SPARK+AI
SUMMIT 2018

Cloud Cost Management and Apache Spark

Xuan Wang, Databricks

#DSSAIS13

Introduction

- Goal of this talk
  - share our experience in managing cloud costs
  - tools and technologies
  - lessons learnt and good practices
  - go wide rather than go deep

Introduction

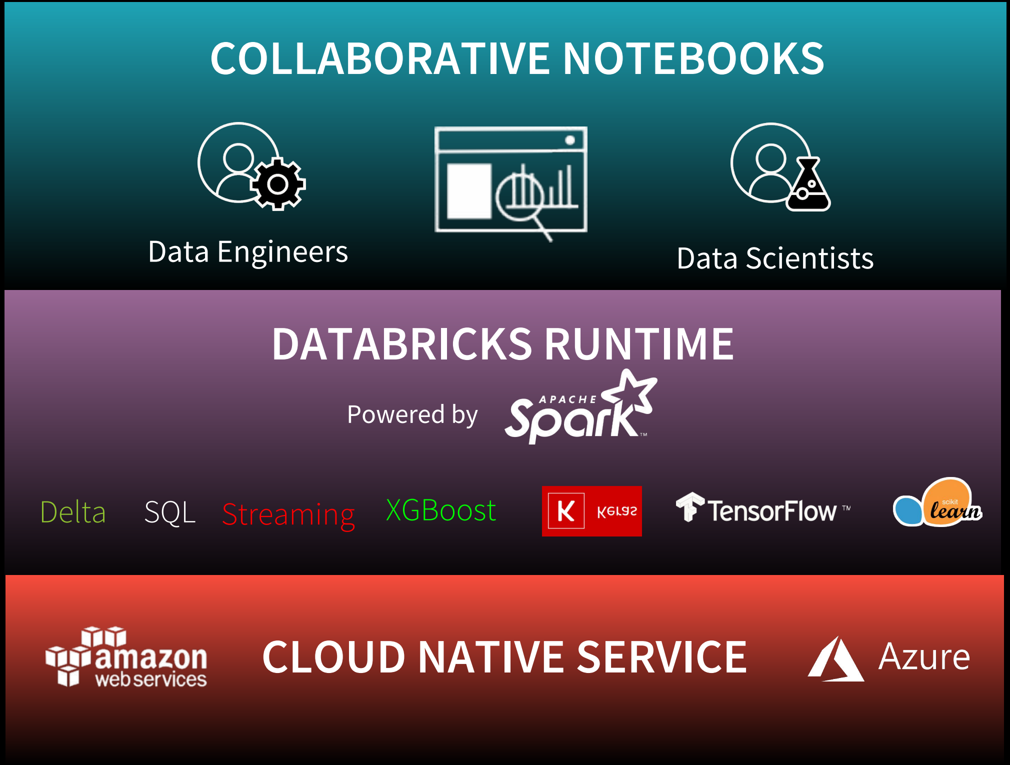
- Goal of this talk
  - share our experience in managing cloud costs
  - tools and technologies
  - lessons learnt and good practices
  - go wide rather than go deep
- Why do we care about cloud cost?
  - growth in cloud revenue in Q1 2018: Amazon: 49%, Microsoft: 58%

Databricks' Unified Analytics Platform

Unifies Data Engineers and Data Scientists

Unifies Data and AI Technologies

Eliminates infrastructure complexity



COLLABORATIVE NOTEBOOKS

Data Engineers

Data Scientists

DATABRICKS RUNTIME

Powered by APACHE Spark

Delta SQL Streaming XGBoost Keras TensorFlow scikit learn

amazon web services CLOUD NATIVE SERVICE Azure

Three paths toward cost control

- Native reporting from cloud providers
  - Good general information and supports
  - Limited options, not scalable as environment grows
- Commercial tools
  - More details and flexibilities, connectors to raw data
  - Not enough customization, additional charges

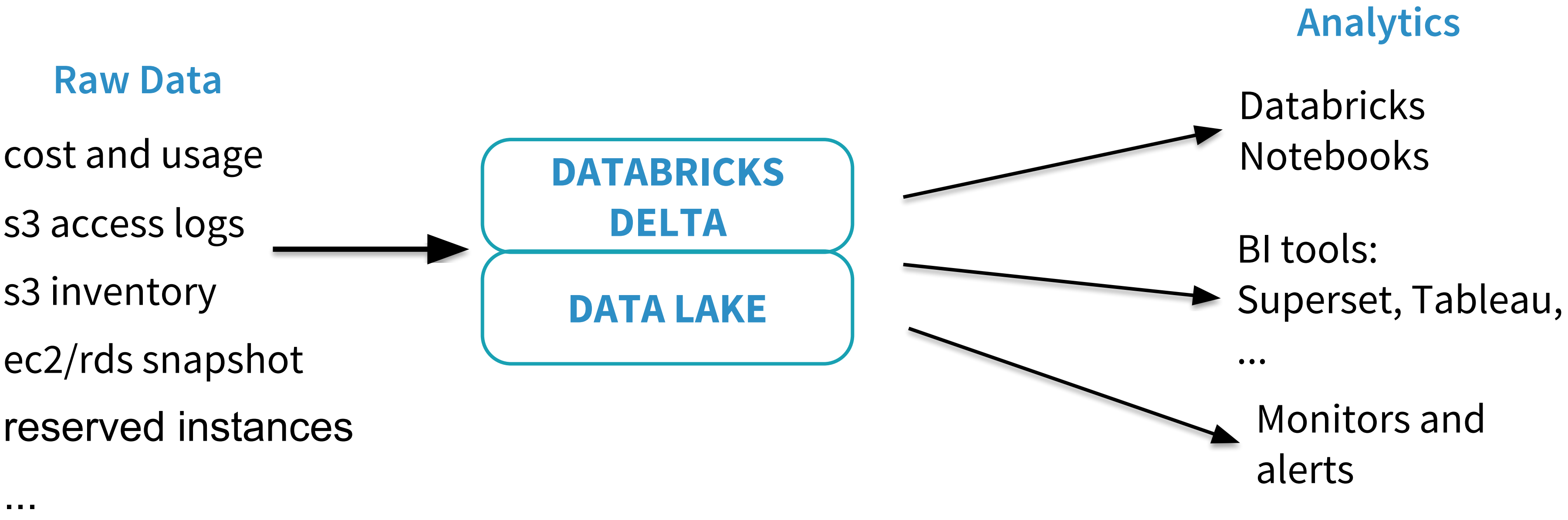
Three paths toward cost control

- Native reporting from cloud providers
  - Good general information and supports
  - Limited options, not scalable as environment grows
- Commercial tools
  - More details and flexibilities, connectors to raw data
  - Not enough customization, additional charges
- In-house solutions
  - Most flexible, deeper understanding of the costs
  - Opportunity costs

Challenges in cloud cost control

- overwhelming and complex usage details
  - need to convert data into insights/actions
- gaps between “hands” and “wallets”
  - developers consume resources without realizing the charges
- evolving cloud landscape
  - external: new services, new discounts, ...
  - internal: new use cases, new architecture, ...

Our solutions



Raw Data

cost and usage

s3 access logs

s3 inventory

ec2/rds snapshot

reserved instances

...

