Managing operations with Temporal in OSM

Mark Beierl (Canonical)
Gulsum Atici (Canonical)

08/03/2023
Agenda

- What is Temporal?
- Managing OSM Operations with Temporal
- Exploring Temporal Concept
- Demo
How Did This Start?
Temporal
Why durable execution changes everything
8 Fallacies of Distributed Systems

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn’t change.
6. Transport cost is zero.
7. The network is homogeneous.
8. There is one administrator.
Engineers have paid the price

This is a distributed system
Engineers have paid the price again

The same code, after adding support for retries during withdrawal
Engineers have paid the price again and again and again

The same code, after adding support for retries during withdrawal and deposit, and performing a compensation if the withdrawal succeeds but the deposit fails
Temporal was created to solve these challenges:

An open source **Durable Execution System**

- Guarantees the successful and correct execution of any feature, function or service in the face of any infrastructure failure.
- Every execution is recorded to allow for recoverability, replayability and correctness.
- Abstracts developers away from the underlying infrastructure and resources.
Managing Operations in OSM

The current state
So What Is a Workflow

Start

NS State = “BUILDING”

Get VCA Public Key

Deploy Execution Envs

NSR status = “creating”

Deploy KDUs

Deploy VDUs

End

Result

Deploy KDUs

Deploy VDUs

NS State = “BUILDING”

Get VCA Public Key

Deploy Execution Envs

NSR status = “creating”

Deploy KDUs

Deploy VDUs

End

Result
Expected Failures

Start

NSR status = “creating”

NS State = “BUILDING”

Get VCA Public Key

Deploy Execution Envs

Deploy KDUs

Deploy VDUs

Result

End
Unexpected Failures

- Start
- NS State = “BUILDING”
- Get VCA Public Key
- NSR status = “creating”
- Deploy KDUs
Exploration of a Concept

Are NS LCM Operations just Workflows?
Adopt incrementally

Learn
- Intro to SDKs

Tutorials
- Hello World
- Money Transfer
- eCommerce

Samples
- temporalio/samples-java

Refactor (or Write)

Client
- Maven
- Gradle Groovy DSL
- Server
Learn

https://github.com/temporalio/samples-python
https://github.com/temporalio/sdk-python

- **hello** - All of the basic features.
  - **hello_activity** - Execute an activity from a workflow.
  - **hello_activity_choice** - Execute certain activities inside a workflow based on dynamic input.
  - **hello_activity_multiprocess** - Execute a synchronous activity on a process pool.
  - **hello_activity_retry** - Demonstrate activity retry by failing until a certain number of attempts.
  - **hello_activity_threaded** - Execute a synchronous activity on a thread pool.
  - **hello_async_activity_completion** - Complete an activity outside of the function that was called.
  - **hello_cancellation** - Manually react to cancellation inside workflows and activities.
  - **hello_child_workflow** - Execute a child workflow from a workflow.
  - **hello_continue_as_new** - Use continue as new to restart a workflow.
  - **hello_cron** - Execute a workflow once a minute.
  - **hello_exception** - Execute an activity that raises an error out of the workflow and out of the program.
  - **hello_local_activity** - Execute a local activity from a workflow.
  - **hello_mtls** - Accept URL, namespace, and certificate info as CLI args and use mTLS for connecting to server.
  - **hello_parallel_activity** - Execute multiple activities at once.
  - **hello_query** - Invoke queries on a workflow.
  - **hello_search_attributes** - Start workflow with search attributes then change while running.
  - **hello_signal** - Send signals to a workflow.
- **activity_sticky_queue** - Uses unique task queues to ensure activities run on specific workers.
- **activity_worker** - Use Python activities from a workflow in another language.
- **custom_converter** - Use a custom payload converter to handle custom types.
- **custom_decorator** - Custom decorator to auto-heartbeat a long-running activity.
- **encryption** - Apply end-to-end encryption for all input/output.
- **open_telemetry** - Trace workflows with OpenTelemetry.
- **pydantic_converter** - Data converter for using Pydantic models.
- **sentry** - Report errors to Sentry.
I have the code for:

- Workflows A, B, C
- Activities 1, 2, 3
Demo
Periodic Gridfs Cleaning with Temporal Workflows
Why FsClean workflow is created?

- Existing production issue ([https://osm.etsi.org/bugzilla/show_bug.cgi?id=2024](https://osm.etsi.org/bugzilla/show_bug.cgi?id=2024))
- NSD/VNFD upload can abandon files (charts, bundles)
- Performance of file synchronization operations can be impacted
- A workflow is created to delete the unused Gridfs files in OSM MongoDB
- The workflow could be scheduled to run periodically so it always keeps the OSM filesystem clean
A Worker invocation Flow

I have the definition of Workflows, Activities

FsUploadWorkflow
Activities:
- upload_unnecessary_file

FsCleanWorkflow
Activities:
- clean_gridfs

LCM

Temporal Client

Temporal Server

Task queue
- Temporal api

inputs to run worker

LCM Worker

I run in LCM when the container started

Now, I am aware of LCM worker, I manage this worker after that.

A worker can only execute the specified workflows and activities which are already registered to the worker during invocation.

