

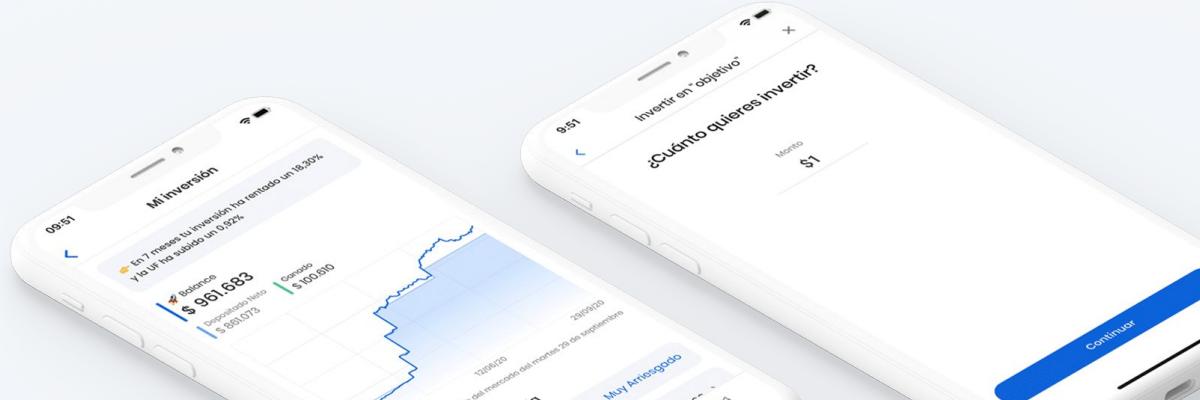


Fintual



ALeRCE
Automatic Learning for the
Rapid Classification of Events

Principios de Ingeniería de Datos: **Fintual** & ALeRCE



Contenido

- Fintual & ALeRCE
- Principios de Ingeniería de Datos
- DE en la Industria: Fintual - Batch Processing
- DE más allá de la industria: ALeRCE - Stream Processing
- Stacks de tecnología para DE
- Airflow: Orquestador de procesos

¿Quien soy yo?

Esteban | reyes.dejong@gmail.com

Me gusta codear IA, la naturaleza y hacer circo



La mejor decisión para tu plata

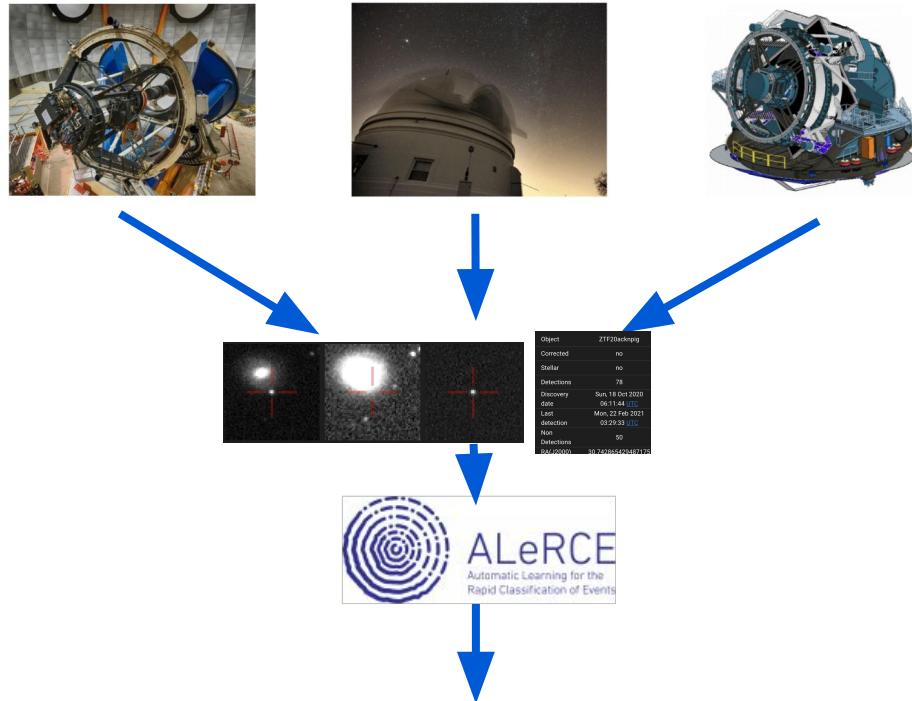
Organiza tus ahorros invirtiendo en alternativas simples y reguladas, con comisiones bajas. Sin papeleos ni letra chica.

Abre tu cuenta gratis



ALeRCE

- Broker Astronómico (único del hemisferio sur).
- Procesamiento masivo de datos astronómicos con IA.
- Interdisciplinario.



Telescopios de seguimiento
&
Comunidad Astronómica

¿Qué es la ingeniería de datos?

Lo que yo pensaba:

- Disponibilizar data para ser interpretada/analizada
- Mover datos de un lugar a otro

¿Qué es la ingeniería de datos?

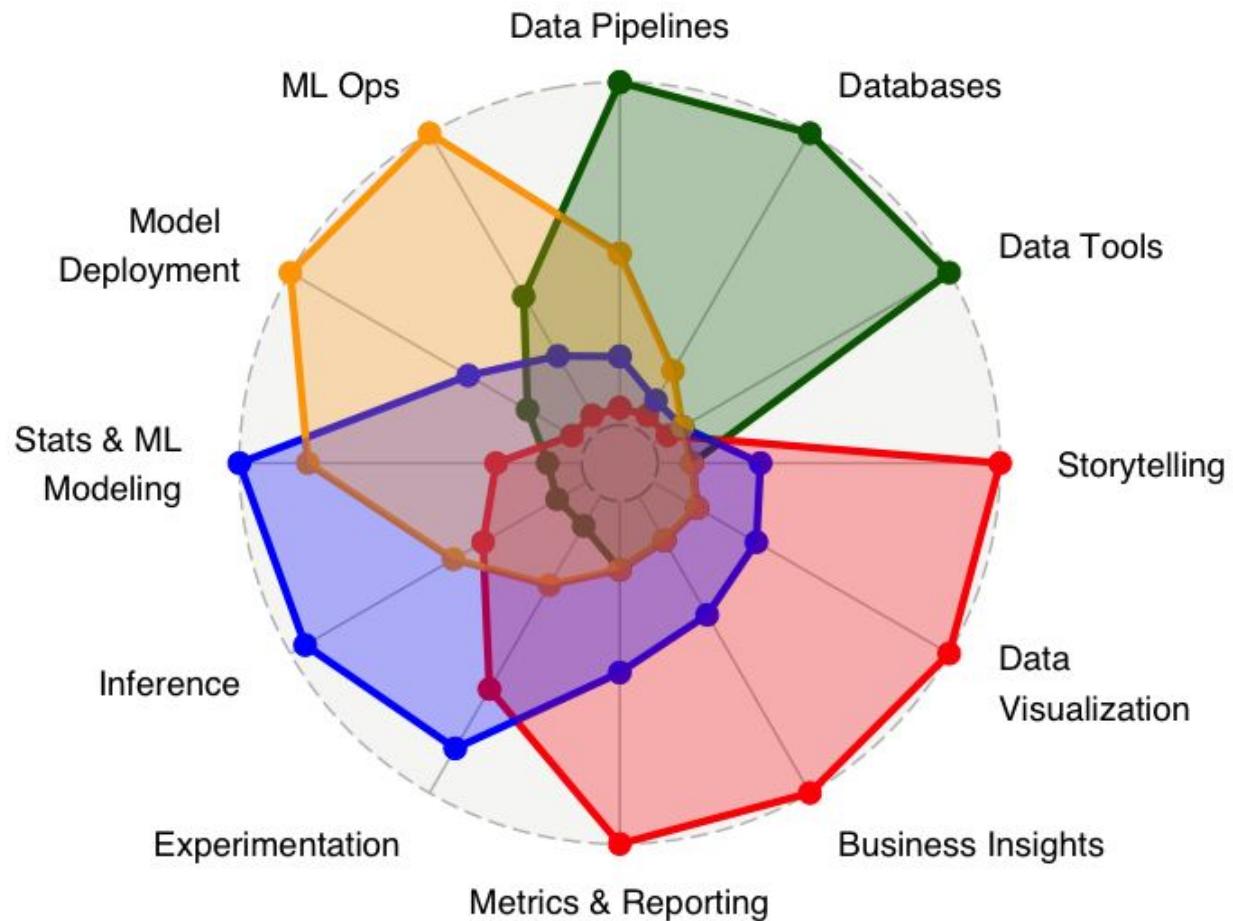
Lo que yo pensaba:

- Disponibilizar data para ser interpretada/analizada
- Mover datos de un lugar a otro

Lo que es:

- Armar pipelines de datos
- Ingeniería de Software
- DevOps
- SQL
- Modelación y mucho más ...

