AWS
re:Invent

ANT325

One Data Lake, Many Uses: Enabling Multi-Tenant Analytics with Amazon EMR

Bruno Faria
EMR Solutions Architect
AWS

Radhika Ravirala
EMR Solutions Architect
AWS

AWS re:Invent

aws

Please register your email address as you come in

We’ll be using a tool called qwiklabs for the labs

Please provide us your email as you come into the room using the following link

https://amzn.to/2DJAxGB

Please register your email address as you come in

Launch the lab using

https://amzn.to/2KA6UJo

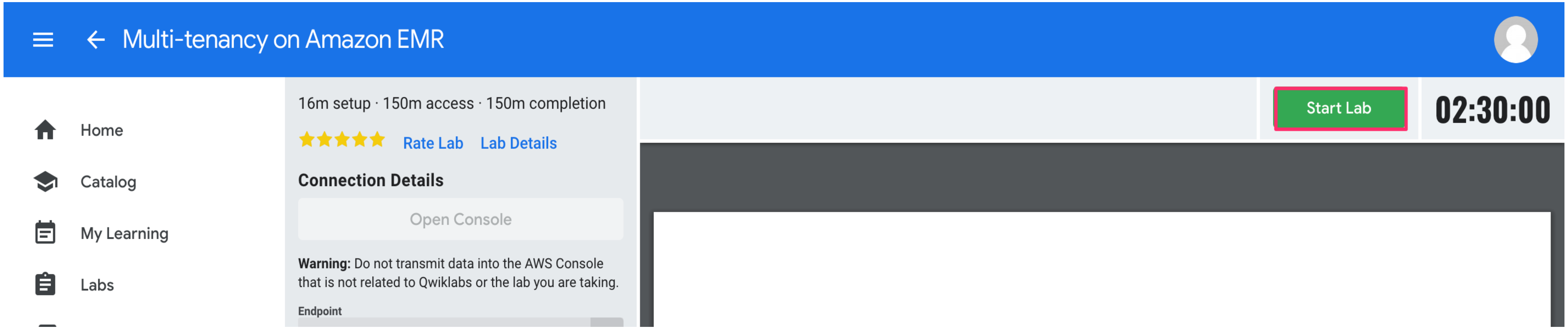
Workshop objectives

- Build a multi-tenant analytics environment with Amazon EMR over an Amazon Simple Storage Service (Amazon S3) data lake
- Explore options to manage and secure a multi-tenant data lake with tools such as LDAP and Kerberos
- Learn techniques to manage resource utilization in a shared environment

What is Qwiklabs?

- Provides access to AWS services for this workshop
- No need to provide a credit card
- Automatically deleted when you’re finished

Sign in and start the lab



16m setup · 150m access · 150m completion

★★★★★

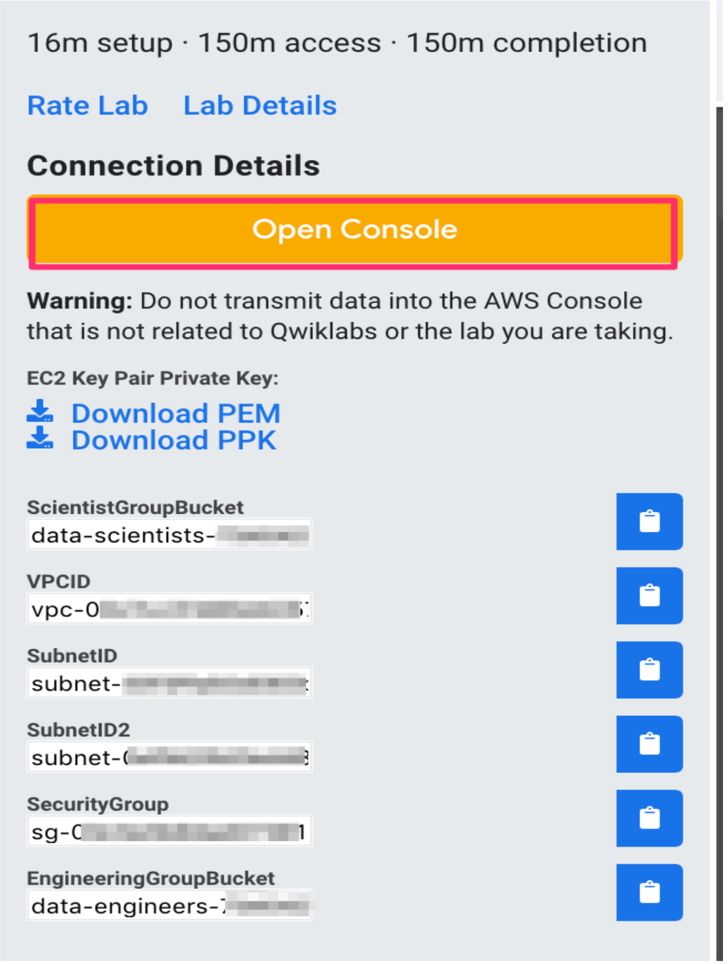
Connection Details

Warning: Do not transmit data into the AWS Console that is not related to Qwiklabs or the lab you are taking.

Endpoint

After the lab is started, you will see a lab setup progress bar. It takes ~20 min for the lab to be set up

Navigating Qwiklabs



16m setup · 150m access · 150m completion

Rate Lab   Lab Details

Connection Details

Warning: Do not transmit data into the AWS Console that is not related to Qwiklabs or the lab you are taking.

EC2 Key Pair Private Key:

Download PEM
Download PPK

ScientistGroupBucket	data-scientists-[redacted]
VPCID	vpc-0[redacted]
SubnetID	subnet-[redacted]
SubnetID2	subnet-[redacted]
SecurityGroup	sg-[redacted]
EngineeringGroupBucket	data-engineers-[redacted]

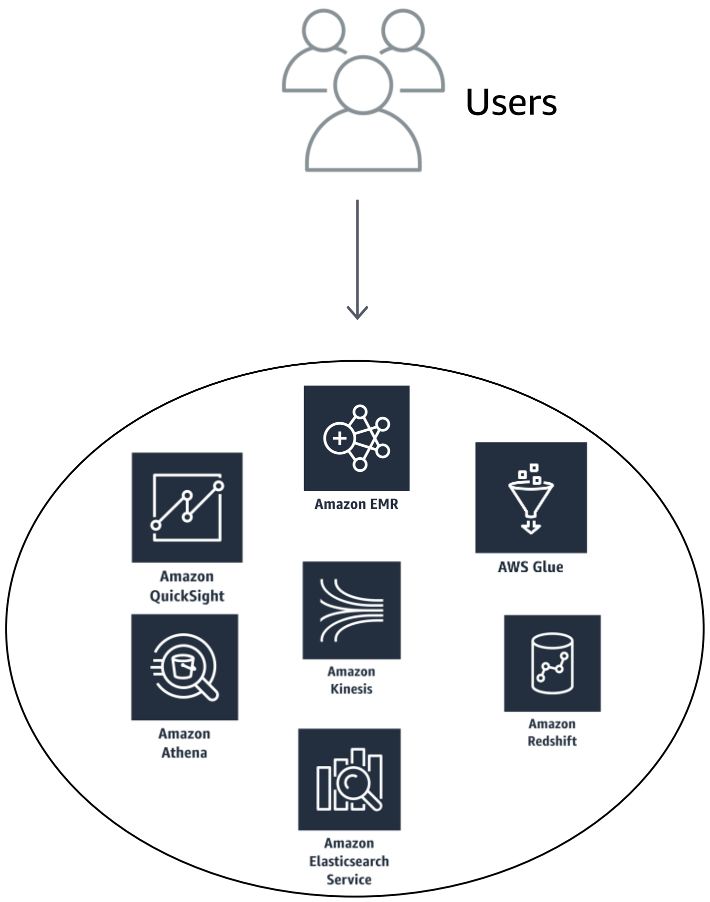
- **Open Console** : Opens AWS Management Console

