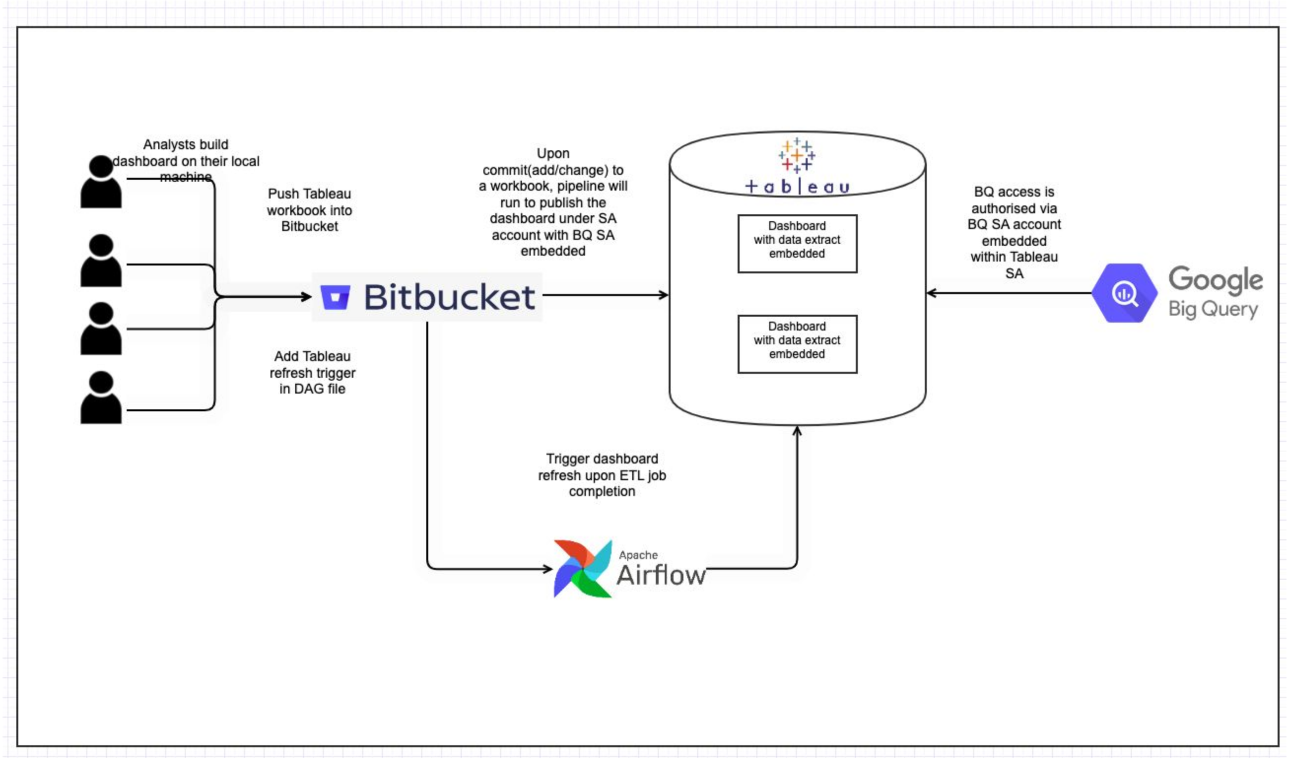
Airflow Operators

- Overview - Cloud Composer & Airflow
- DAG overview
- BashOperator
- BranchPythonOperator
- ShortCircuitOperator
- BigQueryExecuteQueryOperator
- SqlSensor
- Other GCP Operators

Overview - Cloud Composer & Airflow



Analysts build dashboard on their local machine
Push Tableau workbook into Bitbucket
Bitbucket
Add Tableau refresh trigger in DAG file
Upon commit(add/change) to a workbook, pipeline will run to publish the dashboard under SA account with BQ SA embedded
tableau
Dashboard with data extract embedded
Dashboard with data extract embedded
BQ access is authorised via BQ SA account embedded within Tableau SA
Google Big Query
Trigger dashboard refresh upon ETL job completion
Apache Airflow

