In this tutorial, you do the following:

Explore the AWS Secrets Manager service

[What is AWS Secrets Manager?](#_95laxyfjm05z)

[When do I use AWS Secrets Manager?](#_xfu2zgvab4h0)

[What are alternatives to AWS Secrets Manager?](#_f5n4ghi24kq9)

[Creating a Secret in AWS Secrets Manager (Console)](#_3y74q8txiwlk)

[Creating AWS Secrets Manager secret from CLI](#_yhrlxjgqo1ed)

[Creating AWS Secrets Manager secret from Terraform](#_1o7ef24eiqne)

### What is AWS Secrets Manager?

AWS Secrets Manager is a **fully managed service** provided by Amazon Web Services (AWS) that helps you **protect access to applications, services, and IT resources**. It allows you to store and manage secrets, such as database credentials, API keys, and other sensitive information, in a centralized location, which can be accessed by authorized applications or users.

With AWS Secrets Manager, you can **easily rotate, manage, and retrieve secrets programmatically**, without the need for hardcoding secrets in your application code or configuration files. Secrets can be stored encrypted using AWS KMS, which helps you meet compliance requirements.

Some of the key features of AWS Secrets Manager include:

* Secret rotation: AWS Secrets Manager can automatically rotate secrets based on a predefined schedule or in response to specific events, such as a security breach.
* Access control: You can use AWS Identity and Access Management (IAM) to manage access to secrets, and audit access using AWS CloudTrail.
* Integration: AWS Secrets Manager can be integrated with other AWS services such as Amazon RDS, Amazon Redshift, and Amazon DocumentDB, making it easy to securely manage secrets across your applications and services.
* Cost-effective: You pay only for the number of secrets you store and the number of API calls you make, with no upfront fees or long-term commitments.

AWS Secrets Manager is a useful service for managing and securing sensitive information in your applications and services, and can help you reduce the risk of security breaches and meet compliance requirements.

### When do I use AWS Secrets Manager?

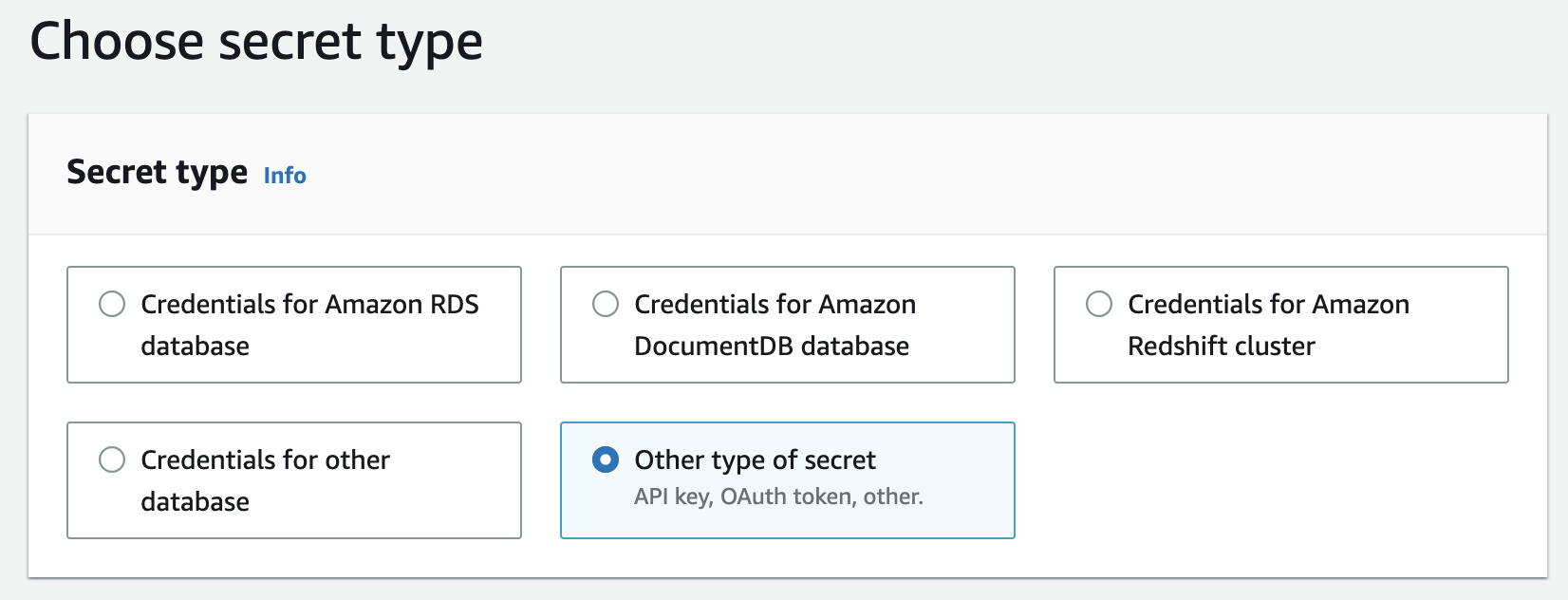
* **Storing database credentials**: You can use Secrets Manager to securely store database credentials, such as usernames and passwords, and retrieve them programmatically when needed. This helps you avoid hardcoding credentials in your application code or configuration files, which can be a security risk.
* **Managing API keys**: If your application uses API keys to access external services, you can store and manage them in Secrets Manager to help protect them from unauthorized access or misuse.
* **Storing encryption keys**: You can use Secrets Manager to store encryption keys used to encrypt and decrypt data in your application or service, and retrieve them securely when needed.
* **Managing certificates**: If your application uses SSL/TLS certificates, you can store and manage them in Secrets Manager to help ensure they are up-to-date and secure.
* **Integrating with other AWS services**: Secrets Manager can be integrated with other AWS services such as Amazon RDS, Amazon Redshift, and Amazon DocumentDB, making it easy to securely manage secrets across your applications and services.

