
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SEC325-R

Beyond database password management: 5 use cases for AWS Secrets Manager

Steven Emelander

Software Dev Manager, AWS Secrets Manager
Amazon Web Services

Rushir Patel

Sr. WW Security Specialist
Amazon Web Services

Agenda

Introduction to AWS Secrets Manager

5 popular use cases

- Generic credentials
- API keys
- SSH keys
- Hybrid workloads
- Third-party tokens

Whiteboarding + Q&A

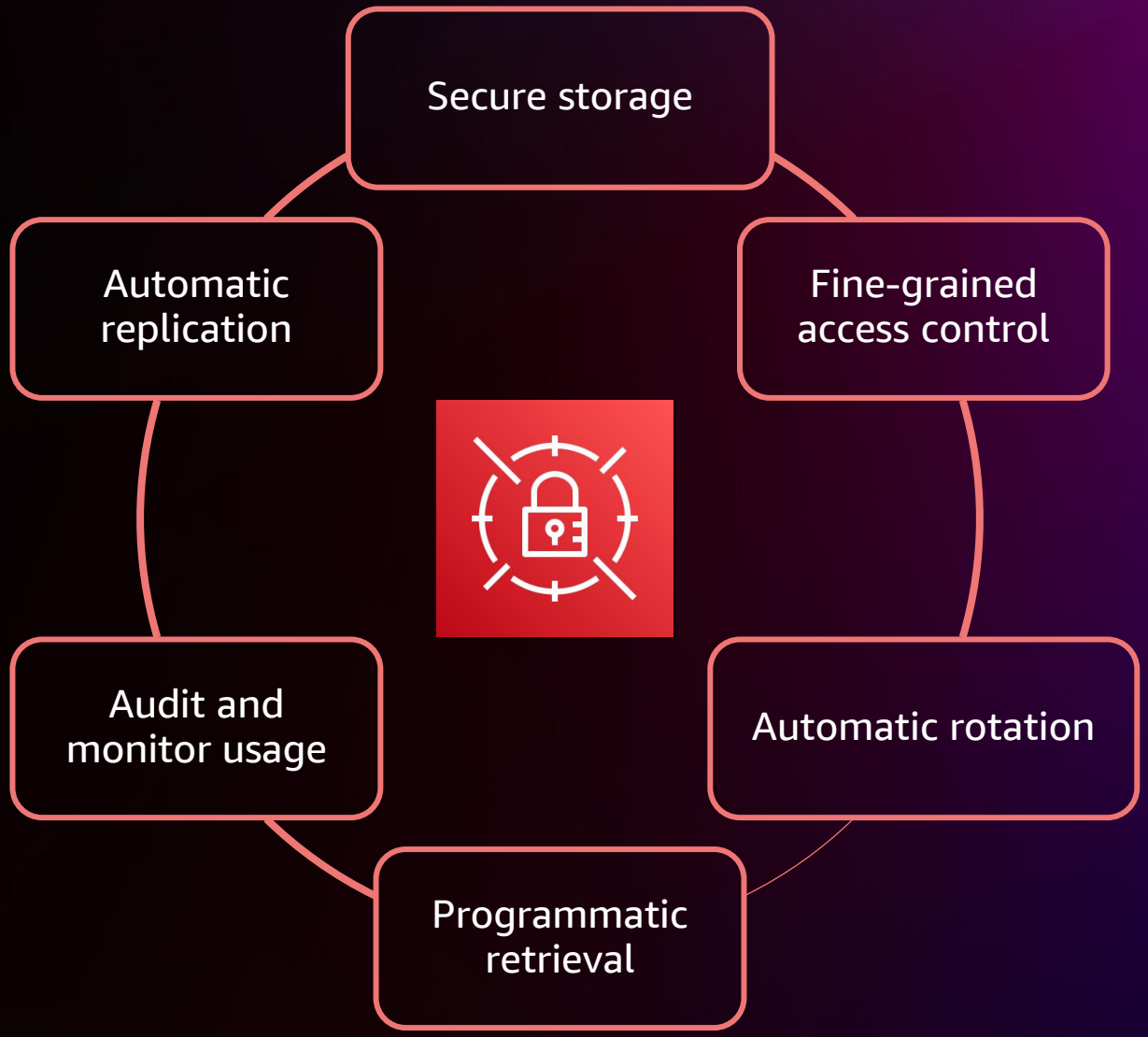
“If you want to keep a secret, you must also hide it from yourself.”

