
AWS
re:Invent

ANT324

# Implement data access controls for multi-tenant Amazon EMR clusters

Jia Liu

Senior Product Manager
AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Agenda

Authorization challenges with multi-tenancy

Amazon EMR User Role Mapper

Amazon EMR native integration with Apache Ranger

Amazon EMR integration with AWS Lake Formation

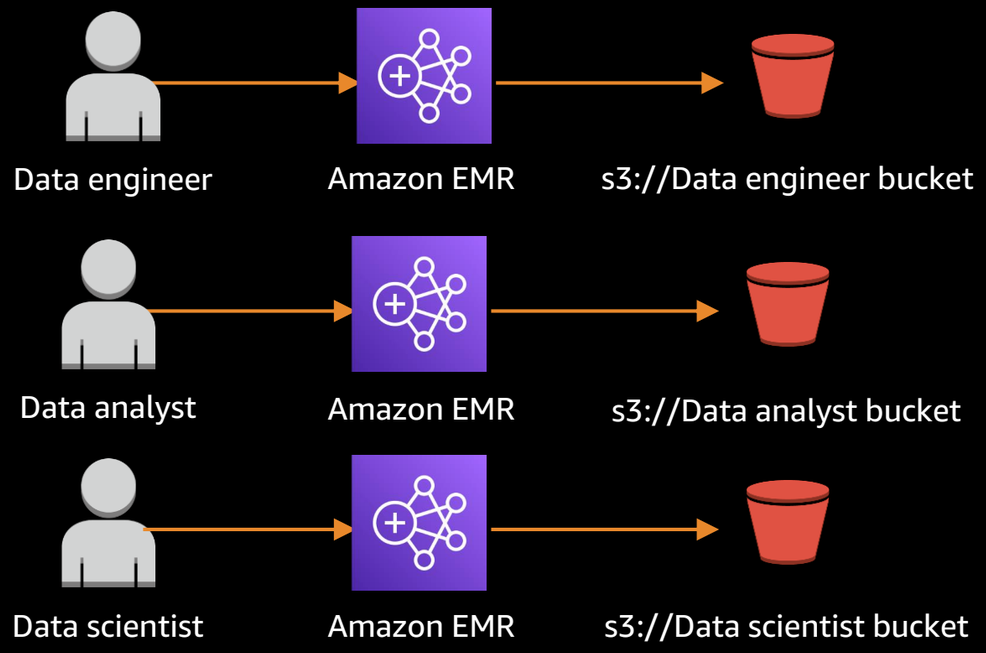
Summary





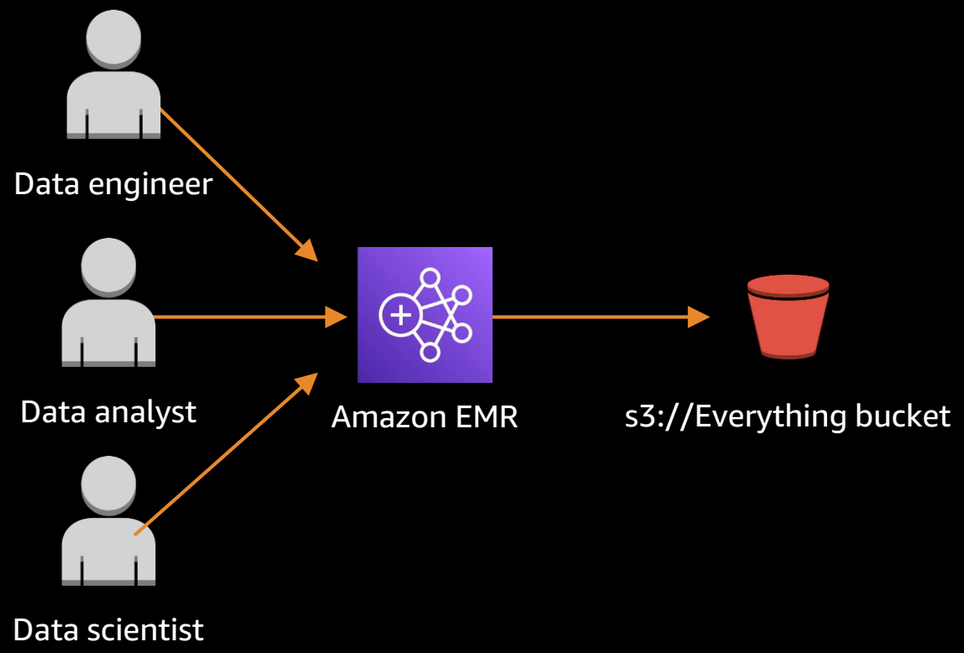
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Single-tenancy on Amazon EMR




