DIGIVATELABS

ENABLING DIGITAL ENTERPRISES

Cron Report Automation with Databricks Jobs

Version 1.0

DIGIVATE LABS PVT. LTD.
reachus@digivatelabs.com

Table of Contents

1. Objectives

- To automate the execution of scheduled reports using Databricks notebooks.
- To create Databricks jobs programmatically using Databricks API with Quartz-based scheduling.
- Execute the jobs as per schedule and save the report outputs to external location (cloud storage).
- To handle email delivery of generated reports with configurable parameters (like frequency, attachments, and mail body).

2. Pre-requisites

- Access to Databricks Workspace and Job API token.
- API Token's owner (Service principal) should have permissions on SQL files, appropriate Unity Catalog configurations for executing SQL scripts, 'attach' permission on compute, permissions(select/modify) on external location.
- Secrets configured for email credentials in Databricks secret scope.
- Currently. serverless mode is not supported for job creation via the Databricks Jobs API, Specific cluster configuration or an existing cluster ID must be provided instead.

3. Workflow Overview

3.1. Driver Notebook

- Reads cron_reports.csv into a Spark DataFrame.
- Iterates over each row and extracts relevant fields (e.g., report_name, frequency, report_owner, cron_tab, etc.).
- Invokes second notebook job orchestration notebook with parameters for each report via dbutils.notebook.run.
- Collects job metadata like job_id and logs failures if job creation fails.

3.2. Job Orchestration Notebook

- Accepts parameters as widgets: report details, frequency, cron expression, email preferences, etc.
- Converts the provided Unix cron expression to Quartz format for Databricks job scheduling.
- Constructs a JSON payload to define a new Databricks Job API.
- Sets up schedule, notebook task (generate report notebook), and job permissions- owner and access control to job, then triggers job creation.
- Returns job metadata (job_id, job_name) to the calling notebook.

3.3. Generate report notebook

- Executes the actual SQL logic to generate the report output.
- Handles:
  - Executing main SQL - daily/weekly/monthly and optional mailbody.sql
  - Writing results as CSV or HTML based on configuration.
  - Generating email content dynamically, appending disclaimers and ownership info
- Sends an email with:
  - CSV attachment (if attach=True), HTML body content (if attach=False)
  - HTML body content from mailbody.sql output (if mail_body=True).

4. Module-wise Description: Cron Report Creation and Execution

- Architecture and workflow for the automated generation and distribution of cron-based reports.

SQL File Location: All SQL scripts used for report generation are stored in the following Databricks Workspace location.

https://dbc-09c95fb5-3672.cloud.databricks.com/browse/folders/925061462551645?o=3389081040467503

- The creation and configuration of scheduled Databricks Jobs are managed through a series of dedicated notebooks.

https://dbc-09c95fb5-3672.cloud.databricks.com/browse/folders/925061462553317?o=3389081040467503

The process involves three notebooks:

- **report_runner_notebook -I**

https://dbc-09c95fb5-3672.cloud.databricks.com/editor/notebooks/925061462554561?o=3389081040467503#command/5883193242704479

Functionality: This notebook is responsible for reading input parameters from a CSV file. It then uses these inputs to orchestrate and potentially create multiple job definitions. Its primary output is job IDs, signifying the successful definition of report jobs.

- **report_orchestration_notebook -II**

https://dbc-09c95fb5-3672.cloud.databricks.com/editor/notebooks/925061462554597?o=3389081040467503#command/925061462554598

  - **Functionality:** This notebook receives inputs (likely parameters or configurations) from report_runner_notebook -I. It then uses these inputs to construct and submit the final Databricks Job creation API calls.
  - **Job Naming Convention:**

        - Jobs created by this process adhere to the following naming convention: Job_{report_name}_{report_folder_name}_{frequency}_test_run
        - **Example Job Name:** Job_NM_report_D1_logistics_vertica_daily
    - **Job Parameters:** Each created job is configured with the following parameters, passed as base parameters to the executing notebook:
        - **cron_tab**: UNIX-style cron expression for scheduling
        - **report_name**: A specific name identifying the report (e.g., "NM_Report").
        - **report_folder**: The logical folder or category of the report (e.g., "logistics_vertica").
        - **catalog_name**: The Databricks Unity Catalog name used for data access.
        - **frequency**: The reporting frequency (e.g., "daily", "weekly", "monthly").
        - **report_owner**: Name/email of the report owner.
        - **attach**: (True/False) Whether to attach the report file to email.
        - **mail_body**: (True/False) Whether to include the report in the email body
        - **to_list** : Email recipients (To)
        - **cc_list**: Email recipients (CC)

- **generate_report_notebook -III**

  https://dbc-09c95fb5-3672.cloud.databricks.com/editor/notebooks/689986750951450?o=3389081040467503#command/689986750951451

    - **Functionality:** This notebook is the core execution component, triggered by scheduled Databricks Jobs. It generates, formats, and distributes reports via email.
    - **Key Responsibilities:**
        - **SQL Execution:** Runs primary SQL logic (e.g., daily.sql) and optional mailbody.sql for dynamic content.
        - **Output Handling:** Saves results to S3 (s3://sd-databricks-prd-job-logs/dwh/reports), formatted as CSV or HTML.
        - **Email Distribution:** Sends email to to_list/cc_list with a CSV attachment (if attach=True) or HTML body (if attach=False), optionally using mailbody.sql output for content (if mail_body=True).

5. Results and Validation

- Once created, the job is scheduled using Quartz expression.

- Databricks will automatically trigger the generate_report_notebook notebook at the specified time.
- Output is written to S3 location (s3://sd-databricks-prd-job-logs/dwh/reports/generate_report).
- Email is sent to recipients as configured with proper formatting, attachment, and fallback handling.
- Proper error handling ensures notebook exits gracefully if required inputs or outputs are missing.
- Job ID and name logs provide traceability.