AWS re:Invent

BDT404

Large-Scale ETL Data Flows

With Data Pipeline and Dataduct

Sourabh Bajaj, Software Engineer, Coursera

October 2015

coursera

amazon web services

What to Expect from the Session

- Learn about:
  - How we use AWS Data Pipeline to manage ETL at Coursera
  - Why we built **Dataduct**, an open source framework from Coursera for running pipelines
  - How Dataduct enables developers to write their own ETL pipelines for their services
  - Best practices for managing pipelines
  - How to start using Dataduct for your own pipelines

coursera

Institutions Log In Sign Up

Take the world's best courses, online.

What would you like to learn about?

or browse catalog ›

15,202,188 learners • 1,339 courses • 127 partners

Newest Specializations

Executive Data Science
Johns Hopkins University

Strategic Business Analytics
ESSEC Business School

Hotel Management: Distribution, Revenue and...
ESSEC Business School

Java Programming: Object-Oriented Design of Data...
University of California, San Die...

Desenvolvimento e Design de Aplicativos para iPhone
Universidade Estadual de Cam...

An Introduction to Programming the Internet...
University of California, Irvine

Data Warehousing for Business Intelligence
University of Colorado System

Data Science at Scale
University of Washington


IESE Business School University of Navarra
BROWN
RUTGERS
MANCHESTER 1824 The University of Manchester
TEL AVIV UNIVERSITY
CASE WESTERN RESERVE UNIVERSITY
COMMONWEALTH EDUCATION TRUST
University of Colorado Boulder
ARTS
UNIL Université de Lausanne
THE UNIVERSITY OF MELBOURNE
UNIVERSITY of ROCHESTER
CURTIS
DTU
ENS
expl
NANYANG TECHNOLOGICAL UNIVERSITY
Georgia Tech
HEC PARIS
The more you know, the more you dare*
JOHNS HOPKINS UNIVERSITY
LUND UNIVERSITY
STVDIVM VRBIS
TU/e
UNM
UNIVERSITY OF COPENHAGEN
LMU
KOÇ UNIVERSITY
NATIONAL GEOGRAPHIC
New Teacher Center
NUS National University of Singapore
UNIVERSITY OF MICHIGAN
RICE
SUNY
Mount Sinai
PRINCETON UNIVERSITY
UK UNIVERSITY OF KENTUCKY
RELAY GSE
UBC
UC San Diego
UF UNIVERSITY of FLORIDA
UH SYSTEM
Nebraska
UNIVERSITY OF LONDON
THE UNIVERSITY OF EDINBURGH
UNIVERSITY OF TORONTO
CBS COPENHAGEN BUSINESS SCHOOL
VANDERBILT UNIVERSITY
WESLEYAN UNIVERSITY
ualberta.ca
MoMA
Duke UNIVERSITY
EMORY
Berklee college of music
McMaster University
UAB Universitat Autònoma de Barcelona
BOARD OF REGENTS OF THE UNIVERSITY SYSTEM OF GEORGIA
KAIST
MIPT
UNIVERSITY of CALIFORNIA IRVINE
CENTRALE PARIS
PENNSTATE
ÉCOLE POLYTECHNIQUE
match EDUCATION
UCSF
Bocconi
EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
UNIVERSITY of CALIFORNIA SANTA CRUZ
NORTHWESTERN UNIVERSITY
TECNOLÓGICO DE MONTERREY
AMERICAN MUSEUM of NATURAL HISTORY
UNIVERSITY


DOPAGE
MICRO BIOME
3D
GLOBE
GREEK PHILOSOPHY
Penn Arts & Sciences
BOOTCAMP
Wharton Operations
Wharton Accounting
Wharton Finance
Wharton Marketing
DINO 101
DRUG POLICY
START

