

Manage Secrets Like a Pro

Avni Rambhia

Sr. Product Manager, AWS Secrets Manager



Customers want to . . .

- Manage secrets centrally
- Store secrets securely
- Control access to secrets
- Retrieve secrets at scale
- Centrally audit and monitor secrets usage
- Ensure compliance

What to expect in today's session

- Choosing the appropriate AWS service to secure a given type of secret
- When using AWS Secrets Manager:
 - What we do for you
 - What you need to take care of
 - What features we provide for each stage of the secret lifecycle
- Getting the most out of Secrets Manager

Secrets are of many types

IAM credentials

Database
credentials

OAuth tokens

Encryption keys

SECRETS

Application
credentials

SSH keys

Certificates (and
private keys)

Application
programming
interface (API) keys

Modernization is an opportunity to select the easiest service

AWS credentials

AWS Identity & Access Management (IAM)

Encryption keys

AWS Key Management Service (KMS)

SSH keys

Amazon EC2 Instance Connect

Private keys + Certificates

AWS Certificate Manager

Database credentials

Application credentials

OAuth secrets

Application programming interface (API) keys

AWS Secrets Manager



Manage secrets through their lifecycle

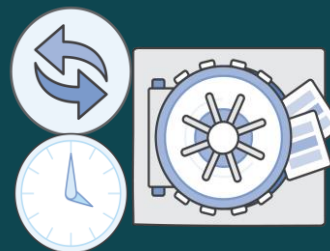
AWS Secrets Manager enables customers to **create, rotate, manage, and retrieve** database credentials, API keys, and other secrets while ensuring **compliance**



