CloudFormation Configuration

Raúl Estrada

Octubre 2020

IaC using CloudFormation

CloudFormation is an AWS service for deploying infrastructure as code. As before, we are going to describe our infrastructure via templates containing parameters (variables), resources, and outputs.

CloudFormation calls each deployed template a **Stack**. Creating, listing, updating, and deleting stacks is possible via the AWS Console, CLI, or API. In a small setup, you would probably deploy each of your stacks individually, but as your architecture becomes more complex, you can start nesting stacks. You would have a top-level or a parent stack (template) that invokes a number of sub-stacks. Nested stacks allow you to pass variables between them and, of course, save you the time of having to deploy each one individually.

Configuration

CloudFormation provides a GUI via the AWS Console; we however, are going to focus on the AWS CLI since it is most suitable for automating tasks in the future.

Depending on the OS you run, you could download an installer from https://aws.amazon.com/cli/ or use Python PIP:

Copy

```
$ pip install awscli
$ aws --version
aws-cli/1.10.34 ...
```

We will need a set of API keys, so let's create a new IAM user called `cloudformation` with the following privileges:

Copy

```
    "cloudformation:CancelUpdateStack",
    "cloudformation:ContinueUpdateRollback",
    "cloudformation:Create*",
    "cloudformation:Describe*",
    "cloudformation:EstimateTemplateCost",
    "cloudformation:ExecuteChangeSet",
    "cloudformation:Get*",
    "cloudformation:List*",
    "cloudformation:PreviewStackUpdate",
    "cloudformation:SetStackPolicy",
    "cloudformation:SignalResource",
    "cloudformation:UpdateStack",
    "cloudformation:ValidateTemplate",
    "autoscaling:CreateAutoScalingGroup",
    "autoscaling:CreateLaunchConfiguration",
    "autoscaling:DeleteLaunchConfiguration",
    "autoscaling:Describe*",
    "autoscaling:UpdateAutoScalingGroup",
    "ec2:AllocateAddress",
    "ec2:AssociateAddress",
```

Copy

```
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:CreateInternetGateway",
    "ec2:CreateNatGateway",
    "ec2:CreateRoute",
    "ec2:CreateRouteTable",
    "ec2:CreateSecurityGroup",
    "ec2:CreateSubnet",
    "ec2:CreateTags",
    "ec2:CreateVpc",
    "ec2:Describe*",
    "ec2:Modify*",
    "ec2:RevokeSecurityGroupEgress",
    "elasticloadbalancing:CreateLoadBalancer",
    "elasticloadbalancing:CreateLoadBalancerListeners",
    "elasticloadbalancing:Describe*",
    "elasticloadbalancing:ModifyLoadBalancerAttributes",
    "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
    "rds:CreateDBInstance",
    "rds:CreateDBSubnetGroup",
    "rds:Describe*"
```

You have the choice of using `aws configure` , which will prompt you for the API credentials, or if you prefer not to store them permanently, you could use an environment variable:

Copy

```
$ export AWS_ACCESS_KEY_ID='user_access_key'
$ export AWS_SECRET_ACCESS_KEY='user_secret_access_key'
```

CloudFormation templates do not store any AWS region information, so to avoid specifying it on the command line each time. It can be exported as well:

Copy

```
$ export AWS_DEFAULT_REGION='us-east-1'
```

With those environment variables in place, `awscli` should be ready for use.