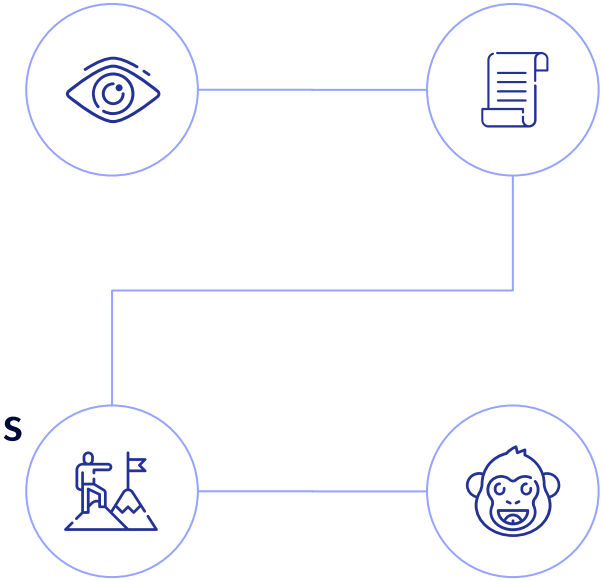
Low-code

- n8n
- Retool
- Pipedream
- Superblocks
- Windmill

Evolution

BPMN

- Camunda
- Camunda Zeebe



DSL

- Netflix Conductor

Programming langs

- Uber Cadence
- Temporal

Low-code / No-code

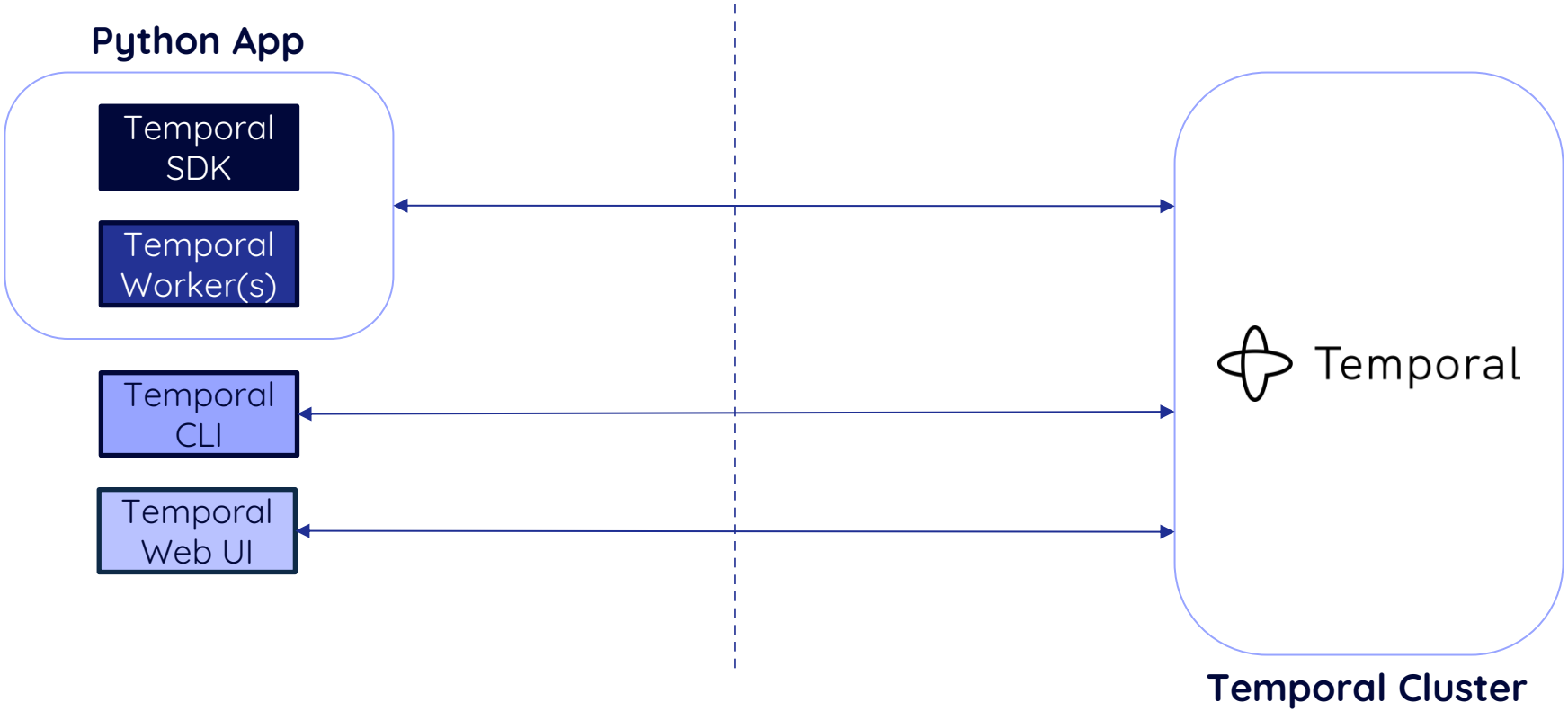
- n8n
- Retool
- Pipedream
- Superblocks
- Windmill
- Zapier
- Make

