# Data Engineering and the Death of Visual ETL

Tuesday | December 3, 2019
16:45 - 17:30

Stewart Bryson, Red Pill Analytics
@stewartbryson

UKOUG

# Data Engineering and the Death of Visual ETL

Monday | September 16, 2019
12:30 - 1:15 PM
Moscone South -Room 306

Stewart Bryson, Red Pill Analytics
@stewartbryson

OOW19

RED PILL ANALYTICS

# Data Engineering and the Death of Visual ETL

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

# Partnerships

## Data Warehouse

BigQuery

snowflake

## Analytics

looker

Data Studio

ORACLE ANALYTICS

## Data-engineering & ETL

kafka

Fivetran

AWS Glue

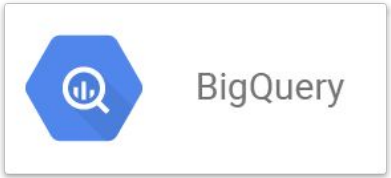
dbt_

databricks

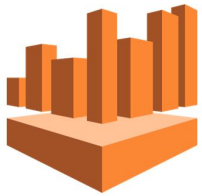
StreamSets

# Technologies We Use

## Data Warehouse

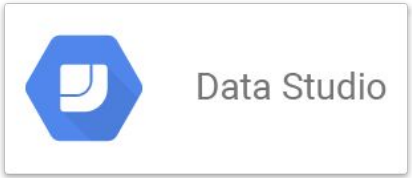






Amazon Athena