- Links to different Interfaces

Everything you need for the lab

- Open AWS Console, log in, and verify the following AWS resources are created
    - Amazon EMR cluster
    - Amazon S3 bucket
    - Amazon Elastic Compute Cloud (Amazon EC2) instance with Apache Ranger
    - Amazon EC2 instance with OpenLDAP

Multi-tenancy motivation



Users

Amazon EMR

Amazon QuickSight

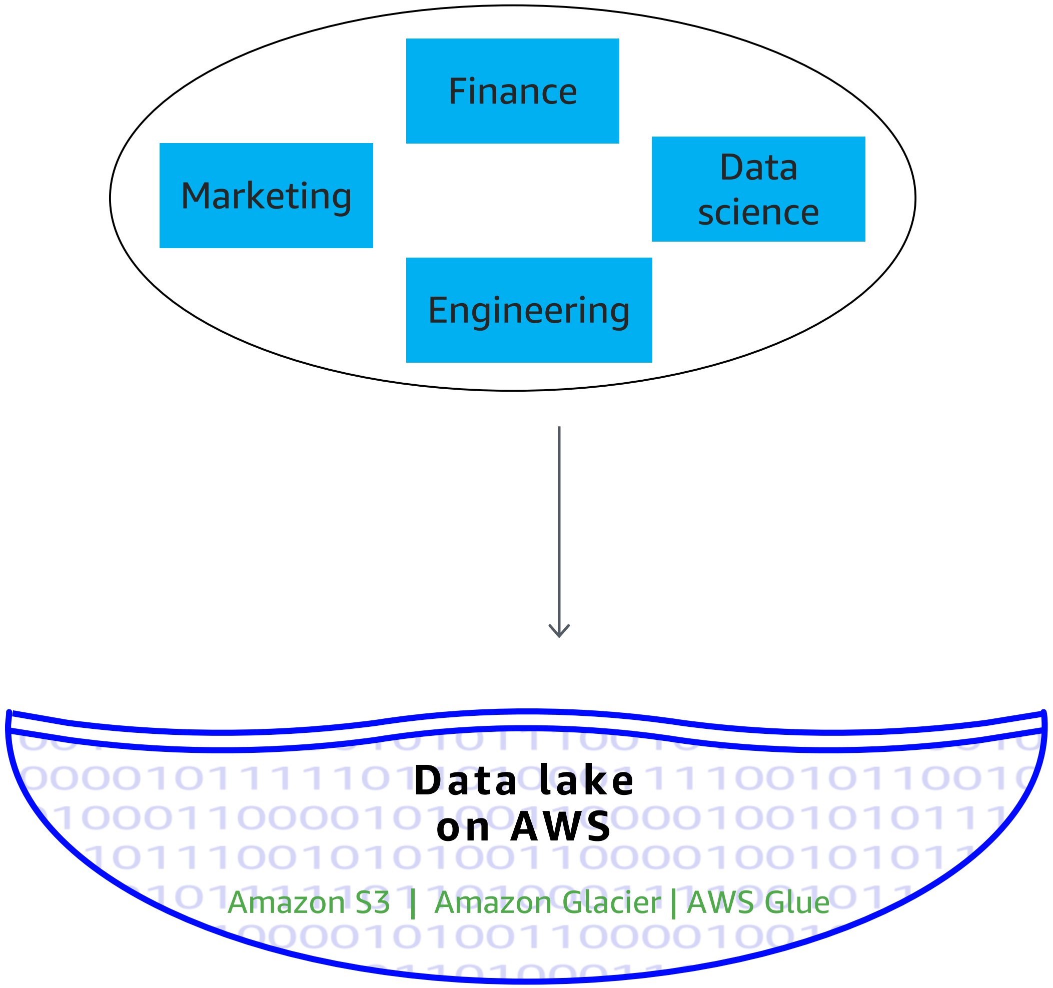
AWS Glue

Amazon Kinesis

Amazon Athena

Amazon Redshift

Amazon Elasticsearch Service



Finance

Marketing

Data science

Engineering

Data lake on AWS

Amazon S3 | Amazon Glacier | AWS Glue

User isolation

- Authentication (User identification)

Data isolation

- Authorization
  - Access rights/privileges to resources
    - Coarse-grained
    - Fine-grained

