drivy

How we use and operate Apache Airflow at Drivy

Eloïse GOMEZ & Faouz EL FASSI

drivy

Find a Drivy nearby

50,000 cars from locals

No subscription needed

Open it with the app

24/7 on-demand access

Prices from £29/day

Enjoy the drive!

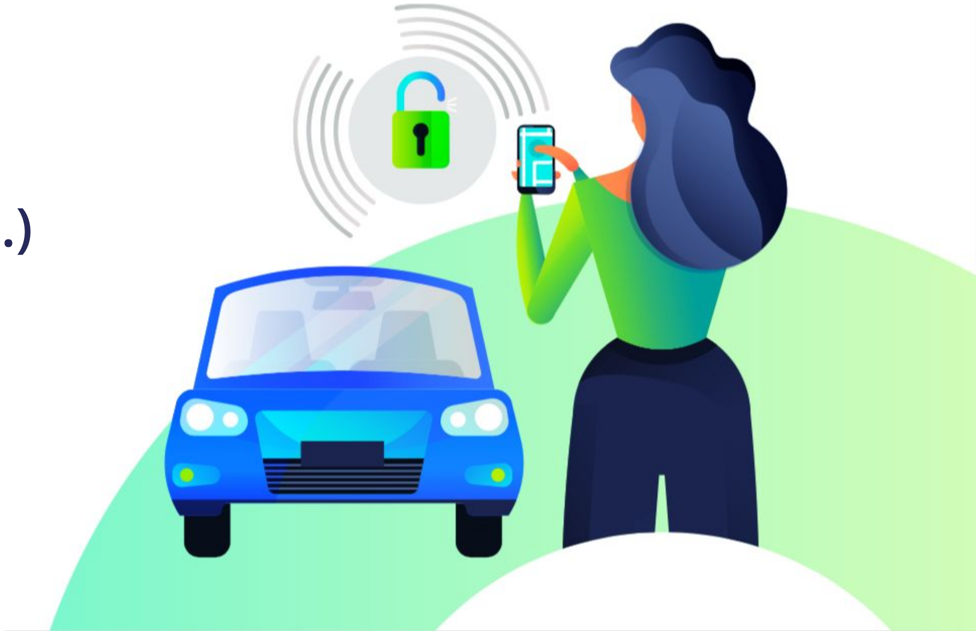
Fully insured round trips

100 mi/day included

Data @ Drivy

- A 3 nodes Redshift WH (Snowflake soon)
- 6TB of data (including tracking)
- 2M users
- 8 employees (6 analysts + 2 engs.)

We are hiring in the data team.
Join us!



Summary

- How do we manage Airflow at Drivy?
- How do we use it?

How do we manage airflow at Drivy?

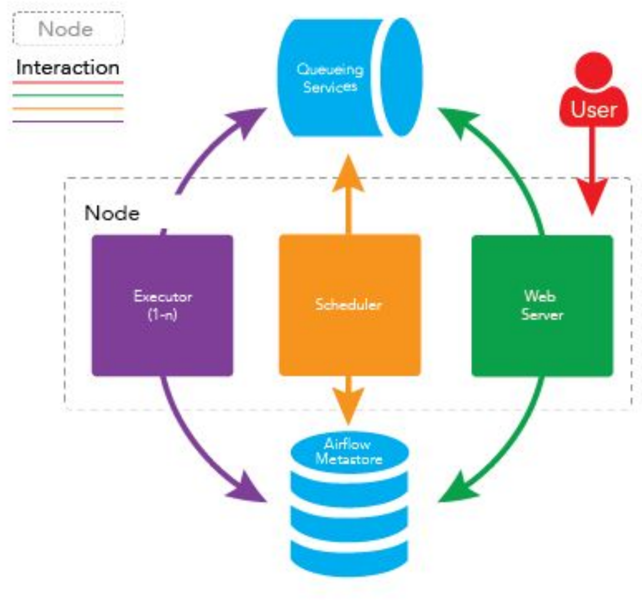
Airflow Recap

"Airflow is a platform to programmatically author, schedule and monitor data pipelines."

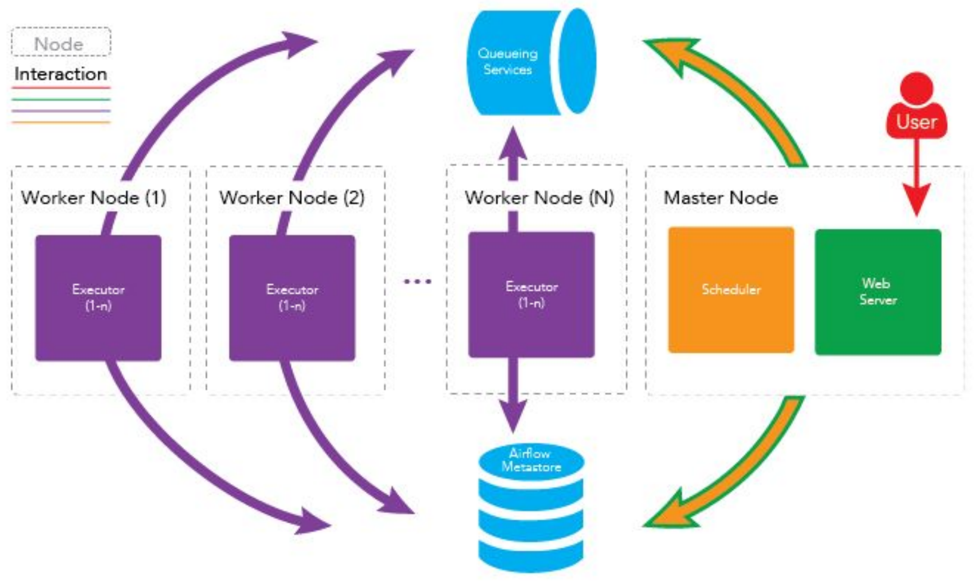
Airflow Components

- **Metadata Database**: stores information regarding tasks state.
- **Scheduler**: decides which tasks need to be executed.
- **Web server**: accepts HTTP requests and allows user to interact with Airflow.
- **Worker**: executes tasks.
- **A queuing service**: hold information about next task to execute.

Airflow Architecture



Local Executor



Celery Executor

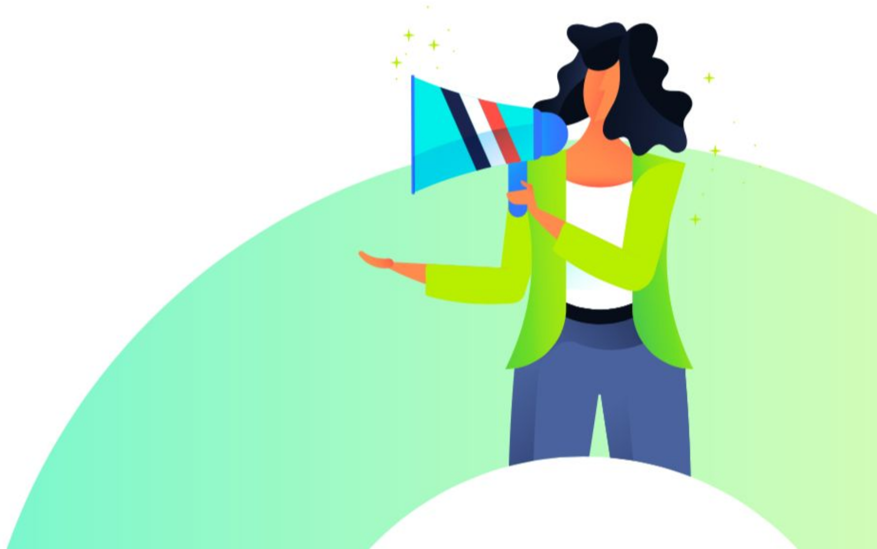
http://site.clairvoyantsoft.com/setting-apache-airflow-cluster/

Single-Node vs Multi-Nodes Architecture

Single-Node Architecture	Multi-Node Architecture
• Easy to set up • Single point of failure • Hard to scale	• Higher availability • Dedicated workers for specific tasks • Scaling horizontally

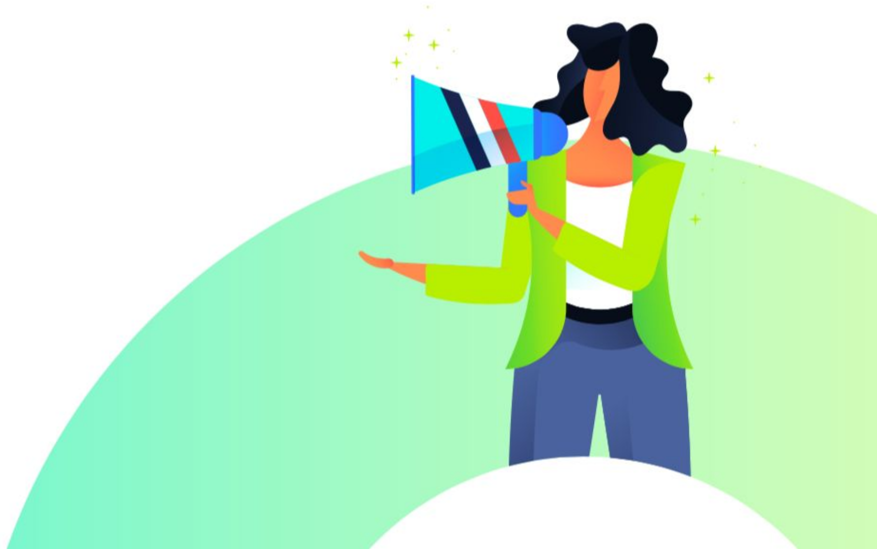
Airflow 6 months ago

- Single-node architecture
- 1 ec2 instance manually set up
- Crontable deployment tool