### What are alternatives to AWS Secrets Manager?

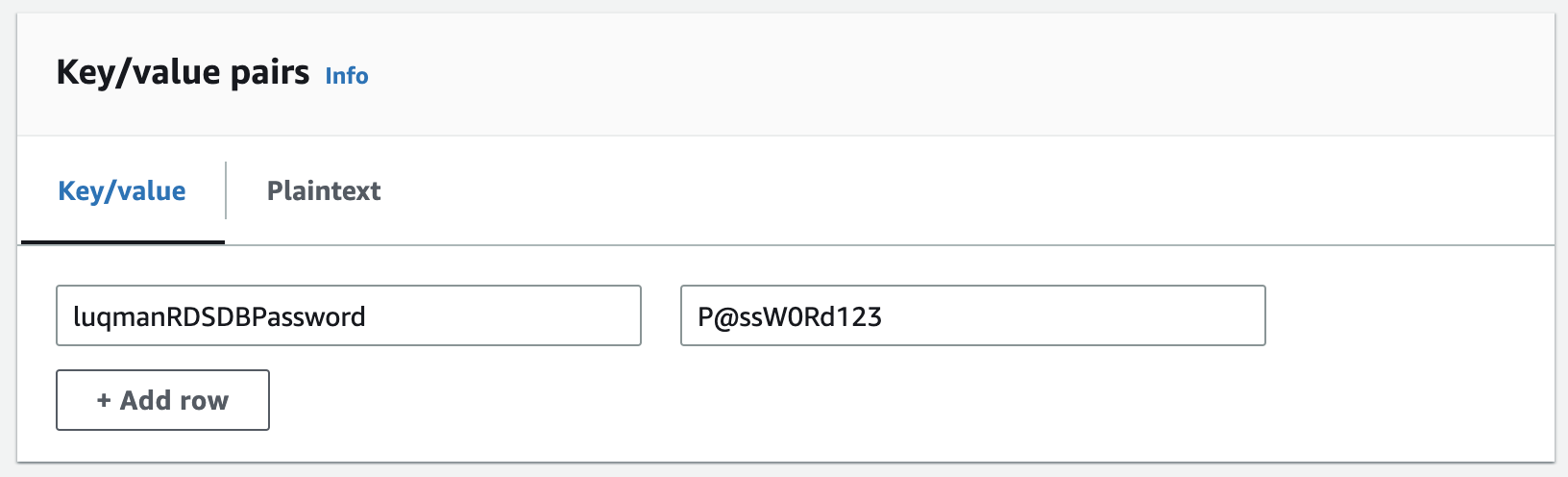
* **AWS Parameter Store**: AWS Parameter Store is a service that allows you to store and manage configuration data, including secrets, for your applications. It provides a simple key-value store that you can use to store secrets, and you can control access to the data using IAM.
* **HashiCorp Vault:** HashiCorp Vault is an open-source tool that provides a centralized system for managing secrets, encryption keys, and other sensitive data. It allows you to store and manage secrets in a secure and auditable manner, and provides fine-grained access control and secret rotation.
* **CyberArk Conjur**: CyberArk Conjur is a security orchestration platform that provides a centralized system for managing secrets, credentials, and other sensitive data. It allows you to store secrets securely and manage access to them using role-based access control and multi-factor authentication.
* **Google Cloud KMS**: Google Cloud KMS is a key management service that allows you to create, use, and manage cryptographic keys and perform cryptographic operations in a secure manner. It provides a centralized system for storing and managing secrets and allows you to control access to them using IAM.
* **Azure Key Vault**: Azure Key Vault is a cloud-based service provided by Microsoft that allows you to store and manage cryptographic keys, certificates, and secrets. It provides a secure and auditable way to manage secrets and allows you to control access to them using Azure Active Directory.