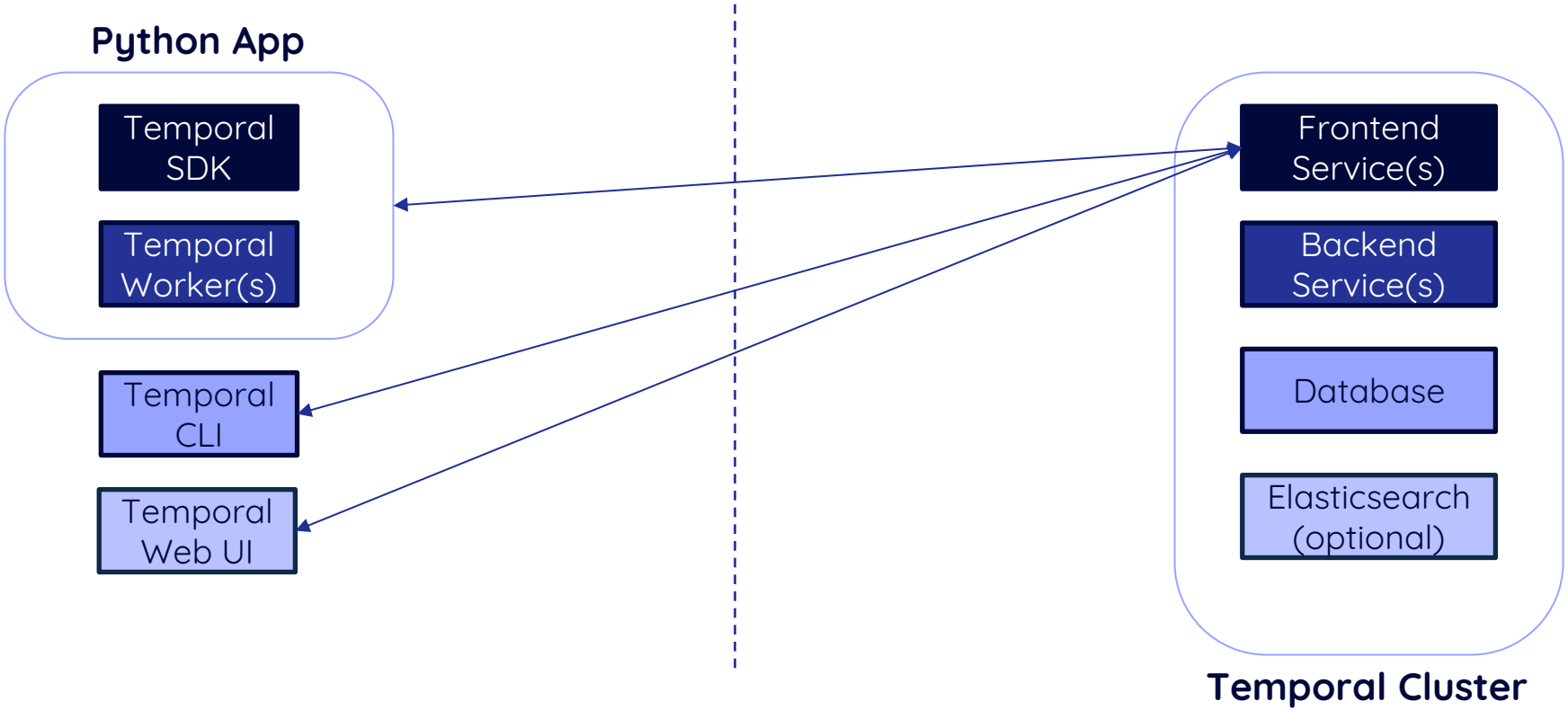
Temporal use cases

- Business process automation (e.g., client lifecycle, order processing, payments)
- Microservice orchestration
- Infrastructure provisioning
- CI/CD, data and ML pipelines
- Run agents that execute for days or weeks

