Reverse ETL With Airflow

Russell Dervay

Airflow Summit 2021

Intro Slide / Bio

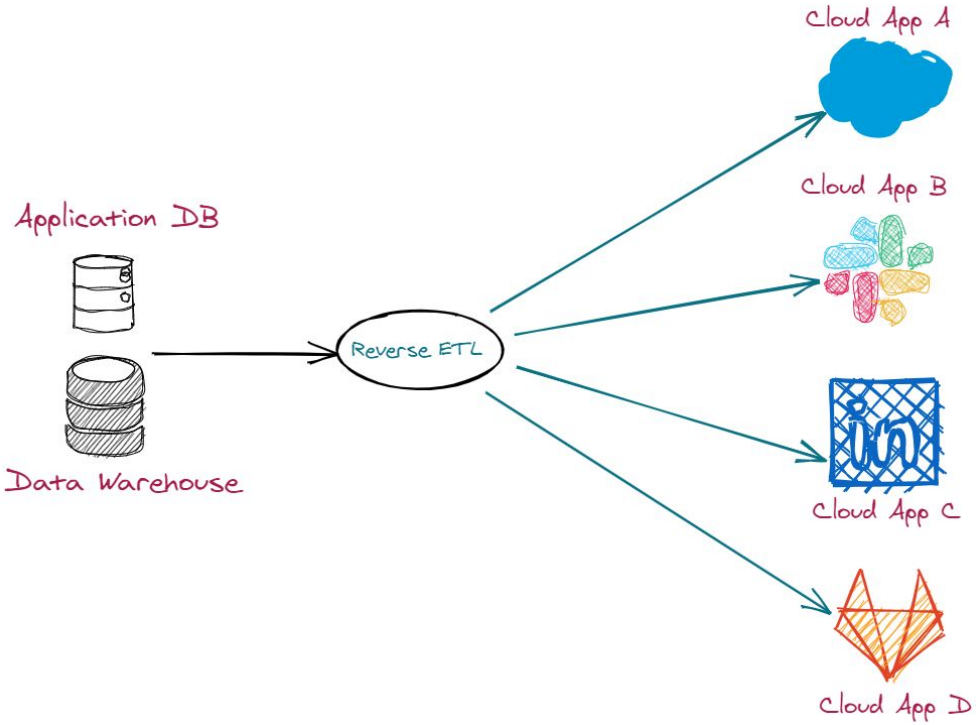
- Data Engineer @ Snowflake
- Focus on Data Transformation and reverse ETL
- Played Volleyball For Stanford

Agenda

- What Is Reverse ETL
  - Current Application Landscape
  - Data Architecture With Reverse ETL
  - Benefits
- Approach and Implementation
  - Design
  - Common Architecture
  - Configurations
- Example Account Scoring
  - Dag / Data Model
  - Example Update (SFDC)
- Considerations / Challenges