### Creating a Secret in AWS Secrets Manager (Console)

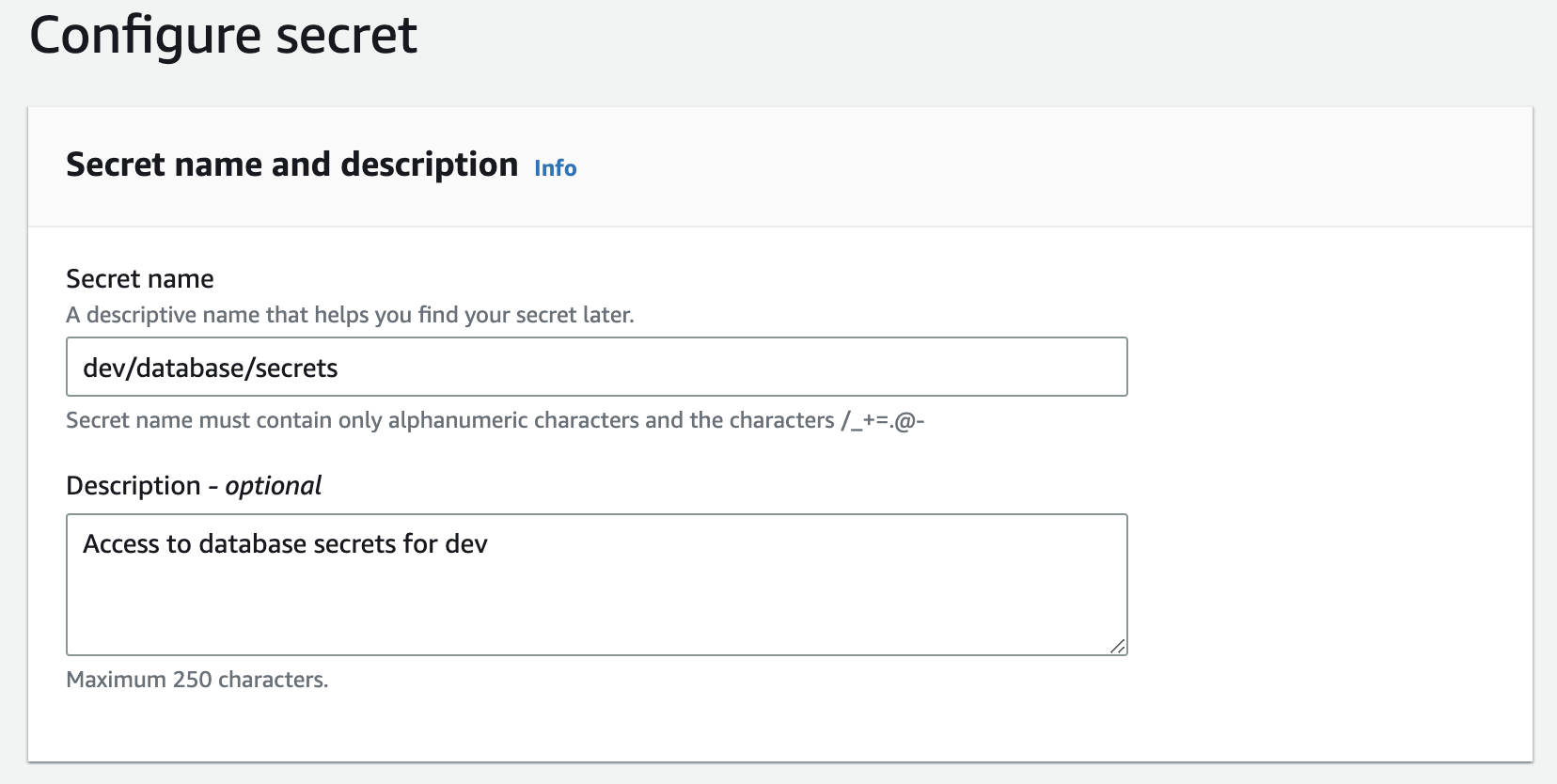
* Sign in to the AWS Management Console and navigate to the Secrets Manager console.
* Click on the "Store a new secret" button.
* AWS allows you to integrate with multiple services for secrets management e.g.:

