



BDT303

# Construct Your ETL with AWS Data Pipeline, Amazon EMR, and Amazon Redshift

Roy Ben-Alta (Big Data Analytics Practice Lead, AWS)  
Thomas Barthelemy (Software Engineer, Coursera)

November 13, 2014 | Las Vegas | NV



# AWS Big Data Portfolio

Collect



AWS Direct Connect



AWS Import/Export



Amazon Kinesis

Store



Amazon S3



DynamoDB



Amazon Glacier

Process & Analyze



Amazon EMR

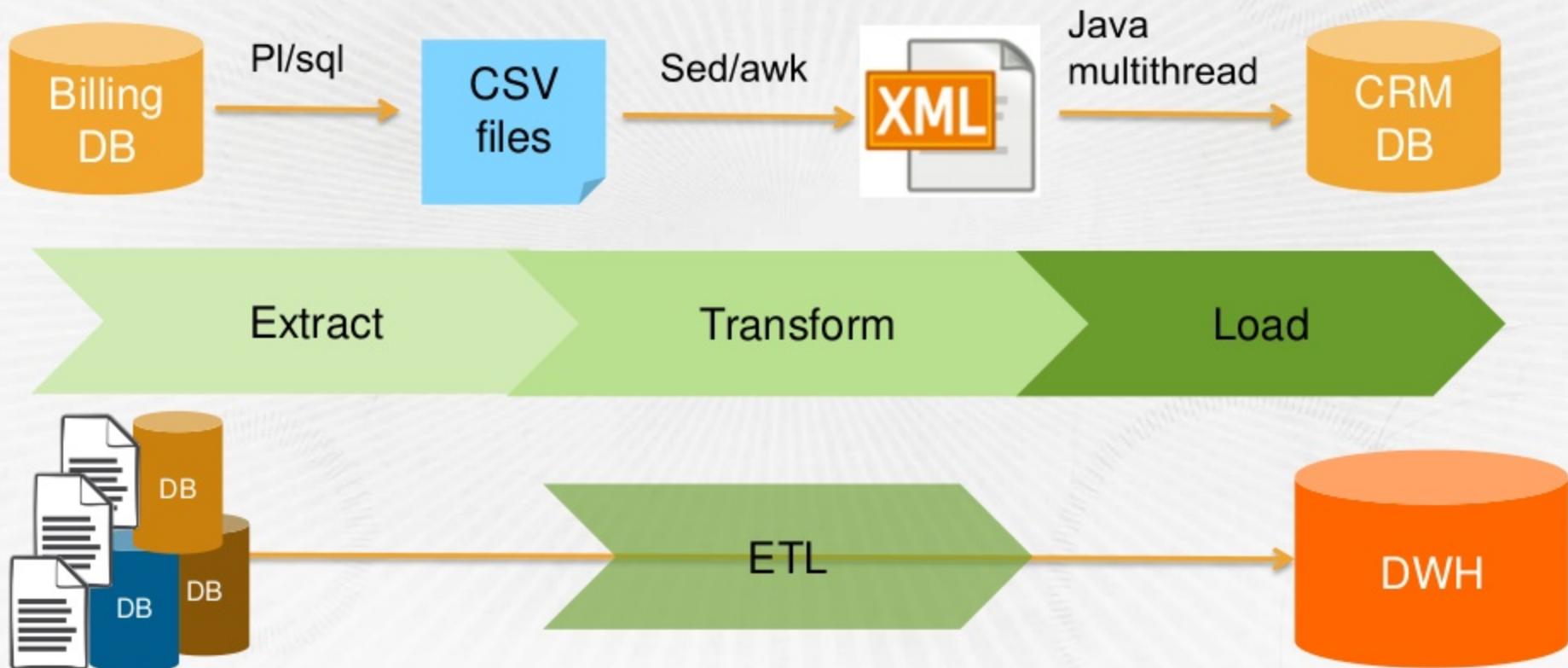


Amazon Redshift



Amazon EC2

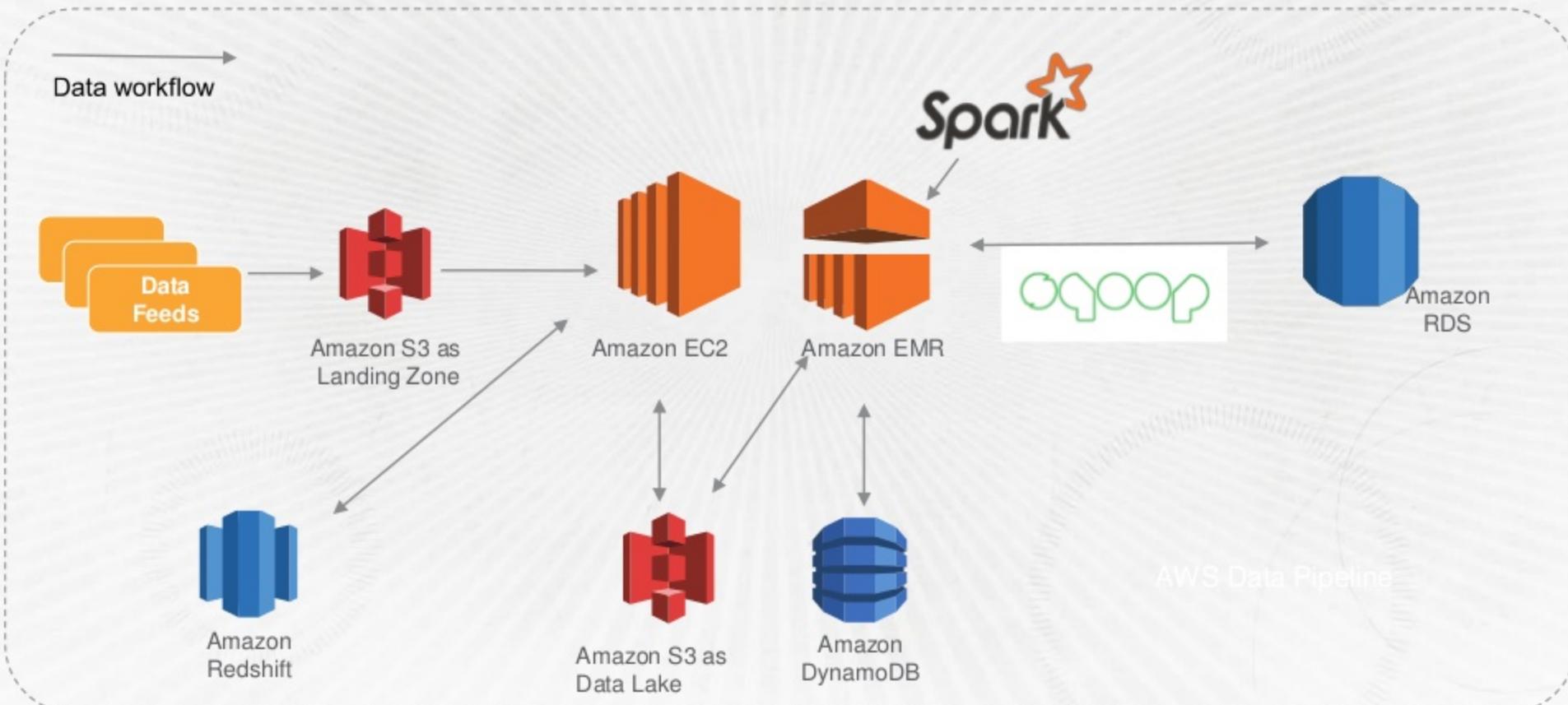
# Hello ETL World



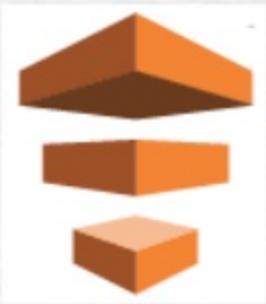
# ETL as Data Driven Workflow

- What happens if your data doesn't arrive on time?
- What happens if one of your processing steps fails?
- What happens if a transient error takes out one of your processing nodes?
- How do you operate/monitor?
- How do you control Scheduling?