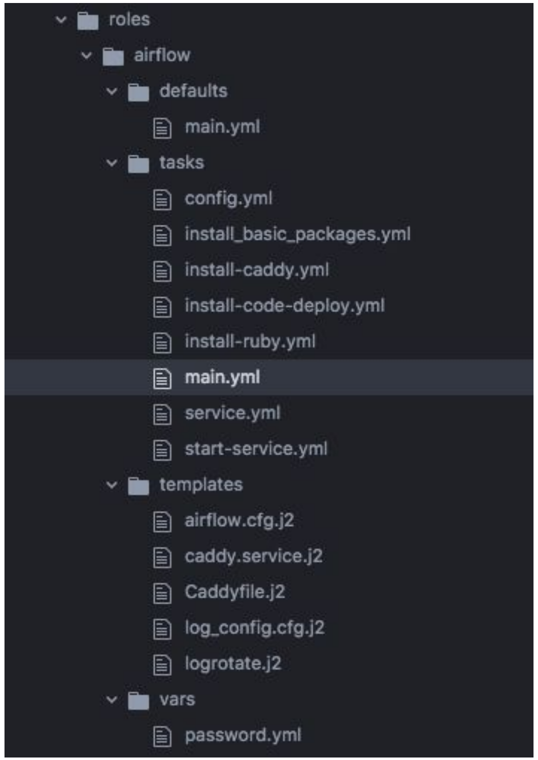


Airflow Architecture now

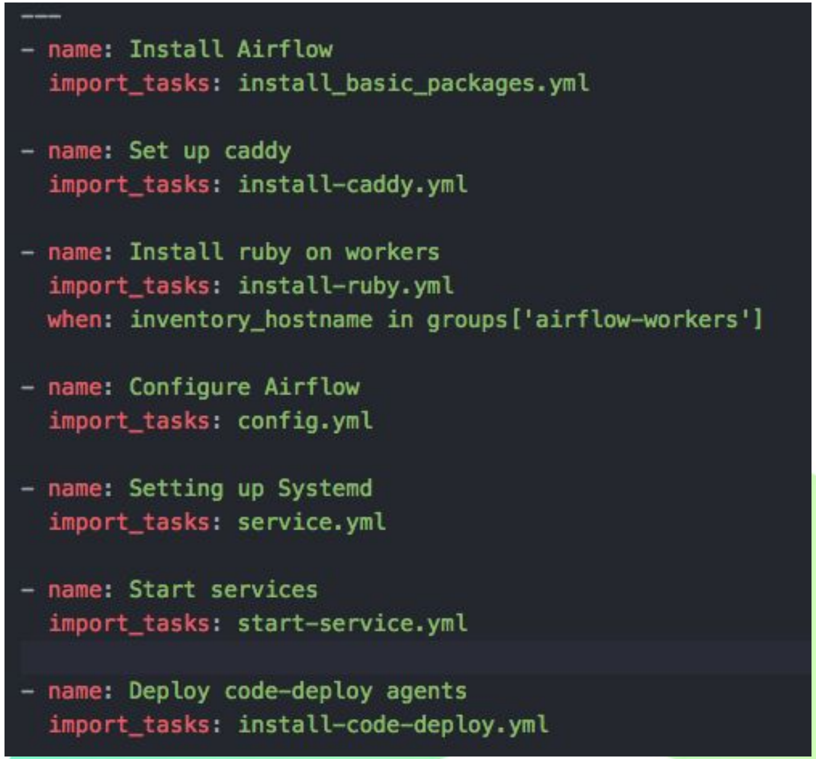
- Multi-node architecture
- 4 ec2 instances installed through ansible
- Code deploy as a deployment tool



Airflow Installation

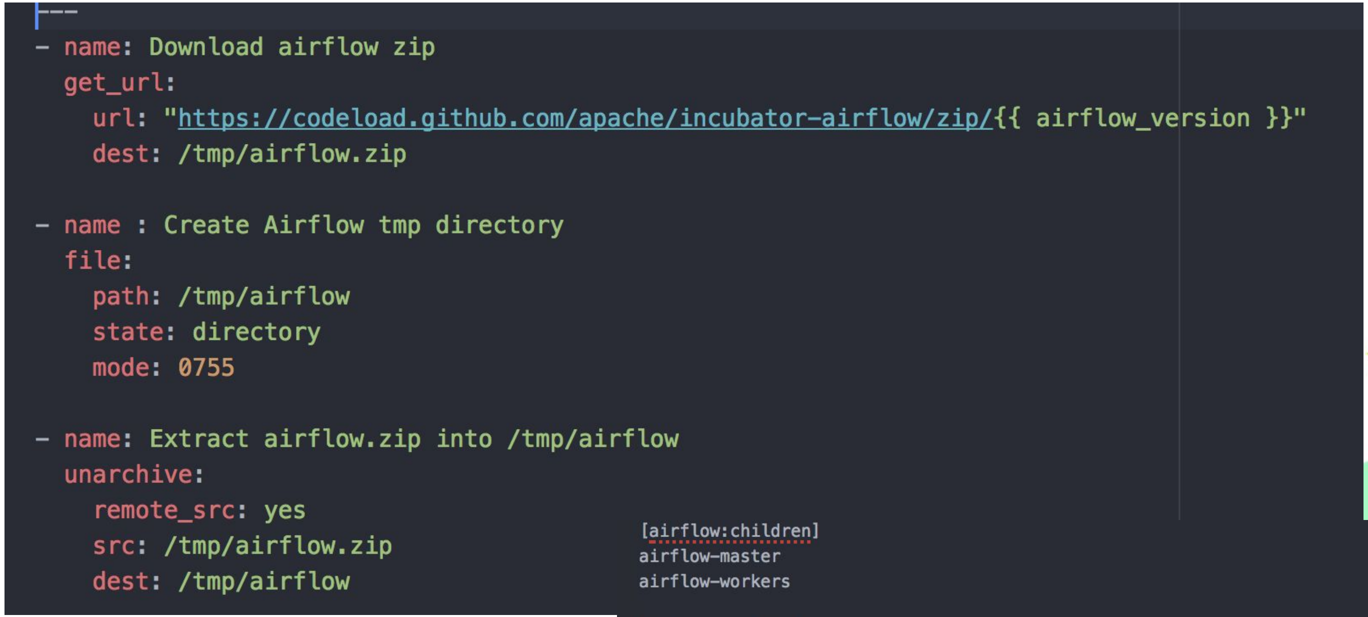


```
roles
  airflow
    defaults
      main.yml
    tasks
      config.yml
      install_basic_packages.yml
      install-caddy.yml
      install-code-deploy.yml
      install-ruby.yml
      main.yml
      service.yml
      start-service.yml
    templates
      airflow.cfg.j2
      caddy.service.j2
      Caddyfile.j2
      log_config.cfg.j2
      logrotate.j2
    vars
      password.yml
```

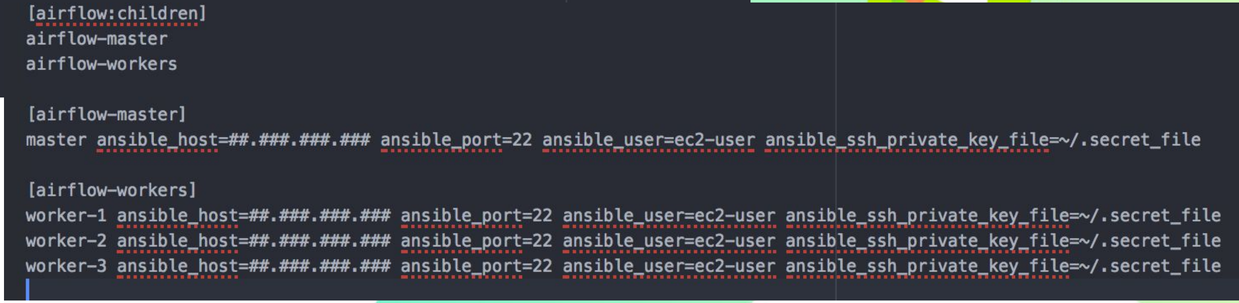


```
---
- name: Install Airflow
  import_tasks: install_basic_packages.yml

- name: Set up caddy
  import_tasks: install-caddy.yml

- name: Install ruby on workers
  import_tasks: install-ruby.yml
  when: inventory_hostname in groups['airflow-workers']

- name: Configure Airflow
  import_tasks: config.yml

- name: Setting up Systemd
  import_tasks: service.yml

- name: Start services
  import_tasks: start-service.yml

- name: Deploy code-deploy agents
  import_tasks: install-code-deploy.yml
```

