# Workflow Integration

## Overview

1. Calling notebook from another notebook
2. Job (workflow)

    a) Every ETL task followed by task (No dependencies)

    b) Every unique task path followed by task

    c) All ETL job tasks followed by a single task (task dependencies)

3. Retrieve tasks info from either

    3.1 metadata_execution table

    3.2 databricks API

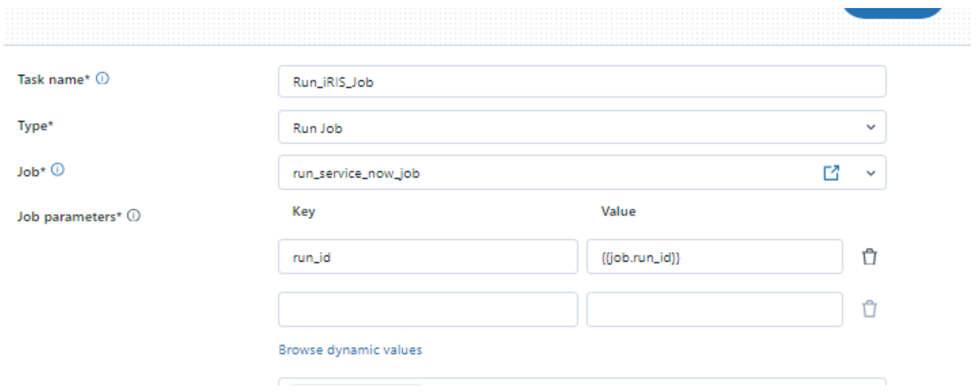
4. Obtaining Access Token to use the API Services
5. Create ServiceNow incident

## Calling notebook from another notebook

- Pass parameters (job_id, task_run_id, job_name)
- Usage: dbutils.notebook.run ("notebook-name", 60, {"argument": "data", "argument2": "data2", ...})

## Triggering notebook from ETL job (workflow)

Get job_id, task_run_id, job_name from workflow using through job parameters.



| | | |
|---|---|---|
| Task name* | Run_iRIS_Job | |
| Type* | Run Job | |
| Job* | run_service_now_job | |
| Job parameters* | Key | Value |
| | run_id | {{job.run_id}} |
| | | |

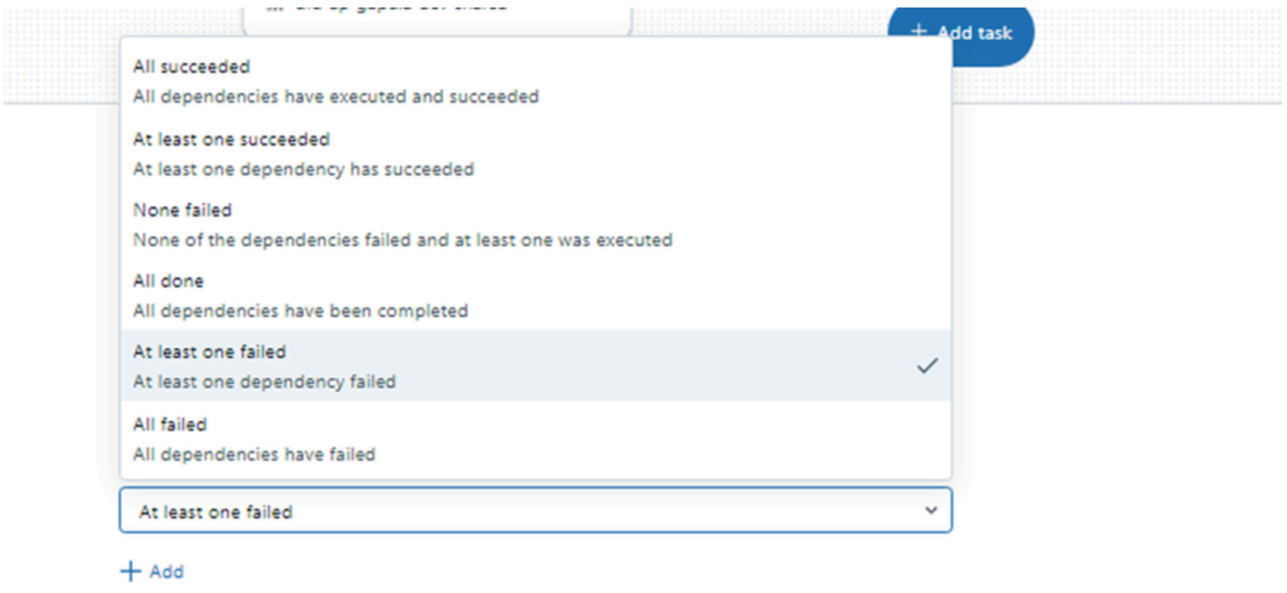
Browse dynamic values

We are passing these as parameters from the workflow task and retrieve these in the databricks notebook using widgets.

Usage: job_id=dbutils.widgets.get('job_id')

To make sure the notebook is successfully triggered, the RUN IF dependencies and task should be specified properly in the workflow.

