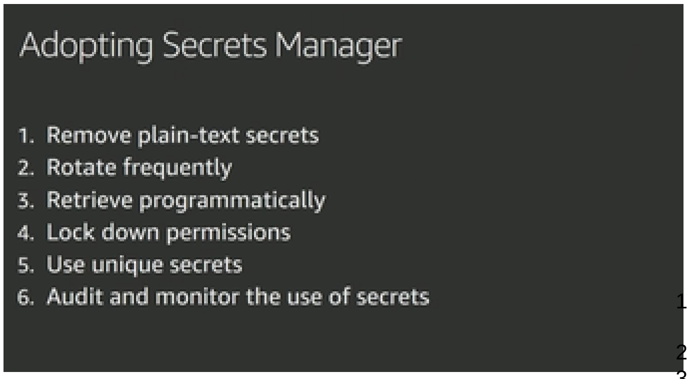
Best Practices for adopting Secrets Manager-



Adopting Secrets Manager

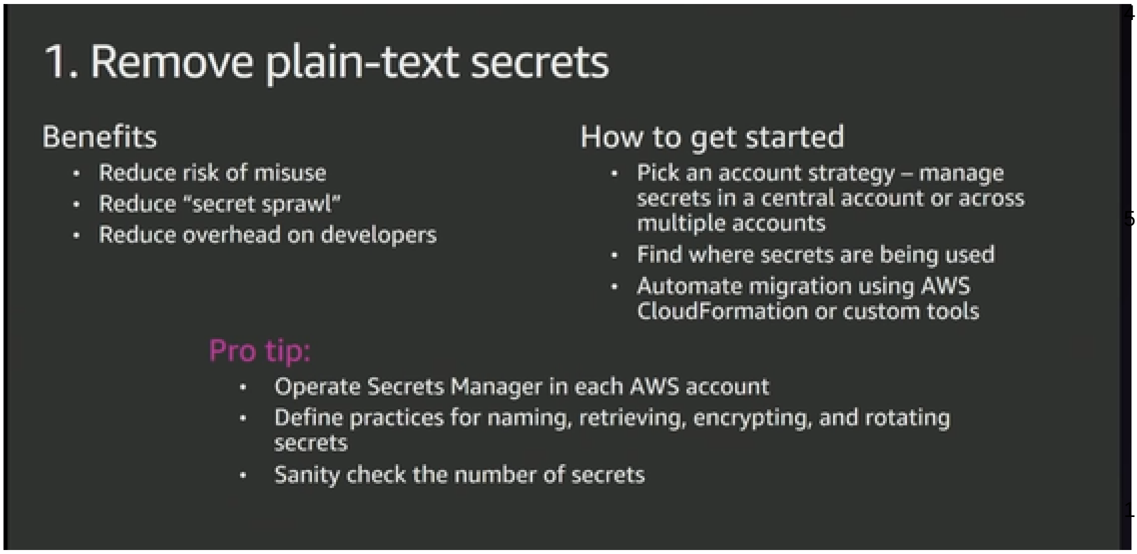
1. Remove plain-text secrets
2. Rotate frequently
3. Retrieve programmatically
4. Lock down permissions
5. Use unique secrets
6. Audit and monitor the use of secrets

Creating the CloudWatch Alarm

To receive a notification when a Secrets Manager GetSecretValue API operation requests to access a version of a secret pending deletion, you must create a CloudWatch alarm and configure notification.

To create a CloudWatch alarm to monitor the usage of a version of a secret pending deletion

1. Sign in to the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. On the top navigation bar, choose the AWS Region to monitor secrets.
3. In the left navigation pane, choose **Logs**.



1. Remove plain-text secrets

Benefits

- Reduce risk of misuse
- Reduce "secret sprawl"
- Reduce overhead on developers

How to get started

- Pick an account strategy – manage secrets in a central account or across multiple accounts
- Find where secrets are being used
- Automate migration using AWS CloudFormation or custom tools

Pro tip:

- Operate Secrets Manager in each AWS account
- Define practices for naming, retrieving, encrypting, and rotating secrets
- Sanity check the number of secrets

4. In the list of **Log Groups**, select the check box next to the log group you created in the previous procedure, such as SecretsLog. Then choose **Create Metric Filter**.
5. For **Filter Pattern**, type or paste the following:

   { $.eventName = "GetSecretValue" && $.errorMessage = "*secret because it was deleted*" }

   Choose **Assign Metric**.

1. On the **Create Metric Filter and Assign a Metric** page, perform the following steps:

   a. For **Metric Namespace**, type **CloudTrailLogMetrics**.

   b. For **Metric Name**, type **AttemptsToAccessDeletedSecrets**.

   c. Choose **Show advanced metric settings**, and then if necessary for **Metric Value**, type **1**.

   d. Choose **Create Filter**.