Education at Scale

15 million
learners worldwide

1300
courses

120
partners

2.5 million
course completions

Data Warehousing at Coursera

Amazon Redshift

167 Amazon Redshift users

1200 EDW tables

22 source systems

6 dc1.8xlarge instances

30,000,000 queries run

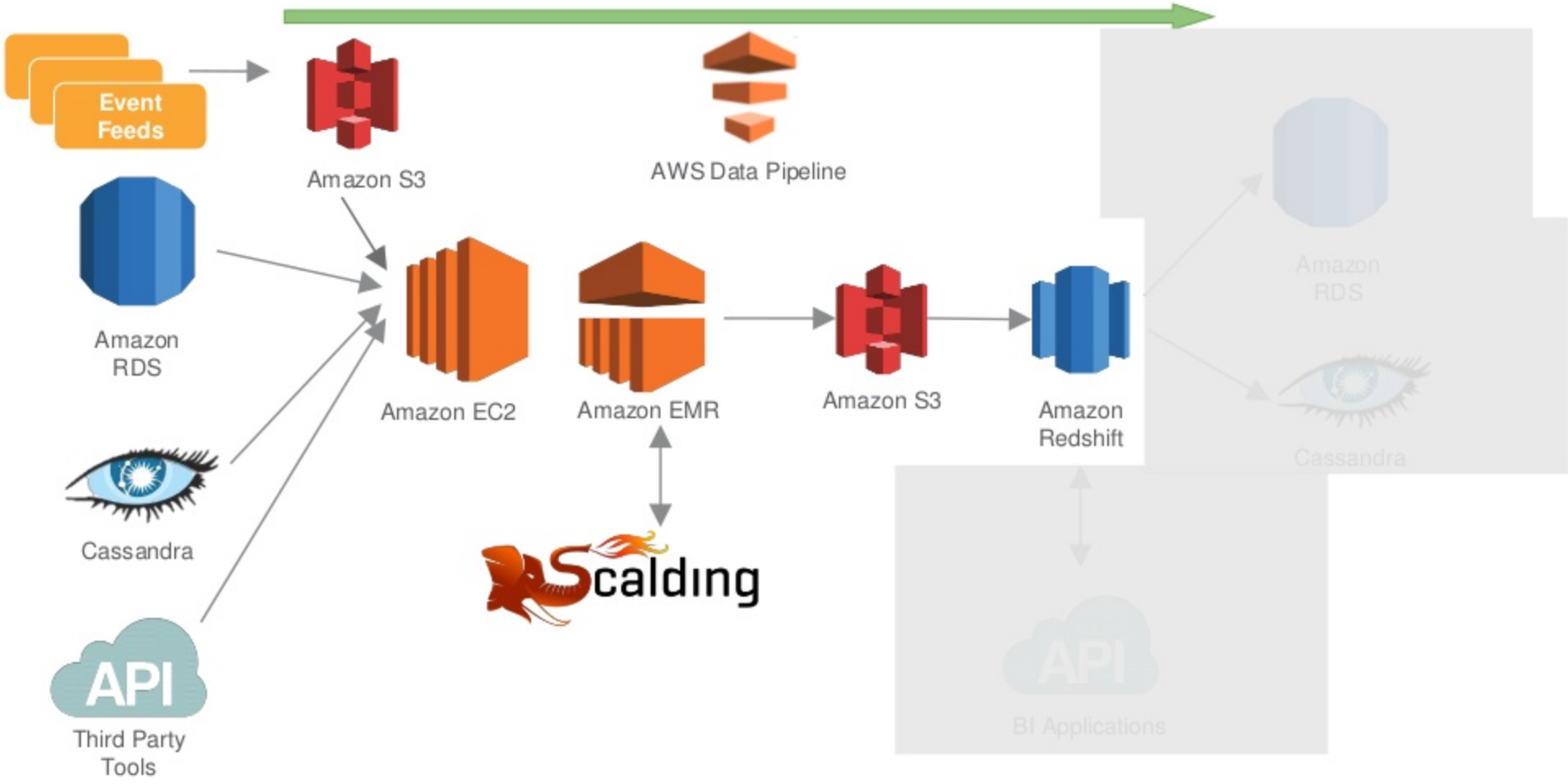
Data Flow


Event Feeds
Amazon S3
AWS Data Pipeline
Amazon RDS
Amazon EC2
Amazon EMR
Amazon S3
Amazon Redshift
Amazon RDS
Cassandra
Cassandra
Scalding
API
BI Applications
API
Third Party Tools

