AWS Secrets Manager

AWS Secrets Manager

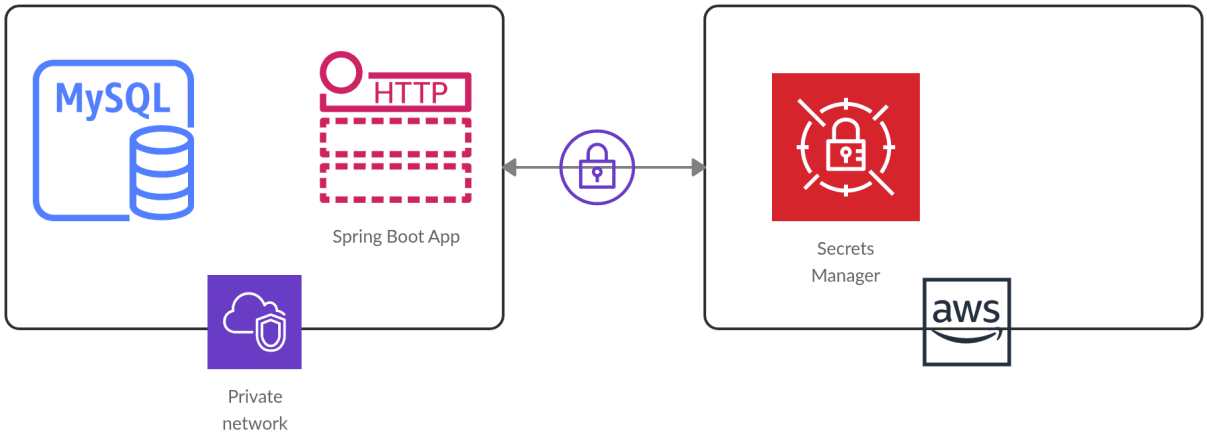
- It is a service provided by AWS to store secrets i.e. **passwords, credentials, third-party keys**, or any such confidential information.
- Secrets Manager allows you to **store** and **manage** access to these credentials.
- It allows you to easily **change or rotate** your credentials, thereby avoiding any code or config changes.
- Leverage Secrets Manager to store your credentials **instead of hard-coding** them in your code or config files.
- It allows you to replace hardcoded credentials in your code with an **API call** Secrets Manager to **retrieve** the secrets **programmatically.**
- It **encrypts** the protected text of a secret by using **AWS Key Management System**

AWS Secrets Manager

Who Can Use Secrets Manager

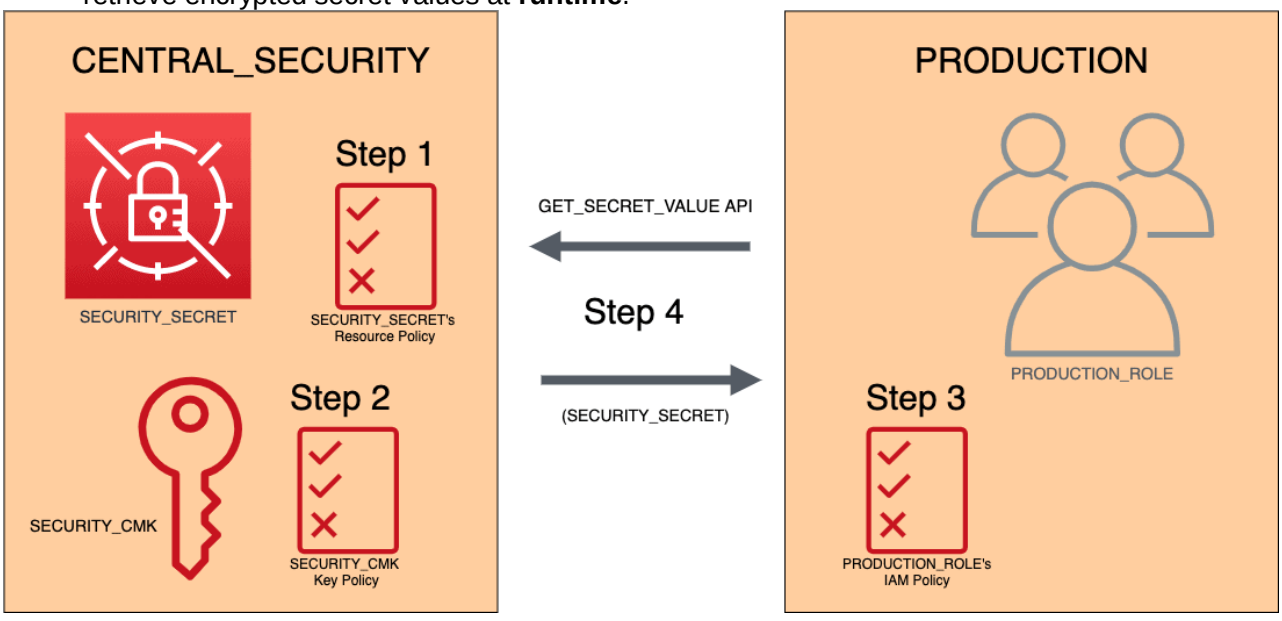
Mainly the users of Secrets Manager can have one of the below-mentioned roles:

- **IT Admins:** If you are an IT Admin who is responsible for **storing** and **managing secrets**, you can use Secrets Manager to perform all these tasks from one location
- **Security Admin:** If you are Security Admin who is responsible for ensuring that your organization follows regulatory and compliance requirements, well Secrets Manager can be used to **audit** and **monitor** the use of your secrets and also that the secrets are **rotated** as frequently as required.
- **Developer:** If you are a developer, you can onboard the Secrets Manager so that you don't have to worry about **managing secrets** as AWS will do that for you. This way you can avoid dealing with secrets in the applications.



Features

- **Rotate Secrets Safely:** Without worrying about updating or deploying the code, you can easily **rotate secrets** using Secrets Manager in order to ensure that your security and compliance requirements are met.
- **Manage Access with fine-grained Policies:** Using certain Identity and Access Management **(IAM**) policies, access to the **secrets can be managed**. For example, a policy can be created that allows developers to get the secrets when they are being used for development purposes.
- **Secure and audit secrets centrally:** By encrypting the secrets with encryption keys you can **secure** your secrets as well. This can be easily achieved using the Amazon Key Management Service **(KMS)** used to encrypt data.
- **Pay as you go:** Secrets Manager is a pay-as-you-go model. You will only be charged for the number of **secrets managed** by the Secrets Manager and the number of Secrets Manager **API calls** made.
- **Retrieve Secrets programmatically:** With Secrets Manager, you can programmatically retrieve encrypted secret values at **runtime**.



Basic Secrets Manager Scenario

Now let's get to know how Secrets Manager stores the credentials which can then be used in an application to access the database. The diagram below represents this scenario.

- The database admin first **provisions a database**. It can be a personnel database, a database for a project, or anything which has a set of credentials (Username and Password). The admin also configures those credentials with the permissions required for the application to access the Personnel database.
- After the database has been provisioned. the database admin **creates a secret in AWS Secrets Manager** and stores the credentials as a secret and also provides a name to that secret. Then, Secrets Manager encrypts and stores the credentials within the secret as the protected secret text.
- When the application accesses the database, the application makes an **API call** to the Secrets Manager to fetch the secret information.
- After AWS Secrets Manager is called, it **retrieves the secret, decrypts the protected secret text**, and returns the secret to the client application over a secured channel using TLS protocol.
- The client application resolves the information which includes credentials, connection string, and any other information from the response, and then uses that information to access the database server.

Storing Secrets Using AWS CLI

For storing secrets using **AWS CLI,** you need to first ensure that AWS CLI is configured and you have an IAM user that has the privileges to access the AWS Secrets Manager. Once these prerequisites have been completed then only can store secrets using AWS CLI. Now let's how we can achieve this

1.) Execute the following commands as shown below in the image.

```
aws secretsmanager create-secret \\
    --name "avApiKey" \\
    --secret-string '{"api_key": "super-secret"}'
```

2.) Now to check if the above commands worked and the secret was created, execute the command shown below.

```
aws secretsmanager list-secrets
```

