Data Engineering and the Death of Visual ETL

Wednesday | April 1, 2020
1:00 PM

Stewart Bryson, Red Pill Analytics
@stewartbryson

RED PILL ANALYTICS

Founder & CEO
Red Pill Analytics

twitter: @stewartbryson
medium: @stewartbryson
linkedin: stewartbryson


Your Data Warehouse is elastic.
Shouldn’t your team be?
ELASTIC DELIVERY

Elastic Crew

MULTIFUNCTIONAL TEAM

Each Crew has the necessary players to move from question to insight in a single sprint.

AUTONOMOUS

Whether you have an existing team or not, a Crew works autonomously to solve whatever data problems you assign to it.

SCALABLE

Scale to answer all of your data requirements by adding additional Crews.

RED PILL ANALYTICS

Elastic Surge

SUPPLEMENT YOUR TEAM

Increase your delivery capacity without adding fulltime resources.

SEMI-PERMANENT

Designed as a long-term enhancement to organizations with functioning teams.

RESPOND TO DEMAND

Be more responsive to user demand: increase in size, swap out functional roles, or simply add a new technical skill for a few sprints.

RED PILL ANALYTICS

Data Warehouse

snowflake

BigQuery

Amazon Athena

Autonomous Data Warehouse

Analytics

looker

ORACLE ANALYTICS

Data Studio

Power BI

tableau

Data Engineering / ETL

Fivetran

databricks

kafka

AWS Glue

dbt_

StreamSets

ORACLE Cloud Infrastructure

The Rise of the Data Engineer

Maxime Beauchemin January 20, 2017 · 12 min read

The Rise of the Data Engineer

Maxime Beauchemin January 20, 2017 · 12 min read

"There's a multitude of reasons why complex pieces of software are not developed using drag and drop tools: it's that ultimately *code is the best abstraction* there is for software."

