

A blue-tinted photograph of a conductor leading an orchestra. The conductor is in the center, facing right, with his arms raised and a baton in his right hand. He is wearing a dark suit and glasses. The orchestra members are visible in the foreground and background, some playing violins and others with other instruments. The background is a dark, textured wall.

Temporal: Bulletproof Workflows

Ruslan Kornichuk | 24.04.2025

ver. 1.0.0

TABLE OF CONTENTS

01

Problem statement

02

Introduction to
Temporal

03

Temporal
architecture

04

Temporal Workflow
and Activity

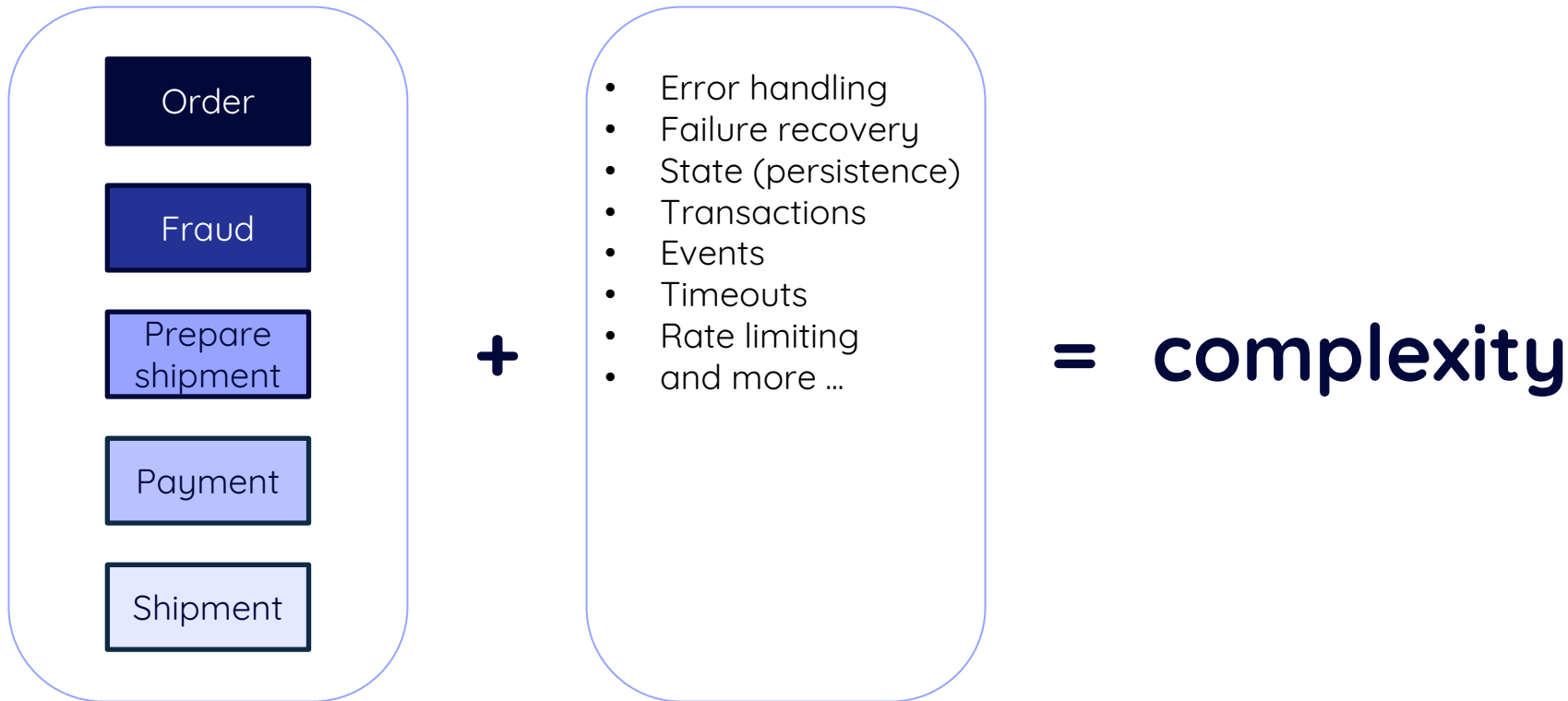
05

Temporal demo

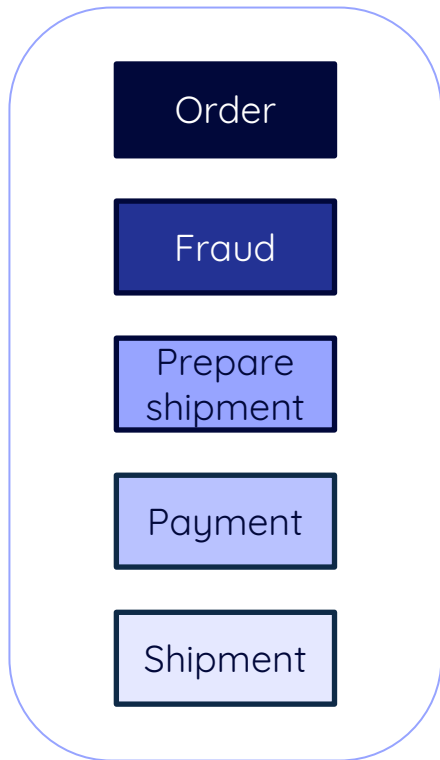
01

Problem statement

Problem statement



Problem statement



+

- Error handling
- Failure recovery
- State (persistence)
- Transactions
- Events
- Timeouts
- Rate limiting
- and more ...