03

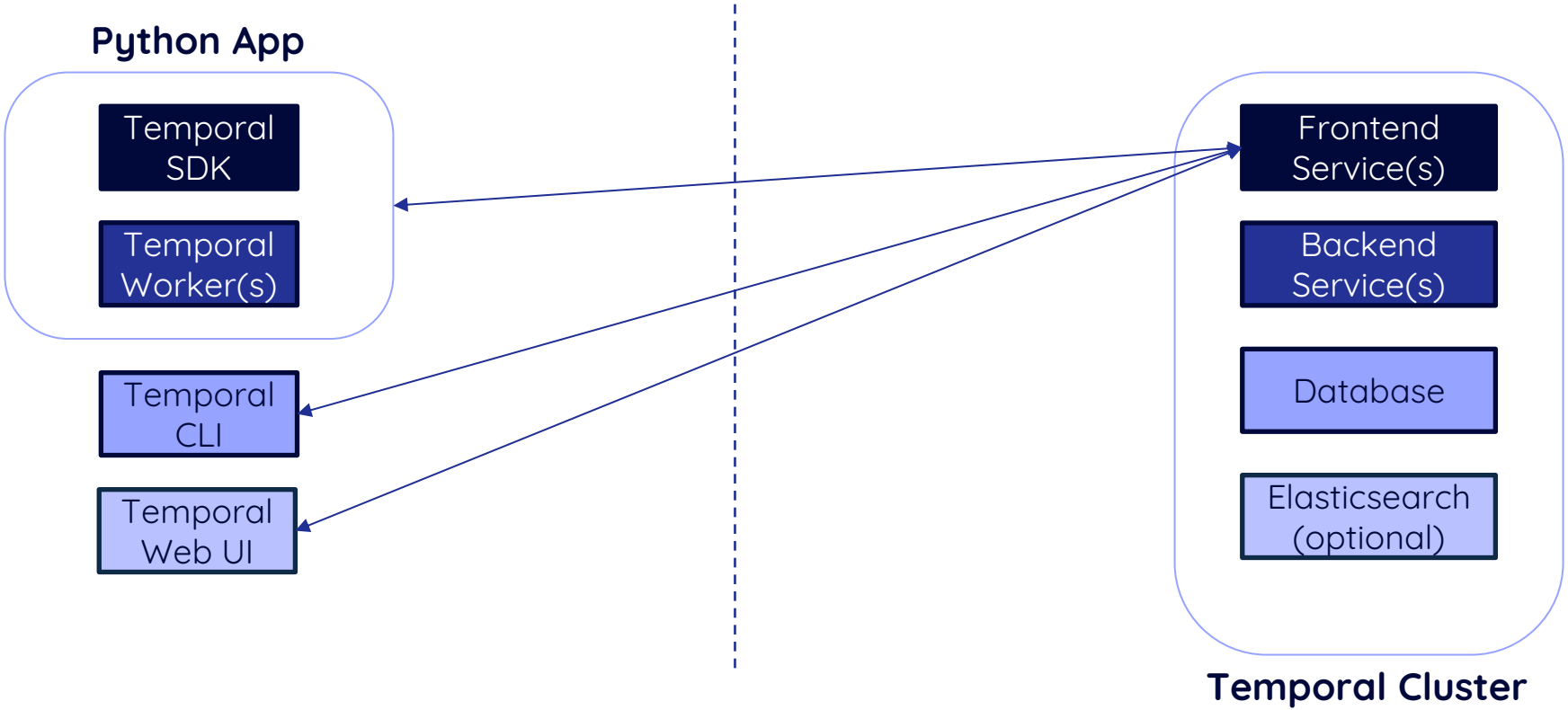
Temporal architecture


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 saves 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, configured and maintained by us (not Temporal)

One platform, two options for using it

- **Option 1:** Self-hosted deployment
  - Run the Temporal ourself
- **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.

Activity

Any operation that introduces the possibility of failure (e.g., calling microservice, querying database) should be done as part of an **Activity**. Activities retry if they fail.

Temporal 101 with Python

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

Tags: courses Python

Estimated time: ~ 2 hours, self-paced.

Cost: Free

Description

In this course, you will explore the basic building blocks of Temporal: Workflows and Activities. You'll use these building blocks along with Temporal's Python SDK to develop a small application that communicates with an external service. You'll see how Temporal helps you recover from failures and explore Temporal's execution model and event history. You'll use the Temporal Web UI and Temporal's command-line tools to explore and interact with your Workflows, and you'll use what you've learned to add new features to your existing Workflow.