## Analytics



## Data-engineering & ETL



AWS Glue


# We’re hiring.

# 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

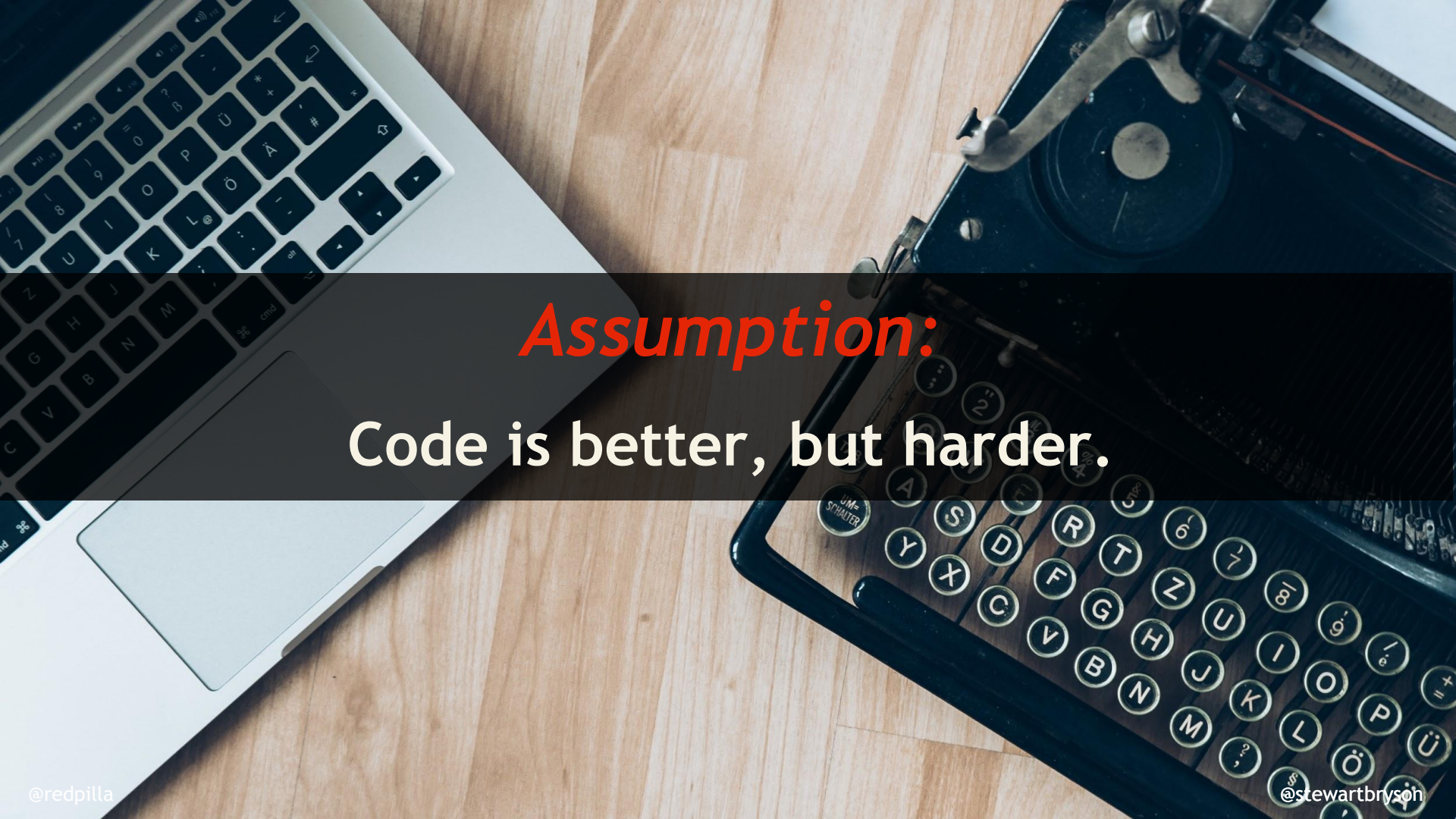
@stewartbryson

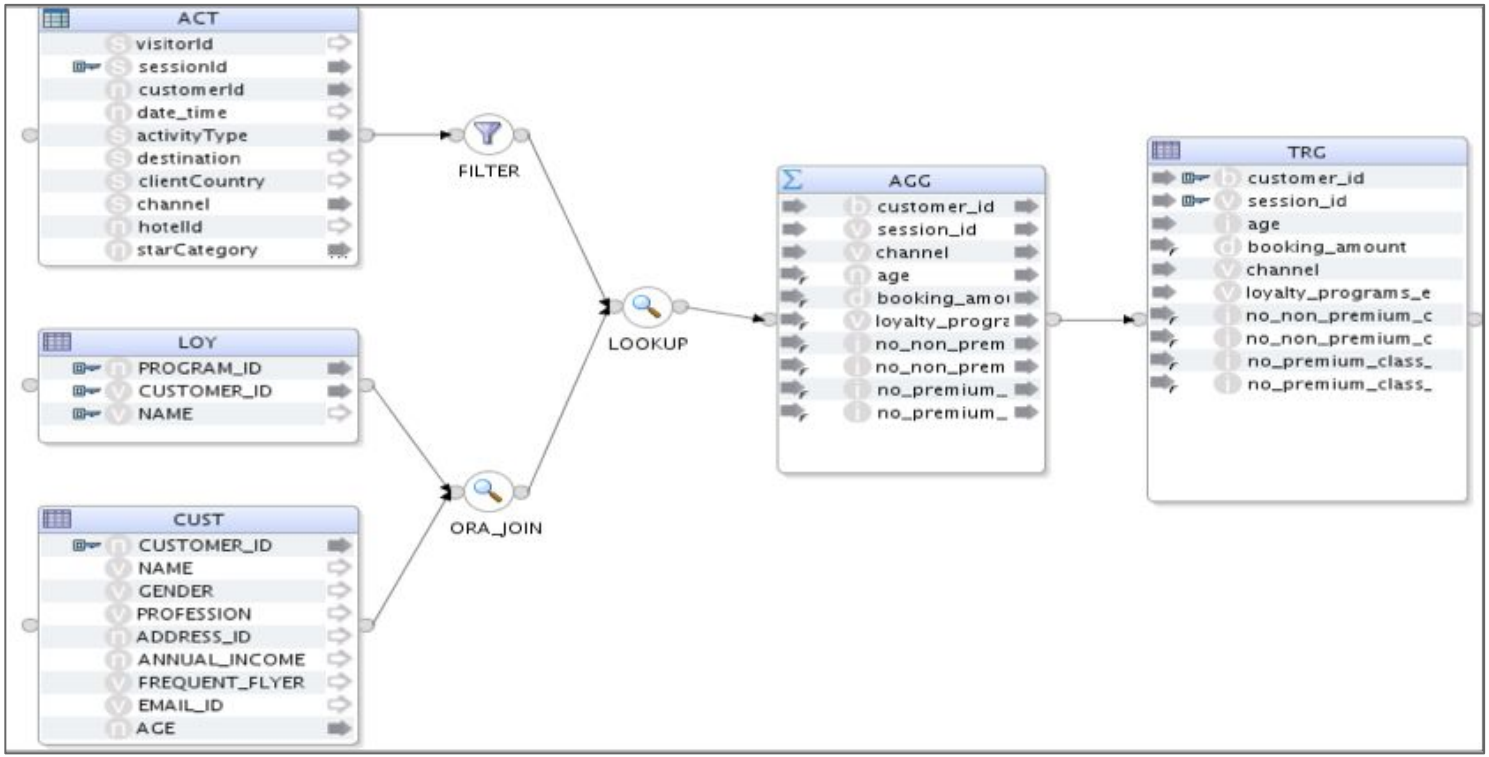

What's so
*different*
about
ETL?
@redpilla
@stewartbryson

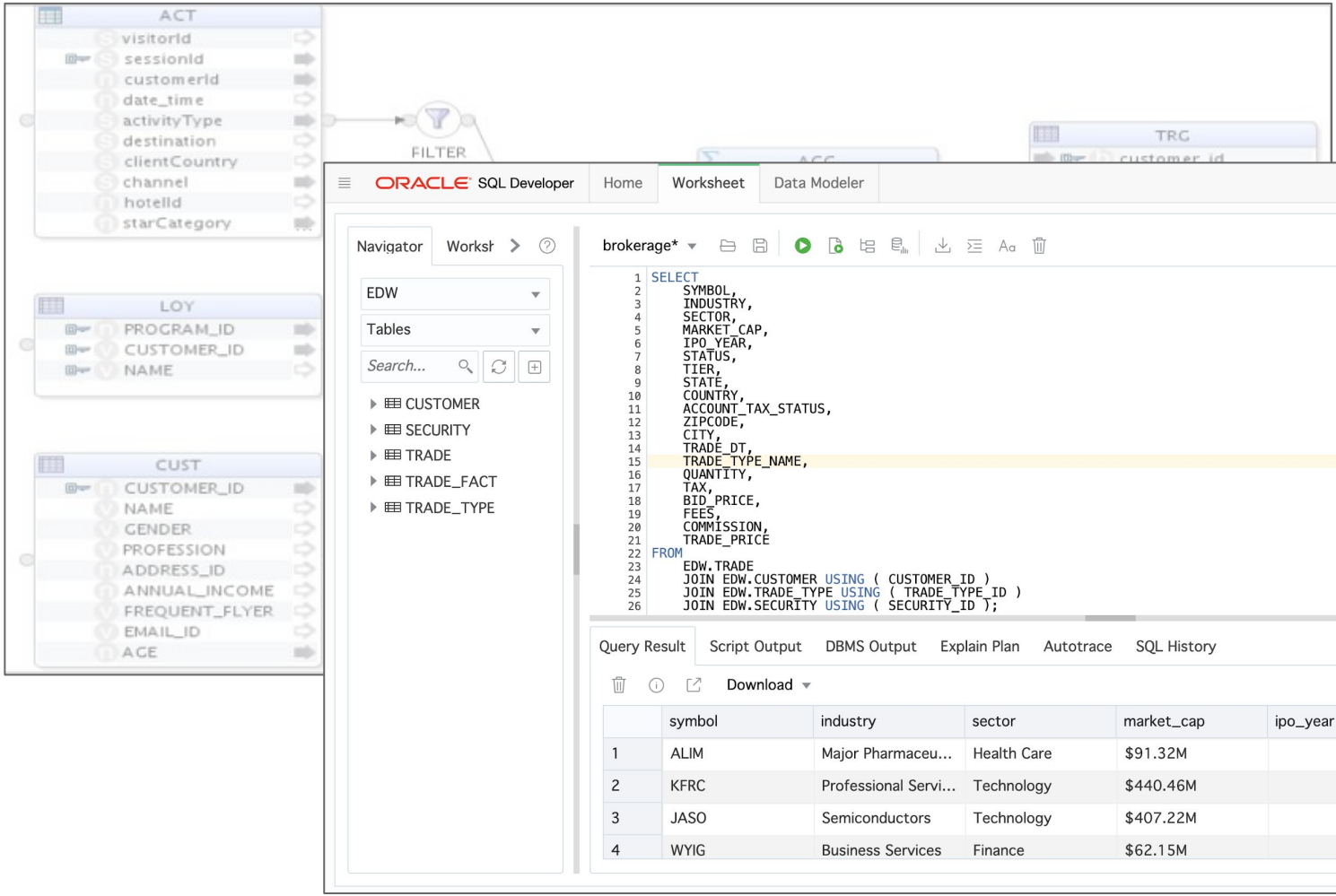
# Code *versus* Clicks

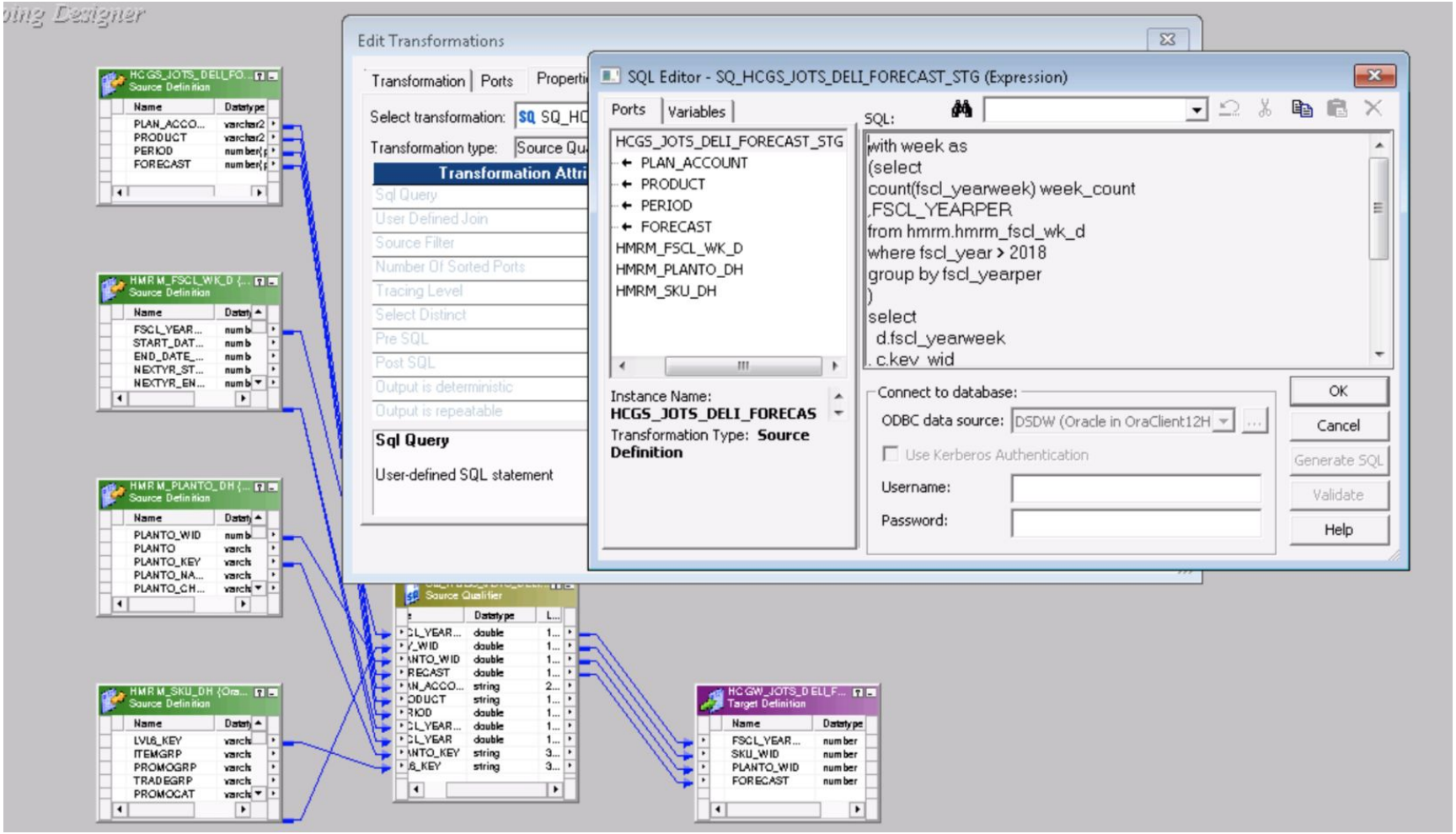
Source control
CI/CD
Collaboration
Code anywhere