DATABRICKS DELTA

DATA LAKE

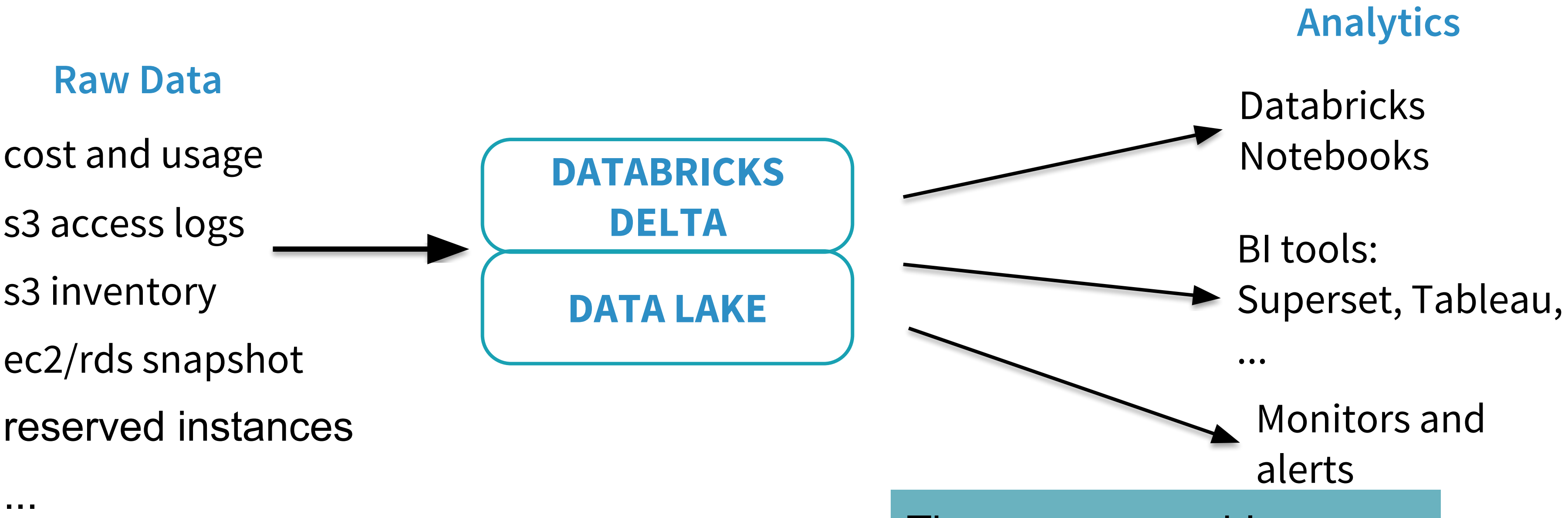
Analytics

Databricks Notebooks

BI tools: Superset, Tableau, ...

Monitors and alerts

Our solutions



Raw Data

cost and usage

s3 access logs

s3 inventory

ec2/rds snapshot

reserved instances

...

DATABRICKS DELTA

DATA LAKE

Analytics

Databricks Notebooks

BI tools: Superset, Tableau, ...

Monitors and alerts

The **data** problem:
ETL and attribute costs

The **process** problem:
prioritize, optimize, monitor, automate

The data problem

- cost and usage report (detailed billing)
    - CSV, grouped by month, updated daily

The data problem

- cost and usage report (detailed billing)
  - CSV, grouped by month, updated daily
- EC2/RDS snapshots and reserved instances
  - JSON, from REST API

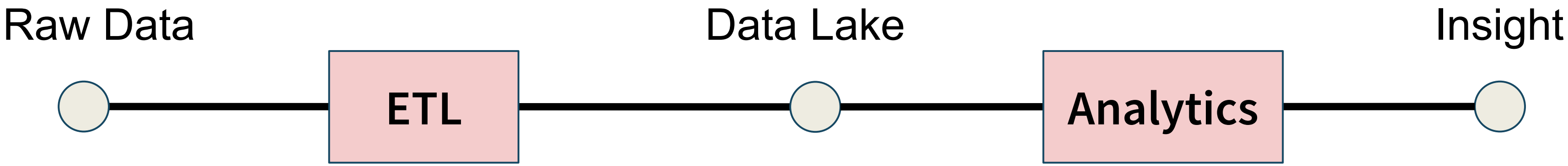
The data problem

- cost and usage report (detailed billing)
  - CSV, grouped by month, updated daily
- EC2/RDS snapshots and reserved instances
  - JSON, from REST API
- S3 inventory
  - CSV/ORC, snapshot, updated daily/weekly
- S3 access logs
  - raw logs in text, updated multiple times a day

The data problem

- cost and usage report (detailed billing)
    - CSV, grouped by month, updated daily
- EC2/RDS snapshots and reserved instances
    - JSON, from REST API
- S3 inventory
    - CSV/ORC, snapshot, updated daily/weekly
- S3 access logs
    - raw logs in text, updated multiple times a day

Data pipelines with Spark



Challenges

- Data corruptions
- Multiple jobs/staging tables
- Reliability and consistency

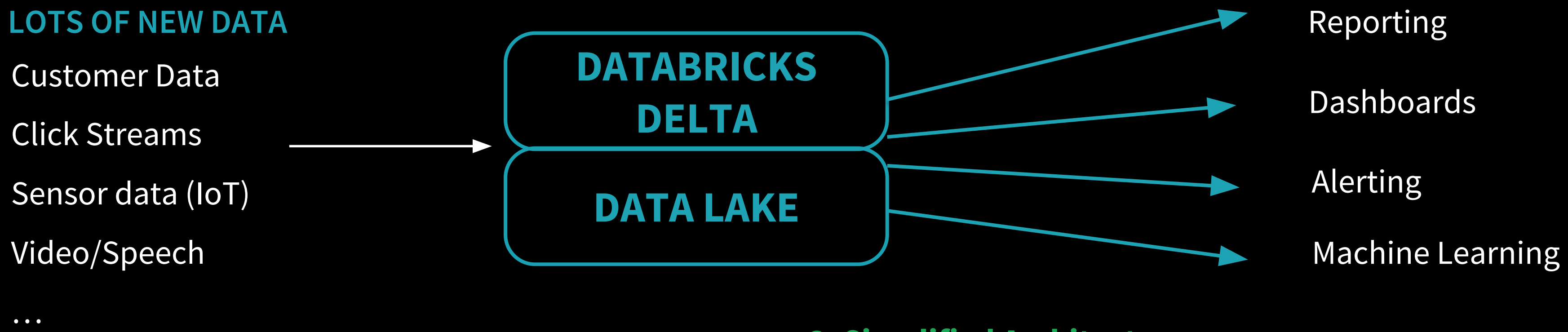
Databricks Delta: Analytics Ready Data

1. Data Reliability

ACID Compliant Transactions
Schema Enforcement & Evolution

2. Query Performance

Very Fast at Scale
Indexing & Caching (10-100x Faster)



LOTS OF NEW DATA

Customer Data

Click Streams

Sensor data (IoT)