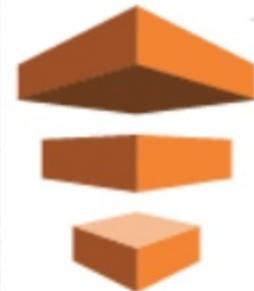
# Data Flow on AWS



# AWS Data Pipeline is...



## ETL of Things



# AWS Data Pipeline

- Templates Repository
- Manages your workflow data orchestration for you based on activities you define
- Automates creation and management of your resources
- Manages dependencies and automated scheduling for you
- Setup notification/alerts using Amazon SNS (On Job Failure/Success)

# ETL with AWS Data Pipeline



## AWS Data Pipeline

AWS Data Pipeline helps you move, integrate, and process data across AWS compute and storage resources, as well as your on-premises resources. AWS Data Pipeline supports integration of data and activities across multiple AWS regions.

[Get started now](#)



### Define Data Nodes

Select input and output data out of S3, DynamoDB, Redshift, RDS, and JDBC sources.

[Learn more](#)



### Schedule Compute Activities

Configure the activities that will process data from EMR, Hive, Pig, SQL, and Shell scripts.

[Learn more](#)



### Activate & Monitor

Activate your pipeline, then let AWS Data Pipeline manage the pipeline execution, resources, retry logic and failure notifications for you.

[Learn more](#)

# Build and Deploy AWS Data Pipeline

- Console
- CLI
- APIs
  - AWS Java SDK
  - AWS Python SDK
  - AWS Ruby SDK

<http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-program-pipeline.html>

# Use Case - Clickstreams

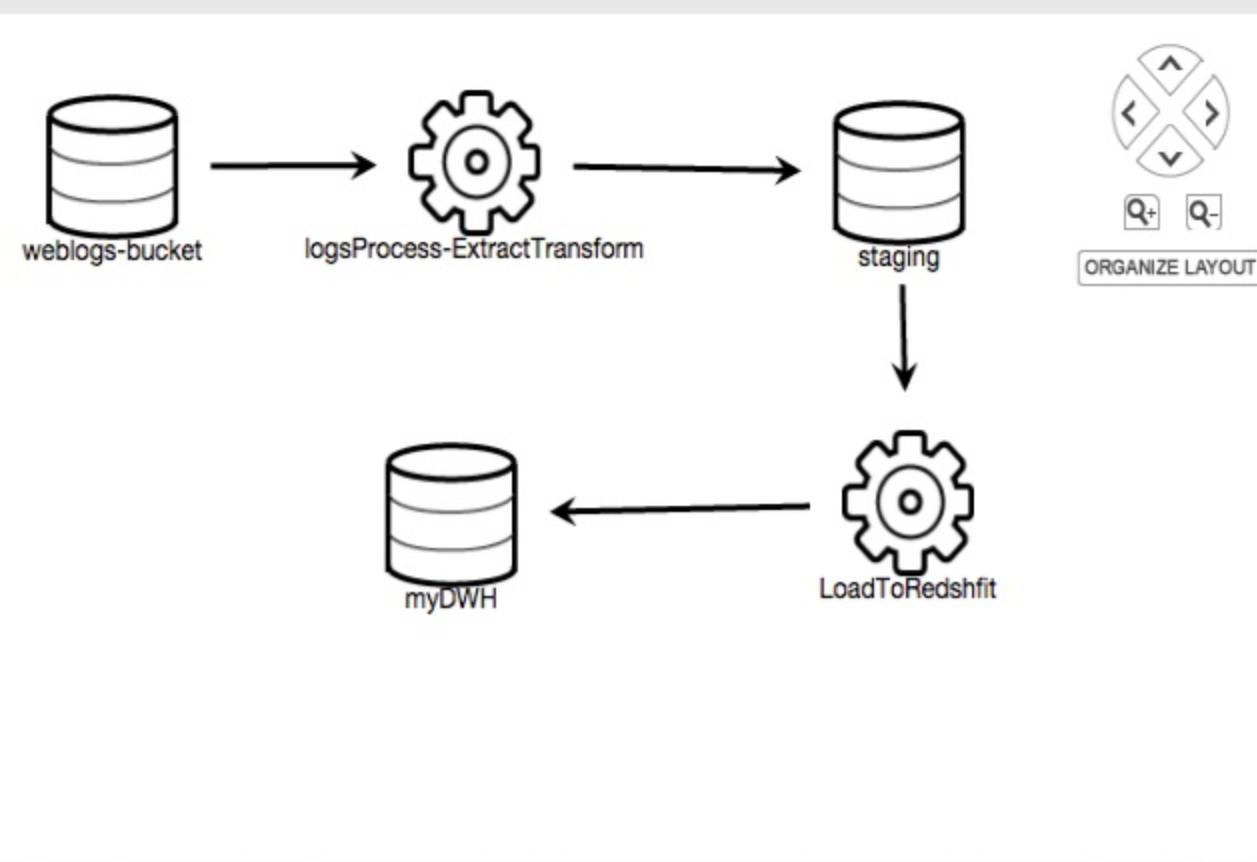
```
tongue-twisting food terms 5243470 Jenni  
tongue-twisting food terms 5243470 Jenni  
ist post leads to stolen bike&#8217;s i  
38703008004 285369353059736088069175308218  
ist post leads to stolen bike&#8217;s i  
31528812854 285929534262353237469175301695  
et Worth Plus:How much income do you need t  
10089951696650244650285088697485061 28607:  
tosses huge tip from &#8216;Tips for Je  
60842068623 28607103195991961254611687189:  
Beckham: &#8216;I had my breast implan
```



Amazon Redshift –  
Production DWH



Visualization

[Add activity](#) [Add data node](#) [Save pipeline](#) [Activate](#) [Export](#) [Templates ▾](#)**Activities**logsProcess-ExtractTransform X

Name:	logsProcess-ExtractTransform
Type:	PigActivity
Schedule:	RunOnce
Input:	weblogs-bucket
Generated Scripts Path:	s3://reinventbd303/scripts/
Script Uri:	s3://reinventbd303/parseltNow/

