Infrastructure as Code

Infrastructure as Code - IaC for short – is a method for **automating the process of creating, updating, or deleting your infrastructure in the cloud**. The infrastructure in the cloud typically consists of servers, databases, network resources such as subnets, security groups, internet gateways etc.

Let's say you are using Amazon Web Services (AWS) for your cloud-based application. The infrastructure for your application in the AWS cloud may consist of Amazon EC2 instances, a database service such an Amazon Aurora, AWS Lambda to handle your business code asynchronously, build and deployment pipelines for various environments etc.

You may have multiple environments such as test, staging and production for deploying your application. Each of these environments usually consist of the same resources such as Amazon EC2 instances, databases, network resources etc. but in a scaled-up or a scaled-down version.

Provisioning, managing, and even deprecating infrastructure in the cloud is a **costly activity in terms of human capital**. Furthermore, repeat attempts to build and modify environments manually can be fraught with errors.

Whether working from prior experience or a well-documented run-book, the tendency for a human to make a mistake is a statistical probability.

Now, consider this:

- you have an option to automate the task of creating a complete environment
- it is done in such a way that you can repeat it consistently and effortlessly

AWS provides a managed service called AWS CloudFormation for this exact need - to implement Infrastructure as Code for your cloud infrastructure.

With **AWS CloudFormation**, you can define your infrastructure in the form of templates. A single template may consist of a part, or the whole environment. More importantly, that **template can be used repeatedly to create the same, identical environment, time and time again**.

Benefits of Infrastructure as Code

When you start using AWS CloudFormation templates to define your cloud infrastructure, you get the following benefits:

- Templates are versioned and managed just like application source code
- Infrastructure is repeatedly and reliably created, turned off, and re-created
- Infrastructure can be created as needed to test the latest version of your application
- Cost savings, creation of multiple environments, and creation of identical environments for multiple customers.

Basic Structure of an AWS CloudFormation Template

You can write a CloudFormation template in either **YAML or JSON** format. The structure shown below is in the **YAML** format.

```
AWSTemplateFormatVersion: 2010-09-09
Description: 'Describe what this will create'
Parameters:
  MyParameter
    Type: String
Mappings:
  RegionMap:
    'us-west-2':
      'ami-abc123ab'
Conditions:
  EnvIsProd:
Resources:
  myEc2Instance:
      Properties:
Outputs:
  myOutput:
    Value: myVal
      Export:
      Name: myExportVal
```

Below is a brief description of what each section means:

Format Version: AWS CloudFormation template version

Description: A text string to describe the template

Parameters: Inputs into template

Mappings: Static variables; Key-Value pairs

Conditions: Controls for if-and-when certain resources are created or updated

Resources: AWS resources to create

Outputs: Values of custom resources created by template (URLs, username, etc.)

Create an Amazon S3 Bucket via CloudFormation

The YAML template below creates an Amazon S3 bucket. The name of the bucket is *sample-bucket*. The bucket is **publicly accessible**. In the **Outputs** section of the template, you can see that the name of the bucket is being referenced. You can see the bucket name in the AWS console under the AWS CloudFormation service.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3Bucket:
    Type: 'AWS::S3::Bucket'
    DeletionPolicy: Retain
    Properties:
      BucketName: sample-bucket
      AccessControl: PublicRead
Outputs:
  BucketName:
    Value: !Ref S3Bucket
    Description: Name of the sample Amazon S3 bucket
```