Temporal

= **resiliency**

02

Introduction to Temporal

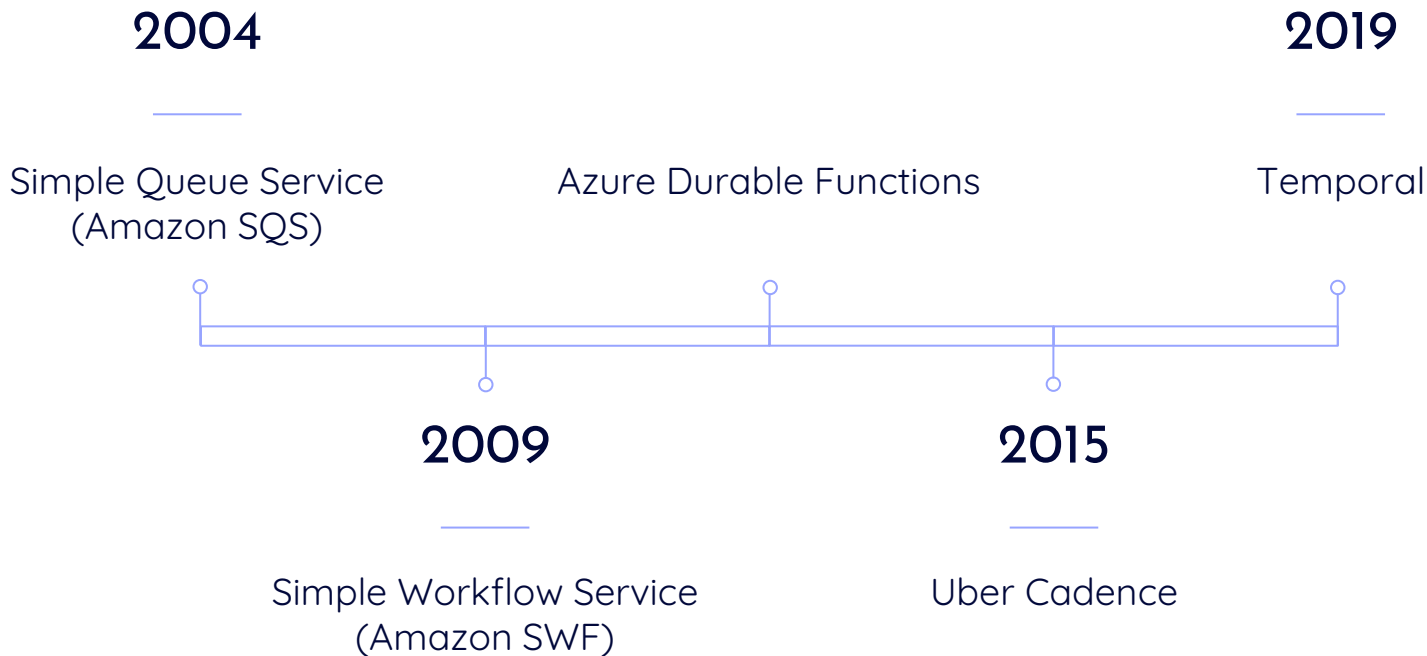
Introduction to Temporal

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



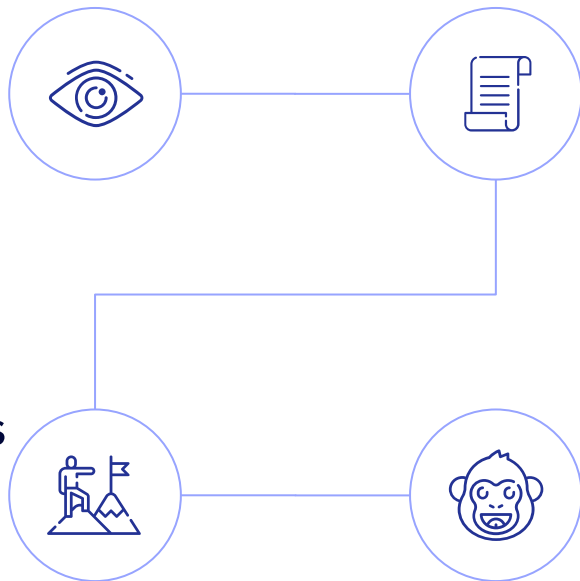
Evolution

BPMN

- Camunda
- Camunda Zeebe

Programming langs

- Uber Cadence
- Temporal



DSL

- Netflix Conductor

Low-code

- Windmill
- Pipedream
- Retool
- Superblocks

Introduction to Components

Concepts >

Console >

Modeler >

Web Modeler >

Desktop Modeler >

BPMN >

BPMN in Modeler

BPMN primer

BPMN coverage

Data flow

Tasks >

Gateways >

Events >

Subprocesses >

