Infrastructure as code (IaC) with AWS Serverless Framework

By: Pablo Galeana Bailey

AWS Certified Solutions Architect - Associate

AWS Certified SysOps Administrator – Associate

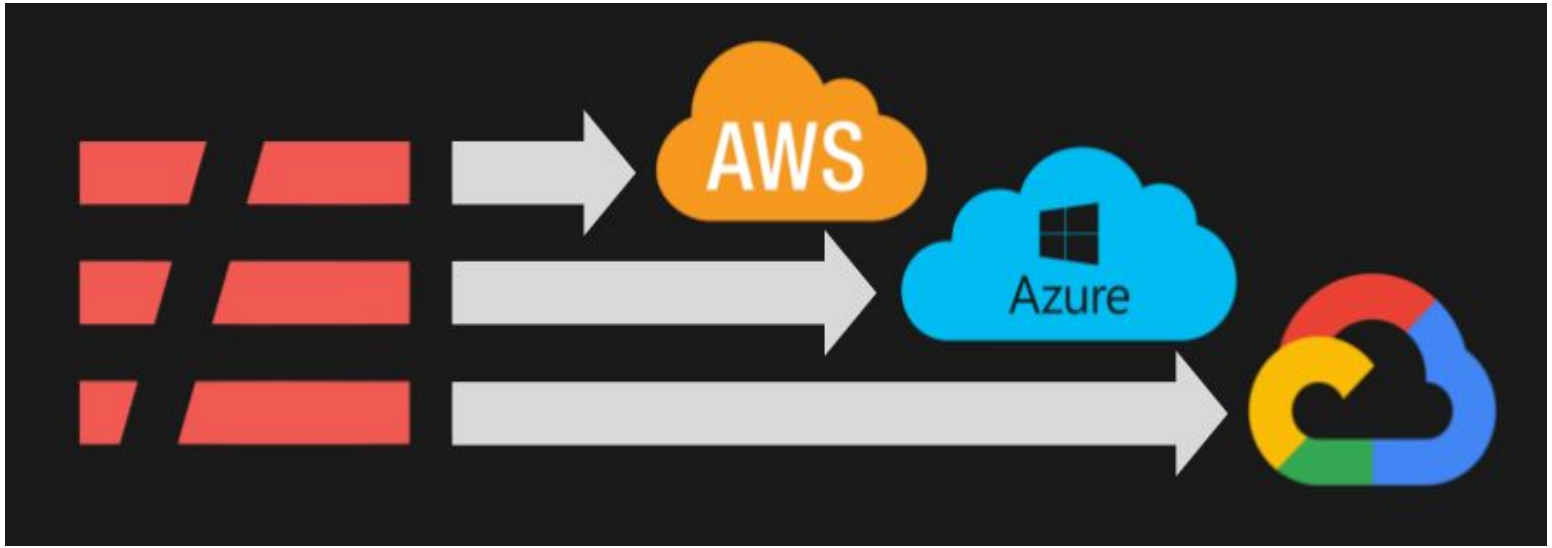
Infrastructure as Code (IaC).

Is a way of managing your devices and servers through machine-readable definition files. Basically, you write down how you want your infrastructure to look like and what code should be run on that infrastructure. Then, with the push of a button you say “Deploy my infrastructure”. Serverless Framework converts your serverless.yml into a CloudFormation template. This is a description of the infrastructure that you are trying to configure as a part of your serverless project.

Serverless Framework.

Is an open-source project introduced in 2015 that's designed to automate some serverless functions. The Serverless Framework also encourages the best practice of defining infrastructure as code. All the cloud platforms provide user interfaces for setting up services, but defining them in configuration files makes your setup far more reproducible, testable and shareable Serverless Framework converts your serverless.yml into a CloudFormation template.

Deploying to the clouds with Serverless Framework.



Deploying to the clouds with Serverless Framework.

The Serverless Framework uses Node.js, you will need to install it, then npm install -g serverless. The framework supports AWS out of the box, but if you're deploying to Azure or Google, you'll need to npm install the appropriate plugin.

