TheThree Reqif to Rst DAG

This is a DAG (Directed Acyclic Graph) implemented using Airflow, a platform to programmatically author, schedule, and monitor workflows. The DAG is designed to convert ReqIF files to RST (reStructuredText) format and upload them to a GitHub repository. The workflow consists of the following tasks:

Task 1: load_reqif_and_configs

This task loads the ReqIF file, mapping configuration file, and GitHub configuration file. It retrieves the file paths from the DAG run's configuration. The task performs the following steps:

1. Load the ReqIF file using the `load_reqif` function from the `thethree` module.
2. Load the mapping configuration using the `load_config` function from the `thethree` module.
3. Load the GitHub configuration using the `load_config` function from the `thethree` module.
4. Determine the output RST file path based on the ReqIF file name.
5. Push the loaded ReqIF, mapping configuration, GitHub configuration, and RST file path to XCom for sharing with other tasks.

Task 2: build_json

This task takes the loaded ReqIF and mapping configuration as inputs and builds a JSON representation of the data. It uses the `build_json` function from the `thethree` module. The task performs the following steps:

1. Pull the ReqIF and mapping configuration from XCom.
2. Build the JSON data using the `build_json` function.
3. Push the JSON data to XCom for sharing with other tasks.

Task 3: build_rst

This task takes the JSON data and mapping configuration as inputs and builds an RST representation of the data. It uses the `build_rst` function from the `thethree` module. The task performs the following steps:

1. Pull the JSON data and mapping configuration from XCom.
2. Build the RST data using the `build_rst` function.
3. Push the RST data to XCom for sharing with other tasks.

Task 4: upload_to_github

This task takes the RST data, RST file path, and GitHub configuration as inputs and uploads the RST file to a GitHub repository. It uses the `upload_to_github` function from the `thethree` module. The task performs the following steps:

1. Pull the RST data, RST file path, and GitHub configuration from XCom.
2. Upload the RST file to GitHub using the `upload_to_github` function.

Task 5: update_index_rst

This task takes the JSON data, mapping configuration, GitHub configuration, and RST file path as inputs and updates the index.rst file in the GitHub repository. It uses the `update_index_rst` function from the `thethree` module. The task performs the following steps:

1. Pull the JSON data, mapping configuration, GitHub configuration, and RST file path from XCom.
2. Extract the module type from the JSON data.
3. Update the index.rst file in the GitHub repository with the module type using the `update_index_rst` function.

The tasks are connected in a linear dependency chain, where each task depends on the successful completion of the previous task.

DAG Configuration

The DAG is configured with the following settings:

- **DAG ID:** thethree_reqif_to_rst
- **Owner:** The Three
- **Depends on Past:** False
- **Retries:** None
- **Start Date:** January 1, 2023
- **Schedule:** None (manually triggered)
- **Catchup:** False
- **Tags:** reqif

, rst, TheThree, Bosch

The DAG is designed to be manually triggered and does not have a fixed schedule. It does not catch up on any missed runs.

To execute the DAG, run the `thethree_reqif_to_rst` task in Airflow. Make sure to provide the following configuration parameters as part of the DAG run:

- `reqif_file_path`: The file path of the ReqIF file to be processed.
- `mapping_config_file_path`: The file path of the mapping configuration file.
- `github_config_file_path`: The file path of the GitHub configuration file.

Note: The path to the files must start from root directory (`/`)

Please ensure that the necessary dependencies and modules are installed in the Airflow environment to successfully execute the DAG.

Docker Compose for Airflow

This Docker Compose configuration allows you to set up and run Airflow, a platform to programmatically author, schedule, and monitor workflows. It includes the necessary services and dependencies to execute the "thethree_reqif_to_rst" DAG.

Prerequisites

Before running the Docker Compose setup, ensure that you have Docker and Docker Compose installed on your system.

Configuration

To trigger the Airflow DAG, you need to provide the following configuration parameters as a JSON object. For example, there are 3 files `requirements.reqif`, `mapping_config.yml` and `github_config.yml` in the `config` file. When run the docker container, the path to those files are `/opt/airflow/config/*`

```
{
  "reqif_file_path": "/opt/airflow/config/requirements.reqif",
  "mapping_config_file_path": "/opt/airflow/config/mapping_config.yml",
  "github_config_file_path": "/opt/airflow/config/github_config.yml"
}
```

Make sure to replace the file paths with the appropriate locations of your ReqIF file, mapping configuration file, and GitHub configuration file.

Initialization

Before running Airflow for the first time, you need to initialize the environment. Run the following command to initialize the Docker services:

```
echo -e "AIRFLOW_UID=$(id -u)" > .env
```

This command sets up the necessary environment variables and initializes the Airflow database.

Running Airflow

Once the initialization is complete, you can start running Airflow with the following command:

```
docker-compose up
```

This command starts all the Airflow services defined in the Docker Compose configuration. Airflow will start executing the "thethree_reqif_to_rst" DAG based on the triggers or schedules you have set.

Clean Up

To stop and remove all the Docker services created for Airflow, you can use the following command:

```
docker-compose down --volumes --rmi all
```

This command stops the services and removes the associated volumes and images.

Note: Be cautious when using the `--volumes` and `--rmi all` flags, as they remove all volumes and images associated with the Airflow setup.

Remember to update the file paths and configuration parameters according to your specific setup before running the Docker Compose commands.