Secrets Manager Builder Sessions

Agenda

- Introduction to Secrets Manager
- Overview of Secrets Manager Builder Sessions
- RDS and Fargate Builder Session

Introduction to AWS Secrets Manager

AWS Secrets Manager - Overview

- Allows you to manage, retrieve, and rotate credentials.
- Can assist you in meeting key security controls associated with credential management.
- Includes native support for RDS PostgreSQL, MySQL, and Aurora.
- Additional credential sources can be rotated via AWS Lambda functions.
- Keeps track of different password versions with staging labels:
  - AWSCURRENT – current version of secret
  - AWSPENDING – new version of secret created during a rotation
  - AWSPREVIOUS – immediately previous version of a secret

What is a secret?

- Definition: Something that is meant to be kept unknown or unseen by others.
- In our context we'll limit our consideration of secrets to those related to securing information
- Many different types:
  - Authenticators
    - Passwords
    - API keys
  - Encryption keys
    - Symmetric
    - Asymmetric
  - Etc…

aws

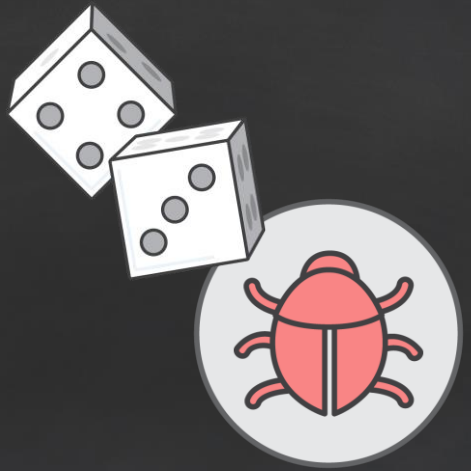
But what is the most important aspect of a secret?

- Definition: Something that is meant to be kept unseen by others.

- The problem:
  - How do you keep something unknown or unseen if it has to be shared in order to be used?

What are the challenges?

Too many humans with unnecessary access to secrets



Unreliable rotation processes

What do you need to do?

Connect to databases, APIs, and other resources, using the secrets that existing resources require.

Rotate secrets regularly without breaking stuff.

Maintain control and visibility over where, how, and by whom secrets are used.

aws

Typical Use Cases

Connect to database from application code

- DBA loads application specific database credentials into AWS Secrets Manager.
- DevOps engineer deploys application with an attached AWS IAM role.
- Application bootstrapping calls Secrets Manager using permissions provided by the IAM role, retrieves credentials, and connects to the database.

Typical Use Cases

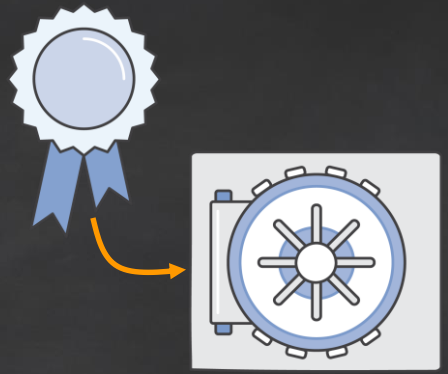
Rotate database credentials used by application code without interruption

- Secrets Manager creates a new credential with equivalent permissions.
- The new credential is promoted and returned via subsequent Secrets Manager API calls.
- Secrets Manager safely disables the original credential.

But won’t my code break if I change the secret while it’s running?

aws

Two scenarios:



- If your application supports only one set of credentials, your code needs to handle retry logic.
- If your code supports more than one set of credentials, use two sets of credentials and stagger the rotation.

Encryption

All secrets protected at-rest and in-transit

At-rest

- Secrets encrypted at rest using AWS Key Management Service (KMS).
- Choose your desired Customer Master Key (CMK) or AWS managed default encryption key.

