AWS Secrets Manager

Apurv Awasthi, Sr. Product Manager (AWS)

Overview

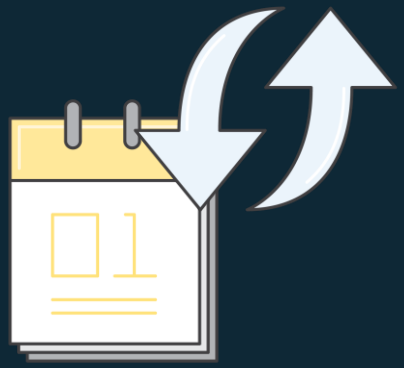
AWS Secrets Manager enables customers to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

- IT Admins: store and manage access to secrets securely and at scale
- Security Admins: audit and monitor the use of secrets, and rotate secrets without a risk of breaking applications
- Developers: avoid dealing with secrets in their applications

aws

What do customers want to do?

Use secrets within their applications to connect to databases, APIs, and other resources

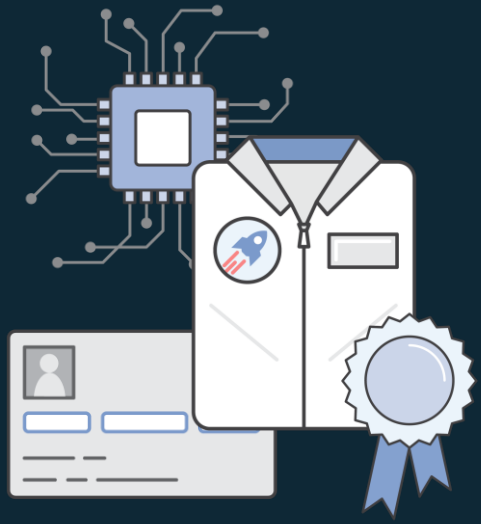


Rotate those secrets regularly

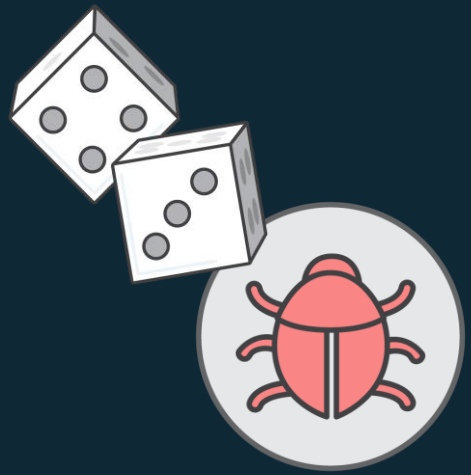
Specify and control where, how, and by whom secrets are used

aws

What challenges are they facing?



Existing solutions are complex to operate or too expensive



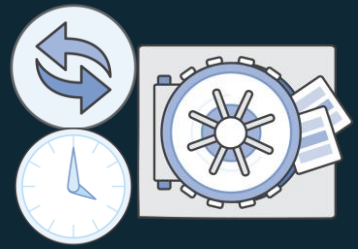
Unreliable rotation processes result in outages

Too many humans with unnecessary access to secrets

aws

AWS Secrets Manager

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



Rotate Secrets Safely

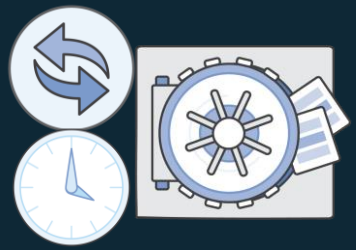
Manage access with fine-grained policies