2. In the filter box, choose **Create Alarm**.
3. In the **Create Alarm** window, do the following:

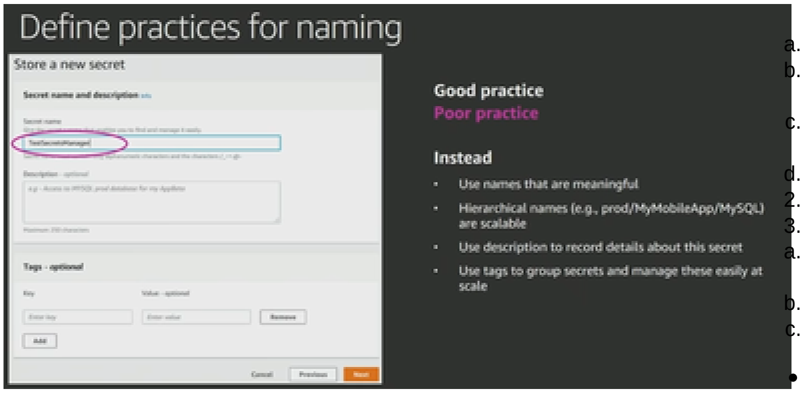
   a. For **Name**, type **AttemptsToAccessDeletedSecretsAlarm**.

   b. **Whenever:**, for **is:**, choose **>=**, and then type **1**.

   c. Next to **Send notification to:**, perform one of the following options:

   - To create and use a new Amazon SNS topic, choose **New list**, and then type a new topic name. For **Email list:**, enter at least one email address. You can enter more than one email address by separating them with commas.
   - To use an existing Amazon SNS topic, choose the name of the topic to use. If a list isn't available, choose **Select list**.

   d. Choose **Create Alarm**.



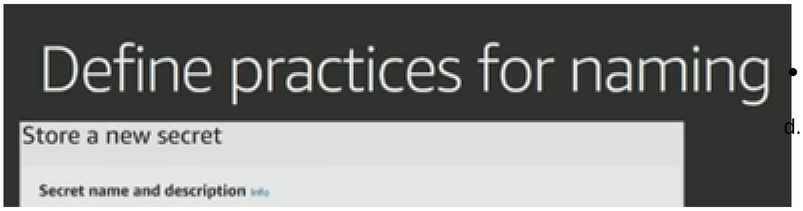
Define practices for naming

Good practice

Poor practice

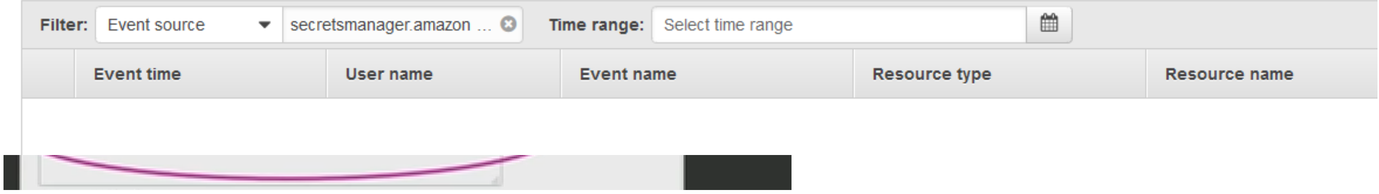
Instead

- Use names that are meaningful
- Hierarchical names (e.g., prod/MyMobileApp/MySQL) are scalable
- Use description to record details about this secret
- Use tags to group secrets and manage these easily at scale



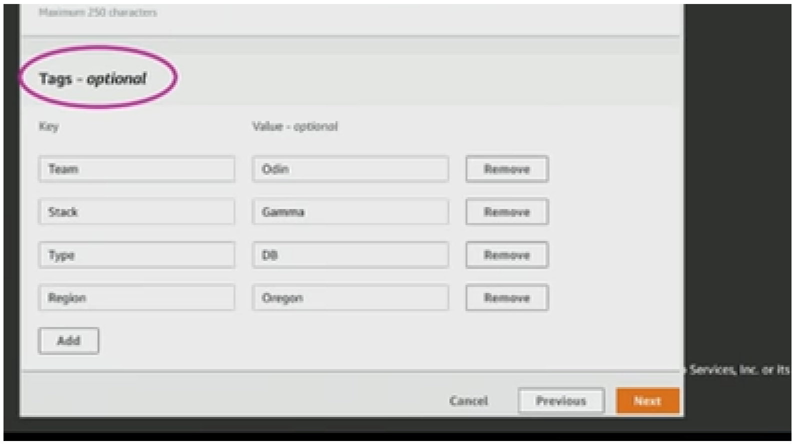
Define practices for naming

Checking on Cloud Trail event history for auditing,



Filter: Event source | secretsmanager.amazon ... | Time range: Select time range

	Event time	User name	Event name	Resource type	Resource name



Maximum 250 characters

Tags - optional

Key	Value - optional	
Team	Odin	Remove
Stack	Gamma	Remove
Type	DB	Remove
Region	Oregon	Remove