Video/Speech

…

DATABRICKS DELTA

DATA LAKE

Reporting

Dashboards

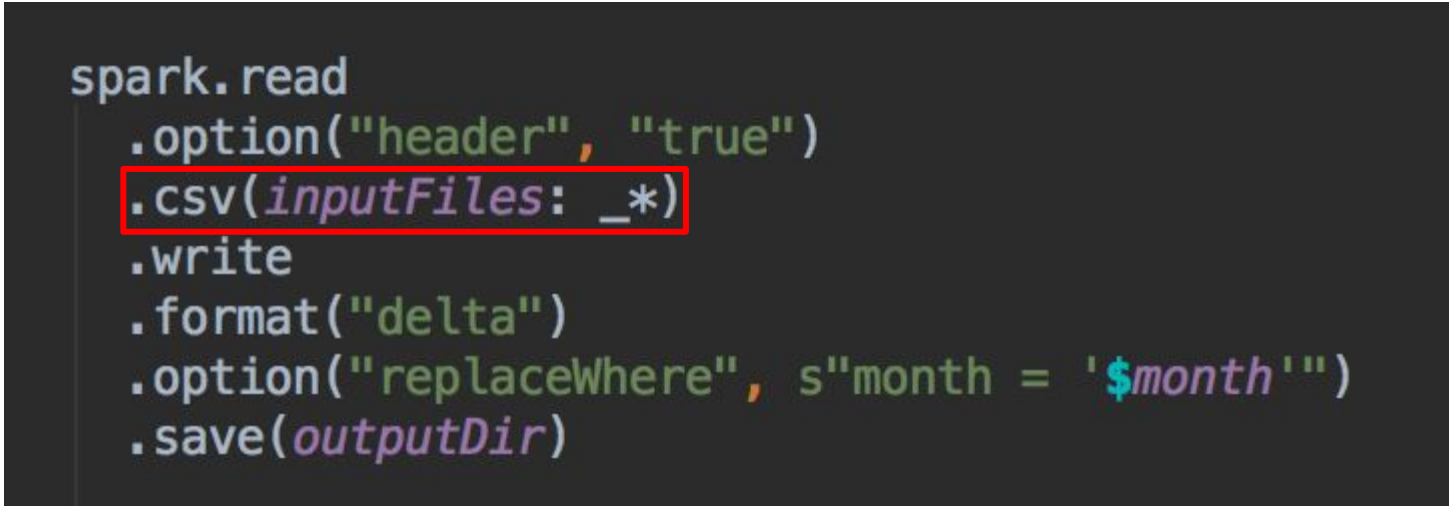
Alerting

Machine Learning

3. Simplified Architecture

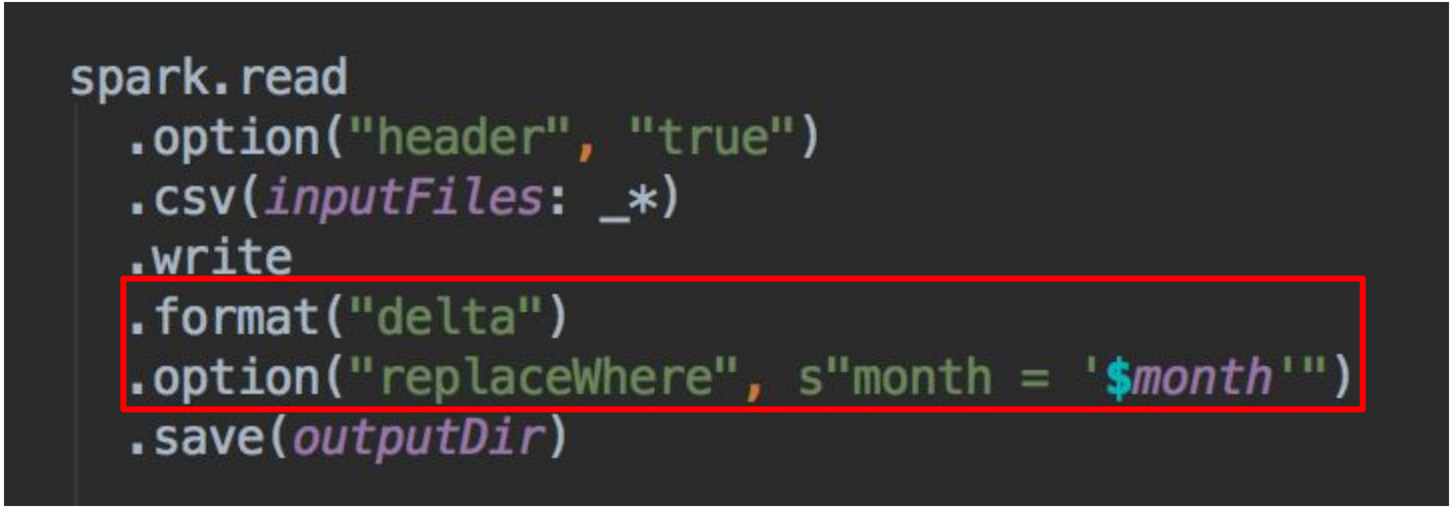
Unify batch & streaming
Early data availability for analytics

ETL: AWS cost and usage



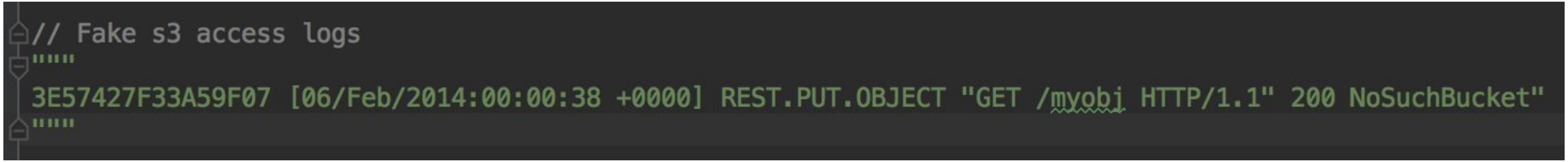
```
spark.read
  .option("header", "true")
  .csv(inputFiles: _*)
  .write
  .format("delta")
  .option("replaceWhere", s"month = '$month'")
  .save(outputDir)
```

ETL: AWS cost and usage

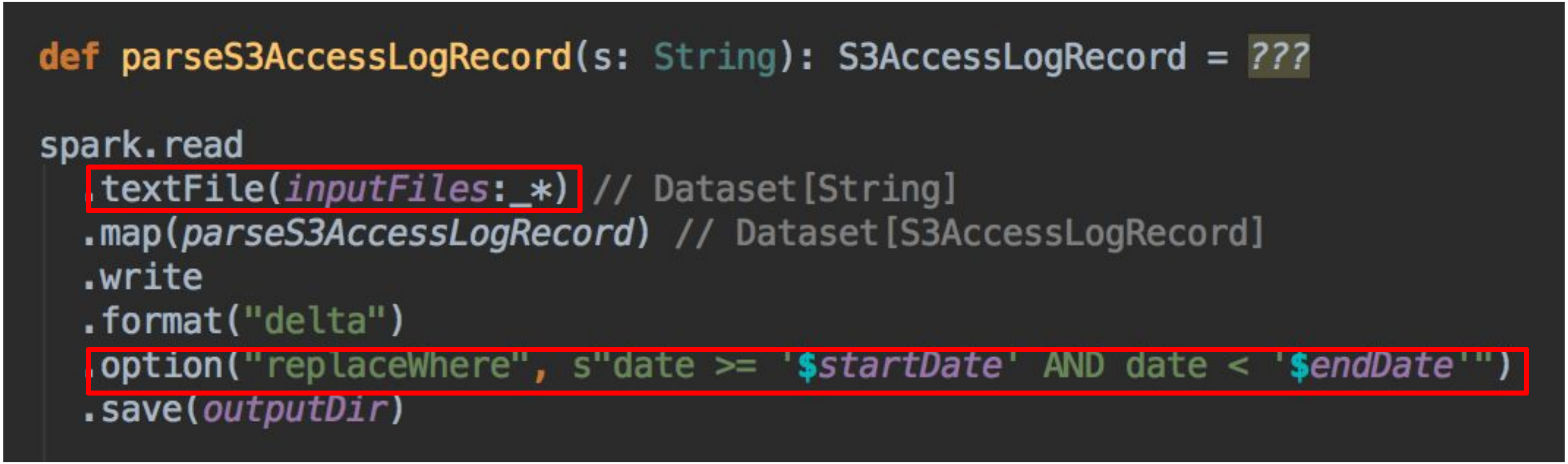


```
spark.read
  .option("header", "true")
  .csv(inputFiles: _*)
  .write
  .format("delta")
  .option("replaceWhere", s"month = '$month'")
  .save(outputDir)
```

