AWS CLOUDFORMATION

simplilearn

What's in it for you?

- Why use AWS CloudFormation?
- What is CloudFormation?
- How does AWS CloudFormation work?
- AWS CloudFormation concepts
  - Template in AWS CloudFormation
  - Stack in AWS CloudFormation
- CloudFormation Access Control
- Demo – LAMP stack on EC2 instance
- Use case - Create a redeployable template

simplilearn

Why use AWS CloudFormation?

On AWS platform, Managing your infrastructure with many services can be hard

Why use AWS CloudFormation?

Creating and managing multiple AWS resources when replaced can be challenging and time consuming

Why use AWS CloudFormation?

Such problems could result in a whole lot of time being spent in managing your aws resources instead of developing your applications

How can we solve this problem?

Why use AWS CloudFormation?

Such problems could result in a whole lot of time being spent in managing your aws resources instead of developing your applications

How can we solve this problem?

AWS Cloud Formation can help here

What is AWS CloudFormation?

simplilearn

What is AWS CloudFormation?

AWS CloudFormation provides users a simple way to create and manage a collection of AWS resources by provisioning and updating them in an orderly and predictable way

Create and mange AWS resources

In simple terms, it allows you to create and model your infrastructure and applications without having to perform manual actions

What is AWS CloudFormation?

AWS CloudFormation provides use a simple way to create and manage a collection of AWS resources by nd predictable way

With AWS CloudFormation, Expedia is able to deploy and easily manage its entire front and backend AWS resources into its cloud environment

For example,

Create and manage AWS resources

What is AWS CloudFormation?

AWS CloudFormation

AWS CloudFormation enables you to manage your complete infrastructure or AWS resources in a text file

What is AWS CloudFormation?

Which is knows as template

AWS CloudFormation

AWS CloudFormation enables you to manage your complete infrastructure or AWS resources in a text file

AWS CloudFormation

And a collection of AWS resources is called a stack

Note: Using stack, AWS resources can be created or updated

What is AWS CloudFormation?

DID YOU KNOW?

- All the resource required by a user in an application can be deployed easily using templates

What is AWS CloudFormation?

DID YOU KNOW?

- All the resource required by a user in an application can be deployed easily using templates
- Also, you can reuse your templates to replicate your infrastructure in multiple environments

What is AWS CloudFormation?

DID YOU KNOW?

- All the resource required by a user in an application can be deployed easily using templates
- Also, you can reuse your templates to replicate your infrastructure in multiple environments
- To make templates reusable, use the parameters, mappings and conditions sections in the template so that you can customize your stacks when you create them

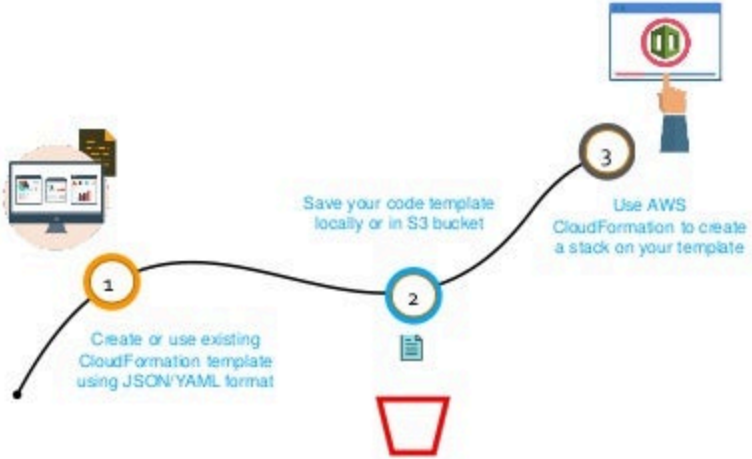
How AWS CloudFormation work?

