# CloudFormation Operations

Raúl Estrada

Octubre 2020

## Operations

If you have been following along, you should now have a `main.json` and a `parameters.json` in your current folder. It is time to put them to use, so here are a few operations we are going to perform:

- Validate a template
- Deploy a stack
- Update a stack
- Delete a stack

## Template validation

First things first, a basic check of our JSON template with `validate-template` :

Copy

```
$ aws cloudformation validate-template --template-body file://main.json
{
"Description": "Provisions EC2, ELB, ASG and RDS resources",
"Parameters": [
{
"NoEcho": false,
"Description": "EC2 AMI ID",
"ParameterKey": "autoscalingGroupImageId"
}
```

If there's no errors, the CLI returns the parsed template. Note that we could have just as easily pointed to a remote location using `--template-url` instead of `-template-body` .

## Deploying a Stack

To deploy our template (stack), we will use `create-stack` . It takes an arbitrary name, the location of the template, and the file containing parameter values:

Copy

```
$ aws cloudformation create-stack --stack-name cfn-test --template-body
      file://main.json --parameters file://parameters.json
{
"StackId": "arn:aws:cloudformation:us-east-1:xxxxxx:stack/cfn-test/xxxxxx"
}
```

CloudFormation starts creating the stack and no further output is returned. To get progress information on the CLI, use `describe-stacks` :

Copy

```
$ aws cloudformation describe-stacks --stack-name cfn-test
{
"Stacks": [
{
"StackId": "arn:aws:cloudformation:us-east-xxxxxx:stack/cfn-test/xxxxxx"
...
"CreationTime": "2016-05-29T20:07:17.813Z",
"StackName": "cfn-test",
"NotificationARNs": [],
"StackStatus": "CREATE_IN_PROGRESS",
"DisableRollback": false
}
]
}
```

And for even more details, use `describe-stack-events` .

After a few minutes (based on our small template) `StackStatus` changes from `CREATE_IN_PROGRESS` to `CREATE_COMPLETE` and we are provided the requested `Outputs` :

Copy


```
"Description": "VPC ID",
"OutputKey": "vpcId",
"OutputValue": "vpc-xxxxxx"
},
{
"Description": "NAT IP address",
"OutputKey": "natEip",
"OutputValue": "x.x.x.x"
},
{
"Description": "ELB DNS",
"OutputKey": "elbDns",
"OutputValue": "cloudformation-elb-xxxxxx.us-east-1.elb.amazonaws.com"
}
],
"CreationTime": "2016-05-29T20:07:17.813Z",
"StackName": "cfn-test",
"NotificationARNs": [],
"StackStatus": "CREATE_COMPLETE",
"DisableRollback": false
```

At this point, the `elbDNS` URL should return the nginx welcome page, as shown here:


If not, you might need to allow some more time for the EC2 node to fully initialize.

## Updating a stack

`CloudFormation` offers two ways of updating a deployed stack.

If you would like to quickly deploy a minor change, then all you need to do is modify the template file and deploy it directly with `update-stack` :

Copy

```
$ aws cloudformation update-stack --stack-name cfn-test
      --template-body file://main.json
      --parameters file://parameters.json
```

Otherwise, a good practice would be to use `Change Sets` to preview stack changes before deploying them. For example, let us update the rules in the ELB security group as we did before:

1. Modify the `main.json` template (add another rule to `elbSecurityGroup` ):

Copy

```
"elbSecurityGroup" : {
"Type" : "AWS::EC2::SecurityGroup",
"Properties" : {
"SecurityGroupIngress" : [ { "ToPort" : "80", "FromPort" : "80",
"IpProtocol" : "tcp", "CidrIp" : "0.0.0.0/0" },

  { "ToPort" : "443", "FromPort" : "443", "IpProtocol" :
    "tcp", "CidrIp" : "0.0.0.0/0" } ]
```

2 Create a Change Set :

Copy

```
$ aws cloudformation create-change-set
      --change-set-name updatingElbSecGroup
      --stack-name cfn-test --template-body file://main.json
      --parameters file://parameters.json
```

3 Preview the Change Set:

Copy

```
$ aws cloudformation describe-change-set
      --change-set-name updatingSecGroup
      --stack-name cfn-test
```

4 Execute the Change Set:

Copy

```
$ aws cloudformation execute-change-set --change-set-name
      updatingSecGroup --stack-name cfn-test
```

## Deleting a stack

In order to tidy up after our experiments, we will need to grant temporary Admin privileges to the CloudFormation IAM user (the same procedure as in the earlier TF section); run `delete-stack` :

Copy

```
$ aws cloudformation delete-stack --stack-name cfn-test
```

Then revoke the Admin privileges.