Markers >

DMN >

FEEL expressions >

Camunda Forms >

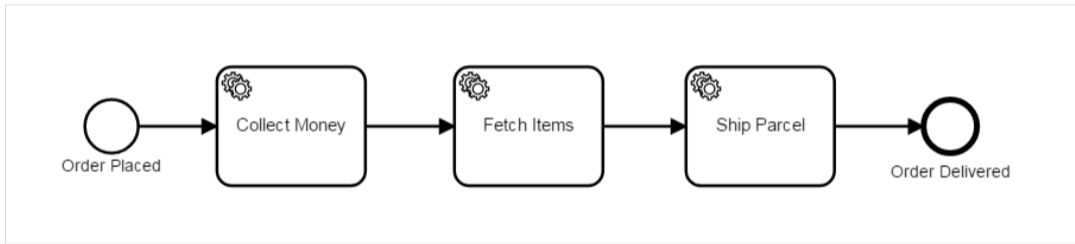
Data handling

Home > Modeler > BPMN > BPMN primer

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_09hqi9g" sourceRef="Task_1f47b9v" targetRef="Task_1109
```

Modeling BPMN diagrams

BPMN elements

Sequence flow: Controlling the flow of execution

Tasks: Units of work

Gateways: Steering flow

Events: Waiting for something to happen

Subprocesses: Grouping elements

Additional resources

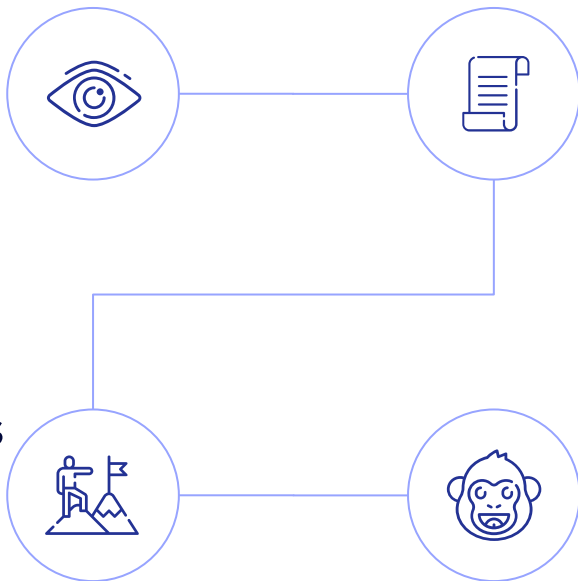
Evolution

BPMN

- Camunda
- Camunda Zeebe

Programming langs