## ▶ DataNodes

## ▶ Schedules

## ▶ Resources

## ▶ Preconditions

## ▶ Others

## ▶ Errors/Warnings

# AWS Data Pipeline CLI

## Define

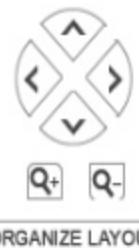
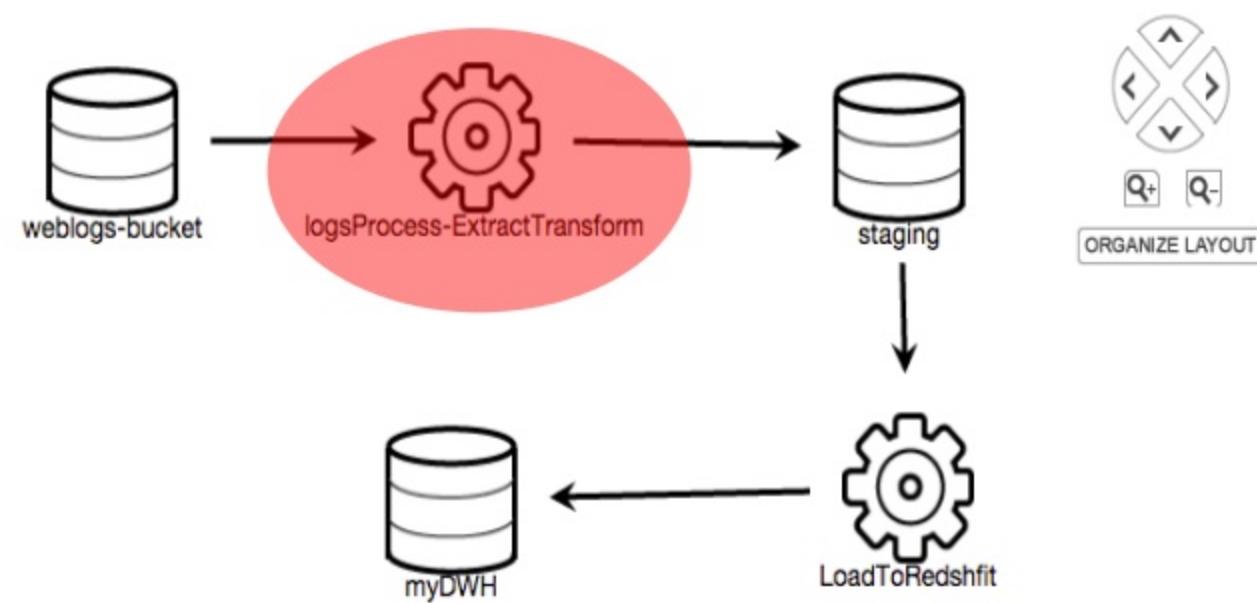
```
aws datapipeline create-pipeline --name myETL --unique-id token  
--output: df-09222142C63VXJU0HC0A
```

## Import

```
aws datapipeline put-pipeline-definition --pipeline-id df-  
09222142C63VXJU0HC0A --pipeline-definition /home/repo/etl_reinvent.json
```

## Activate

```
aws datapipeline activate-pipeline --pipeline-id df-09222142C63VXJU0HC0A
```

[Add activity](#) [Add data node](#) [Save pipeline](#) [Activate](#) [Export](#) [Templates ▾](#)

ORGANIZE LAYOUT

## ▼ Activities

## logsProcess-ExtractTransform

Name:	logsProcess-ExtractTransform
Type:	PigActivity
Schedule:	RunOnce
Input:	weblogs-bucket
Generated Scripts Path:	s3://reinventbd303/scripts/
Script Uri:	s3://reinventbd303/parseltNow

## ▶ DataNodes

## ▶ Schedules

## ▶ Resources

## ▶ Preconditions

## ▶ Others

## ▶ Errors/Warnings

# Extract and Transform Using Pig

```
register /home/hadoop/lib/pig/piggybank.jar;
DEFINE EXTRACT org.apache.pig.piggybank.evaluation.string.EXTRACT;

RAW_LOGS = LOAD 's3://elasticmapreduce/samples/pig-apache/input/' USING
TextLoader as (line:chararray);
LOGS_BASE = FOREACH RAW_LOGS GENERATE
FLATTEN(
    EXTRACT(line, '^(\S+) (\S+) (\S+) \\\[(\w:/+\s[+\-\d{4})\\] "(.
+?)" (\S+) (\S+) "([^\"]*)" "([^\"]*")'
)
as (
    host:     chararray,
    identity: chararray,
    user:     chararray,
    request_time:     chararray,
    request:     chararray,
    status:     int,
    size:     chararray,
    referrer:     chararray,
    agent:     chararray
);
PARS = FOREACH LOGS_BASE GENERATE host,user,referrer,request_time;
STORE PARS into 's3://reinventbdt303/staging3/' USING PigStorage(',');
```

# PigActivity

- Setup Schedule
- Retry
- On Fail/On Success
- Define Input
- Define Output
- **Runs On**

The screenshot shows a configuration interface for a PigActivity. The activity is named "logsProcess-ExtractTransform" and is of type "PigActivity". It is scheduled to run once and has input from "weblogs-bucket". The "Generated Scripts Path" is set to "s3://reinventbdt303/scripts/" and the "Script Uri" is "s3://reinventbdt303/parseltN". The "Runs On" field is currently set to "myEMR" and has a dropdown menu open, showing options like "Add an optional field...", "Attempt Status", "Attempt Timeout", "Depends On", "Input", "Late After Timeout", "Max Active Instances", "Maximum Retries", "On Fail", "On Late Action", "On Success", "Output", "Parent", "Precondition", and "Report Progress Timeout".

Name:	logsProcess-ExtractTransform
Type:	PigActivity
Schedule:	RunOnce
Input:	weblogs-bucket
Generated Scripts Path:	s3://reinventbdt303/scripts/
Script Uri:	s3://reinventbdt303/parseltN
Runs On:	myEMR
Output:	
Stage:	

Runs On Options (Dropdown):

- ✓ Add an optional field...
- Attempt Status
- Attempt Timeout
- Depends On
- Input
- Late After Timeout
- Max Active Instances
- Maximum Retries
- On Fail
- On Late Action
- On Success
- Output
- Parent
- Precondition
- Report Progress Timeout

Navigation:

- ▶ DataNodes
- ▶ Schedules
- ▶ Resources
- ▶ Preconditions
- ▶ Others
- ▶ Errors/Warnings

# Amazon EMR Resource

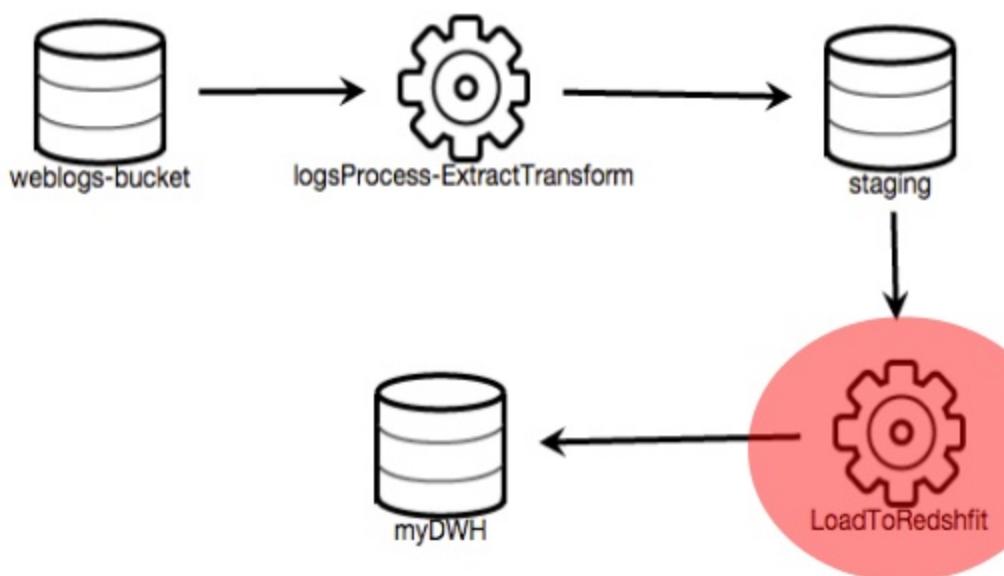
- Transient Cluster
- Use Spot Instance
- Select Instance type
- Other 20+ Optional attributes you can define
- Bootstrap EMRFS

myEMR

Name:	myEMR
Type:	EmrCluster
Terminate After:	30 Minute(s)
Ami Version:	2.4.8
Schedule:	RunOnce
Master Instance Type:	m1.large
Core Instance Type:	m1.large
Core Instance Count:	10
Task Instance Bid Price:	1.57