Airflow Installation



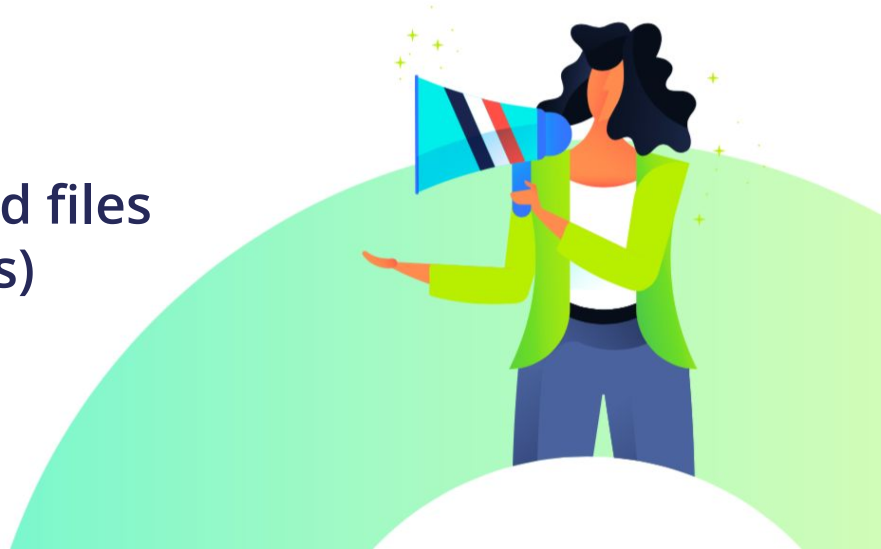
```
---
- name: Download airflow zip
  get_url:
    url: "https://codeload.github.com/apache/incubator-airflow/zip/{{ airflow_version }}"
    dest: /tmp/airflow.zip

- name : Create Airflow tmp directory
  file:
    path: /tmp/airflow
    state: directory
    mode: 0755

- name: Extract airflow.zip into /tmp/airflow
  unarchive:
    remote_src: yes
    src: /tmp/airflow.zip
    dest: /tmp/airflow
```



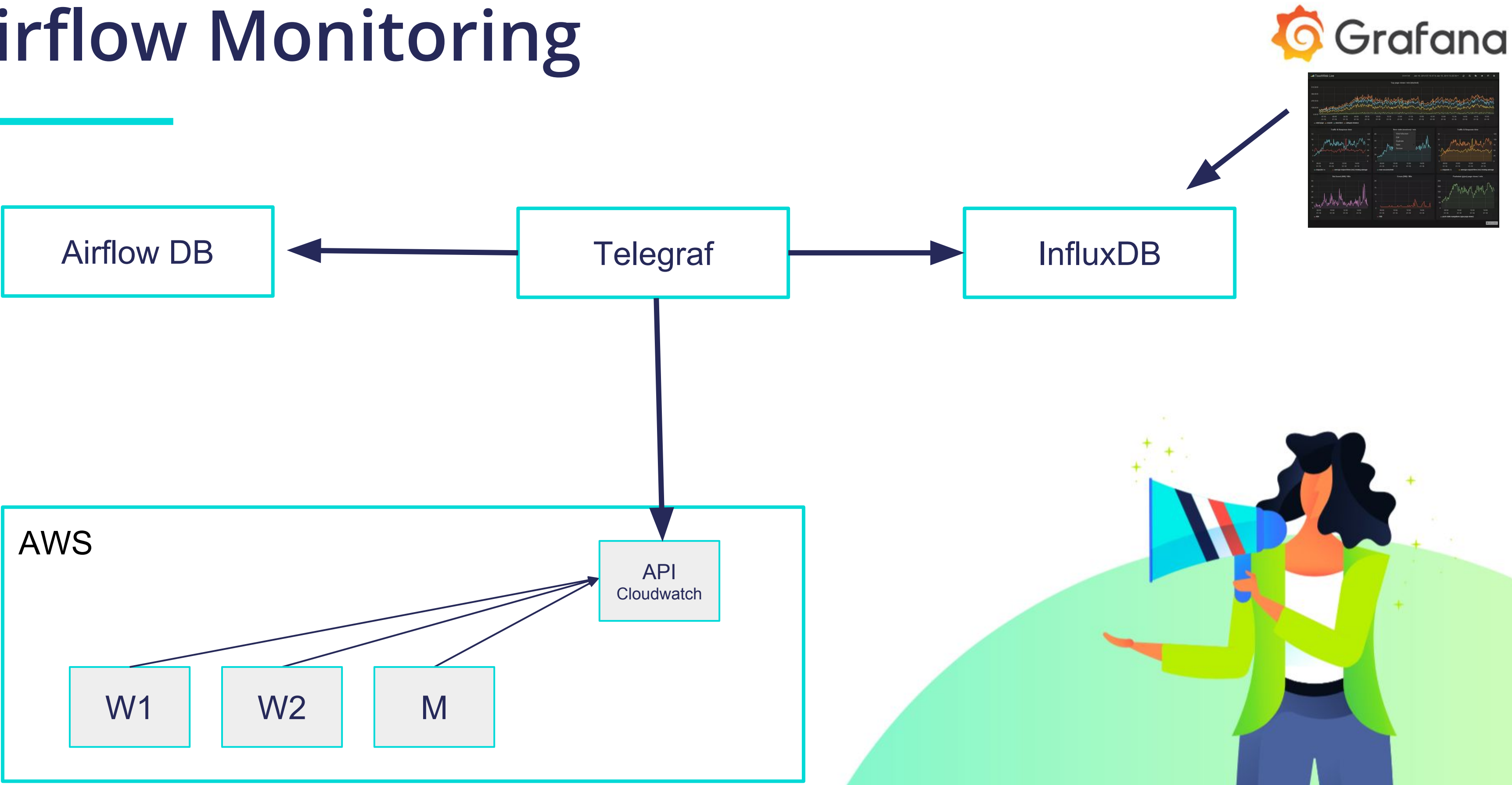
```
[airflow:children]
airflow-master
airflow-workers

[airflow-master]
master ansible_host=##.###.###.### ansible_port=22 ansible_user=ec2-user ansible_ssh_private_key_file=~/.secret_file

[airflow-workers]
worker-1 ansible_host=##.###.###.### ansible_port=22 ansible_user=ec2-user ansible_ssh_private_key_file=~/.secret_file
worker-2 ansible_host=##.###.###.### ansible_port=22 ansible_user=ec2-user ansible_ssh_private_key_file=~/.secret_file
worker-3 ansible_host=##.###.###.### ansible_port=22 ansible_user=ec2-user ansible_ssh_private_key_file=~/.secret_file
```

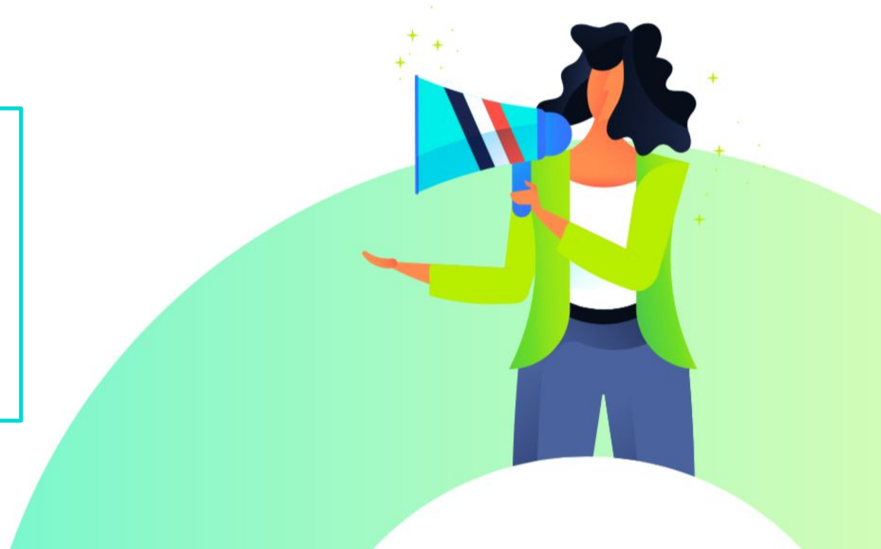
Airflow Migration pain points

- Migrate dag by dag to make sure they are celery executor compliant
- Airflow logs on S3
- Two dag folders
- Need to create symlinks for shared files
 (can’t specify severals dag folders)
- Maintain two Airflows