Secure and audit secrets centrally

Pay as you go

aws

Key Features



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

Fine-grained access control

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

aws

Key Features

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

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

aws

Demonstration 1
Store and retrieve SSH Key

Typical Use Cases

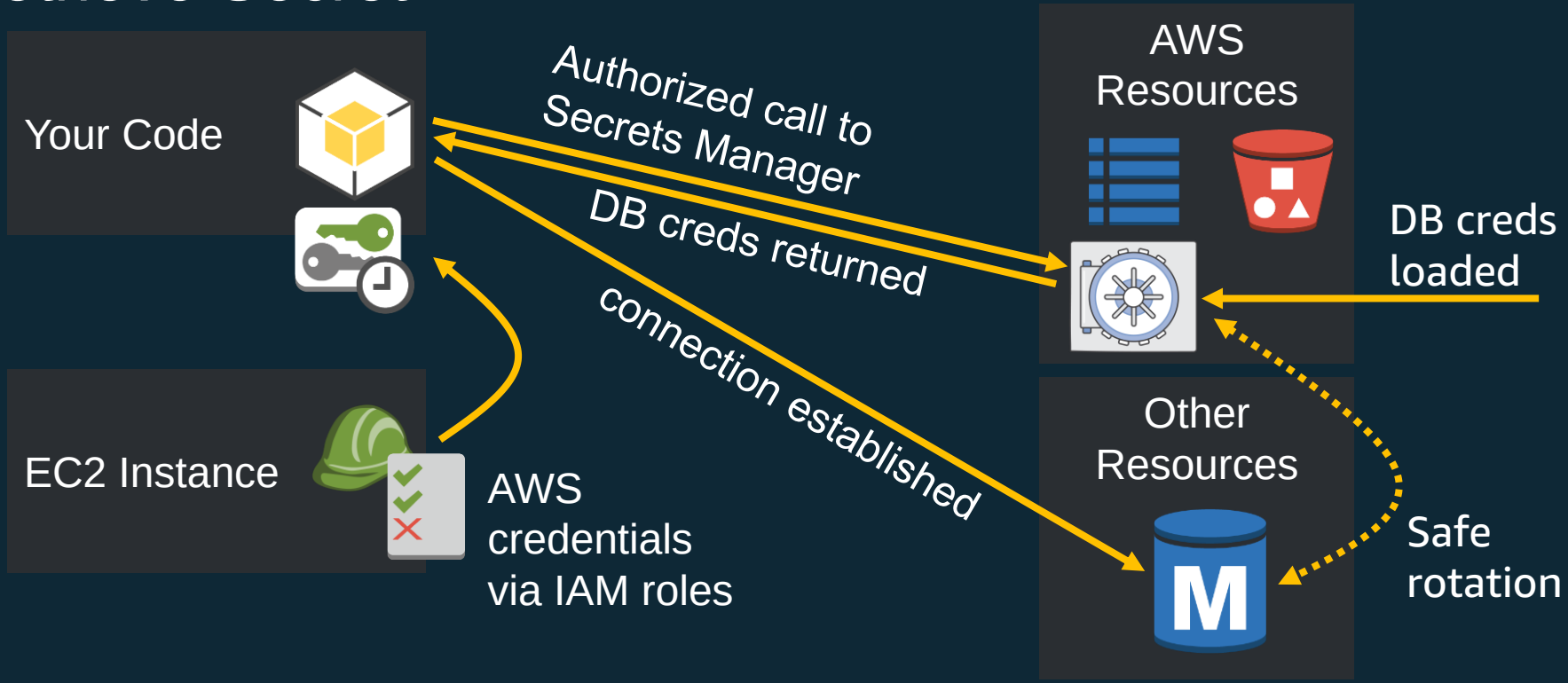
Typical use cases – Use secret within application

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.

aws

Retrieve Secret

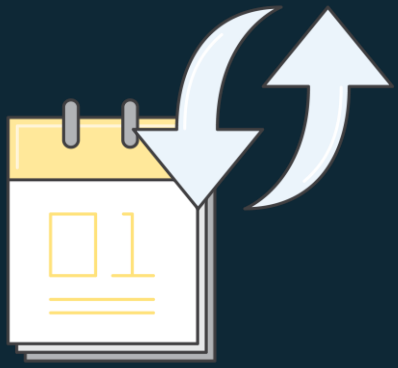


Combo provides your apps a reliable, secure, auto-rotating solution for ALL credentials

aws

Demonstration 2
Manage and retrieve database credential

Typical use cases – rotate secret



Rotate database credentials used by application code without interruption

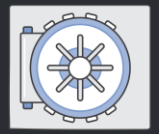
- Secrets Manager creates a new credential with equivalent permissions.
- The new credential is promoted and returned via subsequent Secrets Manager API calls.
- Secrets Manager safely disables the original credential.

aws

Rotate Secret (Integrated)

