Workflow Engine

Design Prologue

Background information

Workflow Engine Analysis

- https://docs.google.com/presentation/d/1iryih2JaMfZ_B1wNhTVedE5l614DrD75eqL60f3WzL0/edit

Design Goals and Objectives

The following goals are supported by the designs outlined in this specification. Each goal has supporting objectives aimed at fulfilling its respective goal.

- **Ingest Temporal as workflow-engine for Meshery server**
    - Implementing/changing handlers into Workflows and Activity functions with temporal-sdk
- **Lifecycle management of Workflows and Activities**
    - A long range of workflows and activities can be triggered/scheduled/retried/restarted.
        - Future objective: Documentation explaining each workflow and activity

Implementation

Integrating Temporal as the workflow engine. It is not a feature, that is it is not a user facing component so we won't have anything to offer if we complete this first.

Temporal SDK will only offer a mechanism to wrap the pre-existing golang code into "workflows" and "activities", this may require us to slightly change the implementation but should not be significant enough to re-do the implementation of all of the pre-existing features.

Few questions needs to addressed

- Are we implementing workflows into each of the handlers?
    - What functions/handlers will be changed into workflows
        - The first and foremost can be the pattern handler. Such that each stage of the pattern engine can be an activity.

Prerequisites to contribute in this area

1. Golang
2. Understanding of Temporal, Workflows, and Activities
3. Understanding of Meshery-server

Terms used in a workflow setup

- workflowIDs
- runIDs
- taskQueues
- queryTypes
- signalNames
- signalRoutes/RouteTypes

Types of workflows to be implemented

Start-up workflows

Workflows could be set up that start with the meshery-server

- Workflows caching different data to be used after
- Workflows setting-up env like deploying meshery-operator, broker, mesh-sync

Recurring workflows

Workflows that are recurring after a certain time, updating the state within so graphql subscriptions will just get the current state of workflows

- Checking mesh-sync status
- Service discovery workflows

Clean-up workflows

Workflows that are triggered before stopping the meshery-server that are responsible for different types of clean-ups such as updating/persisting data in databases, removing different deployments, deleting downloaded/cached files.

There can be many types of workflows to be implemented in meshery-server to enhance efficiency. Workflows opens a **wide-range of new features** for both Meshery-Server and Meshery-UI

An e.g. for Meshery-UI could be persisting data temporarily with workflows, decrease the data update queries in meshery-provider

Database considerations

- Liveness probe added into Kuberenetes deployment for the postgres database to ensure restarts on database crash.
- No need for a separate helm chart for the database, the packaged database is part of Meshery server pod. Further helm charts or other forms of database deployments can be made and attached to the database at runtime. The pre-packaged database doesn't need another helm chart.
- /api/system/database: An endpoint is added to manage the database lifecycle - to remove existing connection, add new one, etc. Still in this iteration, one one database instance can be in-use at a time. POST,DELETE,GET
- Mesheryctl commands: `mesheryctl system database ` will make use of the above endpoint
-