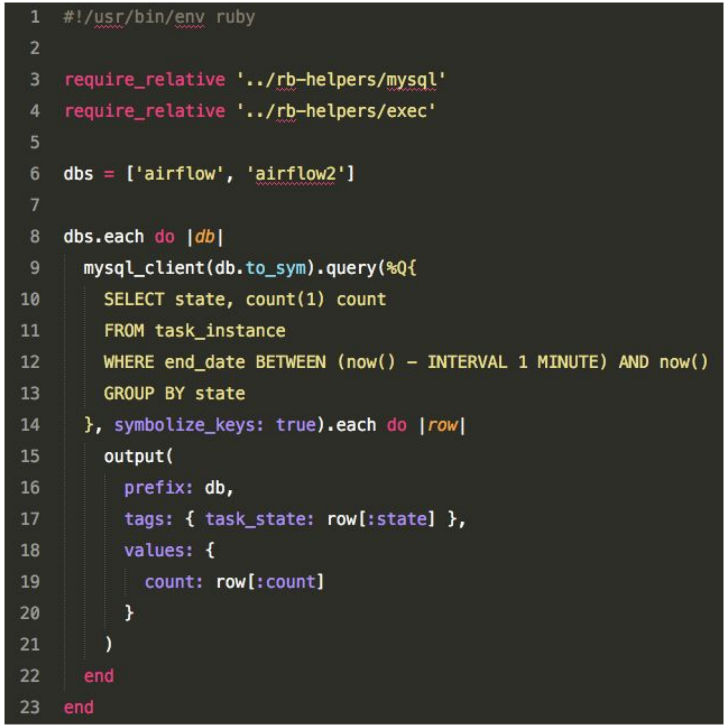


Airflow Monitoring

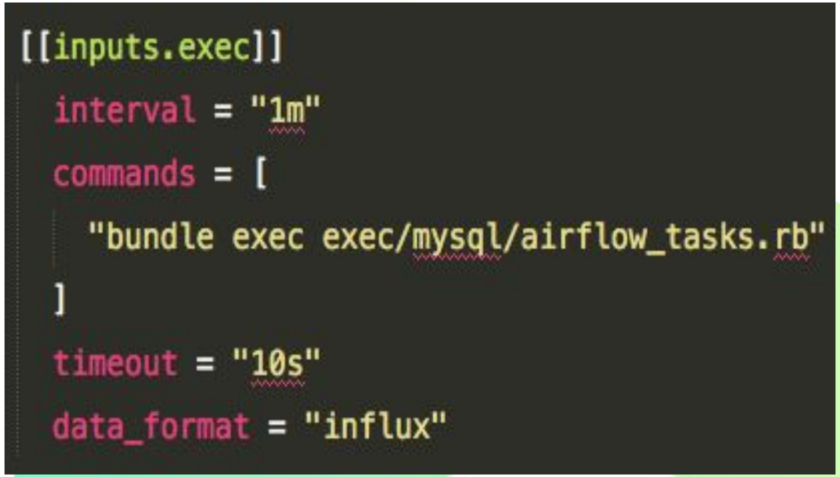




Airflow Monitoring

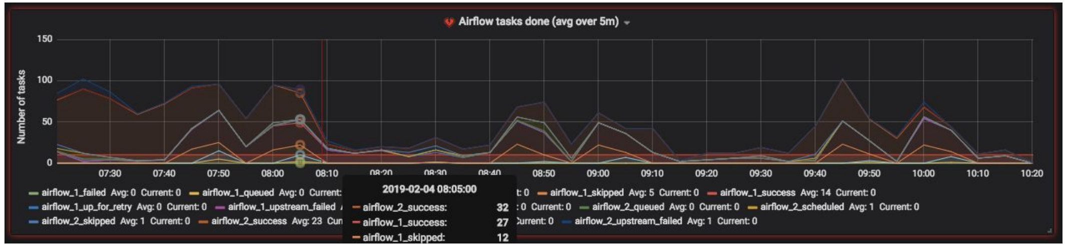


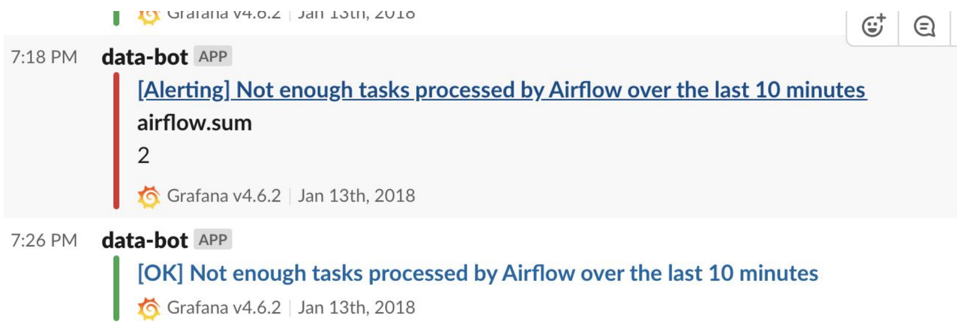
```ruby
#!/usr/bin/env ruby

require_relative '../rb-helpers/mysql'
require_relative '../rb-helpers/exec'

dbs = ['airflow', 'airflow2']

dbs.each do |db|
  mysql_client(db.to_sym).query(%Q{
    SELECT state, count(1) count
    FROM task_instance
    WHERE end_date BETWEEN (now() - INTERVAL 1 MINUTE) AND now()
    GROUP BY state
  }, symbolize_keys: true).each do |row|
    output(
      prefix: db,
      tags: { task_state: row[:state] },
      values: {
        count: row[:count]
      }
    )
  end
end
```



```toml
[[inputs.exec]]
  interval = "1m"
  commands = [
    "bundle exec exec/mysql/airflow_tasks.rb"
  ]
  timeout = "10s"
  data_format = "influx"
```