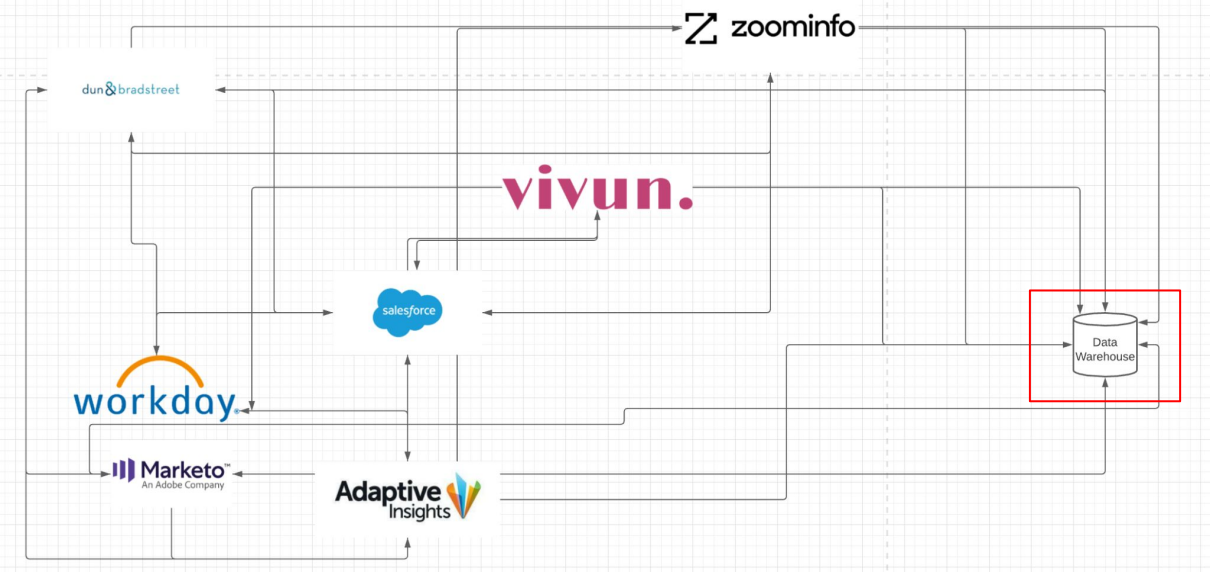
Reverse ETL

What Is Reverse ETL



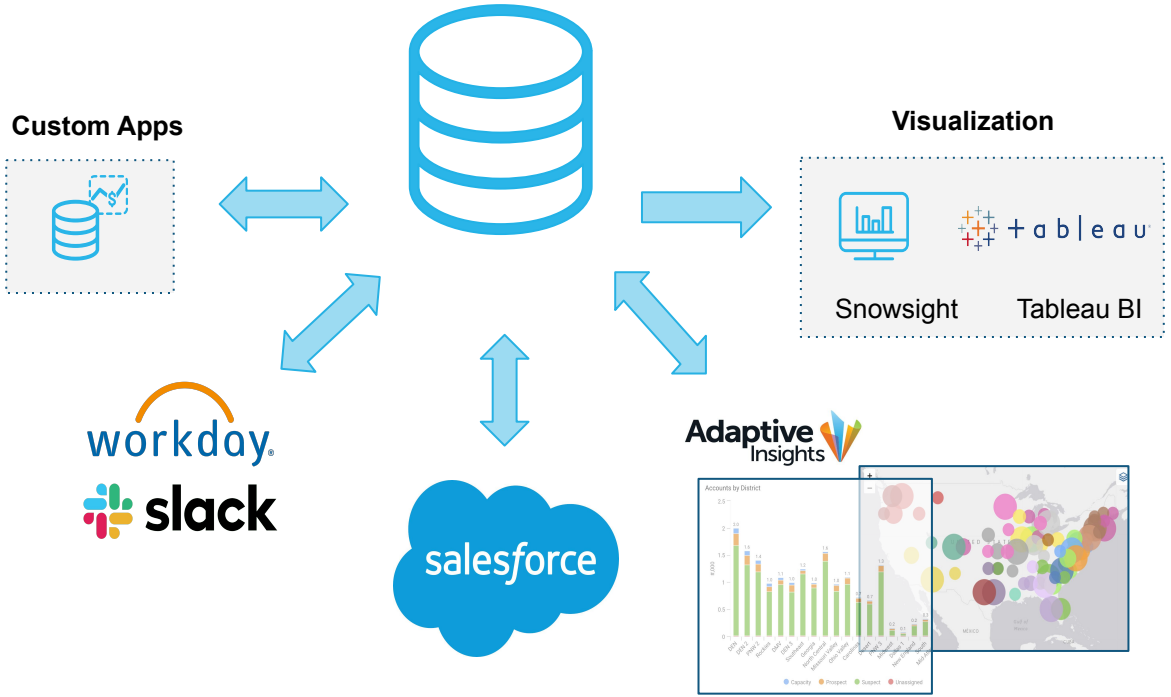
Data Architecture Before Reverse ETL

- No Single Source of Truth
- “Crossing Wires”
- Duplicate Calculations



Data Architecture With Reverse ETL

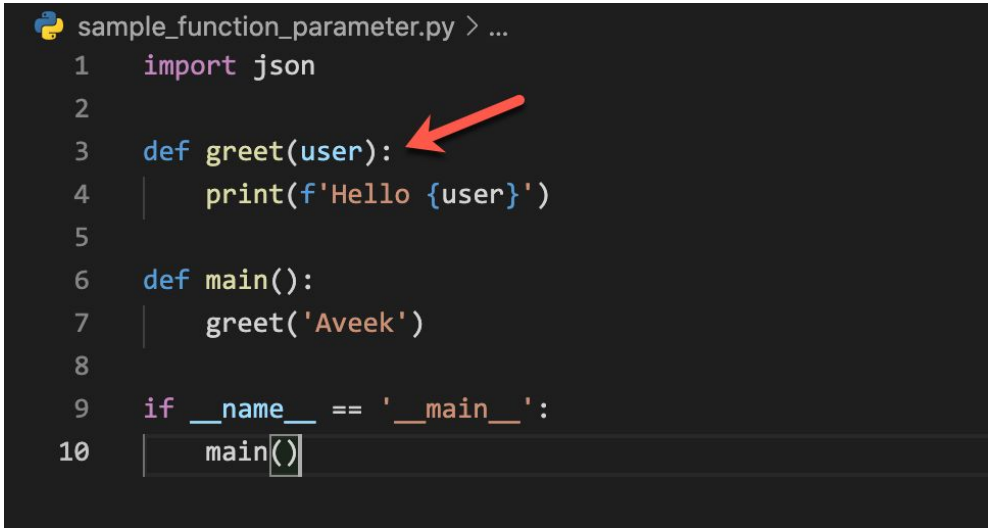
- Data Warehouse becomes center of your “data universe”
- Only 2 “integrations” per application
- Common “Back End” Shared Between Applications



