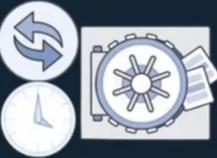


AWS secret manager

"Helps in managing, rotating (renewing) and retrieving database credentials, api keys, and other secrets with authorized access only. It also helps in auditing (when the secret was accessed, who is accessing the secret, etc)".

AWS Secrets Manager

Lifecycle management for secrets such as database credentials and API keys.



Rotate Secrets Safely



Manage access with fine-grained policies



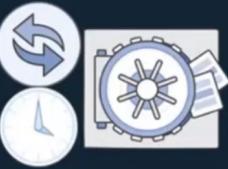
Secure and audit secrets centrally



Pay as you go

Features

1. Rotation



Rotate Secrets Safely

- Built-in integrations for rotating MySQL, PostgreSQL, and Amazon Aurora on RDS
- Extensible with Lambda
- Use versioning so that applications don't break when secrets are rotated

- Easy to rotate other secrets by writing your own lambda code for rotation.
- Versioning of secrets is also done so that applications do not break when secrets are rotated.

2. Fine-grained access



Fine-grained
access control

- IAM policies
- Tag-based access control and hierarchical names for scalability
- Resource-based policies for cross-account access

3. Security, Monitoring, Auditing



Secure, audit, and
monitor

- Encrypted by default using encryption keys owned by the customer
- Integrated with CloudTrail, CloudWatch. E.g., send a SNS notification when an administrator deletes a secret

4. Pricing



Pay as you go

- No annual license or up front cost
- \$0.40 per secret per month (pro-rated based on the number of hours)
- \$0.05 per 10,000 API calls

Note: The code to retrieve a secret in programming languages is provided by AWS at the time of creating the secret.

Sample code
View a code sample that illustrates how to retrieve the secret in your application.

Java | Javascript | C# | Python3

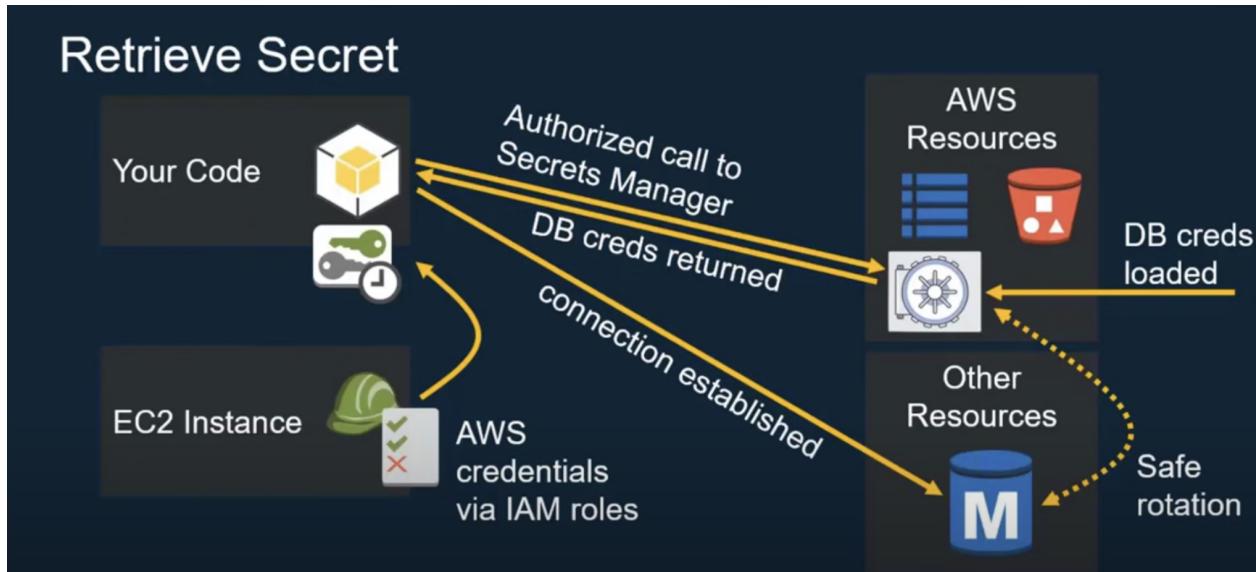
```
1 // Use this code snippet in your app.
2
3 // If you need more information about configurations or implementing the sample code, visit the
4 // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-samples.html#prerequisites
5
6 public static void getSecret() {
7
8     String secretName = "My_Test_Secret/MySQL";
9     String endpoint = "secretsmanager.us-east-2.amazonaws.com";
10    String region = "us-east-2";
11
12    AwsClientBuilder.EndpointConfiguration config = new AwsClientBuilder.EndpointConfiguration(
13        endpoint, region);
14    AWSSecretsManagerClientBuilder clientBuilder = AWSSecretsManagerClientBuilder.standard();
15    clientBuilder.endpointOverride(config);
16
17    AWSSecretsManagerClient client = clientBuilder.build();
18
19    GetSecretValueRequest request = new GetSecretValueRequest().withSecretId(secretName);
20
21    GetSecretValueResponse response = client.getSecretValue(request);
22
23    System.out.println("Secret value: " + response.getSecretString());
24}
```