- Data Engineers
- Data Analysts
- Data Scientists
- ML Engineers





FULL STACK DATA ENGINEER



¿Qué es la ingeniería de datos?

Lo que yo pensaba:

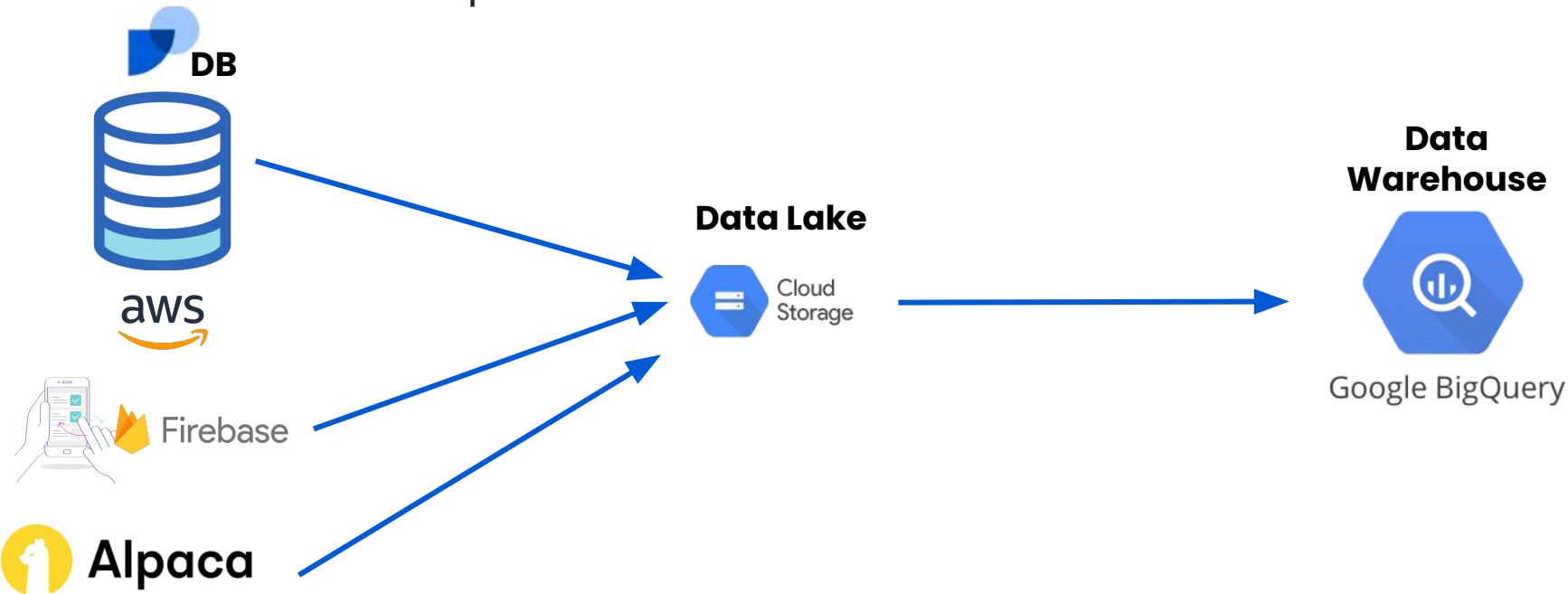
- Disponibilizar data para ser interpretada/analizada
- Mover datos de un lugar a otro

Lo que es:

- **Armar pipelines de datos**
- Ingeniería de Software
- DevOps
- SQL
- Modelación y mucho más ...

¿Qué hace un pipeline de datos?

Copiar datos desde diferentes fuentes a un Data Lake y luego a un Data Warehouse donde se puede hacer analítica



¿Para qué necesitamos un Data Warehouse?

- **Para realizar análisis de datos**
- Tomar decisiones en base a datos
- Consolidar información de múltiples fuentes
- Análisis histórico de data

¿Por qué necesitamos un Data Warehouse?

¿Por qué no hacemos analítica directo en la base de datos (BD) de producción?

1. Las DB de apps (PSQL, MongoDB, etc.) son transaccionales.
2. Liberar recursos, usar BD de producción solo para la operación.
3. Una DB Postgres no está diseñada para hacer analítica, una BD como BigQuery sí (DB columnar)
4. Vistas o tablas intermedias van en el Data Warehouse

¿Por qué necesitamos un Data Warehouse?

- **OLAP: Online Analytical Processing**
 - Pocas consultas pesadas
 - Consultas con agrupaciones y muchos JOINs
 - Enfocado al análisis de datos
 - Lectura de datos optimizada
- **OLTP: Online Transactional Processing**
 - Muchas consultas pequeñas
 - Maneja concurrencia – ACID (atomicity, consistency, isolation, durability)
 - Alto número de usuarios concurrentes

Data Warehouse: Batch/Streaming

- **Batch (Mayoría de los casos: Fintual)**

- Analítica no crítica o no se realizan acciones inmediatas con data
- Mover grandes volúmenes de datos
- Analítica agregada y análisis histórico
- Barato
- Puede desincronizarse

- **Streaming (ALeRCE)**

- Acciones inmediatas con data
- Costoso
- Orientada a eventos

Data Warehouse: ETL/ELT

- **ETL (Extract Transform Load)**
 - Paradigma antiguo, cuando almacenamiento era caro
- **ELT**
 - Paradigma actual, almacenamiento es barato (GCS)
 - Reconstruir data histórica
 - Se guardan todos los datos crudos
 - Diógenes digital (Mejor para ML)

Data Warehouse: cloud/on-premise

- **Cloud (Fintual)**

- Más barato al principio, pero caro con el tiempo
- Inercia de quedarse, amarra
- Mejor para probar ideas
- Mejor para apps con flujos de clientes inciertos

- **On-premise**

- Costoso al principio, pero costo fijo
- Espacio físico

- **Híbrido (ALeRCE)**

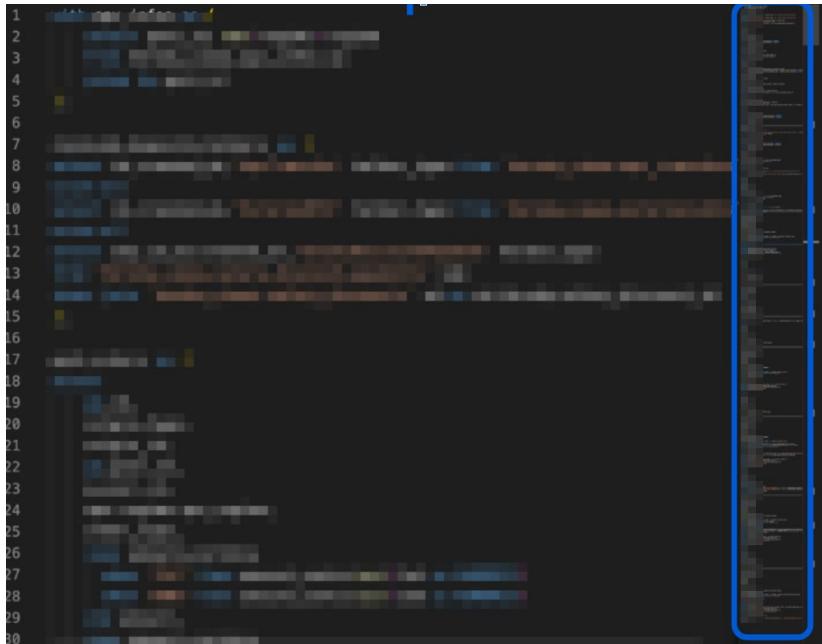
- Difícil encontrar profesionales que manejen ambos paradigmas
- Costos balanceados