Define practices for encryption

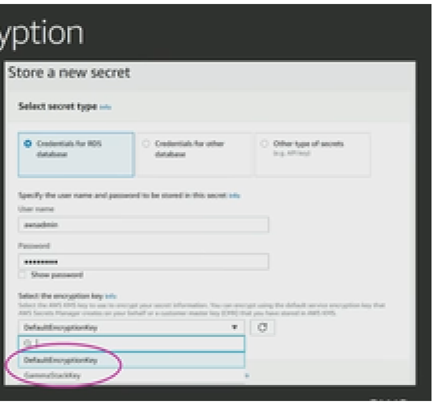
Good practice
Poor practice
Depends

Default service key

- Unique key for each account and region
- No overheard of managing AWS KMS permissions

Customer master key (CMK)

- Unique compliance or security requirements
- Required for cross-account access to secrets
- Another set of access control



Store a new secret

Select secret type Info

● Credentials for RDS database ○ Credentials for other database ○ Other type of secrets (e.g. API key)

Specify the user name and password to be stored in this secret Info

User name: awsadmin

Password: •••••••••

□ Show password

Select the encryption key Info

Select the AWS KMS key to use to encrypt your secret information. You can encrypt using the default service encryption key that AWS Secrets Manager creates on your behalf or a customer master key (CMK) that you have stored in AWS KMS.

DefaultEncryptionKey

2. Rotate frequently

Benefits

- Improve security
- Follow best practice

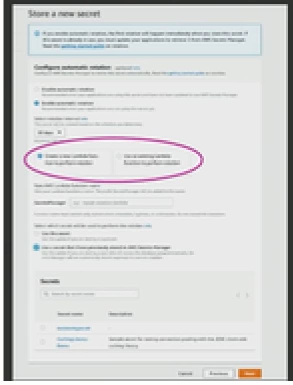
How to get started

- For existing applications, first migrate the secret, then configure rotation
- For new applications, set up rotation from the start
- Create the rotation lambda function

Pro tip:

- Use the default frequency of 30 days; check your compliance and security requirements
- Pay for APIs and use of Lambda; no extra charge for rotation
- Rotation Lambda functions must be able to communicate both with the protected resource (e.g. a database) and with Secrets Manager
- Use VPC end-points
- Update the password policy according to your downstream systems
- Reuse rotation Lambda functions

Rotate frequently



Store a new secret

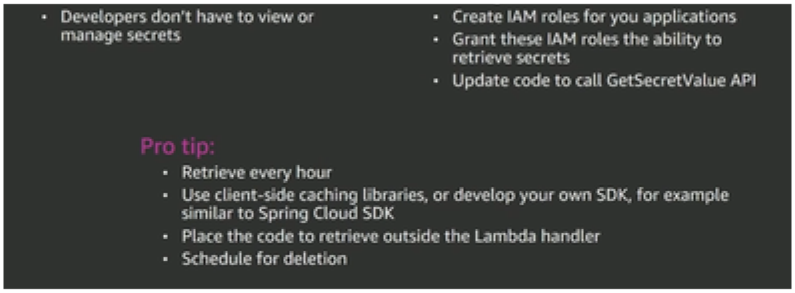
Create new Lambda
Reuse existing Lambda

- Easier to separate IAM permissions from Secrets Manager permissions
- Easier to manage a small number of rotation lambda functions

3. Retrieve programmatically

Benefit

How to get started

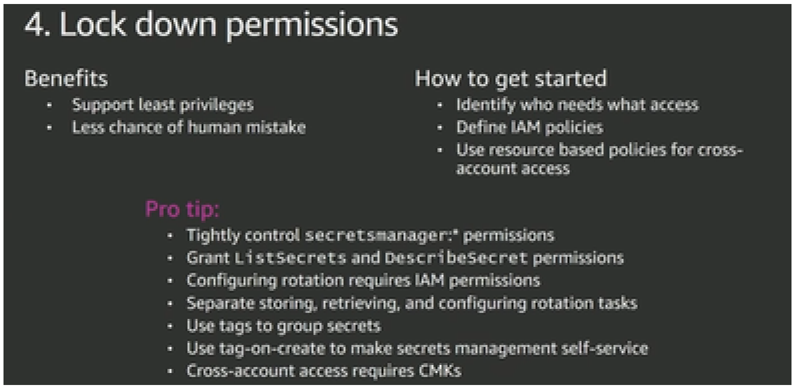


- Developers don't have to view or manage secrets

- Create IAM roles for you applications
- Grant these IAM roles the ability to retrieve secrets
- Update code to call GetSecretValue API