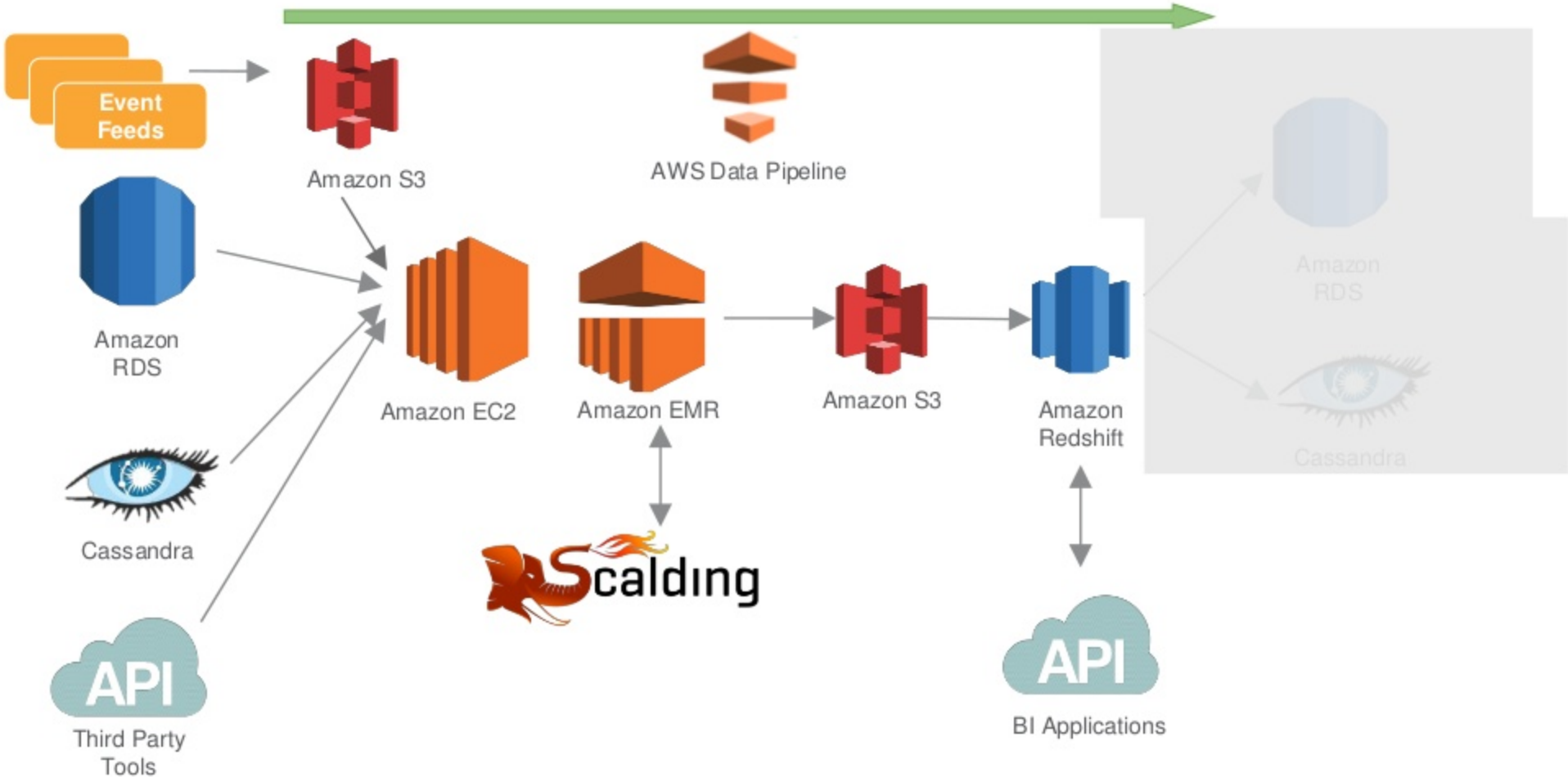
Data Flow


Event Feeds
Amazon S3
AWS Data Pipeline
Amazon RDS
Amazon EC2
Amazon EMR
Cassandra
Scalding
API
Third Party Tools
Amazon RDS
Amazon S3
Amazon Redshift
Cassandra
API
BI Applications