1

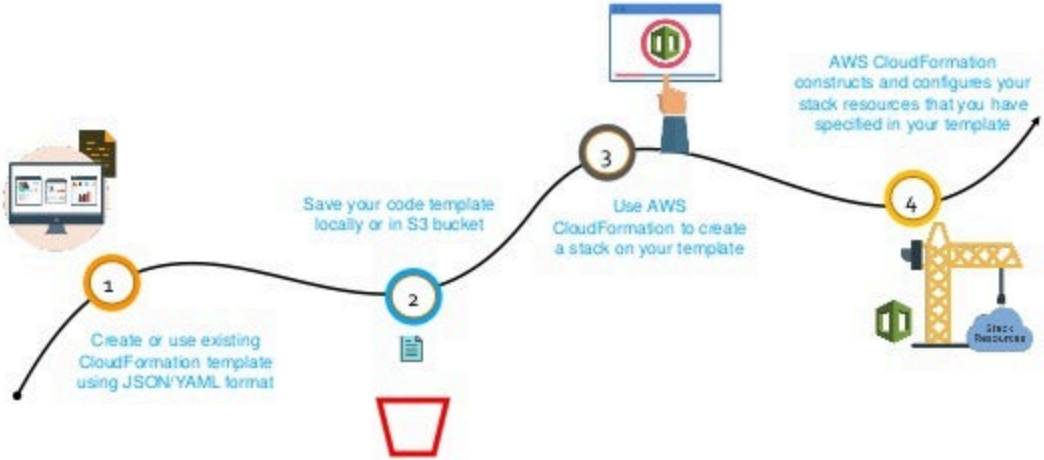
Create or use existing CloudFormation template using JSON/YAML format

How AWS CloudFormation work?

How AWS CloudFormation work?



How AWS CloudFormation work?



CloudFormation concepts

simplilearn

AWS CloudFormation concepts

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure

Create template

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

simplilearn

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Description

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Description

Metadata

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Description

Metadata

Parameters

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Description

Metadata

Parameters

Mappings

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Description

Metadata

Parameters

Mappings

Conditions

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:


Format version


Description


Metadata


Parameters


Mappings


Conditions


Transform

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Format version

Description

Metadata

Parameters

Mappings

Conditions

Transform

Resources

Templates

- A template in AWS CloudFormation is a formatted text file in JSON or YAML language that describes your AWS infrastructure
- To create, view and modify templates you can use AWS CloudFormation Designer or any text editor tool
- Template in Amazon CloudFormation consists of nine main objects:

Template structure

Sample JSON template

```
{ "AWSTemplateFormatVersion" : "version date",

 "Description" : "JSON string",

 "Metadata"    : { set of metadata },

 "Parameters"  : { set of parameters },

 "Mappings"    : { set of mappings },

 "Conditions"  : { set of conditions },

 "Transform"   : { set of transforms },

 "Resources"   : { set of resources  },

 "Outputs"     : { set of outputs  }
}
```

Template structure

Sample JSON template

Now, Let's discuss each and every object of template structure

```
{ "AWSTemplateFormatVersion" : "version date",
 "Description" : "JSON string",
 "Metadata"    :: { set of metadata },
 "Parameters"  : { set of parameters },
 "Mappings"    : { set of mappings },
 "Conditions"  : { set of conditions },
 "Transform"   : { set of transforms },
 "Resources"   :: { set of resources  },
 "Outputs"     : { set of outputs  }
}
```

1. Format version

- Format version defines the capability of a template
- The latest value of a format version is "2010-09-09"

Example in JSON

"AWSTemplateFormatVersion" : "2010-09-09"

2. Description

- Any comments about your template can be specified in description
- It describes the template in a text string

Example in JSON

"Description" : "Give some details about the template"

3. Metadata

Metadata can be used in the template to provide further information using JSON or YAML objects

Example in JSON

```
"MyInstance": {
 "Type": "AWS::S3::Bucket",
 "Metadata" : {
 "AWS::CloudFormation::Init" : {
 "config" : {}
 "packages" : { : },
 "groups" : { : },
 "users" : { : },
 "services" : { : } } }
```

4. Parameters

- Templates can be customized using parameters
- Each time you create or update your stack, parameters help you to give custom values to your template at runtime

Example in JSON

```
"Parameters" : {
"InstanceTypeParameter" : {
"Type" : "String",
"Default" : "t2.micro",
"AllowedValues" : ["t2.micro", "m1.small", "m1.large"],
"Description" : "Enter t2.micro, m1.small, or m1.large. Default is
t2.micro." }
}
```