ETL: AWS s3 access logs



```
// Fake s3 access logs
"""
3E57427F33A59F07 [06/Feb/2014:00:00:38 +0000] REST.PUT.OBJECT "GET /myobj HTTP/1.1" 200 NoSuchBucket"
"""
```



```
def parseS3AccessLogRecord(s: String): S3AccessLogRecord = ???

spark.read
  .textFile(inputFiles:_*) // Dataset[String]
  .map(parseS3AccessLogRecord) // Dataset[S3AccessLogRecord]
  .write
  .format("delta")
  .option("replaceWhere", s"date >= '$startDate' AND date < '$endDate'")
  .save(outputDir)
```

Manage Databricks Delta tables

- Create table

```
CREATE TABLE s3_access_logs USING delta LOCATION '$path'
```

- Optimize table

```
OPTIMIZE s3_access_logs ZORDER BY bucket
```

Manage Databricks Delta tables

- Create table

```
CREATE TABLE s3_access_logs USING delta LOCATION '$path'
```

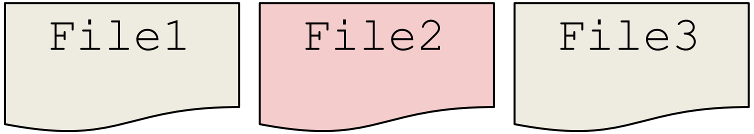
- Optimize table

```
OPTIMIZE s3_access_logs ZORDER BY bucket
```

- Query table

```
SELECT * FROM s3_access_logs WHERE bucket = 'my-bucket'
```

Files layout & statistics:



```
Delta Logs:
File1: min='a', max='g'
File2: min='g', max='n'
File3: min='o', max='z'
```

Attributions

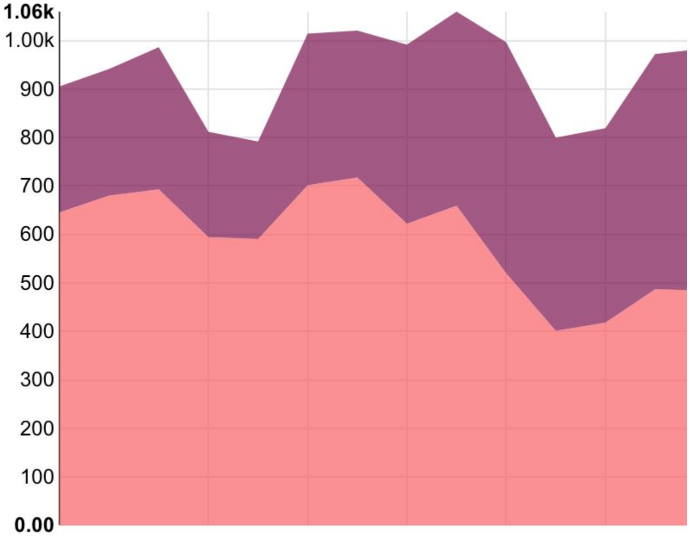
- Rule based attributions
    - accounts
        - dedicated accounts for different teams / use cases
    - tagging
        - tag resources with budget groups
    - manual rules
        - should avoid this as much as possible

The process problem

- Prioritize
  - high data transfer cost
- Optimize
  - reserved instance purchases
- Monitor
  - predictions and alerts
- Automate
  - auto-shutdown unused resources

Story: high data transfer cost

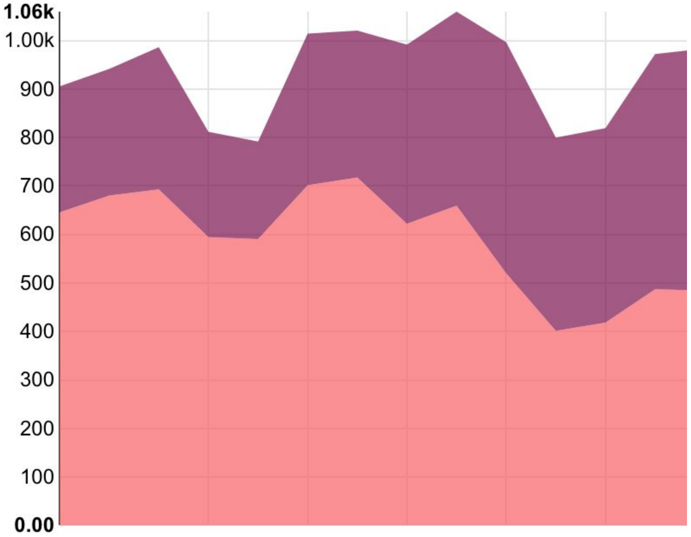
- Observation
  - Cross region data transfers are expensive
  - Two buckets cost about $1k/day



Story: high data transfer cost

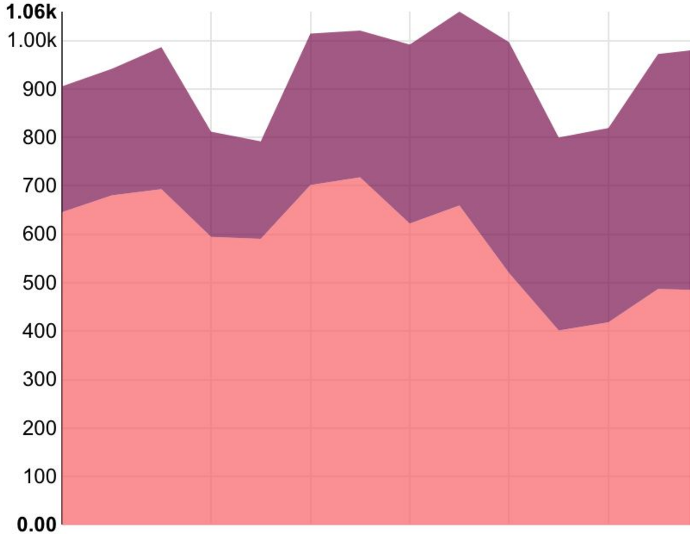
- Observation
  - Cross region data transfers are expensive
  - Two buckets cost about $1k/day

