Migrate Apache Oozie Workflows to Airflow and Run with Amazon EMR

Dipankar Ghosal

Airflow Summit
Let’s flow together
September 19-21, 2023, Toronto, Canada
About me

• Principal Architect at Amazon Web Services
• Advisory and Transformation
• Excited for the football season
• Still like to code 😊

Linked In - www.linkedin.com/in/dipankarghosal
Motivation...need

Computing workloads are moving to the cloud to achieve greater business agility, access to modern technology, and reduce operational costs.

Whether it’s Data First approach or Workload First migration strategy
  - Decouple Storage and Compute
  - Decouple Workload management
Apache Oozie

- Apache Oozie is a workflow management system to manage Hadoop jobs.
- It is deeply integrated with the rest of Hadoop stack supporting a number of Hadoop jobs out-of-the-box.
- Workflow is expressed as XML and consists of two types of nodes: control and action.
- Scalable, reliable and extensible system.
Apache Airflow

- Apache Airflow is a platform to programatically author, schedule, and monitor workflows.
- Workflows are authored as directed acyclic graphs (DAGs), and can be configured as code - using Python 3.x.
- The rich user interface makes it easy to visualize pipelines running in production, monitor progress, and troubleshoot issues when needed.
Workflow vs DAG

Oozie workflow

This is a very simple Oozie workflow that performs a hive action. Pay attention to the following XML elements:

1. start
2. action
3. kill
4. end

```
<workflow-app xmlns="url:oozie:workflow:1.0" name="hive-wf">
  <start to="hive-script"/>
  <action name="hive-script">
    <hive xmlns="url:oozie:hive-action:1.0">
      <resource-manager>${resourceManager}</resource-manager>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.job.queue.name</name>
          <value>${queueName}</value>
        </property>
      </configuration>
      <script q="script"/>
      <param INPUT="s3://airflow-summit-demo/data/input"/></param>
      <param OUTPUT="s3://airflow-summit-demo/data/output/output-data"/></param>
    </hive>
    <ok to="hive-query"/>
    <error to="fail"/>
  </action>
  <action name="hive-query">
    <hive xmlns="url:oozie:hive-action:1.0">
      <resource-manager>${resourceManager}</resource-manager>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.job.queue.name</name>
          <value>${queueName}</value>
        </property>
      </configuration>
      <query>
        DROP TABLE IF EXISTS test_query;
      </query>
      <hive>
        <ok to="end"/>
        <error to="fail"/>
      </hive>
    </action>
    <kill name="fail">
      <message>
        Hive action failed, error message[${wf:errorMessage[wf:lastErrorNode()]}]
      </message>
    </kill>
    <end name="end"/>
  </workflow-app>
```
Workflow vs DAG

DAG code

This is an Airflow DAG equivalent of Oozie Workflow. Important elements to note are:

1. Task Map
2. Operator
3. Relationship
o2a- Converter Design

 workflow XML → [nodes] where node: name, attributes, child elements, etc.

 [nodes] → workflow object:
 - dependencies
 - relationship
 - airflow-nodes

 workflow object → dag.py

We wanted to make it work with Amazon EMR

Run big data applications and petabyte-scale data analytics faster, and at less than half the cost of on-premises solutions.

To submit work, you can add steps, or you can interactively submit Hadoop jobs to the primary node.

Operator Jinja Templates

Dataproc

```python
{{ task_id | to_var }} =
    dataproc_operator.DataProcPigOperator(  
        task_id={{ task_id | to_python }},  
        trigger_rule={{ trigger_rule | to_python }},  
        query_uri='%s/%s' % (CONFIG['gcp_uri_prefix'], {{ script_file_name | to_python }}),  
        variables={{ params_dict | to_python }},  
        dataproc_pig_properties={{ props_macro.props(action_node_properties=action_node_properties, xml_escaped=True) }},  
        cluster_name=CONFIG['dataproc_cluster'],  
        gcp_conn_id=CONFIG['gcp_conn_id'],  
        region=CONFIG['gcp_region'],  
        dataproc_job_id={{ task_id | to_python }},  
        params={{ props_macro.props(action_node_properties=action_node_properties) }},
    )
```

EMR

```python
{{ task_id | to_var }} =
    EmrSubmitAndMonitorStepOperator(  
        task_id={{ task_id | to_python }},  
        steps=[{'Name': {{ task_id | to_python }},'ActionOnFailure': 'CONTINUE','HadoopJarStep':  
            {'Jar': 'command-runner.jar','Args': {{ params_dict | to_python }},},}],  
        job_flow_id=CONFIG['emr_cluster'],  
        aws_conn_id=CONFIG['aws_conn_id'],
    )
```

Pig action template
Single Operator for EMR

```java
sub_wf_run_begin_oozie = EmrSubmitAndMonitorStepOperator(
  task_id="sub_wf-run_begin_oozie",
  trigger_rule="one_success",
  steps=[
  {
    "Name": "sub_wf-run_begin_oozie",
    "ActionOnFailure": "CONTINUE",
    "HadoopJarStep": {
      "Jar": "command-runner.jar",
      "Args": ["/bin/bash",
                "-c",
                "{}/format(shlex.quote("hdfs dfs -mkdir -p '${demo_offer_outreach_running}'"))]
    },
    },
  ],
  job_flow_id=CONFIG["emr_cluster"],
  aws_conn_id=CONFIG["aws_conn_id"],
}
}
```

```java
run_demo_offer_outreach_stage = EmrSubmitAndMonitorStepOperator(
  task_id="run_demo_offer_outreach_stage",
  steps=[
  {
    "Name": "run_demo_offer_outreach_stage",
    "ActionOnFailure": "CONTINUE",
    "HadoopJarStep": {
      "Jar": "command-runner.jar",
      "Args": [
        "hive-script",
        "--run-hive-script",
        "--args",
        "-f",
        "{${demo_offer_outreach_stage_ddl}}",
      ],
      "job_flow_id=CONFIG["emr_cluster"],
      aws_conn_id=CONFIG["aws_conn_id"],
    }
  },
]
```
## Demo Recap

- Oozie to Airflow converter repo: [https://github.com/dgghosalaws/oozie-to-airflow-emr](https://github.com/dgghosalaws/oozie-to-airflow-emr)
- Successfully converted a representative complex Oozie bundle workflow to Airflow DAG with Amazon EMR compatibility
  - Supports Coordinator and bundle parsing
  - Single Airflow EMR Operator
- No post-conversion modification and runs well out of the box
Questions?

Other Articles

Building complex workflows with Amazon MWAA, AWS Step Functions, AWS Glue, and Amazon EMR