Hewlett Packard
Enterprise

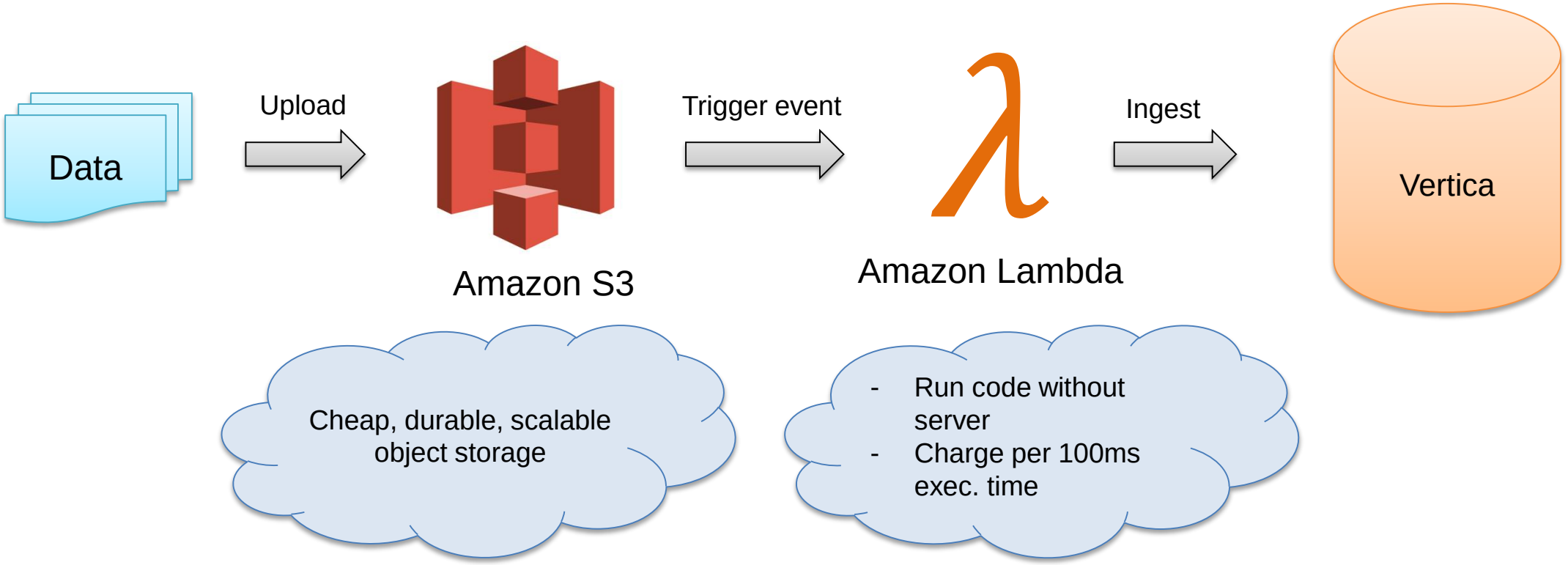
Asynchronous & Event-driven Query Execution in Vertica

Amin Saeidi, Ben Vandiver, Shreya Prasad
HPE Vertica

Hewlett Packard
Enterprise

Jan 2017

Example of an event-driven workflow



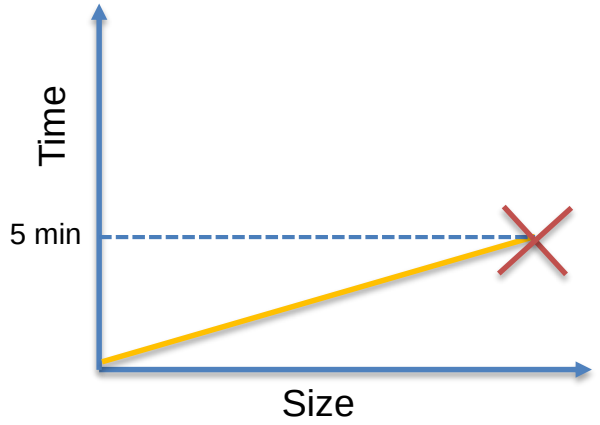
AWS Lambda Example

```
def my_lambda(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']
    key    = urllib.unquote_plus(event['Records'][0]['s3']['object']['key']).encode('utf8')

    conn_info = {…}
    with vertica_python.connect(**conn_info) as connection:
        cur = connection.cursor()
        cur.execute(“ALTER SESSION SET UDPARAMETER …”)
        cur.execute(“COPY T FROM 's3://{}/{}‘”.format(bucket, key))
```