Maxime Beauchemin, *The Rise of the Data Engineer*, Jan 20, 2017

@redpilla

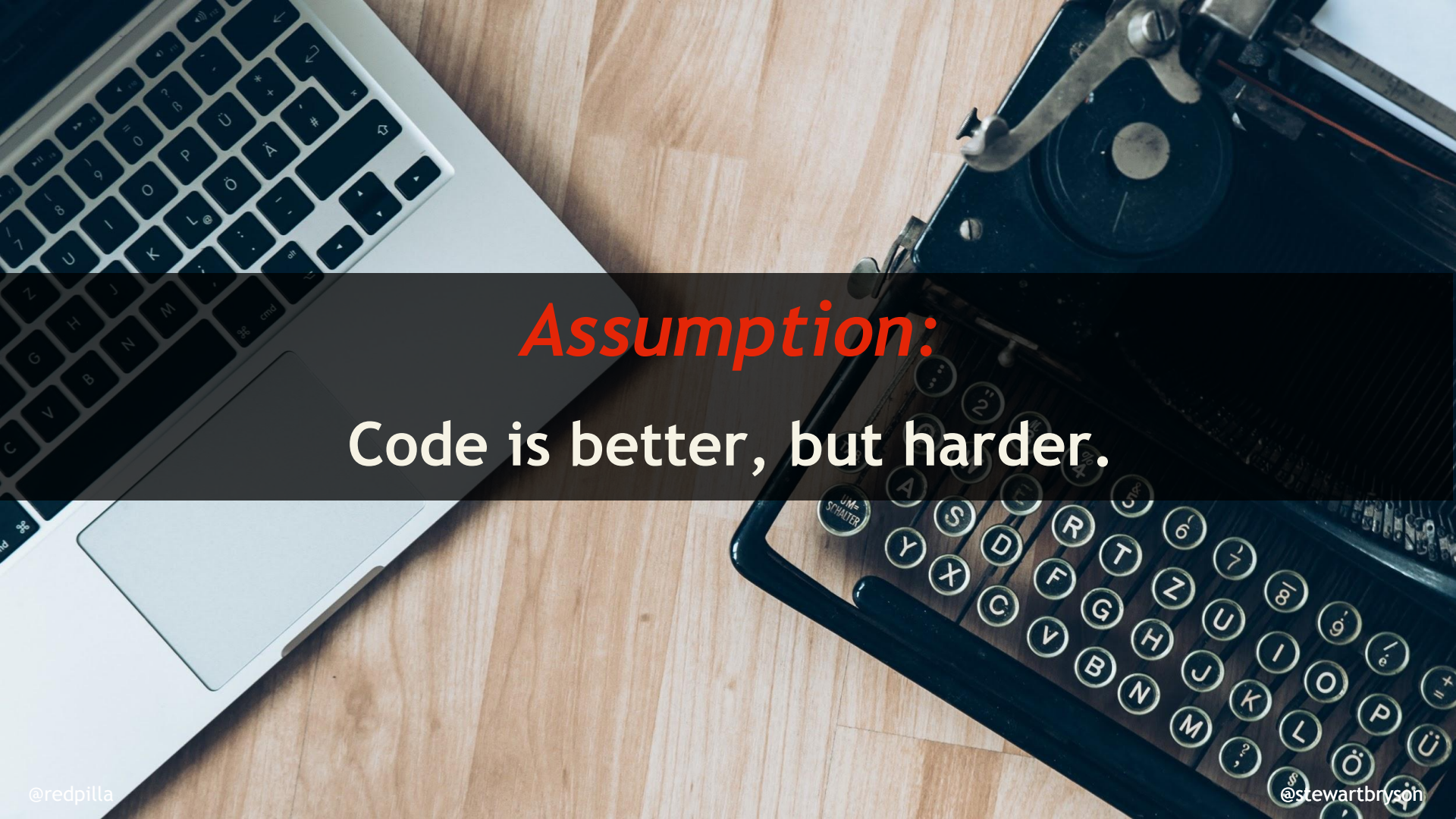
@stewartbryson


What's so
different
about
ETL?
@redpilla
@stewartbryson

Code *versus* Clicks

Source control
CI/CD
Collaboration
Code anywhere

Faster
Easier to start
Any discipline
Business friendly

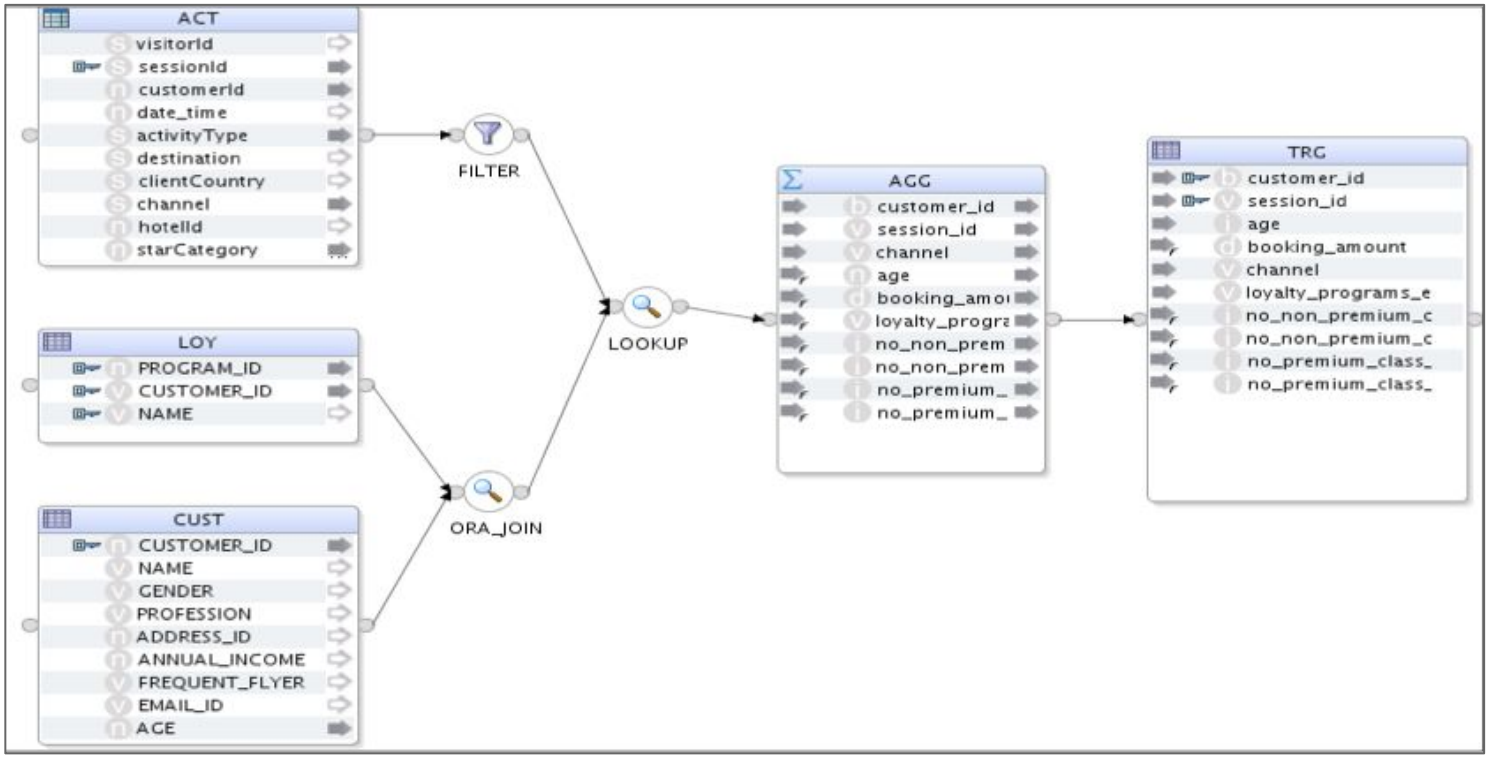

Assumption:
Code is better, but harder.
@redpilla
@stewartbryson

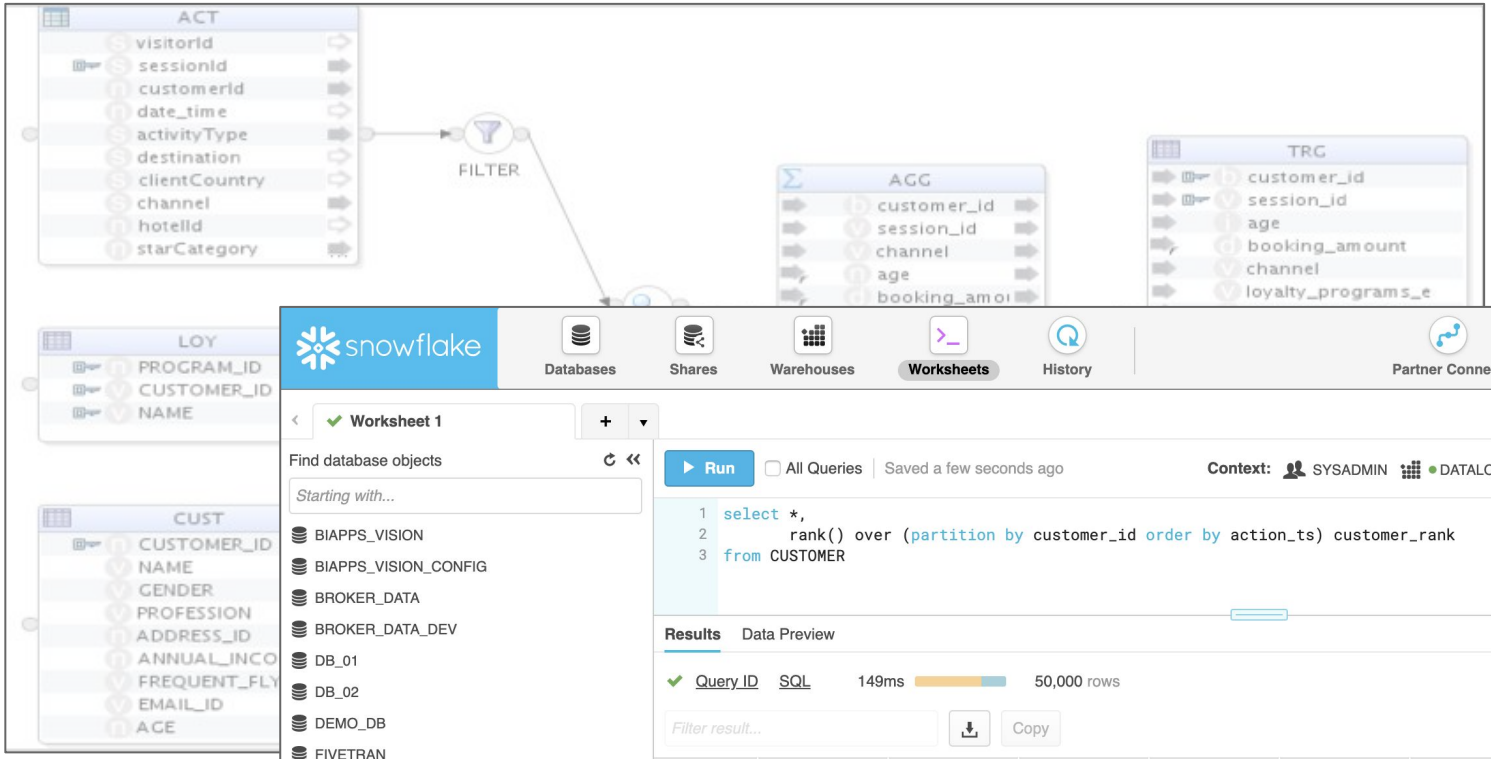
Data Warehouses

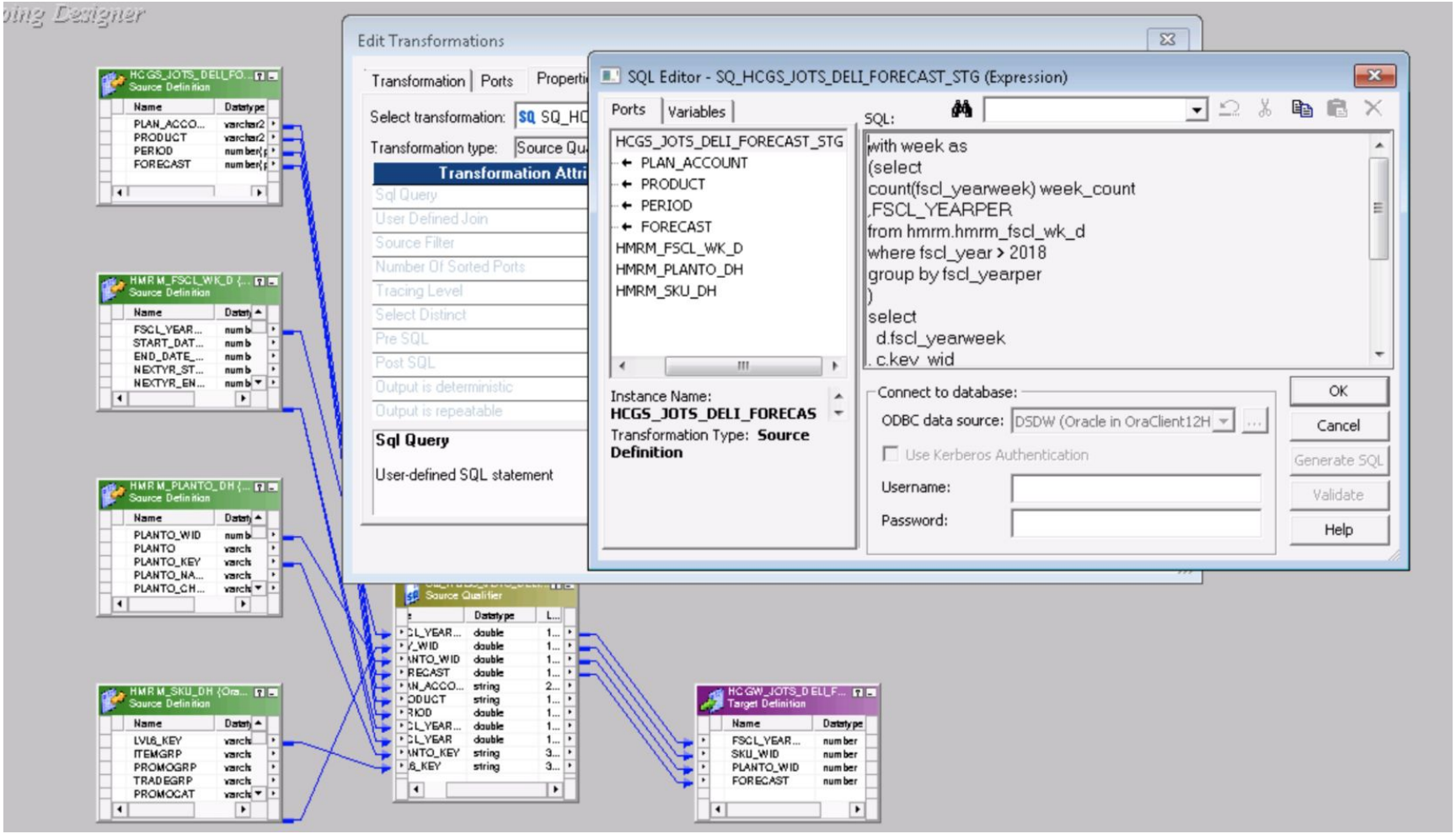
SQL >> GUI (ETL) >> GUI (ELT)

Modern Data Platforms