Secure by default



Fine-grained access control



Rotate



Replicate



Retrieve at scale

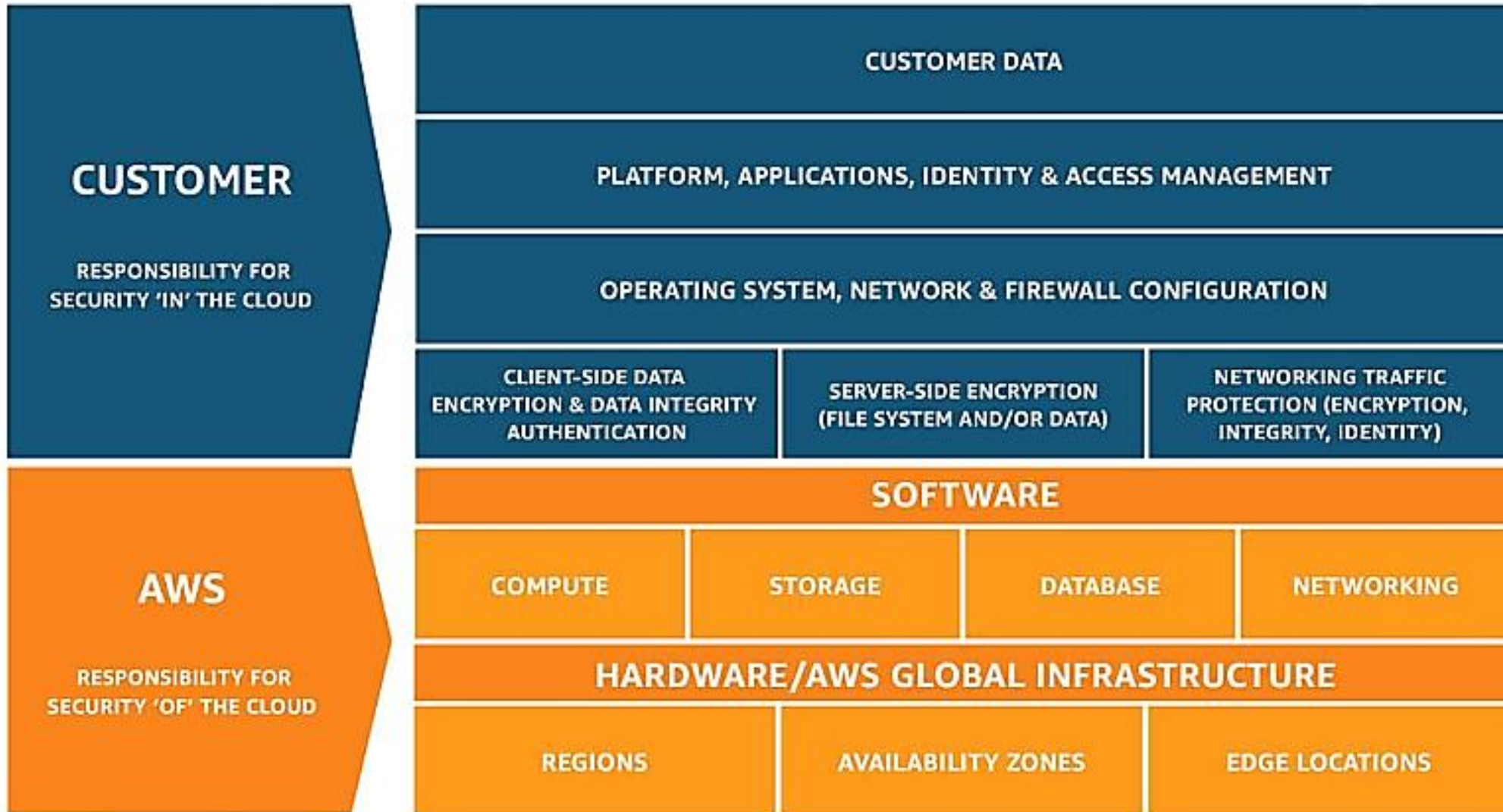


Monitor and audit



Meet compliance

AWS shared responsibility model (SRM)



SRM for Secrets Manager (examples)

Customer

Properly configure policies on secret and key that encrypts it

Use HA strategies and caching libraries to keep applications running while secrets rotate

Set monitors for non-compliant secrets

Be aware of service limits

Load test

AWS

Encrypt secrets by default

Enforce IAM and resource policies

Tools and guidance for safe rotation

Client-caching libraries

Integrations with AWS monitoring services

SLA

Create and use your secrets securely



Why does it matter?

- Secrets enable access to sensitive data
- Important to control access to secrets throughout the lifecycle:
 - At creation
 - During use
 - During updates

Pro tip #1: Automate provisioning of secrets through AWS CloudFormation

- Secret: Specify a value, or let Secrets Manger generate one for you
- Other supported resource types: ResourcePolicy, RotationSchedule, SecretTargetAttachment

Pro tip #2: Rotate secrets immediately after manual creation

Ensures any access to current secret value is controlled

- Via console: Rotation resources can be automatically provisioned, while creating a secret
- Via API/CLI: RotateSecret
- Ad-hoc: PutSecretValue + Resource update

Pro tip #3: Scope down to least privilege access

- Can be liberal with ListSecrets and DescribeSecret
- Applications need read-only permissions to GetSecretValue
- Use conditions to allow access to “Current” version of secret
- Verify which principals have access to secrets through an IAM policy using the IAM Policy Simulator (<https://policysim.aws.amazon.com/>)

Pro tip #4: Use AWS managed policies for elevated access

- SecretsManagerReadWrite
- IAMFullAccess
 - Trusted administrators only
 - Enforce multi-factor authentication (MFA)

Pro tip #5: Leverage ABAC

For convenient bulk-editing of permissions:

- Tag groups of related secrets
- Use Attribute-Based Access Control (ABAC) to simplify and automate permissions

Pro tip #6: Leverage KMS Keys for added control

- Admins can bulk-revoke application's access to secrets by revoking KMS key permissions
- Encrypt secrets with customer-managed keys to share them across accounts
- Note: You must use resource-based policies and customer-managed keys to share secrets across accounts