Resource isolation

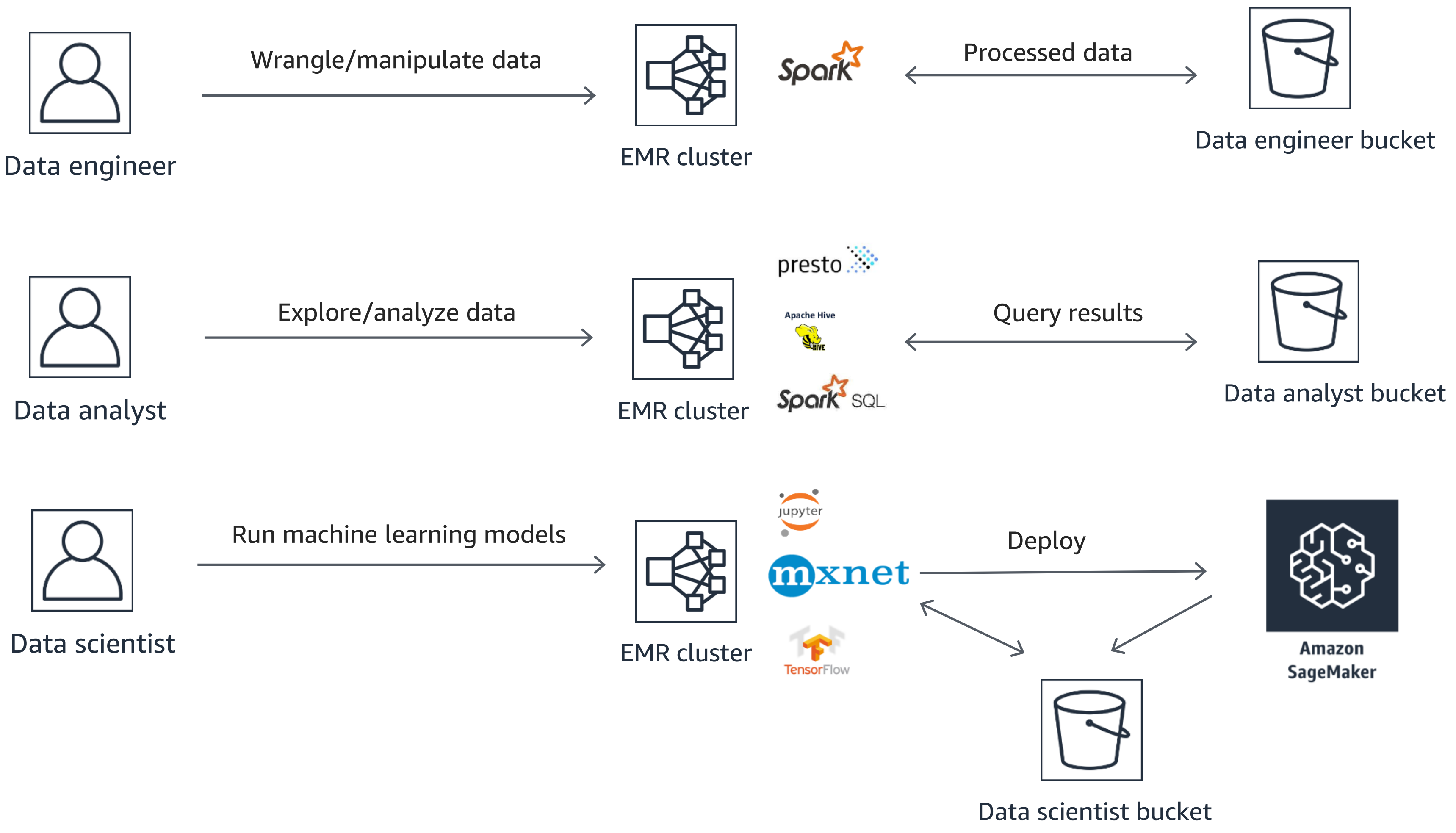
- Queues

Multi-tenancy with Amazon EMR

Terminology

- Silo mode
  - Tenant analytics (data + processing) is fully isolated from other tenants
  - Constructs logically “unique”
- Shared mode
  - Tenants share all analytic resources

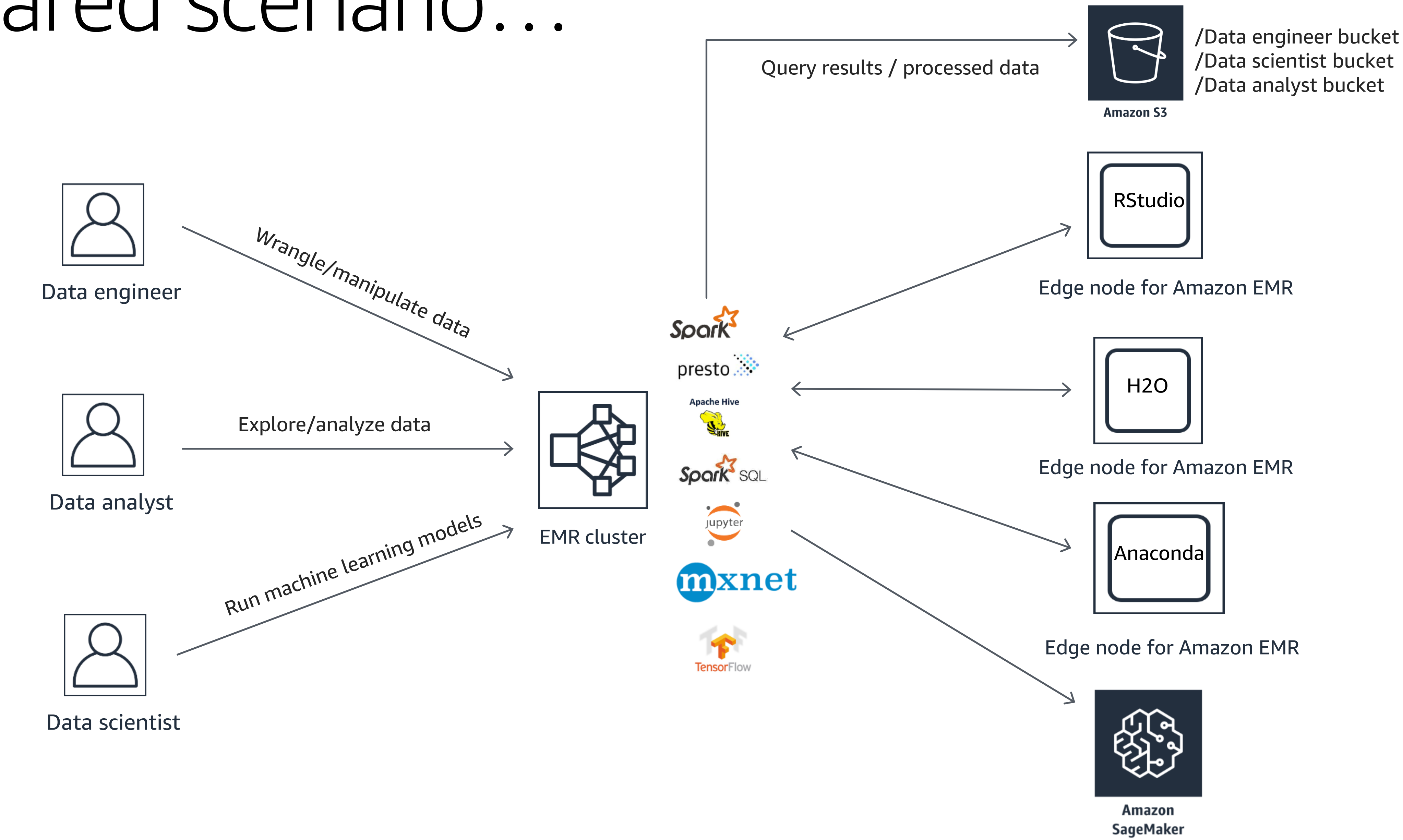
Silo scenario . . .



Multi-tenancy in Amazon EMR

- Silo mode
  - Each tenant gets their own Amazon EMR cluster with specific tools for processing/analyzing
  - Data stored in tenant’s S3 bucket or HDFS on the cluster
  - Hive meta store on the cluster or externally on Amazon Relational Database Service (Amazon RDS)
- Pros
  - Complete isolation
  - Custom configuration, contain blast radius
  - Easy to measure usage and resources
  - Can be cost effective when using Spot instances
- Cons
  - Cannot share data across clusters (especially when using HDFS)
  - Can be expensive

Shared scenario…

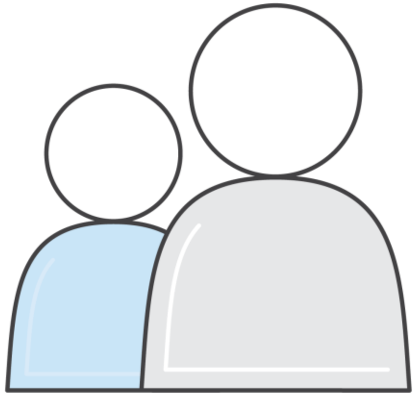


Multi-tenancy in Amazon EMR

- Shared mode
  - Tenants share the Amazon EMR cluster with tools installed for processing/analyzing/data science – all in one cluster
  - Data stored in tenant's S3 bucket or tenant's HDFS folder on the cluster
  - Hive metastore on the cluster or externally on Amazon RDS
- Pros
  - Less operational burden as there is one cluster to maintain
  - Can be cost effective if the cluster is well utilized
- Cons
  - Hard to measure usage and resources
  - Cannot customize the cluster for individual workloads
  - One configuration to fit all use cases

