Module 29
AWS Secrets Manager

Overview

Almost any application is likely to have sensitive data it needs to deal with and manage as part of its normal operation. Here is a list of some common sensitive parameters that you have likely encountered in writing applications:

- Connection parameters to databases: opening a connection to a database requires a host address, a username, a password, among other parameters. Below is an example connection string to PostgreSQL where many sensitive parameters need to be specified (the name of the database, the host it is running on, the database username/password to connect as, etc.)

```
User ID=root; Password=myPassword; Host=localhost; Port=5432; Database=myDataBase; Pooling=true; Min Pool Size=0;
Max Pool Size=100; Connection Lifetime=0;
```

- API keys: If you are using an external web service, you are likely to obtain some access key from the web service vendor. Such a key is not free and is tied to your account (some vendors give you a free trial key to try their service – such keys usually expire after a few weeks or so or are limited to some limited number of calls per day). But once you deploy your application you will need a permanent key which is not free. And you are likely to be charged based on your usage pattern. As a result, such a key is a sensitive piece of information that you own and need to manage in your application.

- Other application secrets: For example if your application is say using Auth0 (https://auth0.com/) for authentication and other services, you will have many secrets to deal with (application id, client secret, etc.). For example, below is C# code that requests an access token from Auth0. You get this access token and you use it in subsequent calls to other Auth0 management services. Notice that you need to pass some sensitive data.

```
var client = new RestClient("https://YOUR_DOMAIN/oauth/token");
var request = new RestRequest(Method.POST);

request.AddHeader("content-type", "application/x-www-form-urlencoded");
request.AddParameter("application/x-www-form-urlencoded",
                     "grant_type=client_credentials&client_id=YOUR_CLIENT_ID&client_secret=YOUR_CLIENT_SECRET&audience=YOUR_API_IDENTIFIER",
                     ParameterType.RequestBody);
IRestResponse response = client.Execute(request);
```

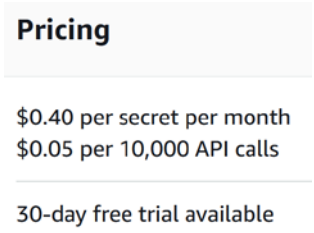
And obviously, you don't want to push the above code to Git with some of your secrets clearly visible. Same applies to database connection string parameters or any other secrets. The above are just a few examples. There are many more situations where your application needs to deal with data without revealing it.

In this module you will look at AWS Secrets Manager which is a service for managing sensitive data that an application needs (Microsoft Azure has an equivalent service known as Azure Key Vault).

Steps to Do

1. Go to the AWS Secrets Manager page and read the introduction page.

2. Go to Canvas and login to the AWS Console. Then go to the **Secrets Manager** service. Notice the pricing (it costs about $5 per year to store a single secret). You are also charged 5 cents for every 10,000 API calls to retrieve the secret.

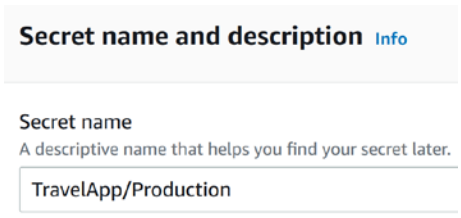


3. Click the **Store a new secret** button.

4. Choose the **Other type of secret** option. Then enter a key and a value (the key is the name you want to give your secret). Let's assume that our application uses the Google map service and Auth0 for authentication, authorization, and other services. As a result, we have two sensitive pieces of information we need to access in our code.

5. Leave the Encryption key setting as is, and click **Next**.

6. For Secret name, enter TravelApp/Production. Leave the Description field empty. Click **Next**.

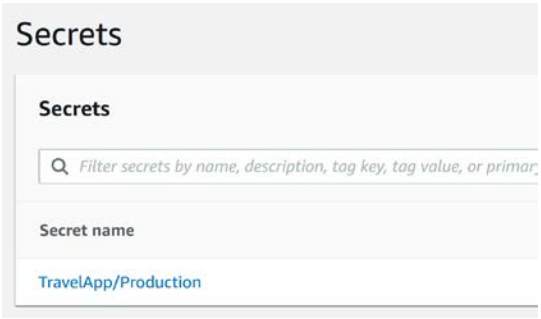


7. Leave everything as-is on the Secret rotation section. Click **Next**.

8. On the last screen, AWS gives you sample code (of different languages) on how to retrieve your secret. Select **C#**. Copy the code and paste it in Notepad. Save the file (you will use it soon).

9. Click the **Store** button.

   NOTE: I had to browse back, then refresh the page to see the new secret I created being listed in my secrets:



10. Open Visual Studio and create a solutions (name it “ManageSecrets”). Under this solution, create a Console application (C# Core) and name it “SecretsRetieval”).

11. Add to your project the needed libraries (AWSSDK.SecretsManager, AWSSDK.Core).

12. Complete the code of Program.cs to retrieve the 2 secrets you stored (GOOGLE_MAP_API_KEY and AUTH0_CLIENT_SECRET). Print their values to the Console and verify that the values match what you entered on the AWS Console.

    Follow the same pattern with using other services. That is, you first need to create a client object to the service (AmazonSecretsManagerClient), then a request object, then you call some service method and get back a response object (see sample code you pasted in Notepad).

What to Submit

Nothing to submit for this module