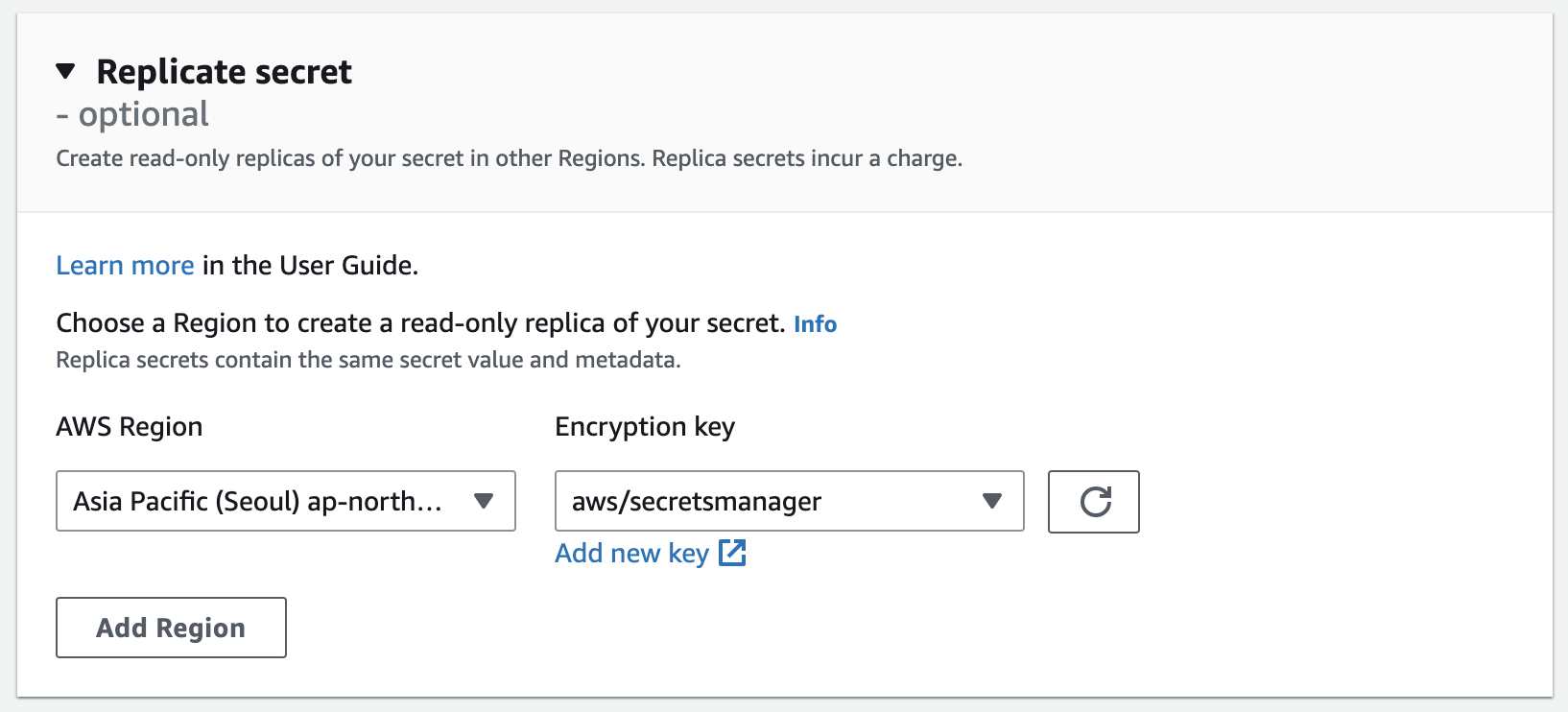


* Choose the type of secret you want to create. In this example, we will create a custom secret, so select "Other type of secrets".
* You will need to provide a name and a value for your secret. For example, you could create a secret with a unique name "luqmanRDSDBPassword" and a value of e.g. "P@ssW0Rd123".