- Uber Cadence
- Temporal



DSL

- Netflix Conductor

Low-code

- Windmill
- Pipedream
- Retool
- Superblocks

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:

```
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:

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



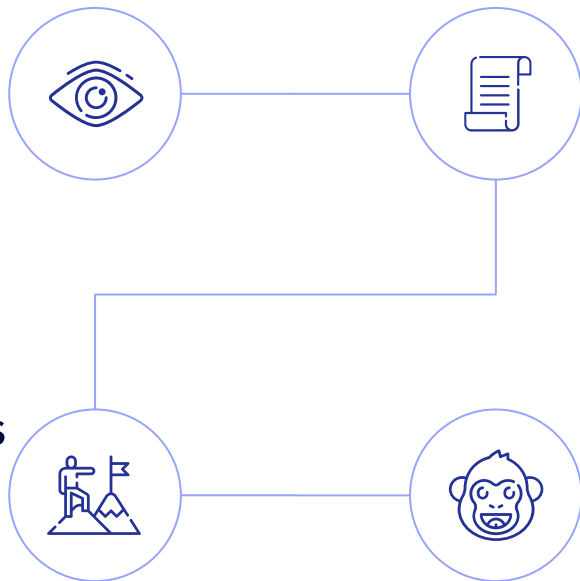
Evolution

BPMN

- Camunda
- Camunda Zeebe

Programming langs

- Uber Cadence
- Temporal



DSL

- Netflix Conductor

Low-code

- Windmill
- Pipedream
- Retool
- Superblocks

Stars 11,136

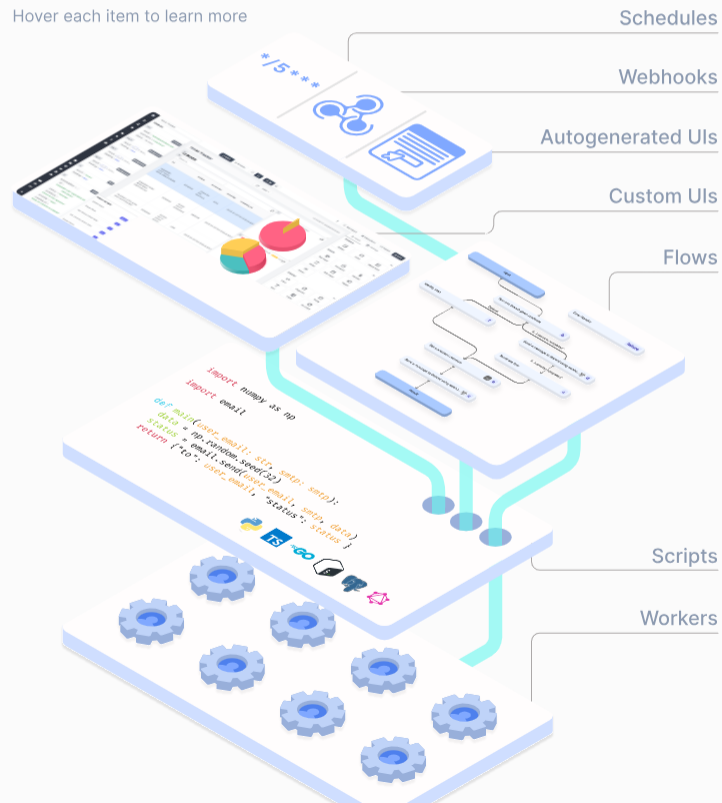
Open-source developer platform and workflow engine

