Herding Elephants: Seamless data access in multi-cluster clouds
Speakers

Pradeep Bhadani
Senior Big Data Engineer at Expedia Group (Hotels.com)
Agenda

• Introduction
  • Data platform cloud migration
  • Multi data lake challenges
• High level solutions
• Technical deep dive
• Current limitations
• Summary
• Questions
On premises Hadoop

- Used by multiple brands
- Management overhead
- Inefficient resource allocation
- Infrequently upgraded
  - Many dependencies
Cloud migration

- Move data processing to AWS
- Brands can do this independently
- Focus on speed and agility
Migrating a data lake

• Data
  • Hadoop compatible file system
  • File store: HDFS, S3, etc.

• Metadata
  • Hive metastore
  • File locations, schema, etc.

• Processes
  • On premise cluster → EMR
  • Focus on new development - migration longer-term
Challenge: Data Silos
Option 1: Consolidate accounts/infrastructure?
Option 2: Replicate between accounts?
Option 3: Federation (Autonomy + Collaboration)?
Waggle Dance

• System for federating Apache Hive Metastores
• Compose multiple independent data lakes
• Provides users with a single ‘virtual data lake’
• Open source:
  • [https://github.com/HotelsDotCom/waggle-dance](https://github.com/HotelsDotCom/waggle-dance)
Waggle Dance: Overview

DATA LAKE FEDERATION

“Primary” Metastore
Federated Metastore(s)

Waggle Dance

Thrift API

Workload

S3

Account A

Account B

Meta Store

Meta Store

Expedia Group Proprietary and Confidential
Metadata silos: User perspective

```
hive> show databases;
```

Account A
- `etl`

Account B
- `ml`
Metadata silos: User perspective

**Account A**

```sql
hive> describe formatted
> etl.hotel;
```

<table>
<thead>
<tr>
<th>col_name</th>
<th>data_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
</tr>
</tbody>
</table>

# Location: s3://account-a/etl/hotel/

**Account B**

```sql
hive> describe formatted
> ml.hero_image;
```

<table>
<thead>
<tr>
<th>col_name</th>
<th>data_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int</td>
</tr>
<tr>
<td>hotel_id</td>
<td>int</td>
</tr>
<tr>
<td>img_name</td>
<td>string</td>
</tr>
</tbody>
</table>

# Location: s3://account-b/ml/hero/
Metadata federation across accounts

Account A (with federated Hive Metastores)

```
hive> set hive.metastore.uris =thrift://waggle-dance:48869;
```

```
hive> show databases;
```

```
etl
ml
```
Hive join across multiple accounts

Account A (with federated Hive Metastores)

```sql
hive> select h.id, h.name, i.img_name
    > from etl.hotel h
    > join ml.hero_image i
    > where h.id = i.hotel_id
    > and h.name like "Hilton Angel%";
```

```
2314  Hilton Angel  0a673kZVrt832.png
```

Fetched: 1 row(s)

```sql
hive>
```
Operational challenges

- Disparate data lake implementations and configurations
  - Inconsistent
  - Difficult to integrate with one another
- Organic topology
  - Dataset ownership unclear
  - Mix of federated and direct pathways
Apiary

- Deployable data lake ‘pattern’
  - Infrastructure as code
  - Developed collaboratively
  - All components needed to deploy a data lake and join the federation

- Open source:
  - https://github.com/ExpediaGroup/apiary
Apiary v1

Apiary Features

• Core Data-Lake – Hive Metastore + S3 (external management)
• Managed Schemas – Integrated management of S3 and Hive databases
• Read-Write (RW) and Read-Only (RO) Metastores
• Federation
• AWS PrivateLinks – To enable Federated Access
• Bring your own tool
• Focus on High Availability
• Immutable deployment using Terraform
Core Data Lake in AWS

- Hive Metastore
- SQL Database
- S3 Buckets
Managed Schemas

Integrated management of S3 buckets and Hive databases.