# Amazon EMR Resource - EMRFS

```
{"terminateAfter": "1 Hour",
"id": "EmrClusterId_tX1ME",
"amiVersion": "3.2.1",
"schedule": {
  "ref": "ScheduleId_E8jhD"},
"keyPair": "myKeyPair",
"masterInstanceType": "m1.large",
"bootstrapAction": "s3://elasticmapreduce/bootstrap-actions/configure-hadoop,-e,fs.s3.consistent=true,-e,
fs.s3.consistent.retryPeriodSeconds=10, -e, fs.s3.enableServerSideEncryption=true",
"coreInstanceType": "m3.xlarge",
"enableDebugging": "true",
"name": "DefaultEmrCluster1",
"coreInstanceCount": "3",
"type": "EmrCluster"}
```

[Add activity](#) [Add data node](#) [Save pipeline](#) [Activate](#) [Export](#) [Templates ▾](#)

### Activities

#### LoadToRedshfit

Name: LoadToRedshfit  
Type: RedshiftCopyActivity  
Schedule: RunOnce  
Input: staging  
Precondition: checkTrigger  
Command Options: copy weblogs\from

#### ▶ DataNodes

#### ▶ Schedules

#### ▶ Resources

#### ▶ Preconditions

#### ▶ Others

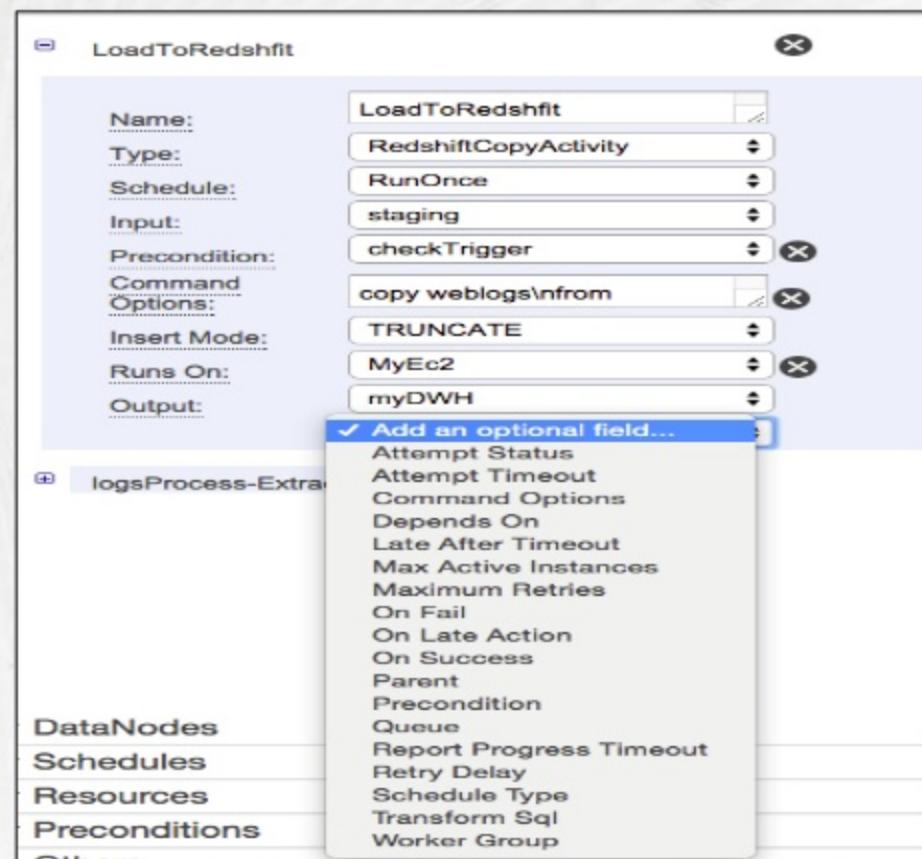
#### ▶ Errors/Warnings

# Load to Amazon Redshift

```
copy weblogs
from 's3://staging/'
credentials 'aws_access_key_id=<my access
key>;aws_secret_access_key=<my secret key>'
delimiter ','
```

# RedshiftCopyActivity

- Copy directly from Amazon S3
- Copy directly from Amazon DynamoDB
- Schedule
- **Precondition**
- Runs On – Set your Amazon EC2 engine to copy data



# Amazon Redshift Data Node

myDWH

Name:	myDWH
Type:	RedshiftDataNode
Schedule:	RunOnce
Table Name:	weblogs
Database:	RedshiftDWH

Add an optional field...

- ✓ Add an optional field...
- Attempt Status
- Attempt Timeout
- Create Table Sql
- Depends On
- Late After Timeout
- Max Active Instances
- Maximum Retries
- On Fail
- On Late Action
- On Success
- Parent
- Precondition
- Primary Keys
- Report Progress Timeout
- Retry Delay
- Runs On
- Schedule Type
- Schema Name
- Worker Group

staging

Name:	
Type:	
Schedule:	
Directory Path:	

weblogs-bucket

Name:	
Type:	

Schedules

RedshiftDWH

Name:	RedshiftDWH
Type:	RedshiftDatabase
Database Name:	xxxx
Username:	xxx
*Password:	xxxxxx
Cluster Id:	redshift

Add an optional field...

- ✓ Add an optional field...
- Connection String
- Jdbc Properties
- Parent
- Region

Errors/Warnings

# Amazon Redshift Copy Wizard

**Redshift Copy**

**DataNode: Input Data Details**

Load Input Data from: Simple Storage Service (S3) ▾  
S3 Directory Path ▾ s3://mybucket/myData

Data compression: none ▾ Data format: tsv ▾

**Schedules**

Copy of Redshift Data should happen every: 12 Hour(s) ▾  
Start (UTC): 2014-11-07 04:10:09 End (UTC): 2014-11-07 16:10:09

**Resource: Ec2Resource Details**

EC2 Security Group:

Save debug logs to the following S3 bucket location: s3://mybucket/myLogPal

---

[Back](#) [Cancel](#) [Next](#)

# AWS Data Pipeline Dashboard

Data Pipeline ▾ List Pipelines DataPipeline Help

Create new pipeline Edit Clone Delete Export

Filter: All Filter pipelines ... 12 pipelines (all loaded) C

Pipeline ID	Name	Schedule State	Health Status	Creation Time (UTC)
df-064779238MJM4ARIL19G	newETL	PENDING	Pipeline is not active	2014-10-27 16:36:40
df-023211210TAWPLQTR9E0	WebLogsETL	PENDING	Pipeline is not active	2014-10-30 01:26:58
df-09222142C63VXJU0HC0A	pipeline_name	PENDING	Pipeline is not active	2014-10-30 01:26:09
df-051320818664UI6HLEXW	my ETL	FINISHED Runs every 1 day	HEALTHY	2014-10-27 02:13:35
df-03493992HHI007IL0A43	SNS Activity	FINISHED Runs every 1 day	HEALTHY	2014-08-18 19:08:42