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 →](#)

Hover each item to learn more



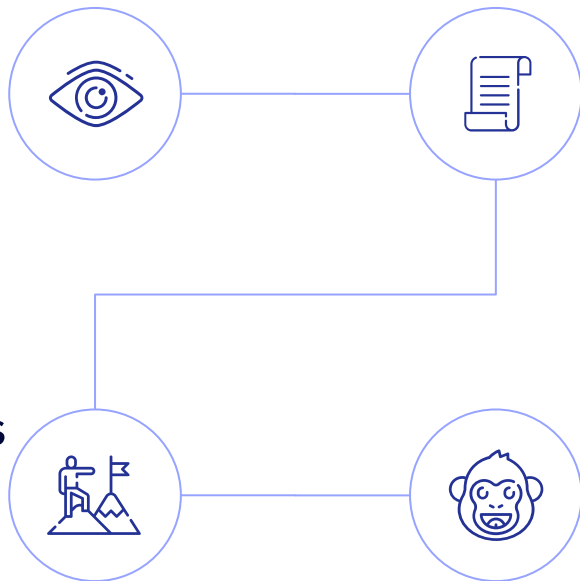
Evolution

BPMN

- Camunda
- Camunda Zeebe

Programming langs

- Uber Cadence
- Temporal



DSL

- Netflix Conductor

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

A slide with no useful information at all

- Just filling in the gap between the last slide and the next one (which will be along in just a moment)
- No need to write this down, unless you feel compelled to do so
- Nothing on this slide is examinable
- In fact, I am not really sure why I bothered with it

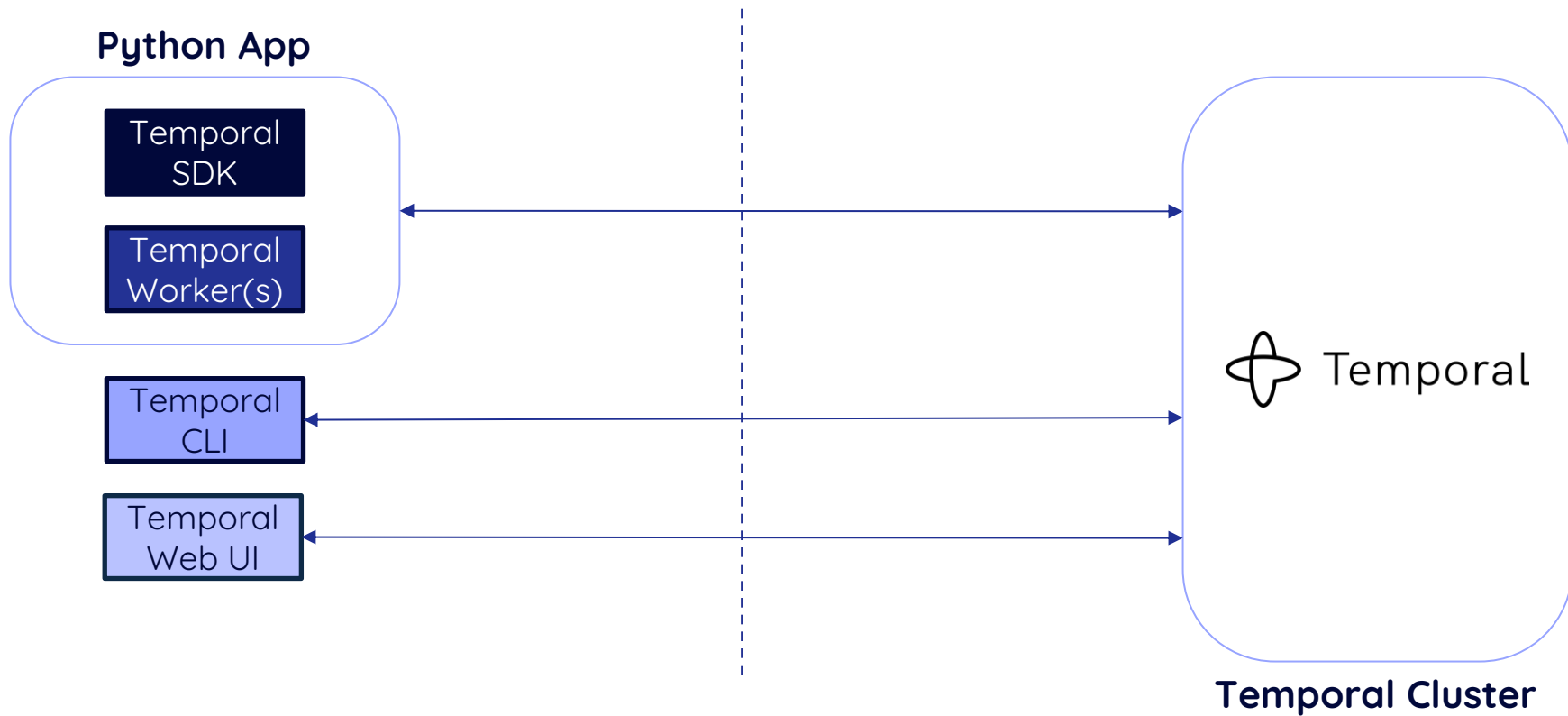
A slide with no useful information at all

