CloudFormation

What is CloudFormation?

- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 machines using this security group
  - I want two Elastic IPs for these EC2 machines
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines

- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

CloudFormation - brief overview

AWS CloudFormation is an Infrastructure as Code (IaC) service provided by Amazon Web Services (AWS). It enables users to define and provision AWS infrastructure and resources in a declarative way using templates, known as CloudFormation templates. This allows for the automated creation, management, and updating of AWS resources.

Key features and functionalities of AWS CloudFormation include:

1. **Templates:** CloudFormation templates are written in JSON or YAML format and describe the AWS resources and their configurations, dependencies, and relationships needed for an application or system.

2. **Declarative Infrastructure Management:** With CloudFormation, infrastructure is defined declaratively, specifying the desired state of resources rather than the sequence of steps to create them. This allows for consistent and reproducible deployments.

3. **Automated Resource Provisioning:** CloudFormation automates the provisioning and configuration of AWS resources, handling the resource creation, updates, and deletions based on the template definitions.

4. **Stacks:** Templates are organized into stacks, which are collections of AWS resources managed as a single unit. Stacks can be easily created, updated, or deleted, and CloudFormation manages the resources' lifecycle accordingly.

5. **Change Sets:** Before making changes to a stack, CloudFormation allows you to preview the changes using Change Sets. This enables reviewing proposed changes before implementing them.

6. **Resource Dependency Management:** CloudFormation manages the dependencies between resources, ensuring that resources are provisioned in the correct order based on their dependencies.

7. **Cross-Account and Cross-Region Support:** CloudFormation supports provisioning resources across multiple AWS accounts and regions, allowing for centralized management of infrastructure.
8. **Integration with AWS Services:** It integrates with various AWS services, enabling the provisioning of a wide range of resources like EC2 instances, S3 buckets, RDS databases, load balancers, IAM roles, and more.

Using CloudFormation, users can codify their infrastructure requirements, enabling automation, repeatability, and version control of infrastructure changes. This streamlines the process of deploying and managing AWS resources and infrastructure in a consistent and efficient manner.

Benefit of AWS CloudFormation

- Infrastructure as code
  - No resources are manually created, which is excellent for control
  - The code can be version controlled for example using git
  - Changes to the infrastructure are reviewed through code

- Cost
  - Each resources within the stack is stagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)

- Separation of concern: create many stacks for many apps, and many layers. Ex:
  - VPC stacks
  - Network stacks
  - App stacks

- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation

CloudFormation Stack Designer

- Example: WordPress CloudFormation Stack
- We can see all the **resources**
- We can see the **relations** between the components

How CloudFormation Work?

- Templates have to be uploaded in S3 and then referenced in CloudFormation
- To update a template, we can't edit previous ones. We have to re-upload a new version of the template to AWS
- Stacks are identified by a name
- Deleting a stack deletes every single artifact that was created by CloudFormation.

Deploying CloudFormation templates

- Manual way:
  - Editing templates in the CloudFormation Designer
  - Using the console to input parameters, etc

- Automated way:
  - Editing templates in a YAML file
  - Using the AWS CLI (Command Line Interface) to deploy the templates
  - Recommended way when you fully want to automate your flow

CloudFormation Building Blocks

Templates components (one course section for each):

1. Resources: your AWS resources declared in the template (MANDATORY)
2. Parameters: the dynamic inputs for your template
3. Mappings: the static variables for your template
4. Outputs: References to what has been created
5. Conditionals: List of conditions to perform resource creation
6. Metadata

Templates helpers:

1. References
2. Functions

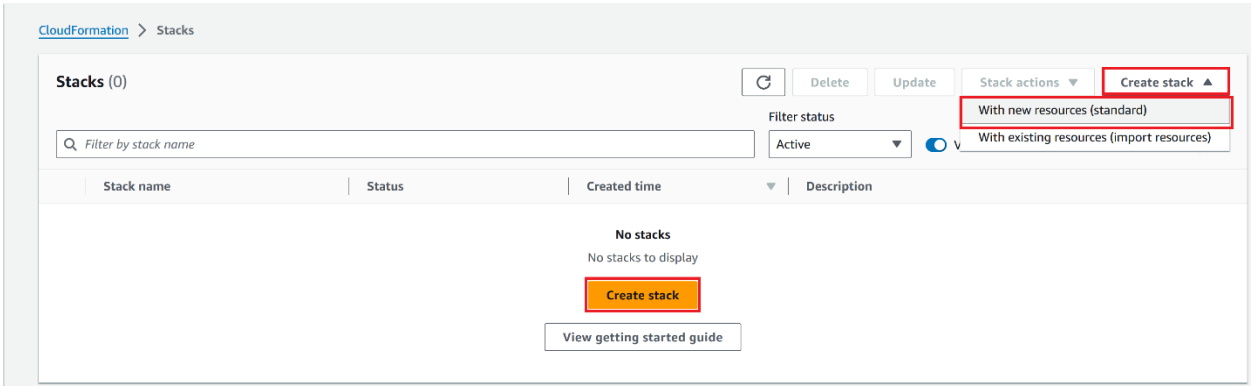
YAML (Yet Another Markup Language)

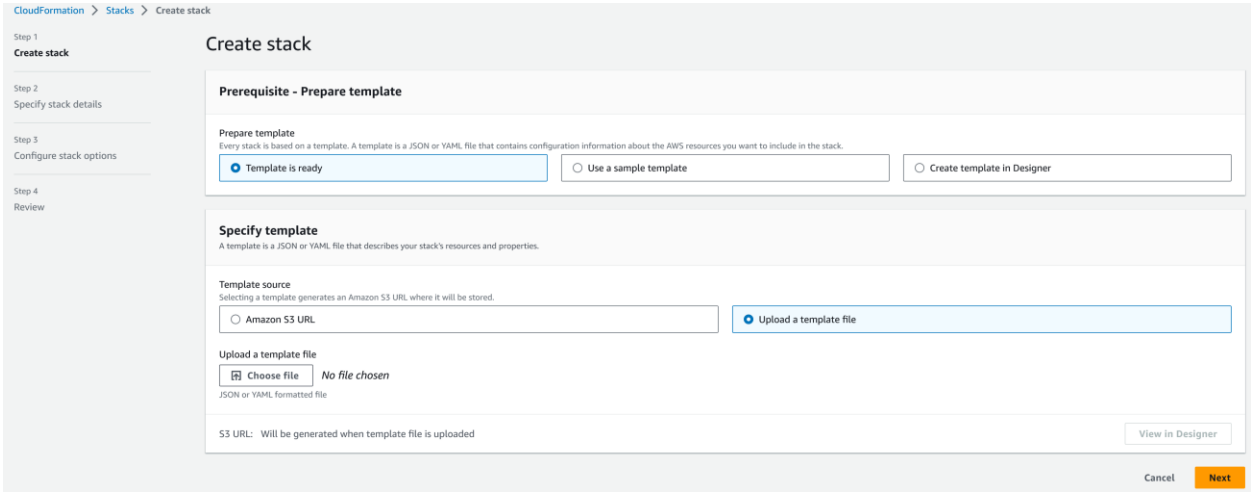
```yaml
invoice:       34843
date   :       2001-01-23
bill-to:
    given  :   Chris
    family :   Dumars
    address:
        lines: |
            458 Walkman Dr.
            Suite #292
        city    : Royal Oak
        state   : MI
        postal  : 48046
product:
    - sku         : BL394D
      quantity    : 4
      description : Basketball
      price       : 450.00
    - sku         : BL4438H
      quantity    : 1
      description : Super Hoop
      price       : 2392.00
```

- YAML and JSON are the languages you can use for CloudFormation.
- JSON
- YAML
  - Key value Pairs
  - Nested objects
  - Support Arrays
  - Multi line strings
  - Can include comments!