Problem

- Cloud limitations and costs for long-running jobs. E.g. 5 min hard limit on AWS Lambda jobs
- Need to implement an event-driven workflow
- No typical SQL semantic for fire-and-forget model
- Not economical and efficient to implement externally



Alternative Approach

- Use ECS to run your long-running jobs in containers on EC2
  - Scheduling flexibility but database running your query can do a better job
  - Failover needs to be handled manually
  - Higher cost of resources (EC2 instances)

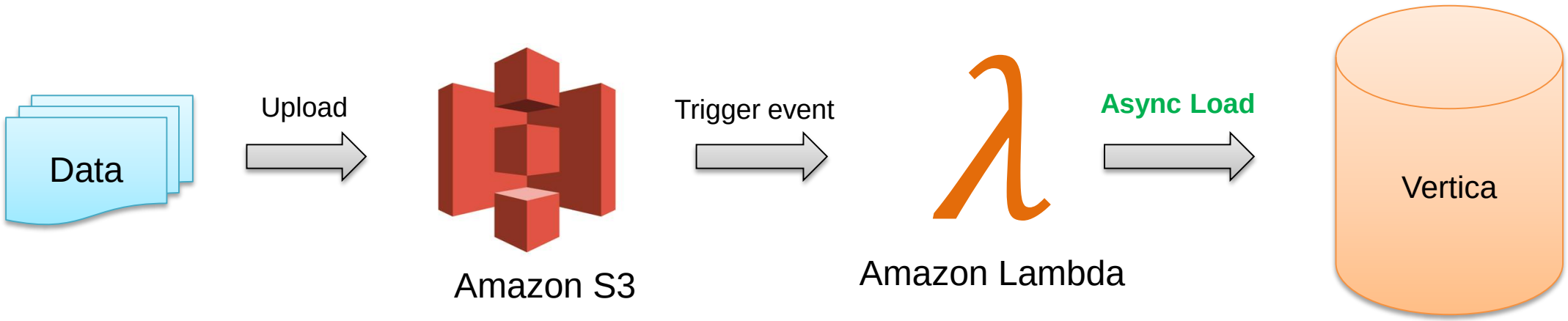
Background

- **Vertica is**
  - Column store
  - Distributed
  - Shared-nothing
  - Transactional
  - Fault tolerant
  - SQL engine