- Single-tenancy for resource isolation
- Dedicated cluster (compute and storage) for a single user

# Multi-tenancy on Amazon EMR




- Multi-tenancy for cost optimization
- Running workloads for multiple clients or lines of business

# Authorization challenges with multi-tenancy

**By default, all users on Amazon EMR share the same access to AWS resources including S3 data**

- Amazon EMR does not recognize the individual users submitting jobs
- All users assume the same IAM role called EC2 instance profile

**AWS IAM permissions do not allow fine-grained access control**

- In context of Amazon S3, access is at bucket / prefix / object level

# How new data access control options work

- Amazon EMR built a process to intercept user requests
- Instead of assuming the default IAM role EC2 instance profile, users assume a customized IAM role depending on the access control option
- Amazon EMR admins can define authorization policies and associate with the customized IAM role
- Enforce data access with these authorization policies via the customized IAM role

# Two ways to control Amazon S3 data access

## Control access to Amazon S3 data directly

- Bucket (e.g., s3://enterprise-data)
- Prefix (e.g., s3://enterprise-data/sales/)
- Object (e.g., s3://enterprise-data/sales/record.xlsx)

## Control access to Amazon S3 data via metadata (aka. fine-grained)

- Databases (e.g., sales)
- Tables (e.g., product1, product2, product3)
- Columns (e.g., customer_ID, purchase_date, region)
- Rows (e.g., customer_a, customer_b)

# Access control via Amazon EMR User Role Mapper



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Amazon EMR User Role Mapper (URM)




Create mappings between users / groups and customized IAM roles via Bootstrap Action; Users assume customized IAM roles for authorizations

**Why?**

- Leverage AWS IAM permissions to control data access; manages all AWS resources including Amazon S3
- Dynamic mapping: allows you to change after launching the cluster
- Auditing with AWS CloudTrail
- Designed to replace EMRFS role mapping

# Before you begin with Amazon EMR URM

## Considerations and limitations

- Currently only available via AWS Labs (https://github.com/awslabs/amazon-emr-user-role-mapper)
- Supports Spark and Hive – Hive requires enabling user / group mapping and authorization like SQL standards-based authorization
- Does not support PrestoDB or PrestoSQL