Customize values

4. Parameters

- Templates can be customized using parameters
- Each time you create or update your stack, parameters help you to give custom values to your template at runtime

Example in JSON

```
"Parameters" : {
"InstanceTypeParameter" : {
"Type" : "String",
"Default" : "t2.micro",
"AllowedValues" : ["t2.micro", "m1.small", "m1.large"],
"Description" : "Enter t2.micro, m1.small, or m1.large. Default is
t2.micro." }
}
```

InstanceTypeParameter has a default value of t2.micro

5. Mapping

- Mapping enables you to map keys to a corresponding named value that you specify in conditional parameter
- Also, you can retrieve values in a map by using the Fn::FindInMap intrinsic function

Mapping

5. Mapping

For example: Based on a region you can set values. In a template, you can create a mapping that uses a key and holds the values that you want to specify for each specific region

Example in JSON

```
"Mappings" :
{
"Mapping01" : {
        "Key01" : {
                "Name" : "Value01"
                },
        "Key02" : {
                "Name" : "Value02"
                }
}
```

Region key value

```
"us-east-1" : { "32" : "ami-6411e20d"}
```

6. Conditions

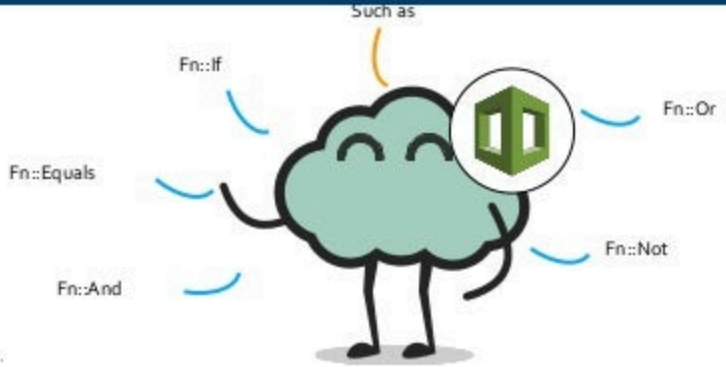
- Conditions can be used when you want to reuse the templates by creating resources in different context
- In a template, during stack creation, all the conditions in your template are evaluated
- Any resources that are associated with a true condition are created and the invalid conditions are ignored automatically

You can use intrinsic functions to define conditions

simplilearn

6. Conditions

- Conditions can be used when you want to reuse the templates by creating resources in different context
- In a template, during stack creation, all the conditions in your template are evaluated
- Any resources that are associated with a true condition are created and the invalid conditions are ignored automatically



6. Conditions

- Conditions can be used when you want to reuse the templates by creating resources in different context
- In a template, during stack creation, all the conditions in your template are evaluated
- Any resources that are associated with a true condition are created and the invalid conditions are ignored automatically

Syntax in JSON format

```
"Conditions" :
{
 "Logical ID" : {Intrinsic function}
}
```

7. Transform

- Transform builds a simple declarative language for AWS CloudFormation and enables reuse of template components

Reuse of template

7. Transform

- Transform builds a simple declarative language for AWS CloudFormation and enables reuse of template components
- Here, you can declare a single transform or multiple transforms within a template

Syntax in JSON format

```
{ "Fn::Transform" :
 {
 "Name" : "AWS::Include",
 "Parameters" :
 "Location" : "S3://MyAmazonS3BucketName/Myfile.json" }
{
"Name": "AWS::Include",
"Parameters":
{ "Location": "S3://bucket/myBucketAcl.json" }
```

Reuse of template

8. Resource

Using resource section, declare the AWS resource that you want to create and specify in the stack, such as Amazon S3 bucket or AWS Lambda

Syntax in JSON format

```
"Resources" :
{
 "MyBucket" : {
 "Type" : "AWS::S3::Bucket",
 "Properties" : { "ImageId" : "ami-212t3546"  }
  }
 }
```

Information of AWS resources

9. Output

- In a template, the output section describes the output values that you can import into other stacks or the values that are returned when a user view's his own stack properties