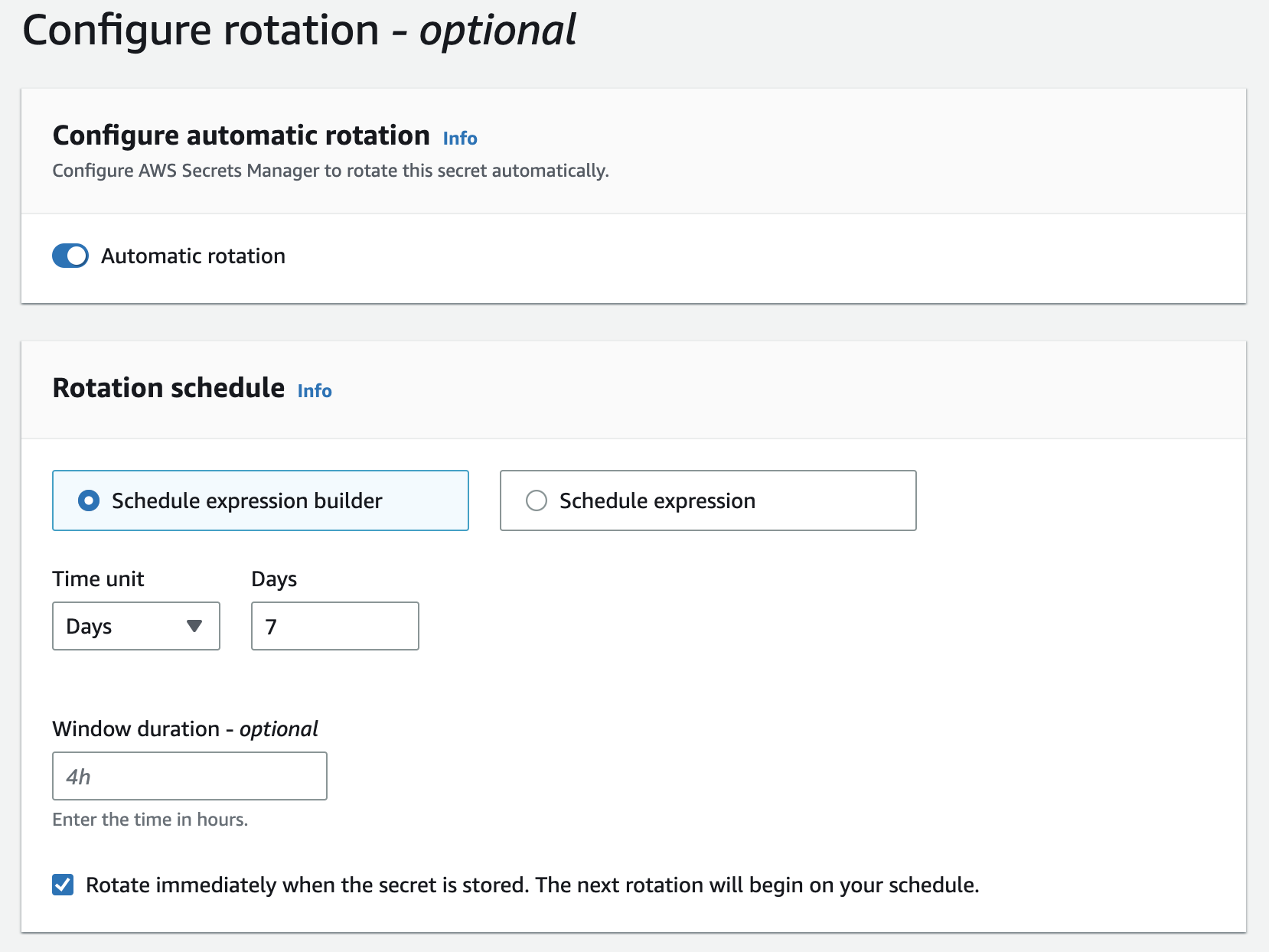


* Keep the encryption key as default: aws/secretsmanager and click Next.
* Configure the secret settings to specify where you store the secret. You can even set the replication of secrets to allow for a more highly-available set up.





* Click "Next" to specify secret rotation settings.
* You are highly recommended to turn on secret rotation. You can do this by specifying the rotation schedule based on your organizational need. For our case, we can keep it to 7 days, but for the purpose of this activity, you can explore a much shorter window e.g. 4 hours to see your secrets rotated.



* Review the secret details and settings. Make sure everything looks correct. You will even get a code snippet to share how you can get started with code to retrieve the secrets. Once you’re ready, click "Store" to create the secret.
* **Challenge: Find out how you can view the secrets from both Console and AWS CLI**

### Creating AWS Secrets Manager secret from CLI

* You can start by exploring in powershell/ terminal the aws secretsmanager help command.
* You can follow this syntax to create a secret:

| aws secretsmanager create-secret --name luqmansecret --secret-string '{"username":"admin","password":"secret123"}' |
| --- |

* If successful, you will receive a similar output message:

{

"ARN": "arn:aws:secretsmanager:ap-southeast-1:255945442255:secret:luqmansecret-OWW9w4",

"Name": "luqmansecret",

"VersionId": "4d5afed3-5dbc-44e1-a644-0bdb9aafff30"

}

* Verify that your secret has been successfully created in the console.