# Fine-grained access control with Apache Ranger

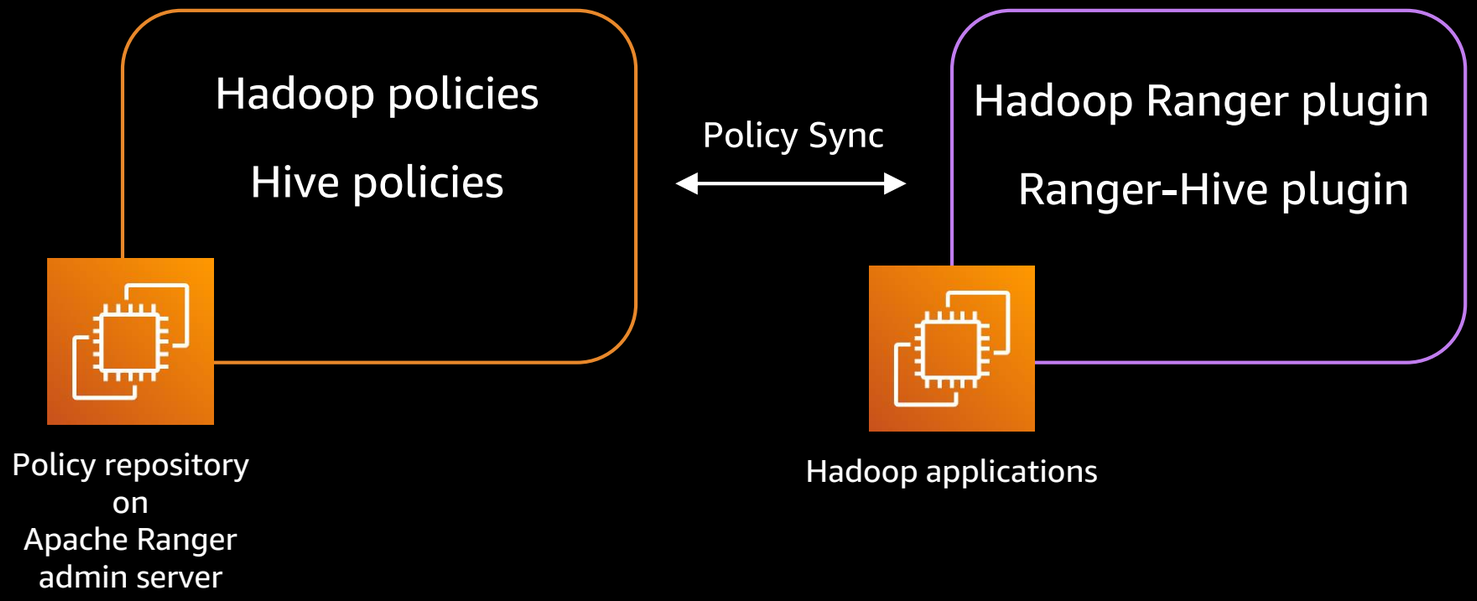


AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# What is Apache Ranger

New




**Apache Ranger**

- Open-source
- Centralized authorization policy repository
- Works with many Hadoop applications (Hadoop, Hive, and more)

# Integration with Apache Ranger

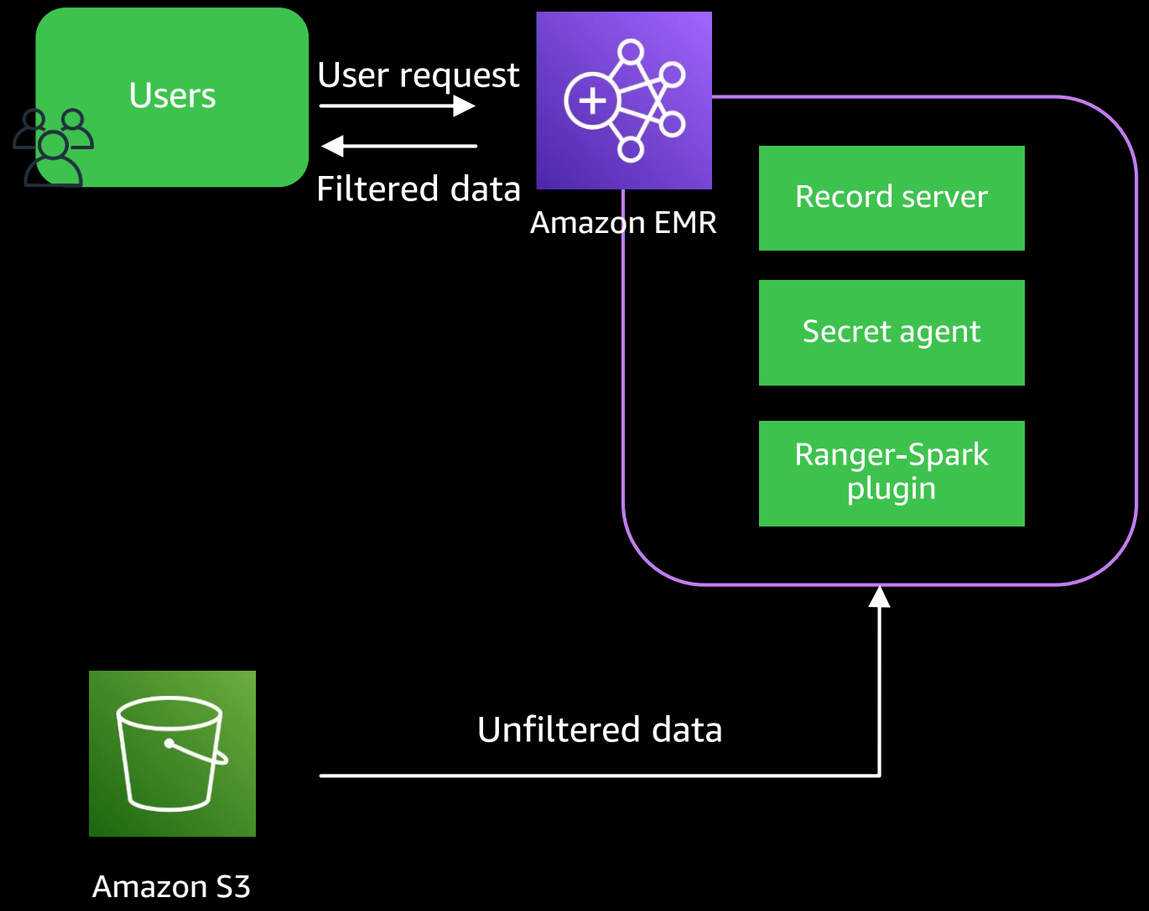
Set up a new or bring an existing Apache Ranger server for centralized data access control with Amazon EMR