Pro tip #7: Use Secrets Manager to secure secrets across AWS services



Containers



Amazon ECS



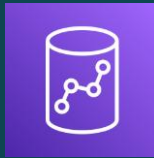
Amazon EKS



AWS Fargate



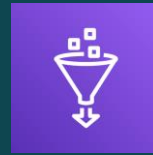
Analytics



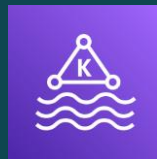
Amazon Redshift



Amazon EMR



AWS Glue



Amazon MSK



Internet of Things



AWS IoT Greengrass



Database



Amazon RDS



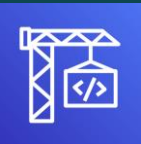
AWS DMS



Amazon DocumentDB



Developer Tools



AWS CodeBuild



Machine Learning



Amazon SageMaker



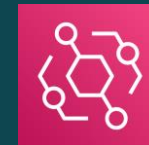
Amazon Kendra



Application Integration



Amazon AppFlow



Amazon EventBridge



Management & Governance



AWS OpsWorks

Pro tip #8: Scan regularly for unprotected secrets with CodeGuru Secrets Detector

- Detect secrets in source code or configuration files
 - Finds passwords, common API keys, SSH keys, and access tokens
 - In Java/Python code, configuration and documentation files
- Secure detected secrets with Secrets Manager



Retrieve secrets securely



Why does it matter?

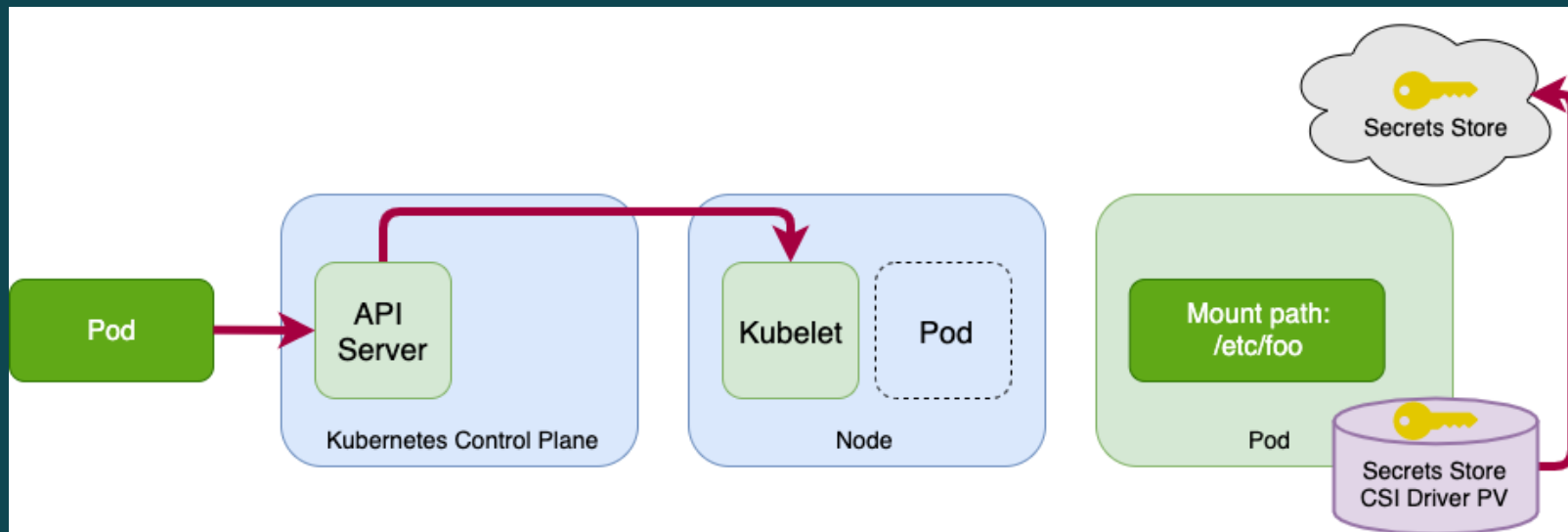
- Eliminate the need for developers to view or handle secrets
- Easily revoke access to secrets
- Update secrets without re-deploying code
- Reduce development time and operational risk