The heart of the Serverless is the serverless.yml file, the serverless.yml are remarkably compact and similar across platforms.

Deploying to the clouds with Serverless Framework.

AWS:

```
service: service-name
provider:
  name: aws
  runtime: nodejs8.10
functions:
  function-name:
    handler: handler.func-name
```

Deploying to the clouds with Serverless Framework.

Azure:

```yaml
service: service-name
provider:
  name: azure
  location: West US
plugins:
  - serverless-azure-functions
functions:
  function-name:
    handler: handler.func-name
    events:
      - http: true
        x-azure-settings:
          authLevel : anonymous
      - http: true
        x-azure-settings:
          direction: out
          name: res
```

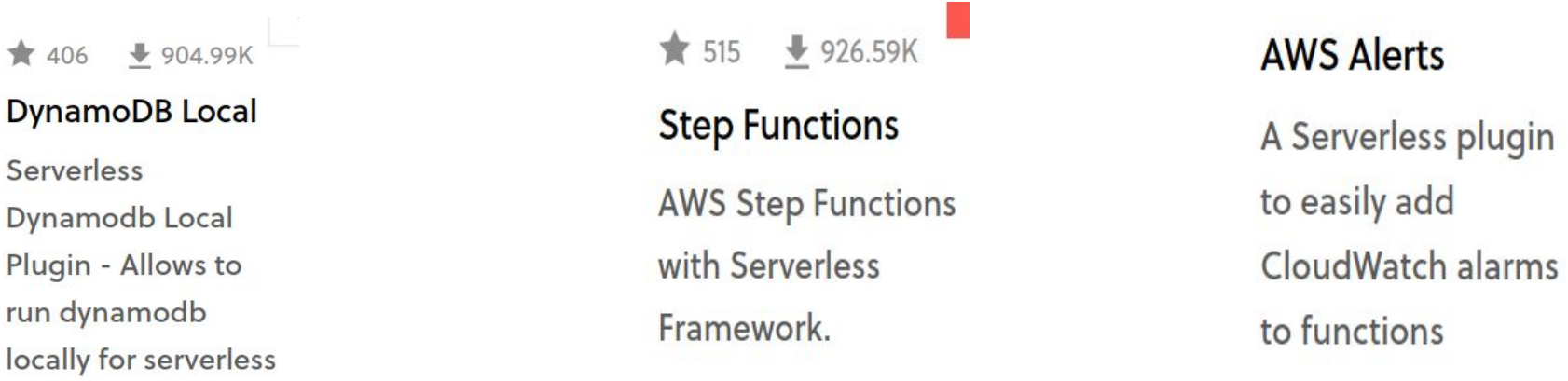
Deploying to the clouds with Serverless Framework.

Google:

```
service: service-name
provider:
  name: google
  runtime: nodejs
  project: project-id
  credentials: ~/.gcloud/keyfile.json
plugins:
  - serverless-google-cloudfunctions
functions:
  function-name:
    handler: func-name
    events:
      - http: path
```

Plugins

Plugins allow anyone to create new or extend existing commands within the Serverless Framework.



https://serverless.com/plugins/

In the next sections we are going to create our infrastructure through with AWS Serverless Framework:

1 .- Create and configure Codepipeline
2 .- Automate the creation of a vpc using serverless
3 .- Automate the creation of an ec2 using serverless
4 .- Remove infrastructure using serverless

1 .- Create and configure Codepipeline

Using the AWS CloudFormation console to create an infrastructure that includes a pipe connected to a GitHub source repository. The pipeline consists of a GitHub source code stage and a CodeDeploy implementation stage.