Schedule information

Start 2014-08-18 19:15:53 (UTC)  
End 2014-08-19 19:15:53 (UTC)  
Period Runs every 1 day

Activity	Type	Health Status	Last Completed Run (UTC)	Active Run (UTC)	Next Run (UTC)
DefaultActivity1	CopyActivity	HEALTHY	2014-08-18 19:15:53	-	-

Edit pipeline View execution details

# AWS Data Pipeline Operation

The screenshot shows the AWS Data Pipeline console interface. At the top, there's a navigation bar with links for Services (dropdown), S3, Elastic MapReduce, EC2, Data Pipeline (selected), Redshift, and Edit (dropdown). Below the navigation bar, the path is shown as Data Pipeline > List Pipelines > Execution Details: my ETL (df-051320818664UI6HLEXW). There are four buttons at the top: Edit Pipeline (blue), Rerun, Cancel, and Mark Finished.

Below these buttons are several filter options: Show dropdown set to 'all', components in dropdown set to 'any', state with dropdown set to 'Schedule Interval', Schedule Interval dropdown set to 'between', date range from '2014-10-25 02:14:09 UTC and 20'. A Filter instances... button is also present. It shows 8 instances (all loaded).

The main area is a table with the following columns: Component Name, Schedule Interval (UTC), Type, and Status. The table lists six components:

Component Name	Schedule Interval (UTC)	Type	Status
LoadToRedshfit	2014-10-27 02:14:09 - 2014-10-28 02:14:09	RedshiftCopyActivity	WAITING_FOR_RUNNER
myDWH	2014-10-27 02:14:09 - 2014-10-28 02:14:09	RedshiftDataNode	WAITING_ON_DEPENDENCIES
MyEc2	2014-10-27 02:14:09 - 2014-10-28 02:14:09	Ec2Resource	CREATING
staging	2014-10-27 02:14:09 - 2014-10-28 02:14:09	S3DataNode	FINISHED
checkTrigger	2014-10-27 02:14:09 - 2014-10-28 02:14:09	S3KeyExists	FINISHED
logsProcess-ExtractTransform	2014-10-27 02:14:09 - 2014-10-28 02:14:09	PigActivity	FINISHED

# AWS Data Pipeline Monitoring

The screenshot shows the AWS Data Pipeline monitoring interface. At the top, there's a navigation bar with 'Services' dropdown, 'S3', 'Elastic MapReduce', 'EC2', 'Data Pipeline' (which is highlighted in blue), 'Redshift', and 'Edit' dropdown. Below the navigation bar, the path 'Data Pipeline > List Pipelines > Execution Details: aaa (df-088737933XWQA943H7NE)' is displayed. There are buttons for 'Edit Pipeline', 'Rerun', 'Cancel', and 'Mark Finished'. A search/filter bar below has 'Show all components in any state with Schedule Interval between 2014-10-24 16:03:43 UTC and' and a 'Filter instances ...' dropdown set to 'All'.

Component Name	Schedule Interval (UTC)	Type	Status
staging	2014-10-26 16:03:43 – 2014-10-27 16:03:43	S3DataNode	FINISHED
myEMR	2014-10-26 16:03:43 – 2014-10-27 16:03:43	EmrCluster	SHUTTING_DOWN
weblogs-bucket	2014-10-26 16:03:43 – 2014-10-27 16:03:43	S3DataNode	FINISHED
logsProcess	2014-10-26 16:03:43 – 2014-10-27 16:03:43	PigActivity	FINISHED

# Tips

- Focus on business logic first
- Design for Failure
- Scheduling – (Min interval 15 minutes)
  - Run Once to test
  - Use Backfill
- SLA
- Build Once, Deploy many

# Coursera

# Coursera Big Data Analytics Powered by AWS

Thomas Barthelemy  
SWE, Data Infrastructure  
[thomas@coursera.org](mailto:thomas@coursera.org)

# Overview

- About Coursera
- Phase 1: Consolidate data
- Phase 2: Get users hooked
- Phase 3: Increase reliability
- Looking forward

# Coursera at a Glance

# About Coursera

- Platform for Massive Open Online Courses
- Universities create the content
- Content free to the public



Child Nutrition and Cooking  
by Stanford University



Next Up:  
**Introduction To Child Nutrition**  
3 min



UNIVERSITY OF VIRGINIA  
DARDEN SCHOOL OF BUSINESS

Foundations of Business Strategy

Learn how to analyze an organization's strategy and make recommendations to improve its value creation by building your strategist's toolkit.

Preview Lectures



Watch Intro Video

### About the Course

In this course, we will explore the underlying theory and frameworks that provide the foundations of a successful business strategy. We will develop your ability to think strategically by providing you the tools for conducting a strategic analysis. Strategic analysis is critical for analyzing the competitive environment in which an organization

### Sessions

Oct 6th 2014 - Nov 24th 2014

Join for Free

# Coursera Stats

- ~10 million learners
- +110 university partners
- >200 courses open now
- ~170 employees

# The value of data at Coursera

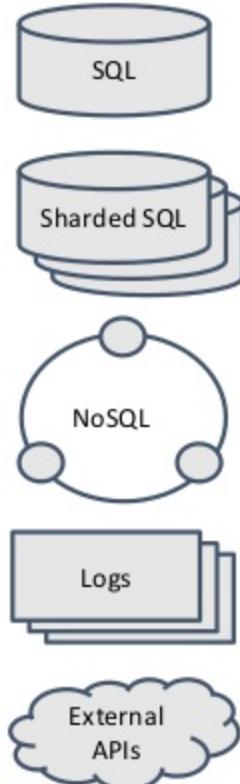
- Making strategic and tactical decisions
- Studying pedagogy

# Becoming More Data-Driven

- Since the early days, Coursera has understood the value of data
  - Founders came from machine learning
  - Many of the early employees researched with the founders
- Cost of data access was high
  - each analysis requires extraction, pre-processing
  - data only available to data scientists + engineers

# Phase 1: Consolidate data

# Sources



- MySQL
  - Site data
  - Course data sharded across multiple database
- Cassandra increasingly used for course data
- Logged event data
- External APIs

(Obligatory meme)



# What platforms to use?

- **Amazon Redshift** had glowing recommendations
- **AWS Data Pipeline** has native support for various Amazon services