Faster
Easier to start
Any discipline
Business friendly


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


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_

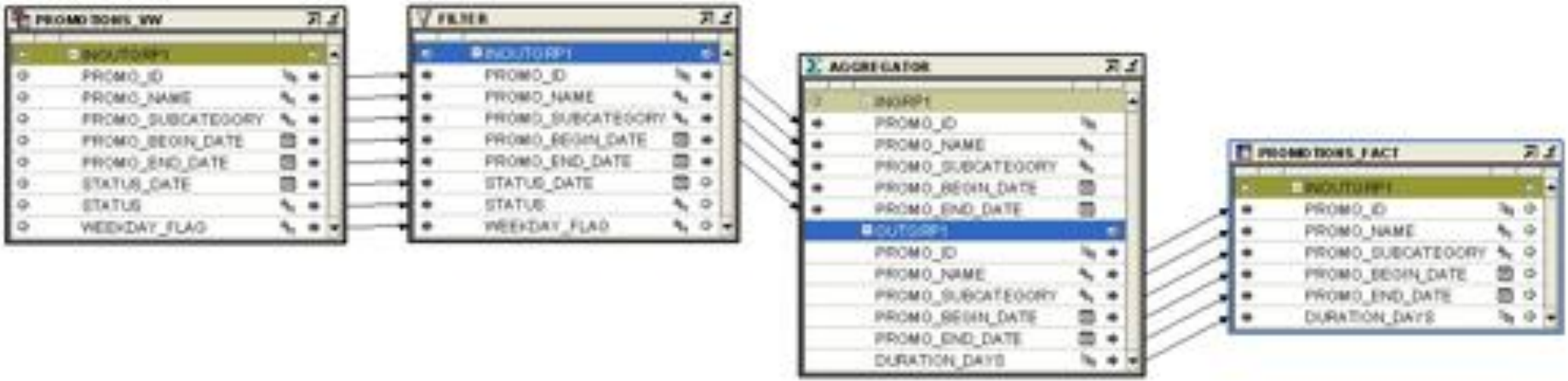

ACT
visitorId
sessionId
customerId
date_time
activityType
destination
clientCountry
channel
hotelId
starCategory
FILTER
TRG
customer_id
LOY
PROGRAM_ID
CUSTOMER_ID
NAME
CUST
CUSTOMER_ID
NAME
GENDER
PROFESSION
ADDRESS_ID
ANNUAL_INCOME
FREQUENT_FLYER
EMAIL_ID
AGE
ORACLE SQL Developer
Home
Worksheet
Data Modeler
ADMIN
Navigator
Worksh
EDW
Tables
Search...
CUSTOMER
SECURITY
TRADE
TRADE_FACT
TRADE_TYPE
brokerage*
1 SELECT
2 SYMBOL,
3 INDUSTRY,
4 SECTOR,
5 MARKET_CAP,
6 IPO_YEAR,
7 STATUS,
8 TIER,
9 STATE,
10 COUNTRY,
11 ACCOUNT_TAX_STATUS,
12 ZIPCODE,
13 CITY,
14 TRADE_DT,
15 TRADE_TYPE_NAME,
16 QUANTITY,
17 TAX,
18 BID_PRICE,
19 FEES,
20 COMMISSION,
21 TRADE_PRICE
22 FROM
23 EDW.TRADE
24 JOIN EDW.CUSTOMER USING ( CUSTOMER_ID )
25 JOIN EDW.TRADE_TYPE USING ( TRADE_TYPE_ID )
26 JOIN EDW.SECURITY USING ( SECURITY_ID );
Query Result
Script Output
DBMS Output
Explain Plan
Autotrace
SQL History
Download
symbol
industry
sector
market_cap
ipo_year
status
tier
1 ALIM Major Pharmaceu... Health Care $91.32M 2010 Completed
2 KFRC Professional Servi... Technology $440.46M 1995 Completed
3 JASO Semiconductors Technology $407.22M 2007 Completed
4 WYIG Business Services Finance $62.15M 2015 Completed