- Root cause
  - downloading spark images



Story: high data transfer cost

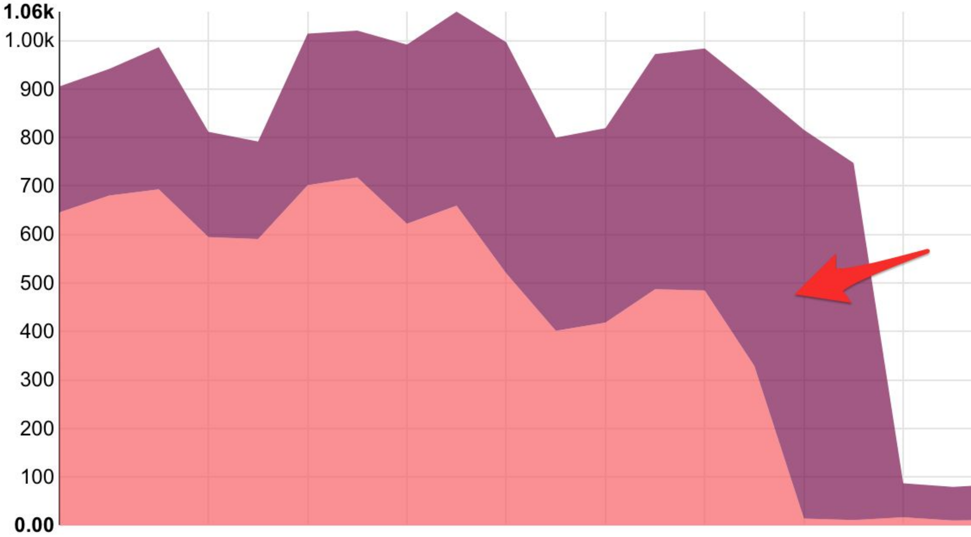
- Actions
  - Distribute images to multiple regions.
  - Monitor on cross region cost



Story: high data transfer cost

- Actions
  - Distribute images to multiple regions.
  - Monitor on cross region cost

- Results
  - Significantly reduced cost
  - Faster cluster creation



Optimization: reserved instances

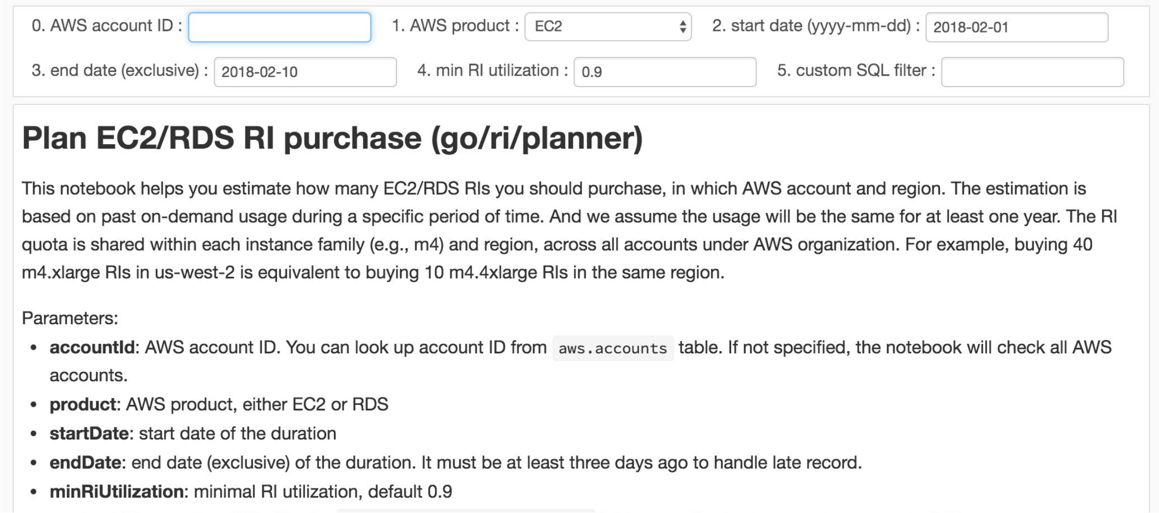
- Reserved instances (RI)
  - 1-yr/3-yr commitment in exchange for discounts
  - underutilized instances, upfront cost
  - significant discounts, availability

Optimization: reserved instances

- Reserved instances (RI)
  - 1-yr/3-yr commitment in exchange for discounts
  - underutilized instances, upfront cost
  - significant discounts, availability
- Challenges
  - non-trivial to decide how much RI to purchase
  - need to predict the future

Optimization: reserved instances

- Assign budgets to teams
- Provide tool to compute the optimal RI to buy
- Define process for RI purchase requests and approvals



0. AWS account ID :	1. AWS product : EC2	2. start date (yyyy-mm-dd) : 2018-02-01
3. end date (exclusive) : 2018-02-10	4. min RI utilization : 0.9	5. custom SQL filter :

Plan EC2/RDS RI purchase (go/ri/planner)

This notebook helps you estimate how many EC2/RDS RIs you should purchase, in which AWS account and region. The estimation is based on past on-demand usage during a specific period of time. And we assume the usage will be the same for at least one year. The RI quota is shared within each instance family (e.g., m4) and region, across all accounts under AWS organization. For example, buying 40 m4.xlarge RIs in us-west-2 is equivalent to buying 10 m4.4xlarge RIs in the same region.

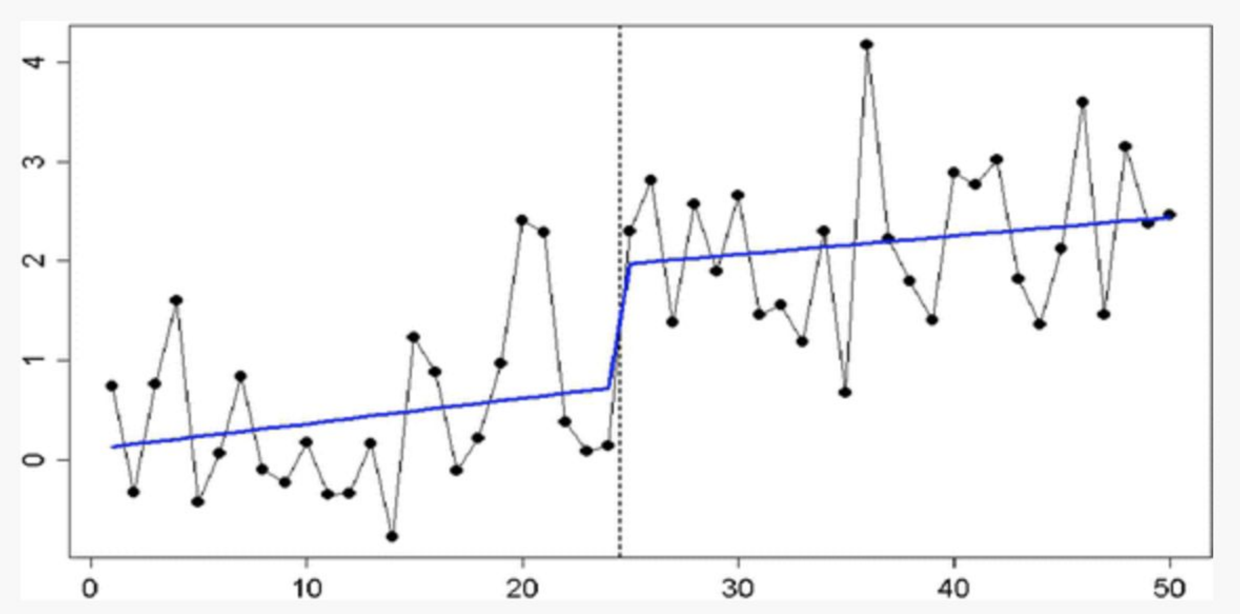
Parameters:

- **accountId**: AWS account ID. You can look up account ID from `aws.accounts` table. If not specified, the notebook will check all AWS accounts.
- **product**: AWS product, either EC2 or RDS
- **startDate**: start date of the duration
- **endDate**: end date (exclusive) of the duration. It must be at least three days ago to handle late record.
- **minRiUtilization**: minimal RI utilization, default 0.9

Monitor and alerts

- Why
  - prevent degenerations
  - proactive to “bill shock”
- Challenges
  - different patterns for different use cases
  - changing baselines

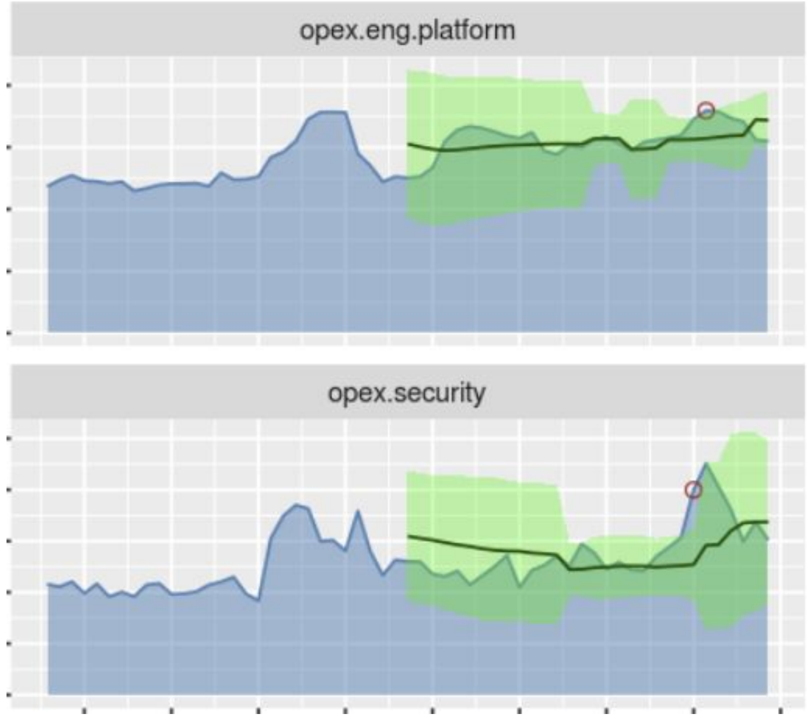
Monitor and alerts



Picture from Sharma, S., Swayne, D.A. & Obimbo, C. Energ. Ecol. Environ. (2016) 1: 123. https://doi.org/10.1007/s40974-016-0011-1

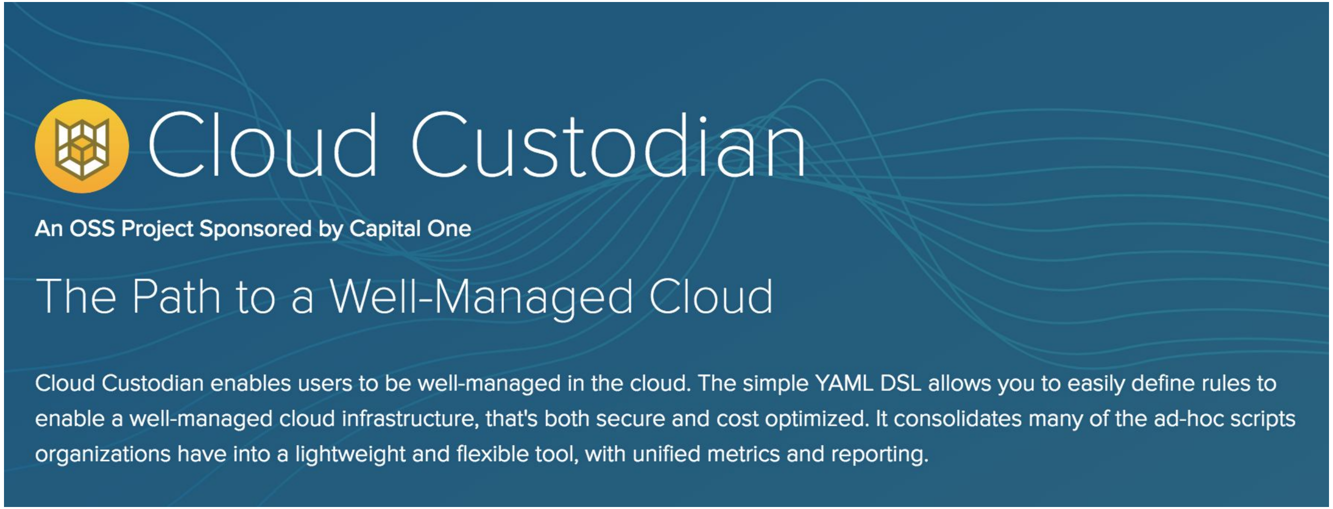
Monitor and alerts

- adaptive prediction with change point detection
- alerts for each budget group
- scheduled jobs and dashboards



Auto-shutdown with Custodian

- https://github.com/capitalone/cloud-custodian
  - Rule based cloud infrastructure management tool



