

# Using AWS Secrets Manager to protect application secrets

Draft 1, v20191009

**USING AWS SECRETS MANAGER TO PROTECT APPLICATION SECRETS..... 1**

INTRODUCTION ..... 2

SCOPE..... 2

OBJECTIVE ..... 2

CREATING A SECRET ..... 2

REVIEWING A SECRET ..... 2

ACCESSING THE SECRET PROGRAMMATICALLY ..... 3

ROTATING SECRET VALUES ..... 3

MANAGE ACCESS WITH FINE-GRAINED IAM POLICIES ..... 3

## Introduction

This document describes the use of AWS Secrets Manager to protect application secrets. Using a vault to store your secrets ensures the information security required to maintain the security posture and compliance.

## Scope

The discussion is limited to creating user keys, example API keys in AWS Secrets Manager and then programmatically accessing it with an API call in the client application. There are other useful security features AWS Secrets Manager provides, example managing and rotating RDS credentials which we will not discuss here.

## Objective

Application development teams need a vault which can store a secret that can later be retrieved programmatically. In the absence of this tool, hardcore sensitive information could be stored in plain text within source code, and worse be checked into a source code repository. In addition, Secrets Manager enables you to control access to secrets using fine-grained permissions and audit secret rotation centrally.

## Creating a secret

A secret is a key/value pair stored in AWS Secrets Manager. The value is encrypted at rest when stored using an encryption key. The following illustrates the AWS Secrets Manager Console screen to create a secret.

☐ Credentials for other database

☒ Other type of secrets (e.g., API key)

Specify the key/value pairs to be stored in this secret [info](#)

Secret key/value	Plaintext
/githubextract/dev/mysecret	testmysecret

[+ Add row](#)

## Reviewing a secret

Once the secret has been created, it can be viewed and edited from the AWS Secrets Manager Console.

Secret value <a href="#">info</a>		Close	Edit
Retrieve and view the secret value.			
Secret key/value	Plaintext		
Secret Key	Secret Value		
mysecret	testmysecret		

## Accessing the secret programmatically

On creating a secret, the AWS Secrets Manager Console generates client API code sample that illustrates how to retrieve the secret in your application. You can choose one the client SDKs from Java, Javascript, C#, Python3, Ruby or Go. In our example, we use a nodejs program uses the Javascript client. The resulting programmatic access looks something like this

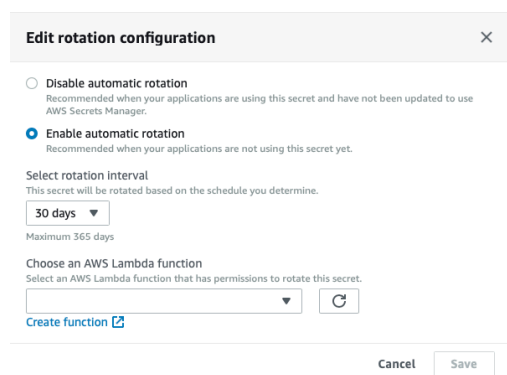
```
MacBook-Pro:awssecretsmanager viyer$ node index.js
{"mysecret":"envoi89-bear"}
MacBook-Pro:awssecretsmanager viyer$
```

The benefit of using the generated code is that it is already configured for using the secret within your client program.

```
// Load the AWS SDK
var AWS = require('aws-sdk'),
    region = "us-east-1",
    secretName = "/githubextract/dev/mysecret",
    secret,
    decodedBinarySecret;
```

## Rotating secret values

The initial secret value, as well as rotating the secret values in an interval schedule is possible to be managed by a Lambda function. Retrieving the secret from Secrets Manager ensures that developers and applications are using the latest version of your secrets.



The screenshot shows the 'Edit rotation configuration' dialog box. It has a title bar with a close button (X). Inside, there are two radio buttons: 'Disable automatic rotation' (unselected) and 'Enable automatic rotation' (selected). Below the radio buttons is a section titled 'Select rotation interval' with a dropdown menu set to '30 days' and a note 'Maximum 365 days'. Below that is a section titled 'Choose an AWS Lambda function' with a dropdown menu and a 'Create function' link. At the bottom are 'Cancel' and 'Save' buttons.

## Manage access with fine-grained IAM policies

With Secrets Manager, you can manage access to secrets using fine-grained AWS Identity and Access Management (IAM) policies and resource-based policies. For example, you can create a policy that enables developers to retrieve certain secrets only when they are used for the development environment. In the example above, the IAM policy could include `/githubextract/dev/*` or `/githubextract/prod/*` to control access to secrets in development or production, respectively.