Benefits: Compute Once

- All Metrics Run on the same Data Set

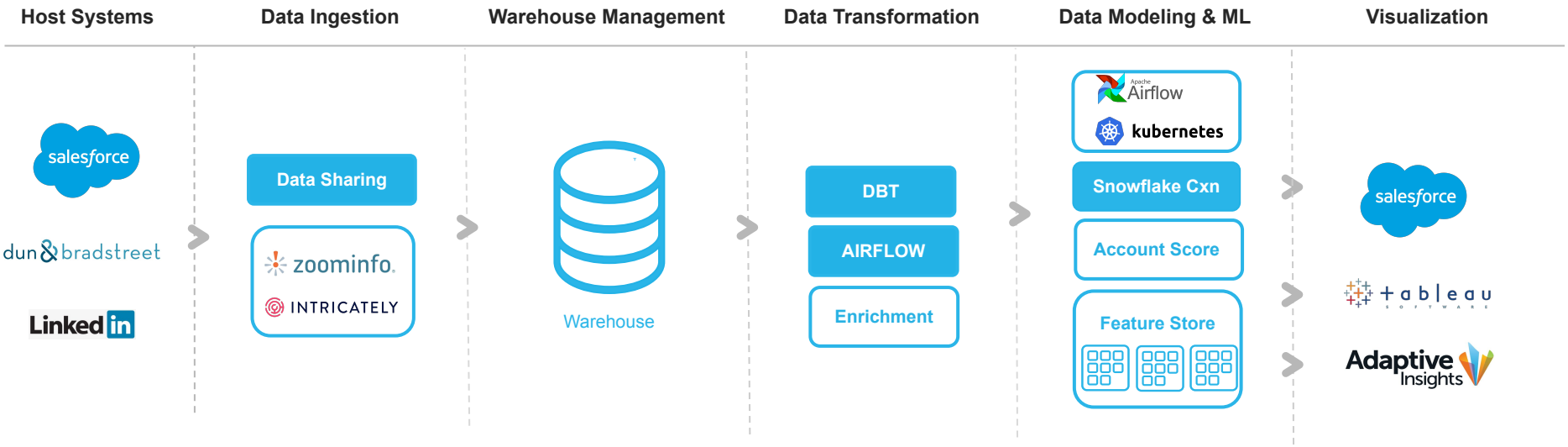
- Removes the possibility of deviant metrics



Benefits: Data Visibility

- Data Consumers have all relevant data in their “primary application”
- Cross Departmental Metrics can easily be shared with little to no lift

Architecture: Account Scoring



Implementation

Approach

- Airflow Based Approach
- “Just Works”
- Easy and accessible

Design

- Python Based Packages Per App
- End User Configurable
- Create, Update, Delete, Insert Operations supported on most applications

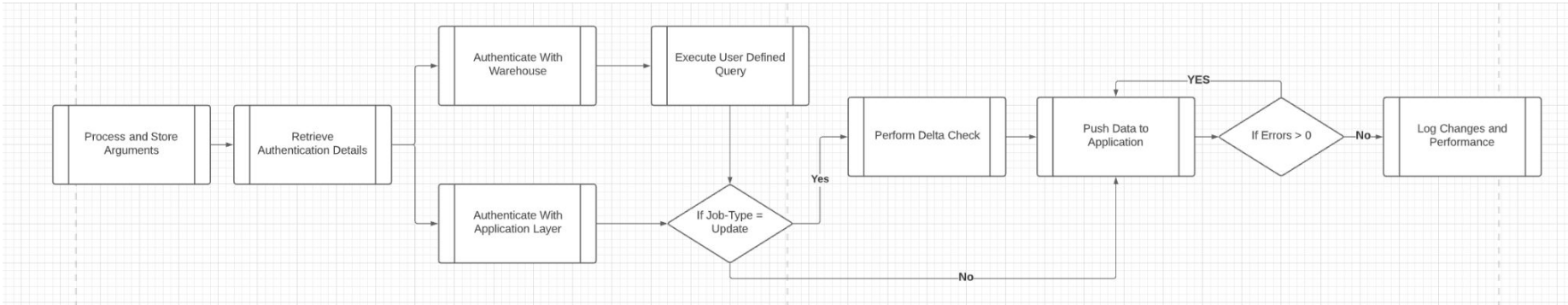
```
from airflow.models.baseoperator import BaseOperator

class HelloOperator(BaseOperator):

    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

End User Configuration

- **Query** - Represents the set of data you want to upload
- **Object / End Point** - Represents the endpoint or api you will be calling
- **Fields** - Columns to update in source
- **Job Type** - What operation the job should perform in source system