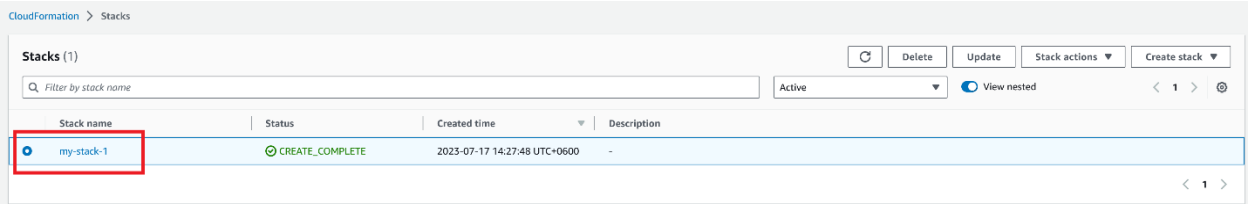
Lab: CloudFormation

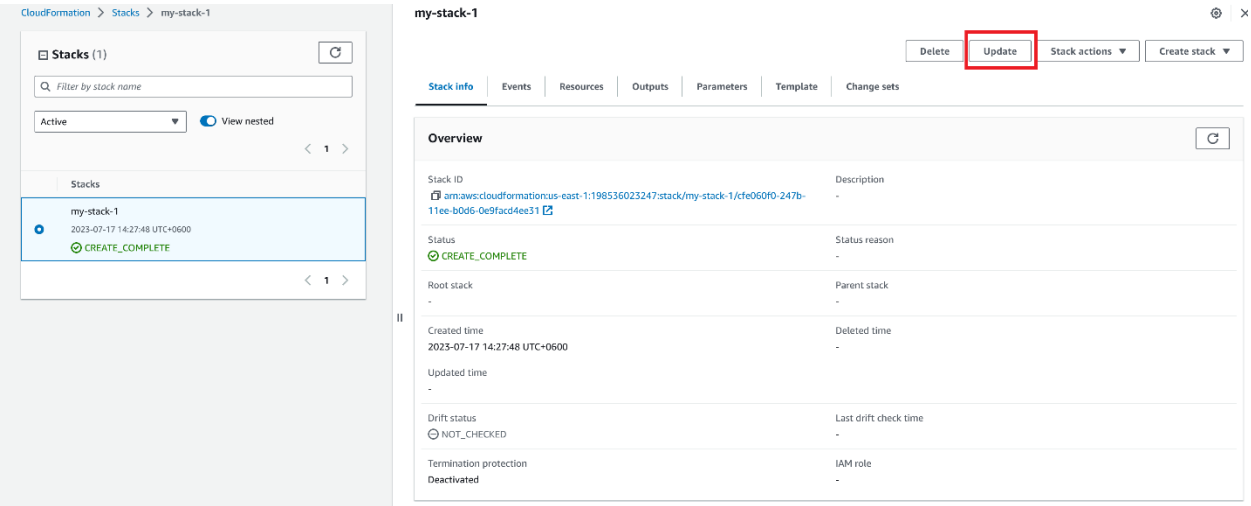
Create a ec2 by cloud formation code

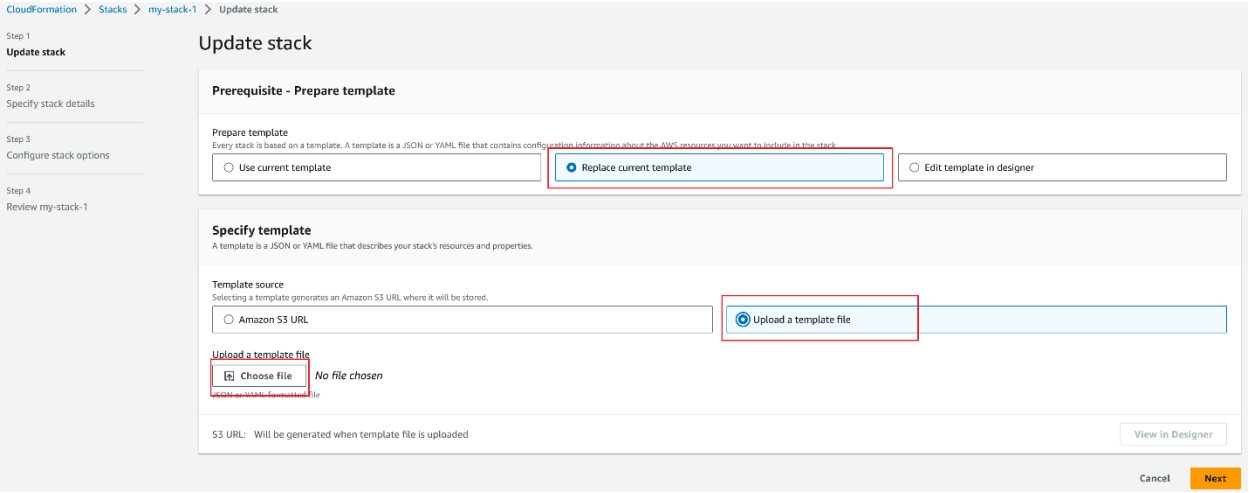


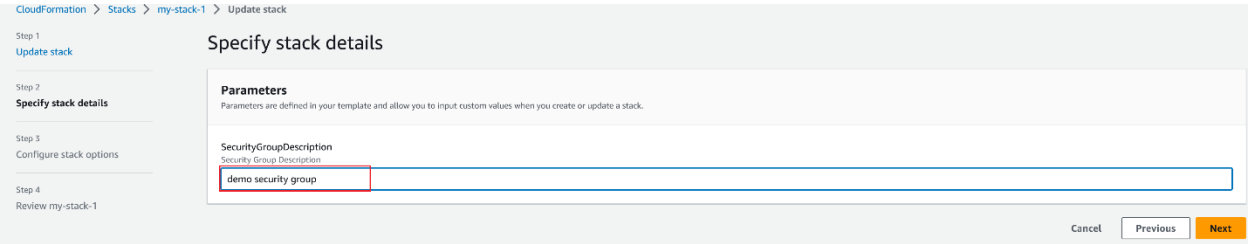


update that ec2 by cloud formation code


CloudFormation > Stacks
Stacks (1)
Delete
Update
Stack actions
Create stack
Filter by stack name
Active
View nested
Stack name
Status
Created time
Description
my-stack-1
CREATE_COMPLETE
2023-07-17 14:27:48 UTC+0600
-


CloudFormation > Stacks > my-stack-1
Stacks (1)
Filter by stack name
Active
View nested
Stacks
my-stack-1
2023-07-17 14:27:48 UTC+0600
CREATE_COMPLETE
my-stack-1
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Overview
Stack ID
arn:aws:cloudformation:us-east-1:198536023247:stack/my-stack-1/cfe060f0-247b-11ee-b0d6-0e9facd4ee31
Description
Status
CREATE_COMPLETE
Status reason
Root stack
Parent stack
Created time
2023-07-17 14:27:48 UTC+0600
Deleted time
Updated time
Drift status
NOT_CHECKED
Last drift check time
Termination protection
Deactivated
IAM role


CloudFormation > Stacks > my-stack-1 > Update stack
Step 1
Update stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review my-stack-1
Update stack
Prerequisite - Prepare template
Prepare template
Use current template
Replace current template
Edit template in designer
Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.
Template source
Selecting a template generates an Amazon S3 URL where it will be stored.
Amazon S3 URL
Upload a template file
Upload a template file
Choose file
No file chosen
S3 URL: Will be generated when template file is uploaded
View in Designer
Cancel
Next


CloudFormation > Stacks > my-stack-1 > Update stack
Step 1
Update stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review my-stack-1
Specify stack details
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
SecurityGroupDescription
Security Group Description
demo security group
Cancel
Previous
Next

CloudFormation > Stacks > my-stack-1 > Update stack

Step 1
Update stack

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review my-stack-1

Configure stack options

Tags

You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack.

Key	Value - *optional*	
name	cloudFormation-1	Remove

Add new tag

You can add 49 more tag(s)

Permissions

IAM role - optional

Choose the IAM role for CloudFormation to use for all operations performed on the stack.

IAM role name | *Sample-role-name* | Remove

Stack failure options

Behavior on provisioning failure

Specify the roll back behavior for a stack failure. Learn more

- (●) **Roll back all stack resources**
  Roll back the stack to the last known stable state.
- ( ) **Preserve successfully provisioned resources**
  Preserves the state of successfully provisioned resources, while rolling back failed resources to the last known stable state. Resources without a last known stable state will be deleted upon the next stack operation.