Authentication

Authentication



LDAP

HiveServer2
Presto coordinator
Spark Thrift server
Hue server
Zeppelin server

Kerberos

HiveServer2
Presto coordinator
Spark Thrift server
HBase

EC2 key pair

SSH as "hadoop"

AD join

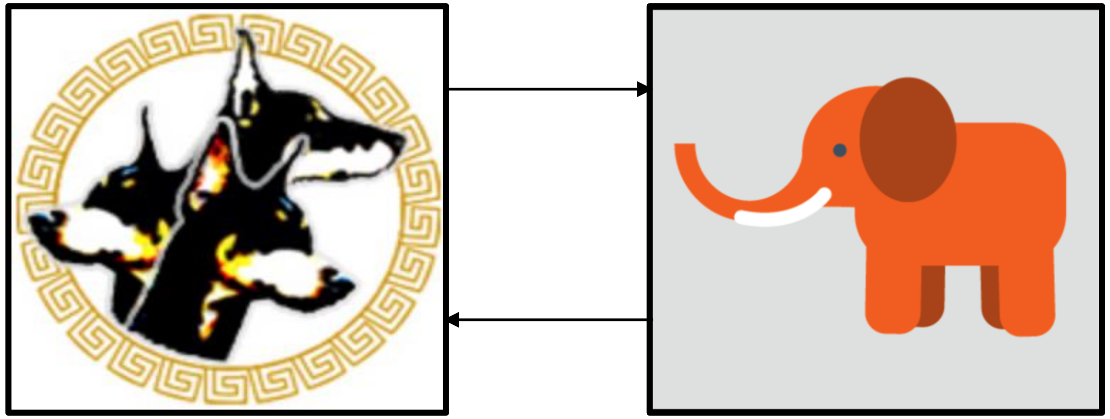
SSH as user

AWS credentials

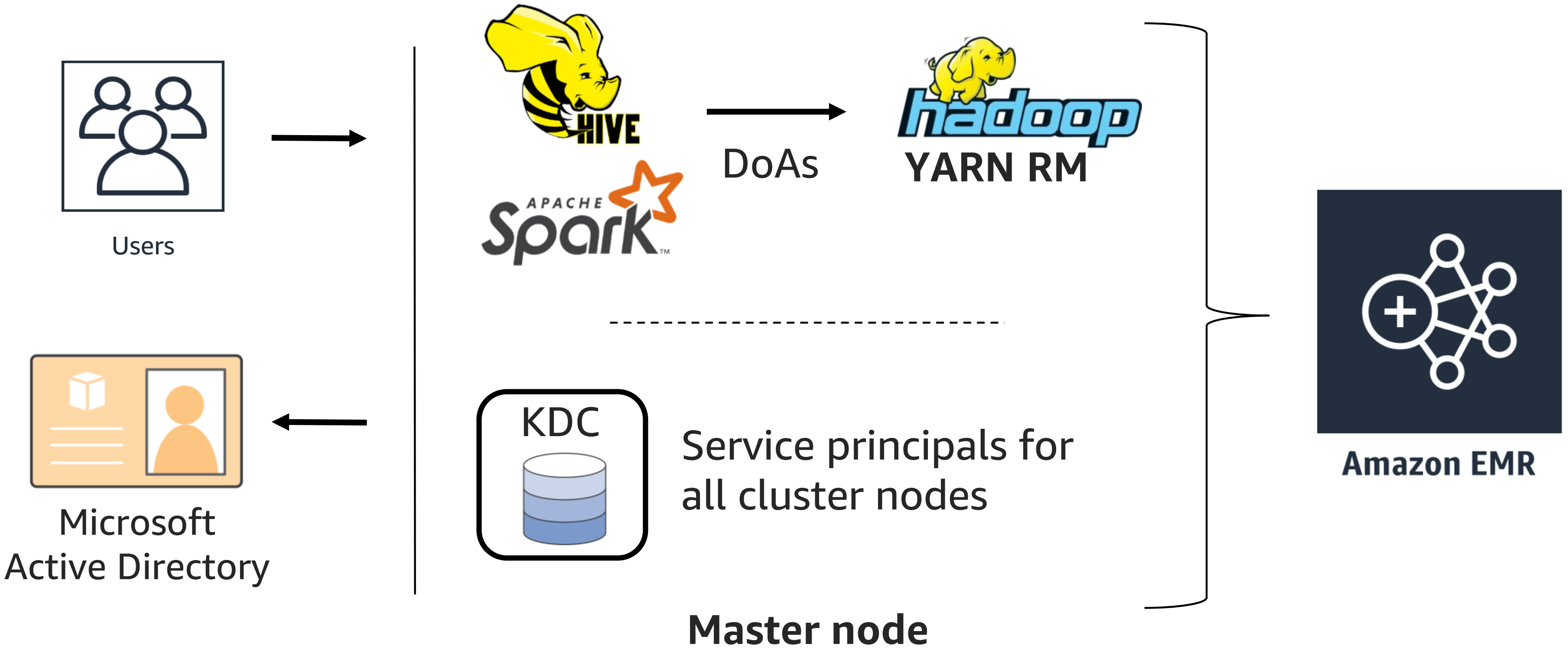
EMR Step (EMR API)

Authentication using Kerberos

- Network authentication protocol
- Eliminates the need for transmission of passwords across network
- Removes potential threat of an attacker sniffing the network