Stack properties

9. Output

- In a template, the output section describes the output values that you can import into other stacks or the values that are returned when a user view's his own stack properties
- For example, for a S3 bucket name, you can declare an output and use the "Description-stacks" command from the AWS CloudFormation service in order to make the bucket name easier to find

Syntax in JSON format

```
"Outputs" :
{
 "Logical ID" : {
 "Description" : "Information about the value",
 "Value" : "Value to return",
}}
```

Search for bucket name

9. Output

- In a template, the output section describes the output values that you can import into other stacks or the values that are returned when a user view's his own stack properties
- For example, for a S3 bucket name, you can declare an output and use the "Description-stacks" command from the AWS CloudFormation service in order to make the bucket name easier to find

Syntax in JSON format

```
"Outputs" :
{
 "Logical ID" : {
 "Description" : "Information about the value",
 "Value" : "Value to return",
}}
```

The value of the resource returned by the "Description-stacks" command

Template Resource Attributes

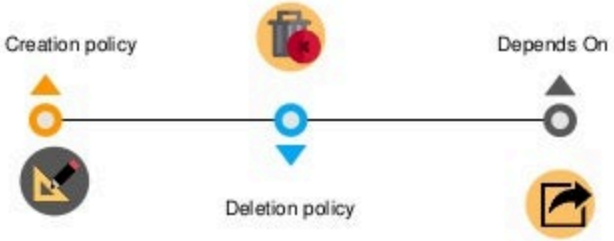
simplilearn

Template Resource Attributes

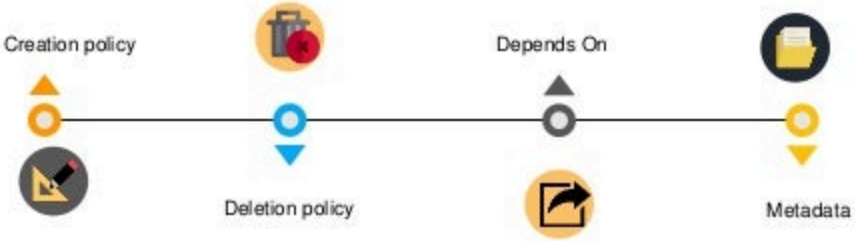
Creation policy

Template Resource Attributes

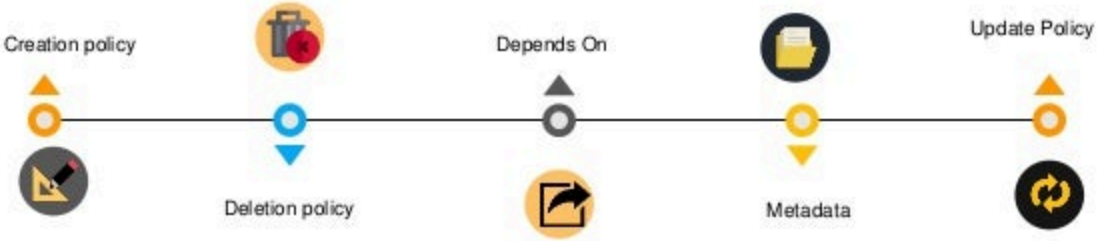
Template Resource Attributes



Template Resource Attributes



Template Resource Attributes



CreationPolicy - Template Resource Attributes

- Associate the CreationPolicy attribute with a resource when you want to delay on resource configuration actions before proceeding with stack creation
- With this attribute a stack creation is delayed, until AWS CloudFormation receives a specified number of success signals
- It can be used only for AWS AutoScaling, AWS EC2 Instance and AWS CloudFormation

Syntax in JSON format

```
"CreationPolicy":{
    "AutoScalingCreationPolicy" : {
        "MinSuccessfulInstancesPercent" : integer
    },
        "ResourceSignal": {
            "Count": "integer",
            "Timeout": "string"
  }
}
```

Specifies how many instances must signal success for the update to succeed

CreationPolicy - Template Resource Attributes

- Associate the CreationPolicy attribute with a resource when you want to delay on resource configuration actions before proceeding with stack creation
- With this attribute a stack creation is delayed, until AWS CloudFormation receives a specified number of success signals
- It can be used only for AWS AutoScaling, AWS EC2 Instance and AWS CloudFormation

