SHEPHERD

Introduction -
Shepherd is a heterogeneous workflow executor service. This helps to execute workflow which can contain heterogeneous nodes as well. This majorly focused to solve those business problem which can not be complete in single process unit.

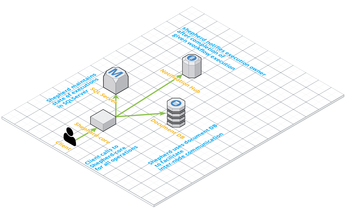
Major features -

- Nodes can be heterogeneous in single workflow.
  - Explaining with e-commerce reversal use-case :
    - In reversal workflow, the major involvements are :
      - Take back already paid taxes from government.
      - Give back payment to customer.
      - Start transportation to pick shipment.
      - Return/Collect money from seller.
      - Do all ledgering.
    - Above nodes together forms a single workflow. All these nodes are individual teams, and they have their codebase in different language. At the same time, they have to collaborate with each other to complete reversal (return/replacement) for given end customer. For this use-case, workflow with heterogeneous node support is must.
- Supported workflows - Conditional and Plain workflow.
  - Conditional workflow - Conditional workflow is a workflow that executes child node on the basis of parent node's response (which is name of an edge)
  - Plain workflow - Plain workflow is a workflow that executes all its child nodes once all parent nodes completed successfully. This type of workflow is very famous in data analytics. Ex -> ETL executions.
- Execute workflow.
- Resume workflow.
- Restart workflow.
- Kill workflow.
- Retry given node with linear/exponential time-off.
- Get status.
- Notify client through AWS SNS/Azure Notification Hub.

Component diagram -

Shepherd majorly contains 4 components in their architecture -

- **Shepherd-Core -**
  This is the core part of Shepherd service. This service maintains workflow nodes sequence, and executes nodes as per DAG sequence.
- **SQL Server -**
  This helps to maintain states of all objects, executions, nodes. These data can be used for auditing purpose in future.
- **Document DB -**
  This helps to establish inter-node communication for a given node. Shepherd allocates 2 MB space to each execution for its inter-node communication.
- **Notification Hub -**
  This helps to notify execution owner about state once execution completed (it can be successfully completed/failed).

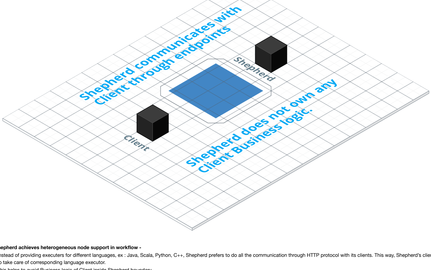


Terminology -

- **Workflow -**
  Workflow is a directed acyclic graph, which consists of Nodes, Edges.
- **Node -**
  Node is the fundamental unit of graph. We can consider it as single processing unit. Two nodes can be connected through edge.
- **Edge -**
  Edge is a connecting bridge between two nodes.
- **Object -**
  Processing request corresponding to a given workflow. It can be in any state -> PROCESSING/KILLED/COMPLETED/FAILED.
- **Execution -**
  Execution is the actual running instance of object. It can be in any state -> PROCESSING/KILLED/COMPLETED/FAILED.

Where Client business logic resides -

1. Shepherd is a platform that provides workflow management concept. Shepherd does not contain any client Business logic/data inside it.
2. Shepherd communicates through endpoints with clients. (Example -> REST endpoints, AWS Lambda, Azure Functions, Database endpoints, etc).



How Shepherd achieves heterogeneous node support in workflow -

1. Instead of providing executers for different languages, ex - Java, Scala, Python, C++, Shepherd prefers to do all the communication through HTTP protocol with its clients. This way, Shepherd's clients has to take care of corresponding language executor.
2. This helps to avoid Business logic of Client inside Shepherd boundary.
3. If Shepherd provides language executors, then it expects business logic of Client in Shepherd boundary. This way, it actually move clients away from SOA architeture. And, it actually gives pain to all its clients who owns multiple endpoints to update their business logic inside Shepherd boundary.