- S3 bucket names are derived and automatically mapped
- Automatic bucket policy management
- Simple inputs – Hive database names, customer accounts
Federation

• Multiple Data lakes across different Expedia Group brands
• Waggle Dance enables connectivity to multiple metadata services (Hive Metastores)
• Terraform
  • Enables the creation of VPC PrivateLinks
  • Automates Waggle Dance configurations
Apiary Federation

Data Lake

Local Workloads

RW Hive Metastore service

DB

RO Hive Metastore service

S3

Cross account sharing
Apiary enables "Bring Your Own Tool" - Teams can use any tool which support Thrift endpoint, and can access EG data via Apiary.
Data Sharing Pattern: Applied

Other Brand | Hotels.com | Expedia.com
---|---|---

Federate
Architecture: Data Lake
Architecture: Storage
Architecture: Hive Metastore Service

RW Hive Metastore Containers

RO Hive Metastore Containers

NLB

ECS Service Discovery

CloudWatch Metrics

Metadata events (SNS Notifications)

AWS

Secrets Manager

Terraform
Architecture: Federation

Diagram showing the architecture of Apiary v1.
Architecture: WaggleDance Cluster
Architecture: Primary Metastore
Architecture: Remote Metastore

Remote Metastore

WaggleDance Containers

AWS Account B

Terraform

Apiary v1
Architecture: Remote Metastore

Terraform

WaggleDance Containers

Remote Metastore

AWS Account C
Data Sharing Pattern
module "apiary-datalake" {
    source = "git::https://github.com/ExpediaGroup/apiary-data-lake.git"
    aws_region = "us-west-2"
    instance_name = "test"
    apiary_tags = "${var.tags}"
    private_subnets = ["subnet1", "subnet2", "subnet3"]
    vpc_id = "vpc-123456"
    hms_docker_image = "${var.hms_docker_repository}"
    hms_docker_version = "1.0.0"
    hms_ro_cpu = "2048"
    hms_rw_cpu = "2048"
    hms_ro_heapsize = "8192"
    hms_rw_heapsize = "8192"
    apiary_log_bucket = "s3-logs-bucket"
    db_instance_class = "db.t2.medium"
    db_backup_retention = "7"
    apiary_managed_schemas = ["db1", "db2", "dm"]
    apiary_customer_accounts = ["aws_account_A", "aws_account_B"]
    ingress_cidr = ["10.0.0.0/8", "172.16.0.0/12"]
}
module "apiary-federation" {
  source                      = "git::https://github.com/ExpediaGroup/apiary-federation.git"
  instance_name              = "waggledance-test"
  wd_ecs_task_count         = "3"
  aws_region                 = "us-west-2"
  vpc_id                      = "vpc-123456"
  subnets                   = ["subnet1", "subnet2", "subnet3"]
  tags                      = "${var.tags}"
  ingress_cidr              = ["10.0.0.0/8", "172.16.0.0/12"]
  docker_image               = "${var.hms_docker_repository}"
  docker_version           = "1.0.0"
  primary_metastore_host     = "primary-metastore.yourdomain.com"
  primary_metastore_whitelist = ["db.*", "dm"]

  remote_metastores = [{
    endpoint         = "com.amazonaws.vpce.us-west-2.vpce-svc-1"
    port             = "9083"
    prefix           = "metastoreA"
    mapped-databases = "mydb_*"
  }]
}
Apiary Extensions

• Metastore SNS Listener (push changes to SNS)
• Event based Data Replication
• Metastore Authorization – Access control using Ranger
• GlueSync – Serverless access (e.g. Athena)
Current limitations

- Apiary can only be deployed on AWS

- Works with Hive 2.x+ clients
  - other versions untested

- Does not support Kerberos
Summary

• Migrated Expedia data lakes to the cloud
  • Allowed brands to operate their data platforms autonomously

• Created a multi data lake strategy
  • Delivers the data sharing capabilities as provided by single monolithic on premises cluster

• Solutions contributed to open source
  • Enabling adoption and contributions
Questions?