George Orwell

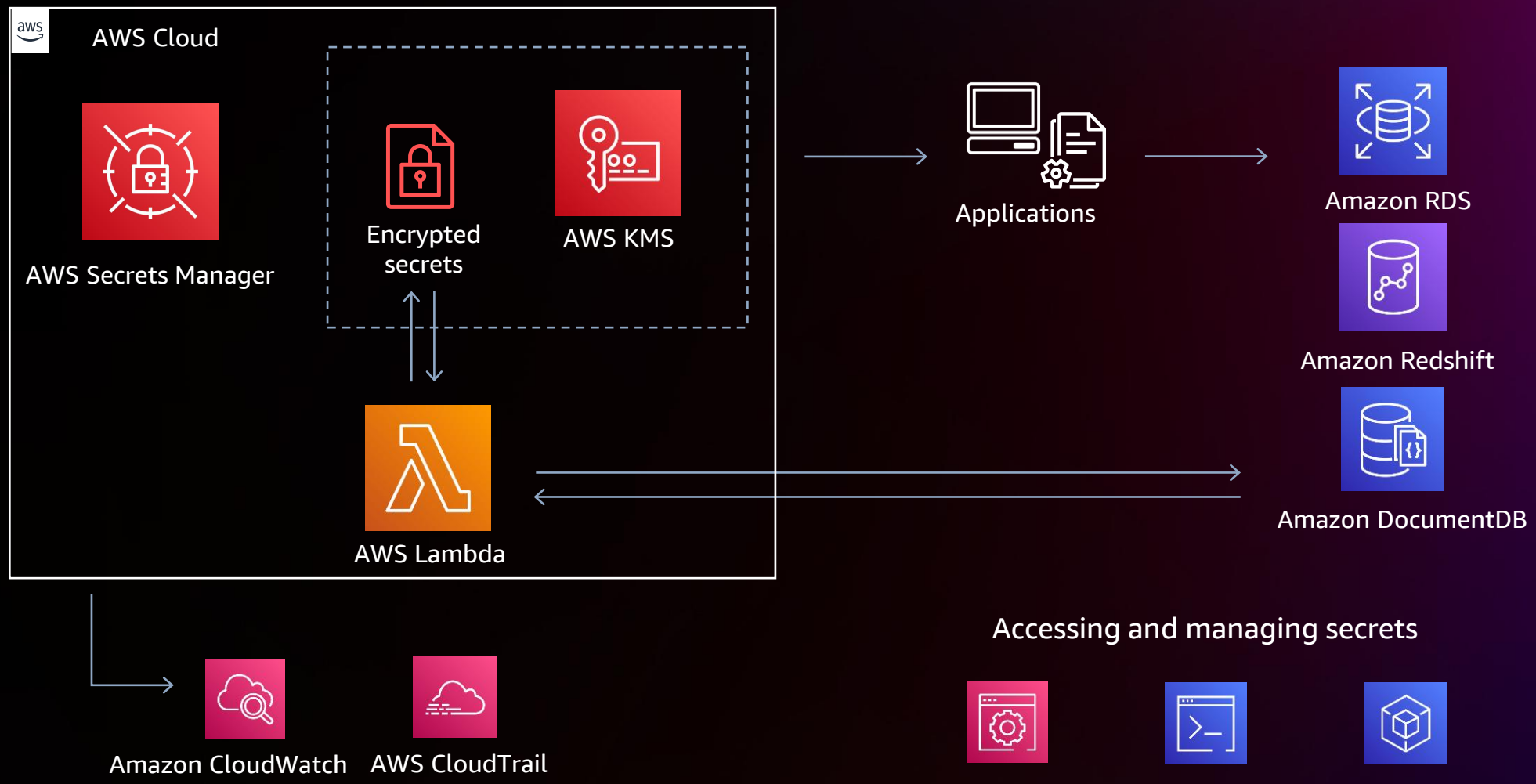
1984

Benefits

AWS Secrets Manager enables customers to **manage, retrieve, and rotate** database credentials, API keys, and other **secrets** throughout their **lifecycle**

