What is AWS CloudFormation?

Managing your infrastructure with many services can be hard. Creating and managing multiple AWS resources can be challenging and time-consuming. In fact, doing those things could result in spending a whole lot of time managing your AWS resources instead of developing your applications. How can we solve this problem?

AWS CloudFormation can help. As mentioned, it provides you with a simple way to create and manage a collection of AWS resources by provisioning and updating them in an orderly and predictable way. In simple terms, it allows you to create and model your infrastructure and applications without having to perform actions manually.

AWS CloudFormation enables you to manage your complete infrastructure or AWS resources in a text file, or template. A collection of AWS resources is called a stack. AWS resources can be created or updated by using a stack.

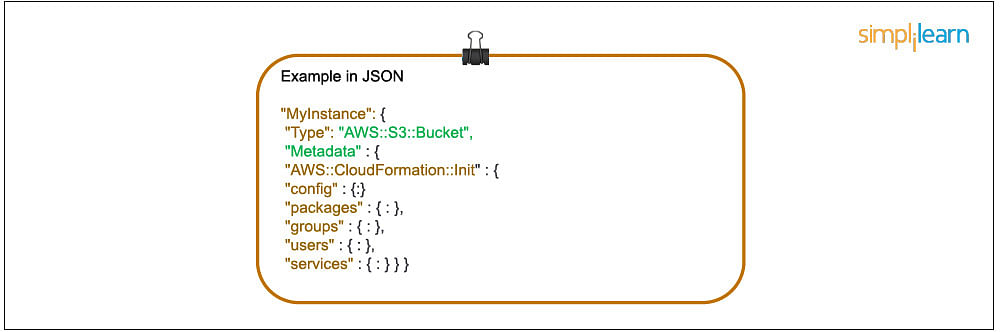
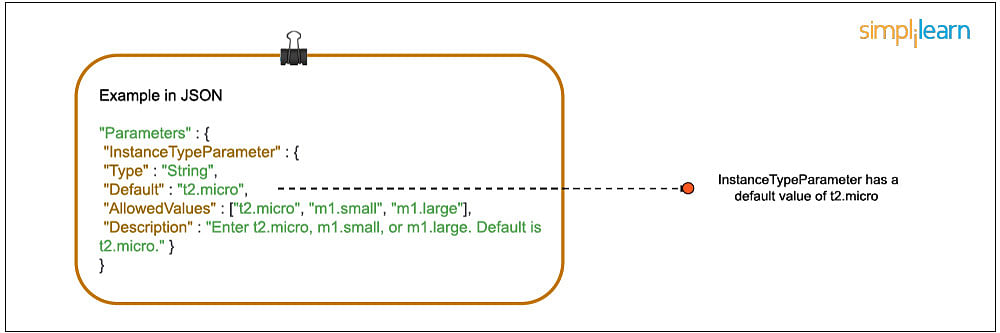
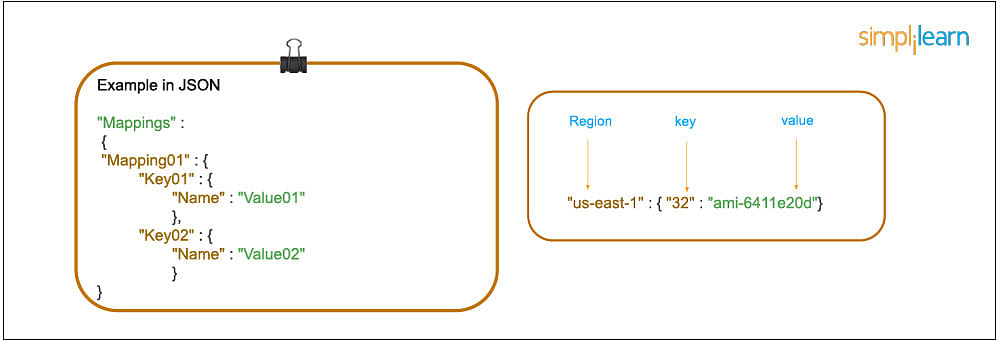
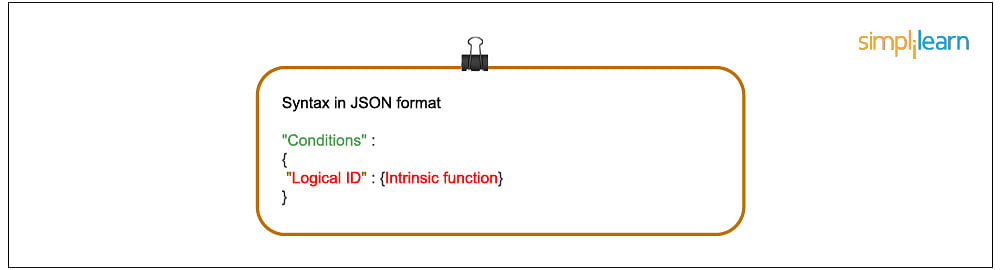
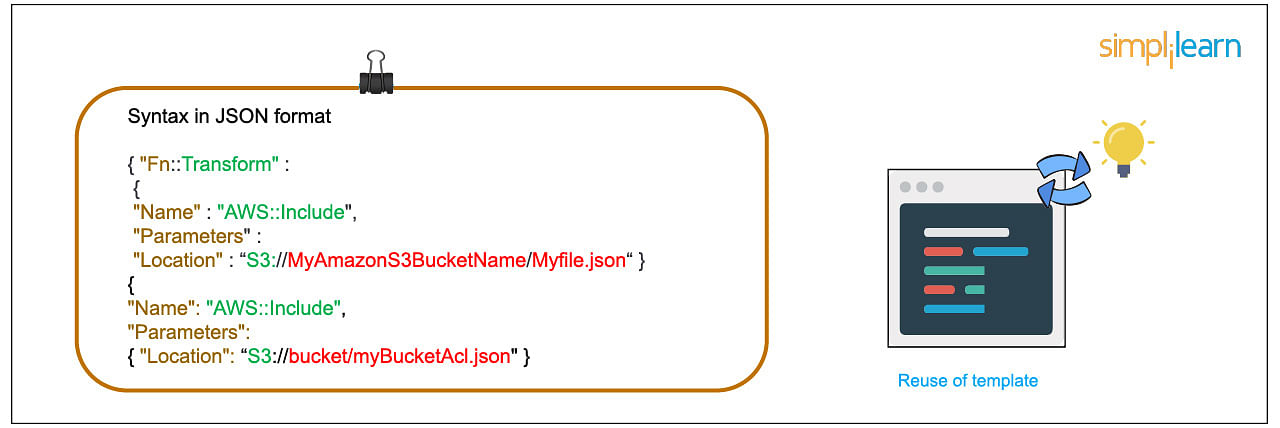
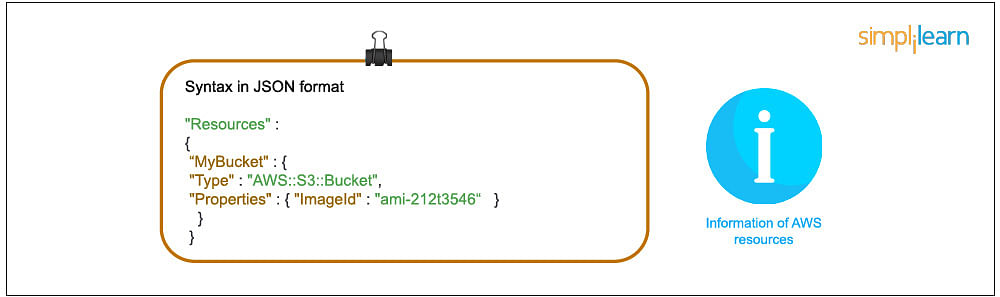
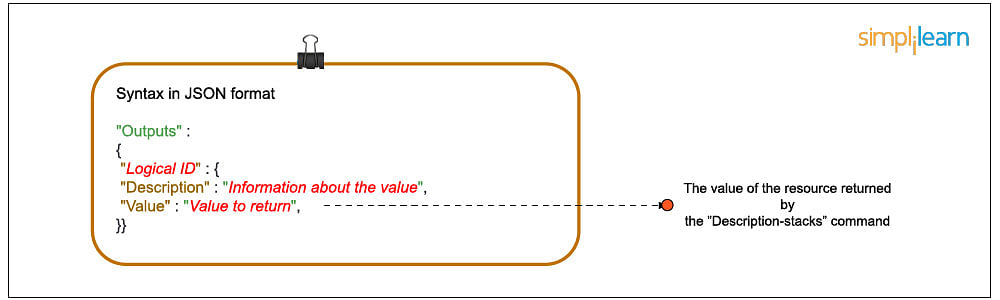
All the resources you require in an application can be deployed easily using templates. Also, you can reuse your templates to replicate your infrastructure in multiple environments. To make templates reusable, use the parameters, mappings and conditions sections in the template so that you can customize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