SQL >> GUI (ETL) >> GUI (ELT) >> Distributed >> SQL

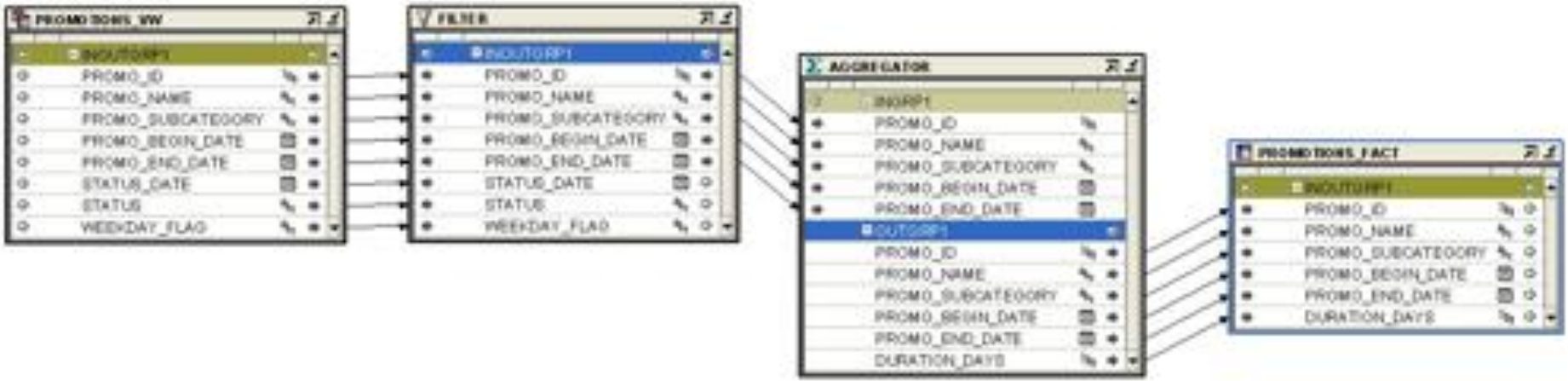

ACT
visitorId
sessionId
customerId
date_time
activityType
destination
clientCountry
channel
hotelId
starCategory
FILTER
LOY
PROGRAM_ID
CUSTOMER_ID
NAME
LOOKUP
CUST
CUSTOMER_ID
NAME
GENDER
PROFESSION
ADDRESS_ID
ANNUAL_INCOME
FREQUENT_FLYER
EMAIL_ID
AGE
ORA_JOIN
AGG
customer_id
session_id
channel
age
booking_amo
loyalty_progra
no_non_prem
no_non_prem
no_premium_
no_premium_
TRG
customer_id
session_id
age
booking_amount
channel
loyalty_programs_e
no_non_premium_c
no_non_premium_c
no_premium_class_
no_premium_class_


