**Infrastructure to host a ‘Hello World’ application using CloudFormation on EC2**

**Step 1:** Used below Yml script to create an cloud formation stack on the name of Hello-world.yml

**yml script:**

AWSTemplateFormatVersion: '2010-09-09'

Description: 'Hello-World page Applications: Mysql,Apache and wordpress'

Parameters:

DBName:

AllowedPattern: '[a-zA-Z][a-zA-Z0-9]\*'

ConstraintDescription: must begin with a letter and contain only alphanumeric characters.

Description: The WordPress database name

MaxLength: '64'

MinLength: '1'

Type: String

DBPassword:

AllowedPattern: '[a-zA-Z0-9]\*'

ConstraintDescription: must contain only alphanumeric characters.

Description: The WordPress database admin account password

MaxLength: '41'

MinLength: '8'

NoEcho: 'true'

Type: String

DBRootPassword:

AllowedPattern: '[a-zA-Z0-9]\*'

ConstraintDescription: must contain only alphanumeric characters.

Description: MySQL root password

MaxLength: '41'

MinLength: '8'

NoEcho: 'true'

Type: String

DBUser:

AllowedPattern: '[a-zA-Z][a-zA-Z0-9]\*'

ConstraintDescription: must begin with a letter and contain only alphanumeric characters.

Description: The WordPress database admin account username

MaxLength: '16'

MinLength: '1'

NoEcho: 'true'

Type: String

InstanceType:

AllowedValues:

- t2.nano

- t2.micro

- t2.small

- t2.medium

- t2.large

- t3.nano

- t3.micro

- t3.small

- t3.medium

- t3.large

- t3a.nano

- t3a.micro

- t3a.small

- t3a.medium

- t3a.large

- t4g.nano

- t4g.micro

- t4g.small

- t4g.medium

- t4g.large

ConstraintDescription: must be a valid EC2 instance type.

Default: t3.small

Description: WebServer EC2 instance type

Type: String

KeyName:

ConstraintDescription: must be the name of an existing EC2 KeyPair.

Description: Name of an existing EC2 KeyPair to enable SSH access to the instances

Type: AWS::EC2::KeyPair::KeyName

SSHLocation:

AllowedPattern: (\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})

ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Default: 0.0.0.0/0

Description: The IP address range that can be used to SSH to the EC2 instances

MaxLength: '18'

MinLength: '9'

Type: String

Mappings:

AWSInstanceType2Arch:

t2.large:

Arch: AL2x86

t2.medium:

Arch: AL2x86

t2.micro:

Arch: AL2x86

t2.nano:

Arch: AL2x86

t2.small:

Arch: AL2x86

t3.large:

Arch: AL2x86

t3.medium:

Arch: AL2x86

t3.micro:

Arch: AL2x86

t3.nano:

Arch: AL2x86

t3.small:

Arch: AL2x86

t3a.large:

Arch: AL2x86

t3a.medium:

Arch: AL2x86

t3a.micro:

Arch: AL2x86

t3a.nano:

Arch: AL2x86

t3a.small:

Arch: AL2x86

t4g.large:

Arch: AL2arm64

t4g.medium:

Arch: AL2arm64

t4g.micro:

Arch: AL2arm64

t4g.nano:

Arch: AL2arm64

t4g.small:

Arch: AL2arm64

AWSRegionArch2AMI:

af-south-1:

AL2x86: ami-0bb140f2ff1df29fc

AL2arm64: ami-0d9c65f17370a7203

ap-east-1:

AL2x86: ami-7284c903

AL2arm64: ami-7386cb02

ap-northeast-1:

AL2x86: ami-00f045aed21a55240

AL2arm64: ami-077527e5c50f1d6d1

ap-northeast-2:

AL2x86: ami-03461b78fdba0ff9d

AL2arm64: ami-09ac600bb6e9c3f0b

ap-northeast-3:

AL2x86: ami-0777add682c903044

AL2arm64: ami-0382375f125289052

ap-south-1:

AL2x86: ami-08f63db601b82ff5f

AL2arm64: ami-0e502bbbe5de26d28

ap-southeast-1:

AL2x86: ami-0d728fd4e52be968f

AL2arm64: ami-00c509ff6d6bd1357

ap-southeast-2:

AL2x86: ami-09f765d333a8ebb4b

AL2arm64: ami-0f4e324b642c97f9f

ca-central-1:

AL2x86: ami-0fca0f98dc87d39df

AL2arm64: ami-09a8233d002600d0e

cn-north-1:

AL2x86: ami-0cf913cef98c31648

AL2arm64: ami-088cc0c104292da9c

cn-northwest-1:

AL2x86: ami-0a12cb9cd7fea53e7

AL2arm64: ami-0b5c6ceb80eb57861

eu-central-1:

AL2x86: ami-0bd39c806c2335b95

AL2arm64: ami-011c2803cfd207fca

eu-north-1:

AL2x86: ami-02511cb3673b49e04

AL2arm64: ami-0fb5db3dec6eaa5d4

eu-south-1:

AL2x86: ami-0ed21bdc9c547dd9b

AL2arm64: ami-011d4067dedd119f5

eu-west-1:

AL2x86: ami-0ce1e3f77cd41957e

AL2arm64: ami-092a4058239e7214c

eu-west-2:

AL2x86: ami-08b993f76f42c3e2f

AL2arm64: ami-0fc8a243adfe06608

eu-west-3:

AL2x86: ami-0e9c91a3fc56a0376

AL2arm64: ami-0ad9db05ff135ce32

me-south-1:

AL2x86: ami-0b38d62acce7fb76a

AL2arm64: ami-0a97e14ecf194f972

sa-east-1:

AL2x86: ami-0096398577720a4a3

AL2arm64: ami-0f0d1eee1a07925ba

us-east-1:

AL2x86: ami-04d29b6f966df1537

AL2arm64: ami-03156384f702d4eaf

us-east-2:

AL2x86: ami-09558250a3419e7d0

AL2arm64: ami-05394aef61908afaa

us-west-1:

AL2x86: ami-08d9a394ac1c2994c

AL2arm64: ami-0264f2bba9a783a26

us-west-2:

AL2x86: ami-0e472933a1395e172

AL2arm64: ami-0b0154d3d8011b0cd

Resources:

WebServer:

Type: AWS::EC2::Instance

CreationPolicy:

ResourceSignal:

Timeout: PT15M

Metadata:

AWS::CloudFormation::Init:

configSets:

wordpress\_install:

- install\_cfn

- install\_wordpress

- configure\_wordpress

configure\_wordpress:

commands:

01\_set\_mysql\_root\_password:

command: !Sub |

mysqladmin -u root password '${DBRootPassword}'

test: !Sub |

$(mysql ${DBName} -u root --password='${DBRootPassword}' >/dev/null 2>&1 </dev/null); (( $? != 0 ))

02\_create\_database:

command: !Sub |

mysql -u root --password='${DBRootPassword}' < /tmp/setup.mysql

test: !Sub |

$(mysql ${DBName} -u root --password='${DBRootPassword}' >/dev/null 2>&1 </dev/null); (( $? !=0))

03\_configure\_wordpress:

command: /tmp/create-wp-config

cwd: /var/www/html/wordpress

install\_cfn:

files:

/etc/cfn/cfn-hup.conf:

content: !Sub |

[main]

stack= ${AWS::StackId}

region=${AWS::Region}

group: root

mode: '000400'

owner: root

/etc/cfn/hooks.d/cfn-auto-reloader.conf:

content: !Sub |

[cfn-auto-reloader-hook]

triggers=post.update

path=Resources.WebServer.Metadata.AWS::CloudFormation::Init

action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServer --configsets wordpress\_install --url https://stackbuilder.amazonaws.com

group: root

mode: '000400'

owner: root

services:

sysvinit:

cfn-hup:

enabled: true

ensureRunning: true

files:

- /etc/cfn/cfn-hup.conf

- /etc/cfn/hooks.d/cfn-auto-reloader.conf

install\_wordpress:

files:

/tmp/create-wp-config:

content: !Sub |

#!/bin/bash -xe

cp /var/www/html/wordpress/wp-config-sample.php /var/www/html/wordpress/wp-config.php

sed -i "s/'database\_name\_here'/'${DBName}'/g" wp-config.php

sed -i "s/'username\_here'/'${DBUser}'/g" wp-config.php

sed -i "s/'password\_here'/'${DBPassword}'/g" wp-config.php

group: root

mode: '000500'

owner: root

/tmp/setup.mysql:

content: !Sub |

CREATE DATABASE ${DBName};

CREATE USER '${DBUser}'@'localhost' IDENTIFIED BY '${DBPassword}';