How nodes can communicate with each other -

1. Shepherd provides 2 MB space to each execution for inter node communication.
2. Client can store shared data in form of document using GET/PUT API.
3. Its recommended to use this space for your metadata only. Store your actual data on blob services (AWS S3 / Azure Blob service).

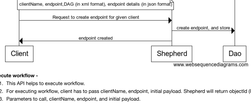
How child node can be pick as next execution node after completion of parent node execution -

1. Every node has to return Edge name as response. Shepherd use this information to execute next node in the given workflow.

Workflows -

- **Client registration workflow -**
  1. This is the very first step. This is the API through which client can register himself in Shepherd.
  2. While registering, client has to pass "clientName". [Currently, skipping security discussion. Will prioritise in V2 cut.]

- **Endpoint creation workflow -**
  1. This API helps to create new endpoint. There can be multiple endpoints against single client. For example, for client -> "data_platform", possible endpoints -> "data_platform_beta", "data_platform_staging", "data_platform_prod", "hitesh_dev_data_platform", etc.
  2. To setup endpoint, client has to share workflow graph (in xml format), and corresponding endpoint details (in son format).



- **Execute workflow -**
  1. This API helps to execute workflow.
  2. For executing workflow, client has to pass clientName, endpoint, initial payload. Shepherd will return objectId (identifier of workflow in Shepherd).
  3. Parameters to call, clientName, endpoint, and initial payload.

- **Resume workflow -**
  1. This helps to resume workflow from where it failed last time.
  2. When Shepherd gets request to execute workflow, it generates objectId for the same. Also, it creates execution_id, which is the actual running instance of given objectId (client request).
  3. One ObjectId can contain multiple executionIds. It depends upon how many times client requested for restart operation.
  4. Parameters : ObjectId
- **Restart workflow -**
  1. This API helps to restart workflow, This means, it kills the current running instance of given objectId, if there is any and start a new running instance.
  2. By this operation, it increases the total execution counts by 1 for given object id.
  3. Parameters : ObjectId
- **Kill workflow -**
  1. This API helps to kill the workflow. This means, it kills the current running instance of given objectId, if there is any.
  2. Parameters : ObjectId
- **Get status -**
  1. This API helps to get status of given workflow. Possible status -> PROCESSING, COMPLETED, FAILED
  2. Parameters : Object

Schema design -
In this section, I will cover schema design, and tables that is required to manage various of Shepherd.

Tables -

Tables	Responsibility	Comments
client_details	Client details.	
endpoint_details	Endpoint details.	
object_execution	Maintains state of given object.	
node_execution	Maintains state of given node (fundamental unit of workflow).	

Table#1 - client_details

Column	Responsibility	Comments
client_id	Client unique identifier	
client_name	Name of client	
Created_at	Creation time	
Updated_at	Last update time	
Created_by	Submitter name	

Table#2 - endpoint_details

Column	Responsibility	Comments
endpoint_id	Endpoint unique identifier	
client_id	Which client is the owner of this endpoint	
workflow_graph	DAG graph details in XML format.	
endpoint_details	Endpoint file details in JSON format	
created_At	Creation time	
updated_At	Last update time	
created_by	Submitter name	

Table#3 - object_execution

Column	Responsibility	Comments
object_id	Object unique identifier	
execution_id	Execution identifier	
endpoint_id	Endpoint identifier, on which this execution is running	
status	Status of this object's execution	
error_message	Reason of failure, if any	
processed_nodes	List of Nodes that already processed in given execution	
current_executing_nodes	List of Nodes that are currently under processing in given execution	
upcoming_nodes	List of Nodes that will come to get processed.	
created_at	Creation time	
updated_at	Last updated time	
created_by	Submitter name	

Table#4 - node_execution

Column	Responsibility	Comments
node_id	Node unique identifier	
Execution_id	Execution identifier, under which this node is running	
status	Status of node	
error_message	Reason of failure, if any	
created_at	Creation time	
updated_at	Last updated time	
created_by	Submitter name	

References -

- [Cosmos DB's maximum document size] https://docs.microsoft.com/en-gb/azure/cosmos-db/sql-api-resources#documents