# ETL development was slow :(

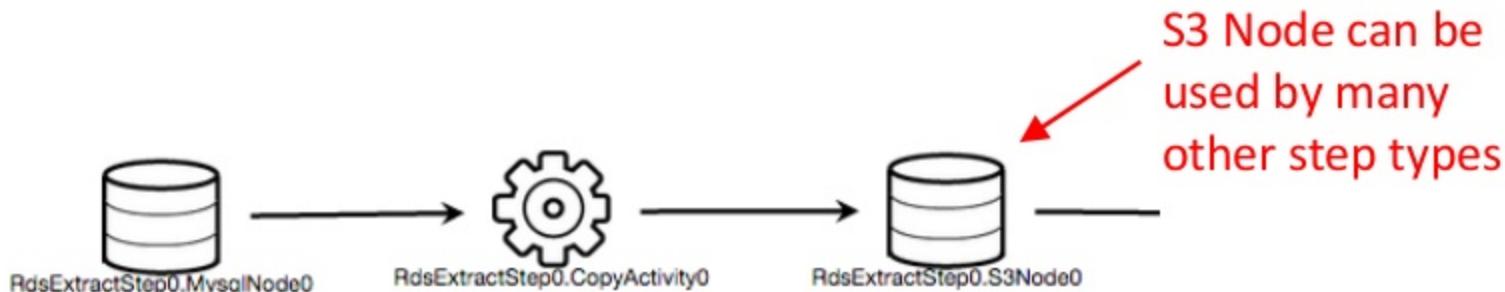
- Slow to do the following in the console:
  - create one pipeline
  - create similar pipelines
  - update existing pipelines

# Solution: Programmatically create pipelines

- Break ETL into reusable **steps**
  - Extract from variety of sources
  - Transform data with Amazon EMR, Amazon EC2, or within Amazon Redshift
  - Load data principally into Amazon Redshift
- Use Amazon S3 as intermediate state for Amazon EMR- or Amazon EC2-based transformations
- Map **steps** to set of AWS Data Pipeline objects

# Example step: extract-rds

- Parameters: hostname, database name, SQL
- Creates pipeline objects:



# Example load definition

## steps:

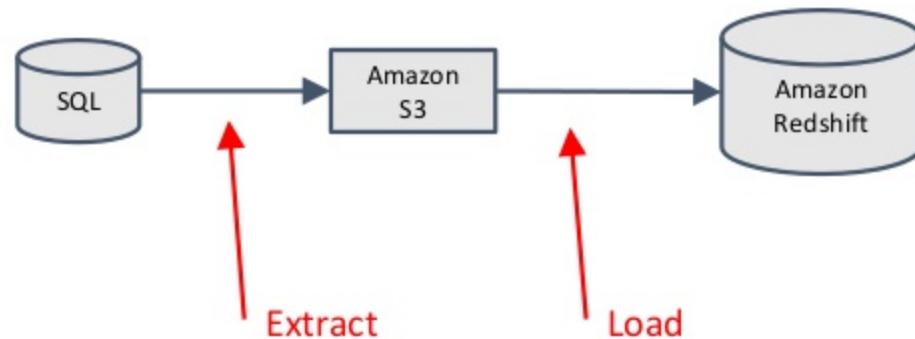
- **type:** extract-from-rds  
**sql:** | SELECT instructor\_id, course\_id, rank  
FROM courses\_instructorincourse;  
**hostname:** maestro-read-replica  
**database:** maestro
- **type:** load-into-staging-table  
**table:** staging.maestro\_instructors\_sessions
- **type:** reload-prod-table  
**source:** staging.maestro\_instructors\_sessions  
**destination:** prod.instructors\_sessions

Extract data from  
Amazon RDS

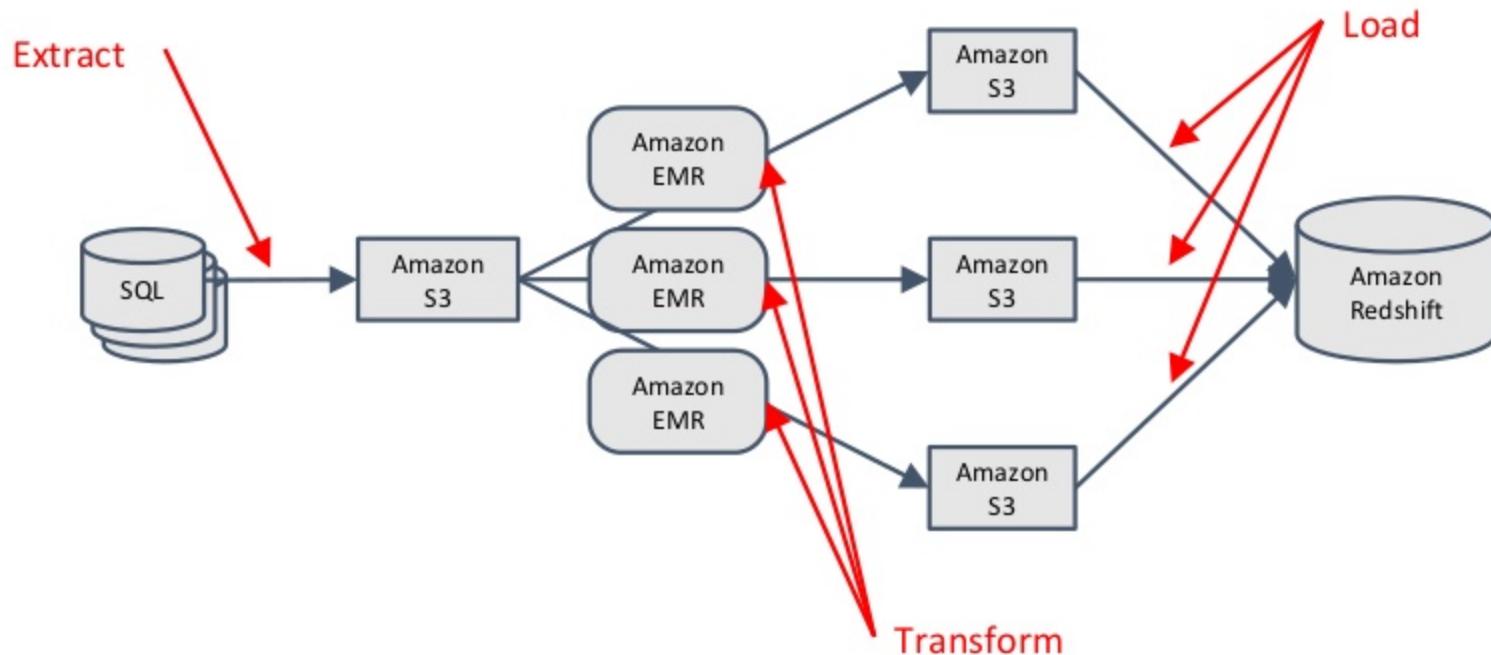
Load intermediate  
table in Amazon  
Redshift