Kerberos authentication



Service principals for all cluster nodes

Master node

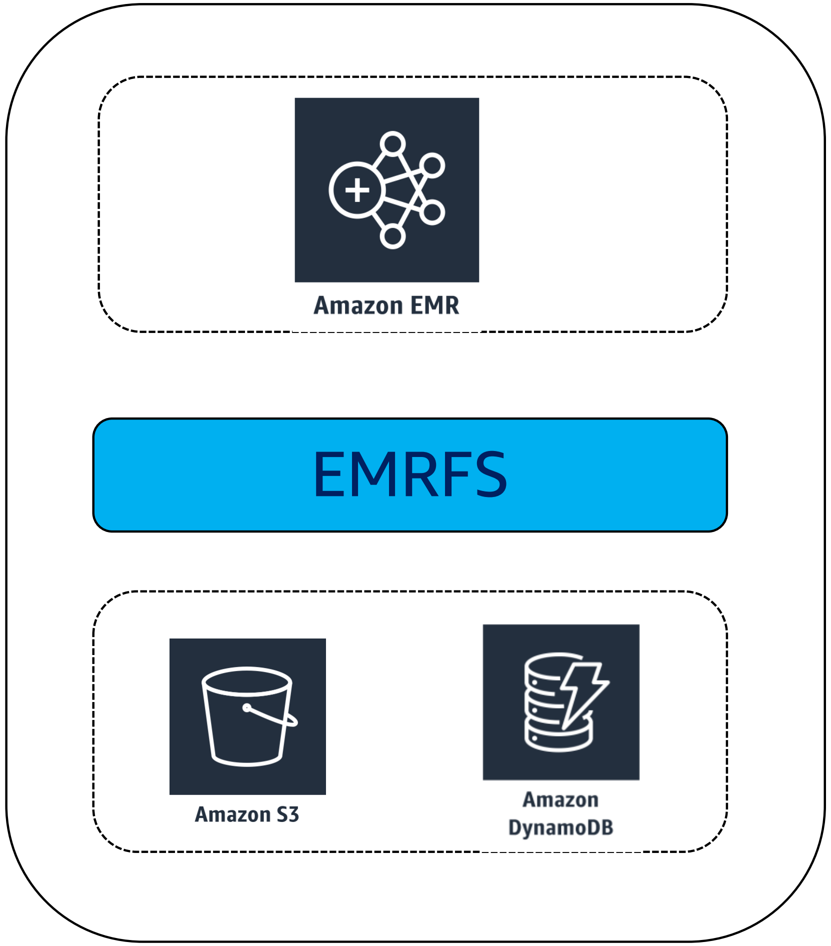
User isolation

Authorization

- Storage-based
  - EMRFS/Amazon S3 *
  - HDFS
- HiveServer2 and Presto (SQL-based)
- HBase
- Access control by cluster tag (AWS Identity and Access Management (IAM))
- Apache Ranger on Amazon EC2 instance (using AWS CloudFormation)

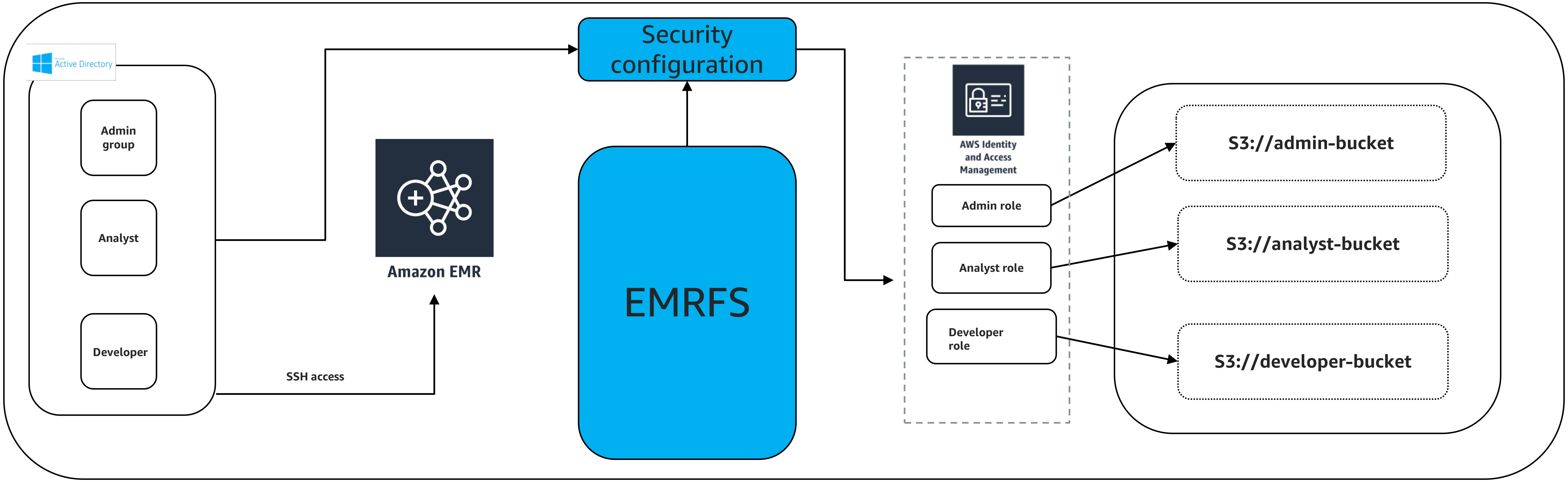
Amazon EMRFS

- Sits between Amazon EMR and and Amazon S3
- Amazon EMR clusters use EMRFS for reading and writing files from Amazon S3
- Provides consistent view and data encryption

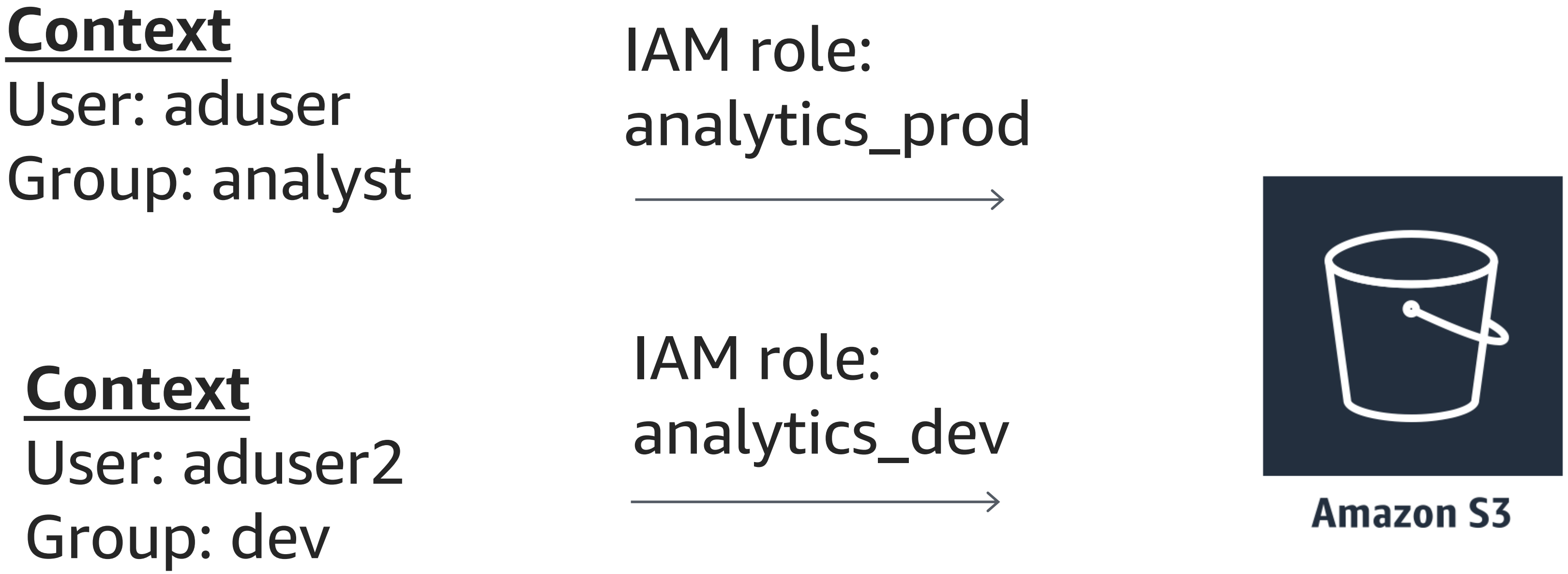


Authorization using EMRFS

- Use different IAM roles for EMRFS requests to Amazon S3
- These IAM roles can be cluster users, groups or the location of EMRFS data in Amazon S3