* Create a new template or use an existing CloudFormation template using the JSON or YAML format.
* Save your code template locally or in an S3 bucket.
* Use AWS CloudFormation to build a stack on your template.
* AWS CloudFormation constructs and configures the stack resources that you have specified in your template.

|  |
| --- |
|  |

AWS CloudFormation Concepts

An AWS CloudFormation template 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. An AWS CloudFormation template consists of nine main objects:

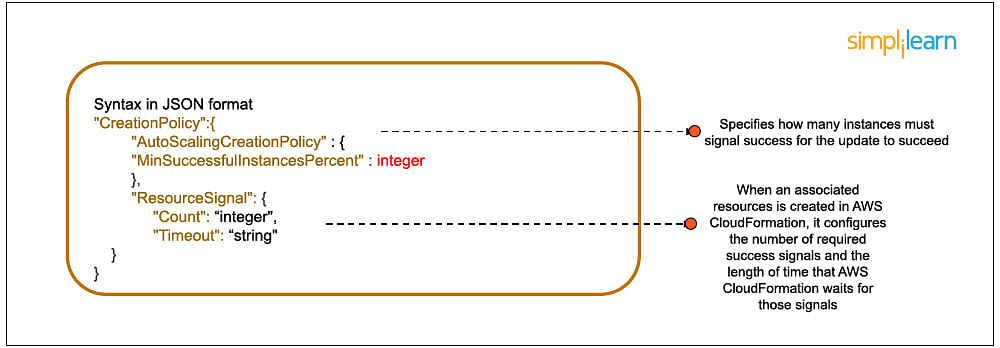
* 1. Format version: Format version defines the capability of a template.
  2. Description: Any comments about your template can be specified in the description.
  3. Metadata: Metadata can be used in the template to provide further information using JSON or YAML objects.   
       
     
  4. Parameters: Templates can be customized using parameters. Each time you create or update your stack, parameters help you give your template custom values at runtime.   
       
     
  5. Mappings: Mapping enables you to map keys to a corresponding named value that you specify in a conditional parameter. Also, you can retrieve values in a map by using the “Fn:: FindInMap” intrinsic function.  
       
     
  6. Conditions: In a template, conditions define whether certain resources are created or when resource properties are assigned to a value during stack creation or updating. Conditions can be used when you want to reuse the templates by creating resources in different contexts. You can use intrinsic functions to define conditions.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.   
       
     
  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.   
       
     
  8. Resources: Using this section, you can declare the AWS resource that you want to create and specify in the stack, such as an [Amazon S3 bucket](https://www.simplilearn.com/tutorials/aws-tutorial/aws-s3) or AWS Lambda.   
       
     
  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 you view your own stack properties. For example, for an S3 bucket name, you can declare an output and use the “Description-stacks” command from the AWS CloudFormation service to make the bucket name easier to find.  
       