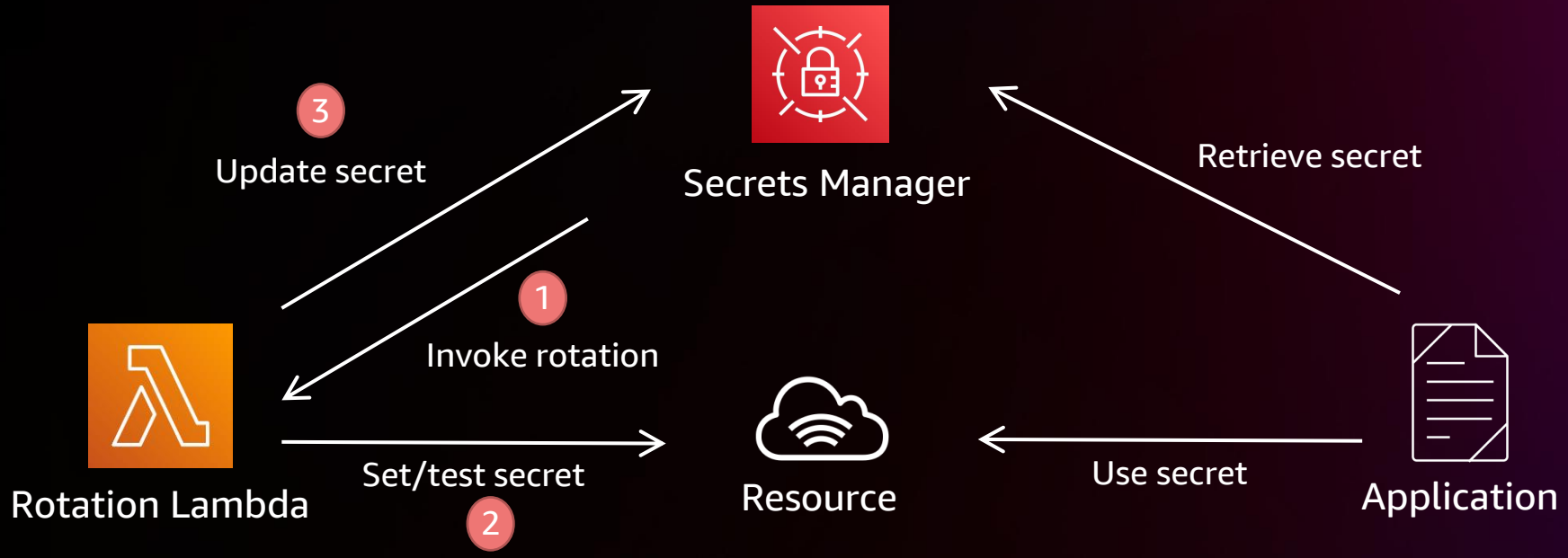


How Secrets Manager works



How Secrets Manager rotation works

Rotation Lambda functions coordinate secret rotation with resources



AWS Secrets Manager provides rotation functions for common resource types and a template to help create custom rotations functions