Pro tip #1: Use client-side caching libraries instead of directly calling GetSecretValue

- Reduce API bill
- Increase availability
- Reduce undifferentiated heavy lifting, esp. for retries and rotation
 - Java Database Connectivity (JDBC) driver automatically refreshes secret after it rotates
 - You can configure refresh interval for other client-side libraries (Java, Python, Go, .NET, Kubernetes)

Pro tip #2: Use the AWS secrets and config provider (ASCP) for Kubernetes workloads

- Easily mount secrets as environment variables via CSI driver plug-in
- Scope secrets to pods within clusters
- Rely on rotation reconciler to mount rotated secret values



Pro tip #3: Retrieve secrets as global variables in AWS Lambda

- Use global variables instead of retrieving secrets inside Lambda handler
- Prevents GetSecretValue API calls every time the Lambda executes
 - Reduces API cost
 - Prevents throttling
 - Improves performance

Pro tip #4: Schedule deletion of unused secrets

- Identify unused secrets using AWS Config

 secretsmanager-secret-unused	SecretsManager, Secret, Unused	Checks if AWS Secrets Manager secrets have been accessed in the past 90 days. This rule is NON_COMPLIANT if a secret has not been accessed in 90 days.
--	--------------------------------	--

- Schedule deletion after 7–30 days to delete safely using `--recovery-window-in-days`
 - Secrets scheduled for deletion cannot be read
 - You can restore secrets during the recovery window

Pro tip #5: Refresh based on rotation settings

- Two secrets valid at a time: refresh at half-life
- One secret valid at a time:
 - Refresh aggressively, cache connections, or proactively broadcast update via rotation Lambda
 - Retries with exponential back-off and jitter

