Create your first Resource

Congratulations! You've recently joined a company that uses AWS CloudFormation extensively to manage its infrastructure.

As part of your onboarding process, you will be diving into hands-on activities that will familiarize you with the team's workflows involving CloudFormation and the AWS Cloud Development Kit (CDK). This session is designed to equip you with practical skills in defining and deploying AWS resources using these powerful tools.

Provisioning a Virtual Machine in the Cloud

CloudFormation
AWS CloudFormation is a managed service that allows you to define and provision AWS infrastructure using code, enabling automated and repeatable deployments with program templates.

An AWS CloudFormation *template* is a declaration of the AWS resources that make up a stack. The template is stored as a text file in either JavaScript Object Notation (JSON) or YAML format. They are simple text files that can be created with any text editor, and managed as part of a source control system with other source code repositories.

A *stack* is composed of a set of values that define how AWS services are launched. The only component required within a stack are *Resources* which lists the services you want to deploy.

Writing your first CloudFormation Template

Let's build our first CloudFormation template to create a single S3 bucket for storing data objects. In the following steps, do not change any of the default values or settings unless instructed to do so.

From the VSCode editor, create a new file called **s3-cloudformation.yaml**. Cut and paste the following two lines into the file:

Line 1 defines the version that should be used by CloudFormation to interpret the template. It is an optional component and the only acceptable value is `2010-09-09` (the latest template format version). This line should always be the first line in the configuration file.

The second line defines the top level **Resources** identifier. As mentioned previously, it is the only mandatory component as it contains all of the AWS service resources for that CloudFormation will create.

Now that we have some basic elements defined, let's add a few additional lines. Copy and paste the following lines into the file after the first two lines:

```
1     S3Bucket:
2         Type: 'AWS::S3::Bucket'
3         Properties:
4             BucketName: "our-first-s3-bucket-workshop"
```

The first line specifies a tag you define for the S3 bucket resource within the CloudFormation template. It is a logical identifier you can refer to througout the template. In this example, we will use the value *S3Bucket*.

Line 2 specifies the type of AWS resource to create. We are creating an S3 bucket, a resource widely used in AWS for storing and managing objects like images, backup files, and other types of data. Detailed descriptions for all of the AWS resources supported by CloudFormation can be found in the CloudFormation documentation.

The third line allows you to specify additional settings or *Properties* for the S3 resource. The is a commonly used parameter for most resources.

Line 4 declares the first property that we have defined for the S3 bucket. The property names the S3 bucket `our-first-s3-bucket-workshop`.

IMPORTANT
The bucket name used in the example is for *illustrative purposes only*. S3 Bucket names ***must be globally unique***, and for the duration of this workshop you will need to assign names that satisfy this requirement. We suggest adding the last 4-digits of the AWS account number to the end of a bucket as this will increase the likelihood that the name will be unique. Consider other tips like your school's name, or your location as part of a bucket name also.

In your `s3-cloudformation.yaml` file, replace the bucket name example on line 4 above with a globally unique bucket name value. NOTE: Be sure to insert the name in between the quotes (" ").

The template now has the minimum required decalarations to deploy a bucket resource. Let's proceed by saving the file, and using the CloudFormation service to deploy the resource in the AWS account.

As part of the AWS account environment, you should already have an existing S3 bucket. Run the following command in the VSCode terminal to list the S3 bucket:

```
aws s3 ls
```

The output sequence will look something like this:

```
ubuntu@dev:/workshop$ aws s3 ls
2024-07-27 13:58:11 vscode-server-ssmlogbucket-XXXXXXXXXX
```

Where `vscode-server-ssmlogbucket-XXXXXXXXXX` is the name of the bucket. Copy the `s3-cloudformation.yaml` into the bucket. This will allow us to reference the file later as input to the CloudFormation operation:

```
aws s3 cp /workshop/s3-cloudformation.yaml s3://vscode-server-ssmlogbucket-XXXXXXXXXX
```

Be sure to replace the sample name of the S3 bucket above with the actual bucket name from your terminal output. You will see a return statement if the upload of the object is successful:

```
upload: ./s3-cloudformation.yaml to s3://vscode-server-ssmlogbucket-XXXXXXXXXX/s3-cloudformation.yaml
```

You can confirm the upload by running the next command to list the objects inside the S3 bucket:

```
aws s3 ls vscode-server-ssmlogbucket-XXXXXXXXXX/
```

Again, be sure to use the actual name of the S3 bucket in your environment for the above commands.

