CloudFormation Conditions

Welcome to the third workstream in the CloudFormation workshop. In this module, we will demonstrate how to use conditions to create resources based on parameters and their values. This technique helps developers create more flexible cloud infrastructure.

CloudFormation Conditions

A ***Condition*** in CloudFormation allows you to manage resources when certain requirements are met. This can be based on parameter values or existing resources. For example, we can set a condition to create the name of a resource *based on the environment it is being used for*. Let's look closely at an example.

Let's create a new file (e.g., *s3-cloudformation-condition.yaml*) and insert the following code:

```
1   AWSTemplateFormatVersion: "2010-09-09"
2   Description: Basic CloudFormation template to demonstrate conditions.
3
4   Parameters:
5     EnvironmentType:
6       Description: The environment where the stack is deployed (prod or preprod)
7       Type: String
8       AllowedValues:
9         - prod
10        - preprod
11      Default: preprod
12
13  Conditions:
14    IsProduction:
15      Fn::Equals:
16        - !Ref EnvironmentType
17        - prod
18    IsPreProd:
19      Fn::Equals:
20        - !Ref EnvironmentType
21        - preprod
```

In this top section of the template, we are defining the following,

- The **Parameters** section starting on Line 4 that contains a single `EnvironmentType` variable. This will define either *prod* value for a production build, or *preprod* for a pre-production environment. We will set the default value to `preprod`.
- In **Conditions** starting on Line 13, we will evaluate the EnvrionmentType parameter and set `IsProduction = prod` for a value of prod. If the parameter evaluates to *preprod*, then we will set `IsPreProd = preprod`.

Add some additional code by cutting and pasting the `Resources:` section to the end of the file below the `Conditions:`

```
1   Resources:
2     ProdBucket:
3       Type: AWS::S3::Bucket
4       Condition: IsProduction
5       Properties:
6         BucketName: !Sub "${AWS::Region}-${AWS::AccountId}-bucket-prod"
7         VersioningConfiguration:
8           Status: Enabled
9
10    PreProdBucket:
11      Type: AWS::S3::Bucket
12      Condition: IsPreProd
13      Properties:
14        BucketName: !Sub "${AWS::Region}-${AWS::AccountId}-bucket-preprod"
15        VersioningConfiguration:
16          Status: Suspended
```

Here, we add more conditions to create a *production* bucket, if `!IsProduction` evaluates to true, or create a preproduction resource if `IsPreProd` evaluates to true. Save the file, and before executing the stack, answer the following question,

Q: Based on the declarations of the template, which environment will the S3 bucket be created for?

▶ **Answer**

Execute the stack, using from the VSCode terminal:

```
aws cloudformation create-stack --stack-name S3Stack-condition --template-body file://s3-cloudformation-condition.yaml
```

Challenge: Use the above instructions to create a bucket for the production environment using the same CloudFormation ***without*** making any edits to the file! Remember to use a new stack name, and recall how we used the `parameters.json` file in the previous workflow.

▶ **Challenge Solution**

Conclusion of workflow

What did you do in this workflow?
In this workflow, you learned how to use conditions in your CloudFormation template to efficiently re-use code and to support multiple environments. Conditions make it easier to develop environment-specific configurations, and better manage resources and outputs.

Previous Next