Rotate secrets



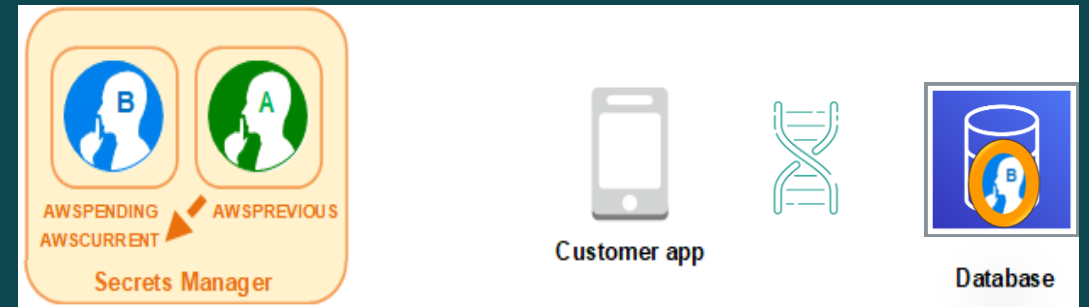
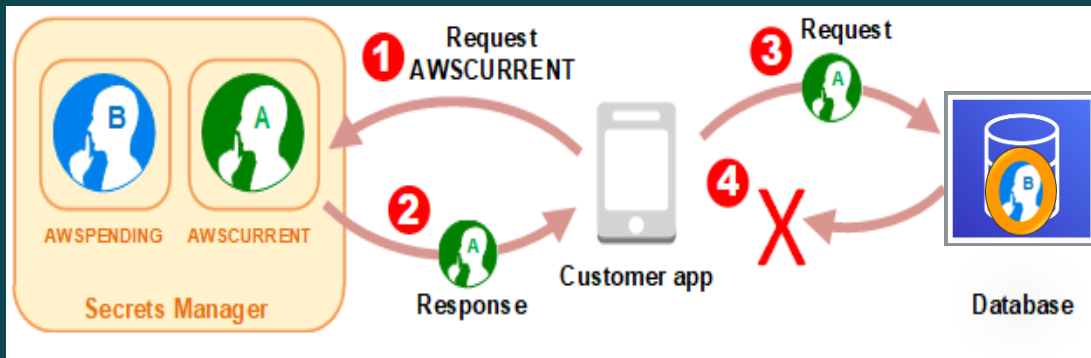
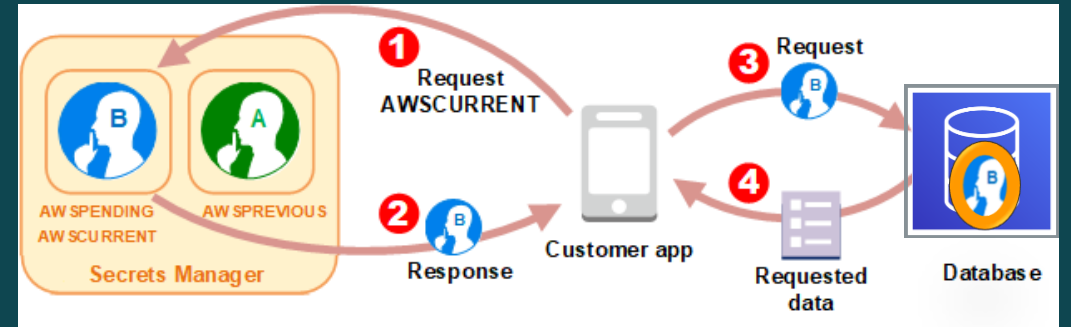
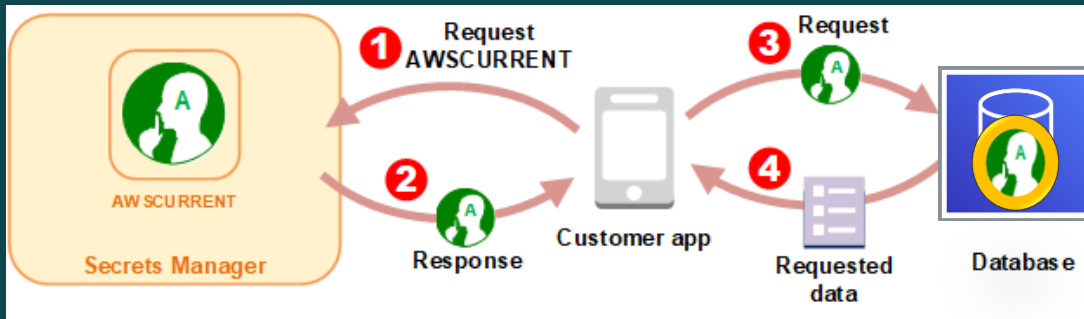
Why does it matter?

- Compliance
- Improved security: short-term secrets have lower blast radius than fixed secrets