**Why?**

- Amazon EMR automatically installs and configures Ranger plugins
- Authorize Amazon S3 data at the prefix and object level
- Enforce table and column level policies for Apache Spark and Hive applications to access the Hive Metastore
- Central auditing with Amazon CloudWatch Logs

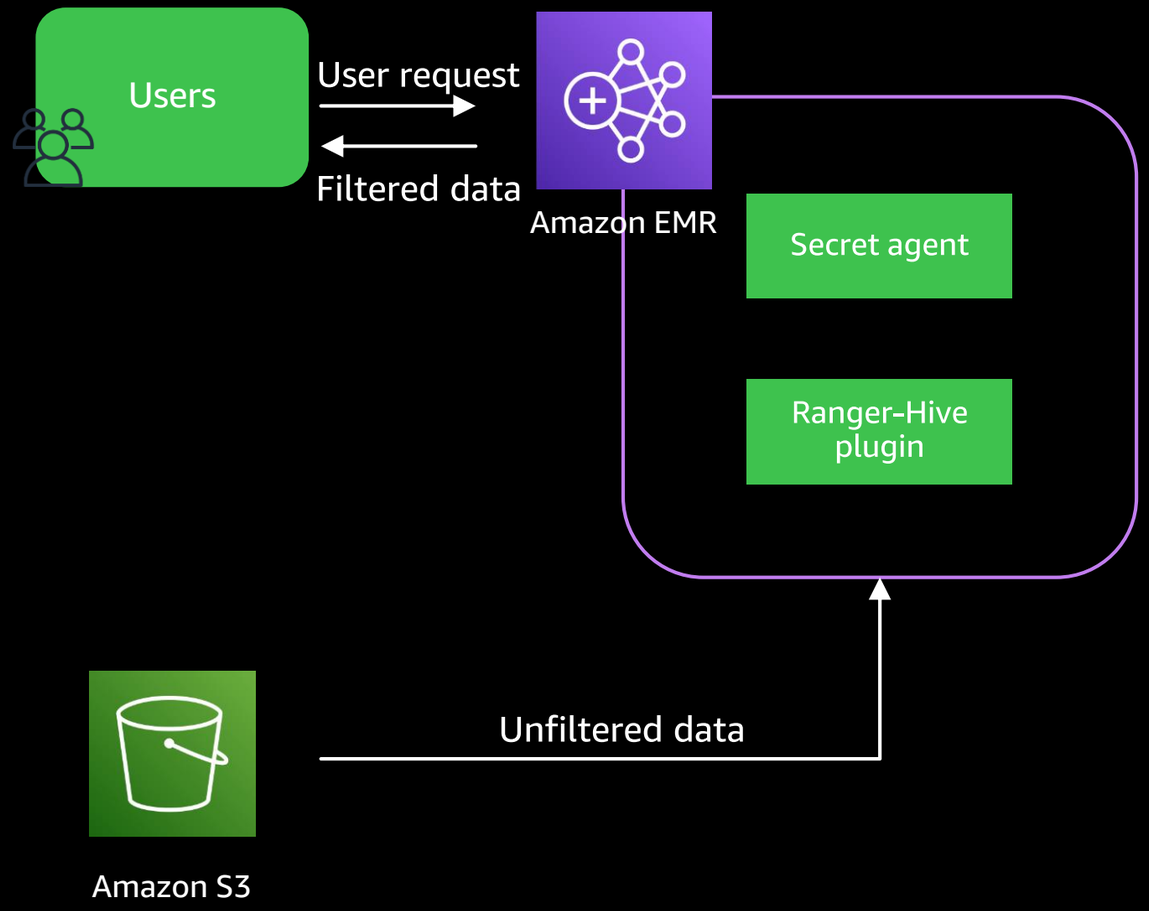
# Amazon EMR Ranger-Spark plugin

New




- **Amazon EMR Secret Agent**
  - Intercepts user requests
  - Identifies the right IAM credentials
  - Manages temporary credentials
- **Amazon EMR Record Server**