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
```



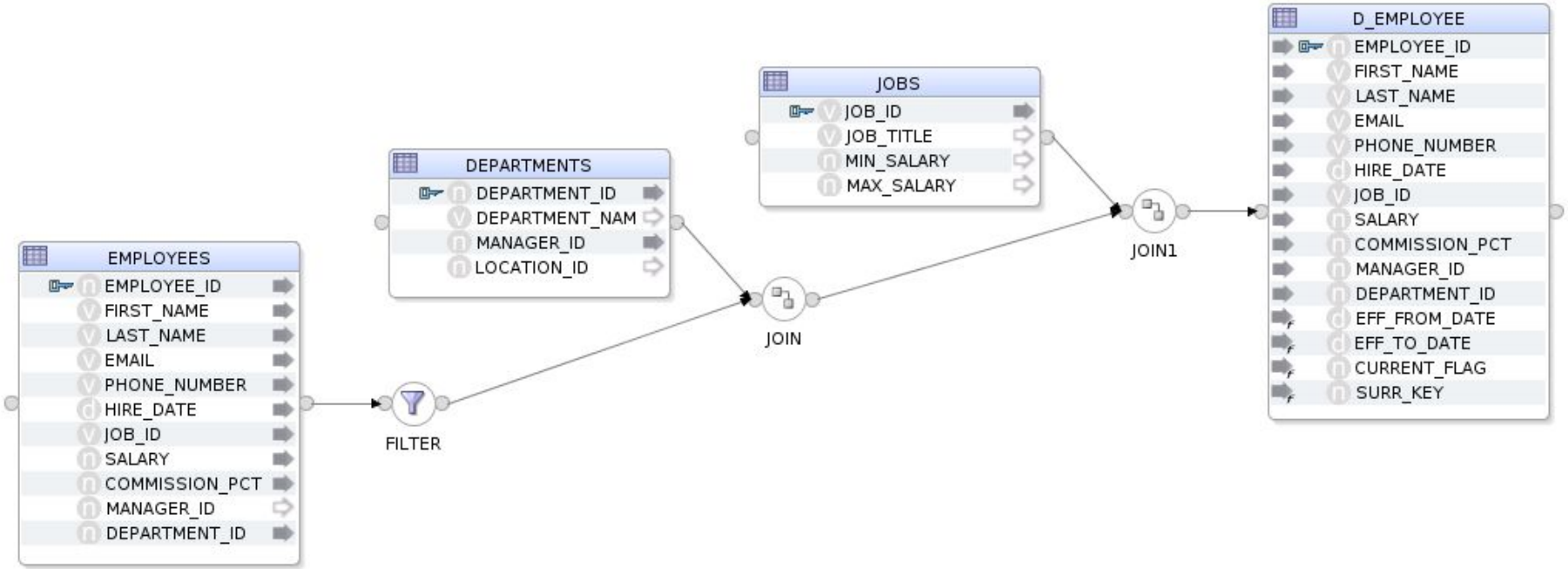
https://www.rittmanmead.com/blog/2009/04/using-a-partition-by-join-to-fill-in-the-gaps



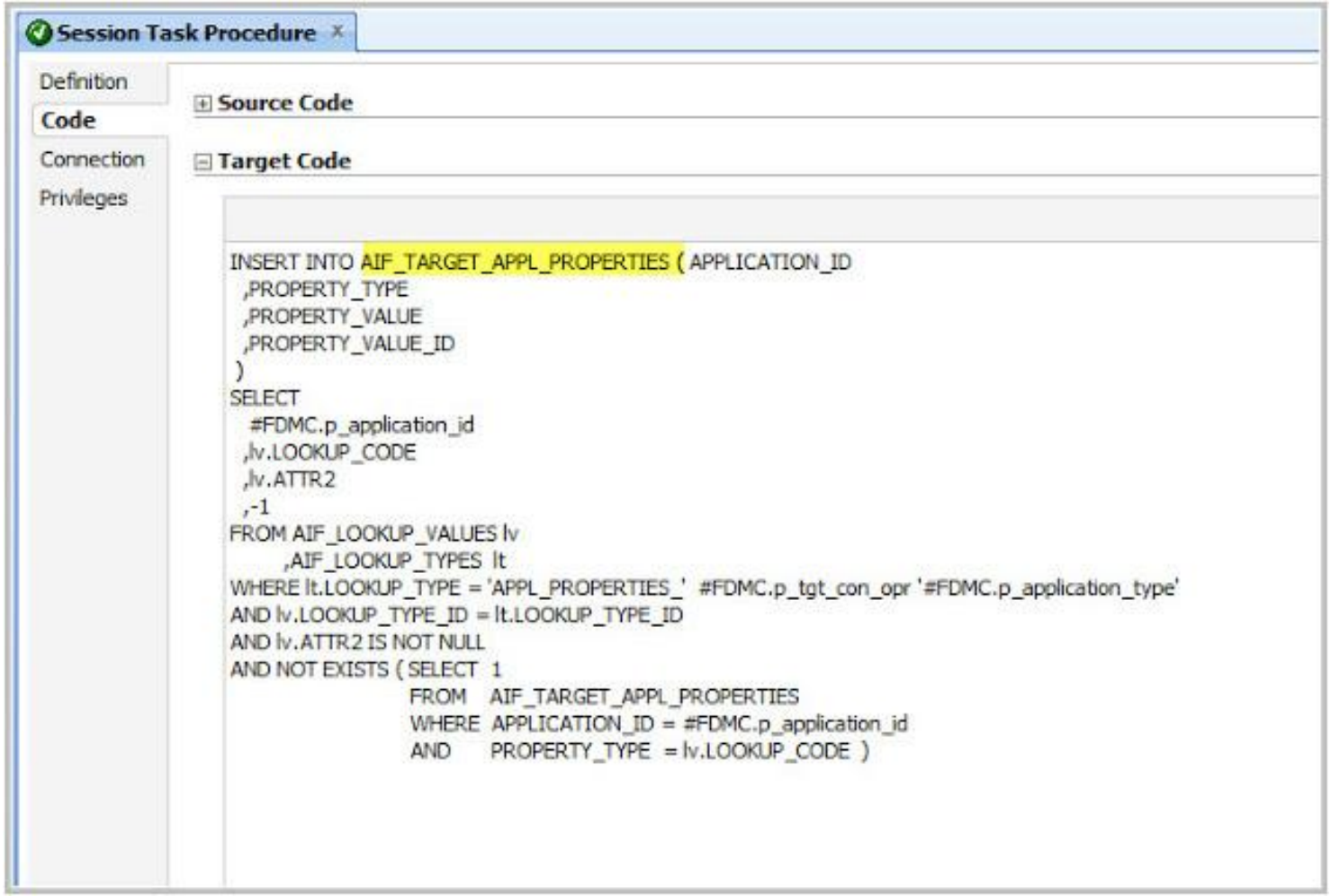

https://www.rittmanmead.com/blog/2009/04/using-a-partition-by-join-to-fill-in-the-gaps