EMRFS storage authorization



Context
User: aduser
Group: analyst

IAM role:
analytics_prod

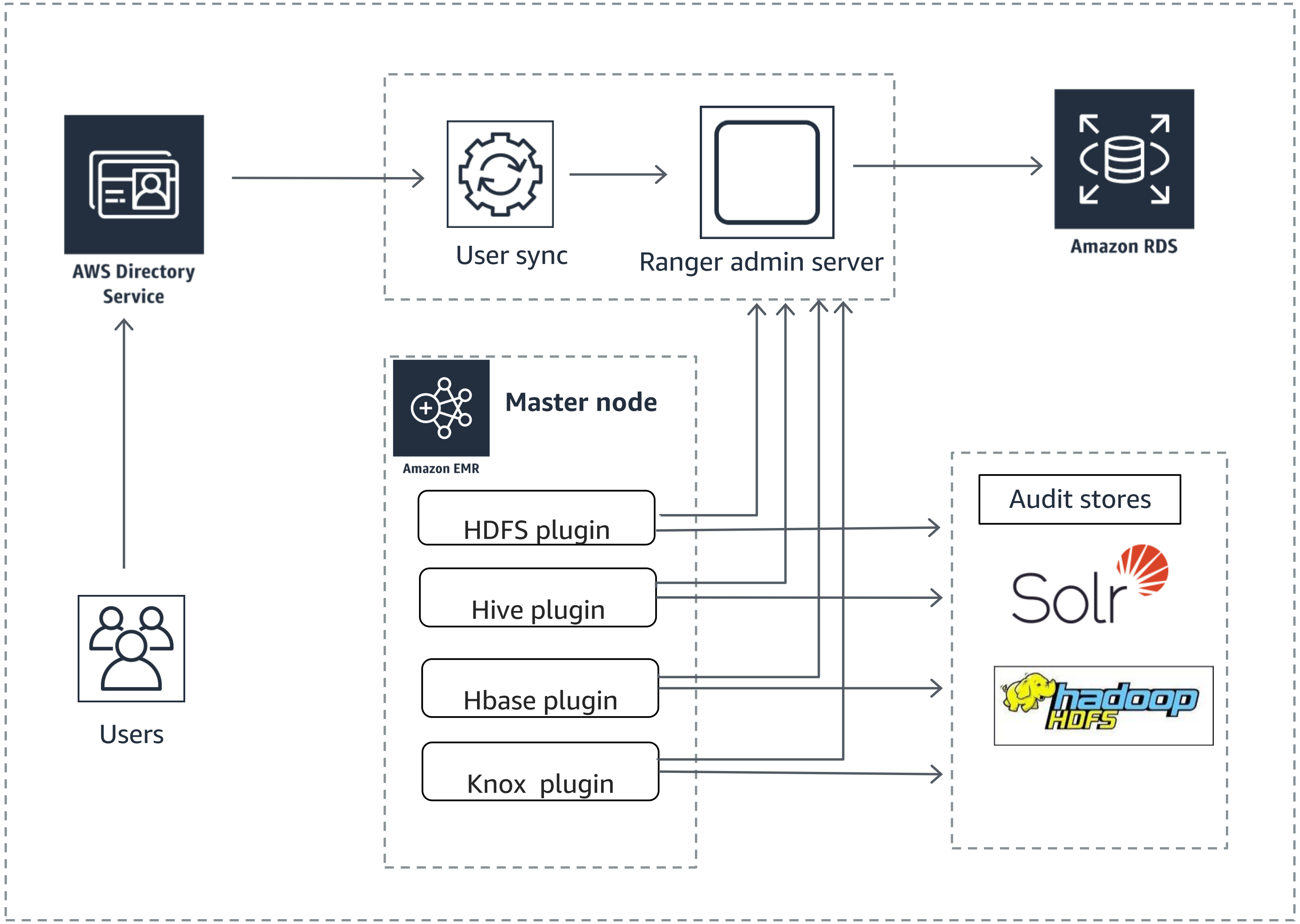
Context
User: aduser2
Group: dev

IAM role:
analytics_dev

Can map IAM roles to user, group, or Amazon S3 prefix

Fine-grained access using Apache Ranger

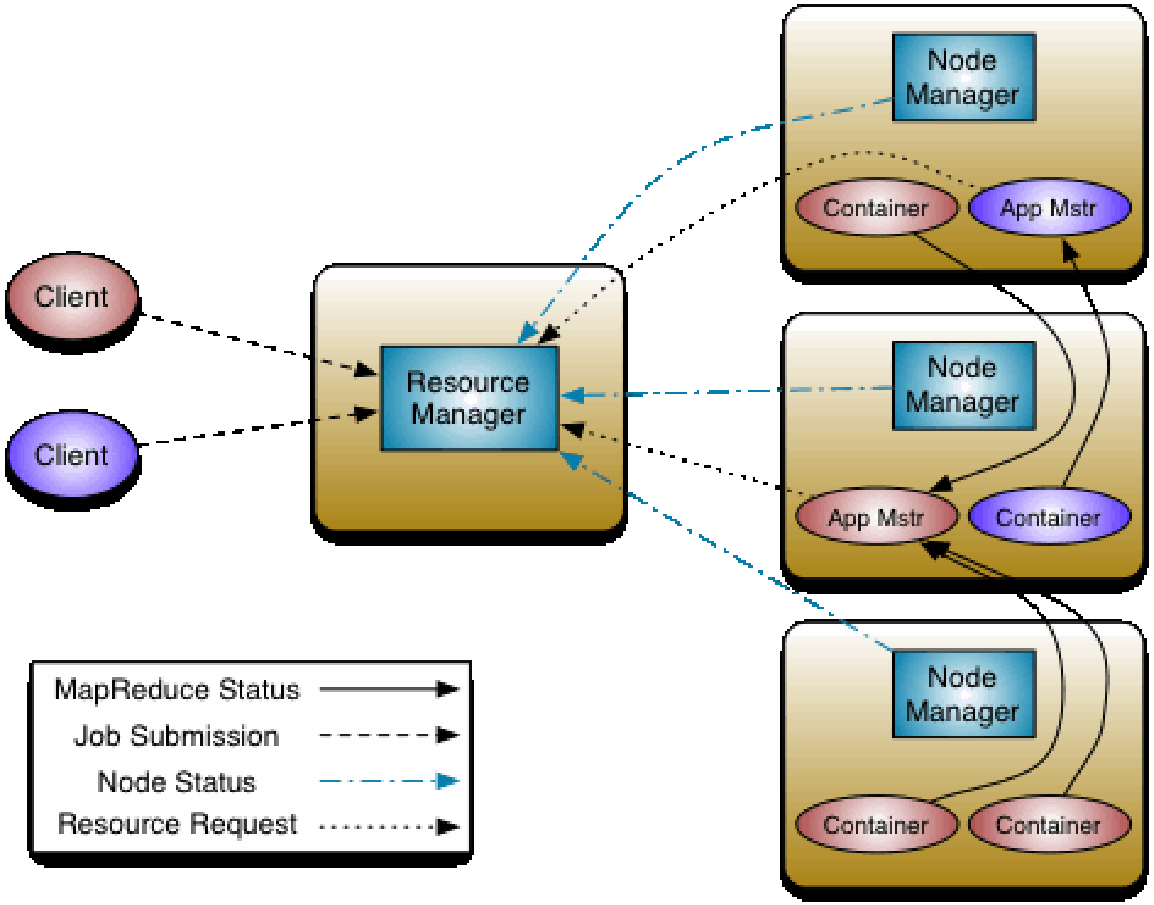
- Centralized web application with
  - Policy administration
  - Audit
  - Reporting modules
- Authorized users manage security policies – UI or REST APIs
- Security policies are enforced using lightweight Ranger Java plugins