GRANT ALL ON ${DBName}.\* TO '${DBUser}'@'localhost';

FLUSH PRIVILEGES;

group: root

mode: '000400'

owner: root

packages:

yum:

httpd: []

mariadb: []

mariadb-devel: []

mariadb-libs: []

mariadb-server: []

php: []

php-mysqlnd: []

services:

sysvinit:

httpd:

enabled: true

ensureRunning: true

mariadb:

enabled: true

ensureRunning: true

sources:

/var/www/html: http://wordpress.org/latest.tar.gz

Properties:

ImageId: !FindInMap [AWSRegionArch2AMI, !Ref 'AWS::Region', !FindInMap [AWSInstanceType2Arch, !Ref InstanceType, Arch]]

InstanceType:

Ref: InstanceType

KeyName:

Ref: KeyName

SecurityGroups:

- Ref: WebServerSecurityGroup

UserData:

Fn::Base64: !Sub |

#!/bin/bash -xe

yum update -y aws-cfn-bootstrap

/opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource WebServer --configsets wordpress\_install --region ${AWS::Region}

/opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackId} --resource WebServer --region ${AWS::Region}

#Set the Headers

echo "<h1>Hello World</h1>" > /usr/share/httpd/noindex/index.html

WebServerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: "Enable HTTP access via port 80 locked down to the load balancer + SSH access"

SecurityGroupIngress:

- CidrIp: 0.0.0.0/0

FromPort: '80'

IpProtocol: tcp

ToPort: '80'

- CidrIp: !Ref SSHLocation

FromPort: '22'

IpProtocol: tcp

ToPort: '22'

Outputs:

PublicIP:

Description: EC2 public IP

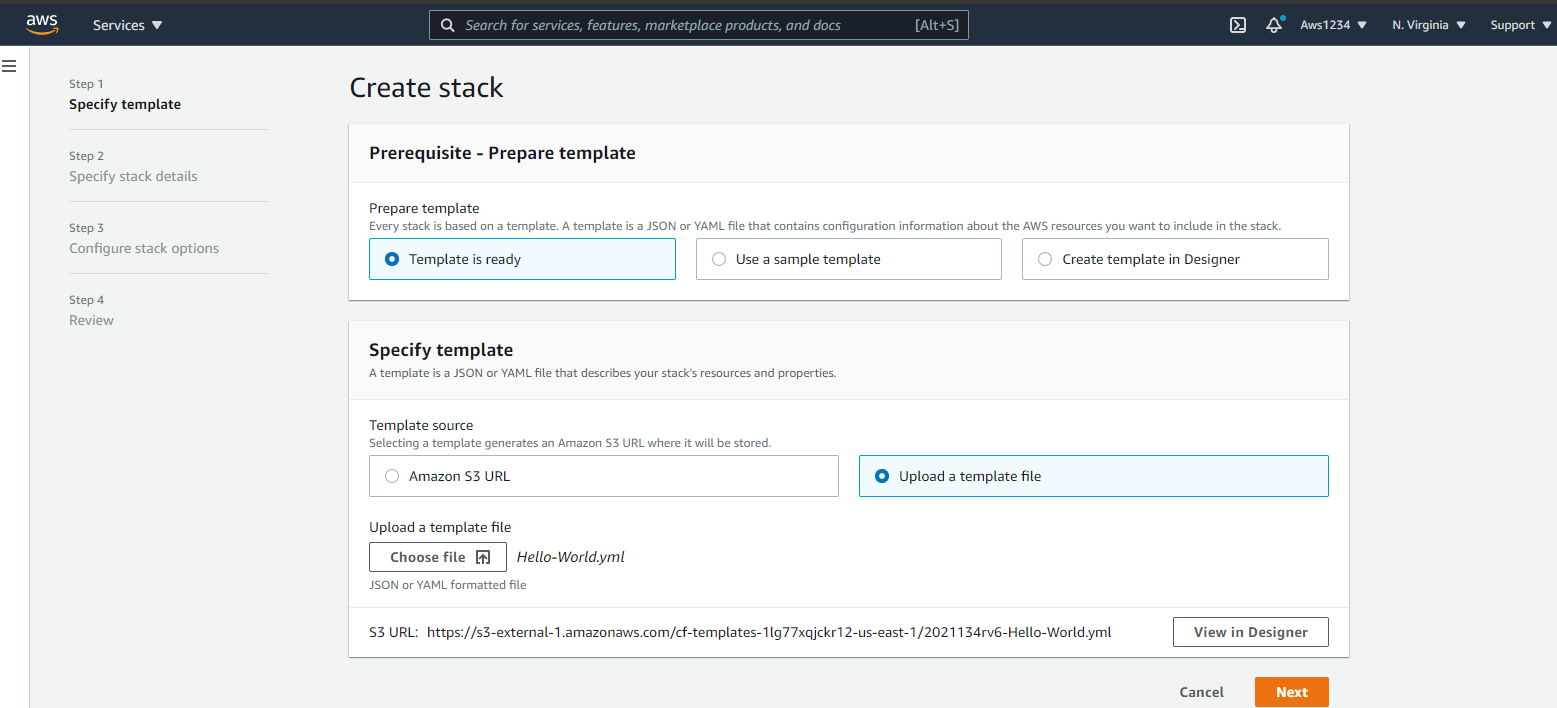
Value: !GetAtt WebServer.PublicIp

WebsiteURL:

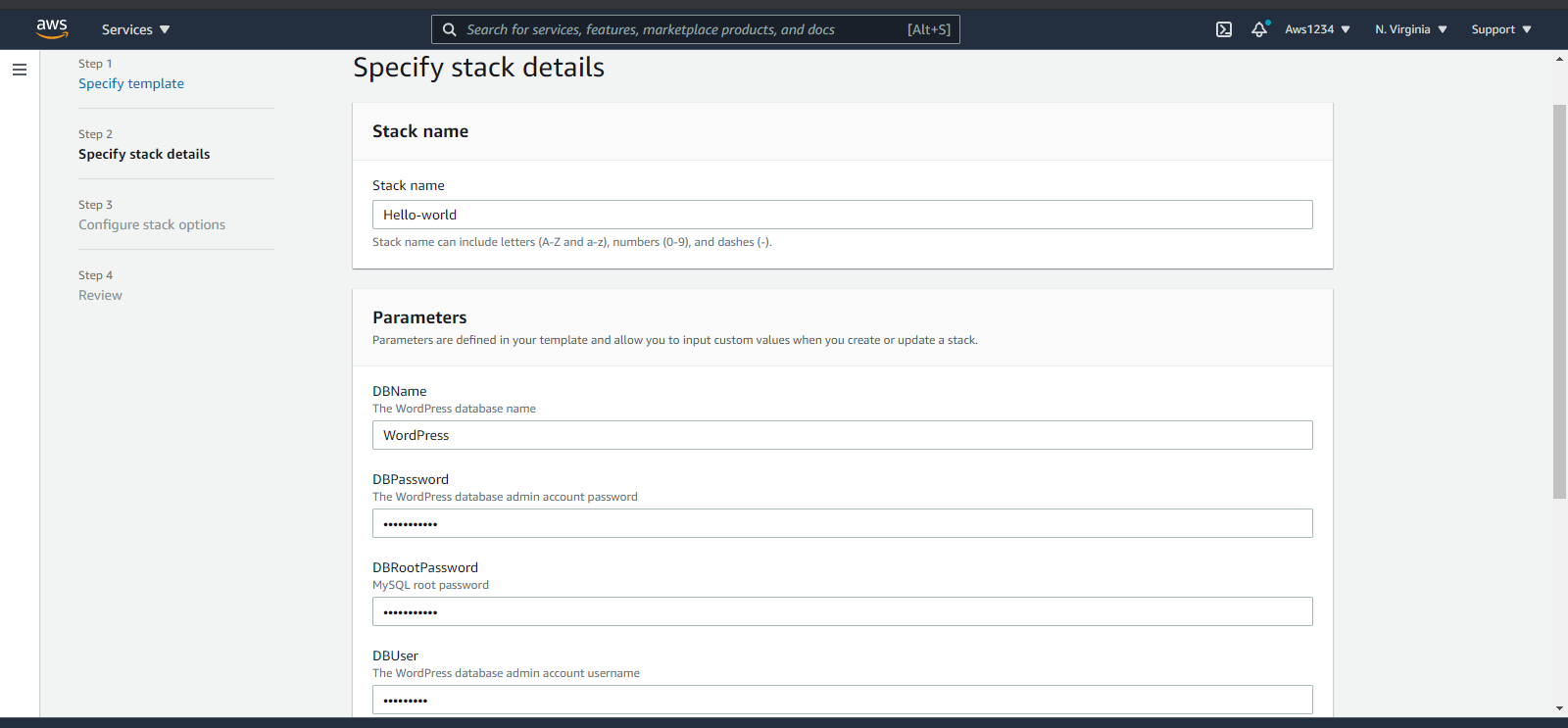
Description: WordPress Website

Value: !Sub "http://${WebServer.PublicDnsName}"