Reload target table  
with new data

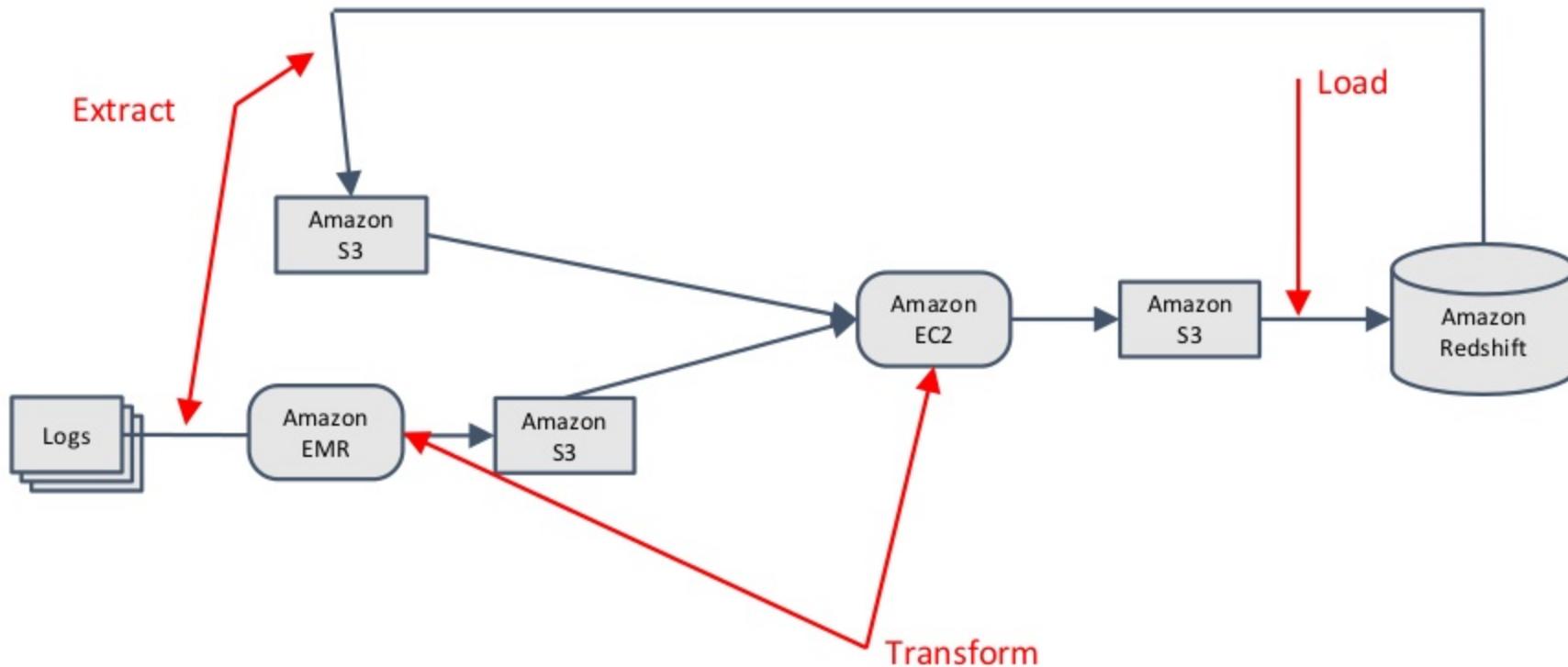
# ETL – Amazon RDS



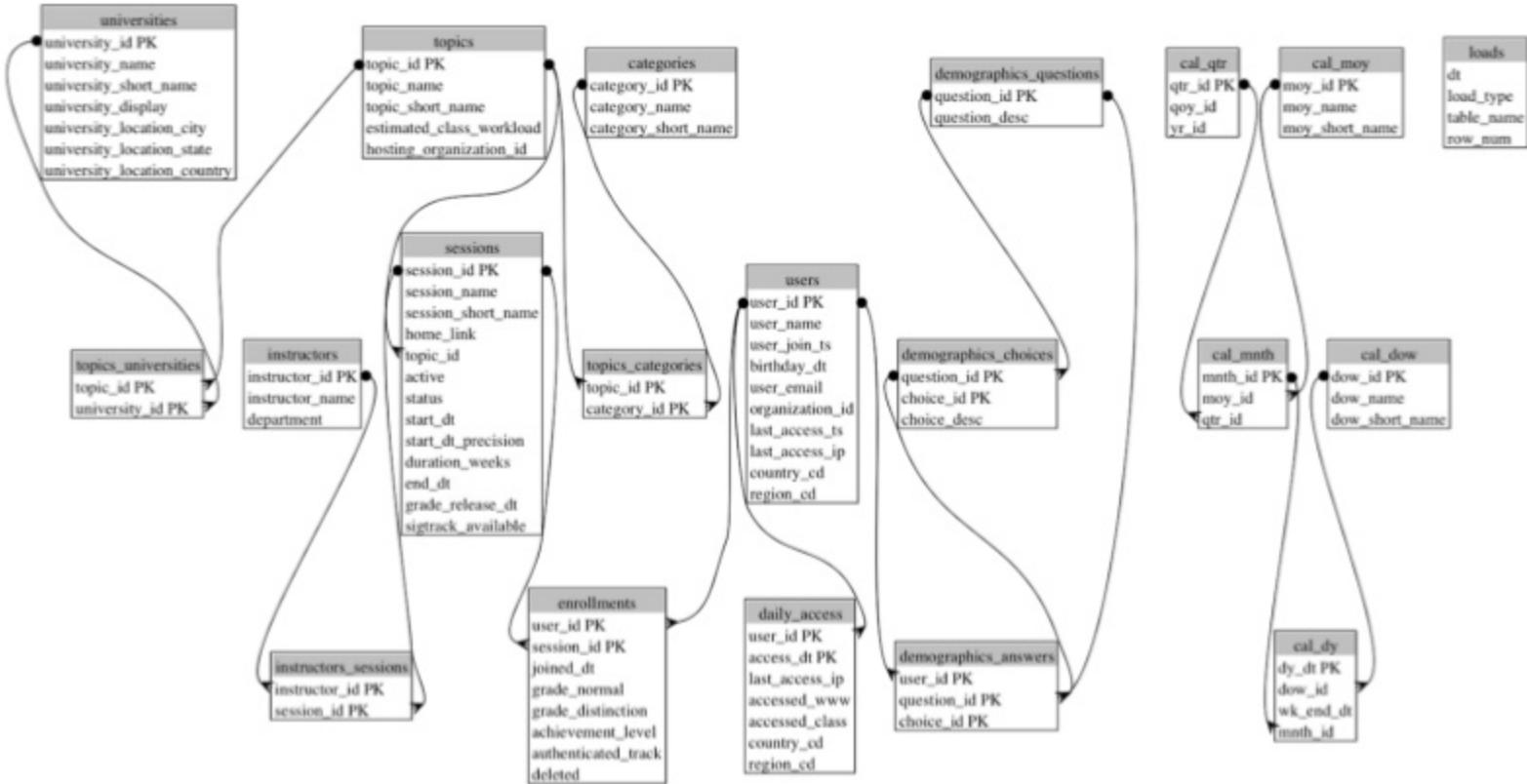
# ETL – Sharded RDS



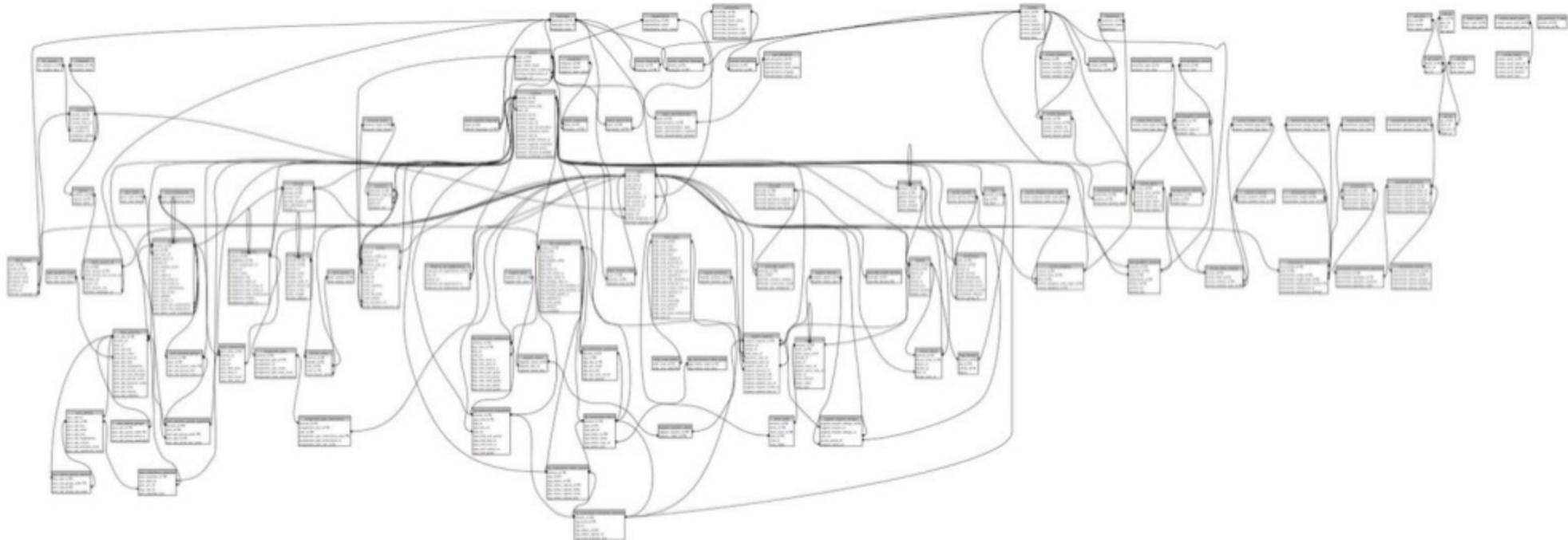
# ETL – Logs



# Reporting Model, Dec 2013



# Reporting Model, Sep 2014



# AWS Data Pipeline

- Easily handles starting/stopping of resources
- Handles permissions, roles
- Integrates with other AWS services
- Handles “flow” of data, data dependencies

# Dealing with large pipelines

- Monolithic pipelines hard to maintain
- Moved to making pipelines smaller
  - Hooray modularity!
- If pipeline B depended on pipeline A, just schedule it later
  - Add a time buffer just to be safe

# Setting cross-pipeline dependencies

- Dependencies accomplished using a script that would wait until dependencies finished
  - **ShellCommandActivity** to the rescue

# The beauty of ShellCommandActivity

- You can use it anywhere
  - Accomplish tasks that have no corresponding activity type
  - Override native AWS Data Pipeline support if it does not meet your needs

# ETL library

- Install on machine as first step of each pipeline
  - With **ShellCommandActivity**
- Allows for even more modularity

## Phase 2: Get users hooked

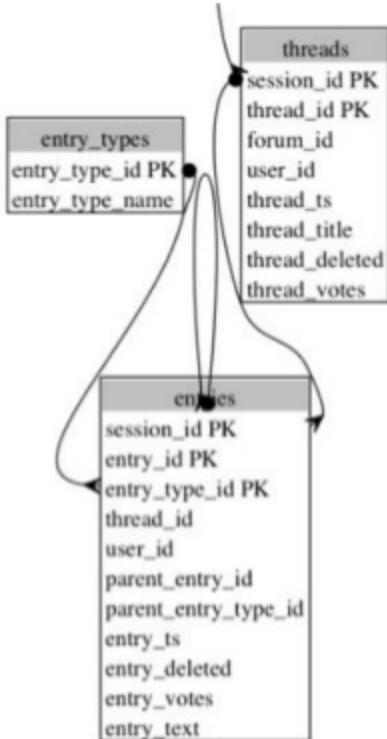
# We have data. Now what?

- Simply collecting data will not make a company data-driven
- First step: make data easier for the analysts to use

# Certifying a version of the truth

- Data Infrastructure team creates a 3NF model of the data
- Model is designed to be as interpretable as possible

# Example: Forums



- Full-word names for database objects
- Join keys made obvious through universally-unique column names
  - e.g. `session_id` joins to `session_id`
- Auto-generated ER-diagram, so documentation is up-to-date

# Some power users need data faster!

- Added new schemas for developers to load data
- Help us to remain data driven during early phases of...
  - Product release
  - Analyzing new data

# Lightweight SQL interface

- Access via browser
  - Users don't need database credentials or special software
- Data exportable as CSV
- Auto-complete for DB object names
- Took ~2 days to make

# Lightweight SQL interface

### SQL Query

```
1 select 1;
```

**Run Query** ▾

### Output

Fetched 1 row(s) in 0.237943 seconds.

Save query as:  Save Query Export to CSV

?column?

1

### Saved Queries

Search by the name assigned to the query, by the author of the query, or both. Leave author blank to search all users' saved queries.

Name:

Author: p.t.barthelemy

### Example Queries

- completions for university
- course details for university
- end of course stories
- end of course stories for signature track
- forum contents for session
- private classes for university
- sigtrack enrollments for university