Now, I am aware of LCM worker, I manage this worker after that.
A Worker invocation Flow

def start(self):
    # Start LCM Temporal Worker
    temporal_api = get_temporal_api(self.main_config)
    workflows_data = get_workflows_data(self.fs, self.main_config.database.uri)
    lcm_worker = WKTemporal(workflows_data, temporal_api, "lcm.temporal")
    self.logger.info("Starting LCM temporal worker")
    try:
        asyncio.run(lcm_worker.run_worker())
    except Exception as err:
        self.logger.exception("Exception '{:}' at messaging read loop", err, exc_info=True)

    return workflows_data

def get_workflows_data(fs, uri):
    data_upload = FsUploadActivities(fs)
    data_cleanup = FsCleanActivities(uri)
    workflows_data = {
        "task_queue": lcm_task_queue,
        "workflows": [FsUploadWorkflow, FsCleanWorkflow],
        "activities": [data_upload.upload_unnecessary_file, data_cleanup.clean_gridfs],
    }
    return workflows_data

    return asyncio.create_task(
        Worker(
            self.temporal_client,
            task_queue=self.workflows_data["task_queue"],
            workflows=self.workflows_data["workflows"],
            activities=self.workflows_data["activities"],
        ).run(),
    )
Executing a Workflow using the Temporal client

- **External Client/NBI**: I want to execute FsCleanWorkflow

- **LCM**: I am aware of LCM worker, I manage this worker

- **Temporal Client**: Temporal Client

- **Temporal Server**: Temporal Server

Inputs to execute WF:
- FsCleanWorkflow
- Args to workflow
- WF ID
- Task queue

Execute: FsCleanWorkflow

Return Result:

Run FsClean WF
...
Executing a Workflow using the Temporal client

def get_upload_workflow():
    return {
        "workflow_name": "FsUploadWorkflow",
        "workflow_id": "FsUploadWF",
        "task_queue": lcm_task_queue,
        "data": {
            "path": str(uid.uid4()),
            "indata": {
                "some_key": "some_value",
                "other_key": "other_value",
            }
        }
    }

async def execute_workflow(
    self, task_queue: str, workflow_name: str, workflow_data: any, id: str = None
):
    handle = await self.start_workflow(
        task_queue=task_queue,
        workflow_name=workflow_name,
        workflow_data=workflow_data,
        id=id,
    )
    result = await handle.result()
    self.logger.info(f"Completed workflow {workflow_name}, id {id}")
    return result

@task
async def do_work():
    workflow = workflow()
    await asyncio.sleep(1)
    return workflow

async def main():
    execution = await asyncio.create_task(do_work())
    print(execution)

if __name__ == '__main__':
    asyncio.run(main())
Executing a Scheduled Workflow by Temporal UI

I want to execute FsCleanWorkflow as a schedule.

Inputs to execute WF:
- FsCleanWorkflow
- Args to workflow
- WF ID
- Task queue

Return Result

Run FsClean WF
Abstract overview of the workflow for the demo

1. Run Worker

   Workflows → Activities
   FsUploadWorkflow → upload_unnecessary_file
   FsCleanWorkflow → clean_gridfs

2. Execute FsUploadWorkflow as gRPC

3. Execute FsCleanWorkflow as cron job

   gRPC

   Temporal

   Web UI

   Services

   MariaDB

   MongoDB

   Gridfs

   Worker

   CLIENT

   gRPC

   1. Run Worker

   2. Execute FsUploadWorkflow

   3. Execute FsCleanWorkflow as cron job

© ETSI
I have the definition of Workflows, Activities

FsUploadWorkflow
Activities:
- upload_unnecessary_file

FsCleanWorkflow
Activities:
- clean_gridfs

I run in LCM when the container started

A worker can only execute the specified workflows and activities which are already registered to the worker during invocation.

External Client

Inputs to execute WF
- FsUploadWorkflow
- Args to workflow
- WF ID
- Task queue

I want to execute FsUpload WF

LCM Worker

Inputs to run worker
- FsUploadWorkflow
  Activities:
  - upload_unnecessary_file
- FsCleanWorkflow
  Activities:
  - clean_gridfs

LCM

Temporal Client

Temporal Server

Run worker
1 Run worker

Execute FsUpload WF
2 Execute WF

I am aware of LCM worker, I manage this worker

Return Result
Execute FsUploadWorkflow in order to **upload unnecessary** files to Gridfs

<table>
<thead>
<tr>
<th>Date &amp; Time</th>
<th>Event Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2023-03-06 UTC 05:36:02.51</td>
<td>upload_unnecessary_file</td>
<td>{ &quot;file_added&quot; : { &quot;file_name&quot; : &quot;unnecessary_file&quot; } }</td>
</tr>
</tbody>
</table>
Create Schedule for FsCleanWorkflow

**Name**
FsCleanWorkflow

**Workflow Type**
FsCleanWorkflow

**Workflow Id**
FsCleanWF-10

**Task Queue**
ksm-queue

**Frequency**

**Interval**

**Days of the Week**

**Days of the Month**

**String**

**Recurring Time**
Specify the time interval for this schedule to run (for example every 5 minutes).

| 00 | days | 00 | hrs | 3 | min | 00 | sec |

**Offset**
Specify the time to offset when this schedule will run (for example 15 min past the hour).

| 00 | min |

Create Schedule  Cancel
Scheduled FsCleanWorkflow cleans

**Recent Runs**

<table>
<thead>
<tr>
<th>Date &amp; Time</th>
<th>Event Type</th>
<th>Event Time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 2023-03-06 UTC 05:37:00.44</td>
<td>clean_gridfs</td>
<td>2023-03-06 UTC 05:37:00.44</td>
<td></td>
</tr>
<tr>
<td>7 ActivityTaskCompleted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 ActivityTaskStarted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 ActivityTaskScheduled</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Frequency**

Every 01min:00sec
Thank You!