Temporal.io: Durable Execution en Python

Arquitecturas resilientes sin plumbing innecesario.

Vicente Esteban Bañuelos

Staff Software Engineer en Grid Dynamics

- Arquitecto de Software experto en aplicaciones distribuidas.
- Especialista en orquestación de workflows complejos.
- Evangelista de la simplificación de infraestructura mediante código.

El Caos de lo Distribuido

En microservicios modernos, el "happy path" es la excepción:

- Fallos de red impredecibles.
- Lógicas de reintento frágiles en el cliente.
- Procesos "zombie" que nunca terminan ni fallan.

Workflows Ad-hoc (El Status Quo)

"Message Queues + Database"

Implementar flujos manualmente requiere una montaña de **plumbing**:

- Manejo manual de colas (RabbitMQ/SQS).
- Persistencia de estado en DB en cada paso.
- Lógica de temporizadores para timeouts.
- Implementación de patrones de Saga complejos.

Riesgos:

Inconsistencia de datos, pérdida de mensajes, observabilidad nula del flujo completo y código de negocio enterrado en infraestructura.

Ad-hoc vs Temporal.io

Temporal elimina el 80% del código de infraestructura necesario para flujos distribuidos.

¿Qué es Durable Execution?

Inmunidad a Fallos

El código es inmune a fallos de infraestructura. Si el worker muere, Temporal lo reanuda exactamente donde se quedó, con todas las variables locales intactas.

Abstracción de Tiempo

Workflows que pueden durar meses. Un `await sleep(days=30)` no consume CPU ni memoria, simplemente suspende el estado en la base de datos de Temporal.

Anatomía de Temporal

Workflow

Lógica determinista. Es el orquestador. En Python, usamos el SDK nativo basado en `asyncio`.

Activity

Donde ocurre el mundo real (APIs, DBs, Side Effects). Son reintentables y no necesitan ser deterministas.

Worker

Tu proceso Python que ejecuta el código. Es ligero y escala horizontalmente de forma independiente.

vs AWS Step Functions

Código > JSON:

</> Python nativo vs. ASL (Amazon States Language).

Testing local simple vs. Simulación compleja.

Sin Vendor Lock-in: Corre en local, K8s o cualquier nube.

Soporte y Python

Lenguaje	Estatus SDK	Nivel de Soporte
Python	**Productivo (GA)**	Soporte asyncio, type hinting, pydantic.
Go / Java	Maduro	Referencia para sistemas de altísima escala.
TS / JS	Productivo	Ideal para apps web y serverless.
.NET / PHP	Productivo	Ecosistemas empresariales y legacy.

Ventajas: Python Experience

Observabilidad

Visibilidad total del historial de ejecución en el UI. Se acabó el debugear logs dispersos entre microservicios.

Testability

Puedes testear workflows complejos como funciones unitarias, simulando el paso del tiempo instantáneamente.

Retries

Políticas de reintento granulares por actividad, manejadas por el motor, no por tu código.

Desafíos y Consideraciones

No todo es "gratis". Implementar Temporal requiere un cambio de mentalidad:

El Determinismo

Los workflows no pueden usar `random`, `datetime.now()` o efectos secundarios directamente. Deben ir en Activities.

Infraestructura

Debes gestionar el Temporal Server (Postgres/Cassandra + Elasticsearch) o usar Temporal Cloud.

💡 Pro-tip:

Empieza con procesos críticos de negocio donde la pérdida de estado sea inaceptable (Pagos, Onboarding, Provisioning).

Python SDK en Acción

```python
@workflow.defn
class OrderWorkflow:
    @workflow.run
    async def run(self, order_id: str):
        # Activity con reintentos
        await workflow.execute_activity(
            process_payment, order_id,
            start_to_close_timeout=timedelta(s=5
        )
        # Sleep durable
        await workflow.sleep(timedelta(days=7))
```

¿Preguntas?

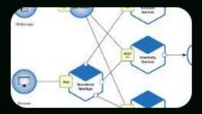
Hablemos de resiliencia y orquestación durable.

linkedin.com/in/vicente-esteban

Vicente Esteban Bañuelos | Staff Software Engineer @ Grid Dynamics

temporal.io/docs/python

Image Sources



http://microservices.io/i/Microservice_Architecture.png

Source: microservices.io

https://media.istockphoto.com/id/1669453534/photo/3d-render-cloud-computing-circuit-board-background.jpg?s=612x612&w=0&k=20&c=hpJ-qm6jT-lp-1grdsbw6Xkr_fWz36UV_JnMB68geJc=

Source: www.istockphoto.com

https://static.vecteezy.com/system/resources/thumbnails/034/972/878/small/programming-code-on-a-computer-screen-source-code-technology-background-photo.jpg

Source: www.vecteezy.com