Syntax in JSON format

```
"CreationPolicy":{
    "AutoScalingCreationPolicy" : {
        "MinSuccessfulInstancesPercent" : integer
    },
        "ResourceSignal": {
            "Count": "integer",
            "Timeout": "string"
  }
}
```

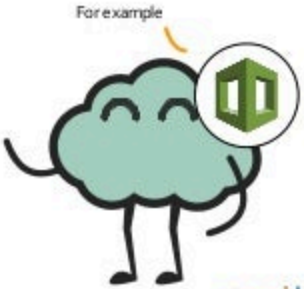
When an associated resources is created in AWS CloudFormation, it configures the number of required success signals and the length of time that AWS CloudFormation waits for those signals

CreationPolicy - Template Resource Attributes

- Associate the CreationPolicy attribute with a resource when you want to delay on resource configuration actions before proceeding with stack creation
- With this attribute a stack creation is delayed, until AWS CloudFormation receives a specified number of success signals
- It can be used only for AWS AutoScaling, AWS EC2 Instance and AWS CloudFormation

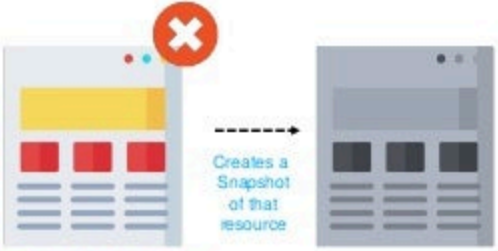
Example in JSON format

```
"CreationPolicy":{
    "AutoScalingCreationPolicy" : {
        "MinSuccessfulInstancesPercent" : 100
    },
        "ResourceSignal": {
            "Count": "3",
            "Timeout": "PT15M"
  }
}
```



DeletionPolicy - Template Resource Attributes

- Using deletion policy, preserving and backing up of resources are possible when its stack is deleted
- By default, AWS CloudFormation deletes the resource and all its content if a resource has no DeletionPolicy attribute in a template
- Before deleting a resource, AWS CloudFormation creates its snapshot



Before deleting a resource

DeletionPolicy - Template Resource Attributes

For example, the code below contains a "Retain" deletion policy for a Dynamo DB resource. When this stack is deleted, AWS CloudFormation leaves the bucket without deleting it

Sample snippet contains syntax for Amazon Dynamo DB

```
{ "AWSTemplateFormatVersion" : "Version ID",
 "Resources" : {
 "Resource name" : {
 "Type" : "AWS::resource",
 "DeletionPolicy" : "Retain"
  } } }
```

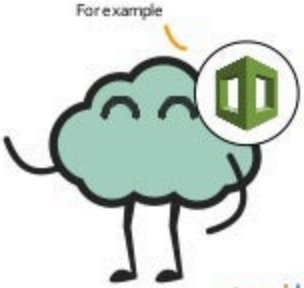
Dynamo DB

DeletionPolicy - Template Resource Attributes

For example, the code below contains a "Retain" deletion policy for a Dynamo DB resource.
When this stack is deleted, AWS CloudFormation leaves the bucket without deleting it

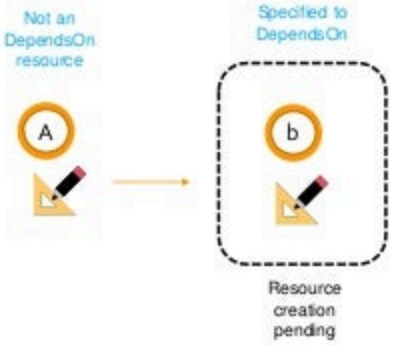
Sample snippet contains an example of Amazon Dynamo DB

```
{ "AWSTemplateFormatVersion" : "2010-09-09",
 "Resources" : {
 "myDynamoDB" : {
 "Type" : "AWS::DynamoDB::Table",
 "DeletionPolicy" : "Retain"
  } } }
```



DependsOn - Template Resource Attributes

Using the DependsOn attribute in a template, any user can define the creation of a specific resource followed by another resource



