Moving Data with Apache Airflow at Zendesk

Resources and Guidelines

An overview of Airflow and how we use it at Zendesk for data engineering

Pitt Fagan
21 September 2017

zendesk

Today’s agenda

MadPy meetup, 21 Sept, 2017

1. Introductions

2. Overview of Airflow (components, architecture, etc.)

3. Demo of the GUI

4. Using Airflow at Zendesk

5. Questions

A bit about me …

1. Education
   - B.A. Geology from Rice University
   - M.S. Soil Science from UW-Madison
   - M.S. Statistics from UW-Madison

2. Work history
   - Scientist at US Geological Survey
   - V.P. Data Engineering at Earthling Interactive
   - Senior Data Analyst at Zendesk

3. BigDataMadison Meetup
   Organizer, 2012 - present

4. BigDataWisconsin Conference
   Co-Organizer, 2016 - present

Apache Airflow

Provenance

- Development began in 2014
- Open-sourced by AirBnB in 2015
- Accepted into Apache Software Foundation (ASF) Incubator in 2016
- Current version 1.8.1
- 100% Python!

Apache Airflow

Activity on GitHub

- URL: https://github.com/apache/incubator-airflow
- 300+ committers
- 4000+ commits
- 6000+ stars

Apache Airflow

What does it do?

- Task-based data orchestration platform
- Create, schedule and monitor workflows
- Workflows consist of DAGs of tasks
- DAGs control WHEN something happens, and under what conditions
- Tasks control WHAT happens

Apache Airflow

General Architecture – Two main building blocks

DAGs - Directed Acyclic Graphs
- a collection of all the tasks you want to run for a job, organized in a way that reflects their ordering, relationships and dependencies.

DAGs aren’t concerned with what its constituent tasks do;
It’s job is to make sure that whatever they do happens:

- at the right time,
- in the right order,
- with the right handling of any unexpected issues

Apache Airflow

General Architecture – Two main building blocks

Operators - a single task in the workflow

Many types:

- SqlOperator
- PythonOperator
- BashOperator
- HTTPOperator
- AWS and GCS operators
- plenty more, including community supplied and custom

Apache Airflow

General Architecture – Major Components

- Auth - authentication protocols (password, Kerberos, LDAP, Google)
- Hooks – interface/connect to external platforms and databases
- Sensors - wait for files, change of state, db rows, etc.
- Executors - control scheduling/how the DAGs are run (local, sequential, Celery, Mesos are defaults available)

Apache Airflow

General Architecture – Controlling the DAG

- Linear DAGs are the simplest variant
- Branching logic - using the BranchPythonOperator
- Sub-dags - useful for repeated patterns (where the subdag is returned from a function
- SLAs (Service Level Agreement) - time-based checks on task success

Apache Airflow

Running Airflow

- Command line interface - full suite of commands: DAG operation, scheduling
- GUI - Lots of functionality here (live demo coming up)
    - Scheduling
    - Dig into code
    - Administration
    - Activity history

LIVE DEMO!

zendesk

Apache Airflow

Why we use it at Zendesk

- Populate BigQuery (GCS data warehouse) with information from multiple third party applications used throughout company
  - NetSuite
  - HEAP
  - Zuora
  - Salesforce
  - Bizible
  - MySQL
  - Hadoop

Apache Airflow

Cloud implementation

- Use a custom executor for Kubernetes*, to scale up in a cloud environment
- Container spawner
- Uses Kubernetes and Docker** instances to spin up each task (sequentially or in parallel) in separate containers

