CloudFormation

Infrastructure as Code

Content Prepared By: Chandra Lingam, Cotton Cola Designs LLC

amazon web services

CloudFormation

- Infrastructure as Code
- Simplify Infrastructure Management
- Replicate Infrastructure
- Progress Notification using SNS
- Free

Concepts

Definition	Description
Template	Blueprint in YAML or JSON for building AWS resources
Stack	Collection of resources created when instantiating a template
Change Sets	Summary of proposed changes to a stack – when updating a stack

YAML Introduction

Five Minute Intro to YAML

YAML is easy to read

In exams, you will see questions using JSON Code

Be comfortable reading in both formats!

Template Structure

Section	Description
Format Version	CloudFormation Template Version that the template conforms to (optional) – currently, only “2010-09-09” is allowed
Description	Friendly description (optional)
Metadata	Additional information about a template (optional)
Parameters	Input values that can be passed to a template
Mappings	Lookup table of key-value pairs
Conditions	Conditionally create resources
Transform	Refer to a snippet of template stored separately
Resources	Specifies resources and their properties
Outputs	Values returned by the stack. Can be referenced by another stack

Demo 1 – S3 Bucket

- Create S3 Bucket
- S3 Bucket Resource
- CloudFormation creates a unique bucket name as bucket name was not explicitly specified
    - Stack Name (first try): s3-bucket-defaultname (try 1)
    - Stack Name (second try): s3-bucket-defaultname2
- Bucket is created in the same region where you create the CloudFormation stack

`Script: 1. S3BucketCreate.yaml`

Resource

- Logical ID – Logical name of the resource. Unique within the template. Refer the resource in other parts of the template using this name.
- Physical ID => Actual ID of the resource created
- Example – EC2 Instance
  - Logical ID: DemoEC2Instance (referred with this name in the template)
  - Physical ID: i-34xyab51 (actual resource created in AWS)

Demo 2- Update S3 Bucket Stack

- Update existing stack
- Change the bucket name
- View Change Set
    - Drops the bucket and creates a new one
- Bucket name is hardcoded and that means only stack can be created!

`Script: 2. S3BucketCreateWithName.yaml`

Demo 3 – User Provided Bucket Name

- User Provides the bucket name
  - Parameters Section is used for capturing input parameters
  - Parameters are referred in rest of the script using logical name
- CloudFormation Script attempts to create a bucket with the specified name
  - Stack Name: s3-bucket-user-provided
  - Bucket Name: hello-bucket-chandra-201801

```
Script:
3. S3BucketCreateWithUserSpecifiedName.yaml
```

Demo 4 – Deletion Policy - Retain

- Default behavior
  - When stack is deleted, resources created are also deleted

  Note: if S3 bucket is not empty, deletion fails
- DeletionPolicy allows you to keep a resource (for example database or S3 bucket or any critical EC2 instance)

Script:

4. S3BucketCreateWithUserSpecifiedNameDeletionPolicy.yaml

Intrinsic Functions

Built-in functions to manage stacks

Function	Purpose
Fn::Base64	Returns Base64 encoded string
Fn::FindInMap	Returns value corresponding to keys in a two level map that is defined in mappings section. Example: AMI IDs by region
Fn::GetAtt	Returns value of an attribute from a resource in the template. Example: DNS Name of an Elastic Load Balancer
Fn::GetAZs	Returns a list of availability zones for the specified region
Fn::ImportValue	Import Value from another stack. Example: Stack A exports values in Output section and Stack B imports the value
Fn::Join	Appends a set of comma separated values into a single value, separated by the specified delimiter

Intrinsic Functions

Function	Purpose
Fn::Select	Returns a single object from a list of objects
Fn::Split	Split a string into a list of string values based on delimiter specified
Fn::Sub	Substitute the value of variables in an input string
Ref	Returns the value of the specified parameter or resource • When parameter's logical name is specified, value of the parameter is returned • When resource's logical name is specified, value of Physical ID of the resource is returned
Condition	Conditionally create stack resources using Fn::If, Fn::Equals, Fn::Not

Pseudo Parameters

Pre-defined parameters provided by CloudFormation

- AWS::Region
- AWS::AccountId
- AWS::StackName
- And so forth

Demo 5 – Auto Scaling, ELB Web Server

Scalable, Load Balanced Setup

Pick us-west-2 region

Stack Name: auto-scaling-elb-demo

Pick: t2.micro

Tag: Complete Stack Deployment

```
Script (AWS provided example):
5. All Inclusive demo.yaml
```

Demo 6 – Resource Group to view all resources

- Create a resource group for “Complete Stack Deployment” tag
- Display all the resources that were created by the CloudFormation stack
- Delete the Stack to remove all resources

Best Practices

CloudFormation Best Practices

- Organize by lifecycle and Ownership (Example: Website versus database)
- Use Cross-Stack References to Export Shared Resources
- Use IAM to Control Access. (User credentials, CloudFormation Service role)
- Verify Quotas
- Reuse Template to replicate stack in multiple environments

Best Practices

CloudFormation Best Practices

- Use Nested Stacks to reuse common template patterns
- Do not embed credentials in templates – use input parameters with NoEcho property
- Use AWS specific Parameter Types (for example EC2 Key pair)
- Use Parameter Constraints
- Use AWS::CloudFormation::Init to deploy software applications
- Manage all stack resources using CloudFormation