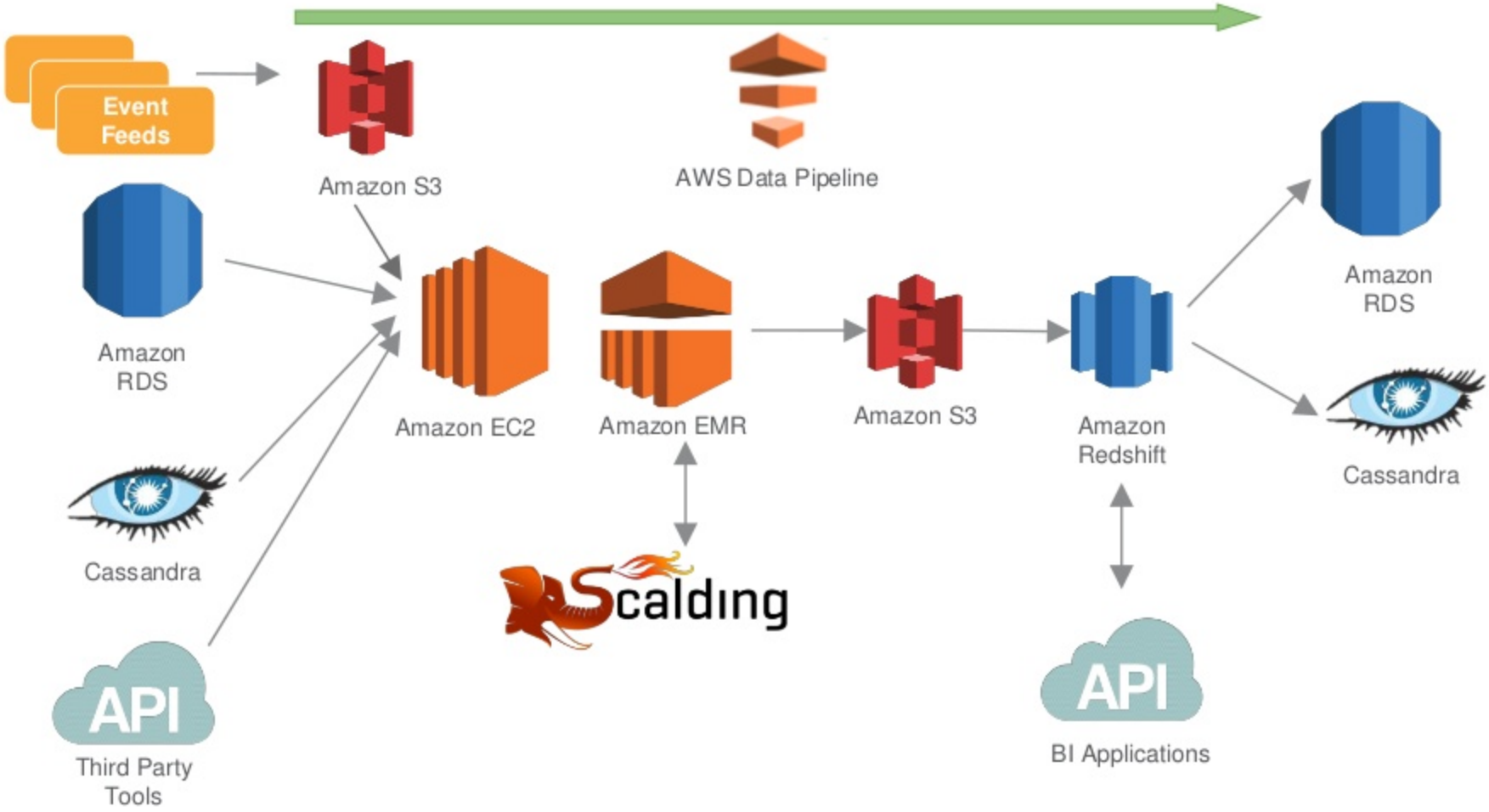
Data Flow


Event Feeds
Amazon S3
AWS Data Pipeline
Amazon RDS
Amazon EC2
Amazon EMR
Amazon S3
Amazon Redshift
Amazon RDS
Cassandra
Cassandra
Scalding
API
Third Party Tools
API
BI Applications

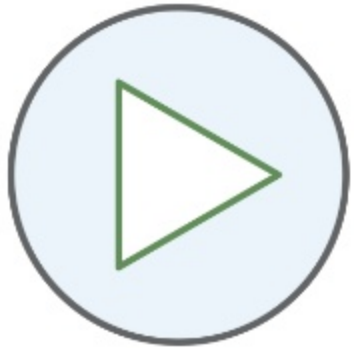
Data Flow


Event Feeds
Amazon S3
AWS Data Pipeline
Amazon RDS
Amazon EC2
Amazon EMR
Amazon S3
Amazon Redshift
Amazon RDS
Cassandra
Cassandra
Scalding
API
Third Party Tools
API
BI Applications

Data Flow


Event Feeds
Amazon S3
AWS Data Pipeline
Amazon RDS
Amazon EC2
Amazon EMR
Amazon S3
Amazon Redshift
Amazon RDS
Cassandra
Cassandra
Scalding
API
Third Party Tools
API
BI Applications

