CloudFormation Section 1

**CloudFormation Intro:**

**CloudFormation-** An AWS service that helps you model and configure AWS resources using code constructs

* An Infrastructure as Code service
* Architecting AWS in code and deploying it within minutes
* The ability to manage your deployed infrastructure through CloudFormation as well
* Updates to your infrastructure can be fast and automated
* A multi-tiered architecture can happen in 1 template
* Or you can break templates up by departments

**CloudFormation Template-** Describes your resources. Includes dependencies

* JSON or YAML
* Reusable
* Minor adjustments allow you to deploy your stack to a whole new region which is good for DR

**CloudFormation Stack-** Managing all your resources. The physical resources that are created from deploying a template.

Use case

Deploy a testing environment with a template. That creates a stack in which you can use to test out your deployment. After you are finished you can delete the stack which allows you to save money and delete all the resources you created.

**CloudFormation Essentials:**

CloudFormation is IaC.

CloudFormation Templates house the code. They describe all your resources. Written in JSON or YAML.

Templates create stacks. Stacks are created by templates.

If a template is well formed you can create stacks within minutes. You should monitor the creation of your stacks.

**Change Sets-** organized safe way to make changes to your stacks and you get to review your changes before they are actually implemented.

**Introduction to JSON:**

**JSON**- Java Script Object Notation is a lightweight data-interchange format.

* Used extensively by the CLI and elsewhere throughout AWS
* Easily readable and resembles, due to the use of {}, Javascript

**Pros**

* Looks like some popular languages
* Used extensively throughout AWS

**Cons**

* No inline comments

**Introduction to JSON:**

* The “Key” component of JSON are Key/Value pairs
* Name/value pairs are separated by a “:”
* Multiple name/value pairs are separated by commas “,”
* The name is a sting enclosed within double quotes
* Example: “t2.micro”
* Value can be a string, but also: Number, Boolean, object, array, or null
* Objects are enclosed in {} as such
* myObject = {“name”: “CloudFormation Deep Dive”, “level”:101}
* Square brackets [] indicate arrays:
* [“t2.micro”, “t2.small”, “t2.large”]
* Certain caveats with JSON in CloudFormation. Key/value pairs, even the value is enclosed in quotes, as such:
* “MinLength”: “9”, “MaxLenght”: “18”
* Null is not used in the traditional sense in CloudFormation

**Objects:**

Empty Object {]

Single value object { key : value}

Multiple value object { “key1” : value , “key2”: value,}

var myObject = {

“first”: “John”,

“last”: “Doe”,

“age”: “35”,

};

**Arrays:**

[] empty array

[ value ] single element array

[ value , value ] multiple value array

**Introduction to YAML:**

**Introduction to YAML:**

* YAML is a superset of JSON. You can parse JSON with a YAML parser
* YAML allows inline comments
* Comments begin with # which is great in collaborative environments
* Human readable
* Designed for data
* Indention conveys structure (it has no {}[])
* Whitespaces and lines have meaning in YAML
* Do not use tables indention is a key part of YAML
* Name/value pairs – name: value the value does not need quotes but you can use them
* Firstname : Johns
* Middlename : Joseph
* No comma is needed, the new line separates the elements
* 2 ways to represent an array
  + Inline Array: Instance :
  + [t2.micro,t2.small,t2.large]
  + Indented array:
  + Instance
    - T2.micro
    - T2.small
    - T2.large

**Syntax Keys:**

* Indentation – Fixed indentation scheme to represent relationships between data layers
* Colons – key/value pairs are separated by colons
* Dashes – to represent lists of items, a single dash followed by a space is used
* Key/value pairs: name : John

**CloudFormation and IAM:**

**CloudFormation and IAM:**

* Use IAM to control what users can do with AWS CloudFormation. Can they view stack templates? Can they create, update, delete stacks?
* When you start building stacks, you can introduce a lot of different services and resources. You need to control which services and resources the user can access through CloudFormation via IAM
* You can specify who can terminate DBs, EC2 instances, update VPCs through CloudFormation
* CloudFormation should be governed by IAM and not be an “Open Gate” for users to do as they wish.
* DO NOT create users just to work with CloudFormation and bypass your IAM procedures
* Users who can create/delete stacks must also have appropriate permissions to the resources in the stack
* Users who use the console require additional permissions
* Use conditions to specify when a policy is in effect
* Specify a service role that allows CloudFormation to create, update, or delete your stacks resources