Deploy your first CloudFormation Template

To deploy our CloudFormation stack, we have two options:

a) We can proceed by **uploading the YAML file manually** using the AWS management console.
b) We can **use the AWS CLI** to execute commands to deploy the stack programatically.

Let's review both options before deciding. **NOTE:** *Choose one of the two options, and make sure you only execute the deployment steps once to avoid any potential errors.*

▶ **AWS Console**

▶ **AWS CLI**

You have successfully deployed a CloudFormation stack and created an S3 bucket!!!

Using Your Own AWS Account
As a best practice, it is always recommended to delete the stack when the resources created are no longer being used to avoid unnecessary charges.

More code settings for the CloudFormation template

Your team wants an S3 bucket to store company data. They request that you to implement a security mechanism to ensure the data is encrypted and protected from accidental deletions.

You need to create a new template with updates to satisfy the required configurations.

Create a duplicate copy of the CloudFormation template, giving the new file a separate name (e.g., `s3-cloudformation-2.yaml`).

Edit the **Properties** section with the following statement (NOTE: Define a new unique value for the name of the S3 bucket you will create):

```
1   AWSTemplateFormatVersion: "2010-09-09"
2   Resources:
3       S3Bucket:
4           Type: 'AWS::S3::Bucket'
5           Properties:
6               BucketName: "our-second-s3-bucket-workshop"
7               BucketEncryption:
8                   ServerSideEncryptionConfiguration:
9                   - ServerSideEncryptionByDefault:
10                      SSEAlgorithm: AES256
```

Line 7 above defines the **BucketEncryption** property that specifies the encryption settings for the S3 bucket. Adding this will allow a bucket to set additional protections on the stored data.

Lines 8 thorugh 10 defines the **ServerSideEncryptionConfiguration** property which sets a server-side encryption configuration. We want to use the **AES-256** algorithm to encrypt our data. AES-256 is a symmetric encryption algorithm that provides strong encryption and is a widely used security standard for encrypting data.
NOTE: Server-side encryption is automatically applied to new objects stored in the bucket by default, however you can include this statement as a security best practice to implement additional encryption settings. For example, S3 buckets can be encrypted using encryption keys.

Now that we have added a change for data protection, let's enable object versioning. *Versioning* in Amazon S3 allows for multiple variants of an object in the same bucket. It is a common data protection practice, and you can use the feature to preserve, retrieve, or restore any version of all objects stored in an S3 bucket.

Add two additional lines to the end of the *Resources* statement in the new file,

```
1   AWSTemplateFormatVersion: "2010-09-09"
2   Resources:
3       S3Bucket:
4           Type: 'AWS::S3::Bucket'
5           Properties:
6               BucketName: "our-second-s3-bucket-workshop"
7               BucketEncryption:
8                   ServerSideEncryptionConfiguration:
9                   - ServerSideEncryptionByDefault:
10                      SSEAlgorithm: AES256
11              VersioningConfiguration:        <--- add this line to the file
12                  Status: Enabled             <--- add this line to the file
```

Line 11 declares a property to enable object versioning. Make sure to set the Status to **Enabled** as shown in **Line 12**.
Save the file. Reminder, that if you are using the management console to execute the stack, be sure to copy the newly created file to the S3 bucket used in the previous workflow.

Deploy an updated version of the CloudFormation template

Let's deploy the new version of the template using the same steps previously executed. **NOTE:** *Choose either step, but make sure you only complete the chosen step once. Be sure to provide a new stack name (e.g., S3Stack-2), and point the configuration to the new template files.*

▶ **AWS Console**

▶ **AWS CLI**

Conclusion of workflow

You have completed the first workflow!

What did you accomplish in this workflow?

- **AWS CloudFormation** is a managed cloud service that allows developers and users to create templates for deploying AWS resources.
- You followed the steps to **deploy** a **CloudFormation Stack** using the **AWS management console** or the **AWS CLI**.
- Two stacks were deployed using several values to configure S3 buckets.
- The second S3 bucket was created using security best practices and versioning using CloudFormation.

Previous Next