```
task_id="run_aps",
query="""
    select account_id
         , aps
    from database.schema.table
""",
object="account",
fields=["aps__c"],
job_type="update",
batch_size=10000,
```

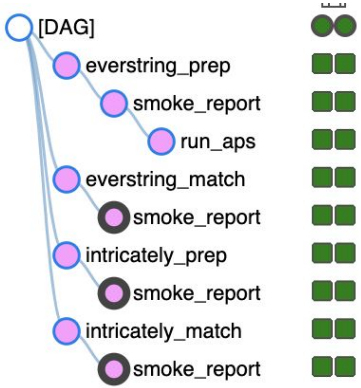
Under The Hood



Account Scoring Example

Account Scoring Example (DAG)

- Last Step Operation
- Many Tasks Needed For Many Systems

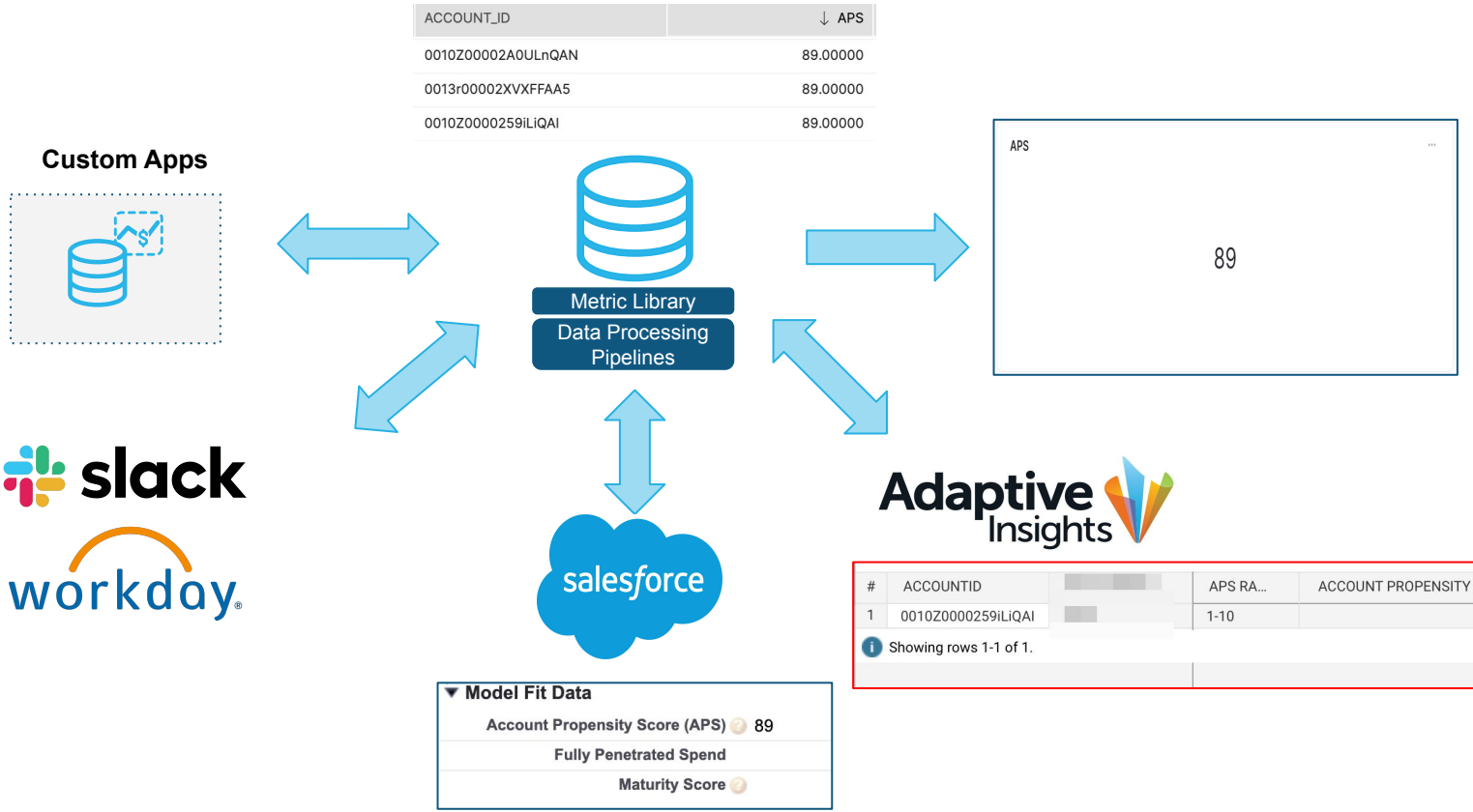


ACCOUNT_ID	↓ APS
0010Z000023qiRbQAI	94.00000
0010Z00002A0JwPQAV	94.00000
0010Z000025AlRzQAK	93.00000
0013r00002EFENAAA5	93.00000
0010Z0000295QrnQAE	92.00000
0010Z000029ztNQQAY	90.00000
0010Z00002D3gqFQAR	90.00000
0010Z0000259iLiQAI	89.00000
0013r00002PaG2MAAV	88.00000

Account Scoring Example (SFDC)

- Application Change alongside your model
- Business users always see the freshest data

Single Source Of Truth