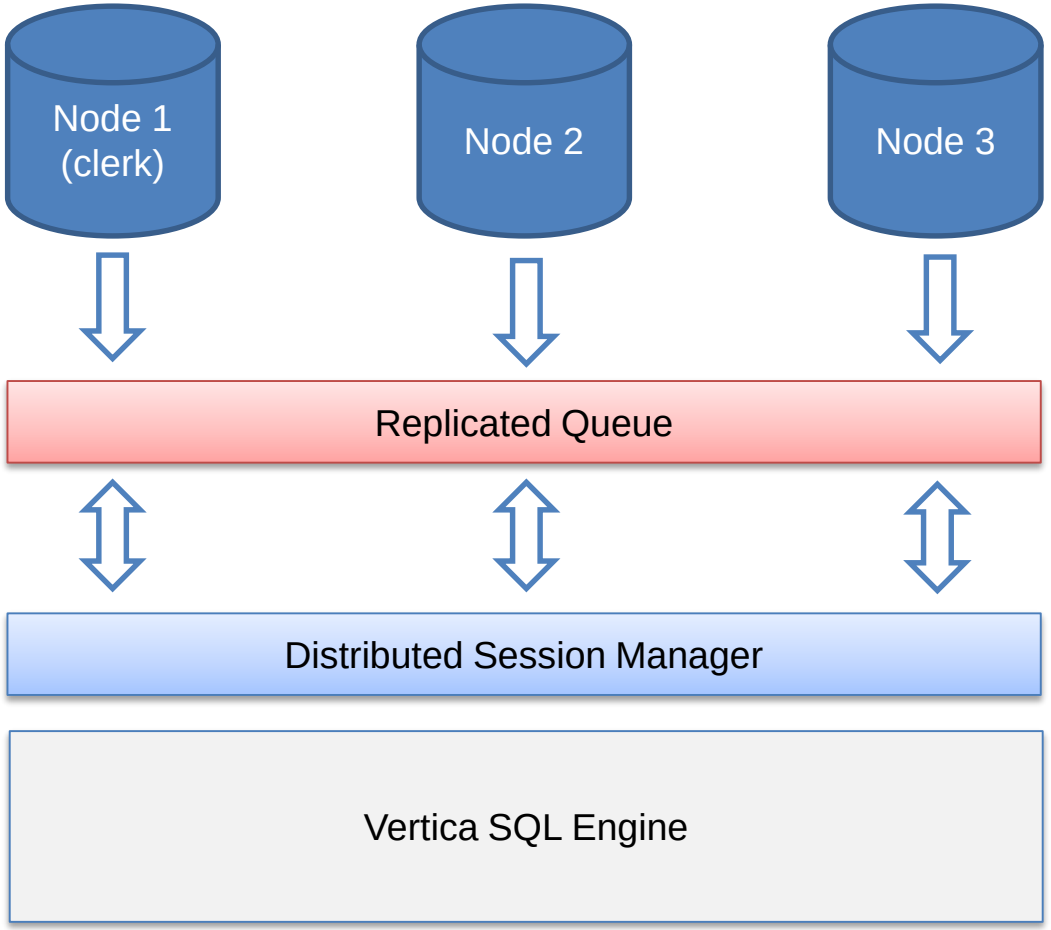
Solution

- Run long-running queries in background sessions
- Implement asynchronous semantics in the database with SQL interface
  - `ASYNC <QUERY>`
- Build on top of Vertica's existing components to handle scheduling, parallelism, and fault tolerance
  - Distributed Session Manager
  - Resource manager
  - Reliable and ordered message delivery