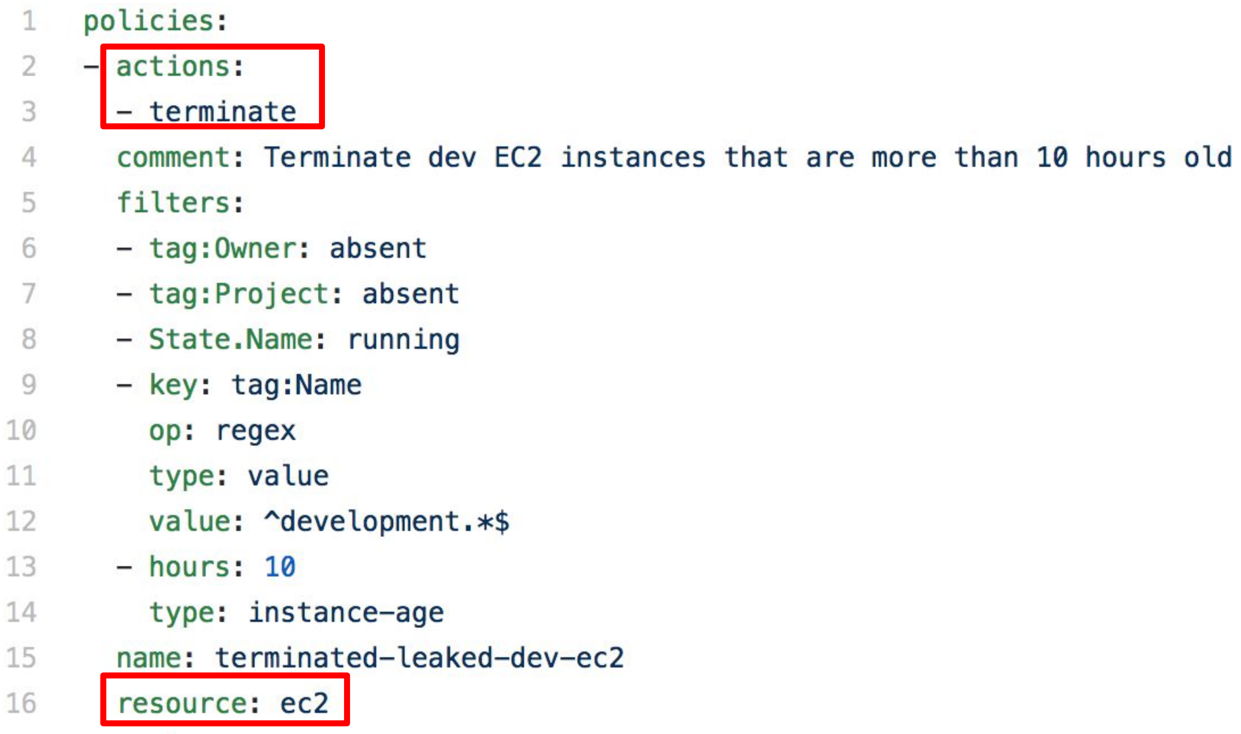
Cloud Custodian

An OSS Project Sponsored by Capital One

The Path to a Well-Managed Cloud

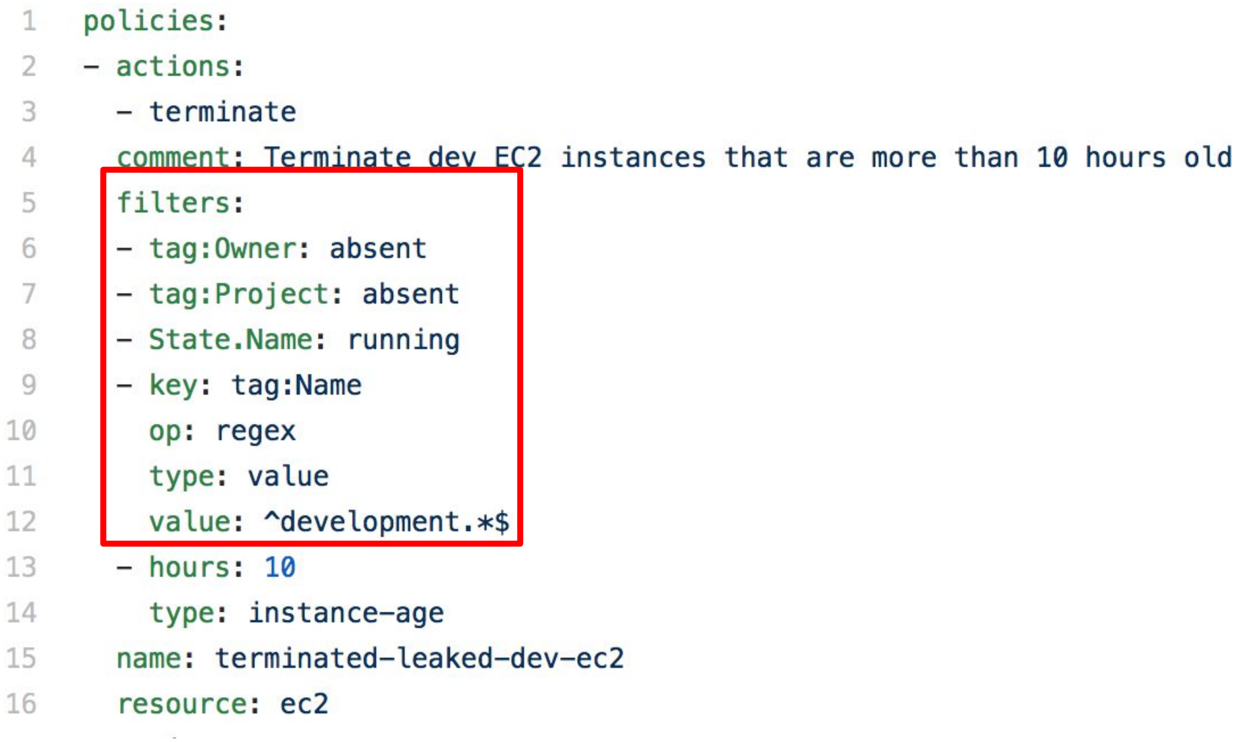
Cloud Custodian enables users to be well-managed in the cloud. The simple YAML DSL allows you to easily define rules to enable a well-managed cloud infrastructure, that's both secure and cost optimized. It consolidates many of the ad-hoc scripts organizations have into a lightweight and flexible tool, with unified metrics and reporting.

Auto-shutdown with Custodian



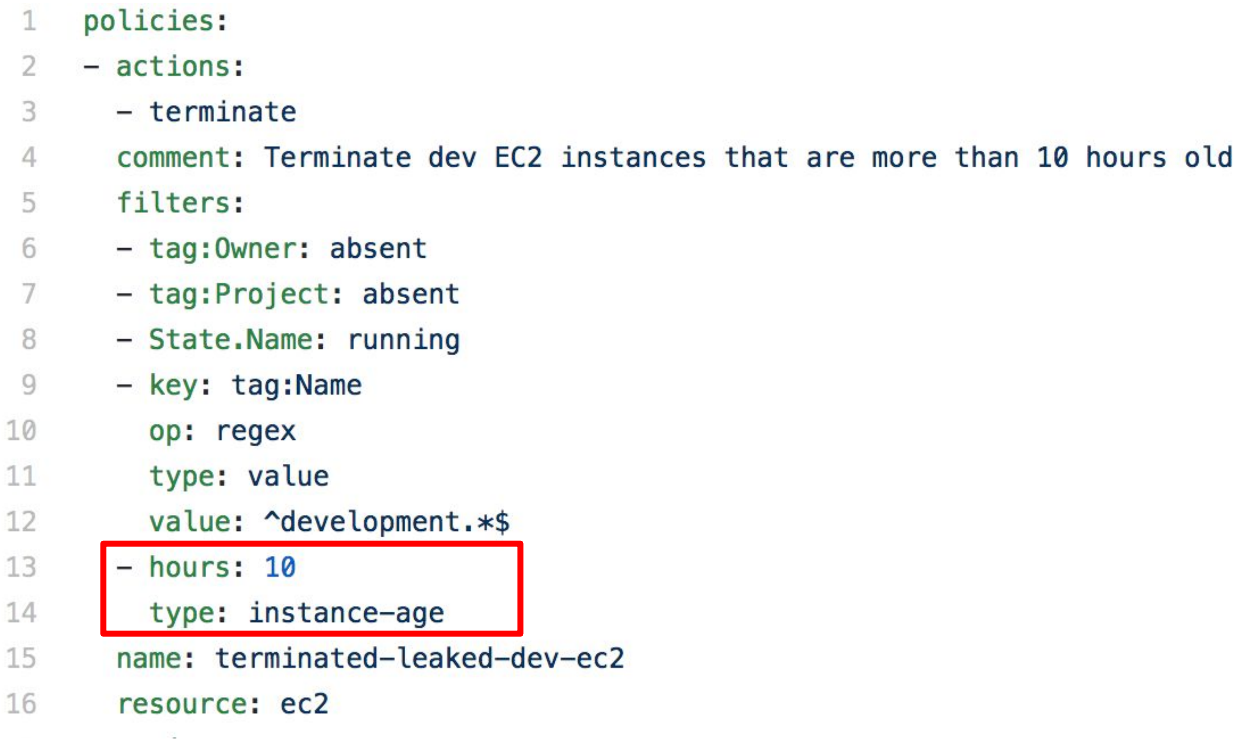
```yaml
policies:
- actions:
  - terminate
  comment: Terminate dev EC2 instances that are more than 10 hours old
  filters:
  - tag:Owner: absent
  - tag:Project: absent
  - State.Name: running
  - key: tag:Name
    op: regex
    type: value
    value: ^development.*$
  - hours: 10
    type: instance-age
  name: terminated-leaked-dev-ec2
  resource: ec2
```