For more information contact Ye Wang: ywang6@woolworths.com.au

DAG overview

```
default args = {
   'owner': 'BI',
   'depends_on_past': False,
   'start_date': datetime(2021, 3, 16, tzinfo=local_tz),
   'email': ['wx-caaids-bi@woolworths.com.au'],
   'email_on_failure': True,
   'email_on_retry': True,
   'retries': 0,
   'retry_delay': timedelta(minutes=5),
   # 'queue': 'bash queue',
   # 'pool': 'backfill',
   # 'priority weight': 10,
   # 'end date': datetime(2021, 3, 14),
   # 'wait_for_downstream': False,
   # 'dag': dag,
   # 'sla': timedelta(hours=2),
   # 'execution timeout': timedelta(seconds=300),
   # 'on_failure_callback': some_function,
   # 'on success callback': some other function,
   # 'on retry callback': another function,
   # 'sla_miss_callback': yet_another_function,
   # 'trigger_rule': 'all_success'
}
dag = DAG(
   'daily-jobs',
   default args=default args,
   description='Daily Jobs run at 3am in the morning',
   schedule interval='0 5 * * *',
   catchup=False,
   tags=['xdiv-handshake-ownerYW', 'xdiv-bas-ownerYW', 'xdvi-push-notification-wow-pdf-ownerSS', 'xdiv-rewards-app-ownerDC',
'xdiv-push-notification-dash-ownerSS'
      ],
)
```

BashOperator

Use the **`BashOperator`** to execute commands in a Bash shell.

Example 1:

```
task_sm_edr_pulse_report = BashOperator(
   task_id='sm-edr-pulse-report',
   bash_command='''python
/home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/run_bq_with_email_notificaiton.py \
                   -p "[WDP] EDR Pulse Report Monday" \
                   -s
"/home/airflow/gcs/data/etl-scripts/sm-others/sm-edr-pulse-report/bq_super_scan_rate_run_monday.sql" \
                ''',
   trigger_rule=TriggerRule.ALL_DONE,   # requires direct parent tasks succeed to start
   dag=dag,
)
```

Example 2:

```
task_sleep_till_830am = BashOperator(
   task_id="sleep_till_830am",
   dag=dag,
   retries=5,
   email_on_retry=False,
   retry_delay=timedelta(seconds=300),
   bash_command='''
       chmod +x /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/sleep_till_time.sh
       /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/sleep_till_time.sh "8:30"
   '''
)
```

BashOperator - templating

You can use Jinja templates to parameterize the `bash_command` argument.

Example :

```
templated command = '''
python /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/run_bq_with_email_notificaiton.py \
   -p "WDP - Super BAS back fill for week {{ macros.ds_add(execution_date.in_timezone('Australia/Sydney').to_date_string(), 6) }}" \
   -s "/home/airflow/gcs/data/etl-scripts/bas-tables/bas_super.sql" \
   -r "{{ macros.ds_add(execution_date.in_timezone('Australia/Sydney').to_date_string(), 6) }}" \
   -e "ywang6@woolworths.com.au" \
'''

t1 = BashOperator(
   task id='step1-run-super-bas-backfill',
   bash command=templated command,
   trigger rule=TriggerRule.ALL_DONE,   # requires direct parent tasks done(fail or succeed) to start
   dag=dag,
)
```

```
task_scan_and_go_branch >> [task_mon_sleep_till_7am,task_tue_to_sun_sleep_till_12m]
```

BranchPythonOperator - flow control

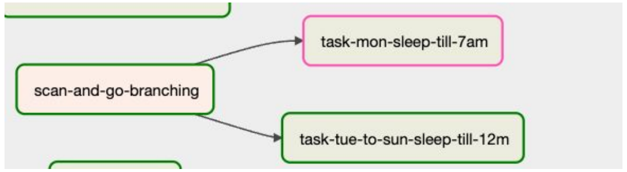
Sometimes you need a workflow to branch, or only go down a certain path based on an arbitrary condition which is typically related to something that happened in an upstream task. One way to do this is by using the `BranchPythonOperator`.