  - Forces Spark to delegate action of reading data
  - Reads and filters data
- **Ranger-Spark plugin** syncs up authorization policies at database / table / column level
- SparkSQL read access only

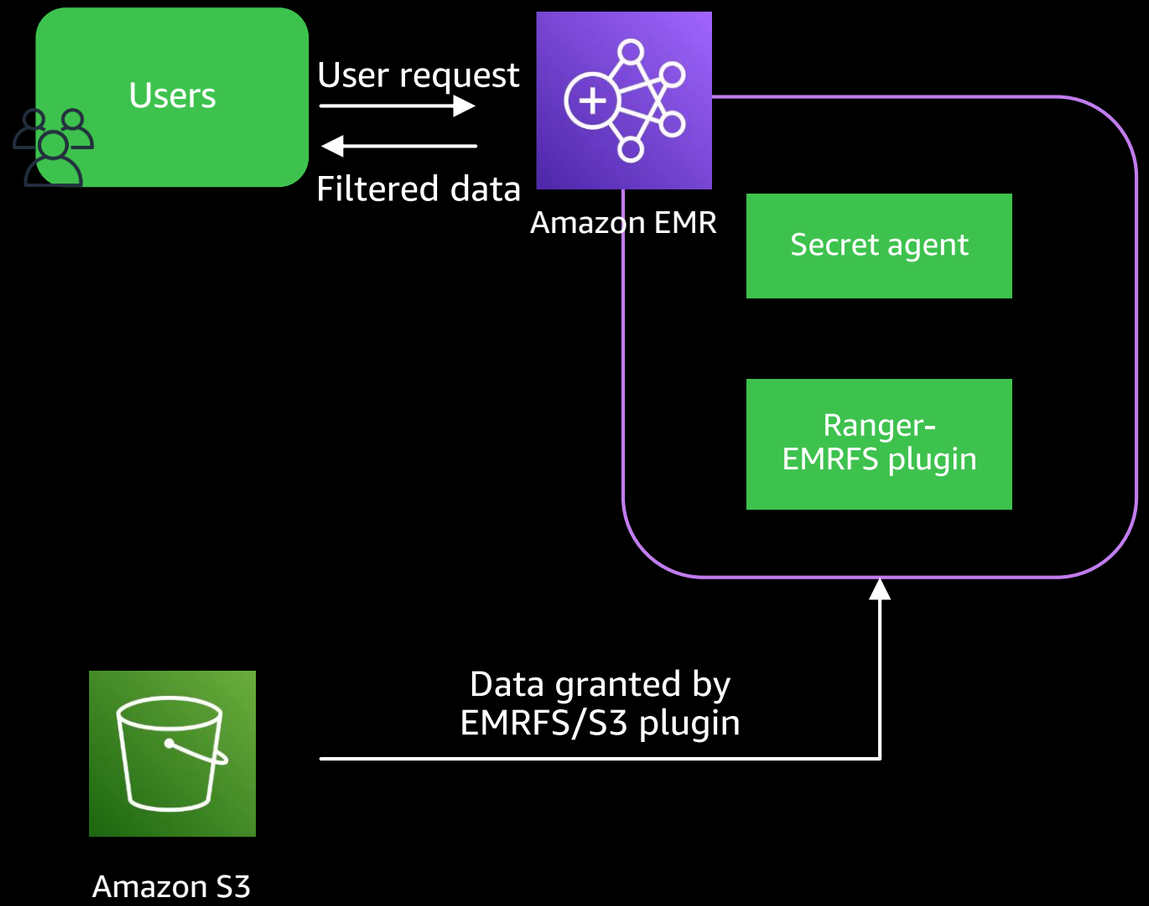
# Amazon EMR Ranger-Hive plugin




- Hive plugin provided by Amazon EMR uses the same plugin provided by the open-source community
- Use your existing Hive policies with Ranger-Hive plugin
- In addition to database / table / column level policies, Ranger-hive plugin supports row-level and data masking