DependsOn - Template Resource Attributes

Using the DependsOn attribute in a template, any user can define the creation of a specific resource followed by another resource

A

Consider an example of resource X and resource Y (where resource X is assigned to DependsOn)

Result: Resource Y is created before resource X

DependsOn - Template Resource Attributes

Using the DependsOn attribute in a template, any user can define the creation of a specific resource followed by another resource

A

Consider an example of resource X and resource Y (where resource X is assigned to DependsOn)

Result: Resource Y is created before resource X

b

Consider another example of AWS EC2 resource with a specified AWS S3 bucket resource (where **S3** is assigned to DependsOn attribute)

When a stack is created by AWS CloudFormation, it first creates EC2 instance, then creates S3 bucket

simplilearn

DependsOn - Template Resource Attributes

Using the DependsOn attribute in a template, any user can define the creation of a specific resource followed by another resource

Sample snippet contains syntax for Amazon Dynamo DB

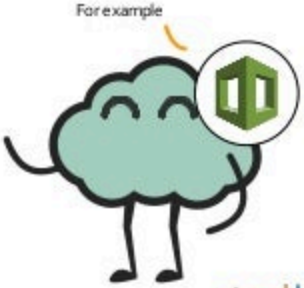
```
{
"Resources" : {
"resource Type" :
"Type" : "AWS::resource",
"DependsOn" : [ String, ... ]
"resource Type" : "AWS::resource",
}
```

DependsOn - Template Resource Attributes

Using the DependsOn attribute in a template, any user can define the creation of a specific resource followed by another resource

Sample snippet contains an example for Amazon Dynamo DB

```
{
"Resources" : {
"EC2" :
"Type" : "AWS::EC2:: Instance",
"DependsOn" : [ S3 bucket]
"resource Type" : "AWS::S3::bucket",
}
```

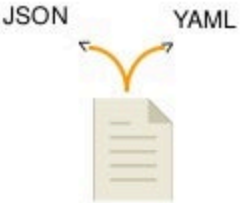


Metadata - Template Resource Attributes

- The Metadata attribute helps you to associate a resource with structured data
- By adding this attribute to a resource, you can specify the data in JSON or YAML language

Sample snippet contains syntax for Amazon Dynamo DB

```
{
"AWSTemplateFormatVersion" : "Integer",
 "Resources" : {
 "Resource name" : {
 "Type" : "Resource Name",
 "Metadata" : { "Object1" : "Location1", "Object2" : "Location2" } } } }
```



Metadata - Template Resource Attributes

- The Metadata attribute helps you to associate a resource with structured data
- By adding this attribute to a resource, you can specify the data in JSON or YAML language

Sample snippet contains an example of Amazon Dynamo DB

```
{
"AWSTemplateFormatVersion" : "2010-09-09",
 "Resources" : {
 "EC2" : {
 "Type" : "AWS::EC2::VPC",
 "Metadata" : { "VPC" : "us-east-1 ",
 "VPC" : " us-west-1 " } } } }
```

For example

UpdatePolicy - Template Resource Attributes

With UpdatePolicy attribute in AWS CloudFormation, you can manage and replace the updates of the instances in the Auto Scaling group

Sample snippet contains syntax of Amazon Dynamo DB

```
"UpdatePolicy": {
"AutoScalingReplacingUpdate" : {
 "WillReplace" : Boolean
}
    }
```

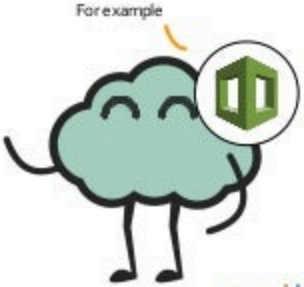
During an update, **WillReplace** specifies whether an Auto Scaling group and the instances it contains are replaced

UpdatePolicy - Template Resource Attributes

With UpdatePolicy attribute in AWS CloudFormation, you can manage and replace the updates of the instances in the Auto Scaling group