Secrets Manager use cases

Use case 1: Generic credentials

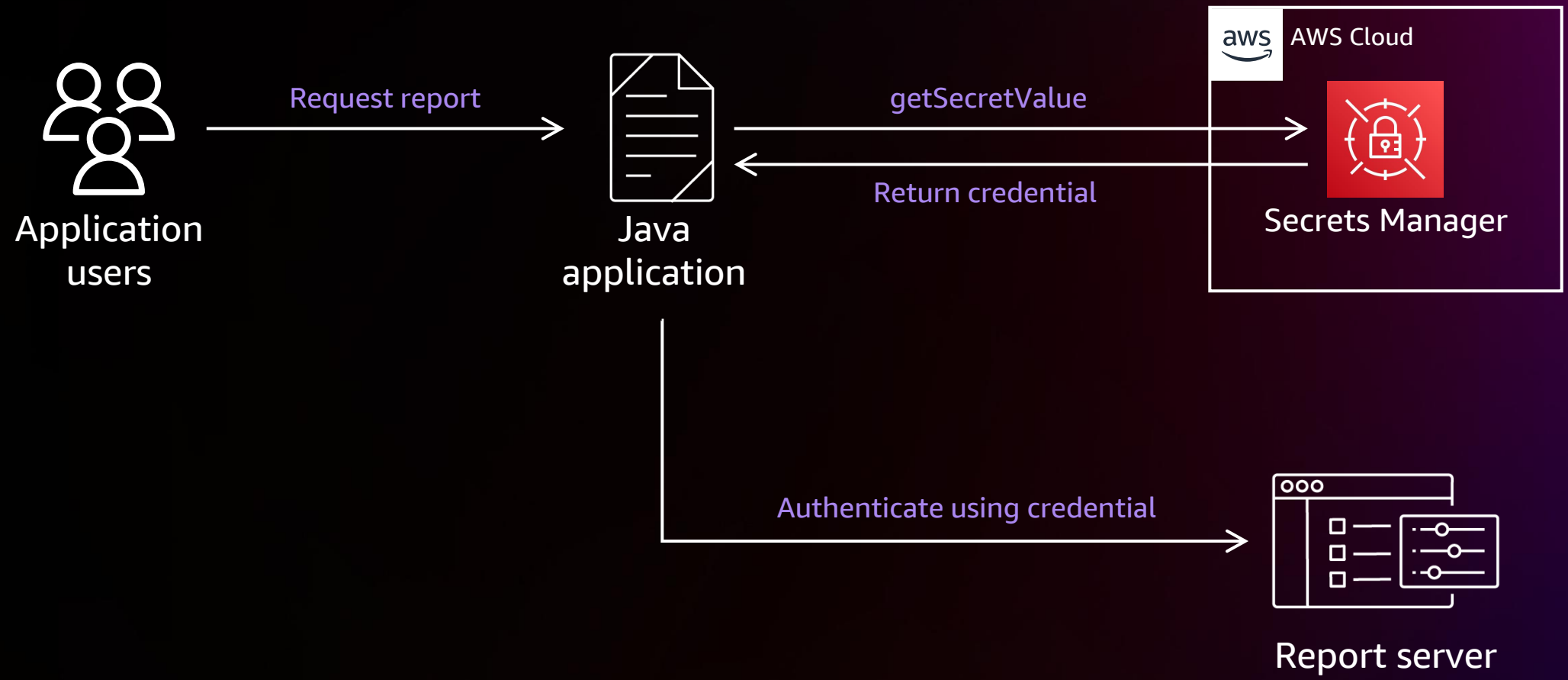
Overview

- Users in an enterprise use a Java application that front-ends a report generation service
- The report generation service requires a **service credential** to be used by the Java application – this service credential is provisioned by the enterprise identity system
- The service credential is stored in Secrets Manager and is queried at runtime by the Java application

Benefits

- Developer is **less likely to hardcode** the service credential into the Java application
- Service credential does not need to be stored on disk or in a configuration file
- If the service credential changes, only a change to the entry in Secrets Manager is required