# Amazon EMR Ranger-EMRFS plugin

New




**Manages access to Amazon S3 via EMR File System (EMRFS)**

- Required as a minimum Apache Ranger configuration
- Define prefix-level and object-level authorization policies
- Does not support S3 Client-Side Encryption

# Configure integration with Apache Ranger

## On Ranger admin server

- Define polices for each Hadoop application

## On Amazon EMR via security configuration

- Establish TLS connectivity between Apache Ranger admin server and Amazon EMR Ranger-plugins
- Leverage AWS Secrets Manger to store secrets for creating truststores and keystores
- Assign IAM roles
- Choose Amazon CloudWatch Log group for auditing

# IAM roles for integration with Apache Ranger

## IAM role for Apache Ranger

- Defines the privileges trusted processes have when submitting Spark and Hive jobs

## IAM role for other AWS services

- Defines the permissions that the end user has when accessing AWS services that are not protected by Ranger plugins
- We recommend not to grant this role access to Amazon S3

# Before you begin with the integration

## Considerations and limitations:

- Compatible with Apache Ranger admin server 2.x
- Supports Hive metastore only
- Supports Apache Zeppelin, Hue, and SSH
- Does not support SparkSQL for INSERT INTO and INSERT OVERWRITE
- Does not support Spark row-level filtering and data masking

