Date: 1st Aug 2019

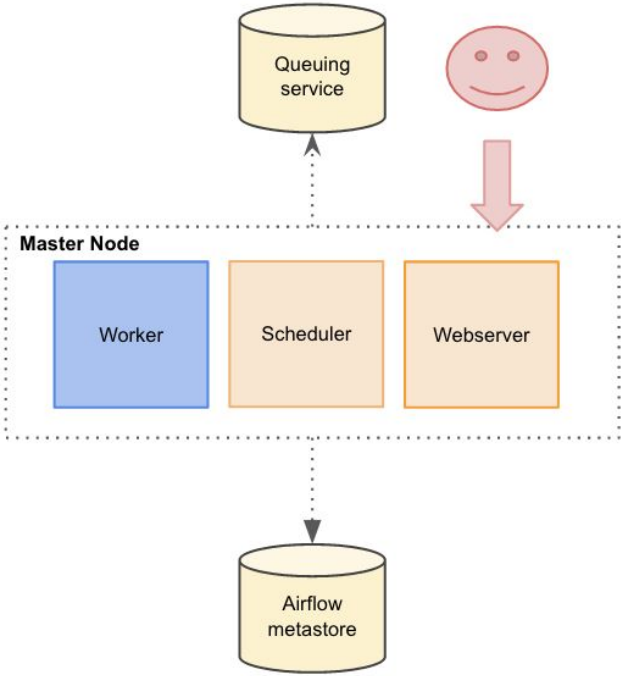
Apache-Airflow-1.10.3

What is Airflow?

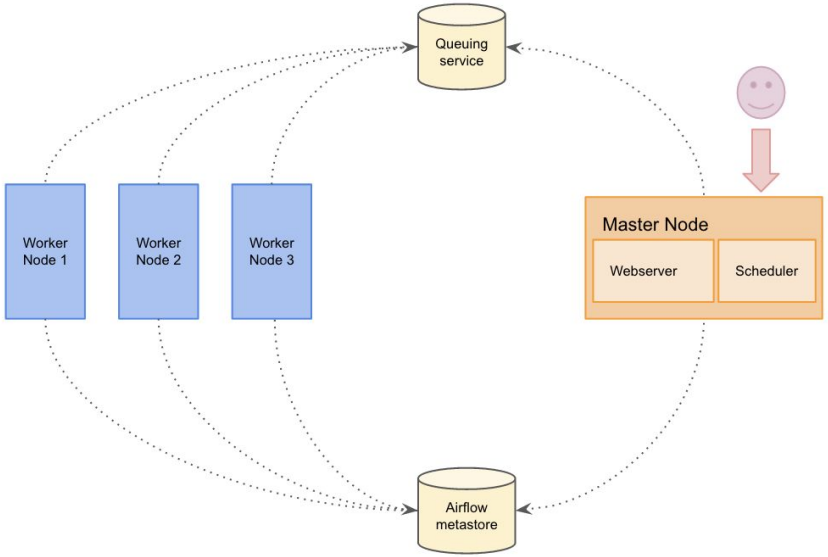
It is a platform to programatically author, schedule and monitor workflow .It uses workflows made of directed acyclic graphs.

- Airflow is a sequence of tasks
- Started on schedule or triggered by event
- Handle big data processing pipeline

Airflow Architecture



Single node



Multiple node

Main blocks of Airflow

- Web-server
- Scheduler
- Executor
- Worker

Webserver

- Its a one kind of user interface …
- Accepts HTTP request and allow user to interact with it.
- Act on activities of DAG like pause, unpause, trigger dag, view running dags Restart failed dag
- Monitor the currenly running dags

Scheduler and Executor

- Monitor and schedule all dags are that are running

Executors used in airflow are:

1. **Sequential executor**: Runs one task at a time
2. **Local executor**:Executes the tasks locally in parallel
3. **Celery executor**:Distribute the execution of task to multiple worker nodes

Scheduling Intervals

preset	meaning	cron
`None`	Don't schedule, use for exclusively "externally triggered" DAGs	
`@once`	Schedule once and only once	
`@hourly`	Run once an hour at the beginning of the hour	`0 * * * *`
`@daily`	Run once a day at midnight	`0 0 * * *`
`@weekly`	Run once a week at midnight on Sunday morning	`0 0 * * 0`
`@monthly`	Run once a month at midnight of the first day of the month	`0 0 1 * *`
`@yearly`	Run once a year at midnight of January 1	`0 0 1 1 *`