ACT
visitorId
sessionId
customerId
date_time
activityType
destination
clientCountry
channel
hotelId
starCategory
FILTER
AGG
customer_id
session_id
channel
age
booking_amo
TRG
customer_id
session_id
age
booking_amount
channel
loyalty_programs_e
LOY
PROGRAM_ID
CUSTOMER_ID
NAME
CUST
CUSTOMER_ID
NAME
GENDER
PROFESSION
ADDRESS_ID
ANNUAL_INCO
FREQUENT_FLY
EMAIL_ID
AGE
snowflake
Databases
Shares
Warehouses
Worksheets
History
Partner Connect
Help
Stewart Bryson
SYSADMIN
Worksheet 1
Find database objects
Starting with...
BIAPPS_VISION
BIAPPS_VISION_CONFIG
BROKER_DATA
BROKER_DATA_DEV
DB_01
DB_02
DEMO_DB
FIVETRAN
FIVETRAN_CLONE
INNIVE_DB
KAGGLE_CYCLESHARE
KITTO
KSCOPE
LOOKER_BLOCKS
MARKETPLACE
Run
All Queries
Saved a few seconds ago
Context:
SYSADMIN
DATALOAD (S)
BROKER_DATA_D...
TRT
1 select *,
2 rank() over (partition by customer_id order by action_ts) customer_rank
3 from CUSTOMER
Results
Data Preview
Open History
Query ID
SQL
149ms
50,000 rows
Filter result...
Copy
Columns
Row
ACTION_TS
TIER
ACCOUNT_ID
STATE
LAST_NAME
FIRST_NAME
ACCOUNT_...
COUNTRY
1 2015-07-20 1... NULL 24476 ME Bezanson Van JXHMCAST... United States
2 2017-02-15 ... 1 29282 UT Baudais Kat Bkx zMTYH... United States
3 2009-06-13 ... 3 5919 SK Remers Idus xvOMLRKLS... Canada
4 2010-06-05 1... NULL 8889 NULL NULL NULL JjmVkihBjFQ... NULL
5 2010-12-27 ... 3 NULL MT NULL NULL NULL United States


Edit Transformations
Transformation
Ports
Select transformation:
Transformation type:
Transformation Attri
Sql Query
User Defined Join
Source Filter
Number Of Sorted Ports
Tracing Level
Select Distinct
Pre SQL
Post SQL
Output is deterministic
Output is repeatable
Sql Query
User-defined SQL statement
SQL Editor - SQ_HCGS_JOTS_DELI_FORECAST_STG (Expression)
Ports
Variables
HCGS_JOTS_DELI_FORECAST_STG
PLAN_ACCOUNT
PRODUCT
PERIOD
FORECAST
HMRM_FSCL_WK_D
HMRM_PLANTO_DH
HMRM_SKU_DH
Instance Name:
HCGS_JOTS_DELI_FORECAS
Transformation Type: Source Definition
SQL:
with week as
(select
count(fscl_yearweek) week_count
,FSCL_YEARPER
from hmrm.hmrm_fscl_wk_d
where fscl_year > 2018
group by fscl_yearper
)
select
d.fscl_yearweek
, c.kev_wid
Connect to database:
ODBC data source:
DSDW (Oracle in OraClient12H
Use Kerberos Authentication
Username:
Password:
OK
Cancel
Generate SQL
Validate
Help
Source Definition
Source Qualifier
Target Definition

```
CREATE OR REPLACE VIEW stage.promotions_vw
AS
SELECT promo_id,
       promo_name,
       promo_subcategory,
       promo_begin_date,
       promo_end_date,
       time_id status_date,
       status,
       weekday_flag
  FROM ( SELECT time_id,
                CASE
                WHEN day_name IN ('Saturday','Sunday') THEN 'N'
                ELSE 'Y'
                END weekday_flag,
                promo_id,
                MAX(promo_name) OVER (partition BY promo_id) promo_name,
                MAX(promo_subcategory) OVER (partition BY promo_id) promo_subcategory,
                MAX(promo_begin_date) OVER (partition BY promo_id) promo_begin_date,
                MAX(promo_end_date) OVER (partition BY promo_id) promo_end_date,
                nvl(status, 'ongoing') status
           FROM stage.promotions_stg partition BY (promo_id)
          right outer JOIN sh.times
                ON time_id=status_date)
 WHERE time_id BETWEEN promo_begin_date AND promo_end_date
```

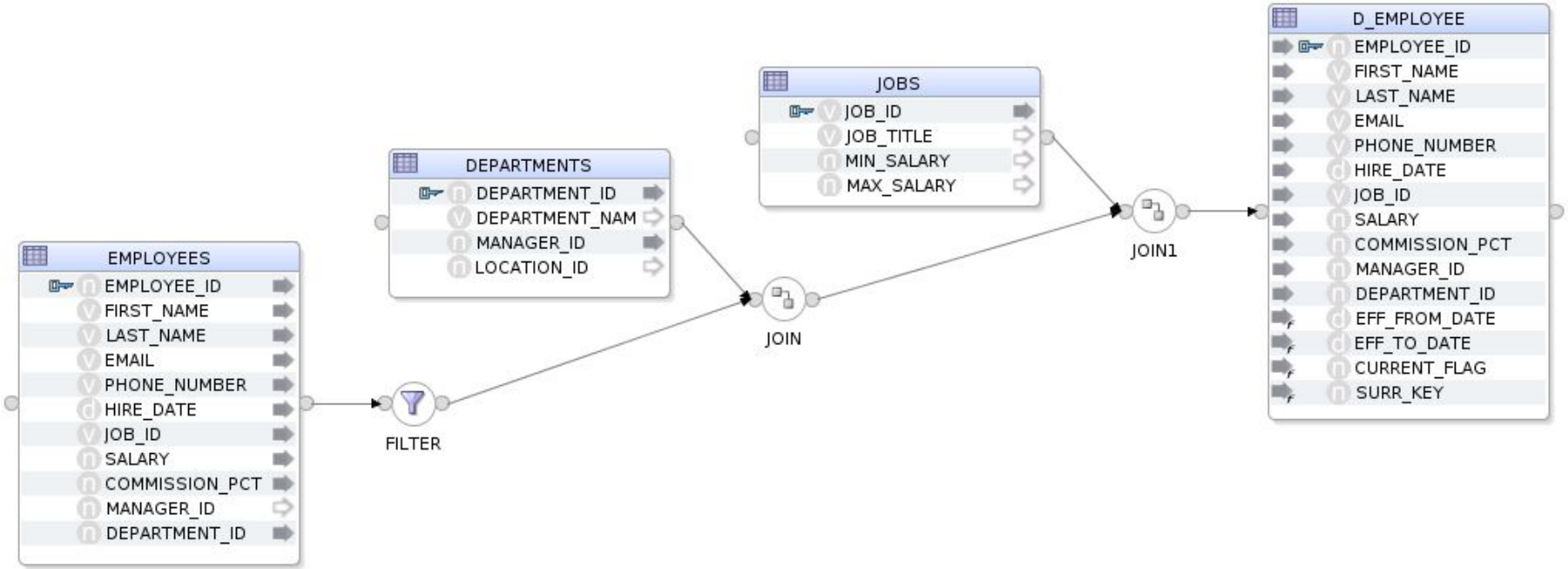

https://www.rittmanmead.com/blog/2009/04/using-a-partition-by-join-to-fill-in-the-gaps