1) Rotation event

AWS Secrets Manager



2) New credential creation initiated

Integrated Backend Resource

3) New credential creation completed

4) New credential promoted to active

Your Code

5) Secret retrieval gets new version

6) Connection using new credential

aws

Typical use cases – control access, monitor, and audit secret

Use the AWS eco-system to control access, monitor, and audit secrets

- IAM policies for access control
- Tag-based access control
- Resource-based policies for access control

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/*"
    }
  ]
}
```

Access control – using Tags

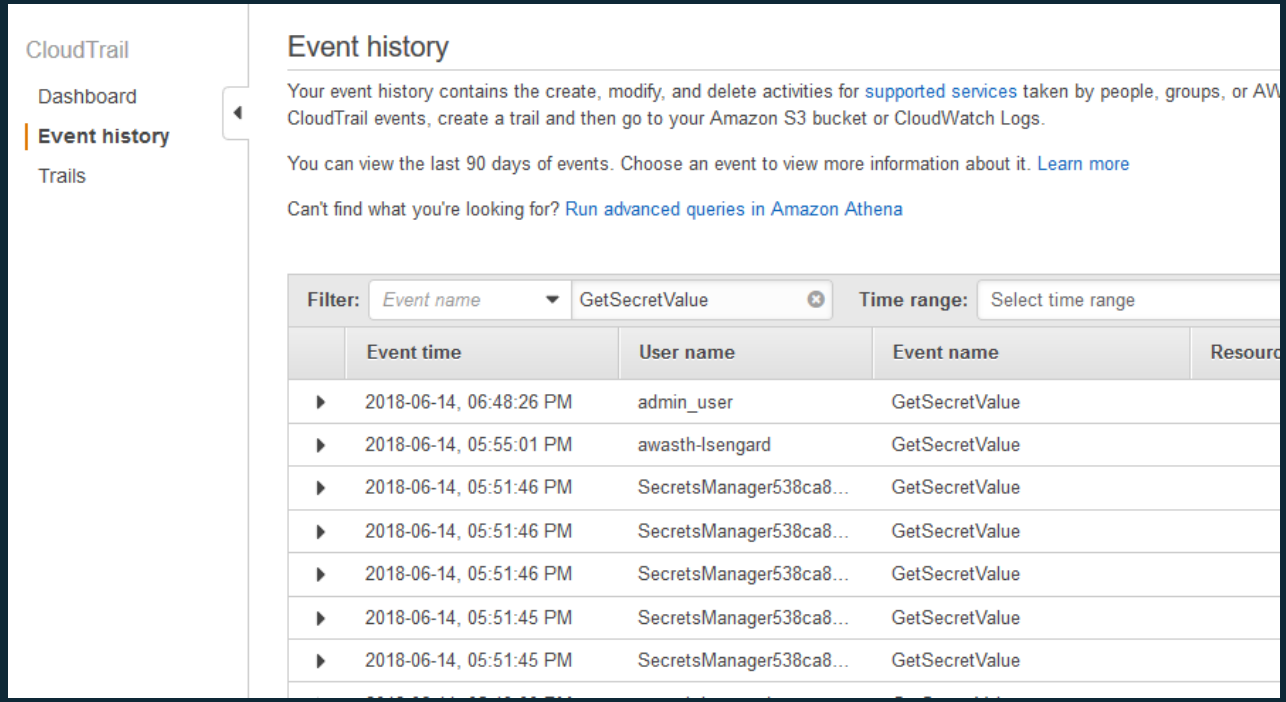
```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
          "Effect": "Allow",
          "Action": ["secretsmanager:Describe*", "secretsmanager:GetSecretValue"],
          "Resource": "*",
          "Condition": {
             "StringEqualsIgnoreCase": {
                "secretsmanager:ResourceTag/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