## RUN IF dependencies.



All succeeded
All dependencies have executed and succeeded

At least one succeeded
At least one dependency has succeeded

None failed
None of the dependencies failed and at least one was executed

All done
All dependencies have been completed

At least one failed ✓
At least one dependency failed

All failed
All dependencies have failed

At least one failed

### Every ETL task followed by task (No dependencies)

### Every unique task path followed by task.

### All ETL job tasks followed by a single task (task dependencies)

## Retrieve failed tasks from metadata table.

- retrieve the info of all the failed tasks from audit metadata table "metadata_execution" and create a ServiceNow incident for each task if the incident not exists in 'metadata__incidents' table

select JOB_ID, TASK_NAME, RUN_ID, JOB_STATUS, EXCEPTION from 'metadata_incidents'

The information required to population description fields can be directly retrieved from this table.

## Retrieve failed task info from databricks API.

Can use either task_run_id or job run_id to get info from the databricks API.

Databricks instance URL and access token is required to retrieve tasks information from databricks API.

databricks_url="https://xxxxxxxxxxxxx.azuredatabricks.net/api/2.1/jobs/runs/get/"

The databricks access token is generated only once and used until it expires, but the ServiceNow access token is generated every time the script is executed.

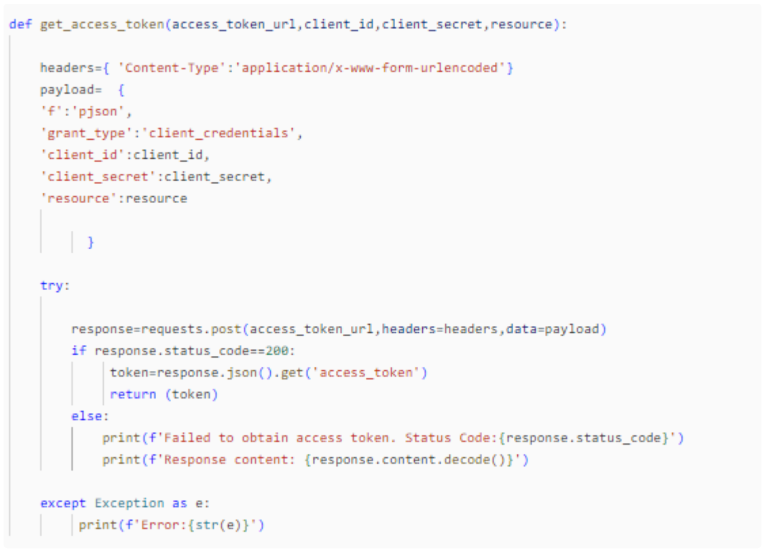
Note: databricks access_token will be valid for up to 400 days once generated.

Failed task info:

[{'task_run_id': 254199708690745, 'task_key': 'practice_task1', 'run_page_url'','base_parameters': {'var1': '5', 'var2': '6'}, 'source': 'WORKSPACE'}, 'result_state': 'FAILED', 'state_message': 'Unable to access the notebook "/path notebook". Either it does not exist, or the identity used to run this job, lacks the required permissions.'}]

Using the info from the above response, we populate short_description and description fields which are mandatory to create an incident. Short_description will have workflow_name, task_key, result_state while the additional info is in description field.

## Obtaining Access Token to use the API Services



```
def get_access_token(access_token_url,client_id,client_secret,resource):

    headers={ 'Content-Type':'application/x-www-form-urlencoded'}
    payload=  {
    'f':'pjson',
    'grant_type':'client_credentials',
    'client_id':client_id,
    'client_secret':client_secret,
    'resource':resource

        }

    try:

        response=requests.post(access_token_url,headers=headers,data=payload)
        if response.status_code==200:
             token=response.json().get('access_token')
             return (token)
        else:
            print(f'Failed to obtain access token. Status Code:{response.status_code}')
            print(f'Response content: {response.content.decode()}')

    except Exception as e:
         print(f'Error:{str(e)}')
```

## Creating ServiceNow incident

creating incident with necessary information (urgency, impact, cmdb_ci, caller_id, location, assignment_group, short_description and description)

HTTP Verb: POST

URL: https://xxxxxxxxxxxxxxxx/apg-001-servicenow/v1/now/table/incident/

Header Key/Value Pairs:

- Authorization: "Bearer access-token"
- Accept: application/json

Body (raw json)

```
{
   "incident_object":{
      "urgency":"3",
      "short_description":"test",
      "impact":"2",
      "cmdb_ci":
      "caller_id:
      "assignment_group":
      "description":"test purpose"
   }
}
```

If the incident is successfully created, you will see the http code 200.

```
create_servicenow_incident(incident_url,access_token_url,client_id,client_secret,resource,urgency,short_description,impact,cmdb_ci,caller_id,location,assignment_group,description)
```

```
<Response [200]>
{'result': {'status_code': '200',
  'status_message': 'Success',
  'return_response': {'number': 'INC000027970958',
   'u_state': 'Assigned',
   'state': 'Assigned'}}}
Command took 3.21 seconds
```