* A system for management/deployment of containers in a cluster

** Container software

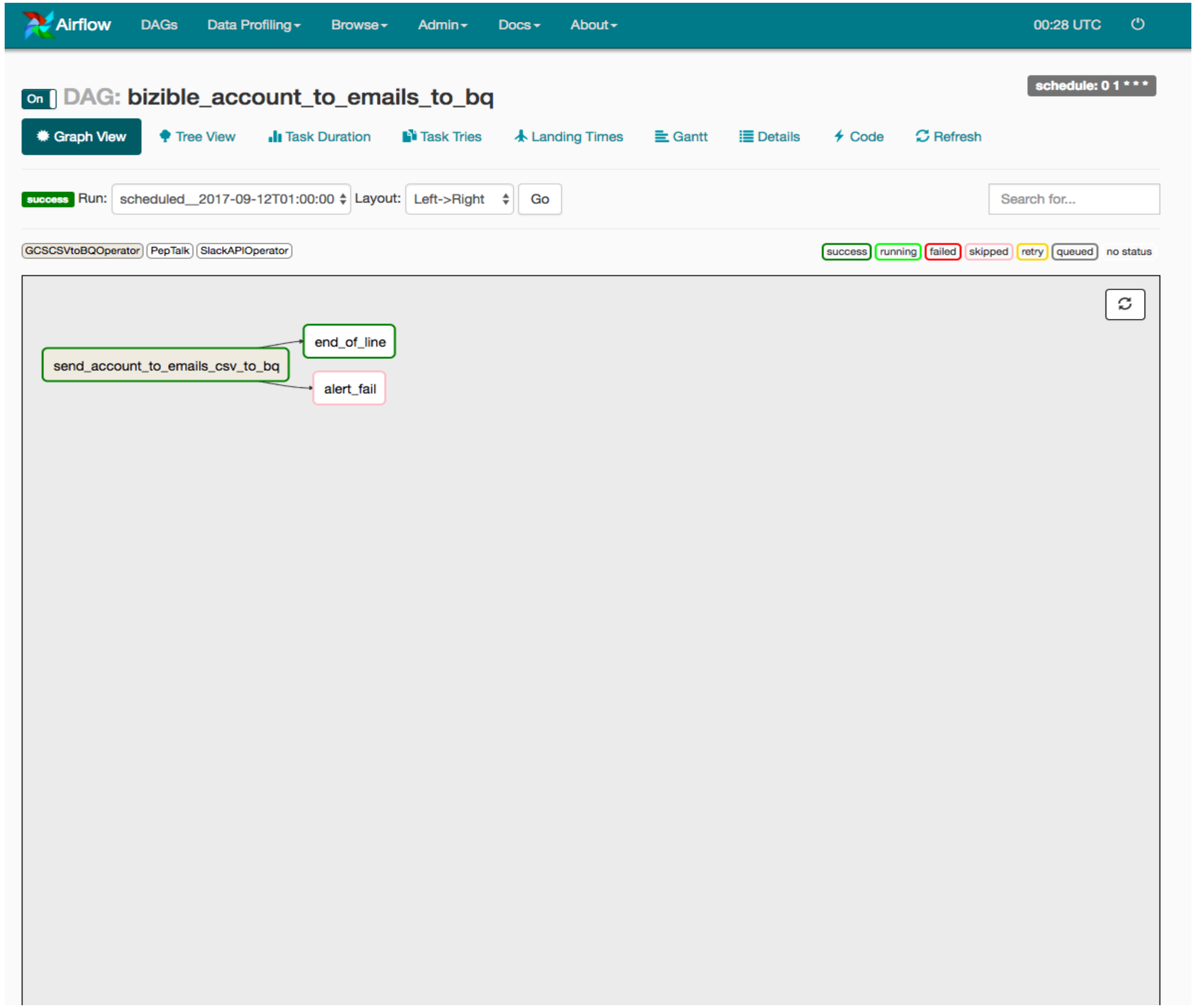
Apache Airflow

General Implementation Pattern at Zendesk

- Use hooks to extract datafile to GCS buckets, employing a consistent nested folder construct
- Pick up the file and load the data to a stage table
  - Sensor detects new file(s)
  - Container spawner fires up new container to process the first task in the DAG (KubeExecutor).
  - Once finished spins down and launches new one (if necessary)
- Perform quality checks on the data, notify via Slack
- Move data to the production table using an SCD type 2 pattern
- Share data out between the separated datasets using views, to avoid exposing underlying source tables