Resource isolation in Amazon EMR

YARN: Yet Another Resource Negotiator

- **What does it do?**
  - Resource management
  - Scheduling/monitoring jobs

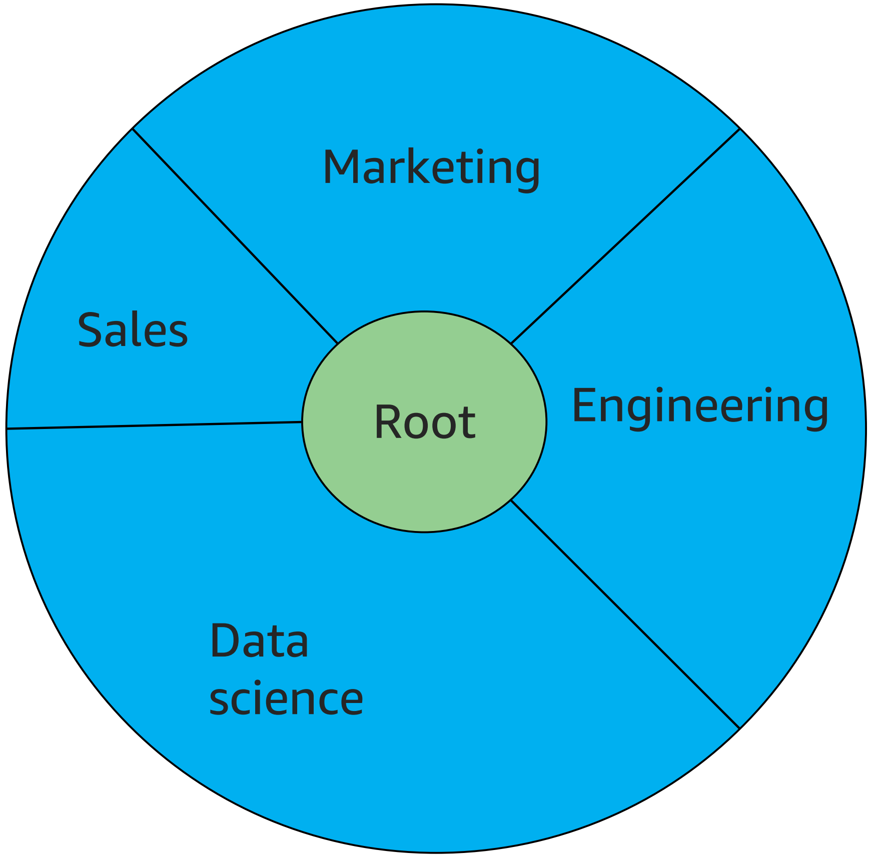


https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/yarn_architecture.gif

YARN

- Queues
  - Share cluster among multiple tenants
- Applications assigned to queues
- ‘root’ – parent of all queues
- Queues correspond to departments, users, or priorities

YARN queue example

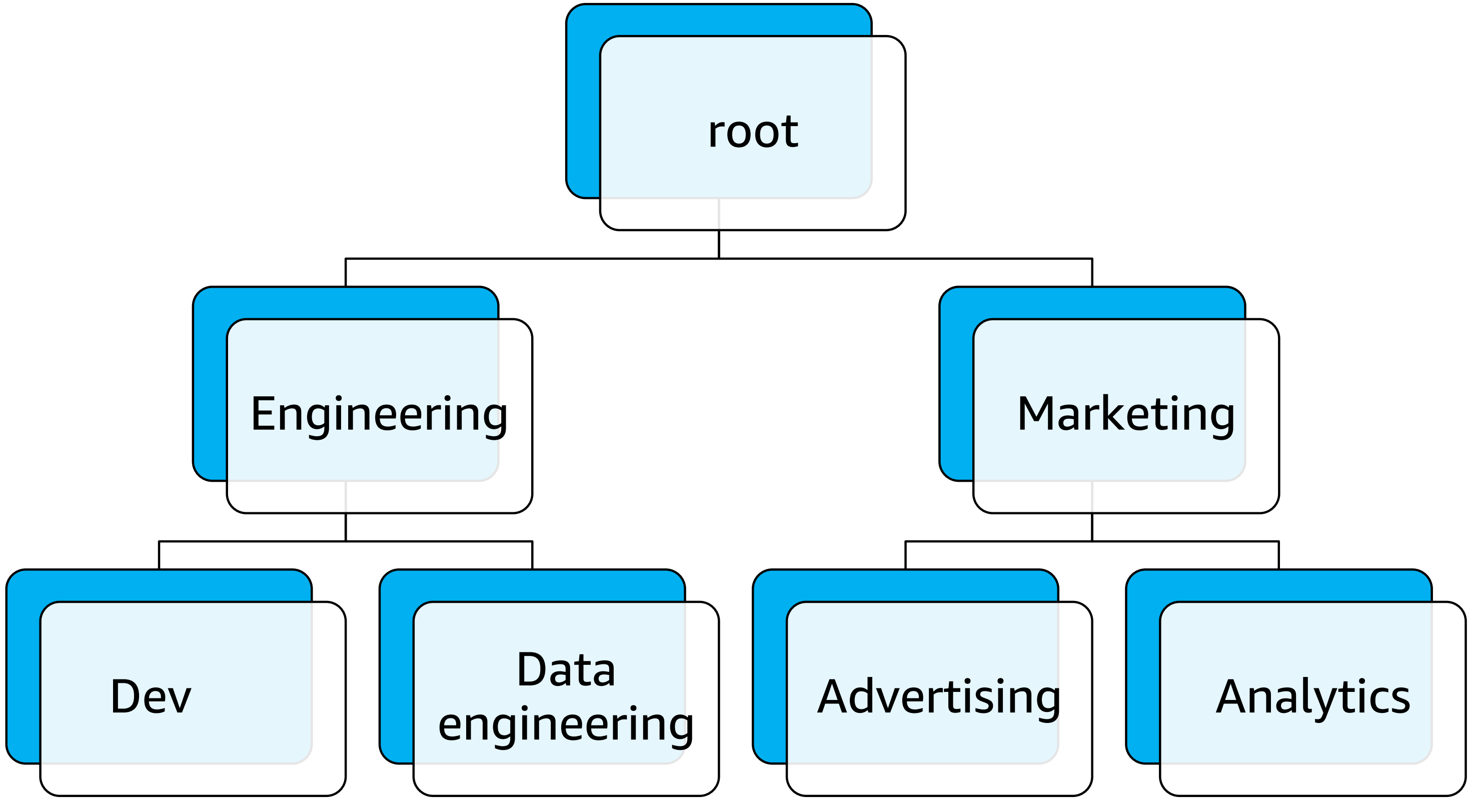


```
{
                    "Classification": "capacity-scheduler",
                    "Properties": {
                                        "yarn.scheduler.capacity.maximum-am-resource-percent": "0.6",
                                        "yarn.scheduler.capacity.resource-calculator":
"org.apache.hadoop.yarn.util.resource.DominantResourceCalculator",
      "yarn.scheduler.capacity.root.queues": "default,engineering,datascience,marketing",
      "yarn.scheduler.capacity.root.default.capacity": "10",
      "yarn.scheduler.capacity.root.default.user-limit-factor": "2",
      "yarn.scheduler.capacity.root.default.maximum-capacity": "40",
      "yarn.scheduler.capacity.root.engineering.capacity": "45",
      "yarn.scheduler.capacity.root.datascience.capacity": "30",
      "yarn.scheduler.capacity.root.marketing.capacity": "15",
      "yarn.scheduler.capacity.root.engineering.user-limit-factor": "2",
      "yarn.scheduler.capacity.root.datascience.user-limit-factor": "2",
      "yarn.scheduler.capacity.root.marketing.user-limit-factor": "2",
      "yarn.scheduler.capacity.root.engineering.maximum-capacity": "75",
      "yarn.scheduler.capacity.root.datascience.maximum-capacity": "55",
      "yarn.scheduler.capacity.root.marketing.maximum-capacity": "50",
      "yarn.scheduler.capacity.root.engineering.state": "RUNNING",
      "yarn.scheduler.capacity.root.datascience.state": "RUNNING",
      "yarn.scheduler.capacity.root.marketing.state": "RUNNING",
      "yarn.scheduler.capacity.root.engineering.acl_submit_applications": "*",
      "yarn.scheduler.capacity.root.datascience.acl_submit_applications": "*",
      "yarn.scheduler.capacity.root.marketing.acl_submit_applications": "*"
                    }
},
{
                    "Classification": "yarn-site",
                    "Properties": {
    "yarn.acl.enable": "true",
    "yarn.resourcemanager.scheduler.class":
"org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler"
                    }
}
```

