How to Create and Use AWS Lambda Layers and Secrets Manager

Introduction

AWS Lambda is a service that lets you run code without provisioning or managing servers. You can use AWS Lambda to create functions that perform various tasks, such as processing data, sending notifications, or calling other services. In order to utilize the Code Samples in this Folder, you will need to add FalconPy as a Lambda Layer and attach that layer to your lambda. You will also need to setup and use AWS Secrets manager to protect your API Keys.

What are AWS Lambda Layers?

AWS Lambda Layers are a feature that allows you to package and reuse code or dependencies that are shared across multiple functions. A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. You can upload layers to AWS Lambda or use layers provided by AWS and other AWS Partners. When a function is invoked, AWS Lambda extracts the layer contents into the /opt directory inside the function execution environment. You can reference the layer contents from your function code as if they were in the same directory.

How to Create a Lambda Layer

To create a Lambda layer, you need to follow these steps:

- Create a ZIP archive that contains the files or folders that you want to include in the layer. The files or folders must be in one of the following directories: bin, lib, lib64, python, nodejs, java, or ruby. For example, if you want to create a layer for Python, you can create a python folder and add your libraries or modules to it. (A Sample FalconPy layer has been included for convenience)
- Go directly to AWS Lambda Layers(Services > AWS Lambda > Additional Resources > Layers).
- Create a layer version by specifying the layer name, description, compatible runtimes, and upload the zip. You will receive an ARN that identifies the layer version.
- Add the layer version to your Lambda function with the following instructions

How to Add a Layer to a Lambda Function

To add a layer to a Lambda function, you need to follow these steps:

- Go into the Lambda Function you wish to add the layer to (You should by default be on the “Code” Tab)
- Scroll all the way to the bottom of the screen and you should see a layers section where you can “Add a layer”
- Click on “Add a layer” and follow the instructions to add a Custom Layer and choose the one you uploaded earlier.

- Now your function should be able to use FalconPy (or the other required layers)

What is AWS Secrets Manager?

AWS Secrets Manager is a service that helps you protect secrets needed to access your applications, services, and IT resources. A secret can be any kind of sensitive information, such as a database password, an API key, or a token. You can store secrets in AWS Secrets Manager and control access to them by using fine-grained policies. You can also rotate secrets automatically by using AWS Lambda functions or custom logic. You can retrieve secrets from AWS Secrets Manager by using the AWS SDK or CLI, or by using the built-in integration with other AWS services.

How to Add an API Key to AWS Secrets Manager

To add an API key to AWS Secrets Manager, you need to follow these steps:

- Create a secret by specifying a name, a description, and the API key value. You can use the AWS CLI or AWS Secrets Manager console to create a secret. You will receive an ARN that identifies the secret.
- Optionally, tag the secret with key-value pairs to help you organize and manage your secrets. You can use the AWS CLI or AWS Secrets Manager console to add tags to a secret.
- Make sure to add the secret as an “Other type of secret” and add 2 key/calue pairs. One that is “clientid” with the API Client ID, and one that is “clientsec” with the Client Secret

How to Grant Permissions to a Lambda Function to Access a Secret

To grant permissions to a Lambda function to access a secret, you need to follow these steps:

- Find the ARN of the secret that you want to access. You can use the AWS CLI or AWS Secrets Manager console to list the available secrets and their ARNs.
- Find the ARN of the Lambda function that needs to access the secret. You can use the AWS CLI or AWS Lambda console to list the available functions and their ARNs.
- Create a policy that allows the Lambda function to get the secret value from AWS Secrets Manager. You can use the AWS CLI or AWS IAM console to create a policy. The policy should have the following statement:
- {"Effect": "Allow", "Action": "secretsmanager:GetSecretValue", "Resource": "arn:aws:secretsmanager:region:account-id:secret:secret-name"}
- Attach the policy to the Lambda function's execution role. You can use the AWS CLI or AWS IAM console to attach a policy to a role. The role should have the following trust relationship:
- {"Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}
- Test your function to verify that it can access the secret value from AWS Secrets Manager. You can use the AWS CLI or AWS Lambda console to invoke the function and check the logs or output.