Default arguments

```
default_args = {
    'owner': 'airflow',
    'start_date': airflow.utils.dates.days_ago(2),
    # 'end_date': datetime(2018, 12, 30),
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    # If a task fails, retry it once after waiting
    # at least 5 minutes
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    }
```

Instantiate a dag

```
dag = DAG(
    'tutorial',
    default_args=default_args,
    description='A simple tutorial DAG',
    # Continue to run DAG once per day
    schedule_interval=timedelta(days=1),
)
```

Tasks

This are the functions/ services that we want to execute in some sequence

```
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag,
)

t2 = BashOperator(
    task_id='sleep',
    depends_on_past=False,
    bash_command='sleep 5',
    dag=dag,
)

t3 = BashOperator(
    task_id='templated',
    depends_on_past=False,
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'},
    dag=dag,
)
```

Setting up dependencies

Setup dependencies that means order in which tasks should be executed

set_upstream is equivalent to <<
set_downstream is equivalent to >>

```
# This means that t2 will depend on t1
# running successfully to run.
t1.set_downstream(t2)

# similar to above where t3 will depend on t1
t3.set_upstream(t1)
```

Triggering Rules

all_success: (default) all parents have succeeded

all_failed: all parents are in a failed or upstream_failed state

all_done: all parents are done with their execution

one_failed: fires as soon as at least one parent has failed, it does not wait for all parents to be done

one_success: fires as soon as at least one parent succeeds, it does not wait for all parents to be done

Operators in airflow

- Sensors
  - HdfsSensor
  - NamedHivePartitionSensor
- Operators
  - BashOperator
  - PythonOperator
  - HiveOperator
  - BigQueryOperator
- Transfers
  - MySqlToHiveTransfer
  - S3ToRedShiftTransfer

User Interface

Airflow

localhost:8080/home

Airflow DAGs Security Browse Admin Docs About 2019-08-01, 04:19:50 UTC vishal kokane

DAGs

Search:

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
Off	Dag2	54 7 * * *	vishal@airflow				
On	Dag3.1	1 * * * *	vishal@airflow		2019-08-01 04:01		
Off	Dagviewuser	1 * * * *	viewuser		2019-07-31 09:01		
Off	example_bash_operator	0 0 * * *	airflow				
Off	example_branch_dop_operator_v3	*/1 * * * *	airflow				
Off	example_branch_operator	@daily	airflow				
Off	example_http_operator	1 day, 0:00:00	airflow				
Off	example_passing_params_via_test_command	*/1 * * * *	airflow				
Off	example_python_operator	None	airflow				
Off	example_short_circuit_operator	1 day, 0:00:00	airflow				
Off	example_skip_dag	1 day, 0:00:00	airflow				
Off	example_subdag_operator	@once	airflow				
Off	example_trigger_controller_dag	@once	airflow				
Off	example_trigger_target_dag	None	airflow				
Off	example_xcom	@once	airflow				
Off	latest_only	4:00:00	Airflow				
Off	latest_only_with_trigger	4:00:00	Airflow				

Tree view

Airflow

localhost:8080/tree?dag_id=Dag3.1

Airflow DAGs Security Browse Admin Docs About 2019-08-01, 04:25:27 UTC vishal kokane

On DAG: Dag3.1

schedule: 1 * * * *

Graph View | Tree View | Task Duration | Task Tries | Landing Times | Gantt | Details | Code | Trigger DAG | Refresh | Delete