```
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/


Session Task Procedure
Definition
Code
Connection
Privileges
Source Code
Target Code
INSERT INTO AIF_TARGET_APPL_PROPERTIES ( APPLICATION_ID
,PROPERTY_TYPE
,PROPERTY_VALUE
,PROPERTY_VALUE_ID
)
SELECT
#FDMC.p_application_id
,lv.LOOKUP_CODE
,lv.ATTR2
,-1
FROM AIF_LOOKUP_VALUES lv
,AIF_LOOKUP_TYPES lt
WHERE lt.LOOKUP_TYPE = 'APPL_PROPERTIES_' #FDMC.p_tgt_con_opr '#FDMC.p_application_type'
AND lv.LOOKUP_TYPE_ID = lt.LOOKUP_TYPE_ID
AND lv.ATTR2 IS NOT NULL
AND NOT EXISTS ( SELECT 1
FROM AIF_TARGET_APPL_PROPERTIES
WHERE APPLICATION_ID = #FDMC.p_application_id
AND PROPERTY_TYPE = lv.LOOKUP_CODE )

# 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

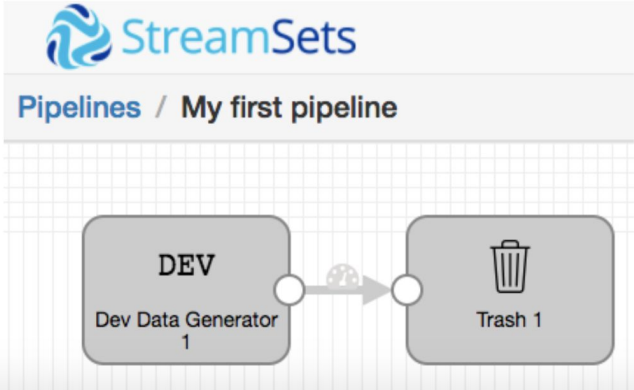
Go

# 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)
```

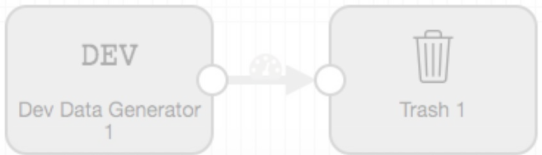
The resulting pipeline can be examined by opening the StreamSets Data Collector user interface and selecting the `My first pipeline` pipeline:



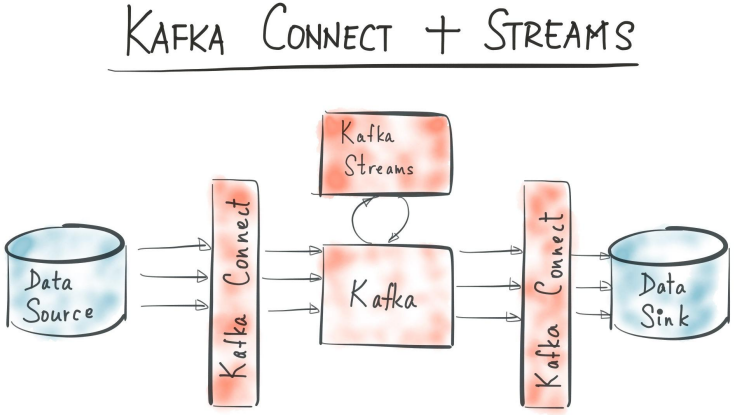
# StreamSets SDK for Python

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)
```