DE en la Industria: Pipeline de Fintual

Pipeline de Fintual

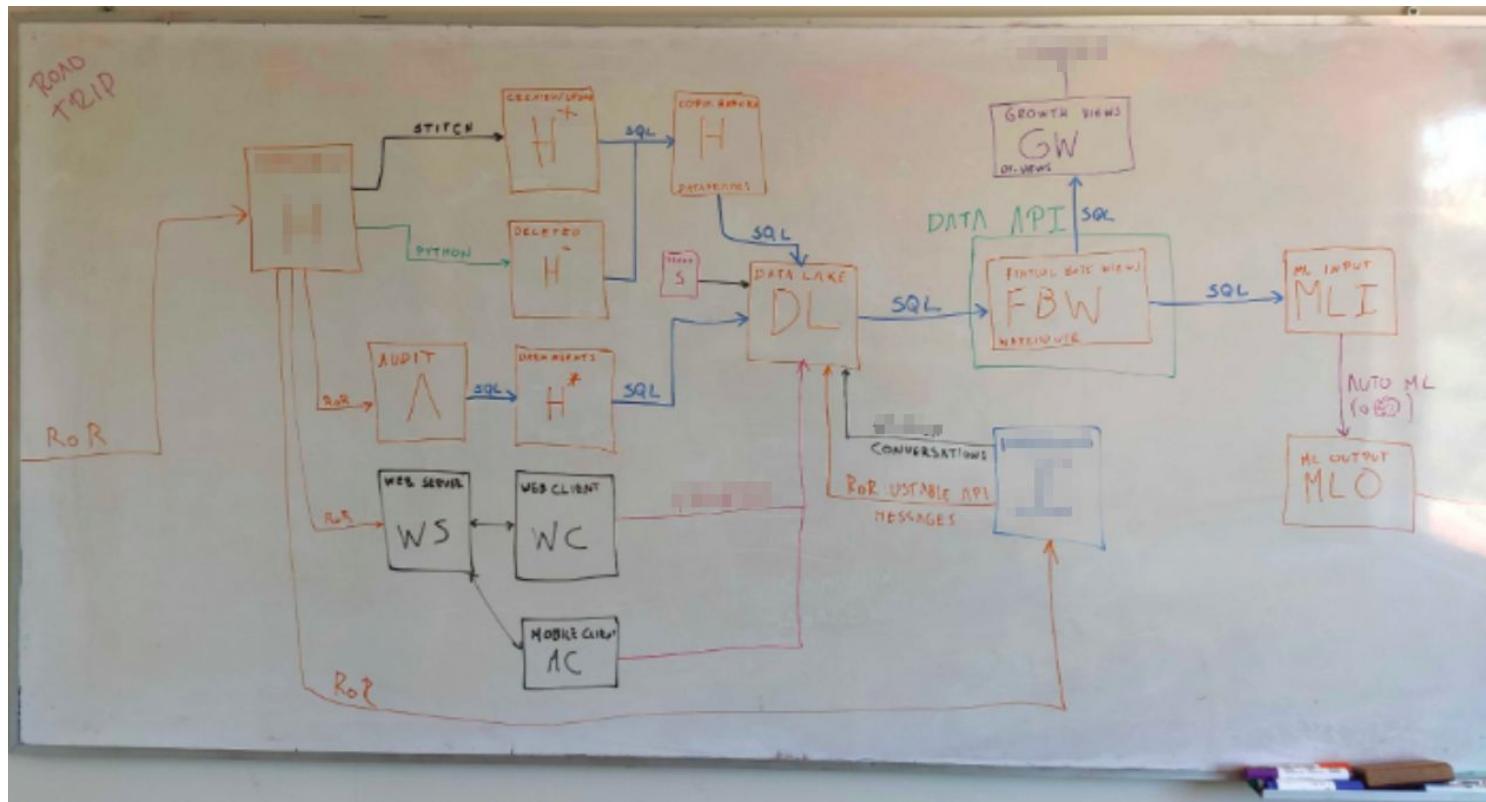


Fintual: En un pasado no tan lejano



- Apurados por sacar cosas como start-up
- Difícil de mantener
- Propenso a errores (de análisis, los datos de la app viven en otra DB)

Fintual: En un pasado no tan lejano



Fintual: Problemas con el legacy

- Lógica en muchos lugares: Stitch, Cloud Functions, scripts locales, Scheduled Queries, ...
- ex-repo (legacy) no era una fuente de verdad

Fintual: Problemas con el legacy

- Lógica en muchos lugares: Stitch, Cloud Functions, scripts locales, Scheduled Queries, ...
- ex-repo (legacy) no era una fuente de verdad



Figura 1: retrato del antiguo pipeline. Óleo sobre tela.

Fintual: El pipeline de datos



@teresa



@alanezz



@estebandido



@luciana

Fintual: El pipeline de datos

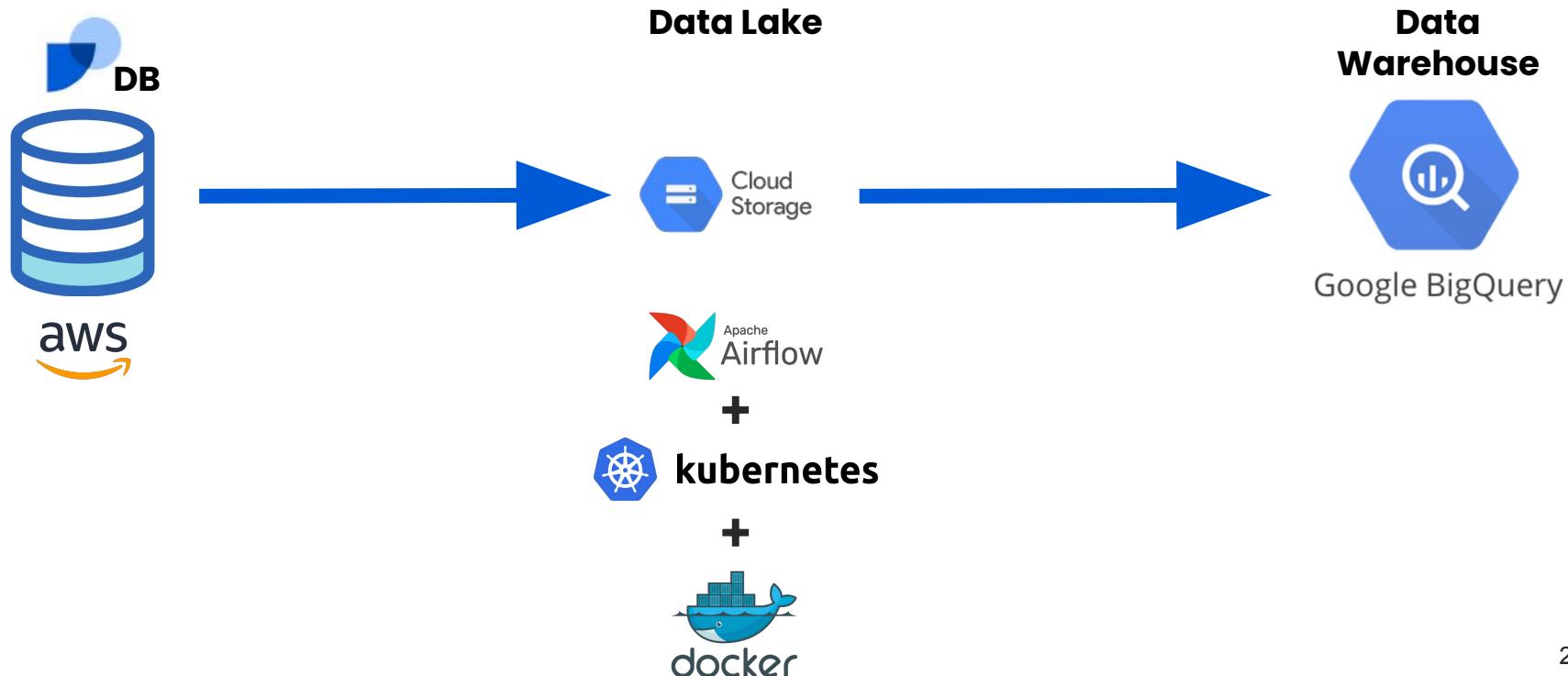


Fintual: El pipeline de datos

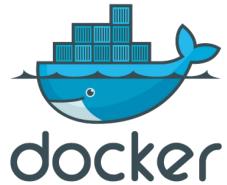
SEQUOIA 



Fintual: El pipeline de datos (Batch)



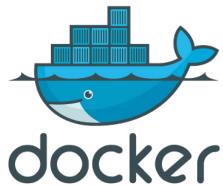
Fintual: Lo que construimos



kubernetes



Fintual: Lo que construimos



docker

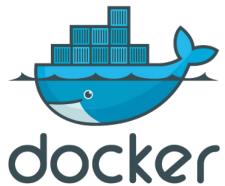


kubernetes



Tenemos *jobs* para
cada una de nuestras
tareas: copiar y limpiar
datos, crear vistas, etc.