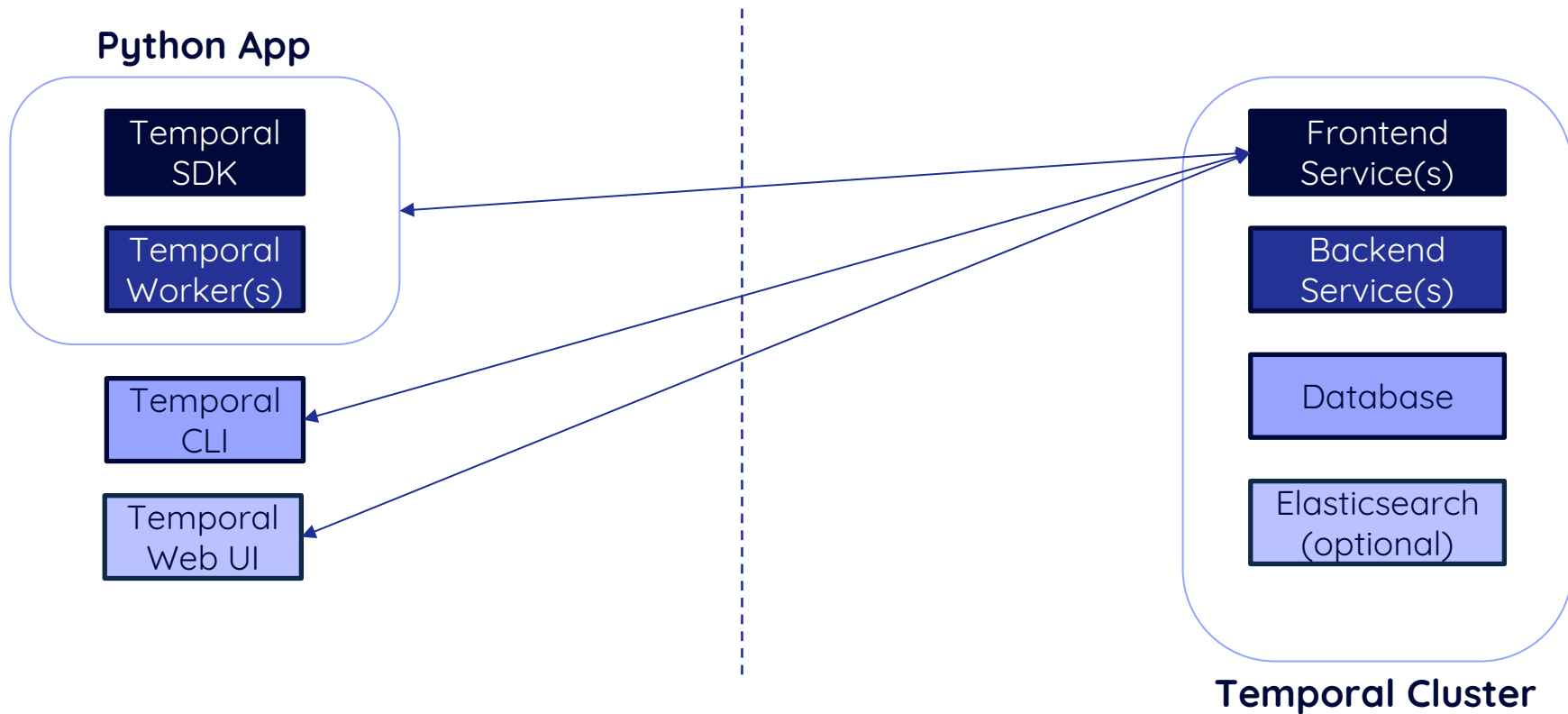
- Just filling in the gap between the last slide and the next one (which will be along in just a moment)
- No need to write this down, unless you feel compelled to do so
- Nothing on this slide is examinable
- In fact, I am not really sure why I bothered with it