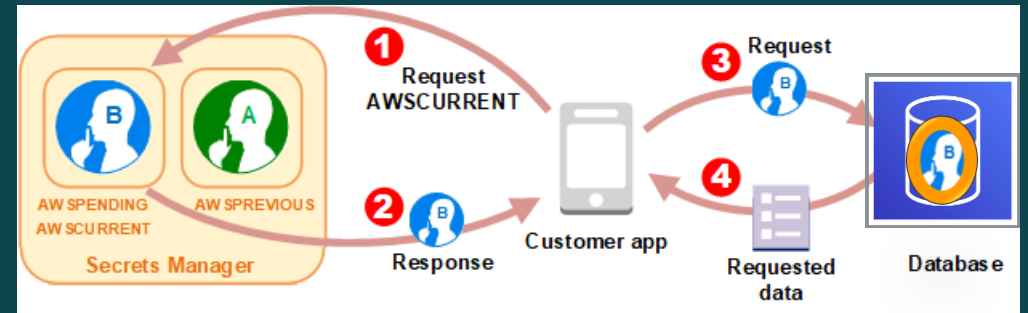
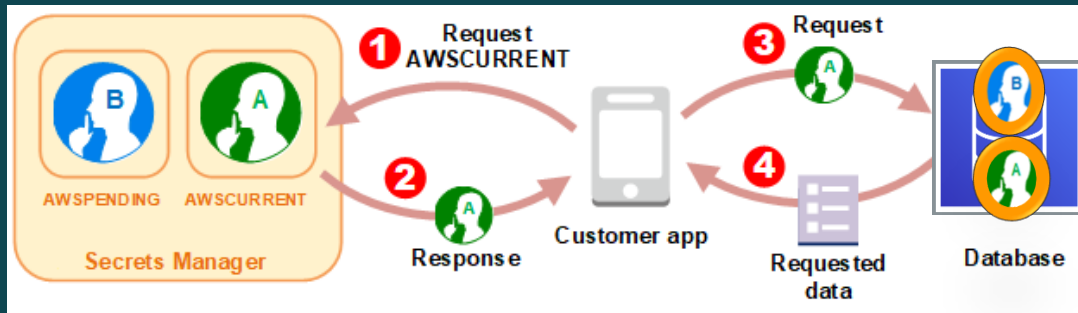
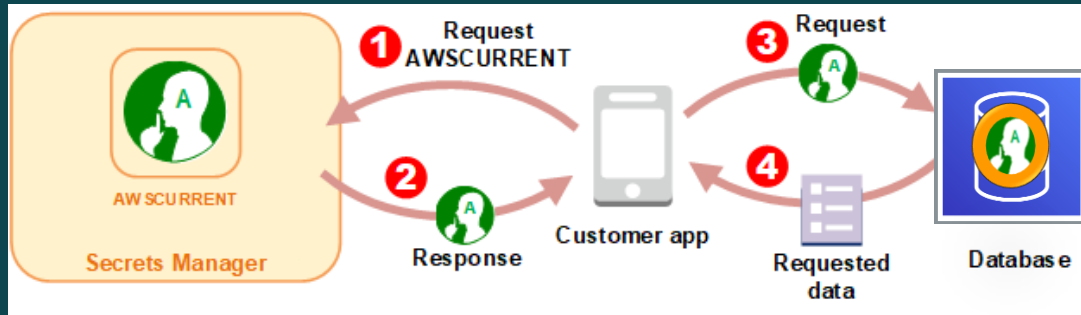
Pro tip #1: Alternate between two credentials for highly available applications

- Leverage concurrent credentials when resources permit this
- Alternate between equivalent accounts for databases
 - Secrets Manager provides multi-user rotation templates for Amazon-managed databases
 - You can customize your rotation Lambda for other resources
- If you must rotate a single credential:
 - Set high lockout limits, esp. for container and serverless workloads
 - Refresh aggressively, cache connections, or proactively broadcast update via rotation Lambda

Detour: Single credential rotation



Detour: Dual credential rotation



Pro tip #2: Reuse rotation Lambdas for multiple secrets

- Easier to manage a small number of rotation Lambda functions
- Can still customize password generation to downstream application requirements