In-transit

- Secrets encrypted in transit using Transport Layer Security (TLS).
- All API calls authenticated by SigV4 verification.

aws

Secrets Manager – Rotation Methodology – Part 1 of 3

1. The rotation function contacts the secured service's authentication system and creates a new set of credentials to access the database. The credentials typically consist of a user name, a password, and connection details, but can vary from system to system.

2. Secrets Manager stores these new credentials as the secret text in a new version of the secret that gets the AWSPENDING staging label attached.

Version	Current	Previous	Future
Values	MyFirstSecret	-	MySecondSecret
Labels	AWSCURRENT	-	AWSPENDING

Secrets Manager – Rotation Methodology – Part 2 of 3

1. The rotation function then tests the AWSPENDING version of the secret to ensure that the credentials work, and that they grant the required level of access to the secured service.

2. If the tests succeed, the rotation function then moves the label AWSCURRENT to the new version to mark it as the "default" version. The AWSPENDING label remains with the new version as well. The function also assigns the label AWSPREVIOUS to the old version, which marks it as the "last known good" version.

Version	Current	Previous	Future
Values	MySecondSecret	MyFirstSecret	-
Labels	AWSCURRENT AWSPENDING	AWSPREVIOUS	-

Secrets Manager – Rotation Methodology – Part 3 of 3

4. If a prior version had the AWSPREVIOUS staging label then that label has been removed.

Overview of Secrets Manager Builder Sessions

Secrets Manager Builder Sessions - Overview

- Each Builder Session demonstrates the use of a AWS Secrets Manager in different scenarios.
- All Builder Sessions can be run independently.
- Each Builder Session takes approximately one hour to run.
- Main Builder Sessions instructions are at:

https://secrets-manager.awssecworkshops.com

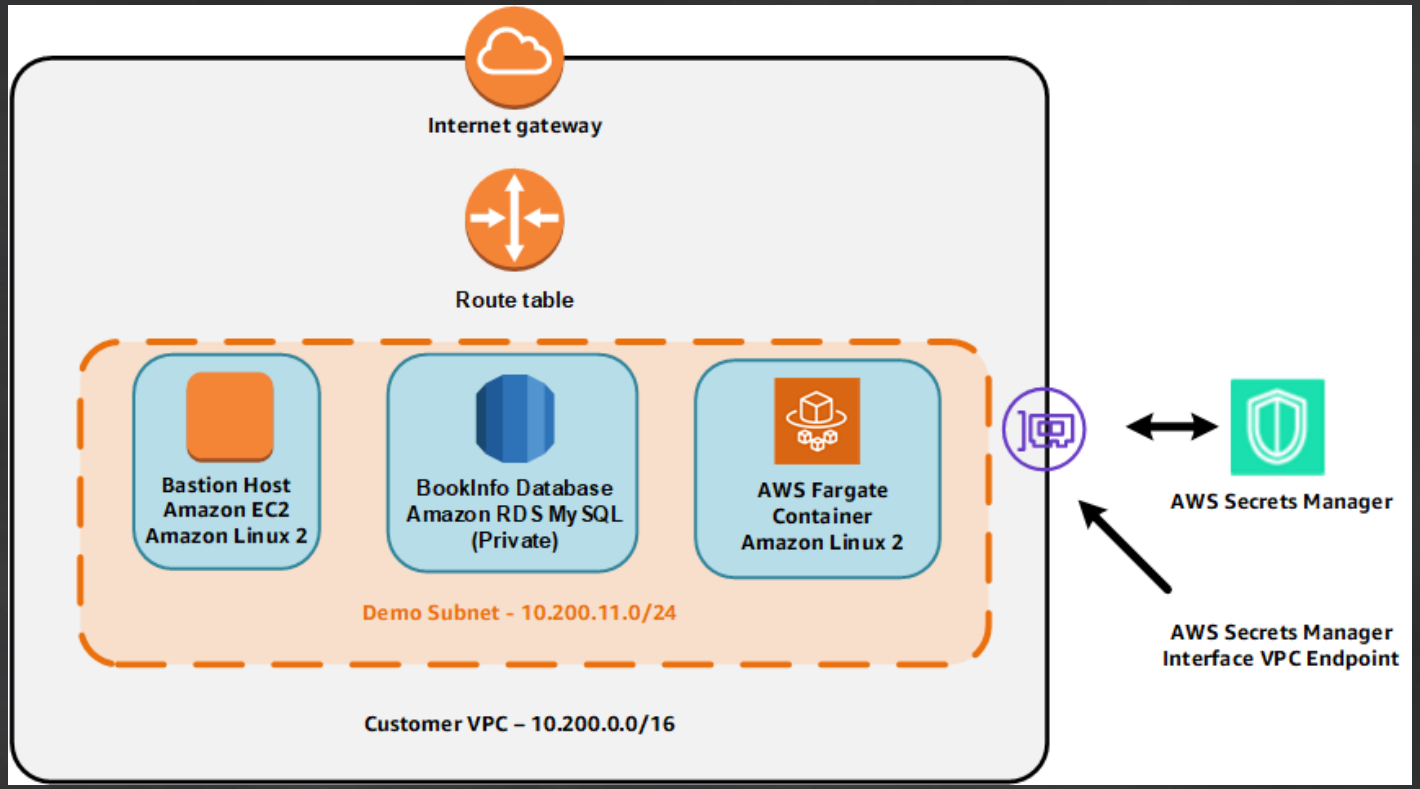
aws

Secrets Manager Builder Sessions - List

- Using Secrets Manager with RDS and Fargate
- More Builder Sessions will be added in the future.

RDS & Fargate Builder Session

RDS & Fargate Builder Session - Architecture



RDS & Fargate Builder Session

There are two phases to this Builder Session

1. Amazon RDS for MySQL phase
2. AWS Fargate phase

RDS & Fargate Builder Session – RDS Phase

In this phase, you will do the following:

1. Run a shell script to access a data base with a hard-coded secret.
2. Run a shell script to access the same data base with AWS Secrets Manager.
3. Run a shell script that attempts to access a data base with the original hard-coded secret. It will fail since the secret has changed.
4. Run a shell script to access the same data base with AWS Secrets Manager. It will succeed!

aws

RDS & Fargate Builder Session – Working with JSON

When you retrieve a secret for RDS, you get a JSON string:

```
{
   "username":”ABCDEFGH",
   "password":“IJKLMNOPQ",
   "engine":"mysql",
   "host":“aaaaaaaa.us-east-1.rds.amazonaws.com",
   "port":3306,
   "dbname":“mydbname",
   "dbInstanceIdentifier":“aaaaaaaa
 }
```

RDS & Fargate Builder Session – Shell Scripts + RDS

When the shell script fetches the value of an Amazon RDS Secret using the AWS CLI, it parses the JSON string using the jq command and storing the values in environment variables so you can see what's going on.

This is ***not*** a best practice since the secrets could become visible in the command line history. For best practices within shell scripts and programs go to:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/best-practices.html

RDS & Fargate Builder Sesion – Fargate Phase

You can pass secrets to AWS Fargate using the Task Definition capability of AWS Elastic Container Service (ECS).

Modify the JSON configuration of the Task Definition to contain the secret name and an environment variable.

```
"secrets": [
    {
        "valueFrom": "arn:aws:secretsmanager:us-east-1:2[redacted]0:secret:SECRETNAME",
        "name": "TASKDEF_SECRET"
    }
],
```


aws

RDS & Fargate Builder Session – Fargate Phase

1. When the Fargate task is started, the Task Definition retrieves the secret string and passes it to the environment variable you specify.

2. The task is responsible for parsing the string.

3. The task creates a script in /etc/profile.d that propagates the value to the login shell. More information is available in the Builder Session site.

aws

RDS & Fargate Builder Session – Get started!

Here is the workshop link:

https://secrets-manager.awssecworkshops.com

Select the Builder Session:

Using AWS Secrets Manager with Amazon RDS and AWS Fargate

Questions?