Generic credential use case



Use case 2: API keys

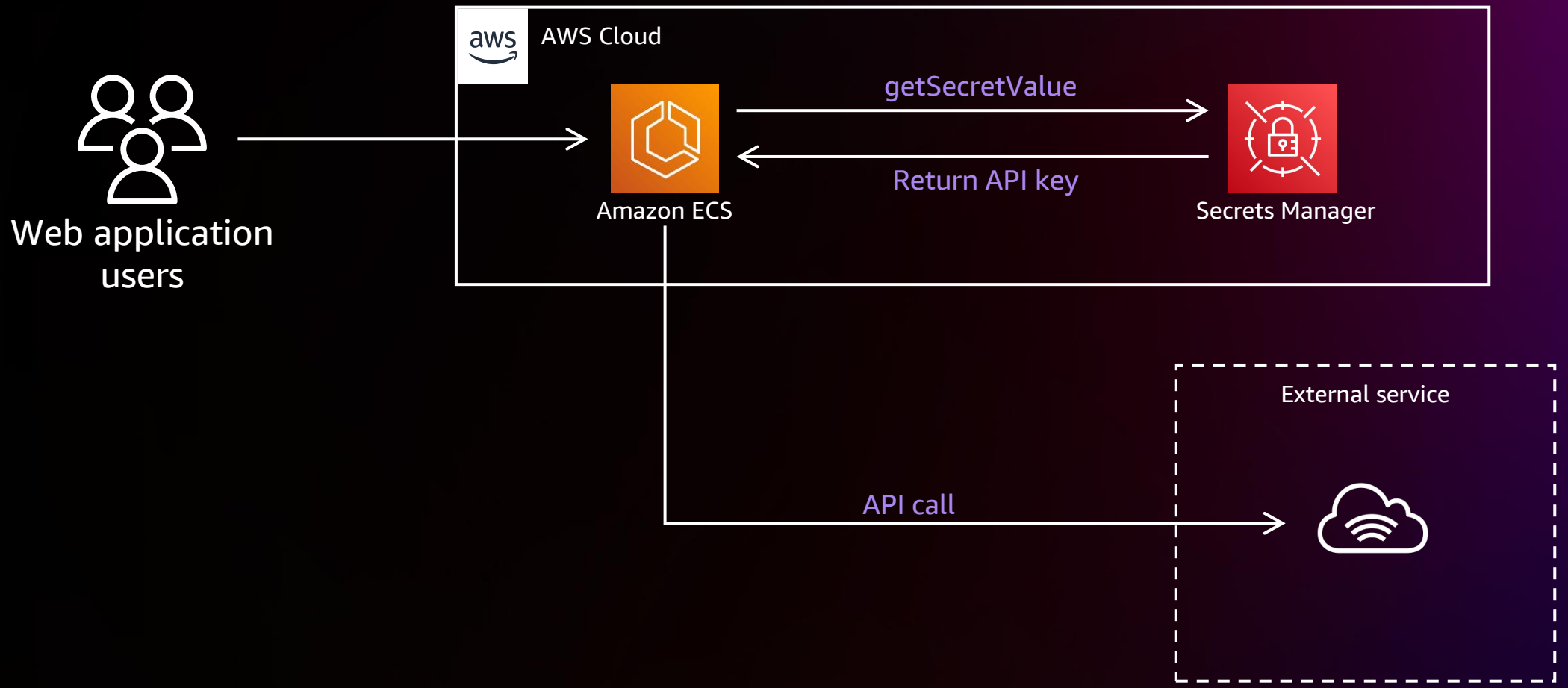
Overview

- A web application calls an external service with an API that is not generally open without prior authorization
  - An API key is not a form of authorization itself; it is **used as an identity verifier**
- Consists of 2 fields
  - Client ID – identifies who you are
  - Client secret – shared secret between you and the server

Benefits

- API key is never stored locally with the application on disk
- Developer is **less likely to commit the API key to source code**; API key is accessed via the Secrets Manager API
- If the API key changes, then the value in Secrets Manager is changed; **will not require a redeployment of the application** if that is required for configuration changes