Apache Airflow

How we use it at Zendesk



DAG: bizible_account_to_emails_to_bq

schedule: 0 1 * * *

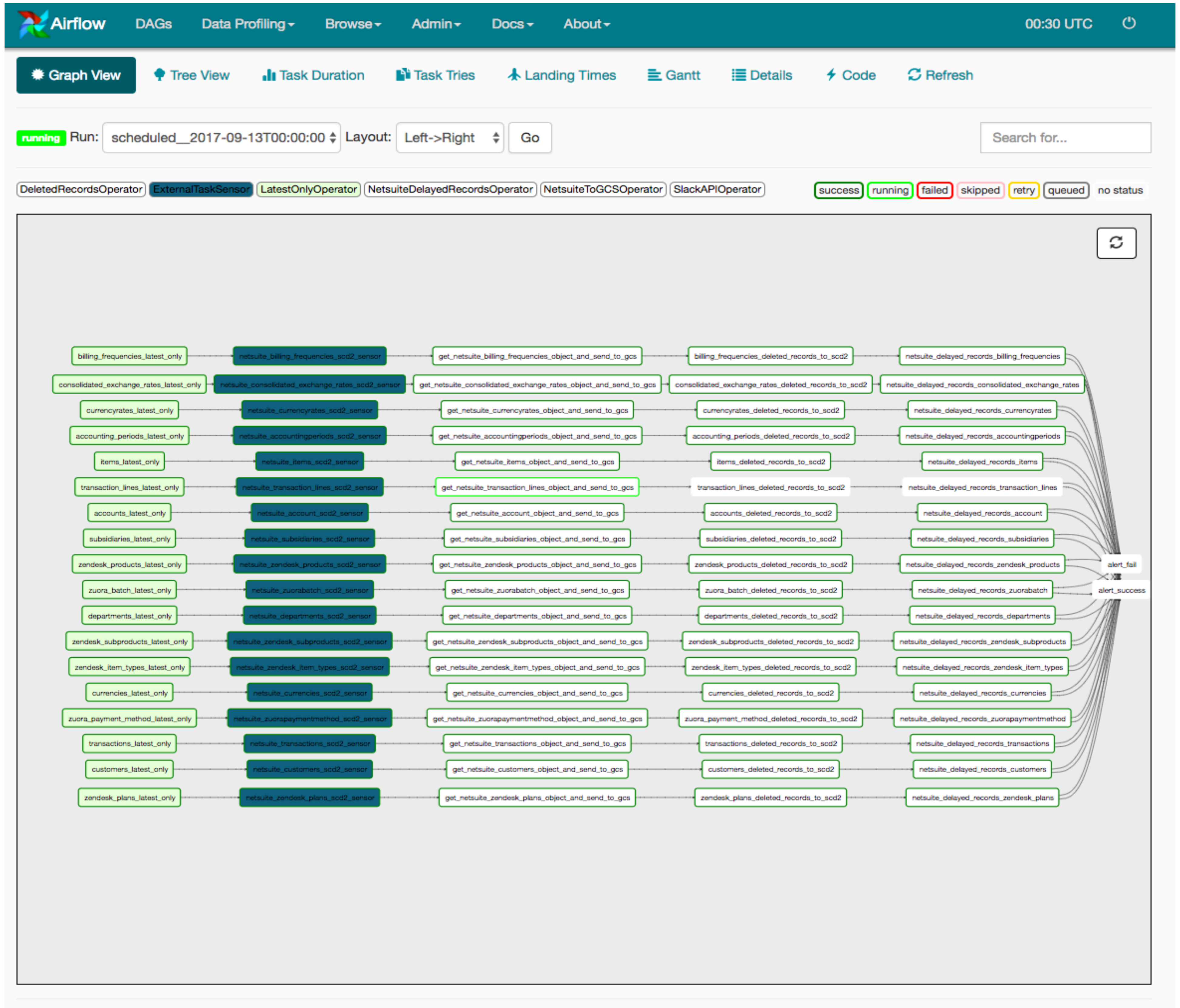
success Run: scheduled__2017-09-12T01:00:00 Layout: Left->Right Go