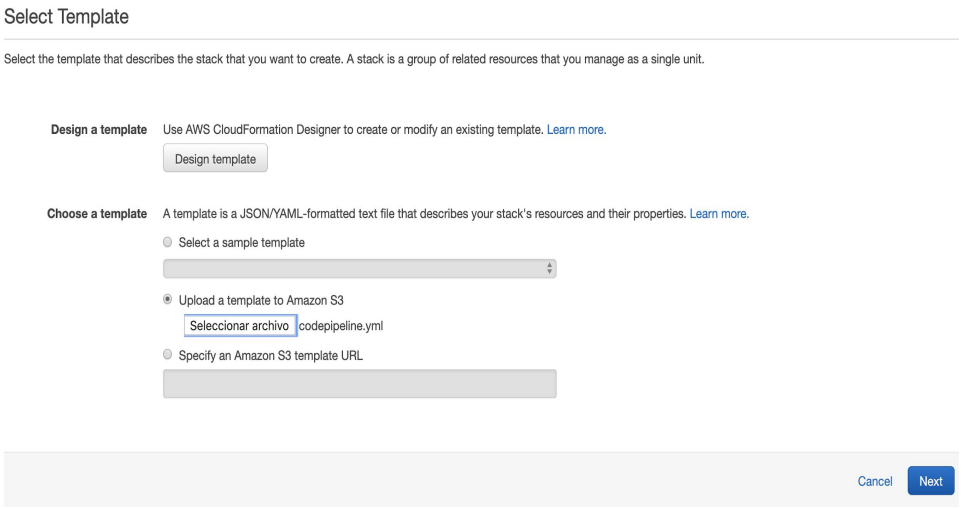
Previous requirements:

GitHub OAuth Token
CodeBuildExecutionRole
CodePipelineExecutionRole
CloudFormationExecutionRole

1 .- Create and configure Codepipeline

Open the AWS CloudFormation console and choose Create Stack, choose Upload a template to Browse and select the template file from the local computer. Select next.

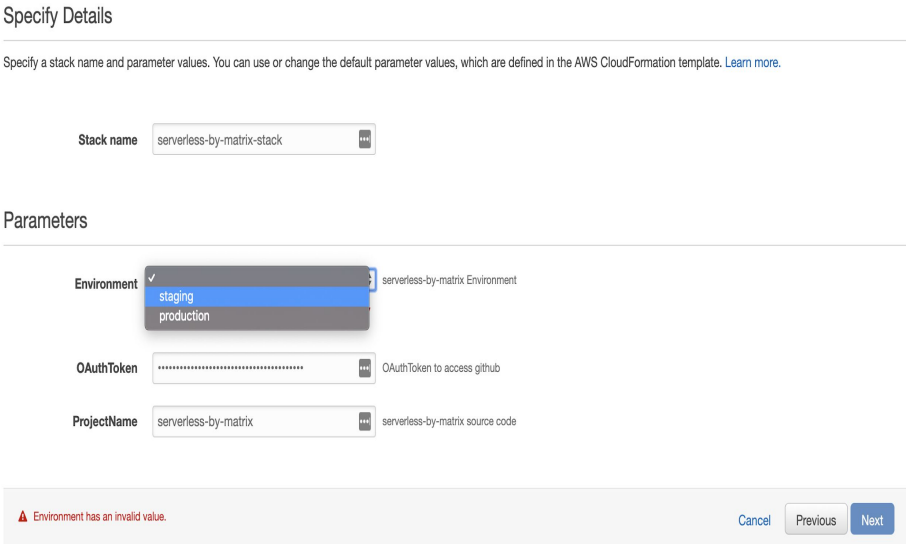
Enter the name of the pipe. The parameters specified in the codepipeline.yml template:



1 .- Create and configure Codepipeline

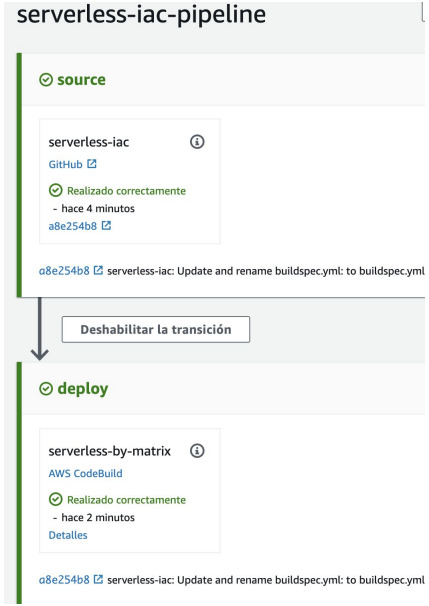
Needed parameters
GitHub OAuth Token → The Token which will be used to create the webhook in the Repo.