ETL at Coursera



150 Active pipelines

44 Dataduct developers

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management

Resource Management

Monitoring

Easy Development

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management

Resource Management

Monitoring

Easy Development

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management

Resource Management

Monitoring

Easy Development

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management

Resource Management

Monitoring

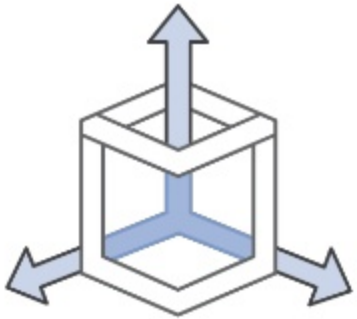
Easy Development

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management



Resource Management

Monitoring

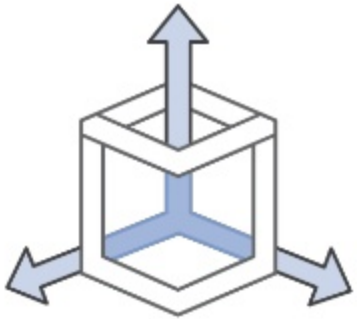
Easy Development

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management



Resource Management

Monitoring

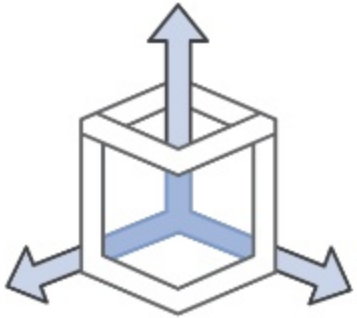
Easy Development

Requirements for an ETL system

Fault Tolerance

Scheduling

Dependency Management



Resource Management

Monitoring

Easy Development

Dataduct

Dataduct

- Open source wrapper around AWS Data Pipeline

Dataduct

- Open source wrapper around AWS Data Pipeline
- It provides:
  - Code reuse
  - Extensibility
  - Command line interface
  - Staging environment support
  - Dynamic updates

Dataduct

- Repository
  - https://github.com/coursera/dataduct
- Documentation
  - http://dataduct.readthedocs.org/en/latest/
- Installation
  - `pip install dataduct`

Let’s build some pipelines

Pipeline 1: Amazon RDS → Amazon Redshift

- Let's start with a simple pipeline of pulling data from a relational store to Amazon Redshift

Pipeline 1: Amazon RDS → Amazon Redshift

```yaml
name: users
frequency: daily
load_time: 12:00
description: Users table from RDS database

steps:
-   step_type: extract-rds
    sql: "SELECT id, user_name, user_email FROM users"
    host_name: maestro
    database: userDb

-   step_type: create-load-redshift
    name: load_staging
    table_definition: tables/staging.maestro_users.sql

-   step_type: upsert
    name: upsert_users
    source: tables/staging.maestro_users.sql
    destination: tables/prod.users.sql

```

Pipeline 1: Amazon RDS → Amazon Redshift

```yaml
name: users
frequency: daily
load_time: 12:00
description: Users table from RDS database

steps:
-   step_type: extract-rds
    sql: "SELECT id, user_name, user_email FROM users"
    host_name: maestro
    database: userDb

-   step_type: create-load-redshift
    name: load_staging
    table_definition: tables/staging.maestro_users.sql

-   step_type: upsert
    name: upsert_users
    source: tables/staging.maestro_users.sql
    destination: tables/prod.users.sql

```

- Definition in YAML
- Steps
- Shared Config
- Visualization
- Overrides
- Reusable code

Pipeline 1: Amazon RDS → Amazon Redshift (Steps)

- Extract RDS
  - Fetch data from Amazon RDS and output to Amazon S3

```
-   step_type: extract-rds
    sql: "SELECT id, user_name, user_email FROM users"
    host_name: maestro
    database: userDb
```

Pipeline 1: Amazon RDS → Amazon Redshift (Steps)

- Create-Load-Redshift
  - Create table if it doesn't exist and load data using COPY.

```
-   step_type: create-load-redshift
    name: load_staging
    table_definition: tables/staging.maestro_users.sql
```

Pipeline 1: Amazon RDS → Amazon Redshift

- Upsert
  - Update and insert into the production table from staging.

```
-   step_type: upsert
    name: upsert_users
    source: tables/staging.maestro_users.sql
    destination: tables/prod.users.sql
```

Pipeline 1: Amazon RDS → Amazon Redshift (Tasks)

- Bootstrap
  - Fully automated
  - Fetch latest binaries from Amazon S3 for Amazon EC2 / Amazon EMR
  - Install any updated dependencies on the resource
  - Make sure that the pipeline would run the latest version of code