Base date: 2019-08-01 04:01:36.054963- Number of runs: 25 Go

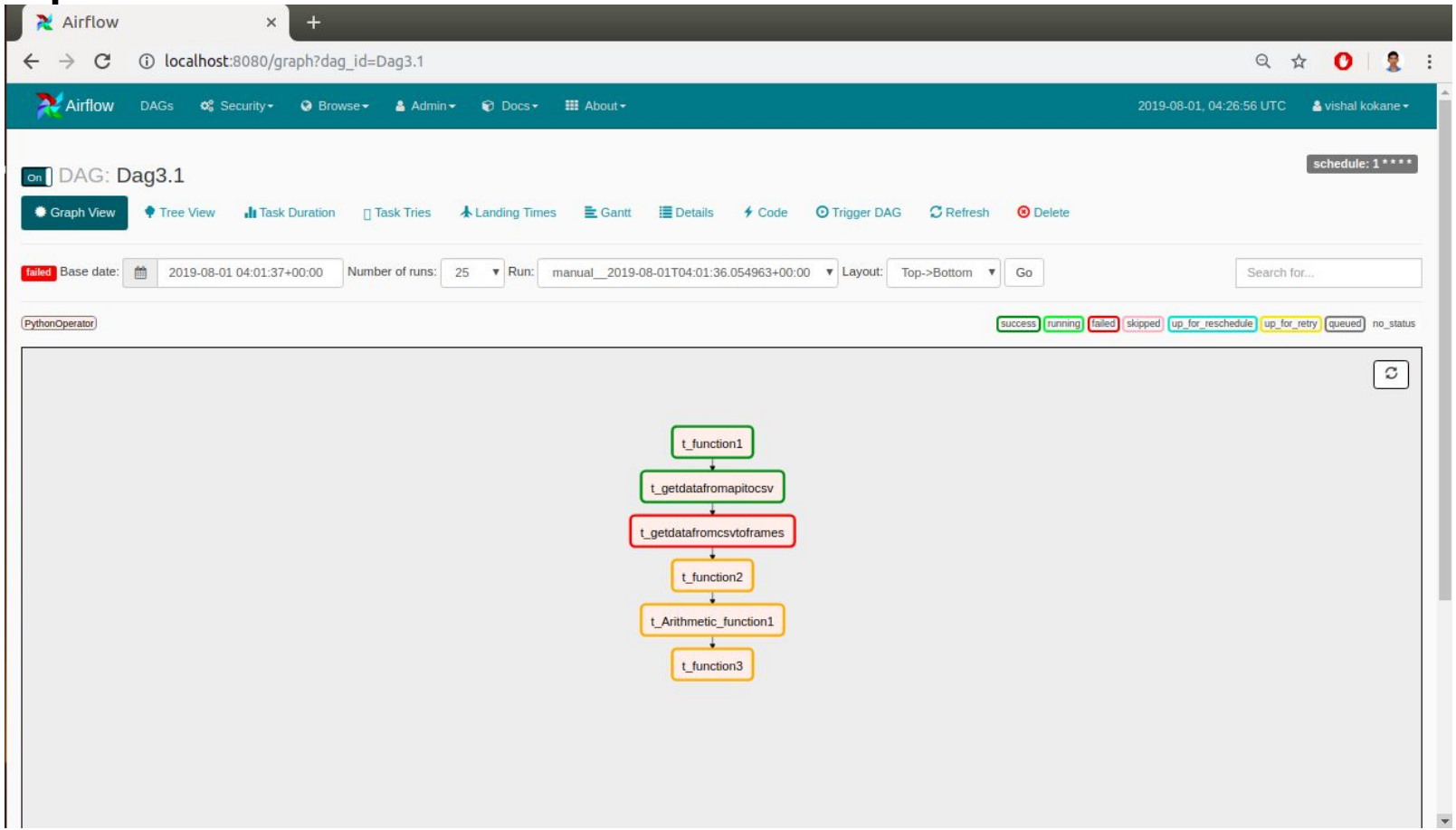
PythonOperator

success running failed skipped up_for_reschedule up_for_retry queued no_status

[DAG]
t_function3
t_Arithmetic_function1
t_function2
t_getdatafromcsvtoframes
t_getdatafromapitocsv
t_function1

12 PM August

Graph view


Airflow
localhost:8080/graph?dag_id=Dag3.1
Airflow
DAGs
Security
Browse
Admin
Docs
About
2019-08-01, 04:26:56 UTC
vishal kokane
On
DAG: Dag3.1
schedule: 1 * * * *
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Trigger DAG
Refresh
Delete
failed
Base date:
2019-08-01 04:01:37+00:00
Number of runs:
25
Run:
manual__2019-08-01T04:01:36.054963+00:00
Layout:
Top->Bottom
Go
Search for...
PythonOperator
success
running
failed
skipped
up_for_reschedule
up_for_retry
queued
no_status
t_function1
t_getdatafromapitocsv
t_getdatafromcsvtoframes
t_function2
t_Arithmetic_function1
t_function3

Dag view

1. **Dags**: This tab contain the all the dags contain in dag directory
2. **Schedule**: This column contain the scheduled time of each dag
3. **Owner**: Owner name of dag
4. **Recent tasks**:Status of last DAGs run
5. **Last run**: Date-time of dag that run last time
6. **Dag runs**: Status of all previous dag runs
7. **Links**:It contain the list of operations that will perform on DAG

Trigger dag	Gantt view
Tree view	Code view
Graph view	Logs
Task duration	Refresh
Task tries	Delete dag
Landing times	

Browse tab

Dag run: shows the status of each dag

Jobs: Shows status of each dag and description

Logs: Logs getting generated while running dag jobs with extra details

SLA Misses: It is a time by which task should get succeeded. An alert mail will send if sla get missed

Task instances: Shows details of each task in dag also if any task get fails we can schedule it again manually

Admin:

Configuration: This contain the configuration settings that are require to beginning of airflow

Connections: This contain the services that are working with airflow. We have to put the credentials of each service in their settings

Variables: It contains the key->value pair data that will require for dags this is stored on metadata database. No need to hardcode the parameters. Simply put key->value in json file and load it in variables.