03

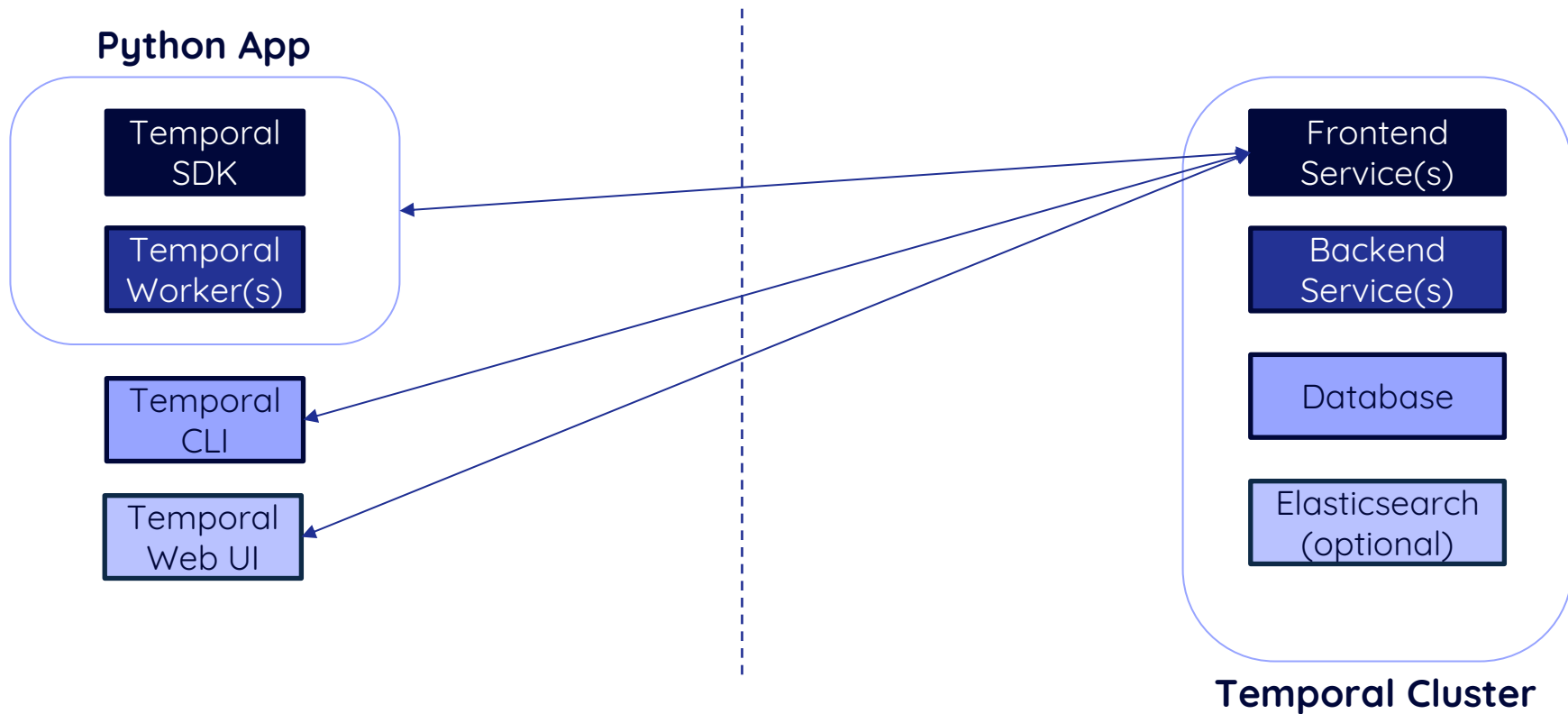
Temporal
architecture





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



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, 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, writing to file) should be done as part of an **Activity**. Activities retry if they fail.



Home

Get started with Temporal

Courses

Temporal 101: Introducing the Temporal Platform

Temporal 101 with Go

Temporal 101 with Java

Temporal 101 with Python

Temporal 101 with TypeScript

Temporal 102: Exploring Durable Execution

Crafting an Error Handling Strategy

Versioning Workflows

Securing Application Data

Interacting with Workflows

Introduction to Temporal Cloud

Project-based Tutorials



Courses



Temporal 101: Introducing the Temporal Platform



Temporal 101 with Python

Temporal 101 with Python

Last updated on **Mar 11, 2024**

Tags: [courses](#) [Python](#)



Estimated time: ~🕒 2 hours, self-paced.

Cost: Free

Description



Ask AI



Durable Execution

Temporal 102: Exploring Durable Execution with Go

Temporal 102: Exploring Durable Execution with Java

Temporal 102: Exploring Durable Execution with Python

Temporal 102: Exploring Durable Execution with TypeScript

Crafting an Error Handling Strategy

Versioning Workflows

Securing Application Data

Interacting with Workflows

Introduction to Temporal Cloud

Project-based Tutorials