     

AWS CloudFormation - Template Resource Attributes

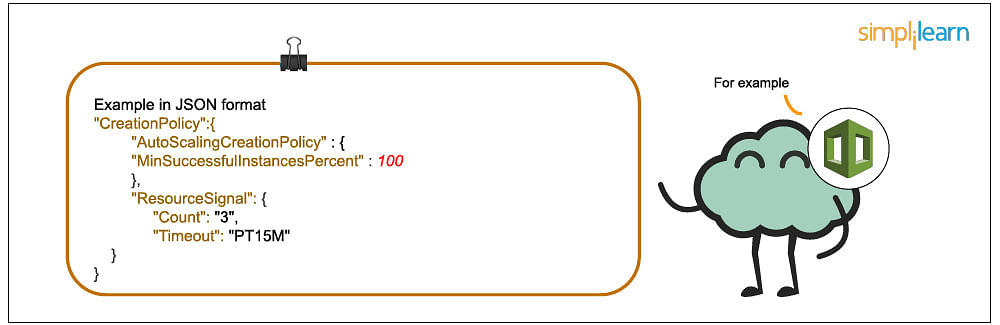
Attributes allow you add to a resource, to control additional behavior and relationships between your templates.

CreationPolicy

Associate the CreationPolicy attribute with a resource when you want to delay 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](https://www.simplilearn.com/tutorials/aws-tutorial/aws-auto-scaling), an [AWS EC2 instance](https://www.simplilearn.com/tutorials/aws-tutorial/aws-ec2), and AWS CloudFormation.

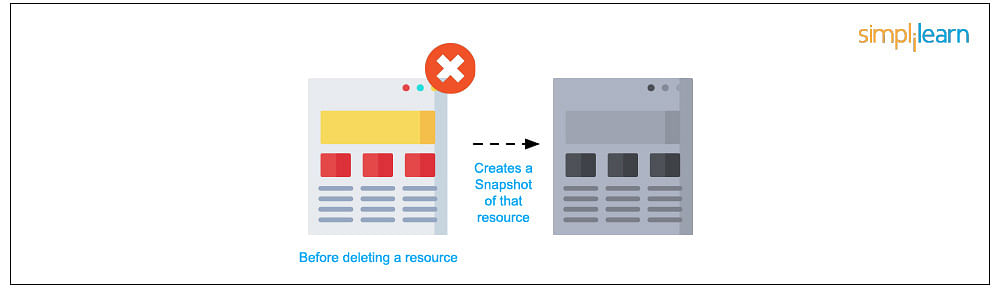


When an associated resource is created in AWS CloudFormation, it configures the number of required success signals and the length of time that AWS CloudFormation waits for those signals.