Pipeline 1: Amazon RDS → Amazon Redshift

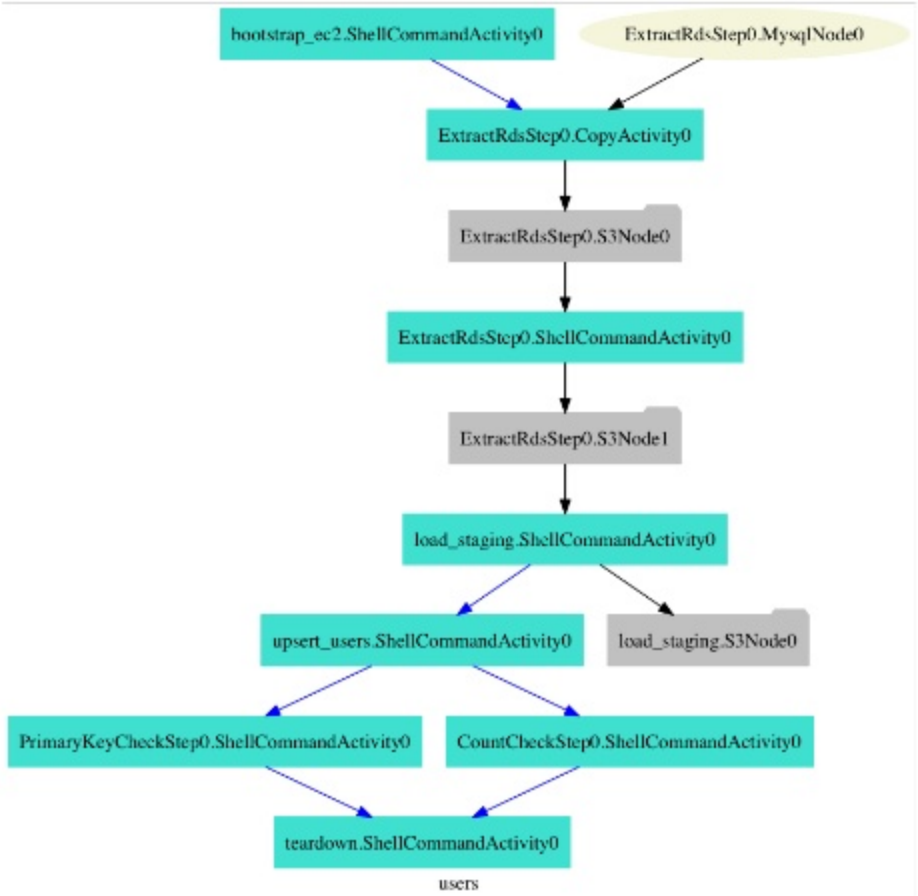
- Quality assurance
  - Primary key violations in the warehouse.
  - Dropped rows: By comparing the number of rows.
  - Corrupted rows: By comparing a sample set of rows.
  - Automatically done within UPSERT

Pipeline 1: Amazon RDS → Amazon Redshift

- Teardown
  - Amazon SNS alerting for failed tasks
  - Logging of task failures
  - Monitoring
    - Run times
    - Retries
    - Machine health

Pipeline 1: Amazon RDS → Amazon Redshift (Config)

- Visualization
  - Automatically generated by Dataduct
  - Allows easy debugging



Pipeline 1: Amazon RDS → Amazon Redshift

- Shared Config
  - IAM roles
  - AMI
  - Security group
  - Retries
  - Custom steps
  - Resource paths

```
ec2:
    ETL_AMI: ami-db995ab0
    INSTANCE_TYPE: m1.small
    SECURITY_GROUP: reinvent

etl:
    CONNECTION_RETRIES: 2
    RETRY_DELAY: 10 Minutes
    REGION: us-east-1
    ROLE: DataPipelineDefaultRole
    S3_BASE_PATH: dev/sbajaj
    S3_ETL_BUCKET: coursera-reinvent
```

Pipeline 1: Amazon RDS → Amazon Redshift

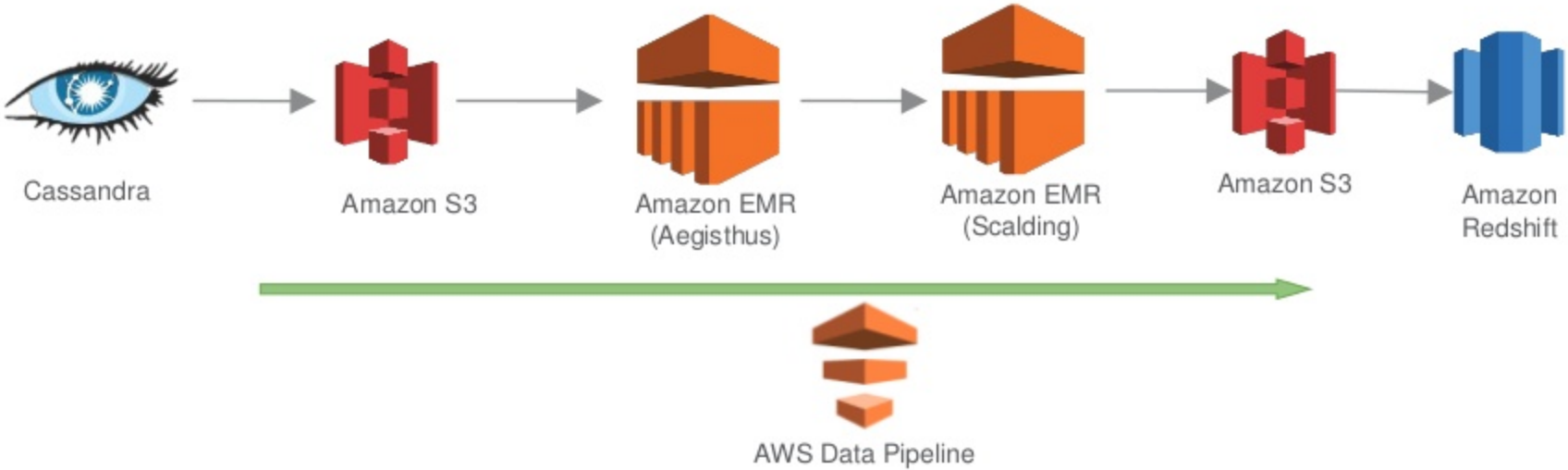
- Custom steps
  - Open-sourced steps can easily be shared across multiple pipelines
  - You can also create new steps and add them using the config