* **Challenge: How can you create a secret and enable automatic secret rotation fo 7 days?**

### Creating AWS Secrets Manager secret from Terraform

* Let’s create a secret from Terraform code snippets.

| resource "aws\_secretsmanager\_secret" "mysecret" {  name = "luqman-secret-2"   tags = {  Environment = "Production"  } }  resource "aws\_secretsmanager\_secret\_version" "mysecret\_version" {  secret\_id = aws\_secretsmanager\_secret.mysecret.id  secret\_string = jsonencode({  username = "admin"  password = "secret123"  }) } |
| --- |

* Here, we're creating a new secret with the name luqman-secret-2 and a tag for the Environment. We then create a new version of the secret using the aws\_secretsmanager\_secret\_version resource type. We specify the ID of the secret we just created using the secret\_id parameter, and we specify the value of the secret as a JSON string using the secret\_string parameter.
* You can customize this code to fit your specific use case, including adding additional versions of the secret, specifying a different type of secret value (such as a binary secret), or adding other options such as a description or rotation policy.
* Once you have your Terraform code set up, you can use the terraform apply command to create the secret in AWS Secrets Manager. Before you can use Terraform to create secrets in AWS Secrets Manager, you will need to set up your AWS credentials and configure Terraform with the appropriate permissions. You can do this by creating an IAM user with the necessary permissions, or by using an existing IAM user with appropriate permissions.
* Try it out!