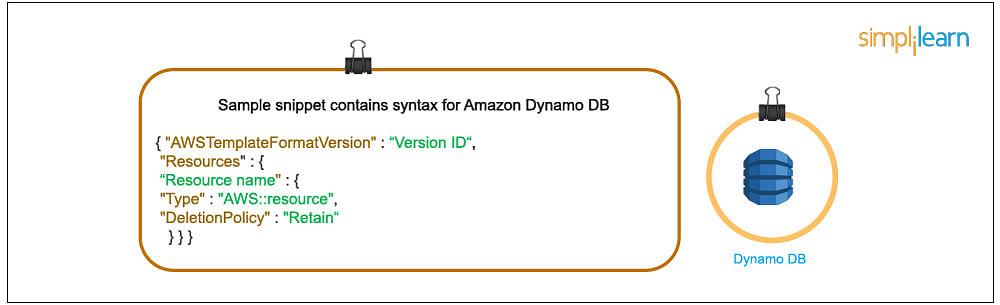
|  |
| --- |
|  |

DeletionPolicy

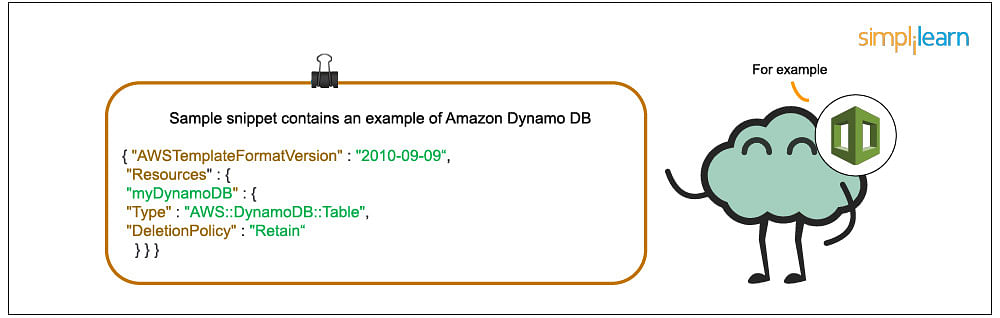
Using DeletionPolicy, preserving and backing up a resource is possible when its stack is deleted.



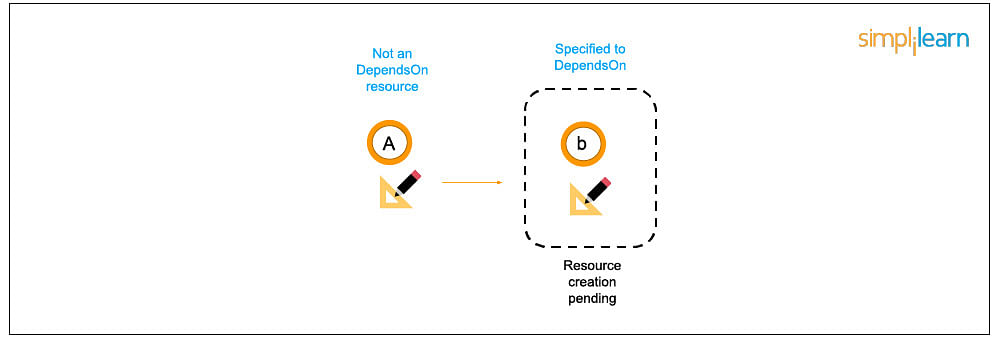
By default, AWS CloudFormation deletes the resource and all its content if a resource has no DeletionPolicy attribute in a template. Before deleting a resource, AWS CloudFormation creates a snapshot of that resource. For example, the code below contains a “Retain” deletion policy for a DynamoDB resource.



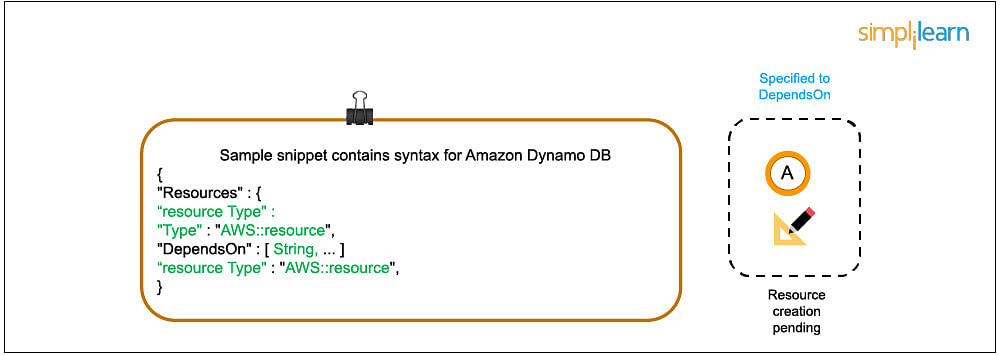
When this stack is deleted, AWS CloudFormation leaves the bucket without deleting it.