Sample snippet contains an example of Amazon Dynamo DB

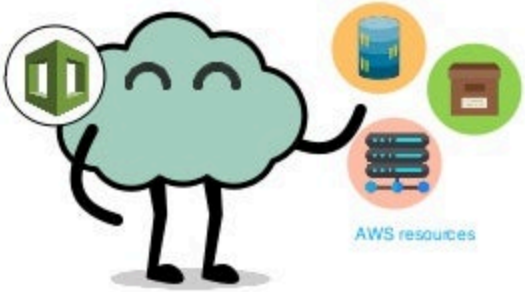
```
"UpdatePolicy": {
"AutoScalingReplacingUpdate" : {
 "WillReplace" : True/False

}
      }
```




AWS
Stacks in AWS CloudFormation
simplilearn

Stack

- A collection of AWS resources is called a stack and it can be managed in a single unit

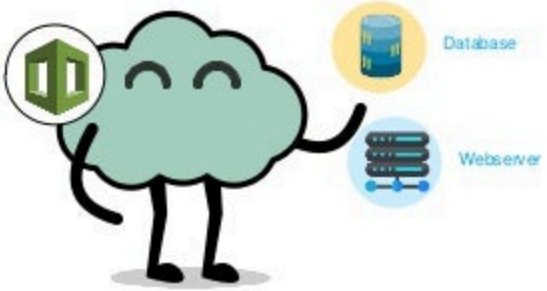


Stack

- A collection of AWS resources is called a stack and it can be managed in a single unit
- CloudFormation's template defines a stack in which the resources can be created, deleted or updated in a predictable way

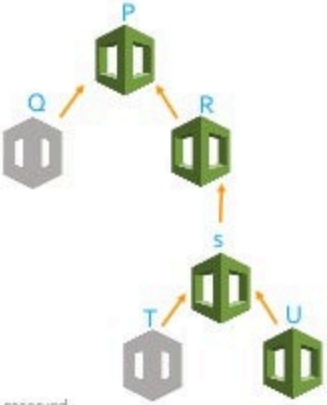
Stack

- A collection of AWS resources is called a stack and it can be managed in a single unit
- CloudFormation's template defines a stack in which the resources can be created, deleted or updated in a predictable way
- A stack can have all the resources (web server, database etc.) that can be required to run a web application



Nested Stack

- Nested stack results a hierarchy of stacks
- Using the CloudFormation stack resource, a user can create a nested stack within another stack



- ✓ Stack **P** is the root stack for its hierarchy stack
- ✓ For stack **R**, stack **P** is the root stack and parent stack as well
- ✓ For stack **U**, stack **S** is the parent class, where as for stack **S**, stack **R** is the parent class

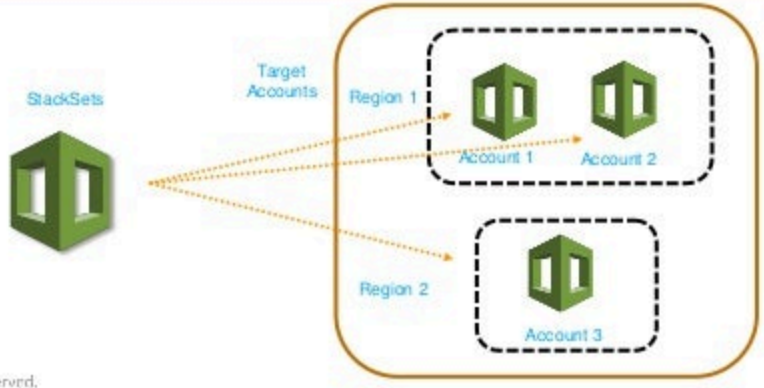
simplilearn

Windows Stack

- Windows stack gives you the ability to update and configure your own stack in windows instances
- With AWS CloudFormation, you can create Microsoft Windows stacks for Amazon EC2's Windows AMI (Amazon Machine Images)

StackSets

- Using an AWS CloudFormation template, you can define a **stackset** that lets you to create stacks in AWS accounts across the globe by using a single template
- After a stack set is defined by you, creating , updating or deleting stacks in the target accounts and regions can also be specified



CloudFormation Access Control

IAM User Access

With IAM, CloudFormation can have access control for users and ensure that only IAM users can create, update and delete stacks