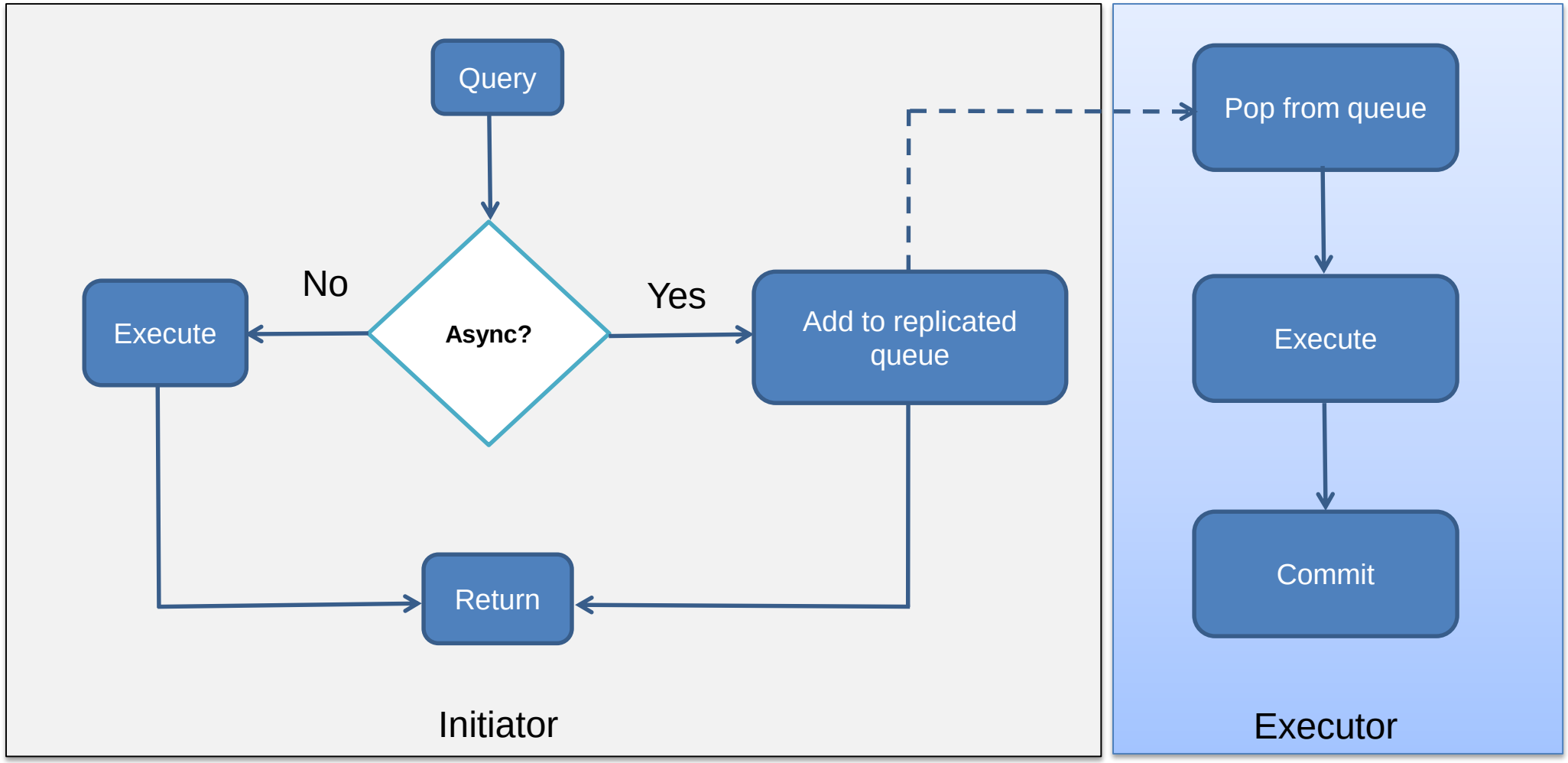
Example of an event-driven workflow



How does it work?



How does it work?



Semantics

- Asynchronous transactions
- Sessions and session-scoped objects
  - Parameters
  - Temporary tables