Fintual: Lo que construimos



kubernetes

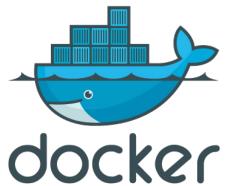


El orden y la frecuencia
de estos jobs se
orquesta con Airflow.

Cloud Composer



Fintual: Lo que construimos

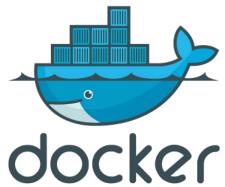


kubernetes



Los recursos para ejecutar nuestros jobs los pedimos en un cluster de Kubernetes

Fintual: Lo que construimos



kubernetes



Todo nuestro código
está en GitHub, y
pasa a producción
automáticamente
con pipelines de
CI/CD

Fintual: Lo que construimos

Armamos un repo basándonos en buenas prácticas devs



QA

Context

Este repositorio es una colección de scripts y herramientas para automatizar las pruebas y el desarrollo de software. Los scripts están escritos en Python y se ejecutan en un entorno de desarrollo local.

Changelog

- Se agrega un `README.md` sobre los tests, con los punteros a la documentación oficial.
- Se agrega el test [REDACTED], que es análogo a [este test](#).

QA

- Partimos por setear la variable de entorno de GCP.
- Corremos `make run` para actualizar las tablas en entorno de staging.
- Corremos `make test` y comprobamos que todos los tests pasen.

Para una prueba adicional, podemos cambiar el archivo [REDACTED] por la siguiente consulta:

```
WITH [REDACTED]
```

Tests (Infra y data)

Review required
At least 1 approving review is required by reviewers with write access. [Learn more](#).

1 pending reviewer

No unresolved conversations
There aren't yet any conversations on this pull request. [View](#)

All checks have passed
2 successful checks

Lint and Test / black-linter (pull_request) Successful in 12s [Required](#) [Details](#)

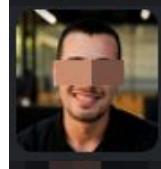
Lint and Test / test (pull_request) Successful in 2m [Required](#) [Details](#)

Merging is blocked
Merging can be performed automatically with 1 approving review.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Fintual: Dolores que resuelve nuevo pipeline

¡Tenemos Tests! Y los errores no pasan desapercibidos



estebanido 6 days ago

estaba revisando los tests y me di cuenta que me
equivoque [REDACTED]

Fintual: Dolores que resuelve nuevo pipeline

Podemos levantar toda nuestra infraestructura en 2 horas o menos, sino le devolvemos su dinero: Cambios de origen en DB son simples

Antes



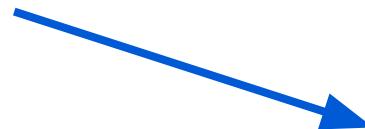
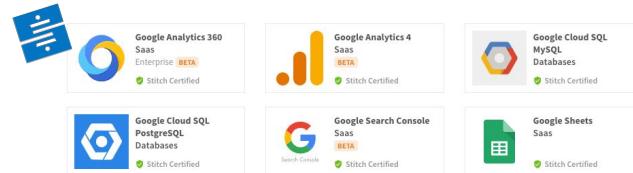
Después



Fintual: Dolores que resuelve nuevo pipeline

Antes

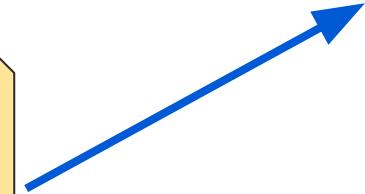
Fuentes de datos limitadas



Después

- Podemos crear jobs a la medida (ej. traer APIs externas)
- Creamos un job que corre un proyecto de DBT y crea vistas automáticamente

Crear vistas y consultas en BigQuery requiere muchos pasos manuales





- Data Warehouse se construye con código versionado en GitHub
- Tal como uno pasa a producción una aplicación, nosotros pasamos a producción nuestro Data Warehouse
- Se crean vistas con un solo comando

Data Warehouse





EXPLORER

DBT-BQ

- check.yml
- TEST_TEMPLATE.md
- sts
- base
 - c.sql
 - c.sql
 - c.sql
 - c.sql
 - schema.yml
- composite
 - caja_inflows.sql
 - docs.md
 - schema.yml
- views
 - docs.md
 - .sql
 - schema.yml

models > caja > views > movements_caja.sql

You, 2 weeks ago | 1 author (You)

```
1 WITH AS (
2   SELECT
3
4
5
6
7
8
9
10 FROM {{ ref('') }})
11 UNION ALL
12
13 SELECT
14
15
16
17
18
19
20
21 FROM {{ ref('') }})
22 UNION ALL
23
24 SELECT
25
26
27
28
29
30
31
32
33
34 FROM {{ ref('') }})
35
36 SELECT * FROM
```

DBT - docs

The screenshot shows the dbt documentation interface. On the left, there's a sidebar with 'Overview' (highlighted), 'Project' (selected), and 'Database'. Under 'Sources', there are three folder icons. Under 'Projects', there is one folder icon. The main content area has a title 'Google de datos - Chile'. It includes a search bar at the top right labeled 'Search for models...'. Below the title, it says 'Esta documentación se enfoca en los modelos de usuarios y movimientos de Chile. Para ver los de México visitar el [Catálogo de México](#)'. It also mentions 'Equipo responsable: [Squad Data](#)'. A section titled 'Tablas documentadas' lists tables under 'CARPETA' and 'TABLAS'. The tables are represented by horizontal bars of varying lengths, indicating their size or complexity. At the bottom, there's a 'Guía de uso' section with a bulleted list:

- El buscador en la parte superior permite buscar por palabras dentro de la consulta sql, por columna, por descripción, por nombre y por tags
- La vista de proyecto de la izquierda permite ver las tablas según la carpeta interna del proyecto, análogo a los datasets de BigQuery
- En cada modelo se pueden revisar las vistas de las que dependen y que vistas dependen del modelo revisando los campos `depends on` y `referenced by`, o viendo la `graph view` en el botón de la parte inferior derecha



@panchito 40

DBT - tests

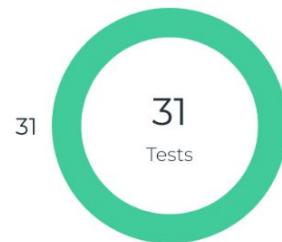
dbtbq | Dev ⓘ

Filter by ▾

▴

Last 7 Days

Test results breakdown



Tables health



- Passed
- Warning
- Failed
- Error

Monitored tables



- Monitored
- Unmonitored

Freshness

0

0 | 0 | 0

Volume

0

0 | 0 | 0

Schema changes

0

0 | 0 | 0

dbt tests

31

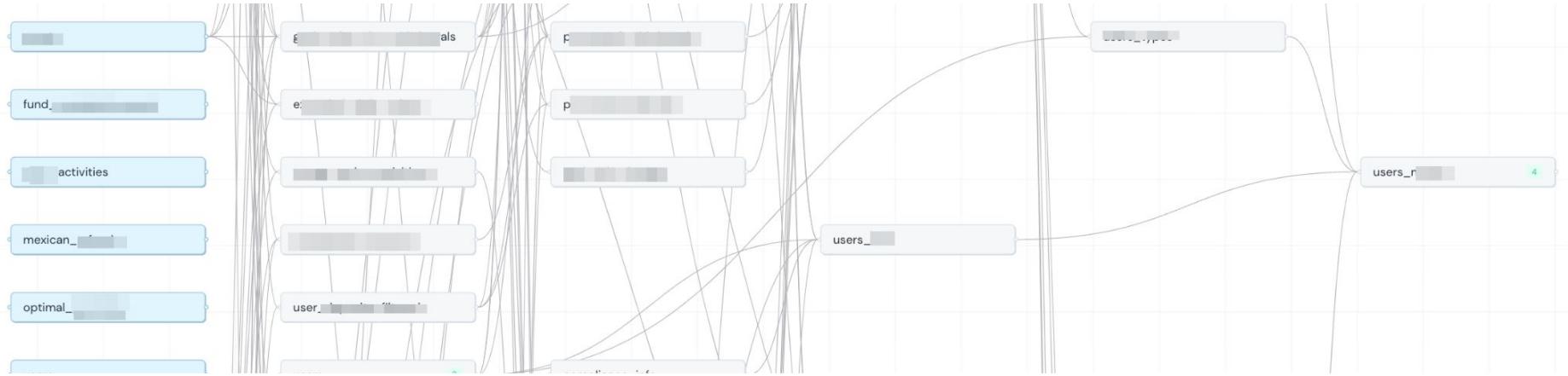
31 | 0 | 0

Anomalies

0

0 | 0 | 0

DBT - Data Lineage



Fintual: Por qué es bkn el pipeline de datos

Docker + Kubernetes:

- Todo el código de data está versionado y hay claridad sobre su ubicación
- Se puede correr cualquier tipo de código, incluso si necesita recursos especiales
- CI/CD automático: un merge a main y tenemos todo en producción
- Usamos K8 que nos permite escalar procesos horizontal y verticalmente

**Si algo cambia, lo
sabemos**



Data Warehouse



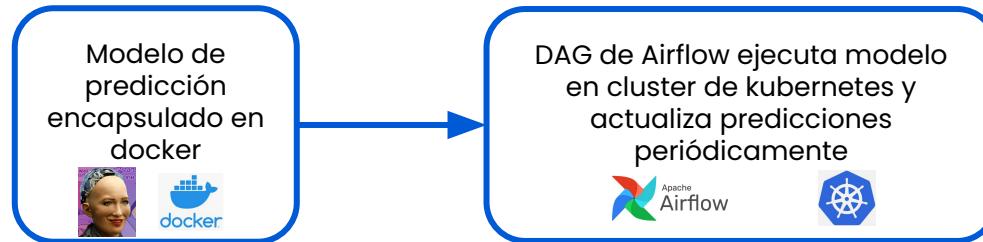
Fintual: Por qué es bkn el pipeline de datos

Desarrollar y hacer pruebas es un agrado (DBT)

- Se eliminaron los pasos manuales incluso para hacer pruebas
- Con un comando generamos tablas de prueba para hacer QA más rápido y mejor

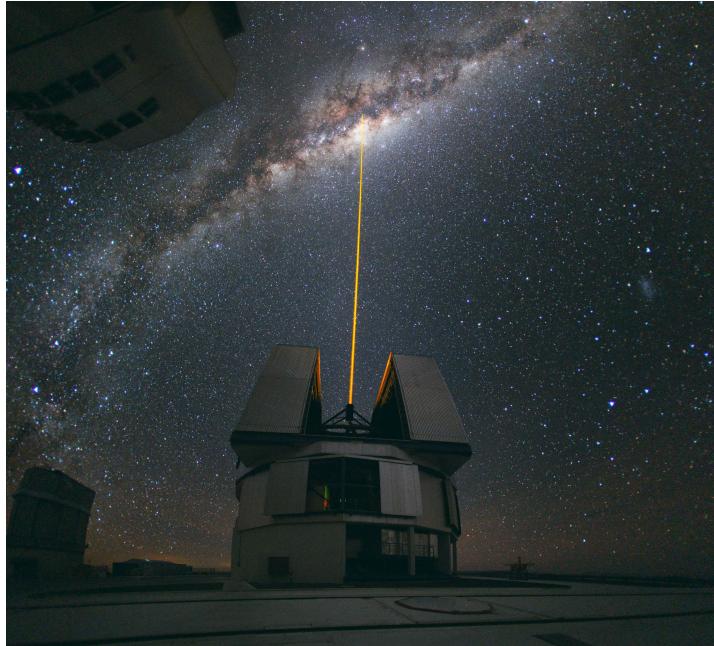


Fintual: Ejemplo - Predicciones en BigQuery



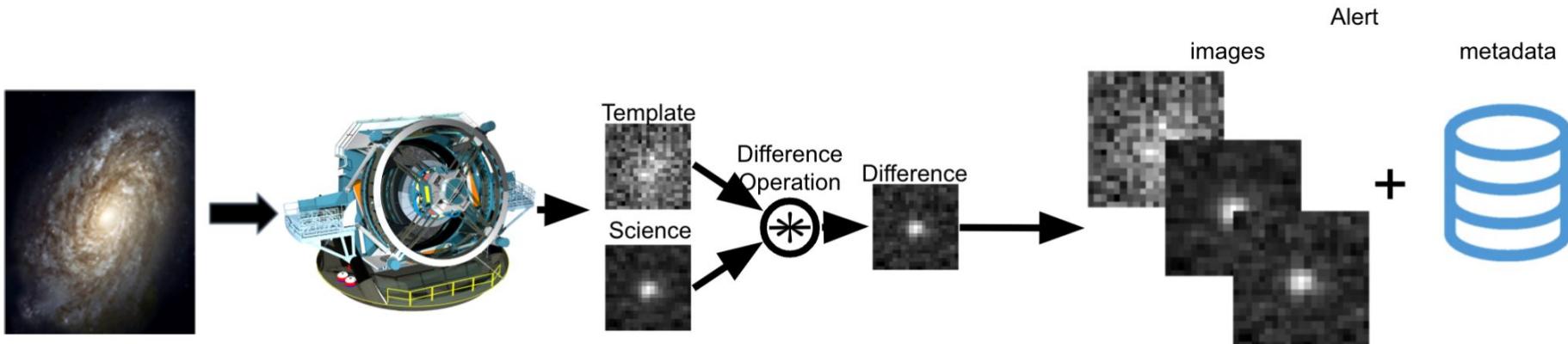
Facilita ML Engineering

DE más allá de la Industria Pipeline de ALeRCE



ALeRCE: Sistema basado en eventos

Eventos en forma de alertas astronómicas



ALeRCE

ZTF ~1MM y LSST ~10MM de alertas por noche.



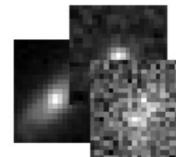
ZTF



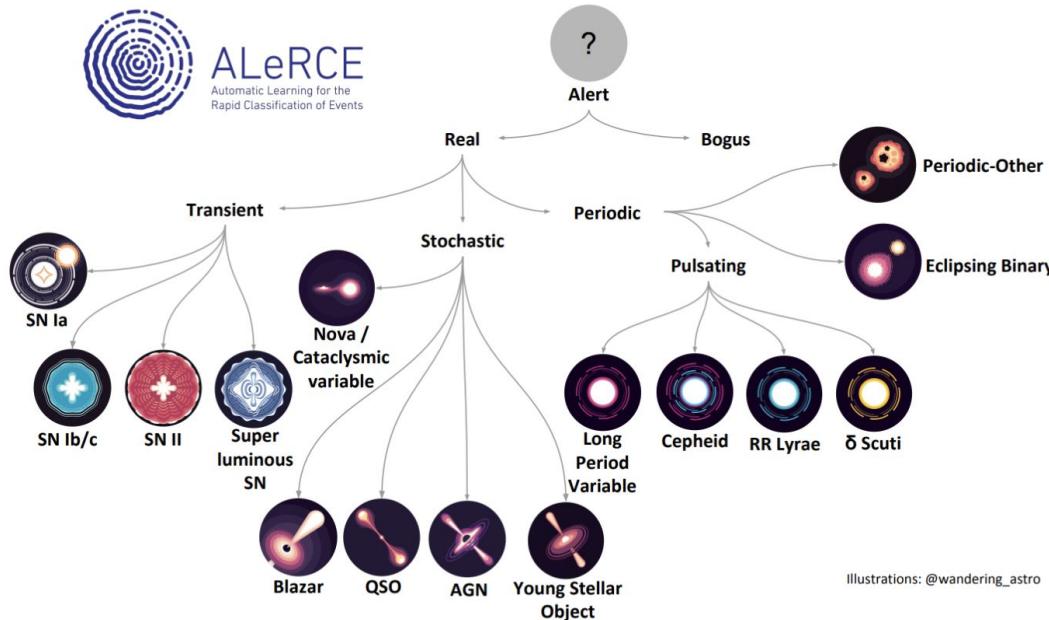
Vera C. Rubin

ALeRCE

¿Qué se hace con las alertas?