Deploying a pipeline

- Command line interface for all operations

usage: **dataduct pipeline activate [-h] [-m MODE] [-f] [-t TIME_DELTA] [-b]**
pipeline_definitions
[pipeline_definitions ...]

Pipeline 2: Cassandra → Amazon Redshift



Pipeline 2: Cassandra → Amazon Redshift

- Shell command activity to rescue

Pipeline 2: Cassandra → Amazon Redshift

- Shell command activity to rescue
- Priam backups of Cassandra to Amazon S3

Pipeline 2: Cassandra → Amazon Redshift

- Shell command activity to rescue
- Priam backups of Cassandra to S3
- Aegisthus to parse SSTables into Avro dumps

Pipeline 2: Cassandra → Amazon Redshift

- Shell command activity to rescue
- Priam backups of Cassandra to Amazon S3
- Aegisthus to parse SSTables into Avro dumps
- Scalding to process Aegisthus output
  - Extend the base steps to create more patterns

Pipeline 2: Cassandra → Amazon Redshift

```
name: discussion_votes
frequency: daily

emr_cluster_config:
    num_instances: 20
    instance_size: m1.xlarge

steps:
-   step_type: aegisthus
    cql_schema_definition: cassandra_tables/vote.vote_kvs_timestamp.cql

-   step_type: scalding
    name: vote_emr
    job_name: org.coursera.etl.vote.VotesETLJob
    output_node:
    -   questions
    -   answers

# Discussion question votes
-   step_type: create-load-redshift
    name: load_discussion_question_votes
    input_node: discussion/questions
    depends_on: vote_emr
    table_definition: tables/staging.cassandra_discussion_question_votes.sql
```

Pipeline 2: Cassandra → Amazon Redshift

- Custom steps
  - Aegisthus
  - Scalding

```
steps:
-   step_type: aegisthus
    cql_schema_definition: cassandra_tables/vote.vote_kvs_timestamp.cql

-   step_type: scalding
    name: vote_emr
    job_name: org.coursera.etl.vote.VotesETLJob
    output_node:
    -   questions
    -   answers
```

Pipeline 2: Cassandra → Amazon Redshift

- EMR-Config overrides the defaults

```
emr_cluster_config:
    num_instances: 20
    instance_size: m1.xlarge
```

Pipeline 2: Cassandra → Amazon Redshift

- Multiple output nodes from transform step

```
-   step_type: scalding
    name: vote_emr
    job_name: org.coursera.etl.vote.VotesETLJob
    output_node:
    -   questions
    -   answers
```

Pipeline 2: Cassandra → Amazon Redshift

- Bootstrap
  - Save every pipeline definition
  - Fetching new jar for the Amazon EMR jobs
  - Specify the same Hadoop / Hive metastore installation

Data products

- We've talked about data into the warehouse
- Common pattern:
  - Wait for dependencies
  - Computation inside redshift to create derived tables
  - Amazon EMR activities for more complex process
  - Load back into MySQL / Cassandra
  - Product feature queries MySQL / Cassandra
- Used in recommendations, dashboards, and search

Recommendations

- Objective
  - Connecting the learner to right content
- Use cases:
  - Recommendations email
  - Course discovery
  - Reactivation of the users