Advanced options

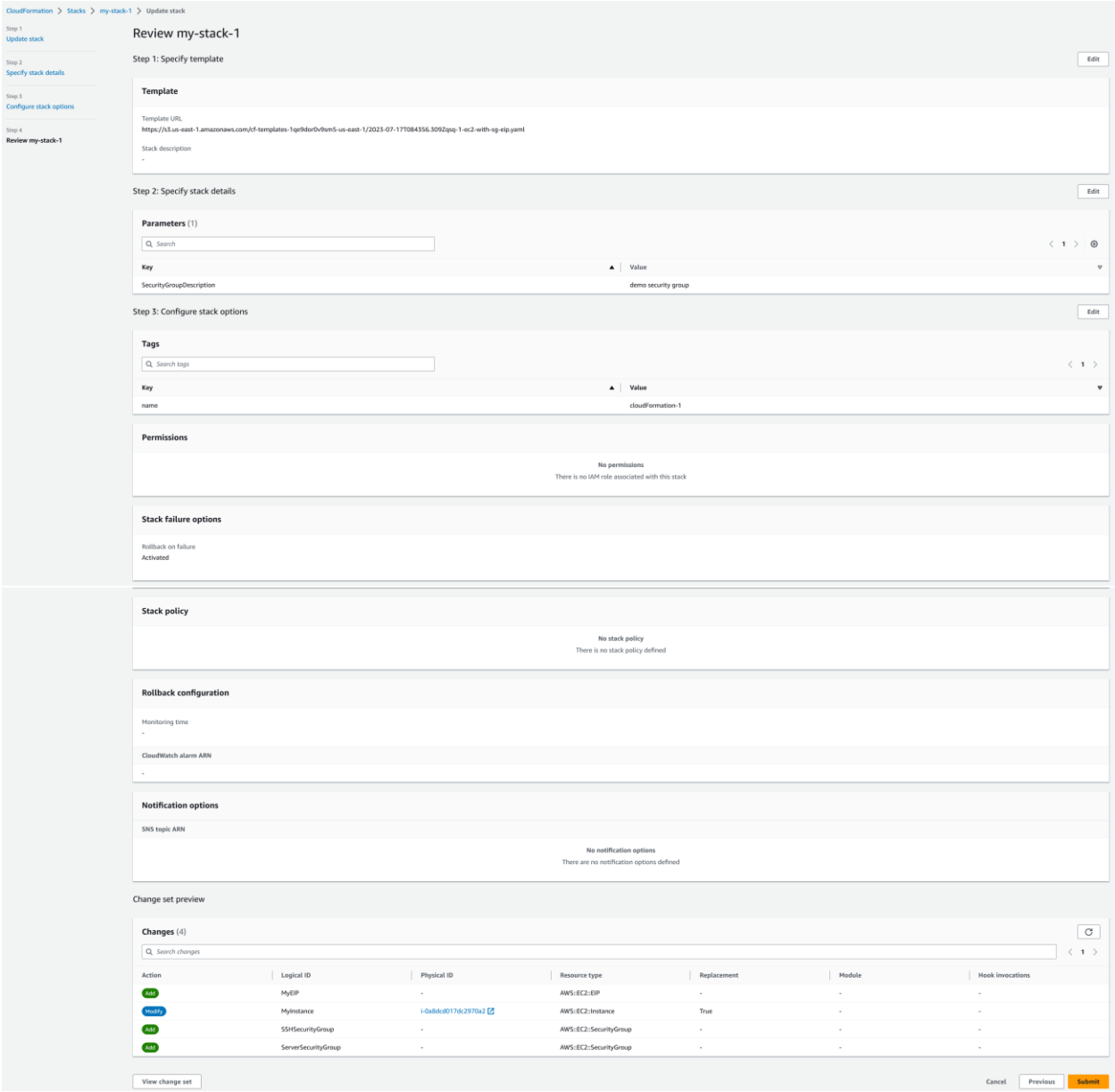
You can set additional options for your stack, like notification options and a stack policy. Learn more

▸ **Stack policy during update**
Defines the resources that you want to protect from unintentional updates during a stack update.