CloudFormation Access Control

IAM User Access

With IAM, CloudFormation can have access control for users and ensure that only IAM users can create, update and delete stacks

Service Role

On the user's behalf, a service role allows AWS CloudFormation to make calls to resources in a stack

CloudFormation Access Control

IAM User Access

With IAM, CloudFormation can have access control for users and ensure that only IAM users can create, update and delete stacks

Stack Policy

It is applied to all AWS CloudFormation users who attempt to update the stack. Here, you cannot include different users with different stack policies

Service Role

On the user's behalf, a service role allows AWS CloudFormation to make calls to resources in a stack

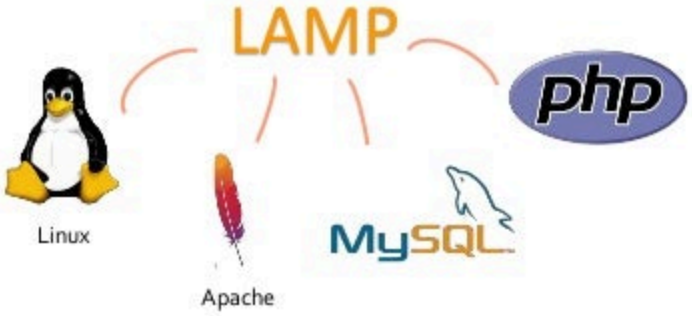
Demo - LAMP stack on EC2 instance

simplilearn

Demo – LAMP stack on EC2 instance

Problem Statement

Using AWS cloudFormation, create an EC2 instance and deploy a LAMP stack



Use case - Create a redeployable template

simplilearn

Use case – Create a redeployable template

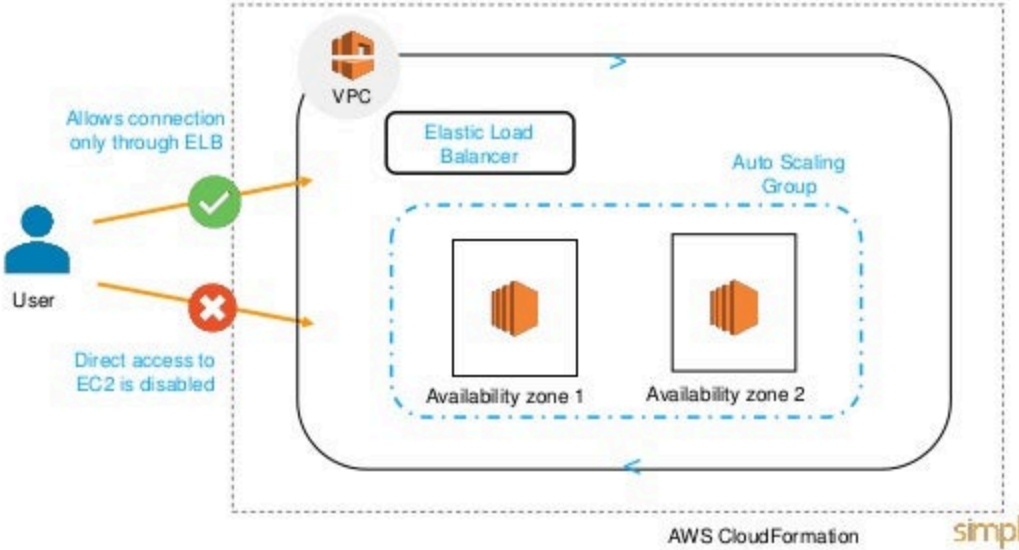
Problem Statement

To simplify a complicated environment using CloudFormation's re-deployable template

Task

To create an Elastic Load Balancer and Auto scaling group where your load balancer service acts as the single point of contact for all incoming traffic to EC2 instances

Use case – Create a redeployable template



Key Takeaways


THANK YOU
For more information, visit
www.simplilearn.com
simplilearn

Streams in C#

What Is Kali NetHunter?

simplilearn

What is Dropshipping

simplilearn

Packages in Python

simplilearn

Operator Overloading in Python

simplilearn

Generators in Python

simplilearn

Golang vs Rust

simplilearn

Golang vs Python

simplilearn