XComs: This contains the data return by the task and want to forward it to another task or want to process it or use it.. It’s a communication between tasks

Pools: Can be used to **limit the execution parallelism** on arbitrary sets of tasks

Creating user in airflow

```
Command:
airflow create_user [-h]
              [-r ROLE]
              [-u USERNAME]
              [-e EMAIL]
              [-f FIRSTNAME]
              [-l LASTNAME]
              [-p PASSWORD]
              [--use_random_password]
```

Type of Roles: VLAC(View Level Access Control)

Role	Description
Admin	Has access to all the functions and views inside airflow. Read/write/grant access to users
Op	Has access to all views except User view
User	This role corresponds to regular user and will have access to all non-admin and non-data profiling views.
Viewer	Only has read-only access to all the non-admin and non-data profiling Airflow views.

Example:

Admin: sudo airflow create_user -r Admin -u vishalkokane -e vishal@gmail.com -f vishal -l kokane -p Vishal@123

Viewer: sudo airflow create_user -r Viewer -u viewuser -e vishalview@gmail.com -f vishalview -l kokaneview -p Vishal@123

User: sudo airflow create_user -r User -u user -e vishaluser@gmail.com -f vishaluser -l kokaneyser -p Vishal@123

Op: sudo airflow create_user -r Op -u opuser -e vishalopuser@gmail.com -f vishalop -l kokaneop -p Vishal@123

Airflow

localhost:8080/users/add

Airflow DAGs Security Browse Admin Docs About 2019-08-01, 06:53:08 UTC vishal kokane

Add User

First Name *	firstname Write the user first name or names
Last Name *	lastname Write the user last name
User Name *	firstname_lastname Username valid for authentication on DB or LDAP, unused for OID auth
Is Active?	☐ It's not a good policy to remove a user, just make it inactive
Email *	firstname_lastname@gmail.com The user's email, this will also be used for OID auth
Role	× Admin × Viewer × User × Op × Public The user role on the application, this will associate with a list of permissions
Password *	•••••••••• Please use a good password policy, this application does not check this for you
Confirm Password	•••••••••• Please rewrite the user's password to confirm

Airflow

localhost:8080/users/list/

Airflow DAGs Security Browse Admin Docs About 2019-08-01, 06:53:38 UTC vishal kokane

Added Row

List Users

Search

Record Count: 6

	First Name	Last Name	User Name	Email	Is Active?	Role
	vishal	kokane	vishalkokane	vishal@gmail.com	True	[Admin]
	vishalview	kokaneview	viewuser	vishalview@gmail.com	True	[Viewer]
	vishaluser	kokaneyser	user	vishaluser@gmail.com	True	[User]
	vishalop	kokaneop	opuser	vishalopuser@gmail.com	True	[Op]
	vishalview2	kokaneview2	viewuser2	vishalview2@gmail.com	True	[Viewer]
	firstname	lastname	firstname_lastname	firstname_lastname@gmail.com	False	[Admin, Public, Viewer, User, Op]

Data profiling(Deleted from current version 1.10.3)

- **Adhoc queries:** sql interaction with database connections registered in airflow
- **Charts:** It allows building data visualizations and charts easily
- **Gantt View**: It shows the runtime of each tasks in dag. Gantt chart get prepared when the task is running
- **Landing time**:Over time of each task in dag

Command Line Interface

There are list of commands to handle many operations through commandline
Example:

- `list_dag_runs`
- `list_dags`
- `List_tasks`
- `airflow pause [-h] [-sd SUBDIR] dag_id`
- `airflow unpause [-h] [-sd SUBDIR] dag_id`
- `airflow trigger_dag [-h] [-sd SUBDIR] [-r RUN_ID] [-c CONF] [-e EXEC_DATE] dag_id`
- `airflow delete_dag [-h] [-y] dag_id`
- `airflow initdb [-h]`

Command Line Interface [contd..]

- `airflow list_dags [-h] [-sd SUBDIR] [-r]`
- `airflow dag_state [-h] [-sd SUBDIR] dag_id execution_date`
- `airflow webserver`
- `Airflow scheduler`

Thank You...