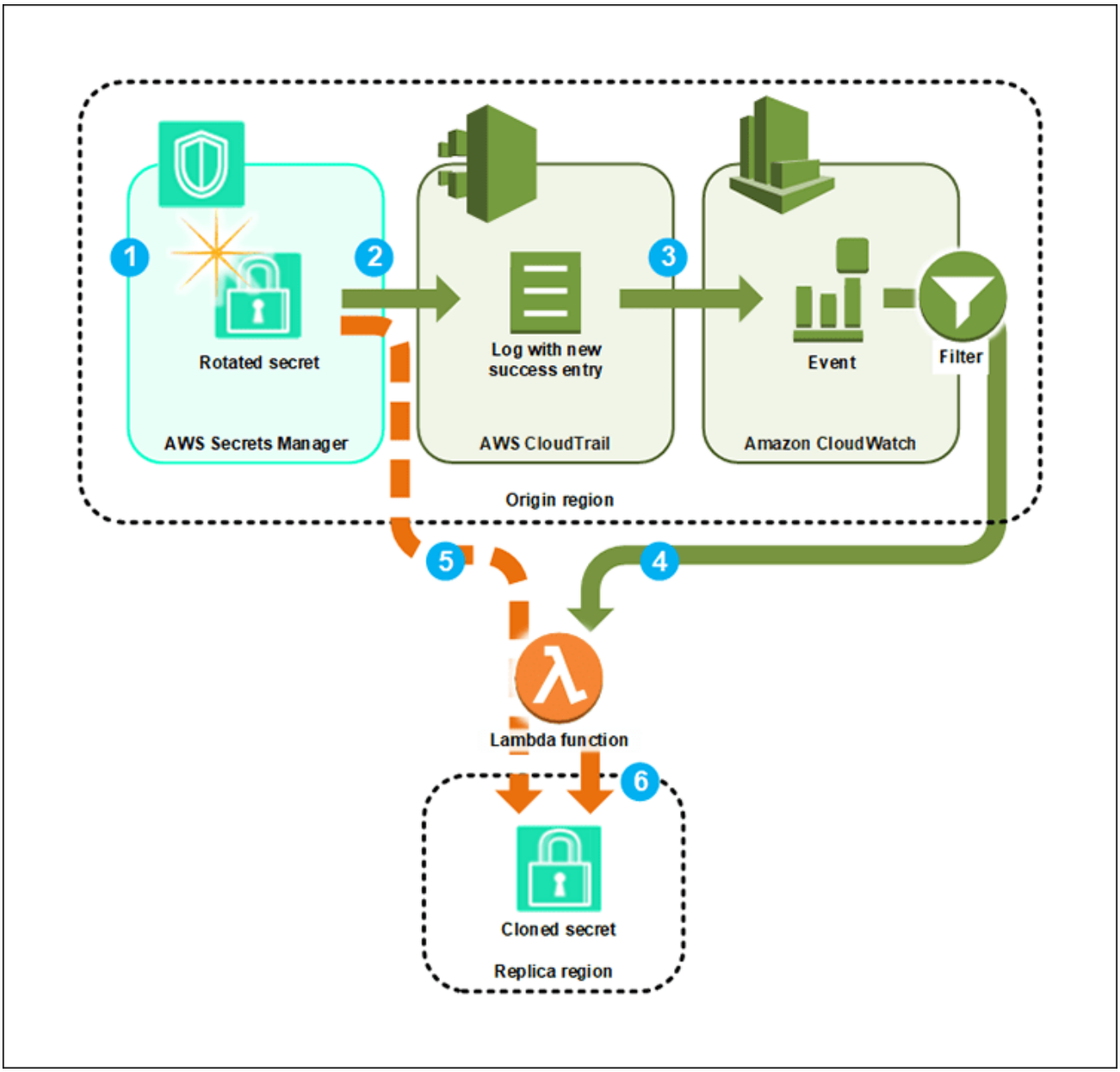
3.) Further, if you want to update the credentials (like you changed the password) that can also be done using the following commands.

```
aws secretsmanager update-secret \\
    --secret-id "avApiKey" \\
    --secret-string '{"api_key": "secret_v2"}'
```

Replication Of Secrets Across AWS Regions

By replicating secrets across AWS Regions, you can minimize the amount of time needed to get back up and running in production in a disaster recovery situation by making sure that the credentials are stored securely in a replica region.

This involves a combination of **AWS Secrets Manager, AWS CloudTrail, Amazon CloudWatch Events, and AWS Lambda.** First, you create a secret in Secrets Manager that stores your RDS database credentials. AWS KMS encrypts this secret and Lambda will automate the replication of the secret's value in your AWS Region by performing a **PUT** operation on a secret by the same name in the same AWS Region as your read-replica. By doing this, your RDS database credentials are always in sync for recovery in case of any disaster.



Secrets Manager Pricing

As AWS Secrets Manager is a pay-as-you-go service, so you pay based on the number of secrets stored and API calls made. There are no upfront costs or long-term contracts. There is a **30 day free trial period** where you can try AWS Secrets manager at no additional cost and the free trial will start when you store your first secret. After the free trial is over, you will be charged **$0.40 per secret per month** and **$0.05 per 10,000 API calls.** Please click **here** to get more insights regarding its pricing.