# But what about people who don't know SQL?

- Reporting tools to the rescue!
- In-house dashboard framework to support:
  - Support
  - Growth
  - Data Infrastructure
  - Many other teams...
- Tableau for interactive analysis

# But what about people who don't know SQL?

How is this metric calculated?



# Phase 3: Increase reliability

# If you build it, they will come

- Built ecosystem of data consumers
  - Analysts
  - Business users
  - Product team

# Maintaining the ecosystem

- As we accumulate consumers, need to invest effort in...
  - Model stability
  - Data quality
  - Data availability

# Persist Amazon Redshift logs to keep track of usage

- Leverage database credentials to understand usage patterns
- Know whom to contact when
  - Need to change the schema
  - Need to add a new set of tables similar to an existing set (e.g. assessments)

# Data Quality

- Quality of source systems (GIGO)
  - Encourage source system to fix issues
- Quality of transformations
  - Responsibility of the data infrastructure team

# Quality Transformations

- Automated QA checks as part of ETL
  - Again, use **ShellCommandActivity**
- Key factors
  - Counts
  - Value comparisons
  - Modeling rules (primary keys)

# Ensuring High Reliability

- Step retries catch most minor hiccups
- On-call system to recover failed pipelines
- Persist DB logs to keep track of load times
  - Delayed loads can also alert on-call devs
  - Bonus: users know how fresh the data is
  - Bonus: can keep track of success rate
- AWS very helpful in debugging issues

# Looking Forward

# Current bottlenecks

- Complex ETL
  - How do we easily ETL complex, nested structures?
- Developer bottleneck
  - How do we allow users to access raw data more quickly?

# Amazon S3 as principal data store

- Amazon S3 to store raw data as soon as it's produced (data lake)
- Amazon EMR to process data
- Amazon Redshift to remain as analytic platform

Thank you!

## Want to learn more?

- [github.com/coursera/dataduct](https://github.com/coursera/dataduct)
- <http://tech.coursera.org>
- [thomas@coursera.org](mailto:thomas@coursera.org)

# Call To Action

- Use AWS Data Pipeline API
- Use AWS Data Pipeline for testing/QE
- Open source integration
- Reusable artifacts
- Amazon RDS, Amazon DynamoDB,  
Amazon S3, Amazon Redshift,  
Amazon EMR

# Call To Action (2)

- AWS Big Data Blog
  - <http://blogs.aws.amazon.com/bigdata/>
- AWS Data Pipeline documentation
  - <http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-concepts-schedules.html>
- AWS re:Invent Big Data Booth
- AWS Professional Service/SA/Support



Please give us your feedback on this session.  
Complete session evaluations and earn re:Invent swag.

BDT303

<http://bit.ly/awsevals>



Join the conversation on Twitter with **#reinvent**