Example applications

Zines

Dev guide

Documentation



[Courses](#)

[Temporal 102: Exploring Durable Execution](#)

[Temporal 102: Exploring Durable Execution with Python](#)

Temporal 102: Exploring Durable Execution with Python

Last updated on **Mar 7, 2024**

Tags: [courses](#) [Python](#)



Estimated time: ~🕒 4 hours, self-paced.

Cost: Free



Ask AI

05

Temporal demo

temporal-python-summit-2024 Public

📌 Pin

👁️ Unwatch 1 ▾

🍴 Fork 0 ▾

★ Star 0 ▾

🔗 main ▾

🔗 1 Branch

🏷️ 0 Tags

🔍 Go to file

📁 Add file ▾

<> Code ▾

korniichuk Add images to readme	40df194 · 15 minutes ago	🕒 3 Commits
📁 img	Add images to readme	15 minutes ago
📄 .gitignore	Initial commit	1 hour ago
📄 LICENSE	Initial commit	1 hour ago
📄 README.md	Add images to readme	15 minutes ago
📄 app.py	Initial commit	31 minutes ago
📄 example.py	Initial commit	31 minutes ago
📄 worker.py	Initial commit	31 minutes ago

📖 README

🔒 Unlicense license

✎

☰

🔗 Temporal: Bulletproof Workflows

Step 1/4: Installation

Install Temporal CLI

To install the Temporal CLI, download the version for your architecture:

About

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



Suggested workflows

Based on your tech stack

📄 SLSA Generic

Configure



RUSLAN KORNIICHUK

Software and Platform Engineer in Poland



Hire me

Solid experience of 10+ 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/PYCONLT3

