Overview

Infrastructure as Code (IaC) 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
- Network resources such as VPC, subnets, security groups, etc.
- A database service such as Amazon Aurora
- AWS Lambda to handle your business code asynchronously
- Build and deployment pipelines for various environments

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 deprecating infrastructure in the cloud is a costly activity in terms of human capital. Furthermore, repeat attempts to build and modify environments manually can lead to 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.
- The automation is done in such a way that you can repeat it consistently and effortlessly.

AWS provides a managed service called **AWS CloudFormation** 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, you can use the template repeatedly to create the environment.

Benefits of Infrastructure as Code

Using AWS CloudFormation templates to define your cloud infrastructure, gives you the following benefits:

- You can version-control the templates. Templates are versioned and managed just like application source code
- You can repeatedly and reliably create, turn-off, and re-create the cloud infrastructure.
- You can create infrastructure as needed to test the latest version of your application.
- You can save costs by creating multiple, identical environments for multiple customers.

CloudFormation template structure

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: 'Brief description of the template'
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 in the YAML template means:

Section	Description
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 creating / updating certain resources
Resources	AWS resources to create
Outputs	Values of custom resources created by template (URLs, username, etc.)
Export	Export values from the template

Example

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 by using the `!Ref` keyword. You can view the bucket name in AWS console under the AWS CloudFormation service.

```yaml
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
```