API key use case



Use case 3: Private SSH keys

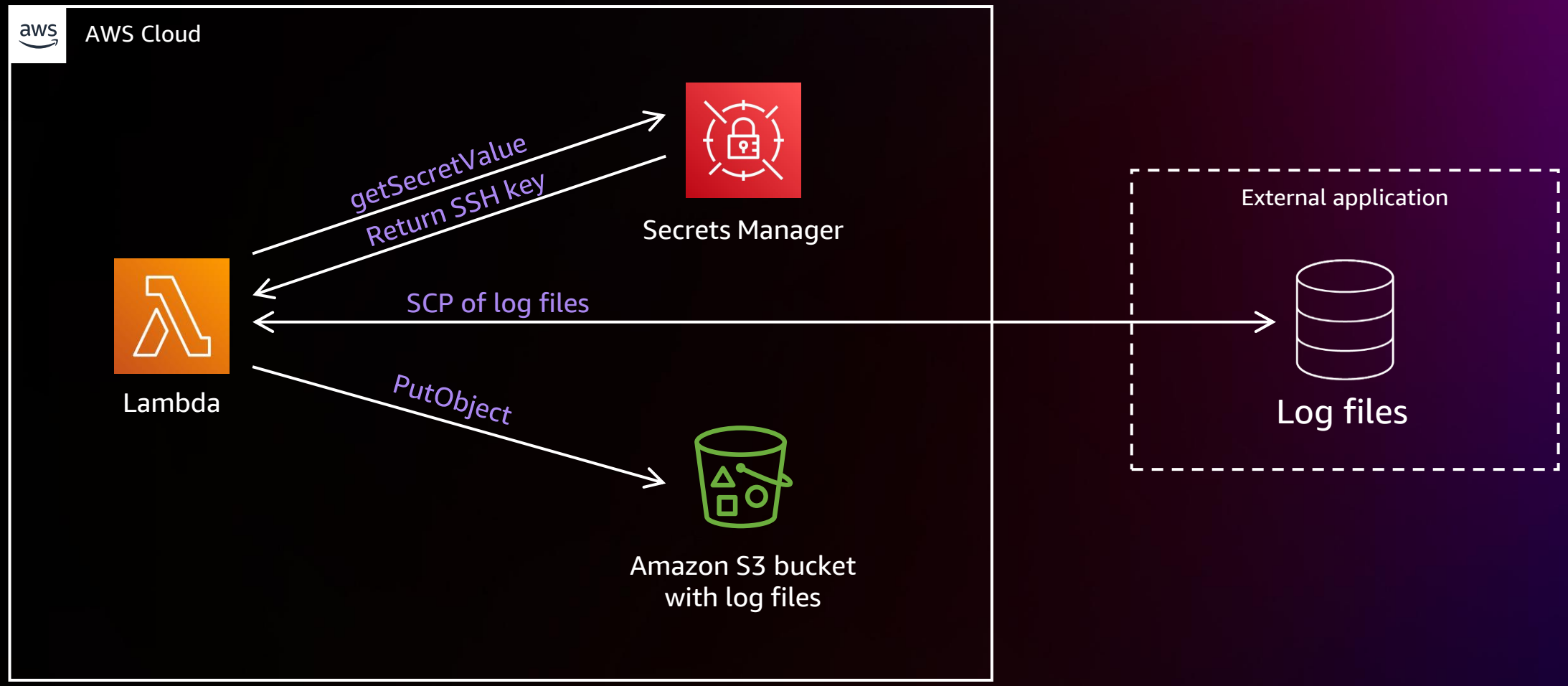
Overview

- A customer requires transfer of log files from an external service to their Amazon S3 bucket for log analysis (automated log retrieval)
- The customer creates a Lambda function to initiate the transfer from the external service via SCP; the Lambda function needs an **SSH private key** to present to the external service
- Once instantiated, the Lambda function calls Secrets Manager to get the SSH private key before initiating the transfer

Benefits

- Private SSH key does not need to be bundled with the Lambda function
- If the SSH key is changed, then it can be changed in Secrets Manager without having to redeploy the Lambda function
- Use of the SSH key can be audited via CloudTrail

Private SSH key use case



Use case 4: Enable hybrid workloads

Overview

- A customer wants to use AWS to store and manage database credentials that can be accessed by resources **in AWS as well as on-prem devices** such as application servers or Kubernetes clusters
- Use the **same IAM roles and policies** you have configured for your AWS workloads to provide access to AWS resources

Benefits

- **Eliminate use of long-lived credentials** to access AWS services such as AWS Private CA or AWS Secrets Manager
- Improve integration of hybrid workloads by standardizing on AWS security services
- Migrate from products with expensive licensing fees in favor of pay-as-you-go services from AWS
- **Reduce costs** by eliminating expensive certificate authorities and hardware security modules on-premises

IAM Roles Anywhere

EXTENDS THE USE OF IAM ROLES TO WORKLOADS OUTSIDE OF AWS

IAM Roles Anywhere

Uses the same IAM roles and policies you have already configured for your AWS workloads to access AWS resources on-prem