https://www.rittmanmead.com/blog/2009/04/using-a-partition-by-join-to-fill-in-the-gaps

```
mapping.create('MY_PROJECT', 'DEMO_FOLDER', 'EMPLOYEE_DIM_LOAD')
       .datastores([
          [name: "HR.EMPLOYEES"],
          [name: "HR.DEPARTMENTS"],
          [name: "HR.JOBS"],
          [name: "PERF.D_EMPLOYEE", integration_type: "SCD"],
        ])
       .select("EMPLOYEES")
       .filter('NAME_FILTER', [filter_condition: "EMPLOYEES.FIRST_NAME LIKE 'D%'" ])
       .join('EMP_DEPT', ['DEPARTMENTS'],
                         [join_condition: "EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID" ])
       .join('DEPT_JOBS', ['JOBS'],
                          [join_condition: "EMPLOYEES.JOB_ID = JOBS.JOB_ID" ])
       .connect("D_EMPLOYEE",
           [[ attr: "employee_id", key_indicator: true ],
           [ attr: "eff_from_date", expression: "sysdate", execute_on_hint: "TARGET"],
           [ attr: "eff_to_date", expression: "sysdate", execute_on_hint: "TARGET"],
              [ attr: "current_flag", expression: 1, execute_on_hint: "TARGET"],
           [ attr: "surr_key",
             expression: ":RM_PROJECT.D_EMPLOYEE_SEQ_NEXTVAL",
             execute_on_hint: "TARGET"],])
       .commit()
       .validate()
```

https://www.rittmanmead.com/blog/2017/03/accelerating-your-odi-implementation-rittmanmead-style/



https://www.rittmanmead.com/blog/2017/03/accelerating-your-odi-implementation-rittmanmead-style/

```
SELECT first_name,
       last_name,
       email,
       phone_number,
       hire_date,
       job_id,
       manager_id,
       department_id,
       salary,
       commission_pct,
       eff_from_date,
       eff_to_date,
       current_flag
FROM EMPLOYEES
JOIN DEPARTMENTS using (DEPARTMENT_ID)
JOIN JOBS using (JOB_ID)
```

Code >> GUI >> Code

I would *rather* write this.

```
SELECT first_name,
       last_name,
       email,
       phone_number,
       hire_date,
       job_id,
       manager_id,
       department_id,
       salary,
       commission_pct,
       eff_from_date,
       eff_to_date,
       current_flag
FROM EMPLOYEES
JOIN DEPARTMENTS using (DEPARTMENT_ID)
JOIN JOBS using (JOB_ID)
```

Navigation

Quick search

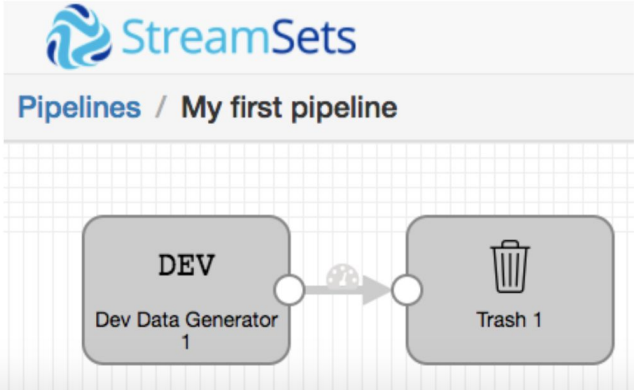
Go

StreamSets SDK for Python

The StreamSets SDK for Python enables users to interact with StreamSets products programmatically using Python 3.4+. As an example, with a running instance of StreamSets Data Collector, you can create and import a functional pipeline in less than 10 lines of code:

```
from streamsets.sdk import DataCollector
server_url = 'http://localhost:18630'
data_collector = DataCollector(server_url)

builder = data_collector.get_pipeline_builder()
dev_data_generator = builder.add_stage('Dev Data Generator')
trash = builder.add_stage('Trash')

dev_data_generator >> trash  # connect the Dev Data Generator origin

pipeline = builder.build('My first pipeline')
data_collector.add_pipeline(pipeline)
```

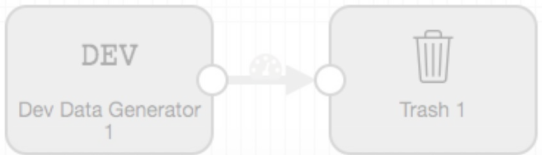
The resulting pipeline can be examined by opening the StreamSets Data Collector user interface and selecting the `My first pipeline` pipeline:



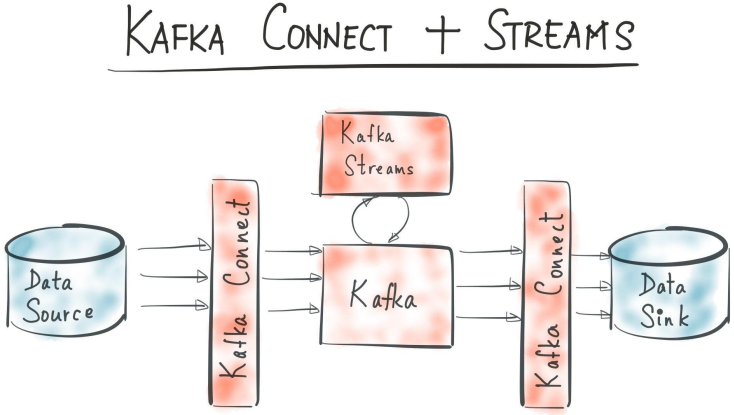
StreamSets SDK for Python

The StreamSets SDK for Python enables users to interact with StreamSets products programmatically using Python 3.4+. As an example, with a running instance of StreamSets Data Collector, you can create and import a functional pipeline in less than 10 lines:

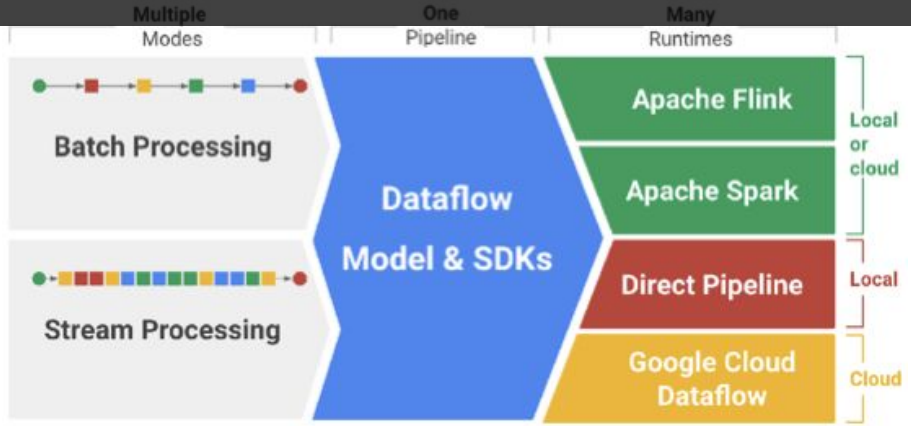
```
from streamsets.sdk import DataCollector
server_url = 'http://localhost:18630'
data_collector = DataCollector(server_url)

builder = data_collector.get_pipeline_builder()
dev_data_generator = builder.add_stage('Dev Data Generator')
trash = builder.add_stage('Trash')

dev_data_generator >> trash  # connect the Dev Data Generator origin

pipeline = builder.build('My first pipeline')
data_collector.add_pipeline(pipeline)
```

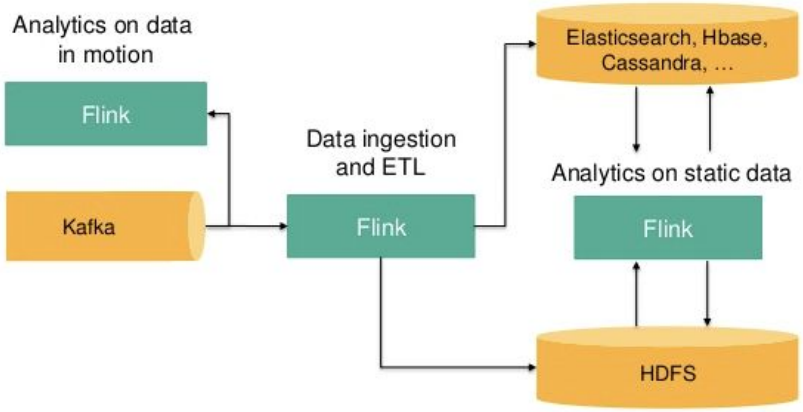


We aren’t starting with a *blank* slate.



