AWS CLOUDFORMATION BASICS

In the cloud native era, infrastructure as code (IaC) is a critical part of ensuring consistency and reusability. Most public providers have a version of IaC they offer; for AWS, it is CloudFormation.

CloudFormation helps you model your resources by describing it in a template that can be deployed as a stack on AWS. With CloudFormation, you can go from creating resources from the console to automating complex architecture on demand. Let's get started with these basics of AWS CloudFormation.

(This tutorial is part of our AWS Guide. Use the right-hand menu to navigate.)

The benefits of using CloudFormation

CloudFormation offers a variety of benefits, including:

- **Improved automation.** The simplicity of the template allows you to declare what you want your resources to look like. This eliminates the need to rely on other scripting tools to create the resources.
- **Quick infrastructure replication.** You can quickly replicate your infrastructure without affecting other resources that your template previously created. The template can be used to create as many stacks as needed.
- **Infrastructure consistency.** The declarative way of defining templates allows for consistency—you can be assured that stacks created with the template will be identical.
- **Easy-to-read template.** If you are in the web application or microservice space, you have used **yaml** or **JSON** at some point. They are both widely used, therefore making it easy to understand or find resources on it.

How does CloudFormation work?

There are three concepts you need to be aware of when using CloudFormation, and these concepts are fundamental to how it works:

- Template
- Stack
- Change Set

Let's look at each.

Template

A template is a declarative way of defining your resources as a yaml or json file. This template can then be used to deploy the resources either using the console or CLI.

This demo.yaml template shows an example of a yaml template file that creates an EC2 instance and Elastic IP and attaches the IP to the instance.

```
AWSTemplateFormatVersion: "2010-09-09"

Description: A demo template

Resources:

MyEC2Instance:

Type: "AWS::EC2::Instance"

Properties:

ImageId: "ami-0f7919c33c90f5b58"

InstanceType: t2.nano

KeyName: testkey

MyEIP:

Type: AWS::EC2::EIP

Properties:

InstanceId: !Ref MyEC2Instance
```

Stack

When you deploy a template like the example we had above, it creates both resources (EIP and EC2) as a stack. These resources are created as a unit; therefore, any update or deletion of resources

will be applied to the stack.

You can use a single template to create multiple stacks as long as there are no naming conflicts.

Change Set

When a stack needs to be updated, you can simply run an update on the stack and let AWS take care of replacing the necessary resources. Change Set takes that further and gives you the ability to see the impact of the changes you are applying before they are actually applied. In the terraform world this would be equivalent to *terraform plan*.

A basic CloudFormation example

To demonstrate how CloudFormation works, we will deploy the **demo.yaml** template. There are two ways to deploy the template, from the console or CLI. For this demo we will use AWS CLI which allows us to trigger CloudFormation API actions.

(This example assumes you have an AWS account, networking setup, access keys and AWS CLI installed.)

Create a stack

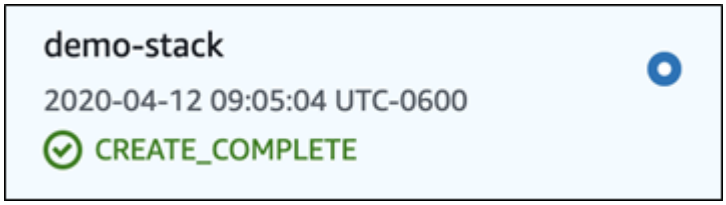
To create a new stack, run:

```
aws cloudformation create-stack --stack-name  demo-stack --template-body file://demo.yaml
```

After running this command, you should see an output that looks like this:

```
{
"StackId": "arn:aws:cloudformation:us-east-2:<ACCOUNT>:stack/demo-stack/a2ade760-7ccc-11ea-bcf5-06d398e7edd6"
}
```

If there are no errors with the deployment, you should see the stack creation as complete in the AWS console.



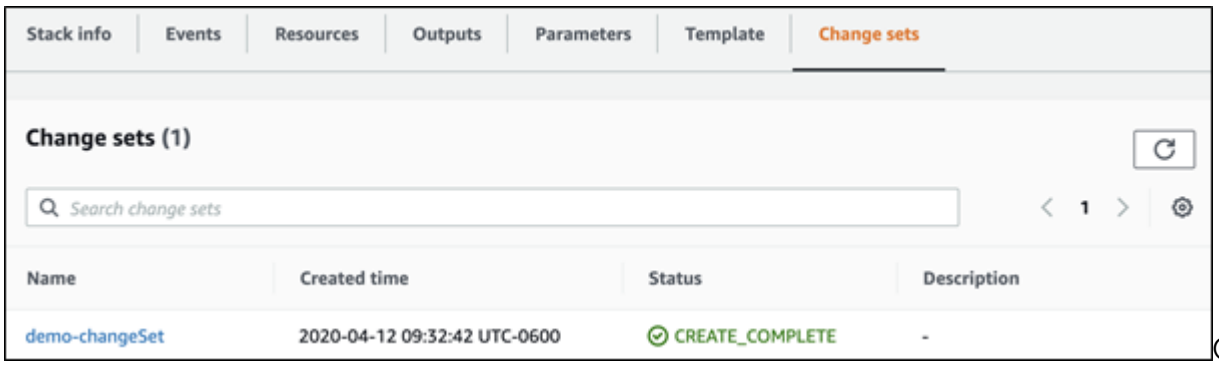
Updating a stack

Let's say you want to update the stack to use a different instance type. You can simply update the template, removing **t2.nano** and adding **t2.micro**. With that change we can then run the update-stack API action to deploy that change. This is also a good time to use a **change set** to check the impact of our changes.

Run:

```
aws cloudformation create-change-set --stack-name demo-stack --change-set-name demo-changeSet --template-body file://demo.yaml
```

In the AWS console we can see the changeset we just created:



Click on the **change set name** and you should see what will happen if you apply this change set. The image below shows the one for **demo-changeSet**.

As we can see, both the EC2 and ElasticIP will be modified. Also, it looks like the EC2 instance will be replaced. In our case, we are fine to replace the EC2 instance so we can go ahead and apply the changeset.

To apply the change set, we can run:

```
aws cloudformation execute-change-set --stack-name demo-stack --change-set-name demo-changeSet
```

In the AWS console, we can see the changes being applied.

Deleting a stack

To delete the stack, you can run the delete-stack API action. In the demo example, run:

```
aws cloudformation delete-stack --stack-name demo-stack
```

In the console, you can see the stack being deleted.