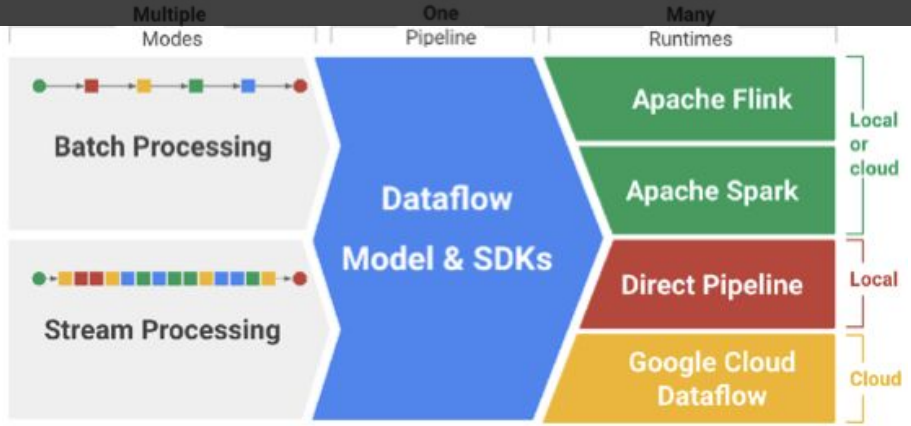


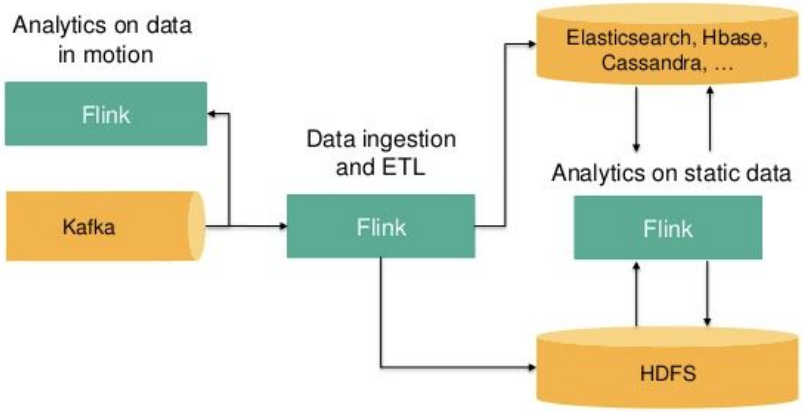
We aren’t starting with a *blank* slate.





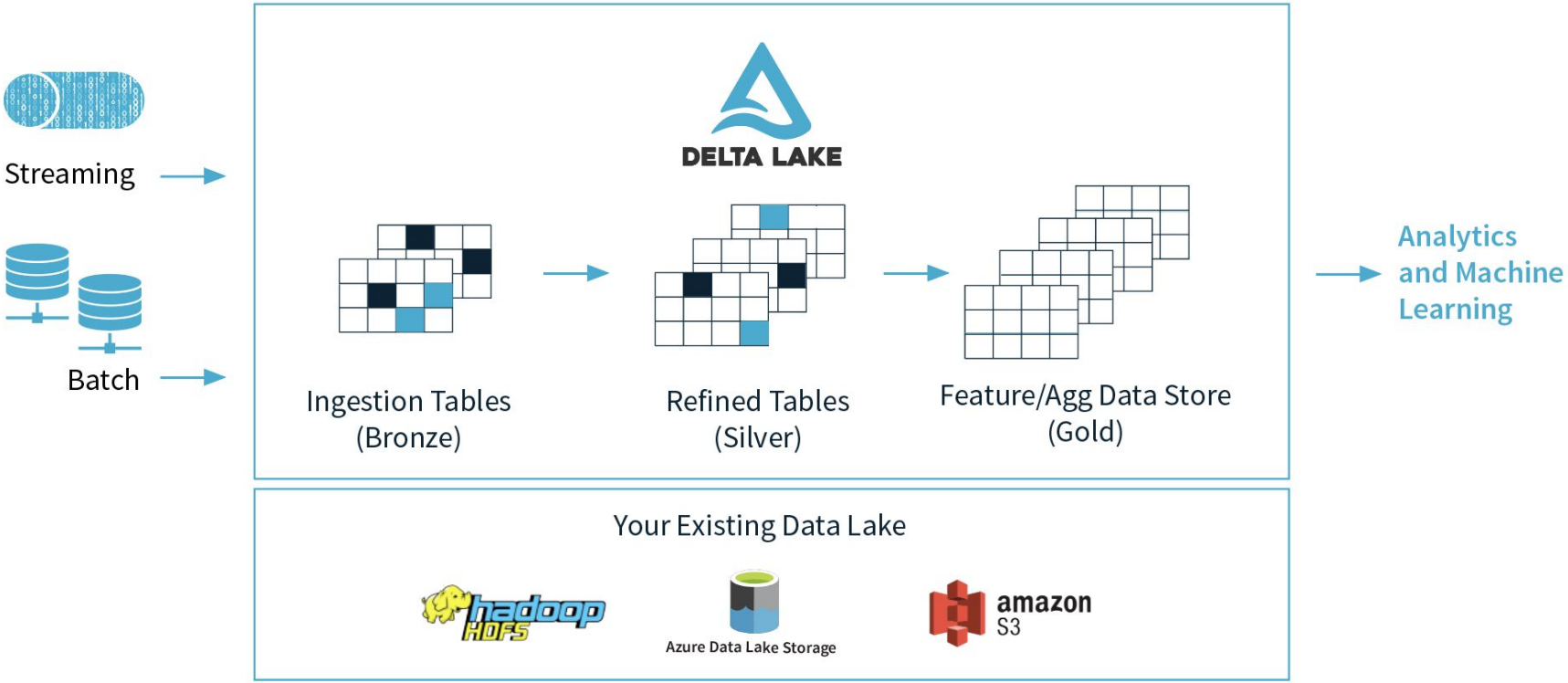
# An *embarrassment* of riches.