An *embarrassment* of riches.



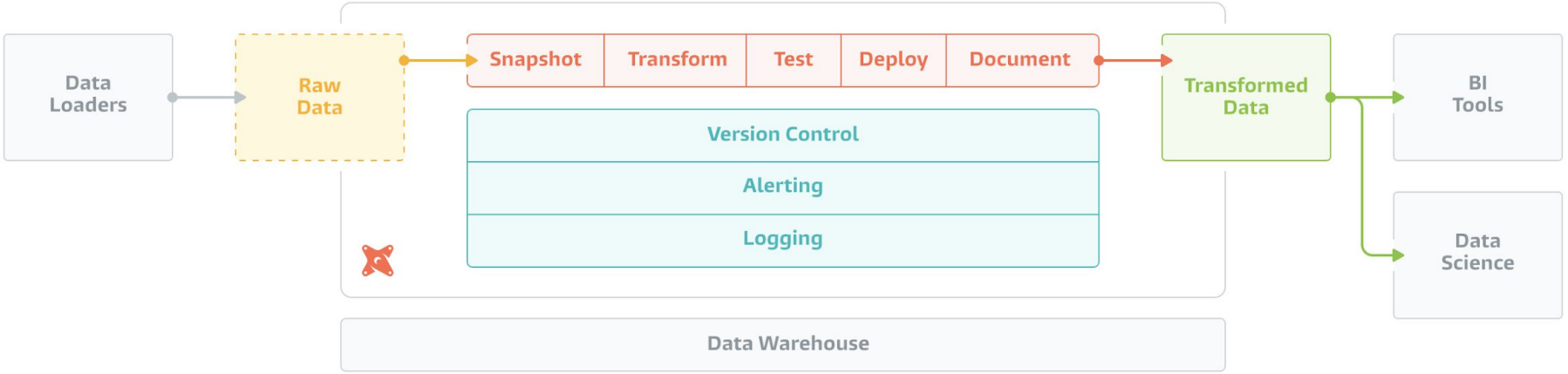


Shouldn’t my *pipelines* just live in my data warehouse?

Data Build Tool *(dbt)*

Analytics engineering tools designed for analysts

dbt is a command line tool that speaks the preferred language of data analysts everywhere—SQL. With dbt, analysts take ownership of the entire analytics engineering workflow, from writing data transformation code to deployment and documentation.



Dataform

Dataform SDK

Develop reliable and scalable SQL pipelines

Open source. Run Dataform locally or anywhere.

Manage dependencies between your tables.

Reuse code across all your scripts.

Write tests to assert your data quality.

```
# Install Dataform CLI
npm i -g @dataform/cli
# Create new project
dataform init snowflake my_project
# Create new table
echo 'config { type: "table" } SELECT 1 as one'
 > my_table.sqlx
# Run project
dataform run my_project
```

Read the **docs** or view on **GitHub**

Dataform web

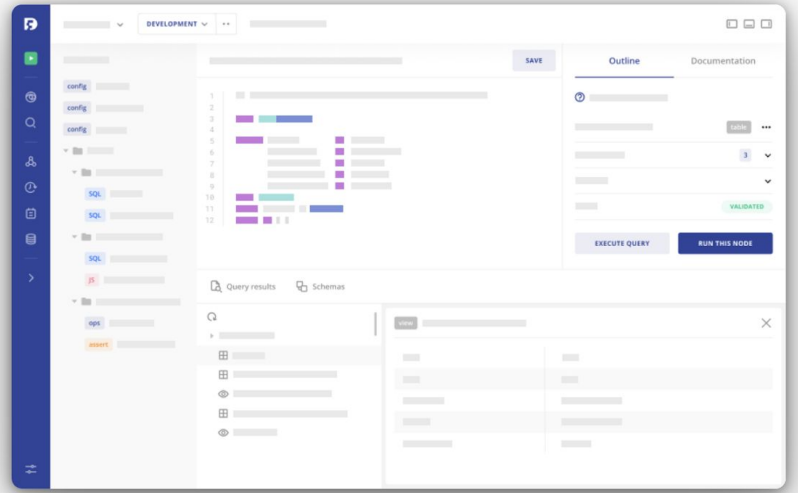
A complete solution for data warehouse management

Develop as a team in a collaborative web environment.

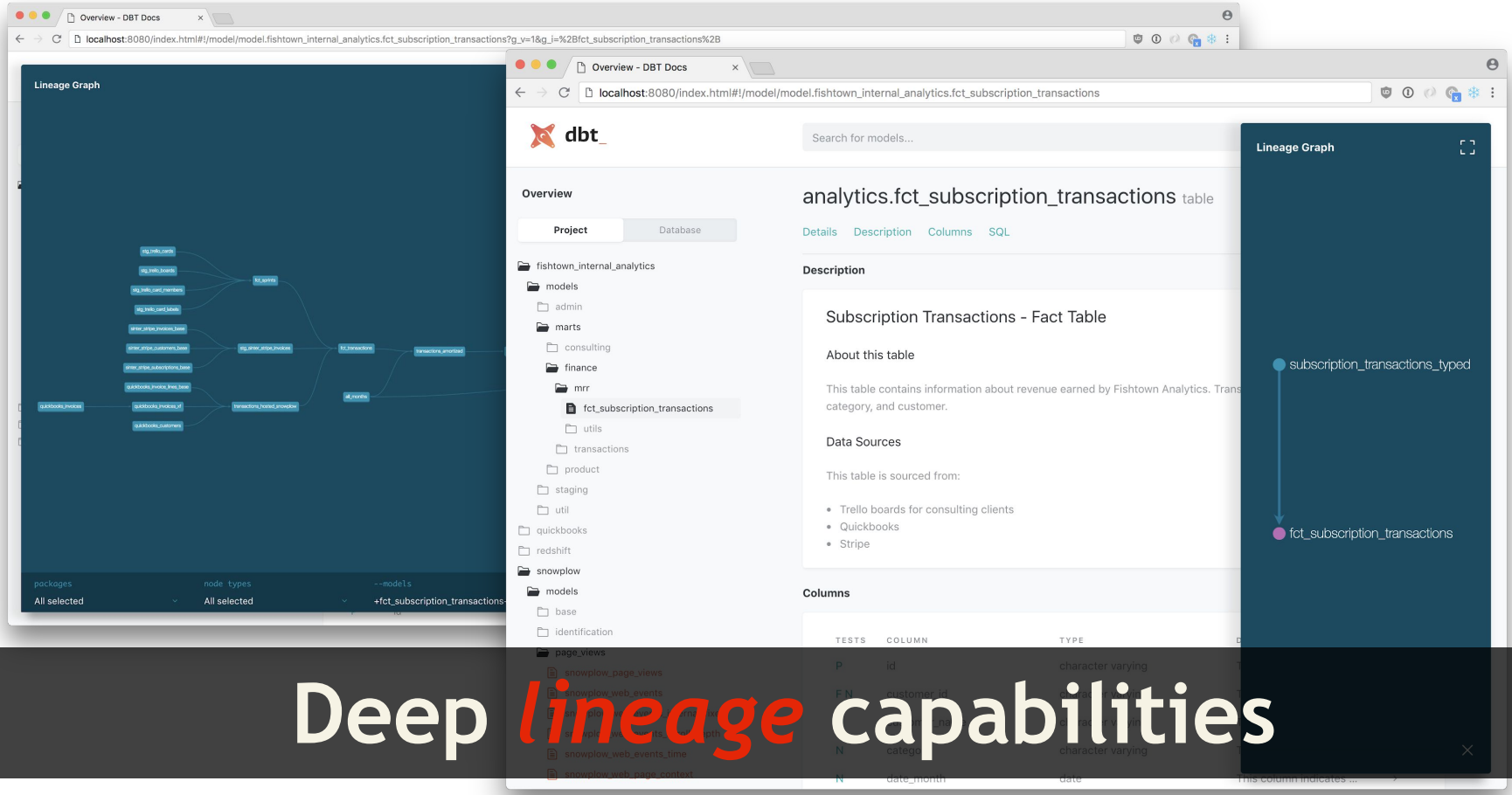
Version control all your code with a native GitHub integration.