## Coming soon:

- Amazon EMR 6.x
- AWS Glue metadata store support
- PrestoSQL support

# Fine-grained access control with AWS Lake Formation

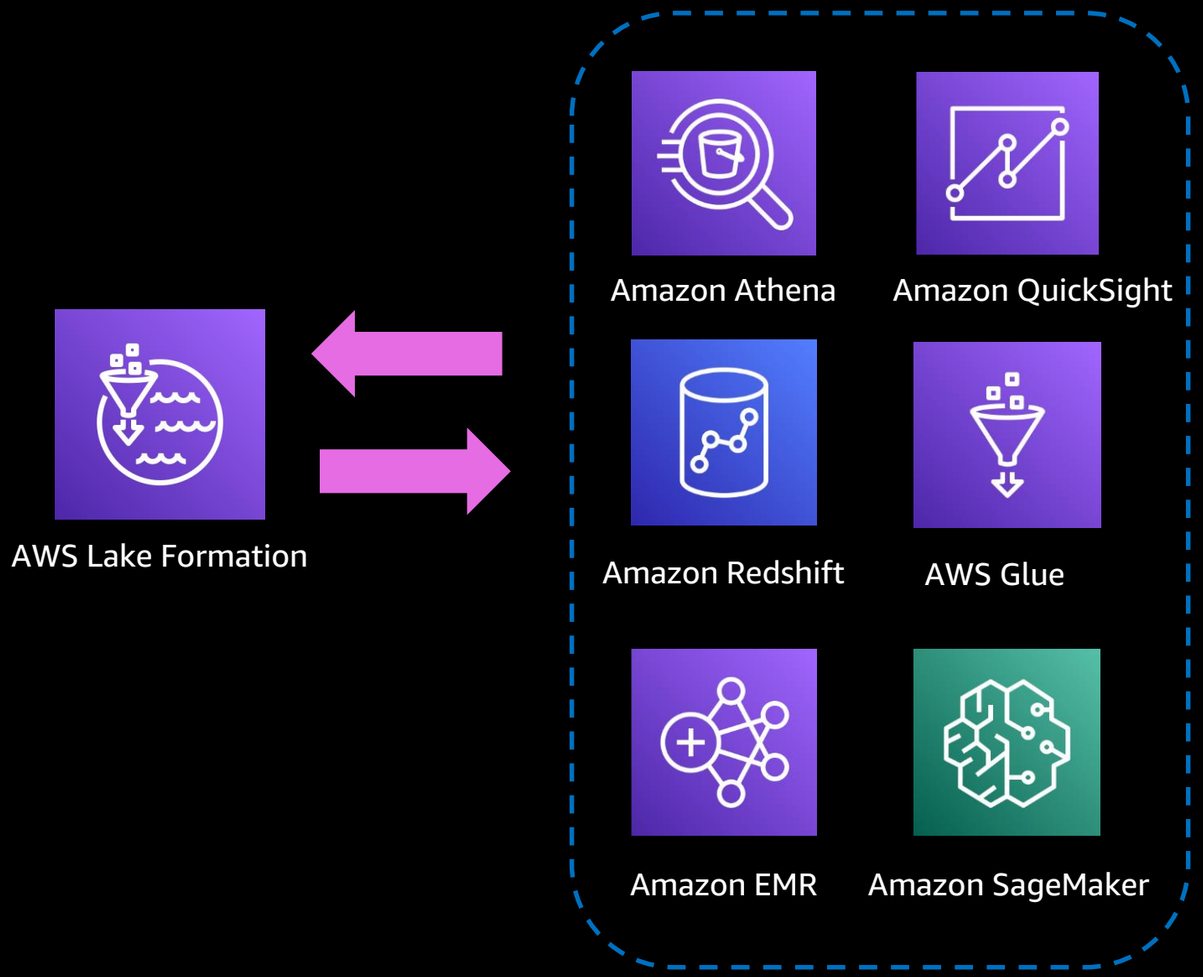


AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# What is AWS Lake Formation

New




- Fully managed AWS service
- Works with various AWS Analytics services
- Discovers, catalogs, cleanses, and secures data in an Amazon S3 data lake.