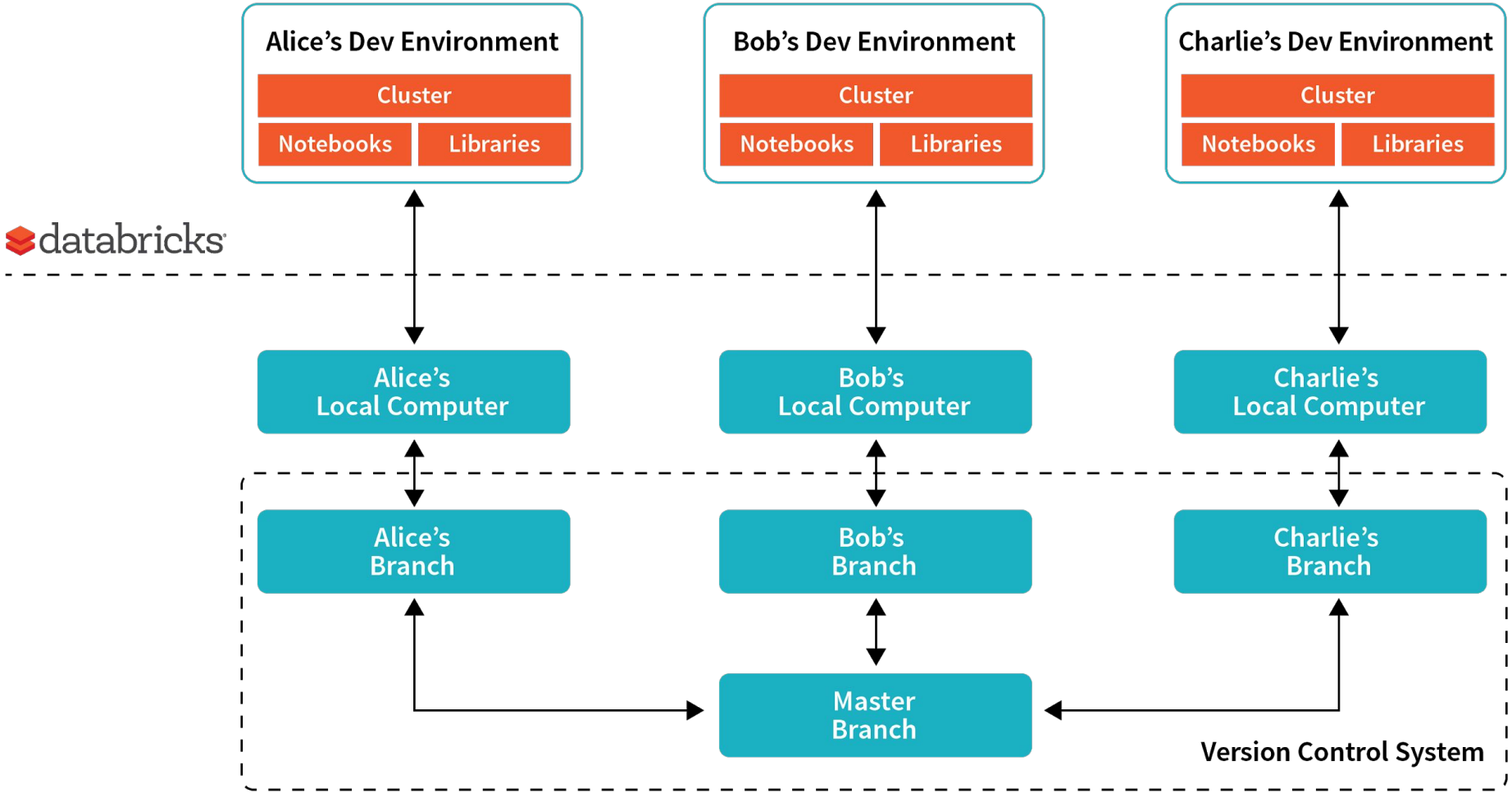


# Why *Spark?*

# 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