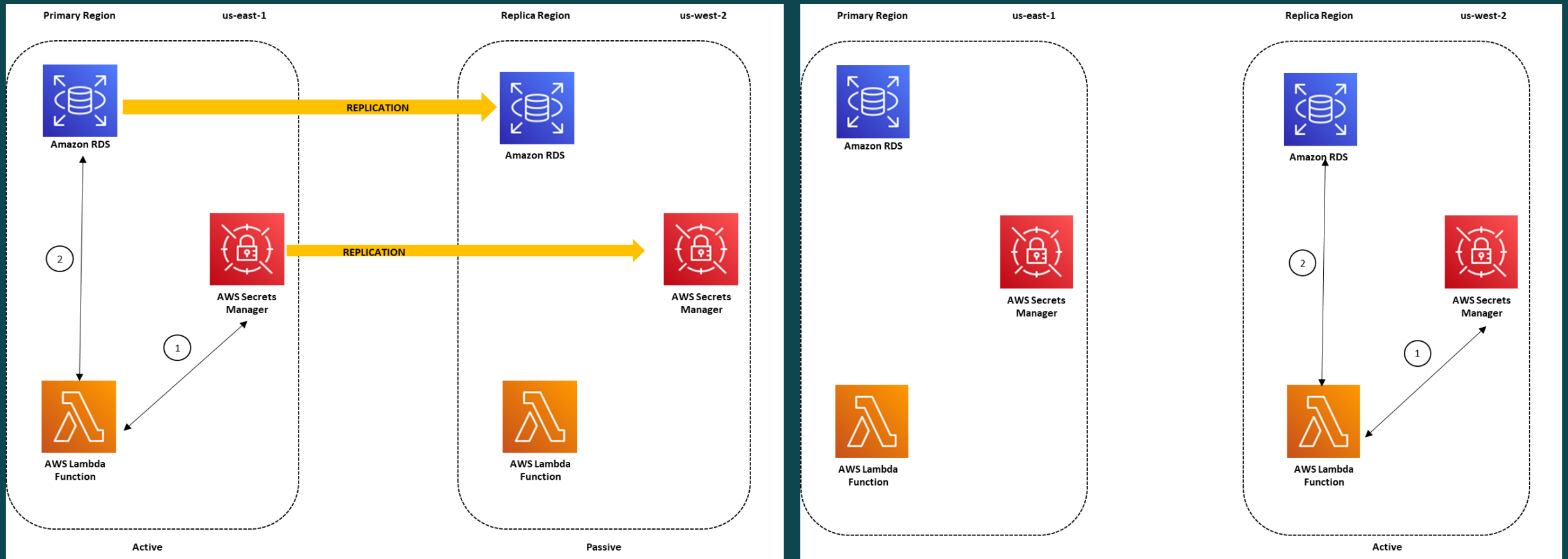
Pro tip #3: Align rotation schedule with maintenance windows

- By default, secrets rotate within a 24 hour timeframe every N days
- With scheduled windows, you can:
 - Limit rotation to certain hours of the day
 - Schedule rotation by day of week or month

Pro tip #4: Manage and monitor rotation

- By default, secrets rotate every 30 days. Set your rotation interval based on your compliance needs
- Configure AWS Config rules to watch for secrets that have not rotated as expected
- Leverage AWS CloudTrail events for rotation status

Pro tip #4: Use multi-region secrets for multi-region resources



Audit and monitor



Why does it matter?

- Identify unexpected usage or changes, so you can investigate
- Identify unwanted changes, so they can be rolled back
- Compliance

Pro tip #1: Leverage AWS services to monitor secrets

- IAM Access Analyzer to analyze permissions to secrets
- AWS CloudTrail logs to monitor use (including data plane)
- Amazon CloudWatch events to customize responses to changed secrets
- AWS Config to evaluate rules (e.g. rotation enabled & rotation succeeds)
- AWS Security Hub to easily evaluate compliance across organization
- AWS Config Aggregator for unified views across accounts

Account-level checks in AWS Config

AWS Managed Rules (5)

Q secretsmanager X < 1 > ⚙

	Name ▲	Labels	Description
<input type="radio"/>	secretsmanager-rotation-enabled-check	secretsmanager, secret	Checks whether AWS Secret Manager secret has rotation enabled. If the maximumAllowedRotationFrequency parameter is specified, the rotation frequency of the secret is compared with the maximum allowed frequency.
<input type="radio"/>	secretsmanager-scheduled-rotation-success-check	secretsmanager, secret	Checks and verifies whether AWS Secret Manager secret rotation has rotated successfully as per the rotation schedule.
<input type="radio"/>	secretsmanager-secret-periodic-rotation	SecretsManager, Secret, Periodic, Rotation	Checks if AWS Secrets Manager secrets have been rotated in the past 90 days by examining 'LastRotatedDate' value of the secret. If 'LastRotatedDate' doesn't exist then this rule will check for the creation date of the secret.
<input type="radio"/>	secretsmanager-secret-unused	SecretsManager, Secret, Unused	Checks if AWS Secrets Manager secrets have been accessed in the past 90 days. This rule is NON_COMPLIANT if a secret has not been accessed in 90 days.
<input type="radio"/>	secretsmanager-using-cmk	SecretsManager, CMK	Checks if all secrets in AWS Secrets Manager are encrypted using an AWS Key Management Service (AWS KMS) customer master key (CMK). This rule is NON_COMPLIANT if a secret is encrypted using the default AWS KMS key.