Example :

```python
def branch_day_of_week():
   bq_client = bq.Client()
   sql = '''
           select WeekDayNumber as weekday
           from  `gcp-wow-ent-im-wowx-cust-prod.adp_wowx_dm_masterdata_view.dim_date_v`
           where CalendarDay = current_date("Australia/Sydney")
   '''
   query_result = bq_client.query(sql).result().to_dataframe()
   dow = query_result[ "weekday"][0]
   print("dow: "+str(dow))

   if int(dow) == 1:
       return('task-mon-sleep-till-7am')
   else:
       return('task-tue-to-sun-sleep-till-12m')

task_scan_and_go_branch = BranchPythonOperator(
       task_id='scan-and-go-branching',
       python_callable =branch_day_of_week,
       dag=dag,
       provide_context =True
)
```

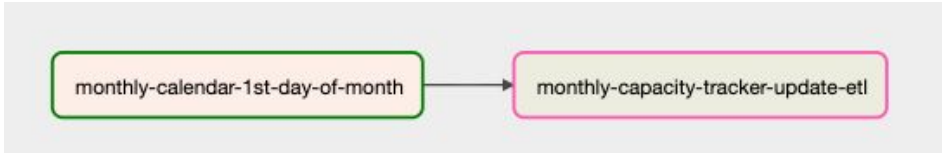
BranchPythonOperator - flow control (continue)

```
task_mon_sleep_till_7am = BashOperator(
   task_id='task-mon-sleep-till-7am',
   retries=5,
   bash_command='''
       chmod +x /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/sleep_till_time.sh
       /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/sleep_till_time.sh "7:00"
   ''',
   dag=dag,
)

task_tue_to_sun_sleep_till_12m = BashOperator(
   task_id='task-tue-to-sun-sleep-till-12m',
   retries=5,
   bash_command='''
       chmod +x /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/sleep_till_time.sh
       /home/airflow/gcs/data/etl-scripts/bau-etl-automation-tool/sleep_till_time.sh "12:00"
   ''',
   dag=dag,
)
task_scan_and_go_branch >> [task_mon_sleep_till_7am,task_tue_to_sun_sleep_till_12m]
```



ShortCircuitOperator - flow control

Allows a pipeline to continue based on the result of a `python_callable`


monthly-calendar-1st-day-of-month
monthly-capacity-tracker-update-etl

Example :

```
def day_of_month(dom,month_type):
    bq_client = bq.Client()
    sql = '''
            select DayNumber as day_of_cal_mt
                , DATE_DIFF(CalendarDay,FiscalPeriodStartDate,DAY) + 1 as day_of_fin_mt
            from  `gcp-wow-ent-im-wowx-cust-prod.adp_wowx_dm_masterdata_view.dim_date_v`
            where CalendarDay = current_date("Australia/Sydney")
    '''
    query_result = bq_client.query(sql).result().to_dataframe()
    dom_cal = query_result[ "day_of_cal_mt"][0]
    dom_fin = query_result[ "day_of_fin_mt"][0]
    print("dom_cal: "+str(dom_cal))
    print("dom_fin: "+str(dom_fin))
    if month_type.lower() ==  'cal':
        return str(dom_cal) ==  str(dom)
    elif month_type.lower() ==  'fin':
        return str(dom_fin) ==  str(dom)
    else:
        return False

task_run_on_1st_calendar_day_of_month = ShortCircuitOperator(
    task_id='monthly-calendar-1st-day-of-month',
    python_callable =day_of_month,
    op_kwargs={'dom':'1','month_type':'cal'},
    dag=dag,
)
task_run_on_1st_calendar_day_of_month >> task_capacity_tracker_update
```

BigQueryExecuteQueryOperator - Run BQ query

Executes BigQuery SQL queries in a specific BigQuery database.