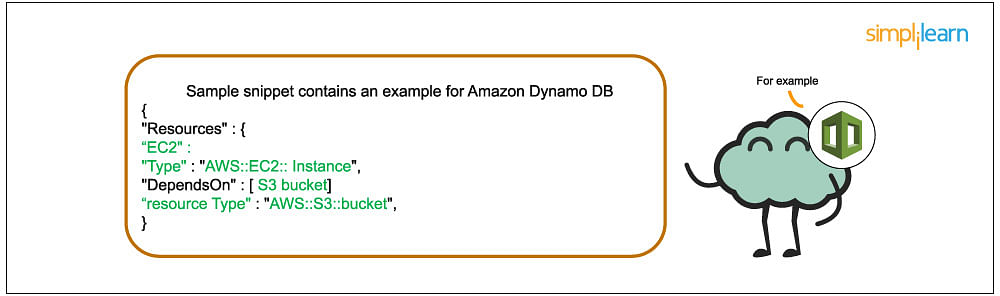
DependsOn



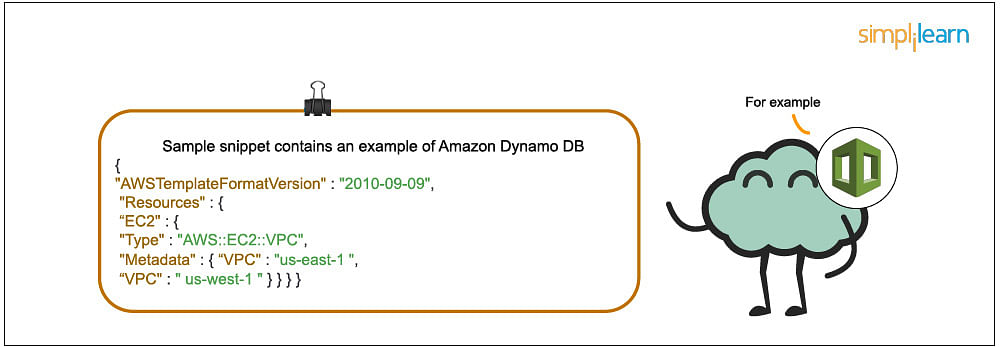
Using the DependsOn attribute in a template, you can define the creation of a specific resource followed by another resource.



* Example 1: Resource X and resource Y (where resource X is assigned to DependsOn). Result: Resource Y is created before resource X.
* Example 2: 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 an EC2 instance, then creates an S3 bucket.



Metadata



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

UpdatePolicy

With the UpdatePolicy attribute in AWS CloudFormation, you can manage and replace the updates of the instances in the Auto Scaling group. During an update, WillReplace specifies whether an Auto Scaling group and the instances it contains are replaced.

Stacks in AWS CloudFormation

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.) required to run a web application.

A [nested stack](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-nested-stacks.html) results in a hierarchy of stacks. Using the CloudFormation stack resource, you can create a nested stack within another stack.

A 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).

Using an AWS CloudFormation template, you can define a [StackSet](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html" \o "StackSet" \t "_blank) that lets you create stacks in AWS accounts around the globe using a single template. After a StackSet is defined by you, creating, updating or deleting stacks in the target accounts and regions can also be specified.

CloudFormation Access Control

With [IAM](https://www.simplilearn.com/tutorials/aws-tutorial/aws-iam), CloudFormation can give users access control and ensure that only IAM users can create, update and delete stacks. On the user’s behalf, a service role allows AWS CloudFormation to make calls to resources in a stack. It is applied to all AWS CloudFormation users who attempt to update the stack. Here, you cannot include different users with different stack policies.

Demonstration and Use Case: Lamp Stack on an Ec2 Instance

The final sections of the video demonstrate how to put AWS CloudFormation to work through two examples:

1. LAMP stack on an EC2 instance
2. Building a complicated environment that includes an elastic [load balancer,](https://www.simplilearn.com/tutorials/aws-tutorial/aws-load-balancer) auto-scaling, and an EC2 instance, which allows access to the website only through the load balancer and restricts access when accessed directly from the instance