In Capabilities select I acknowledge that AWS CloudFormation might create IAM resources.
Select Next and Create.



1 .- Create and configure Codepipeline

In Pipelines, choose the pipe and then View. The diagram shows the stages of source code and pipeline implementation.



2 .- Automate the creation of a vpc using serverless

To deploy the vpc we must add the following line in the buildspec.yml file, which is located the root directory of the project.

- (cd aws/vpc && serverless deploy --force -v --env=${STAGE})

```yaml
version: 0.2
env:
  variables:
    STAGE: staging
phases:
  install:
    commands:
    - npm install --silent --progress=false -g serverless@1.42.3
    - npm install --silent --progress=false -g npm
    - npm install --silent serverless-deployment-bucket --save-dev
  pre_build:
    commands:
    - echo Prebuild phase...
  build:
    commands:
    - echo Build started on `date`
    - (cd aws/vpc && serverless deploy --force -v --env=${STAGE})
  post_build:
    commands:
    - echo Build completed on `date`
```

2 .- Automate the creation of a vpc using serverless

At the end of the execution of the vpc code using serverless the following resources are generated:

serverless-vpc-staging			vpc-0d998e6fb933b5747	available	198.19.0.0/16		
private-az2-staging	subnet-0f4e3909322608909	available	vpc-0d998e6fb933b5747 | serverless-vpc-staging		198.19.80.0/20	4091	-	us-east-1b
private-az1-staging	subnet-099289c9d4b637955	available	vpc-0d998e6fb933b5747 | serverless-vpc-staging		198.19.64.0/20	4091	-	us-east-1a
public-az2-staging	subnet-0d1ef593fa7610f66	available	vpc-0d998e6fb933b5747 | serverless-vpc-staging		198.19.16.0/20	4091	-	us-east-1b
public-az1-staging	subnet-0e915b679a10ab6fc	available	vpc-0d998e6fb933b5747 | serverless-vpc-staging		198.19.0.0/20	4090	-	us-east-1a

private-route-az1-staging	rtb-03d3450266ceff84a	subnet-099289c9d4b637955	No	vpc-0d998e6fb933b5747 | serverless-vpc-staging
private-route-az2-staging	rtb-07998d4a932240c...	subnet-0f4e3909322608909	No	vpc-0d998e6fb933b5747 | serverless-vpc-staging
public-route-staging	rtb-0281ce9b0b436d1c0	2 subnets	No	vpc-0d998e6fb933b5747 | serverless-vpc-staging

serverless-igw-staging	igw-0359fe537a95a8193	attached	vpc-0d998e6fb933b5747 | serverless-vpc-staging

NAT Gateway ID	Status	Stat	Elastic IP Addre	Private IP Add	Network Interface	VPC
nat-06d3ab8ee77cf3120	available	-	3.230.165.203	198.19.14.121	eni-051f41dbf431...	vpc-0d998e6fb93...

3 .- Automate the creation of an ec2 using serverless

To deploy the vpc we must add the following line in the buildspec.yml file, which is located the root directory of the project.

- (cd aws/ec2 && serverless deploy --force -v --env=${STAGE})

```yaml
version: 0.2
env:
  variables:
    STAGE: staging
phases:
  install:
    commands:
    - npm install --silent --progress=false -g serverless@1.42.3
    - npm install --silent --progress=false -g npm
    - npm install --silent serverless-deployment-bucket --save-dev
  pre_build:
    commands:
    - echo Prebuild phase...
  build:
    commands:
    - echo Build started on `date`
    - (cd aws/ec2 && serverless deploy --force -v --env=${STAGE})
  post_build:
    commands:
    - echo Build completed on `date`
```

3 .- Automate the creation of an ec2 using serverless

At the end of the execution of the ec2 code using serverless the following resources are generated:

Name	Instance ID	Instance Type	Availability Zone	Instance State
serverless-instance	i-0780003506807e572	t3a.small	us-east-1a	running

Name	aws	Group ID	Group Name	VPC ID
securitygroup-serverless	ar...	sg-07fe77ca871f386f7	serverless-iac-instance-staging-SecurityGroup-1SH8ZINO49UDB	vpc-0d998e6fb933b5747

4 .- Remove infrastructure using serverless

To remove the ec2 and vpc we must add the following line in the buildspec.yml file, which is located the root directory of the project.