▸ **Rollback configuration**
Specify alarms for CloudFormation to monitor when creating and updating the stack. If the operation breaches an alarm threshold, CloudFormation rolls it back.

▸ **Notification options**

Cancel | Previous | Next



What are resources?

- Resources are the core of your CloudFormation template (MANDATORY)
- They represent the different AWS Components that will be created and configured
- Resources are declared and can reference each other

- AWS figures out creation, updates and deletes of resources for us
- There are over 224 types of resources (!)
- Resource types identifiers are of the form:

  `AWS::aws-product-name::data-type-name`

  AWS::EC2::SecurityGroup

All resources documentation?

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html

Example here for Ec2 instance
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/AWS_EC2.html

Analysis of CloudFormation Template

Going back to the example of the introductory section, let's learn why it was written this way.

Relevant documentation can be found here:

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/awsproperties-ec2-instance.html
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/awsproperties-ec2-security-group.html
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/awsproperties-ec2-eip.html

FAQ for resources

- Can I create a dynamic amount of resources?
  - ➢No, you can't. Everything in the CloudFormation template has to be declared. You can't perform code generation there

- Is every AWS Service supported?
  - ➢Almost. Only a select few niches are not there yet
  - ➢You can work around that using AWS Lambda Custom Resources

Get Ec2 from Resource URL documentaion

https://docs.aws.amazon.com/cloudformation/
>>User Guide >> Template reference >>Resource and property reference>>Amazon Ec2

Parameters - What are Parameters?

- Parameters are a way to provide inputs to your AWS CloudFormation template
- They're important to know about if:
  - You want to reuse your templates across the company
  - Some inputs can not be determined ahead of time
- Parameters are extremely powerful, controlled, and can prevent errors from happening in your templates thanks to types.

When should you use a parameter?

- Ask yourself this:
  - Is this CloudFormation resource configuration likely to change in the future?
  - If so, make it a parameter.

- You won't have to re-upload a template to change its content ☺

```
Parameters:
  SecurityGroupDescription:
    Description: Security Group Description
    (Simple parameter)
    Type: String
```

Parameters Settings

Parameters can be controlled by all these settings:

- Type:
  - String
  - Number
  - CommaDelimitedList
  - List<Type>
  - AWS Parameter (to help catch invalid values – match against existing values in the AWS Account)
- Description
- Constraints
- ConstraintDescription (String)
- Min/MaxLength
- Min/MaxValue
- Defaults
- AllowedValues (array)
- AllowedPattern (regexp)
- NoEcho (Boolean)

How to reference a parameter

- The Fn::Ref function can be leveraged to reference parameters
- Parameters can be used anywhere in a template.
- The shorthand for this in YAML is !Ref
- The function can also reference other elements within the template

```yaml
DbSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref MyVPC
```

Concept: Pseudo Parameters

- AWS offers us pseudo parameters in any CloudFormation template.
- These can be used at any time and are enabled by default

Reference Value	Example Return Value
AWS::AccountId	1234567890
AWS::NotificationARNs	[arn:aws:sns:us-east-1:123456789012:MyTopic]
AWS::NoValue	Does not return a value.
AWS::Region	us-east-2
AWS::StackId	arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1c2fa620-982a-11e3-aff7-50e2416294e0
AWS::StackName	MyStack

Use of Pseudo Parameters

In AWS CloudFormation templates, **AWS::AccountId** is a pseudo parameter that automatically resolves to the AWS account ID of the account in which the stack is being created.

Here's an example of how you might use **AWS::AccountId** within a CloudFormation template:

```
Resources:
  MyBucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'my-bucket-${AWS::AccountId}'
```

In this example, the **!Sub** function performs string substitution, and **${AWS::AccountId}** is replaced with the actual AWS account ID during stack creation. This enables you to create a unique S3 bucket name that includes your account ID to ensure uniqueness across different AWS accounts.

Pseudo parameters like **AWS::AccountId** are useful for referencing metadata about the environment or AWS account within CloudFormation templates. They provide dynamic values based on the context in which the stack is created.