Pro tip:

- Retrieve every hour
- Use client-side caching libraries, or develop your own SDK, for example similar to Spring Cloud SDK
- Place the code to retrieve outside the Lambda handler
- Schedule for deletion



4. Lock down permissions

Benefits

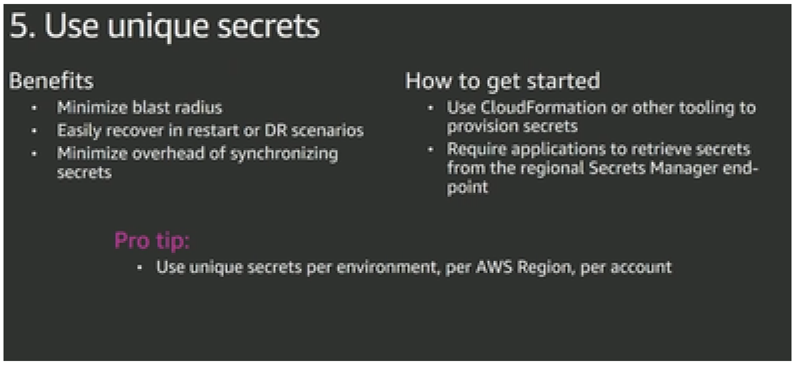
- Support least privileges
- Less chance of human mistake

How to get started

- Identify who needs what access
- Define IAM policies
- Use resource based policies for cross-account access

Pro tip:

- Tightly control `secretsmanager:*` permissions
- Grant `ListSecrets` and `DescribeSecret` permissions
- Configuring rotation requires IAM permissions
- Separate storing, retrieving, and configuring rotation tasks
- Use tags to group secrets
- Use tag-on-create to make secrets management self-service
- Cross-account access requires CMKs



5. Use unique secrets

Benefits

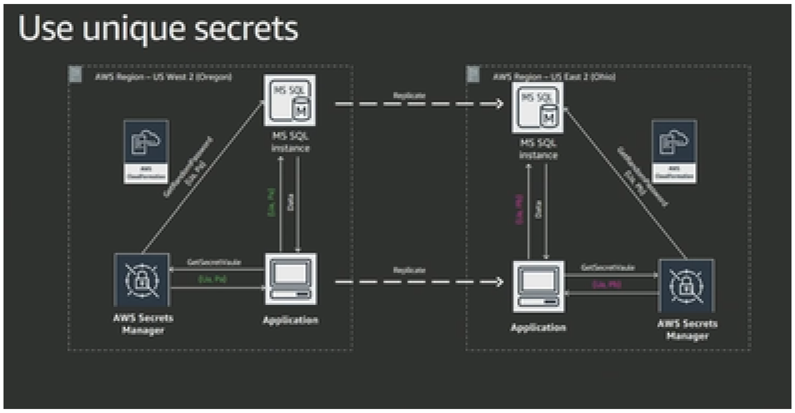
- Minimize blast radius
- Easily recover in restart or DR scenarios
- Minimize overhead of synchronizing secrets

How to get started

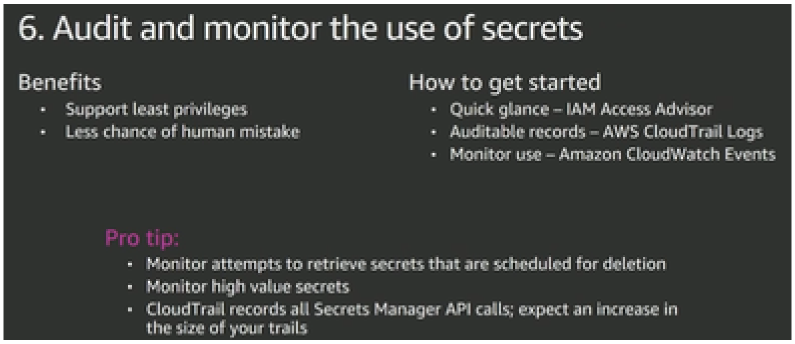
- Use CloudFormation or other tooling to provision secrets
- Require applications to retrieve secrets from the regional Secrets Manager end-point

Pro tip:

- Use unique secrets per environment, per AWS Region, per account



Use unique secrets



6. Audit and monitor the use of secrets

Benefits

- Support least privileges
- Less chance of human mistake

How to get started

- Quick glance – IAM Access Advisor
- Auditable records – AWS CloudTrail Logs
- Monitor use – Amazon CloudWatch Events

Pro tip:

- Monitor attempts to retrieve secrets that are scheduled for deletion
- Monitor high value secrets
- CloudTrail records all Secrets Manager API calls; expect an increase in the size of your trails

[redacted]