When you've completed the course, you'll be able to:

- Configure an environment for developing Temporal Applications
- Use Temporal to describe and implement a business process
- Interpret Temporal's Workflow execution model

Temporal 102: Exploring Durable Execution with Python

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

Tags: courses Python

Estimated time: ~ 4 hours, self-paced.

Cost: Free

Description

In this course, you'll go beyond the basics of Temporal application development, acquiring skills that will help you on your journey to production deployment. Along the way, you'll encounter several common problems faced by Temporal developers, find out why they occur, and more importantly, how to identify and solve them, as well as how to avoid them in the future. By emphasizing key concepts and best practices, you'll gain a deeper understanding of how Temporal works and how to use it effectively.

When you've completed the course, you'll be able to:

- Apply Temporal best practices for application development

05

Temporal demo

temporal-wdi-2026

Public

Pin Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags Go to file t Add file Code

korniichuk Fix typos in readme 7f2f344 · 3 minutes ago 8 Commits

img	Update 'Option 2' section	6 minutes ago
.gitignore	Initial commit	3 hours ago
LICENSE	Initial commit	3 hours ago
README.md	Fix typos in readme	3 minutes ago
app.py	Initial commit	2 hours ago
example_workflow.py	Initial commit	2 hours ago
nbp.py	Initial commit	2 hours ago
requirements.txt	Initial commit	2 hours ago
worker.py	Initial commit	2 hours ago

README Unlicense license

Orchestrating Mission-Critical Workflows with Temporal and Python

Table of Contents:

Option 1: Demo with local development environment

About

Orchestrating Mission-Critical Workflows with Temporal and Python

Readme
Unlicense license
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 1

korniichuk Ruslan Korniichuk

Languages

Python 100.0%

Suggested workflows

Based on your tech stack

Pylint Configure

Lint a Python application with pylint.

RUSLAN KORNIICHUK

Ukrainian Data and Platform Architect in Poland

Hire me

Solid experience of 10+ years in IT. Former Lead Engineer and Architect at Fortune 500 companies. Core areas: modern data platforms and cloud computing.

Public speaker at European SME Congress, Warsaw IT Days, Data Science Summit, Linux Autumn, and EuroPython. Taught information system design and server-side technologies at the University of Silesia.

The honest, reliable, and hardworking person with always a keen eye for work details. **Click the button above to hire me**

DONATE TO UKRAINE ARMY FUNDRAISER

ARMY ASSISTANCE | SUPPORT THE FUND

How will my money be used?

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 | TRANSFERS IN UKRAINE | SWIFT TRANSFERS | CRYPTOCURRENCY

Regularity

ONE TIME | SUBSCRIPTION EVEN BETTER

Amount of donation*

€ 10 EUR, €

+100 EUR +200 EUR +500 EUR +1000 EUR

Email*

example@example.com

Do you want to receive emails from "Come Back Alive": analytics, news, and reports on how your support saves lives and strengthens the Ukrainian Defense Forces?*

○ Agree ○ Disagree

DONATE

Payment secured

tinyurl.com/WITD26

Thank you for watching!

Remember to leave your questions and rate the presentation in the section below.

WARSAW IT DAYS

A lecture selected by a Program Council consisting of recognized leaders in the IT and Data Science field.

Warsaw,
19.03.2026 - 20.03.2026

OFFICIAL LECTURE OF THE WARSAW IT DAYS

ACADEMIC PARTNERS

Event organized by: Academic Partners Foundation

Feedback

Zeskanuj kod i zostaw swoją opinię

WARSAW IT DAYS

Orchestrating Mission-Critical Workflows with Temporal and Python

Ruslan Korniichuk

https://warszawskiedniinformatyki.pl/user.html#!/lecture/WDI26-0973/rate

How did you like the lecture:

"Orchestrating Mission-Critical Workflows with Temporal and Python" ?

Rate the lecture on a scale of 1 to 5

Track: **Python by Python Summit (VoD)**

Orchestrating Mission-Critical Workflows with Temporal and Python

Online VoD

Feedback

Zeskanuj kod i zostaw swoją opinię

WARSAW IT DAYS

Orchestrating Mission-Critical Workflows with Temporal and Python

Ruslan Korniichuk

https://warszawskiedniinformatyki.pl/user.html#!/lecture/WDI26-0973/rate