Example :

```
task trigger push notification wow dash cube = bigquery.BigQueryExecuteQueryOperator(
   task_id='trigger-push-notification-wow-dash-mstr-cube',
   sql="""
       insert into `gcp-wow-rwds-ai-data-prod.loyalty_bi_analytics.mstr_cubes_to_be_triggered` values

(GENERATE_UUID(),'C:\\\\Command_Manager\\\\push_notification_wow_dash.scp',current_datetime("Australia/S
ydney"),null,false,'waiting to be triggered','Prod')
   """,
   use legacy sql=False,
   trigger rule=TriggerRule.ALL_SUCCESS,  # requires direct parent tasks done(fail or succeed) to start
   dag=dag,
)
```

SqlSensor

Runs a sql statement repeatedly until a criteria is met.

Example :

```
check_super_sql = '''
   select min(case when coalesce(supers,'') = 'Y' and SupersLoadDttm is not null then true else false
end) as status
   from `gcp-wow-ent-im-wowx-cust-prod.adp_wowx_dm_masterdata_view.dim_date_v` dd
   left join
`gcp-wow-ent-im-wowx-cust-prod.adp_wowx_dm_integrated_sales_view.sales_summary_load_status_v` ssls on
dd.CalendarDay = ssls.TXNStartDate
   where CalendarDay between current_date("Australia/Sydney") - 8 and current_date("Australia/Sydney") -
1
'''

task_check_handshake_super = BigQuerySqlSensor(
   conn_id='bigquery_default',
   sql=check_super_sql,
   fail_on_empty=False,
   task_id='check-handshake-table-super',
   poke_interval=600,  # retry every 10m
   timeout=36000,  # fail after 10h
   trigger_rule=TriggerRule.ALL_DONE,  # requires direct parent tasks done(fail or succeed) to start
   dag=dag,
)
```

Other GCP operators

GoogleCloudStorageDeleteOperator

```python
task_sfmc_data_extract_gcs_delete = GoogleCloudStorageDeleteOperator(
   task_id='sfmc-data-extract-step1-clear-gcs',
   bucket_name='us-central1-wx-caaids-bi-wo-776d68c8-bucket',
   prefix='data/staging_files/to-be-transferred-to-sfmc-sftp/sfmc-data-uplift/',
   # google_cloud_storage_conn_id='google_cloud_storage_default',
   dag=dag,
)
```

GoogleCloudStorageToGoogleCloudStorageOperator

```python
task_sfmc_data_extract_archive = GoogleCloudStorageToGoogleCloudStorageOperator(
   task_id='sfmc-data-extract-step5-archive',
   source_bucket='us-central1-wx-caaids-bi-wo-776d68c8-bucket',
   source_object='data/staging_files/to-be-transferred-to-sfmc-sftp/sfmc-data-uplift/*.gz',
   destination_bucket='us-central1-wx-caaids-bi-wo-776d68c8-bucket',
   destination_object='data/staging_files/to-be-transferred-to-sfmc-sftp/sfmc-data-uplift-archive/',
   move_object=True,
   # google_cloud_storage_conn_id='google_cloud_storage_default',
   dag=dag,
)
```

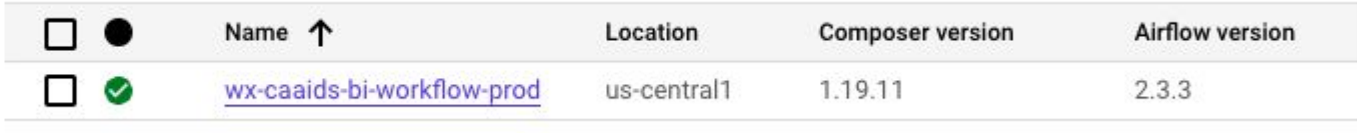
Reference

Airflow Document: https://airflow.apache.org/docs/apache-airflow/2.3.3/index.html

Cloud Composer: https://cloud.google.com/composer/docs/concepts/overview

Re-usable Tool:
https://woolworthsdigital.atlassian.net/wiki/spaces/WXC/pages/1614677145/Miscellaneous+Topics

The version we are currently using:



☐	●	Name ↑	Location	Composer version	Airflow version
☐	✓	wx-caaids-bi-workflow-prod	us-central1	1.19.11	2.3.3