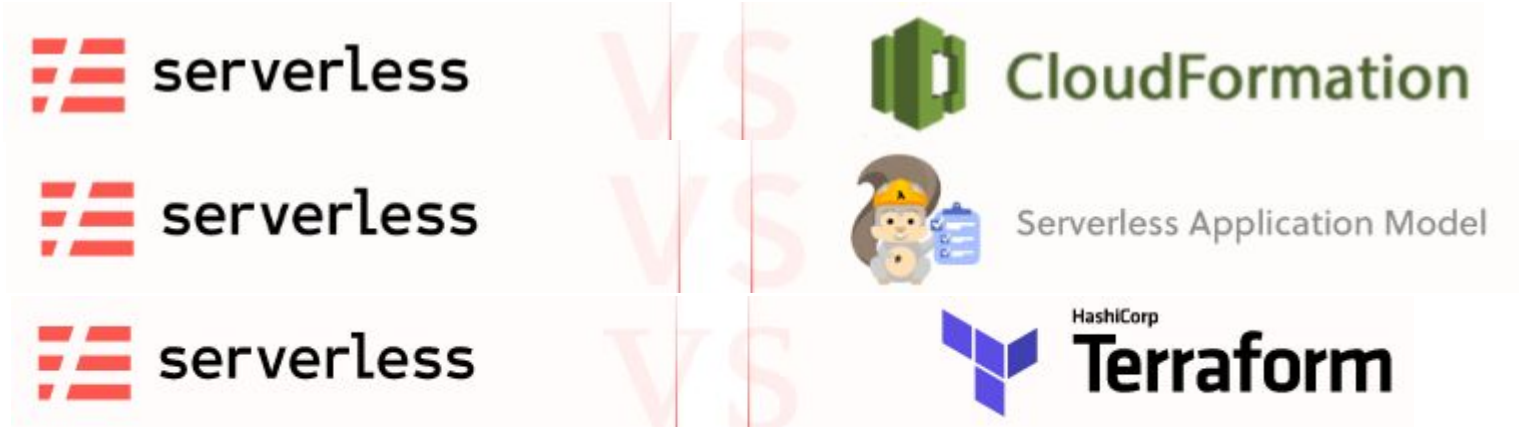
- (cd aws/ec2 && serverless remove --force -v --env=${STAGE})

- (cd aws/vpc && serverless remove --force -v --env=${STAGE})

```
version: 0.2
env:
  variables:
    STAGE: staging
phases:
  install:
    commands:
    - npm install --silent --progress=false -g serverless@1.42.3
    - npm install --silent --progress=false -g npm
    - npm install --silent serverless-deployment-bucket --save-dev
  pre_build:
    commands:
    - echo Prebuild phase...
  build:
    commands:
    - echo Build started on `date`
    - (cd aws/ec2 && serverless remove --force -v --env=${STAGE})
    - (cd aws/vpc && serverless remove --force -v --env=${STAGE})
  post_build:
    commands:
    - echo Build completed on `date`
```

Comparisons.

Serverless Architectures are enabling a wide range of use cases, but they're not right for every situation.



https://serverless.com/learn/comparisons/

Conclusion

Serverless framework to be easy to test different cloud providers. The experience also prompted me to write more platform independent code, which is always a good thing. All these platforms are evolving rapidly, so it is up to you to try them, using design patterns that can be translated across the platforms.

serverless framework

The easy and open way to build serverless applications

aws Azure Google Cloud Platform IBM Cloud kubernetes

serverless inc.