Airflow Alerting




Grafana v4.6.2 | Jan 13th, 2018

7:18 PM **data-bot** APP

[Alerting] Not enough tasks processed by Airflow over the last 10 minutes

airflow.sum

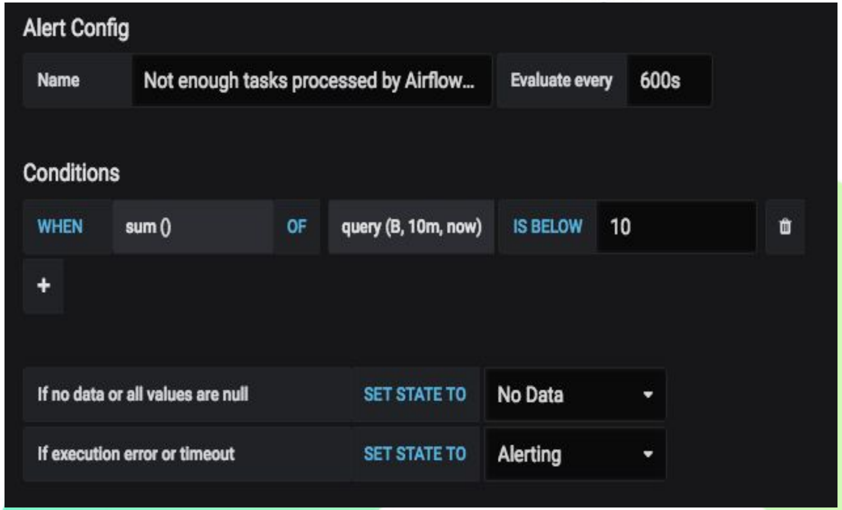
2

Grafana v4.6.2 | Jan 13th, 2018

7:26 PM **data-bot** APP

[OK] Not enough tasks processed by Airflow over the last 10 minutes

Grafana v4.6.2 | Jan 13th, 2018


Alert Config

Name	Not enough tasks processed by Airflow...	Evaluate every	600s

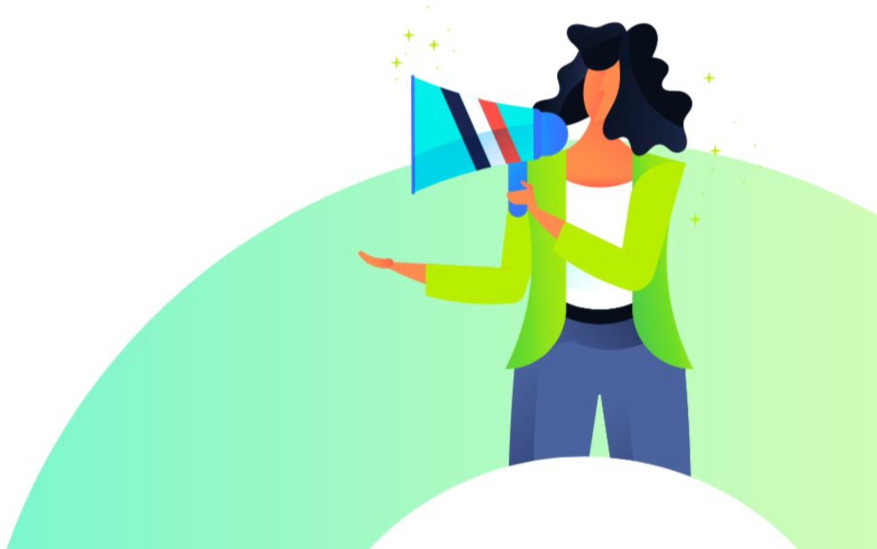
Conditions

WHEN	sum ()	OF	query (B, 10m, now)	IS BELOW	10

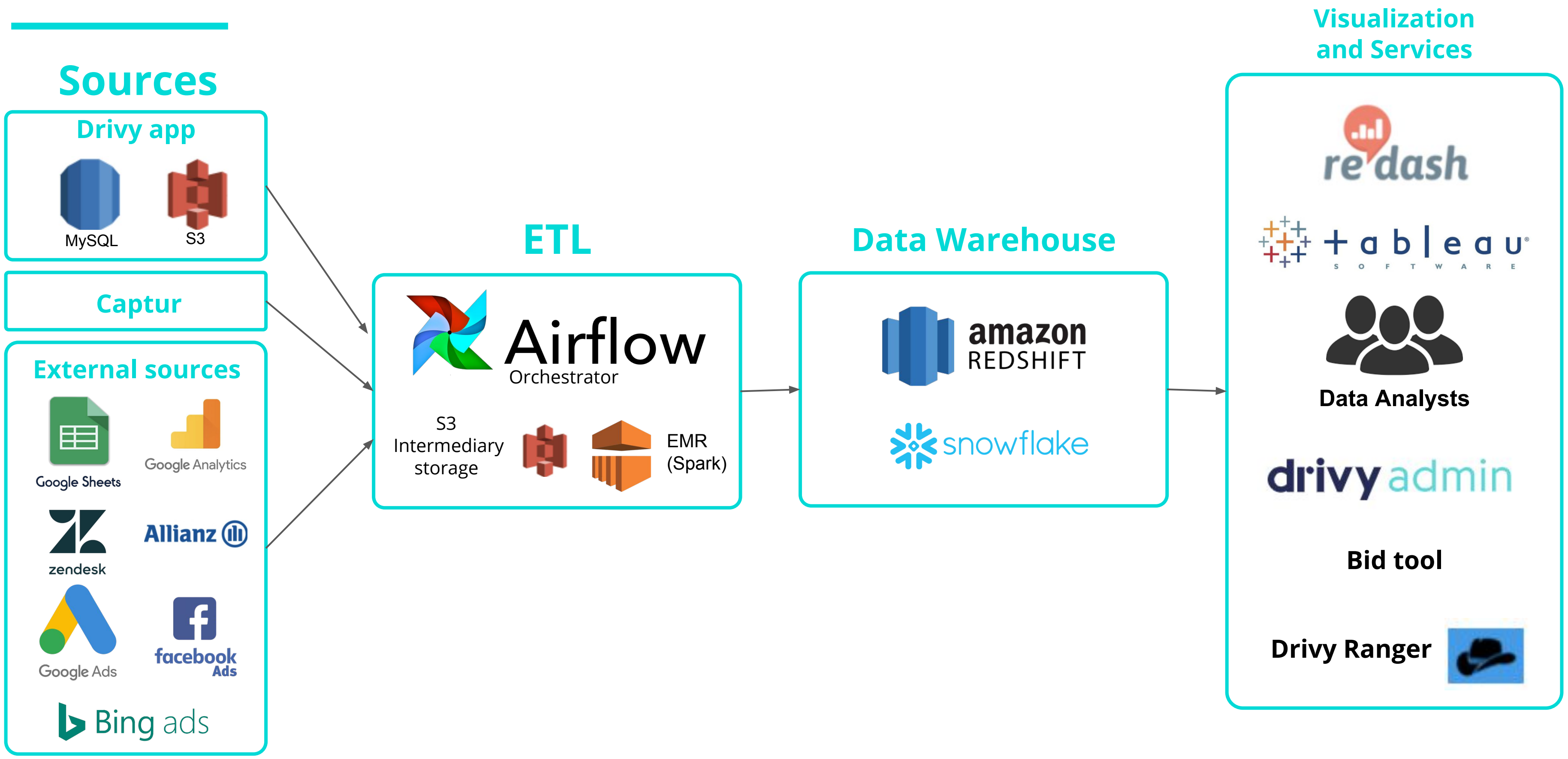
If no data or all values are null	SET STATE TO	No Data
If execution error or timeout	SET STATE TO	Alerting

Airflow tomorrow

- Improve Airflow deployment
- Test managed services
  (Cloud Composer, Astronomer)
- Create a test environment



Infrastructure



How do we use Airflow at Drivy?

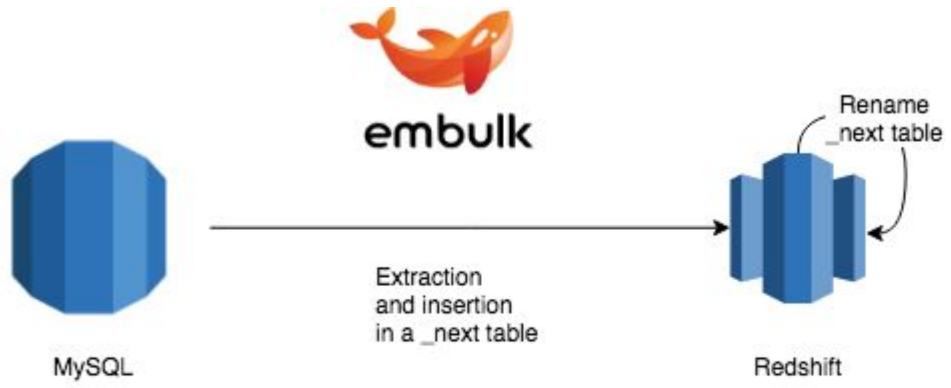
MySQL dump and enrichment

Three kinds of ingestions:

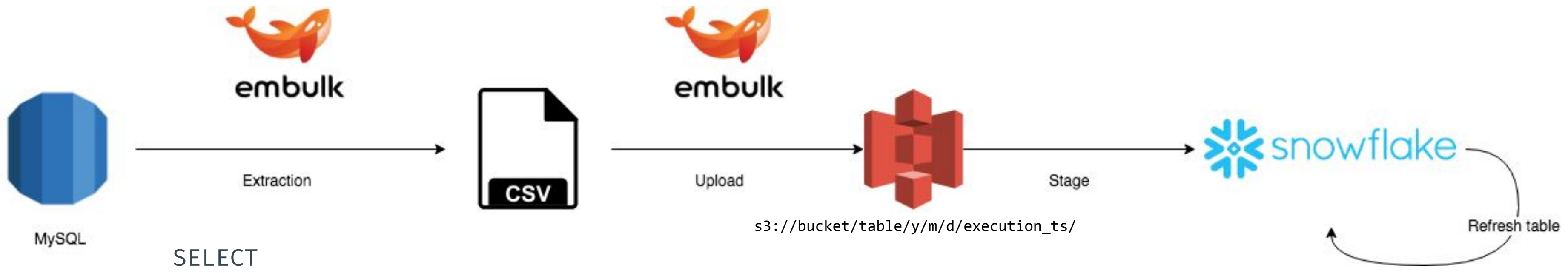
- ➜ Full dump: tables where records can be updated in the past without being timestamped, or having low volumes.
- ➜ Incremental append-only: tables where only new records are added.
- ➜ Incremental UPSERT: tables where new records are added and updated ones are timestamped.

Each of those can be URGENT (every 2h30 to 6h) or NON URGENT (every 6 to 12h)

Full dump imports, the old way



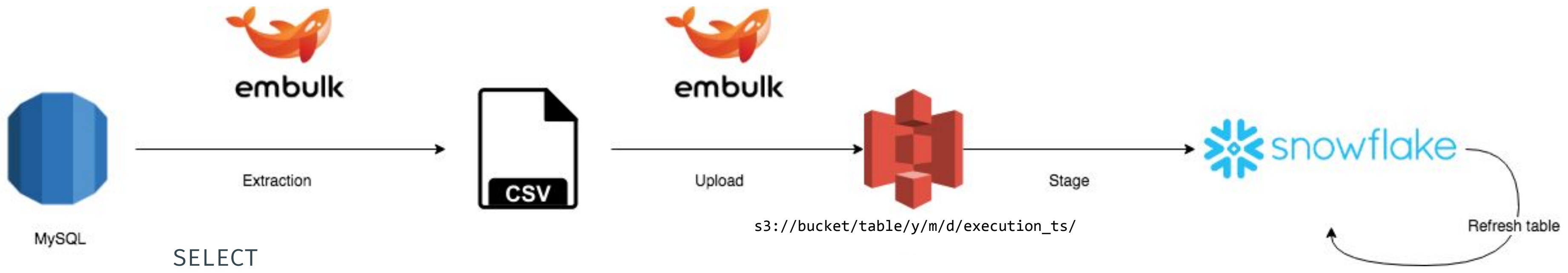
Full dump imports, the new way



s3://bucket/table/y/m/d/execution_ts/

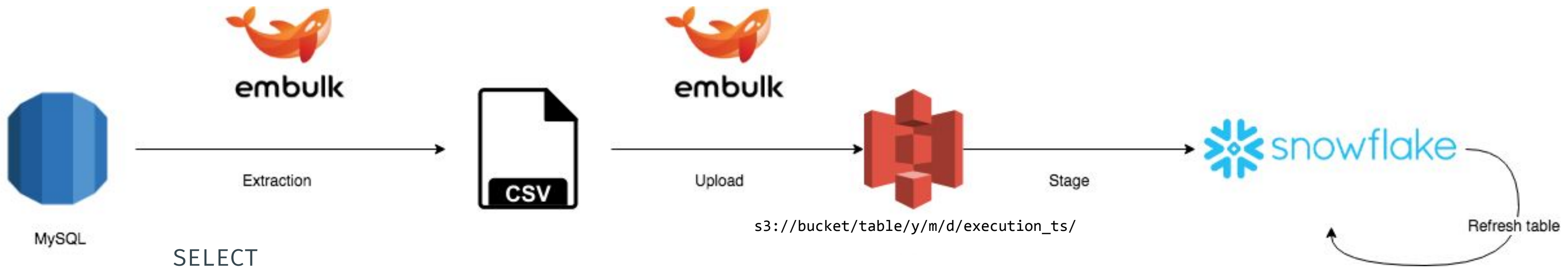
```
SELECT
    col1,
    col2…
FROM TABLE;
```

Incremental append



```
SELECT
    col1,
    col2…
FROM TABLE
WHERE ID > {LAST_ID};
```