Orchestrate your pipelines and get alerted if anything goes wrong.

Share a data catalog with all your data definitions.



Learn more about Dataform web →



Deep *lineage* capabilities

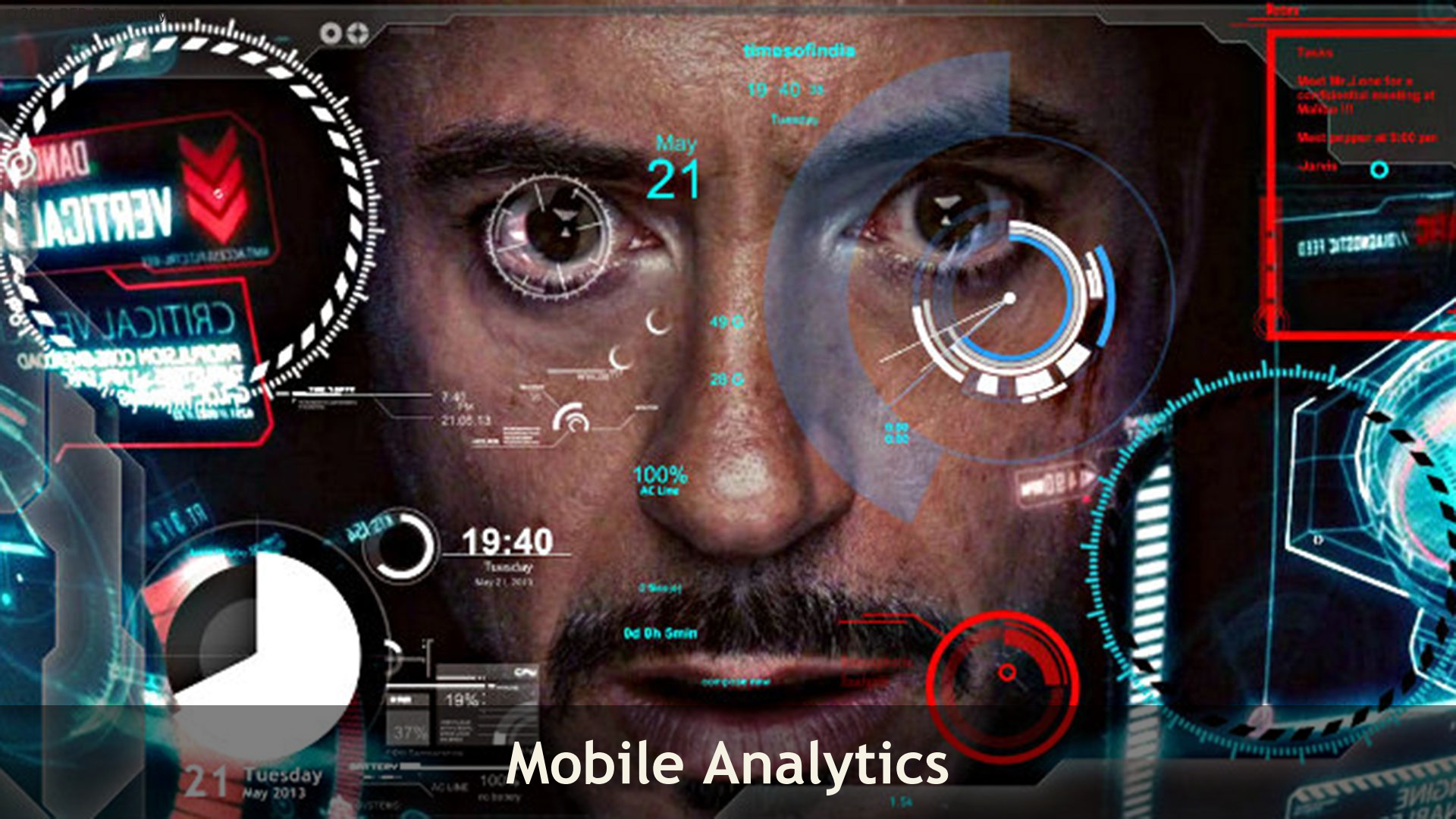
@redpilla

Am I feeding other
downstream systems?

Data warehouses *typically*
aren't very good at that.

What are some other *use cases* we might want to consider?