Graph View simple DAG

- Failure vs Success

Apache Airflow

How we use it at Zendesk


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
00:30 UTC
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Search for...
DeletedRecordsOperator
ExternalTaskSensor
LatestOnlyOperator
NetsuiteDelayedRecordsOperator
NetsuiteToGCSOperator
SlackAPIOperator
success
running
failed
skipped
retry
queued
no status
billing_frequencies_latest_only
netsuite_billing_frequencies_scd2_sensor
get_netsuite_billing_frequencies_object_and_send_to_gcs
billing_frequencies_deleted_records_to_scd2
netsuite_delayed_records_billing_frequencies
consolidated_exchange_rates_latest_only
netsuite_consolidated_exchange_rates_scd2_sensor
get_netsuite_consolidated_exchange_rates_object_and_send_to_gcs
consolidated_exchange_rates_deleted_records_to_scd2
netsuite_delayed_records_consolidated_exchange_rates
currencyrates_latest_only
netsuite_currencyrates_scd2_sensor
get_netsuite_currencyrates_object_and_send_to_gcs
currencyrates_deleted_records_to_scd2
netsuite_delayed_records_currencyrates
accounting_periods_latest_only
netsuite_accountingperiods_scd2_sensor
get_netsuite_accountingperiods_object_and_send_to_gcs
accounting_periods_deleted_records_to_scd2
netsuite_delayed_records_accountingperiods
items_latest_only
netsuite_items_scd2_sensor
get_netsuite_items_object_and_send_to_gcs
items_deleted_records_to_scd2
netsuite_delayed_records_items
transaction_lines_latest_only
netsuite_transaction_lines_scd2_sensor
get_netsuite_transaction_lines_object_and_send_to_gcs
transaction_lines_deleted_records_to_scd2
netsuite_delayed_records_transaction_lines
accounts_latest_only
netsuite_account_scd2_sensor
get_netsuite_account_object_and_send_to_gcs
accounts_deleted_records_to_scd2
netsuite_delayed_records_account
subsidiaries_latest_only
netsuite_subsidiaries_scd2_sensor
get_netsuite_subsidiaries_object_and_send_to_gcs
subsidiaries_deleted_records_to_scd2
netsuite_delayed_records_subsidiaries
zendesk_products_latest_only
netsuite_zendesk_products_scd2_sensor
get_netsuite_zendesk_products_object_and_send_to_gcs
zendesk_products_deleted_records_to_scd2
netsuite_delayed_records_zendesk_products
alert_fail
zuora_batch_latest_only
netsuite_zuorabatch_scd2_sensor
get_netsuite_zuorabatch_object_and_send_to_gcs
zuora_batch_deleted_records_to_scd2
netsuite_delayed_records_zuorabatch
alert_success
departments_latest_only
netsuite_departments_scd2_sensor
get_netsuite_departments_object_and_send_to_gcs
departments_deleted_records_to_scd2
netsuite_delayed_records_departments
zendesk_subproducts_latest_only
netsuite_zendesk_subproducts_scd2_sensor
get_netsuite_zendesk_subproducts_object_and_send_to_gcs
zendesk_subproducts_deleted_records_to_scd2
netsuite_delayed_records_zendesk_subproducts
zendesk_item_types_latest_only
netsuite_zendesk_item_types_scd2_sensor
get_netsuite_zendesk_item_types_object_and_send_to_gcs
zendesk_item_types_deleted_records_to_scd2
netsuite_delayed_records_zendesk_item_types
currencies_latest_only
netsuite_currencies_scd2_sensor
get_netsuite_currencies_object_and_send_to_gcs
currencies_deleted_records_to_scd2
netsuite_delayed_records_currencies
zuora_payment_method_latest_only
netsuite_zuorapaymentmethod_scd2_sensor
get_netsuite_zuorapaymentmethod_object_and_send_to_gcs
zuora_payment_method_deleted_records_to_scd2
netsuite_delayed_records_zuorapaymentmethod
transactions_latest_only
netsuite_transactions_scd2_sensor
get_netsuite_transactions_object_and_send_to_gcs
transactions_deleted_records_to_scd2
netsuite_delayed_records_transactions
customers_latest_only
netsuite_customers_scd2_sensor
get_netsuite_customers_object_and_send_to_gcs
customers_deleted_records_to_scd2
netsuite_delayed_records_customers
zendesk_plans_latest_only
netsuite_zendesk_plans_scd2_sensor
get_netsuite_zendesk_plans_object_and_send_to_gcs
zendesk_plans_deleted_records_to_scd2
netsuite_delayed_records_zendesk_plans

running Run: scheduled__2017-09-13T00:00:00 Layout: Left->Right Go

Graph View – complex DAG

- A series of linear DAGs tied together and dependent on one another for success or failure

Apache Airflow

Where to find more information on Airflow

- Documentation - http://pythonhosted.org/airflow
- Repo - https://github.com/apache/incubator-airflow
- Community: https://gitter.im/apache/incubator-airflow

Questions

Pitt Fagan

- http://www.pittfagan.com
- https://www.linkedin.com/in/pittfagan/
- https://www.meetup.com/BigDataMadison/

Thanks for coming!