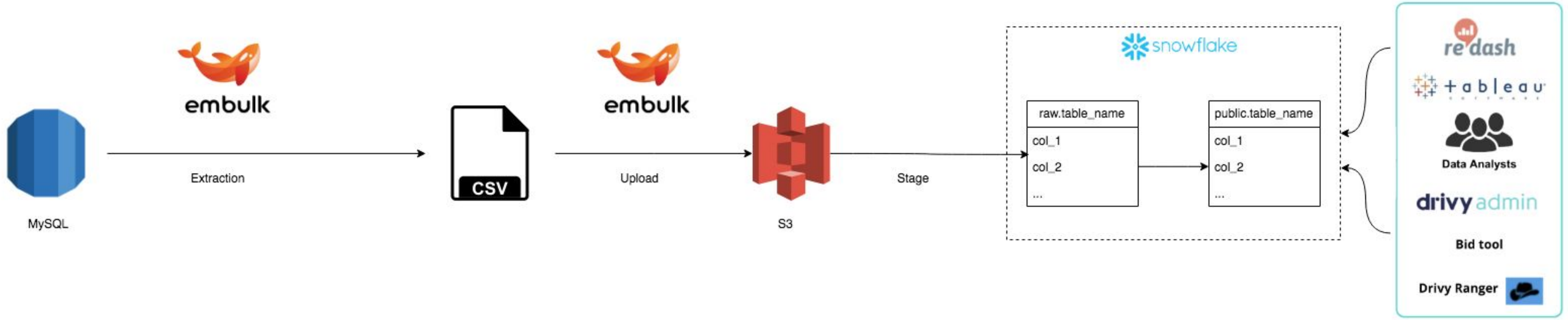
Incremental upsert



s3://bucket/table/y/m/d/execution_ts/

```
SELECT
    col1,
    col2…
FROM TABLE
WHERE ID > {LAST_ID}
OR (UPDATED_AT <> {LAST_UPDATED_AT};
```

Raw / Transformed isolation

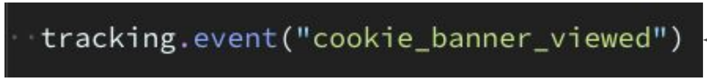


Captur

“Captur is our homemade cross-device tracking framework that mimics Segment architecture”

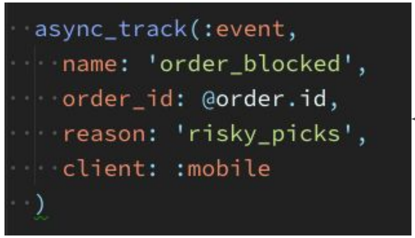
Captur - Usage

The goal of our backend engineers was to make sending events simple, for every developer.



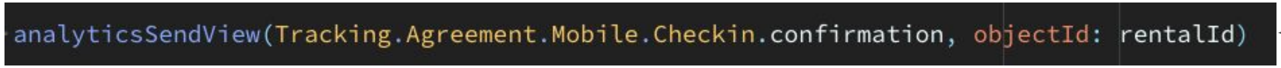
```
  tracking.event("cookie_banner_viewed")
```

javascript



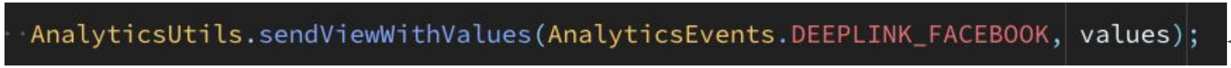
```
  async_track(:event,
    name: 'order_blocked',
    order_id: @order.id,
    reason: 'risky_picks',
    client: :mobile
  )
```

ruby



```
analyticsSendView(Tracking.Agreement.Mobile.Checkin.confirmation, objectId: rentalId)
```

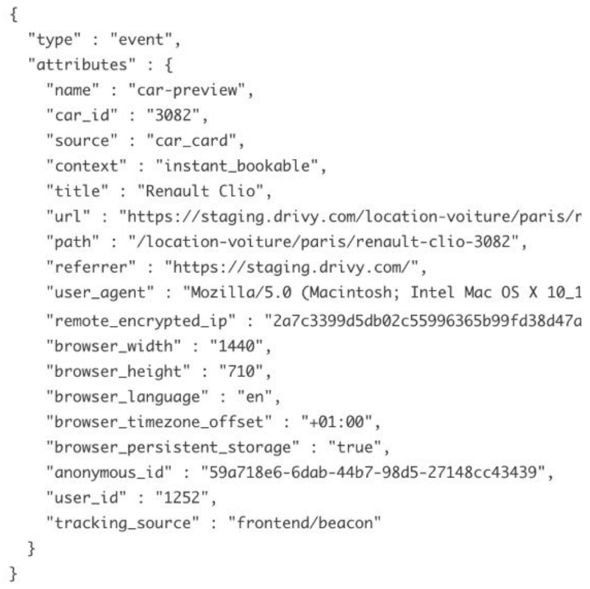
swift



```
 AnalyticsUtils.sendViewWithValues(AnalyticsEvents.DEEPLINK_FACEBOOK, values);
```

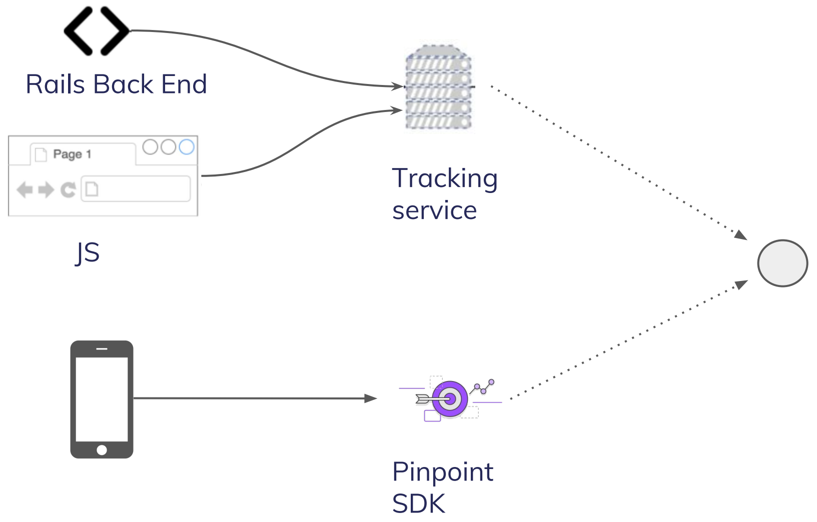
java

Captur - Data collection



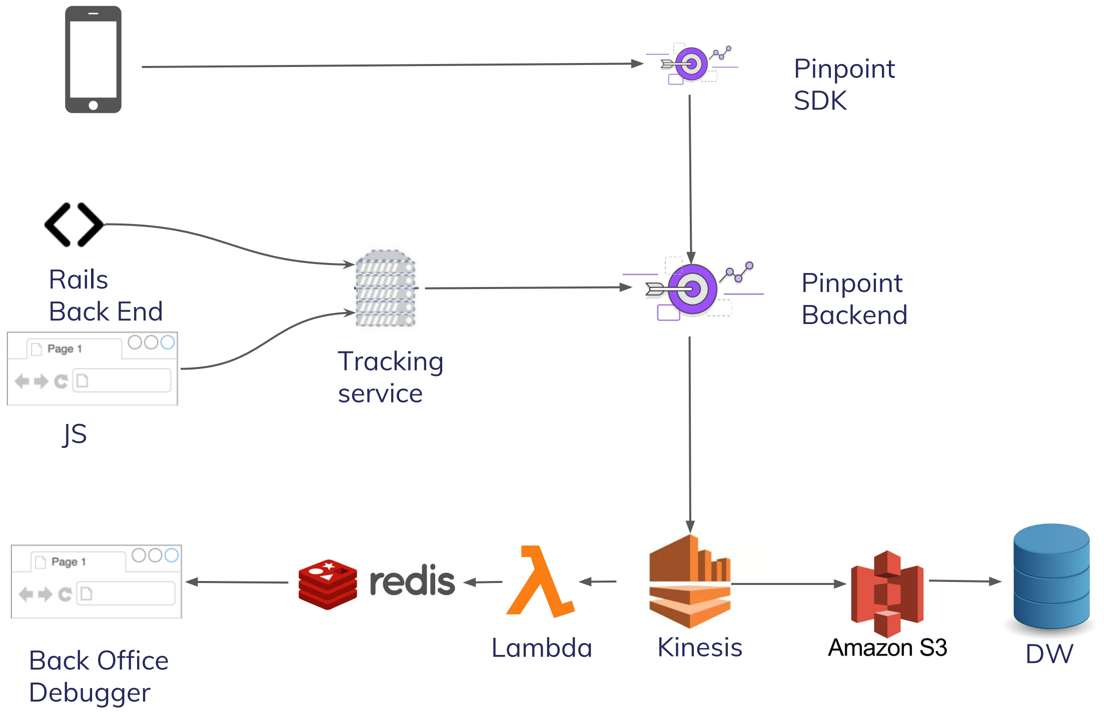
```
{
  "type" : "event",
  "attributes" : {
    "name" : "car-preview",
    "car_id" : "3082",
    "source" : "car_card",
    "context" : "instant_bookable",
    "title" : "Renault Clio",
    "url" : "https://staging.drivy.com/location-voiture/paris/r
    "path" : "/location-voiture/paris/renault-clio-3082",
    "referrer" : "https://staging.drivy.com/",
    "user_agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_1
    "remote_encrypted_ip" : "2a7c3399d5db02c55996365b99fd38d47a
    "browser_width" : "1440",
    "browser_height" : "710",
    "browser_language" : "en",
    "browser_timezone_offset" : "+01:00",
    "browser_persistent_storage" : "true",
    "anonymous_id" : "59a718e6-6dab-44b7-98d5-27148cc43439",
    "user_id" : "1252",
    "tracking_source" : "frontend/beacon"
  }
}
```