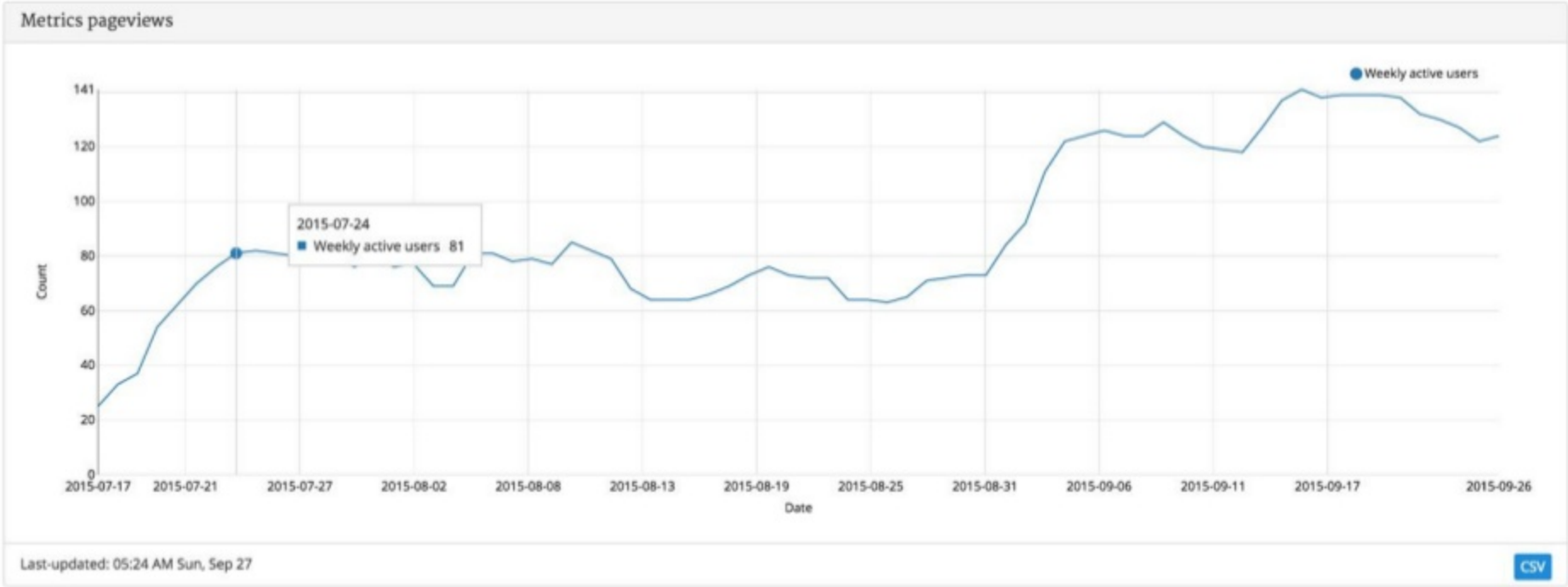
Recommendations

- Computation inside Amazon Redshift to create derived tables for co-enrollments
- Amazon EMR job for model training
- Model file pushed to Amazon S3
- Prediction API uses the updated model file
- Contract between the prediction and the training layer is via model definition.

Internal Dashboard

- Objective
  - Serve internal dashboards to create a data driven culture
- Use Cases
  - Key performance indicators for the company
  - Track results for different A/B experiments

Internal Dashboard



Learnings

- Do:
  - Monitoring (run times, retries, deploys, query times)

Learnings

- Do:
  - Monitoring (run times, retries, deploys, query times)
  - Code should live in library instead of scripts being passed to every pipeline

Learnings

- Do:
  - Monitoring (run times, retries, deploys, query times)
  - Code should live in library instead of scripts being passed to every pipeline
  - Test environment in staging should mimic prod

Learnings

- Do:
  - Monitoring (run times, retries, deploys, query times)
  - Code should live in library instead of scripts being passed to every pipeline
  - Test environment in staging should mimic prod
  - Shared library to democratize writing of ETL

Learnings

- Do:
  - Monitoring (run times, retries, deploys, query times)
  - Code should live in library instead of scripts being passed to every pipeline
  - Test environment in staging should mimic prod
  - Shared library to democratize writing of ETL
  - Using read-replicas and backups

Learnings

- Don’t:
  - Same code passed to multiple pipelines as a script

Learnings

- Don’t:
  - Same code passed to multiple pipelines as a script
  - Non version controlled pipelines

Learnings

- Don’t:
  - Same code passed to multiple pipelines as a script
  - Non version controlled pipelines
  - Really huge pipelines instead of modular small pipelines with dependencies

Learnings

- Don’t:
  - Same code passed to multiple pipelines as a script
  - Non version controlled pipelines
  - Really huge pipelines instead of modular small pipelines with dependencies
  - Not catching resource timeouts or load delays

Dataduct

- Code reuse
- Extensibility
- Command line interface
- Staging environment support
- Dynamic updates

Dataduct

- Repository
  - https://github.com/coursera/dataduct
- Documentation
  - http://dataduct.readthedocs.org/en/latest/
- Installation
  - `pip install dataduct`

Questions?

Also, we are hiring!

https://www.coursera.org/jobs

Remember to complete your evaluations!

AWS re:Invent

Thank you!

Sourabh Bajaj
sb2nov
@sb2nov

Also, we are hiring!
https://www.coursera.org/jobs