Auto-shutdown with Custodian



```yaml
policies:
- actions:
  - terminate
  comment: Terminate dev EC2 instances that are more than 10 hours old
  filters:
  - tag:Owner: absent
  - tag:Project: absent
  - State.Name: running
  - key: tag:Name
    op: regex
    type: value
    value: ^development.*$
  - hours: 10
    type: instance-age
  name: terminated-leaked-dev-ec2
  resource: ec2
```

Auto-shutdown with Custodian



```yaml
policies:
- actions:
  - terminate
  comment: Terminate dev EC2 instances that are more than 10 hours old
  filters:
  - tag:Owner: absent
  - tag:Project: absent
  - State.Name: running
  - key: tag:Name
    op: regex
    type: value
    value: ^development.*$
  - hours: 10
    type: instance-age
  name: terminated-leaked-dev-ec2
  resource: ec2
```

Summary

- in-house solutions because
  - flexibility and deeper understanding of cost and usage
- cost attribution - a **data** problem
  - ETL: Databricks Delta for analytics ready data
  - explore: Databricks Notebooks and BI tools via JDBC
  - attribute: rule-based, tagging is important
- cost control - a **process** problem
  - prioritize: get the work done!
  - optimize: distributed ownerships, and centralized tools
  - monitor: change points + basic linear model
  - automate: Custodian for managing cloud infrastructure

Thank you

xwang@databricks.com

https://www.linkedin.com/in/xuanwang2

https://databricks.com/product/unified-analytics-platform

SPARK+AI
SUMMIT 2018

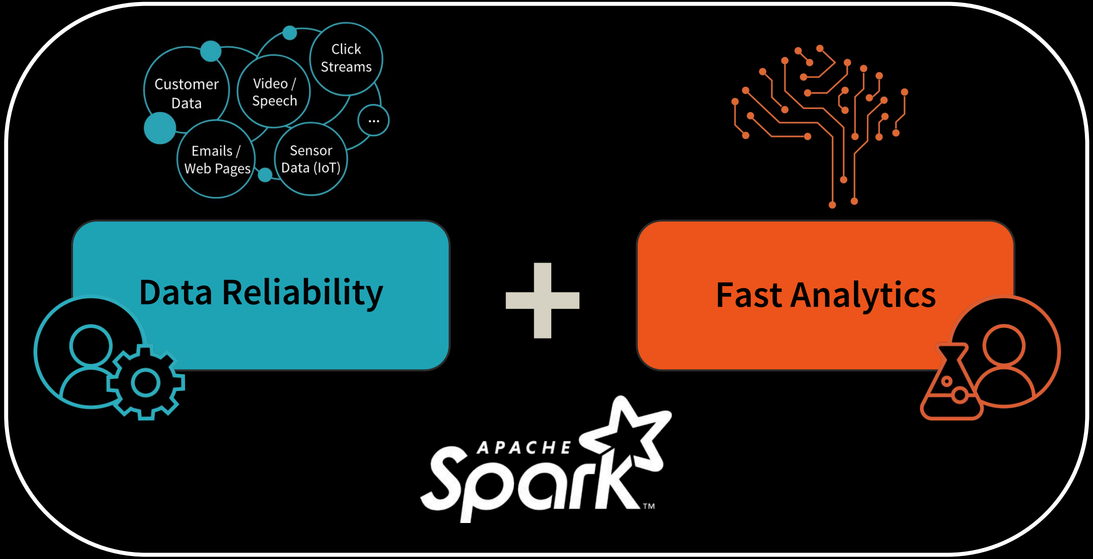
BACKUP SLIDES

Xuan Wang, Databricks

#AssignedHashtagGoesHere

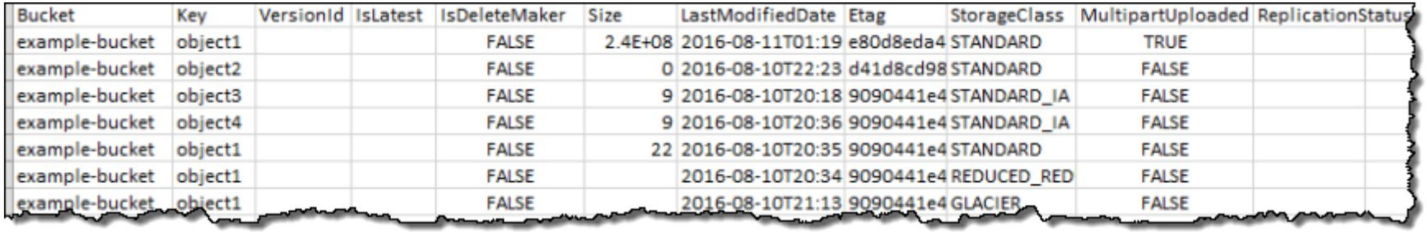
New: Databricks Delta

*Extends Apache Spark to simplify data **reliability** and **performance***



The data problem

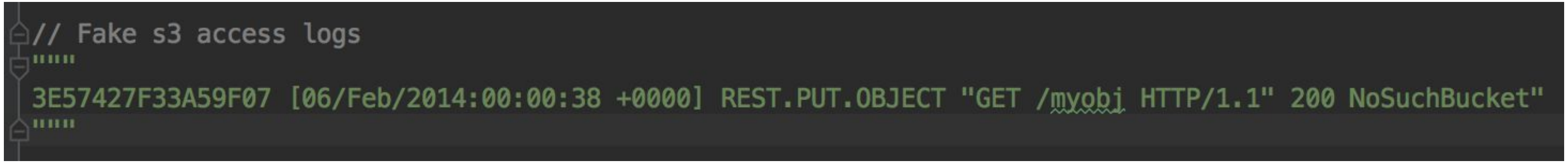
- cost and usage report
  - detailed usage and billing information by hours
  - CSV, delivered at least once a day
- s3 inventory
  - list of all objects in s3 and associated metadata
  - CSV/ORC, delivered daily/weekly



Bucket	Key	VersionId	IsLatest	IsDeleteMaker	Size	LastModifiedDate	Etag	StorageClass	MultipartUploaded	ReplicationStatus
example-bucket	object1			FALSE	2.4E+08	2016-08-11T01:19	e80d8eda4	STANDARD	TRUE	
example-bucket	object2			FALSE	0	2016-08-10T22:23	d41d8cd98	STANDARD	FALSE	
example-bucket	object3			FALSE	9	2016-08-10T20:18	9090441e4	STANDARD_IA	FALSE	
example-bucket	object4			FALSE	9	2016-08-10T20:36	9090441e4	STANDARD_IA	FALSE	
example-bucket	object1			FALSE	22	2016-08-10T20:35	9090441e4	STANDARD	FALSE	
example-bucket	object1			FALSE		2016-08-10T20:34	9090441e4	REDUCED_RED	FALSE	
example-bucket	object1			FALSE		2016-08-10T21:13	9090441e4	GLACIER	FALSE	

The data problem

- s3 access logs
  - requests/API calls to access s3 objects
  - raw logs, delivered frequently



```
// Fake s3 access logs
"""
3E57427F33A59F07 [06/Feb/2014:00:00:38 +0000] REST.PUT.OBJECT "GET /myobj HTTP/1.1" 200 NoSuchBucket"
"""
```

- EC2/RDS snapshots and reserved instances
  - json from REST API