AWS re:Invent

YARN scheduler

- Nested queues
- Queue weights – Control fair share of apps in the queue
- Manage queue access through ACLs



Action time!

AWS Lake Formation

Build a secure data lake in days

Register existing data or load new data using blueprints. Data stored in Amazon S3.

Secure data access across multiple services using single set of permissions.

No additional charge. Only pay for the underlying services used.

Quickly build data lakes

Move, store, catalog, and clean your data faster. Use ML transforms to de-duplicate data and find matching records.

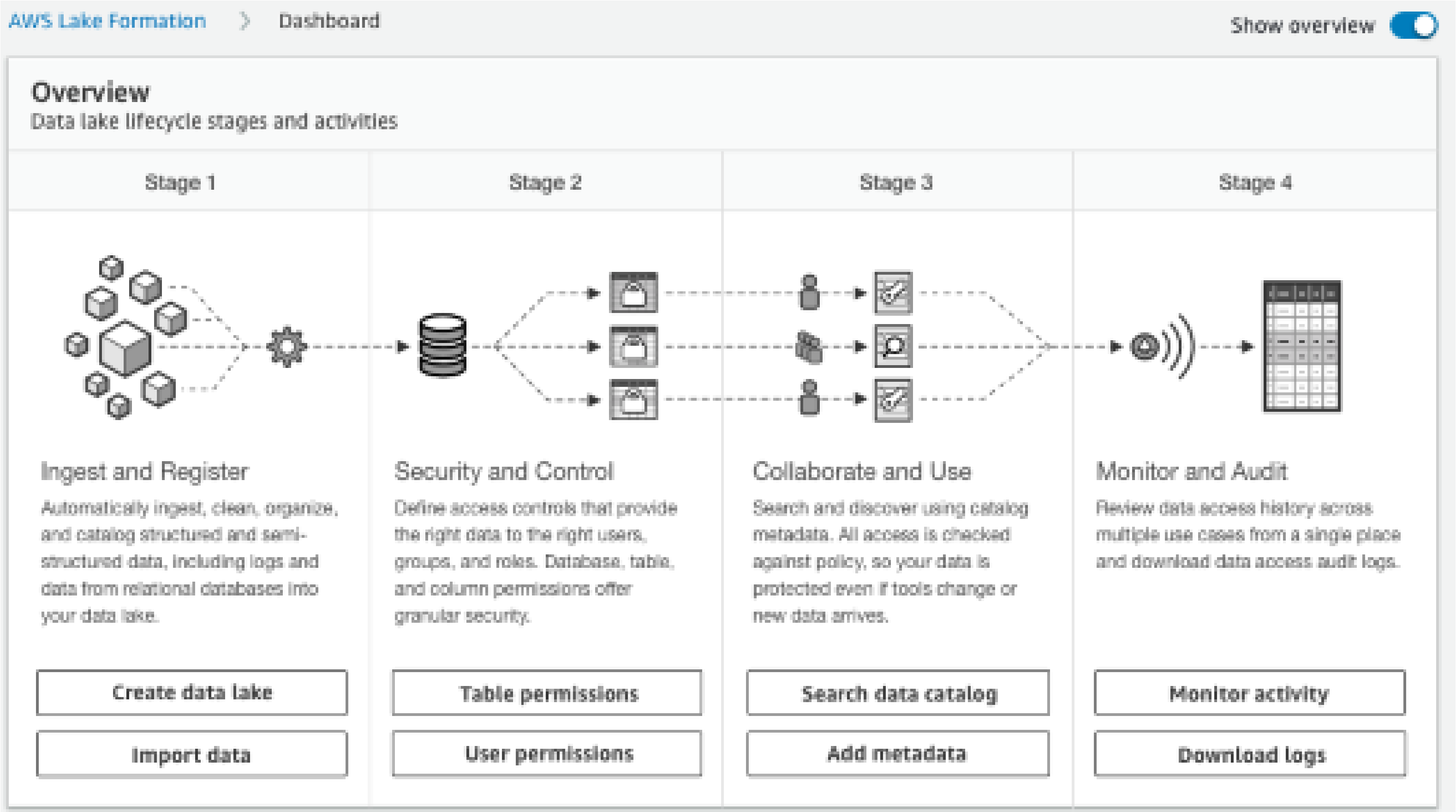
Easily secure access

Centrally define table and column-level data access and enforce it across Amazon EMR, Amazon Athena, Amazon Redshift Spectrum, Amazon SageMaker, and Amazon QuickSight

Share and collaborate

Use data catalog in Lake Formation to search and find relevant data sets and share them across multiple users and accounts

How it works



Overview

Data lake lifecycle stages and activities

Ingest and Register

Automatically ingest, clean, organize, and catalog structured and semi-structured data, including logs and data from relational databases into your data lake.

Security and Control

Define access controls that provide the right data to the right users, groups, and roles. Database, table, and column permissions offer granular security.

Collaborate and Use

Search and discover using catalog metadata. All access is checked against policy, so your data is protected even if tools change or new data arrives.

Monitor and Audit

Review data access history across multiple use cases from a single place and download data access audit logs.

Thank you!

Bruno Faria
EMR Solutions Architect
AWS

Radhika Ravirala
EMR Solutions Architect
AWS

AWS re:Invent

aws

!

Please complete the session survey in the mobile app.

AWS re:Invent

aws