Cancel Next

Cross-account checks in AWS Config Aggregator

The screenshot displays the AWS Config Aggregator dashboard in an 'Aggregated view'. At the top, there are filters for 'Aggregator' (set to 'myOrganizationsSecre...'), 'Region' (set to 'All regions'), and 'Account' (set to 'All accounts'). A note indicates that data is refreshed from multiple sources and may be delayed.

Resources

Resource Type	Total
SecretsManager Secret	246
CloudFormation Stack	84
EC2 NetworkInterface	74
EC2 SecurityGroup	63
CloudWatch Alarm	61
EC2 Subnet	51
Lambda Function	36
S3 Bucket	30
KMS Key	30
EC2 RouteTable	27

[View all 890 resources](#)

Config rule compliance status

- 7 noncompliant rule(s)
- 11 compliant rule(s)

Top 5 noncompliant rules

Rule name	Region	Account	Compliance
encrypted-volumes	us-east-1	[Redacted]	10 noncompliant resource(s)
restricted-ssh	us-east-1	[Redacted]	9 noncompliant resource(s)
secretsmanager-rotation-ena...	us-east-1	[Redacted]	5 noncompliant resource(s)
ec2-volume-in-use-check	us-east-1	[Redacted]	3 noncompliant resource(s)
restricted-ssh	us-west-2	[Redacted]	3 noncompliant resource(s)

[View all noncompliant rules](#)

Accounts by noncompliant rules

Account	Compliance
[Redacted]	4 noncompliant rule(s)
[Redacted]	3 noncompliant rule(s)

[View all noncompliant rules](#)

Accounts by resource count

Account	Resource count
[Redacted]	666 resource(s)
[Redacted]	224 resource(s)

[View all 890 resources](#)

Organizational compliance with Security Hub

All enabled 96	Failed 0	Unknown 0	No data 96	Passed 0	Disabled 0
--------------------------	-------------	--------------	---------------	-------------	---------------

All enabled (4) Disable Download

Q Filter enabled controls

"secrets" X Clear filters

	Status ▼	Severity ▼	ID ▼	Title ▼	Failed checks ▼
<input type="radio"/>	⋮ No Data	■ Medium	SecretsManager.1	Secrets Manager secrets should have automatic rotation enabled	0 of 0
<input type="radio"/>	⋮ No Data	■ Medium	SecretsManager.2	Secrets Manager secrets configured with automatic rotation should rotate successfully	0 of 0
<input type="radio"/>	⋮ No Data	■ Medium	SecretsManager.3	Remove unused Secrets Manager secrets	0 of 0
<input type="radio"/>	⋮ No Data	■ Medium	SecretsManager.4	Secrets Manager secrets should be rotated within a specified number of days	0 of 0

Decentralize secrets



Why does it matter?

- Quotas are at account-level
- Operational visibility is at account-level
- Monitoring tools are account-level and org-level

Pro tip #1: Create and store secrets in the accounts where they will be used

- Preserves velocity of development teams
- Reduces blast radius from compromised account credentials
- Prevents cross-application operational impact

Pro tip #2: Monitor secrets centrally

- Discussed in previous section
- Provides a balance between control and flexibility

Final Thoughts



Key Takeaways

- Minimize opportunities for secrets and humans to mix
- Plan ahead for access control and rotation
- Caching libraries reduce undifferentiated heavy lifting
- Centralized monitoring and decentralized secrets can co-exist

AWS manages our secrets with Secrets Manager – When you use Secrets Manager, you benefit from our mindset that “security is top priority”

Thank you for joining us!

- Questions?
- Please fill out the session survey!