Access control – using resource-based policies

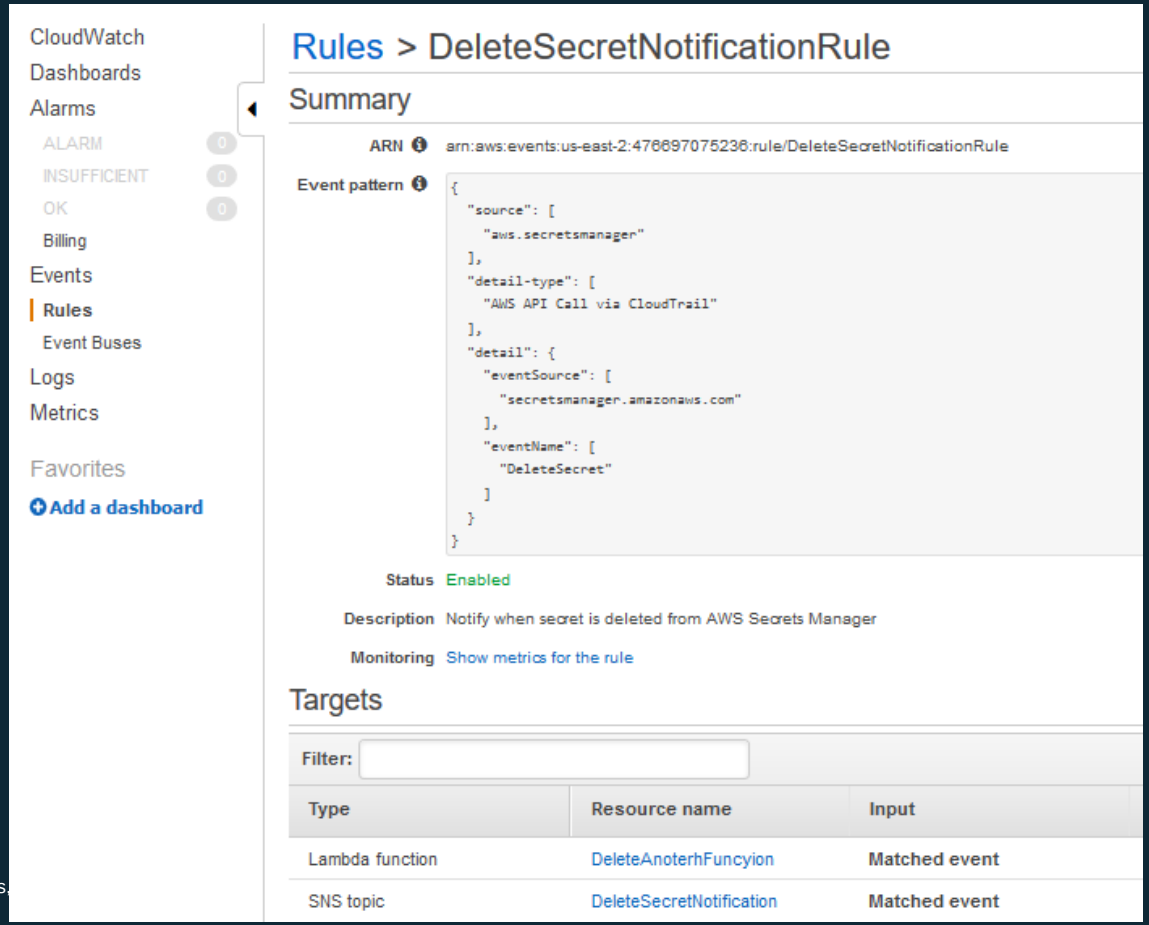
```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::ACCOUNT_NUMBER:role/demo_role"},
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*",
      "Condition": {"ForAnyValue:StringEquals": {"secretsmanager:VersionStage": "AWSCURRENT"}}
    }
  ]
}
```

```
$aws secretsmanager put-resource-policy --secret-id MY_TEST_SECRET --resource-policy
file://RESOURCE_POLICY.json
```

Audit access – using AWS CloudTrail



Monitor access – using Amazon CloudWatch



As you get started…

1. No plaintext secrets
2. Unique secrets per region, per environment, per account
3. Rotate secrets regularly
4. Control permissions
5. Tags and hierarchical names to scale secrets management
6. Monitor and audit use; deprecate unused secrets
7. No charge for versions of a secret; no charge for using the default encryption key

aws

Thank you!

Questions?

Advanced Use Cases

Advanced use cases – manage and rotate Twitter API key

Step 1: Store a Twitter API key and bearer token in Secrets Manager.

Step 2: Create a custom Lambda function to rotate the bearer token.

Step 3: Configure your application to retrieve the bearer token from Secrets Manager.

Step 4: Configure Secrets Manager to use the custom Lambda function to rotate the bearer token automatically.

Detailed instructions available at: https://aws.amazon.com/blogs/security/how-to-rotate-your-twitter-api-key-and-bearer-token-automatically-with-aws-secrets-manager/