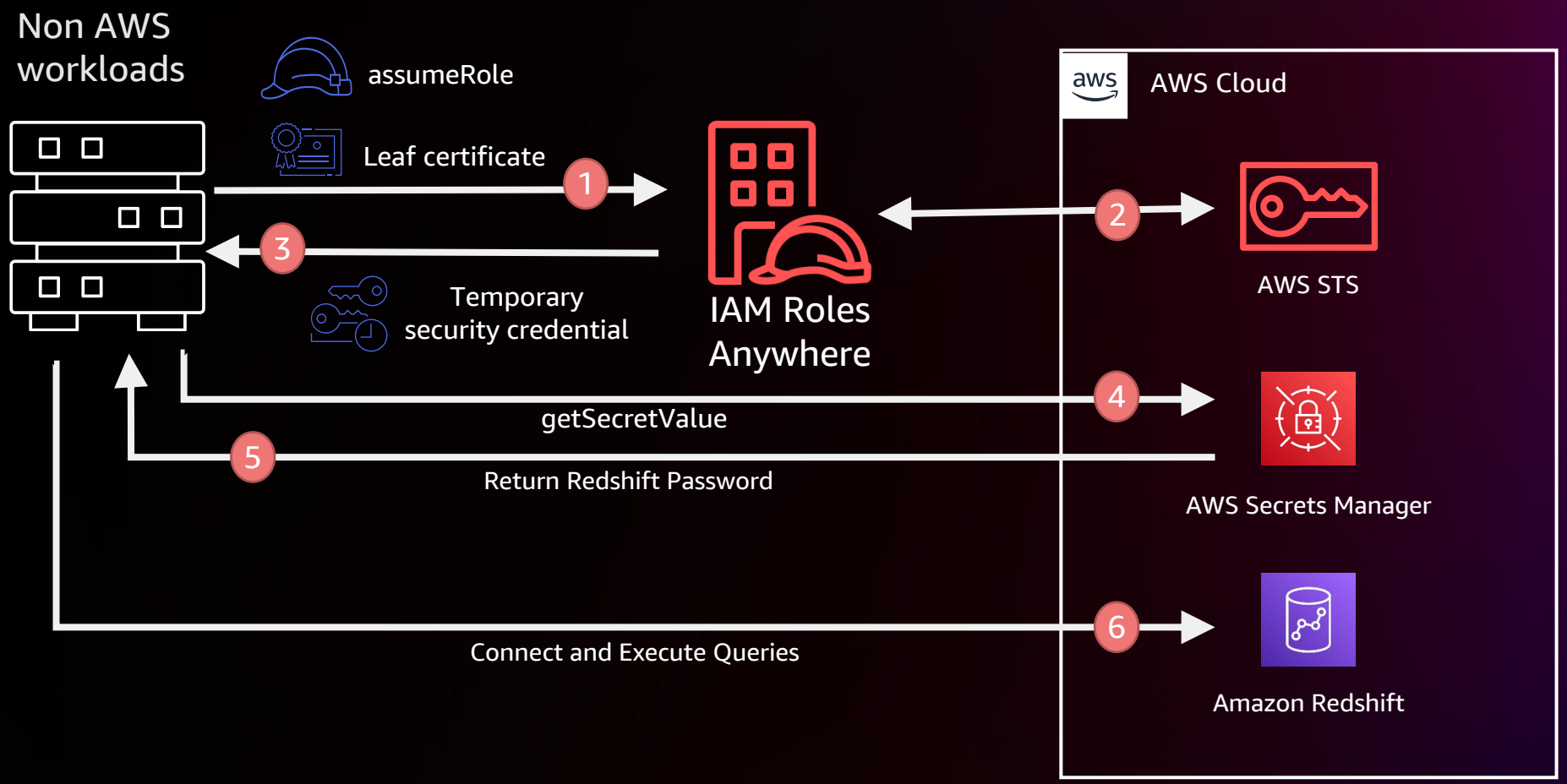
Obtain temporary AWS credentials and eliminate long-lived credentials

Use the same access controls, deployment pipelines, and testing processes across all your workloads