Realtime Events



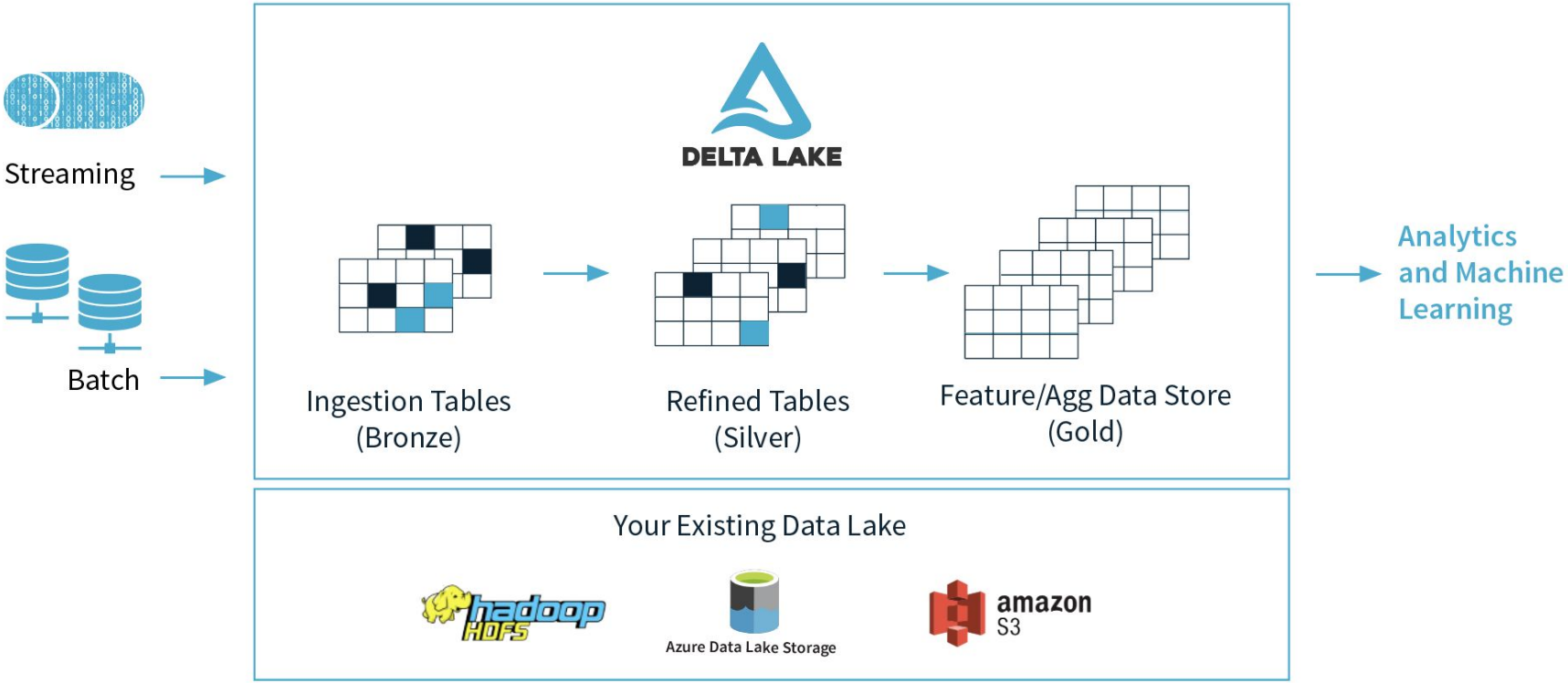
Mobile Analytics

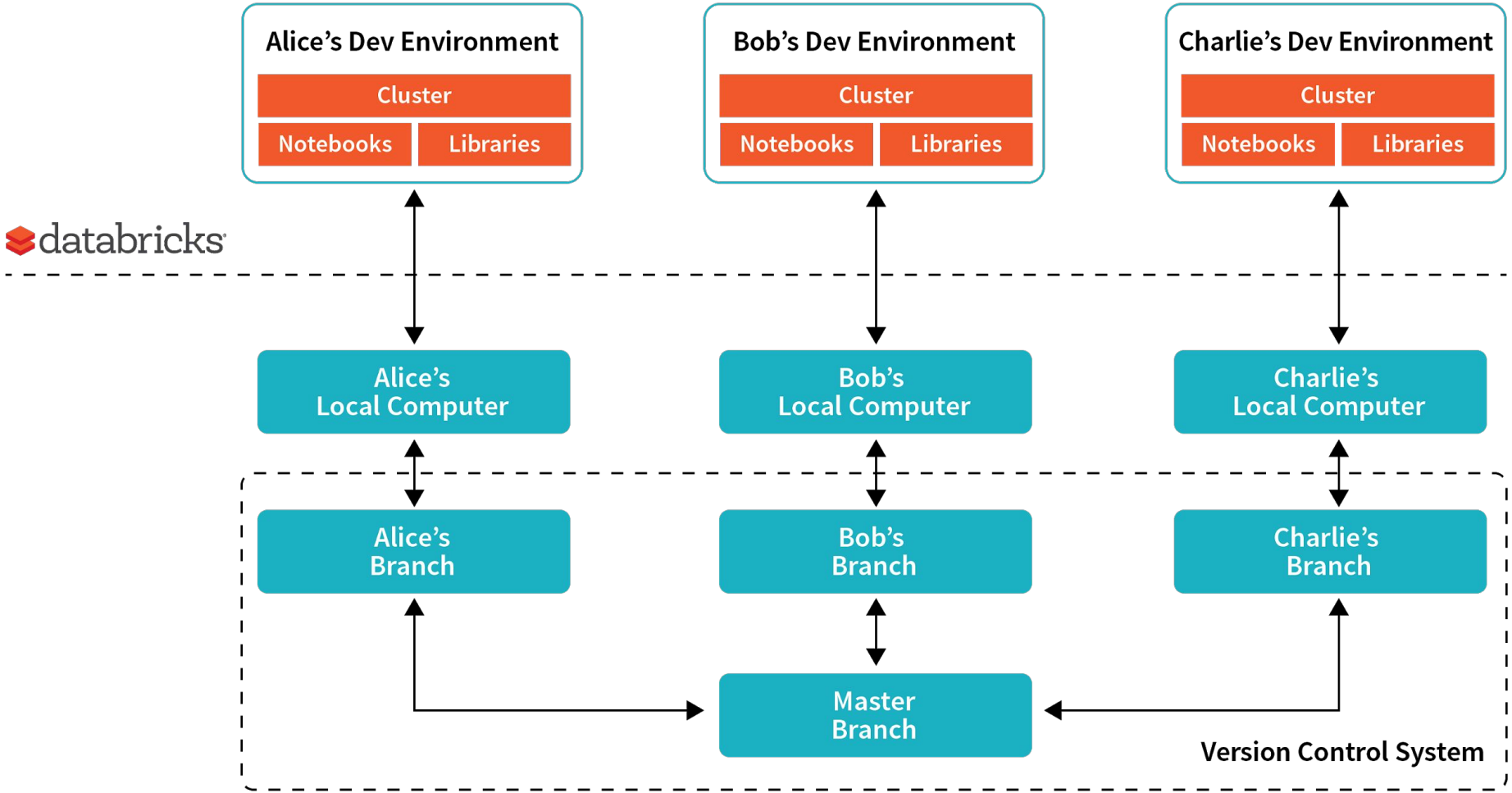
Search

Machine *Learning*

Why *Spark?*

Why *Databricks?*


Streaming
Batch
DELTA LAKE
Ingestion Tables
(Bronze)
Refined Tables
(Silver)
Feature/Agg Data Store
(Gold)
Analytics
and Machine
Learning
Your Existing Data Lake
hadoop
HDFS
Azure Data Lake Storage
amazon
S3


Alice's Dev Environment
Cluster
Notebooks
Libraries
Bob's Dev Environment
Cluster
Notebooks
Libraries
Charlie's Dev Environment
Cluster
Notebooks
Libraries
databricks
Alice's Local Computer
Bob's Local Computer
Charlie's Local Computer
Alice's Branch
Bob's Branch
Charlie's Branch
Master Branch
Version Control System


Demonstration

YOU CAN **CHOOSE** TO SEE **DATA** DIFFERENTLY.

RED PILL ANALYTICS

RedPillAnalytics.com
@RedPillA