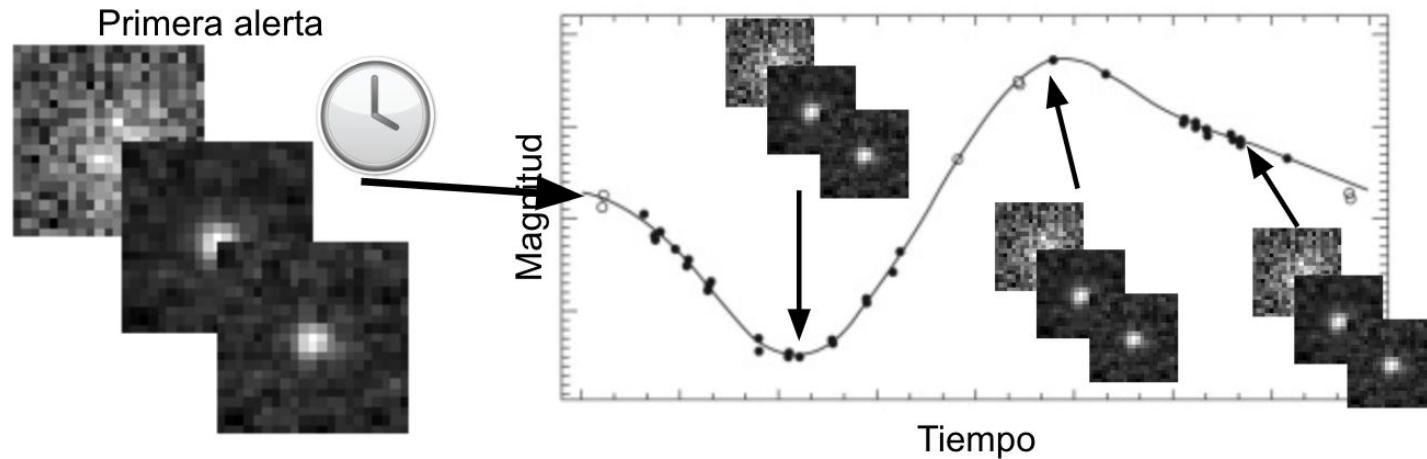


¡Clasificarlas!

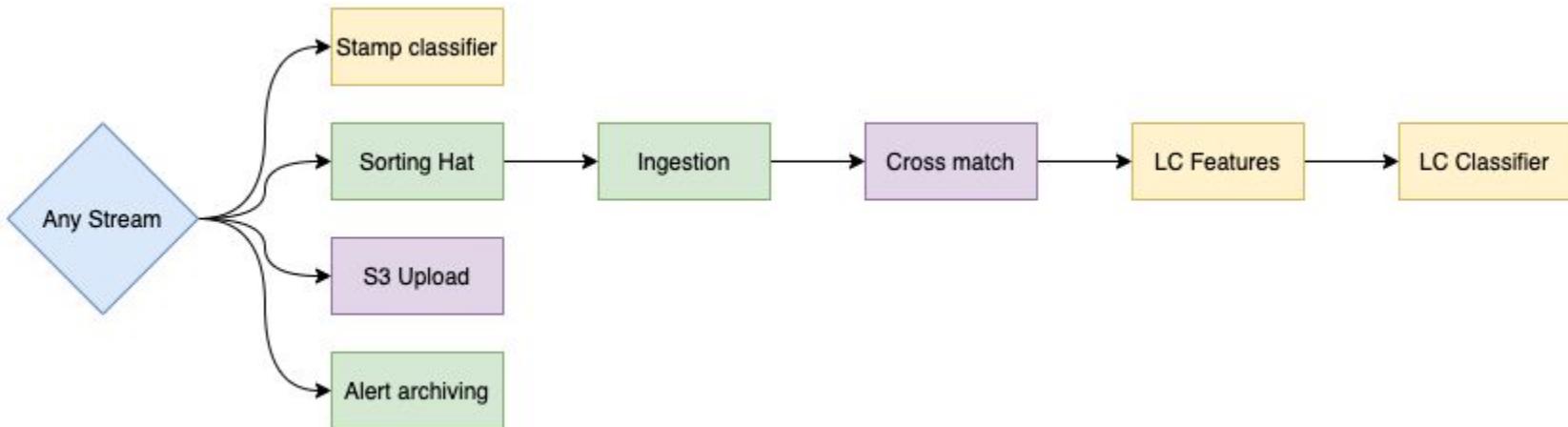


ALeRCE

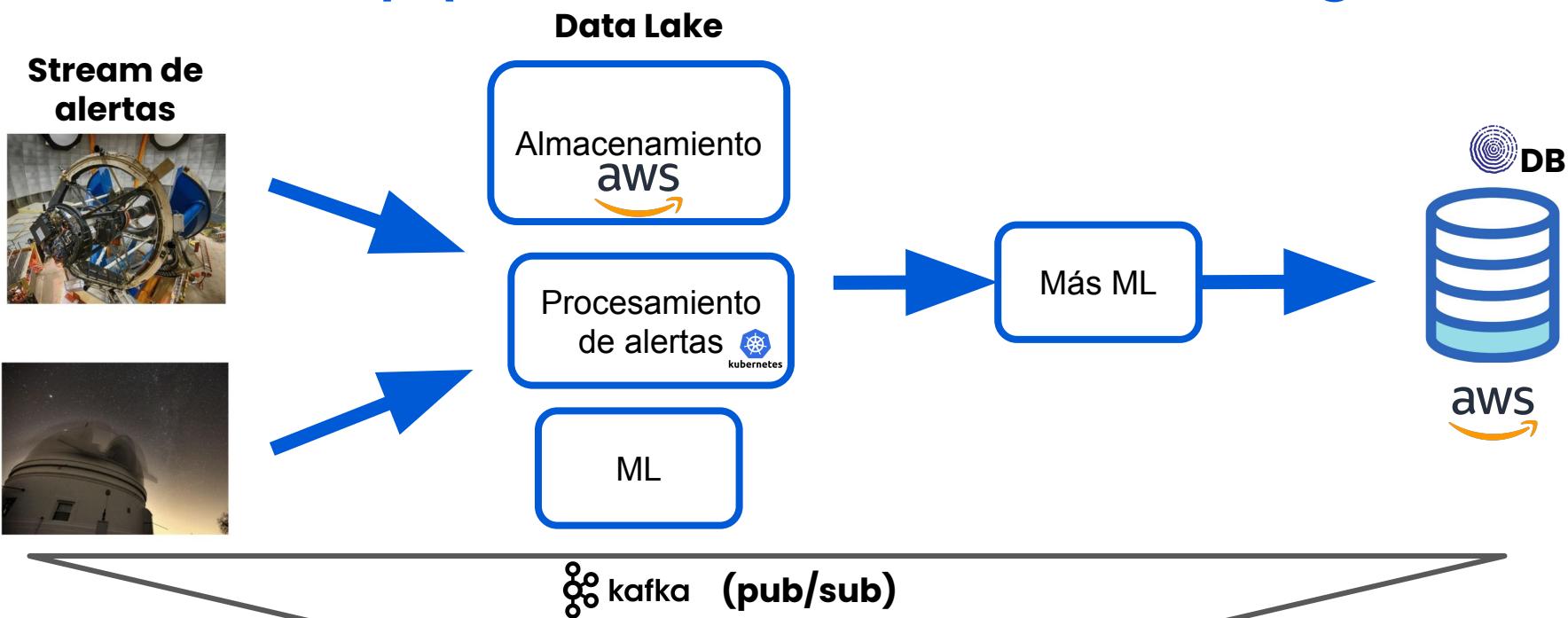
Clasificadores basados en imágenes y series de tiempo (curvas de luz)