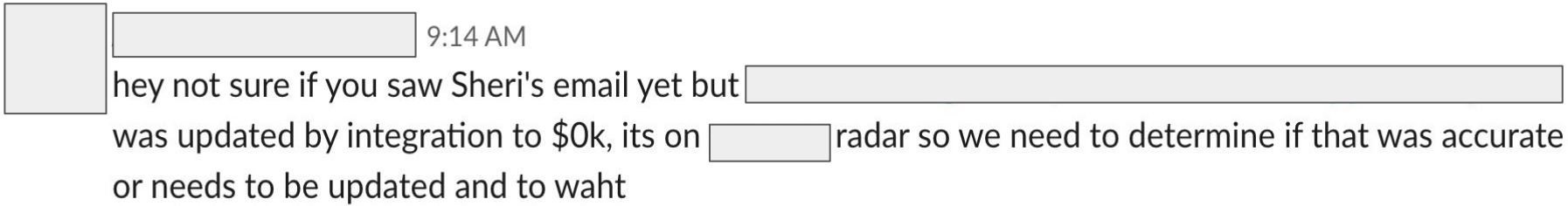
Challenges

Challenges (Compliance / Control)

- “Connectors” Are Widely Accessible
- New Jobs Added Frequently
- QA Testing Not Always Followed
- Requires Good Cross Departmental Communication

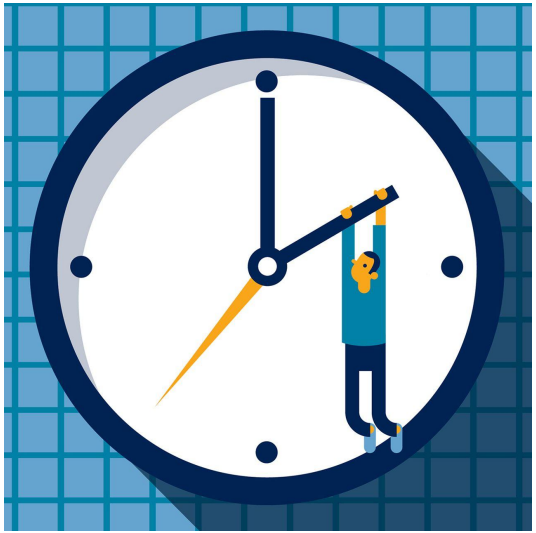
Challenges (Infectious Data)

- Garbage In Garbage Out
- Incorrect data in the source table or pipeline will flow downstream impacting all other applications



Challenges (Lift)

- Engineering Time Can Be Considerable
- Some APIs are more complex than others
- Maintenance can take some cycles away from development



Acknowledgements

Shradha Adsule
Data Engineer

Yamini Nawlani
Data Engineer

Kristen Werner
Director, Data Science and Engineering

Satya Kota
Senior Data Engineer

Ganesh Gadakar
Data Engineer

Questions?