[Download AWS SDK for Java](#)

Command to get a secret,

aws secretsmanager get-secret-value --secret-id <secret_name_here>

Flow for retrieving the secret



1. Give IAM permissions to your resource (where the application is running, EC2 in this case) to authenticate with an AWS secret manager.
2. Application will then request for secrets (db credentials in this case).
3. Applications can consume other AWS resources (RDS in this case) using those credentials.
4. If the rotation in the secret manager is enabled it will rotate/change the secrets with the resource (RDS in this case) for rotation.

Attaching policies to users for authorized access on secrets

This policy will allow user to perform get secret, and describe secret operations on the secrets having prefix "My_Test_Secrets/*"

The screenshot shows the AWS IAM Policies page. At the top, there are three buttons: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly'. Below these buttons is a note: 'Attach one or more existing policies directly to the users or create a new policy. Learn more'. There are two buttons at the bottom left: 'Create policy' and 'Refresh'. A search bar with the text 'demo' is also present. The main area displays a table with one result. The table has columns: Policy name, Type, Attachments, and Description. The 'Policy name' column shows 'demo'. The 'Type' column shows 'AWS Lambda function'. The 'Attachments' and 'Description' columns are empty. The 'Description' column contains a link: 'View policy details'. The table header includes a 'Showing 1 result' message.

	Policy name	Type	Attachments	Description
1	demo	AWS Lambda function		View policy details

Access control – using IAM policies

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": ["secretsmanager:GetSecretValue", "secretsmanager:DescribeSecret"],  
      "Resource": "arn:aws:secretsmanager:us-east-2:476697075236:secret:My_Test_Secret/*"  
    }  
  ]  
}
```

We can also use tags in our policies,

Access control – using Tags

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["secretsmanager:Describe*", "secretsmanager:GetSecretValue"],  
            "Resource": "*",  
            "Condition": {  
                "StringEqualsIgnoreCase": {  
                    "secretsmanager:ResourceTag/<TAG_KEY>": "<TAG_VALUE>"  
                }  
            }  
        }  
    ]  
}
```

Auditing, Monitoring

This can be done by integrating AWS CloudTrail with AWS Secret Manager (ASM). In CloudTrail, we can search for the events triggered on the secrets. We can see who tried to access the secrets, when, the request was denied or grant, etc

The screenshot shows the AWS CloudTrail console with the 'Event history' tab selected. The left sidebar includes links for CloudTrail, Dashboard, Event history (which is highlighted), and Trails.

The main area displays the 'Event history' details:

- Filter:** Event name (selected) - GetSecretValue
- Time range:** Select time range
- Event details:** A single event listed:
 - Event time: 2018-06-15, 08:57:33 AM
 - User name: admin_user
 - Event name: GetSecretValue
 - AWS access key: AKIAJNWHHCHXVCIX5YNA
 - AWS region: us-east-2
 - Error code: AccessDenied
 - Event ID: c6fe4553-97b1-497e-bd14-b4cbf42c07a2
 - Event name: GetSecretValue
 - Event source: secretsmanager.amazonaws.com
 - Event time: 2018-06-15, 08:57:33 AM
 - Request ID: d0031a9a-70b4-11e8-8860-4b1c19:
 - Source IP address: 52.95.4.16
 - User name: admin_user
- Resources Referenced (0)**