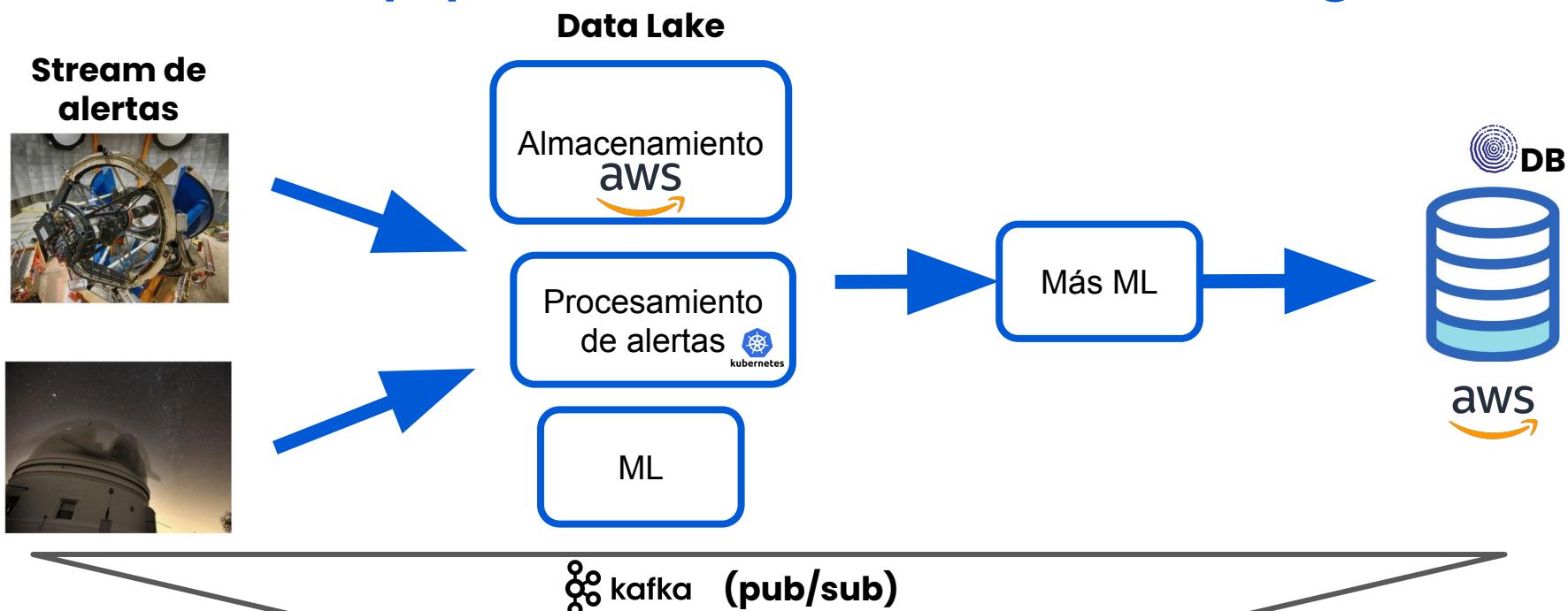
ALeRCE: El pipeline de datos (Streaming)



ALeRCE: El pipeline de datos (Streaming)



ALeRCE: El pipeline de datos (Streaming)



- Todo en la nube
- Queremos dejar solo apps en la nube y el resto on-prem

ALeRCE

<https://snhunter.alerce.online/>



Apache
Airflow

¿Que es Airflow?

“Apache Airflow™ is an open-source platform for developing, scheduling, and monitoring batch-oriented workflows”

¿Que es Airflow?

“Apache Airflow™ is an open-source platform for developing, scheduling, and monitoring batch-oriented workflows”

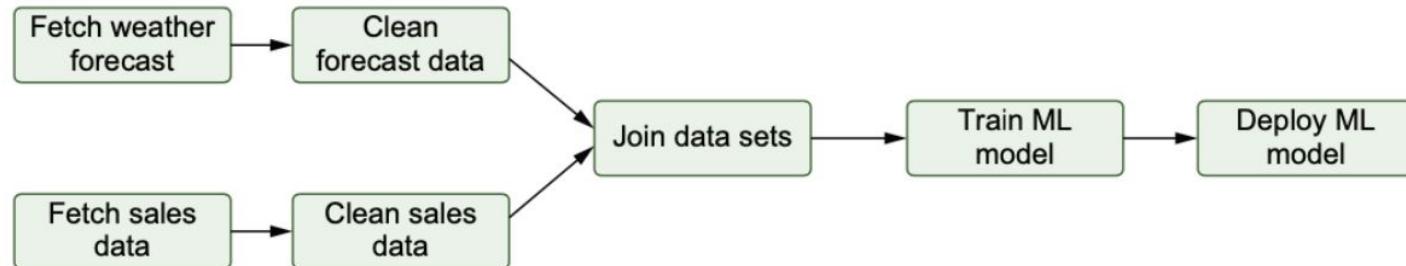
Framework basado en python para orquestar ejecución de código (jobs) de forma calendarizada y coordinada

Estas tareas son encapsuladas en operadores



¿Qué es Airflow?

Se basa en la idea de definir un DAG (grafo dirigido acíclico) de tareas; este DAG se define mediante código Python



¿Cómo se ve un DAG de airflow?

```
dags > 🐍 example-python-op.py > ...
1  import os
2  from datetime import datetime
3
4  from airflow import models      Import "airflow" could not be resolved
5  from airflow.operators.python_operator import PythonOperator
6
7  with models.DAG(
8      dag_id="example_pythonop",
9      schedule_interval="@once", # "5 8 * * *" (cron format)
10     start_date=datetime(2022, 1, 1),
11     catchup=False,
12     tags=["example"],
13 ) as dag:
14
15     def return_constant_1():
16         print("Primer Python Operator")
17         return 1
18
19     def return_constant_2():
20         print("Segundo Python Operator")
21         return 2
22
23     pythonop_example_1 = PythonOperator(
24         task_id="pythonop_example_1",
25         python_callable=return_constant_1,
26     )
27
28     pythonop_example_2 = PythonOperator(
29         task_id="pythonop_example_2",
30         python_callable=return_constant_2,
31     )
32
33     pythonop_example_1 >> pythonop_example_2
34
```

Airflow

Instalación local muy sencilla

```
pyenv virtualenv 3.11 airflow-example-3
pyenv shell airflow-example-3
pip install pandas pyarrow apache-airflow psycopg2
mkdir airflow-example-3 && cd airflow-example-3
export AIRFLOW_HOME=~/airflow-example-3
airflow db init # y desactivamos ejemplos load_examples = False

airflow users create --role Admin --username admin --email admin
--firstname admin --lastname admin --password admin

airflow standalone
```

Airflow

Veamos ejemplo

```
cp -r ~/airflow-example-template/dags ~/airflow-example-3  
code .
```

```
airflow standalone
```

- Dependencias entre DAGs con triggers
- Scheduling <https://crontab.guru/>

Airflow

Creemos un pipeline de datos



Extraer de
nyc.gov

Transformamos
de parquet a
csv



Cargar los
datos en DB
local psql

Airflow

1. Preparar la DB local

- a. psql -U postgres
- b. CREATE DATABASE nyc_taxi3;
- c. \c nyc_taxi_3
- d. CREATE TABLE nyc_trip_table_3; **(COMANDO COMPLETO EN COMENTARIOS)**
- e. SELECT * FROM nyc_trip_table_3;

Airflow

Creemos un pipeline de datos



Extraer de
nyc.gov

Airflow

Creemos un pipeline de datos



Extraer de
nyc.gov



Transformamos
de parquet a
csv

Airflow

Creemos un pipeline de datos



Extraer de
nyc.gov



Transformamos
de parquet a
csv



Cargar los
datos en DB
local psql

- Modificar DAG con sufijos número correcto, reiniciar airflow y correr DAG 1
- `SELECT COUNT(*) FROM nyc_trip_table_3;`

Airflow en la nube

Cloud Composer para GCP

The screenshot shows the Google Cloud Platform interface for Cloud Composer. At the top, there is a navigation bar with the Google Cloud logo, a dropdown for BigQuery Data, a search bar, and a 'Search' button. Below the navigation bar, the 'Composer' tab is selected, showing the 'Environments' section. A table lists environments, with one row selected: 'data-cloud-composer-2-5'. This row includes columns for State (green checkmark), Name (data-cloud-composer-2-5), Location (us-central1), Composer version (2.3.5), Airflow version (2.5.3), Creation time (8/4/23, 11:49 AM), Update time (4/29/24, 5:47 PM), and Airflow webserver (link). A blue arrow points from the 'data-cloud-composer-2-5' link in the table to the 'DAGs' tab below. The 'DAGs' tab is highlighted, and the page title 'data-cloud-composer-2-5' is displayed. Below the title, there are buttons for All (26), Active (23), and Paused (3), and a 'Filter DA' button. At the bottom, there are filters for DAG, Owner, and Runs, along with a toggle for 'airflow_monitoring'.