Simplify the migration of your workloads running outside of AWS

Hybrid workloads use case

Hybrid workloads use case



Use case 5: Access token for 3rd party services

Overview

- A customer wants to use AWS to store and manage access tokens for a third-party resource (such as GitHub)
- Customer creates a rotation function to automate the refresh of these access tokens utilizing the 4 hour rotation functionality in AWS Secrets Manager
- Customer uses a Lambda function that automates actions on the third-party resource using the access token

Benefits

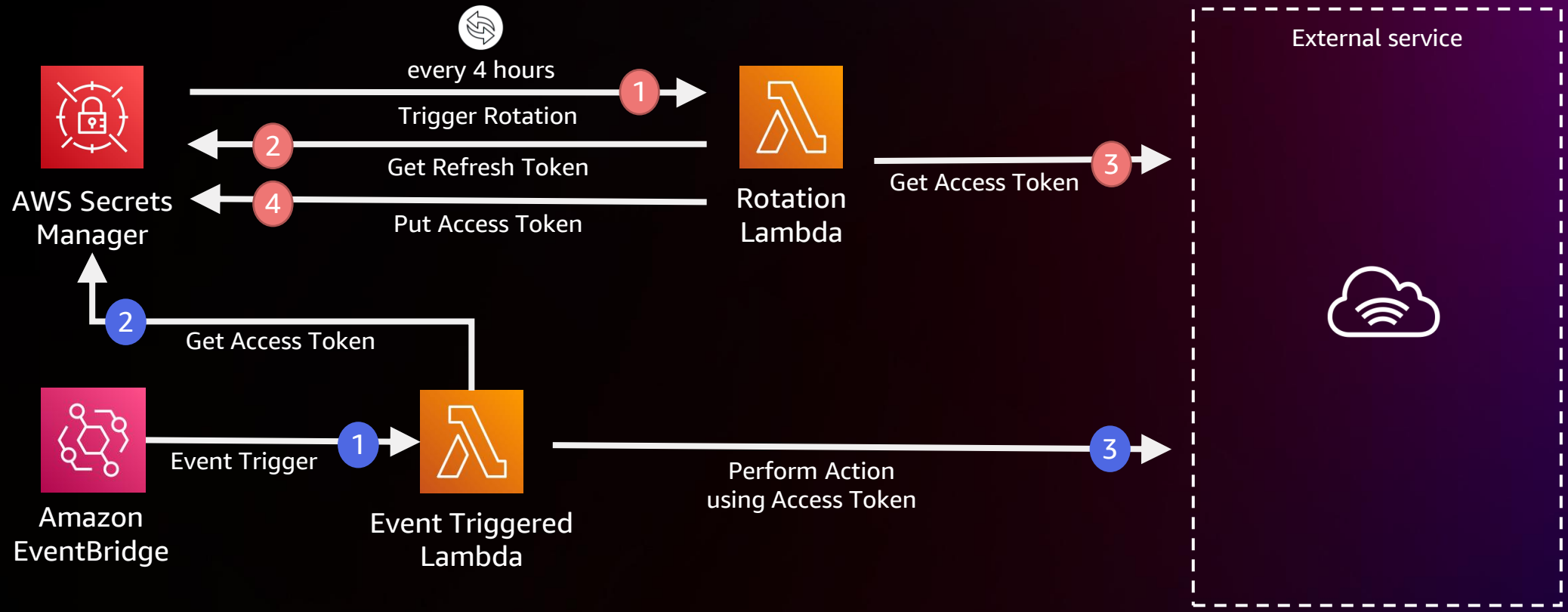
- Short-lived credentials limit ability to hardcode credentials
- Restrict and monitor access to longer term refresh tokens
- Tokens and other short lived credentials **limit the risk** of what can happen if a customer accidentally leaks a credential
- **Enables automated actions** to be triggered on resources that require access tokens

New feature enabling this use case

4 hour secret rotation

- Recently launched a new feature where secrets can now be rotated up to once every four hours using AWS Secrets Manager
- Developers can now manage longer-lived secrets and shorter-lived credentials through a single service, with the same managed rotation experience

Access token use case



Thank you!

Steven Emelander

emelande@amazon.com

linkedin.com/in/semelander/

Rushir Patel

rushirp@amazon.com

linkedin.com/in/rushirpatel/

Please complete the session survey in the **mobile app**