Payload example



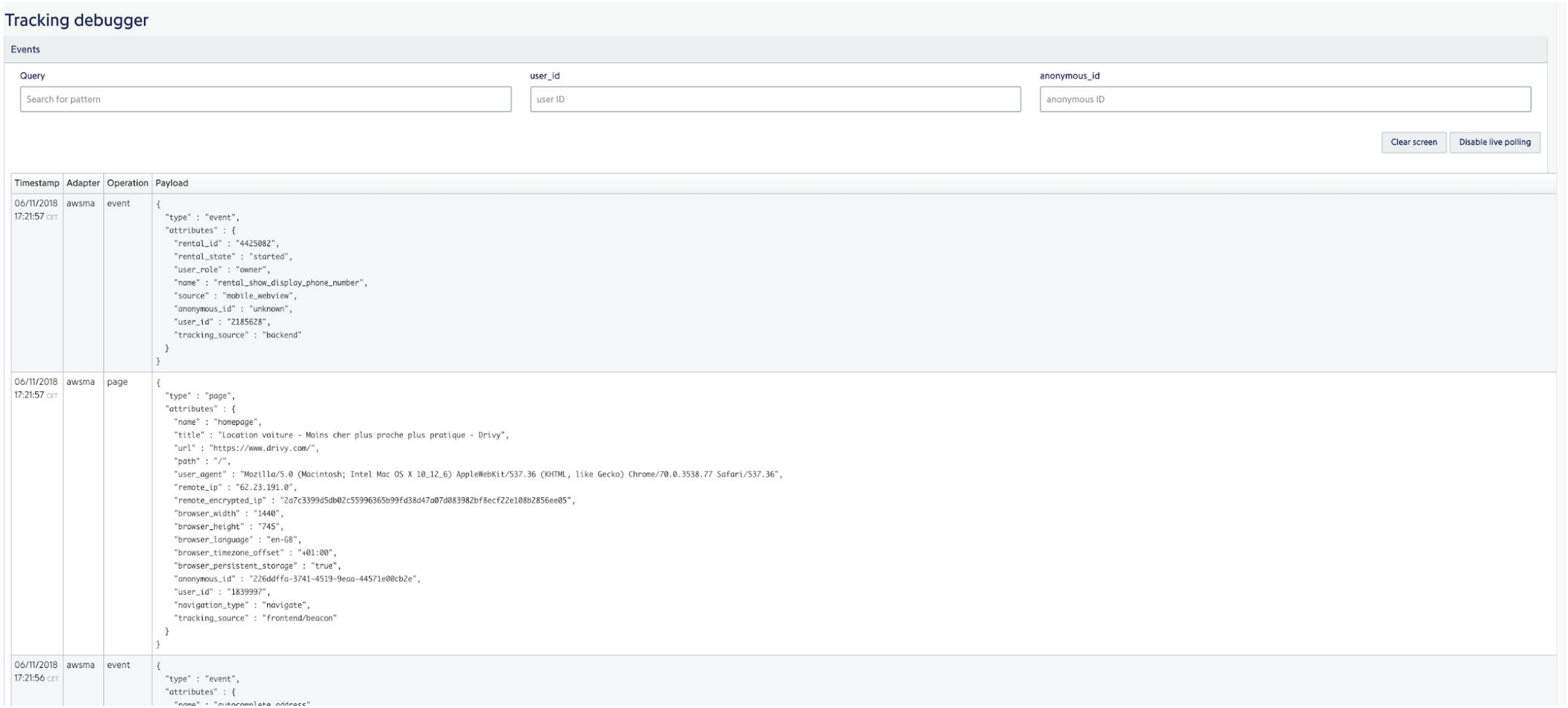
Client side Infrastructure

Captur - Data collection



Infrastructure

Captur - Debug

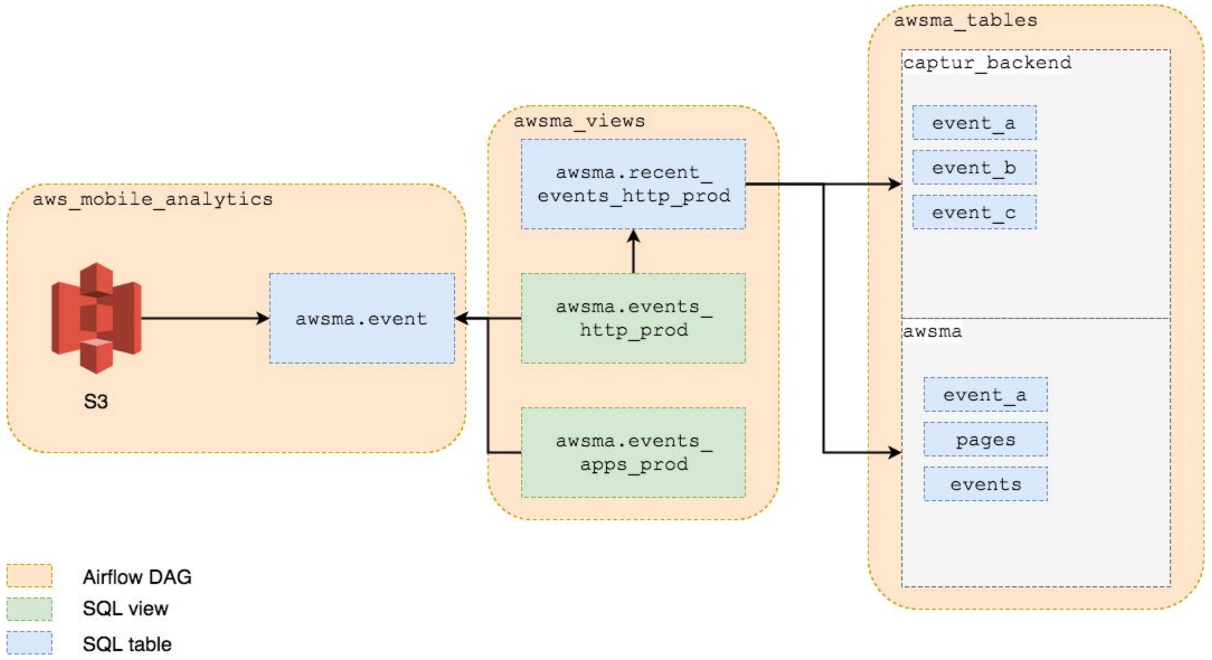


Tracking debugger

Events

```
Timestamp   Adapter  Operation  Payload
06/11/2018  awsma    event      {
17:21:57 CET                      "type" : "event",
                                  "attributes" : {
                                    "rental_id" : "4425082",
                                    "rental_state" : "started",
                                    "user_role" : "owner",
                                    "name" : "rental_show_display_phone_number",
                                    "source" : "mobile_webview",
                                    "anonymous_id" : "unknown",
                                    "user_id" : "2185628",
                                    "tracking_source" : "backend"
                                  }
                                }
06/11/2018  awsma    page       {
17:21:57 CET                      "type" : "page",
                                  "attributes" : {
                                    "name" : "homepage",
                                    "title" : "Location voiture - Moins cher plus proche plus pratique - Drivy",
                                    "url" : "https://www.drivy.com/",
                                    "path" : "/",
                                    "user_agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36",
                                    "remote_ip" : "62.23.191.0",
                                    "remote_encrypted_ip" : "2a7c3399d5db02c55996365b99fd38d47a07d083982bf8ecf22e108b2856ee05",
                                    "browser_width" : "1440",
                                    "browser_height" : "745",
                                    "browser_language" : "en-GB",
                                    "browser_timezone_offset" : "+01:00",
                                    "browser_persistent_storage" : "true",
                                    "anonymous_id" : "226ddffa-3741-4519-9eaa-44571e00cb2e",
                                    "user_id" : "1839997",
                                    "navigation_type" : "navigate",
                                    "tracking_source" : "frontend/beacon"
                                  }
                                }
06/11/2018  awsma    event      {
17:21:56 CET                      "type" : "event",
                                  "attributes" : {
                                    "name" : "autocomplete_address"
```