State	Name	Location	Composer version	Airflow version	Creation time	Update time	Airflow webserver
<input type="checkbox"/>	data-cloud-composer-2-5	us-central1	⚠ 2.3.5	2.5.3	8/4/23, 11:49 AM	4/29/24, 5:47 PM	Airflow

DAGs

data-cloud-composer-2-5

All 26 Active 23 Paused 3

Filter DA

DAG Owner Runs

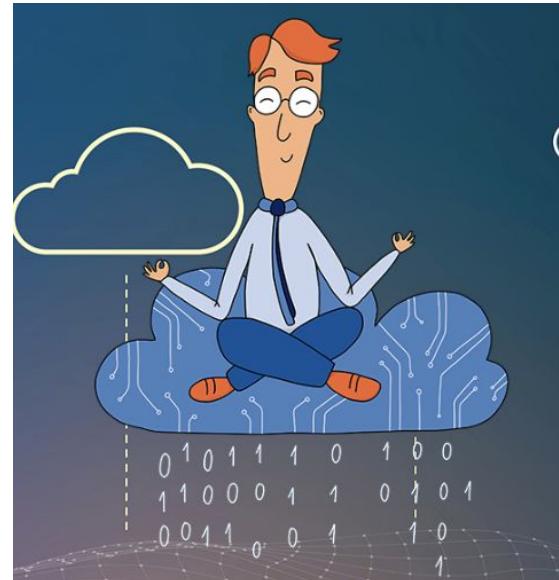
airflow_monitoring airflow 4647 3

Airflow en la nube

¿Por qué usar la nube?

Airflow es complejo de manejar, requiere levantar diferentes tipos de instancias que deberían vivir separadas.

- Scheduler
- Web Server: UI
- Worker
- DB
- Triggerer

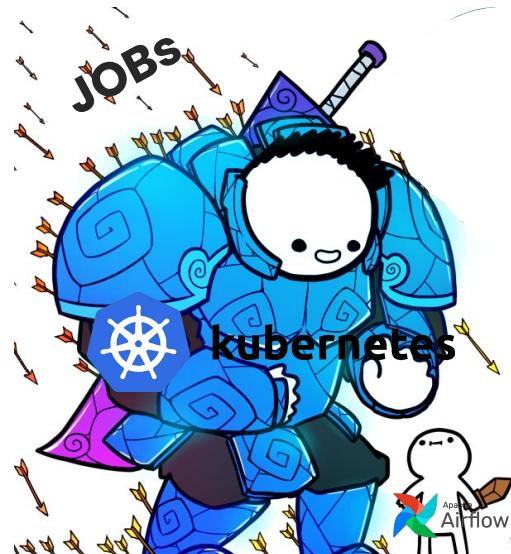


Airflow Fintual



Airflow – Docker

Si bien Airflow tiene un operador para correr código Python, los cálculos pesados no se deberían realizar en Airflow, para liberar carga al scheduler. Por ejemplo un cluster de Kubernetes



Airflow – Docker

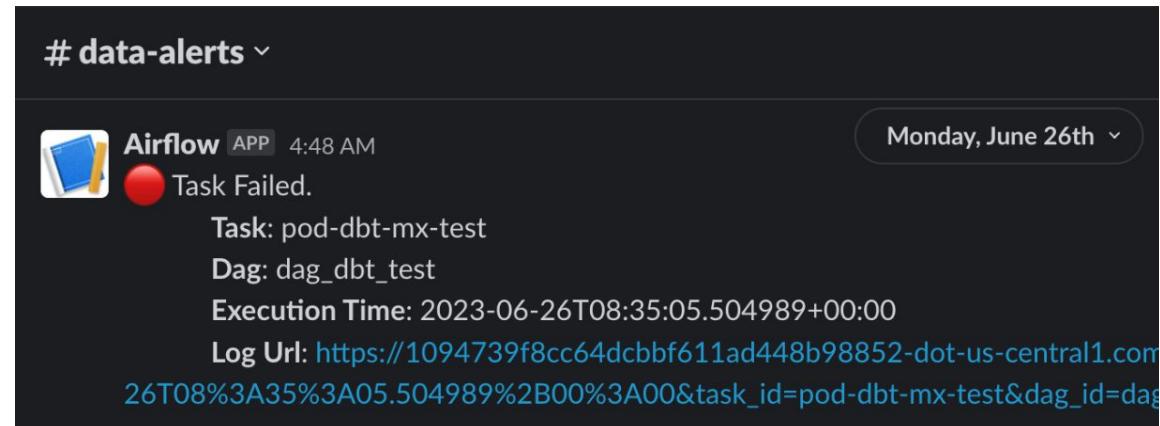
Para ejecutar código en kubernetes desde airflow usamos el **KubernetesPodOperator**, que ejecuta contenedores de docker

Docker: herramienta que nos permite encapsular código para luego ser ejecutado en cualquier máquina.

Airflow

Otras recomendaciones:

- Tests a DAGs
- Style lints
- CI/CD
- Alert task failing



Resumen

Partes importantes de un sistema de DE

- Tests en el DataWarehouse
- Profesionalizar el DE como si fuera SE
- Versionar código - git - PRs!
- Tener en mente las prioridades del negocio y el \$ al tomar decisiones

Stack de tecnología para DE

- Airflow
- DBT
- Kafka
- Docker
- Kubernetes
- BQ
- **SQL**
- **SQL**
- **SQL**

Donde aprender más de Ingeniería de datos?

<https://github.com/DataTalksClub>

The screenshot shows the GitHub profile page for DataTalksClub. At the top, there's a blue circular logo with 'DTC' in white. The profile name 'DataTalksClub' is in bold, followed by the tagline 'The place to talk about data'. Below this, it shows 3.7k followers, a worldwide location, and links to their website and social media accounts. A 'Popular repositories' section displays six repositories:

- data-engineering-zoomcamp** (Public): Free Data Engineering course! Includes a Jupyter Notebook, 23.4k stars, and 5k forks.
- mlops-zoomcamp** (Public): Free MLOps course from DataTalks.Club. Includes a Jupyter Notebook, 10.5k stars, and 2k forks.
- machine-learning-zoomcamp** (Public): Learn ML engineering for free in 4 months! Includes a Jupyter Notebook, 8.5k stars, and 2k forks.
- llm-zoomcamp** (Public): LLM Zoomcamp - a free online course about building a Q&A system. Includes a Jupyter Notebook, 1.4k stars, and 106 forks.
- project-of-the-week** (Public): Learn by doing: DIY project groups at DataTalks.Club. Includes 355 stars and 83 forks.
- awesome-data-podcasts** (Public): A list of awesome data podcasts. Includes 349 stars and 56 forks.

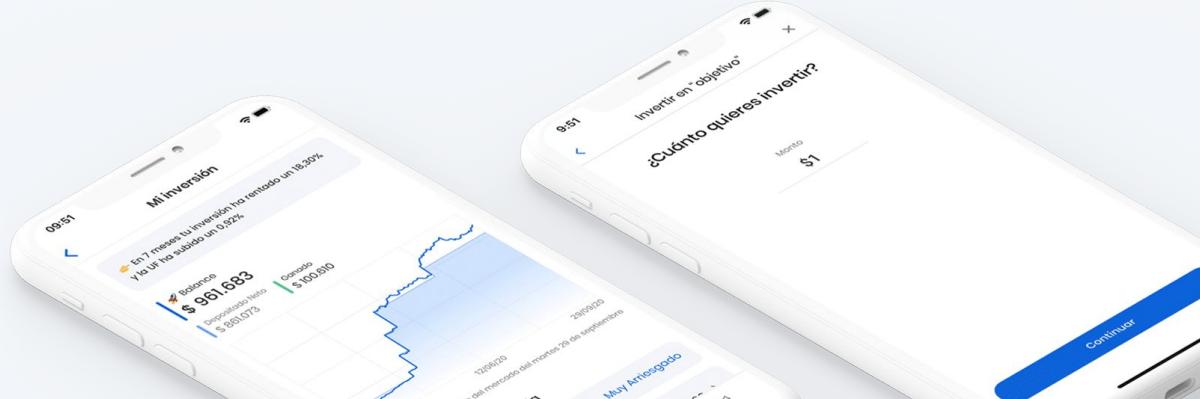


Fintual



ALeRCE
Automatic Learning for the
Rapid Classification of Events

Principios de Ingeniería de Datos: **Fintual** & ALeRCE



reyes.dejong@gmail.com