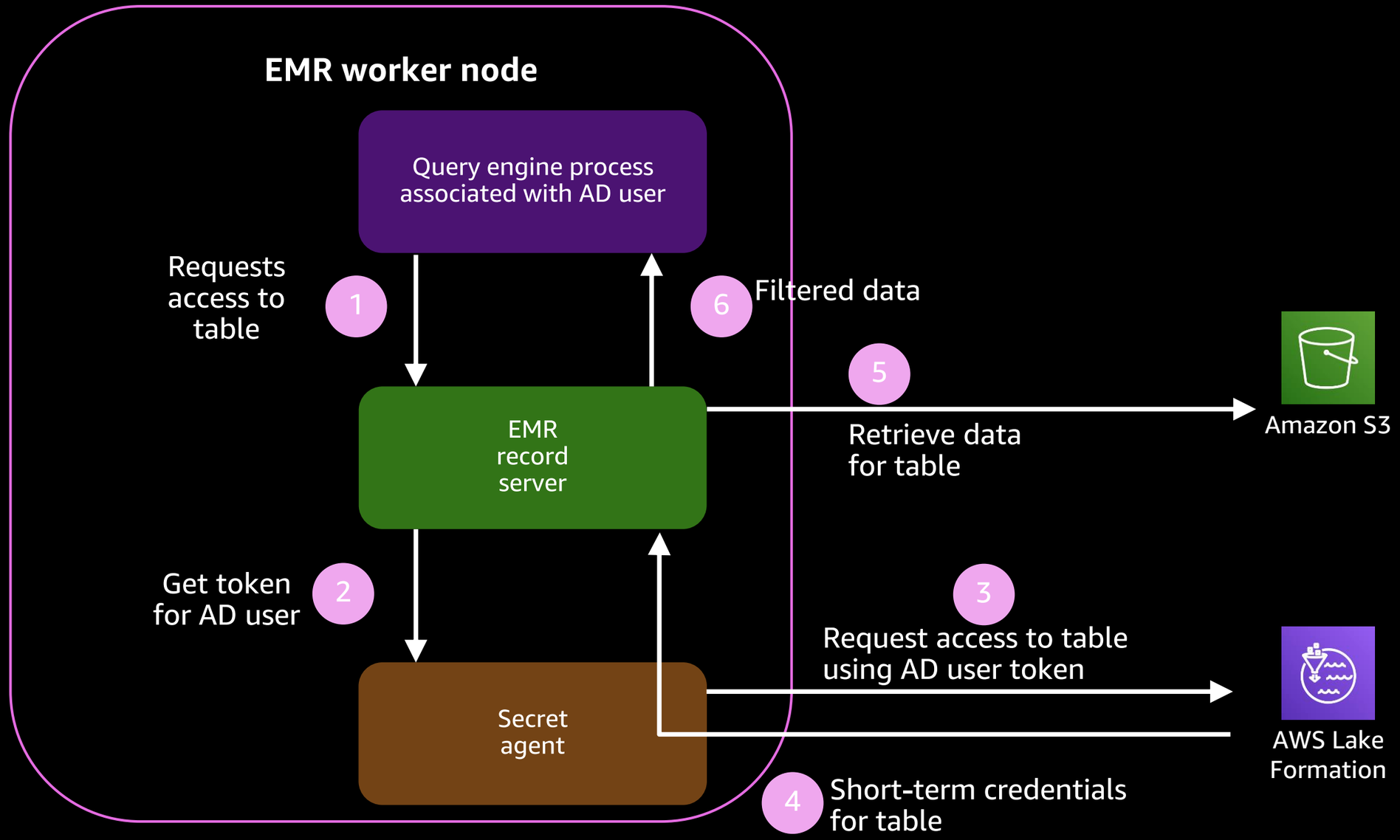
# Integration with AWS Lake Formation

Define and manage corporate identity-based permissions in AWS Lake Formation for Spark users to access databases, tables, and columns in AWS Glue Data Catalog

**Why?**

- Metadata protection with Spark applications by Amazon EMR record server
- Federated single sign-on from your enterprise identity system
  - Active Directory Federation Services (AD FS), Auth0, Okta, and many others
  - Uses Security Assertion Markup Language (SAML) 2.0
- Cross-account authorization
- Centralized auditing with AWS CloudTrail

# How data access is controlled



# Configure integration with AWS Lake Formation

**On AWS Lake Formation**

- Create and manage table and column level permissions
- Enable data filtering explicitly via Lake Formation console
- Enable cross-account filtering as you need

**On Amazon EMR via security configuration**

- Configure IAM roles
- Specify the metadata of Identity Provider via Amazon S3

# AWS IAM roles for this integration

## IAM role for AWS Lake Formation

- Defines the privileges for users accessing through identity provider
- The federated user should access Amazon S3 data only through Lake Formation

## IAM role for other AWS services

- Defines the permissions for accessing non-AWS Lake Formation services

# Before you begin with the integration

**Considerations and limitations:**

- Supports identity providers via SAML only
- Supports AWS Glue Data Catalog only; no permissions on S3 buckets directly
- Supports Spark only
- Supports Zeppelin and Notebooks
- Does not support SparkSQL for row-level filtering and writing to/creating new tables

**Coming soon:**

- Amazon EMR 6.x
- Amazon EMR Steps

# Summary

| Feature | User Role Mapper (URM) | Apache Ranger | AWS Lake Formation |
|---|---|---|---|
| **Fine-grained access control database / table / column** | N/A | Yes | Yes |
| **Metadata store** | All | Hive metastore | AWS Glue Data Catalog |
| **Read / write to metadata table** | Read and write | Does not support SparkSQL to write | Does not support SparkSQL to write |
| **Direct control on S3** | Yes | Yes, via EMRFS plugin | No |
| **EMR release requirement** | 5.29+ | 5.32+ | 5.31+ |
| **Kerberos is required** | Yes | Yes | Yes |
| **Auditing** | AWS CloudTrail | Amazon CloudWatch Logs | AWS CloudTrail |
| **Query engine** | Apache Spark and Hive | Apache Spark and Hive | Apache Spark |
| **Job submission** | All | Apache Zeppelin, Hue, SSH | Apache Zeppelin and EMR Notebooks |
| **Multi-master node** | Yes | No | No |

# Summary

| Feature | User Role Mapper (URM) | Apache Ranger | AWS Lake Formation |
|---|---|---|---|
| **Fine-grained access control database / table / column** | N/A | Yes | Yes |
| **Metadata store** | All | Hive metastore | AWS Glue Data Catalog |
| **Read / write to metadata table** | Read and write | Does not support SparkSQL to write | Does not support SparkSQL to write |
| **Direct control on S3** | Yes | Yes, via EMRFS plugin | No |
| **EMR release requirement** | 5.29+ | 5.32+ | 5.31+ |
| **Kerberos is required** | Yes | Yes | Yes |
| **Auditing** | AWS CloudTrail | Amazon CloudWatch Logs | AWS CloudTrail |
| **Query engine** | Apache Spark and Hive | Apache Spark and Hive | Apache Spark |
| **Job submission** | All | Apache Zeppelin, Hue, SSH | Apache Zeppelin and EMR Notebooks |
| **Multi-master node** | Yes | No | No |

# Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



# Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.