Captur: ETL side



Usage example: Tableau

Web Analytics

Ex: Booking Form Funnel

Usage example: Fraud Prevention

Drivy Rangers

Rules engine that can detect various types of fraud in minutes

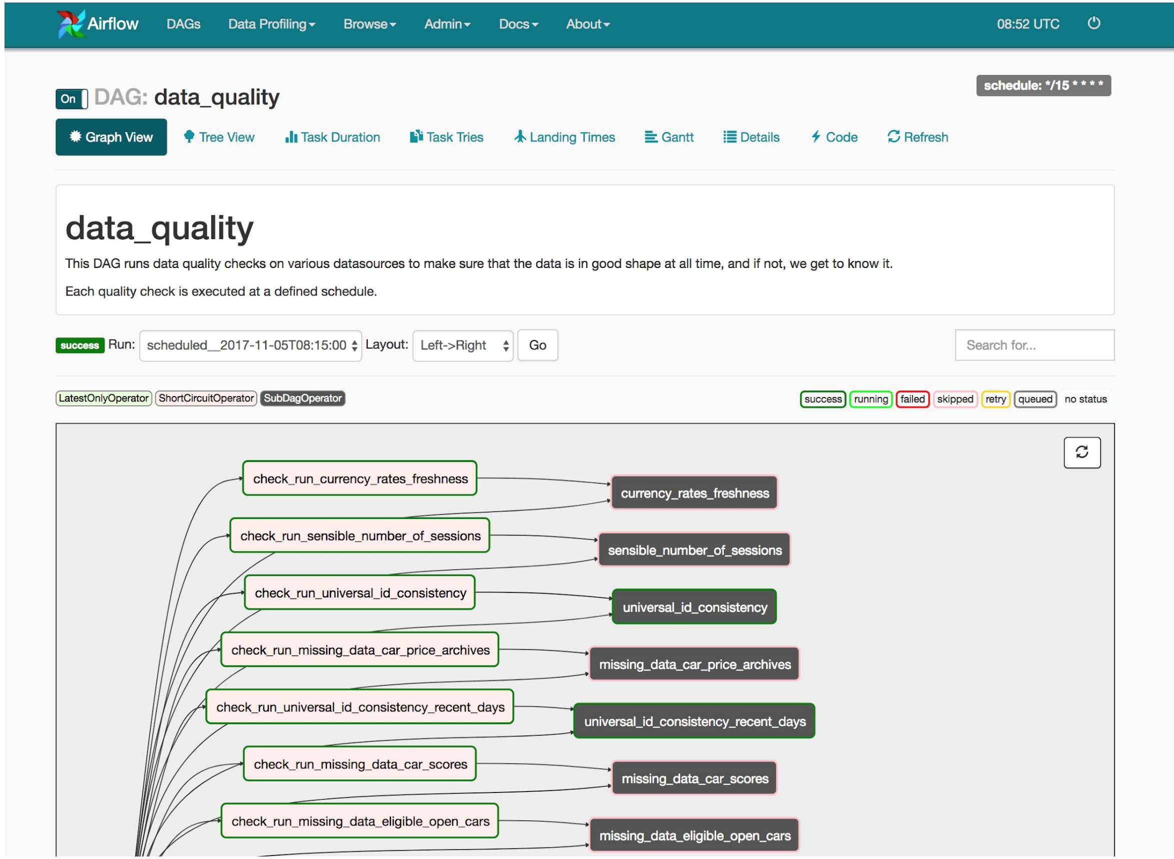
- ➔ >50 alerts in place (theft, insurance fraud, bad usage, phishing)
- ➔ >100 todos / day
- ➔ >40 users blacklisted / day
- ➔ Alerts added instantaneously in prod
- ➔ Escalation process per alerts (slack, pager duty...)
- ➔ Operated by 3 analysts 24/7

Data Quality Checkers

“Data quality checks are performed at a specified interval on one or multiple datasets that are coming from various datasources, using predicates we define. Checks have a tolerance and trigger alerts on alert destinations with an alert level defined by the severity of the found errors.”

Data Quality Checkers

- **ShortCircuitOperator**: determine if the checker should be executed or not.
- **SubDAGOperator**: actually runs the checks.


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
08:52 UTC
schedule: */15 * * * *
On
DAG: data_quality
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh

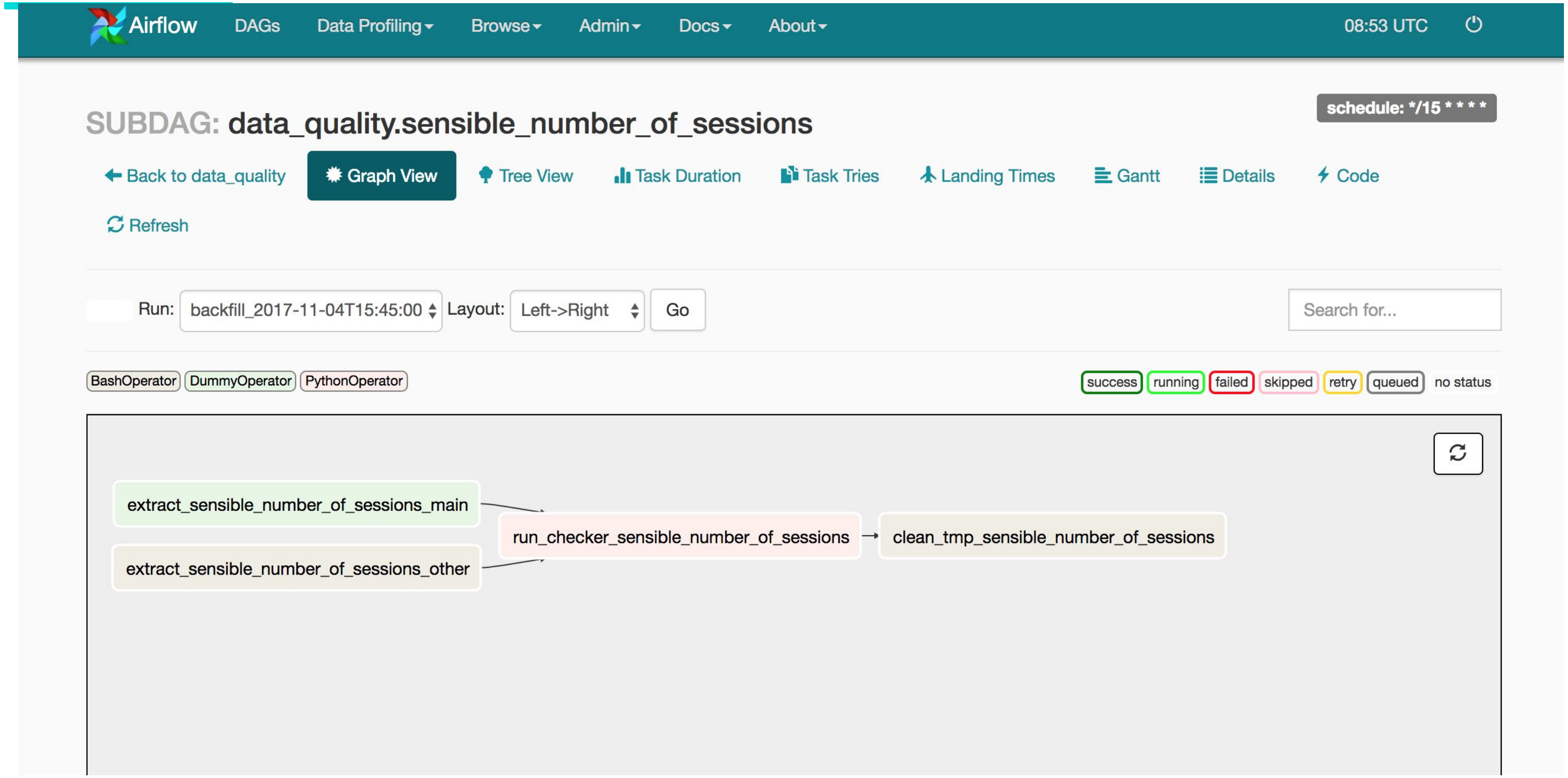
data_quality

This DAG runs data quality checks on various datasources to make sure that the data is in good shape at all time, and if not, we get to know it.

Each quality check is executed at a defined schedule.

success
Run: scheduled__2017-11-05T08:15:00
Layout: Left->Right
Go
Search for...
LatestOnlyOperator
ShortCircuitOperator
SubDagOperator
success
running
failed
skipped
retry
queued
no status
check_run_currency_rates_freshness
currency_rates_freshness
check_run_sensible_number_of_sessions
sensible_number_of_sessions
check_run_universal_id_consistency
universal_id_consistency
check_run_missing_data_car_price_archives
missing_data_car_price_archives
check_run_universal_id_consistency_recent_days
universal_id_consistency_recent_days
check_run_missing_data_car_scores
missing_data_car_scores
check_run_missing_data_eligible_open_cars
missing_data_eligible_open_cars

Data Quality Checker example



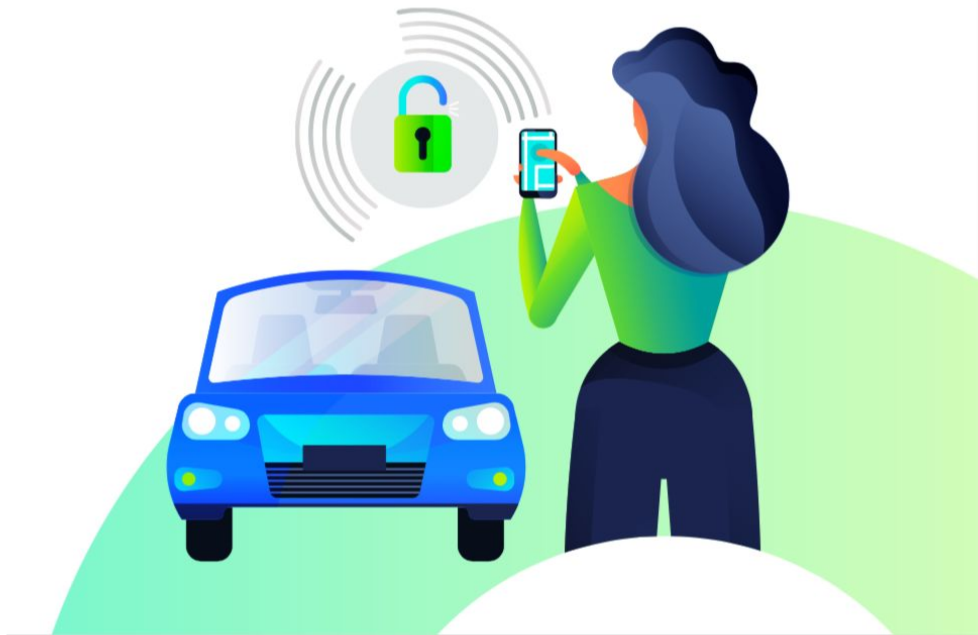
SUBDAG: data_quality.sensible_number_of_sessions

schedule: */15 * * * *

Run: backfill_2017-11-04T15:45:00 Layout: Left->Right

Go
Search for...
BashOperator
DummyOperator
PythonOperator
success
running
failed
skipped
retry
queued
no status
extract_sensible_number_of_sessions_main
extract_sensible_number_of_sessions_other
run_checker_sensible_number_of_sessions
clean_tmp_sensible_number_of_sessions

Questions?



Thank you!