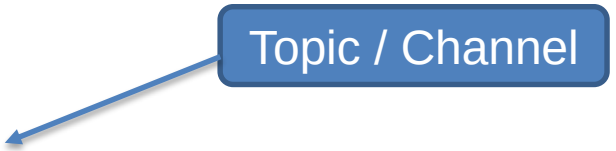
Get Notified

- Build on top of **notifiers**, our push-based messaging system

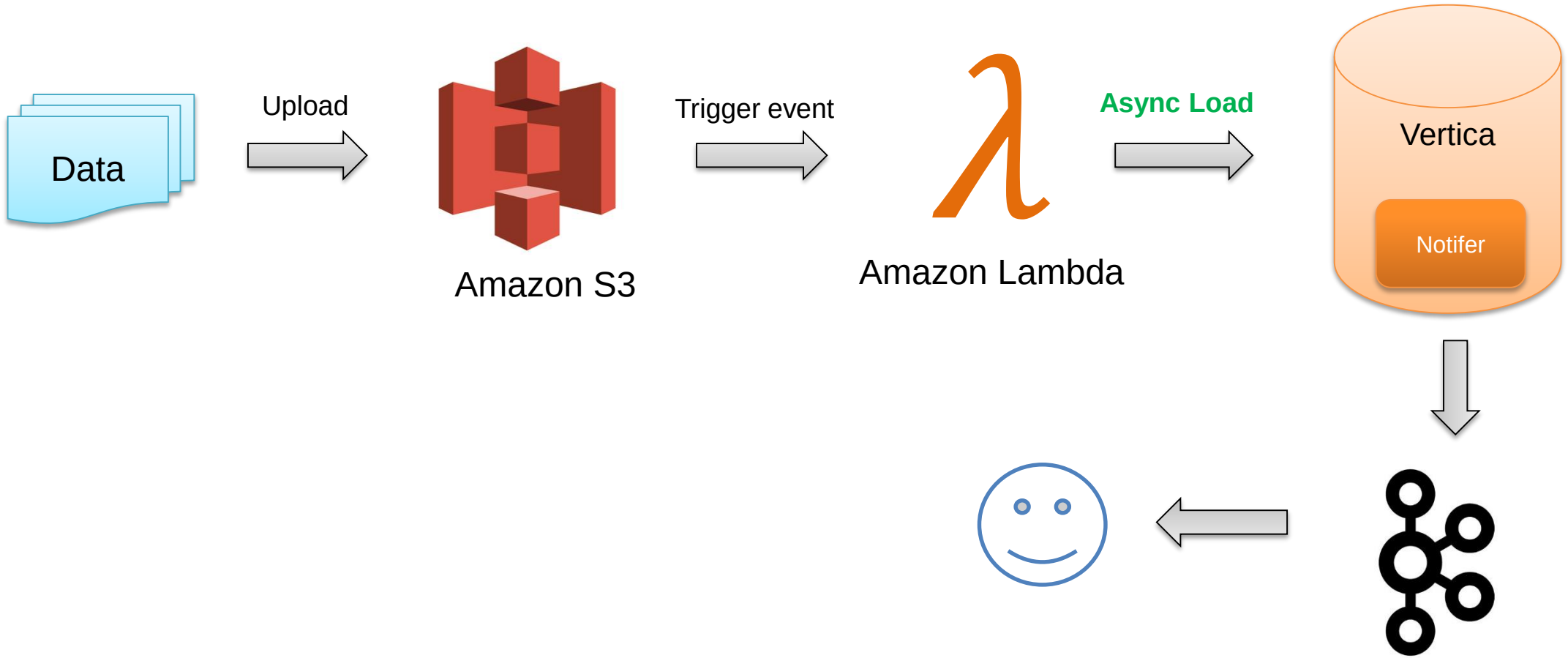
```
CREATE NOTIFIER my_etl_notifier
                ACTION 'kafka://1.2.3.4:9092'
```

- Notifiers are mostly used for pushing monitoring data
- Asynchronous query results/events can be pushed out using notifiers

```
ASYNC COPY t FROM 's3://bucket/key'
      ON COMPLETE NOTIFY my_etl_notifier 'done_events'
      ON ERROR NOTIFY my_etl_notifier 'error_events'
```



Example of an event-driven workflow



AWS Lambda Example - Async

```
def my_lambda(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']
    key    = urllib.unquote_plus(event['Records'][0]['s3']['object']['key']).encode('utf8')

    conn_info = {…}
    with vertica_python.connect(**conn_info) as connection:
        cur = connection.cursor()
        cur.execute("ALTER SESSION SET UDPARAMETER …")
        cur.execute("ASYNC COPY T FROM 's3://{}/{}' ON ERROR NOTIFY etl_notifier 'errors'".format(bucket, key))
```

Next Steps

- Background session resource management
- Failover exactly-once implementation
- Monitoring asynchronous jobs
- Async queries with results delivered via notifiers or S3

Thank you

- Visit our website at my.vertica.com
- Contact me at amin@hpe.com
- We are hiring!

Hewlett Packard
Enterprise

Questions

Hewlett Packard
Enterprise