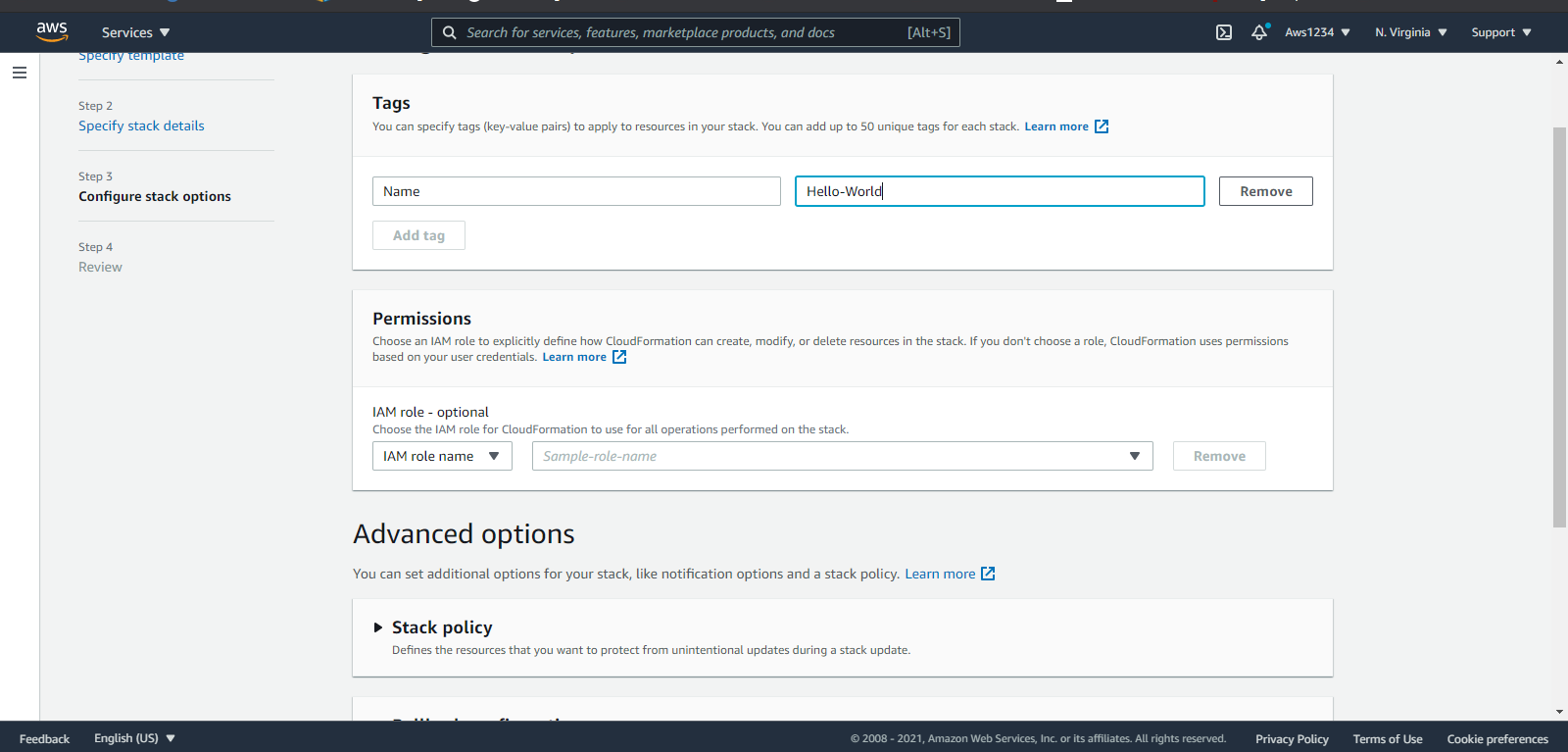
**Step2:** In created stack dashboard select template is ready option and uploaded file from my local in the format of Hello-World.yml file.



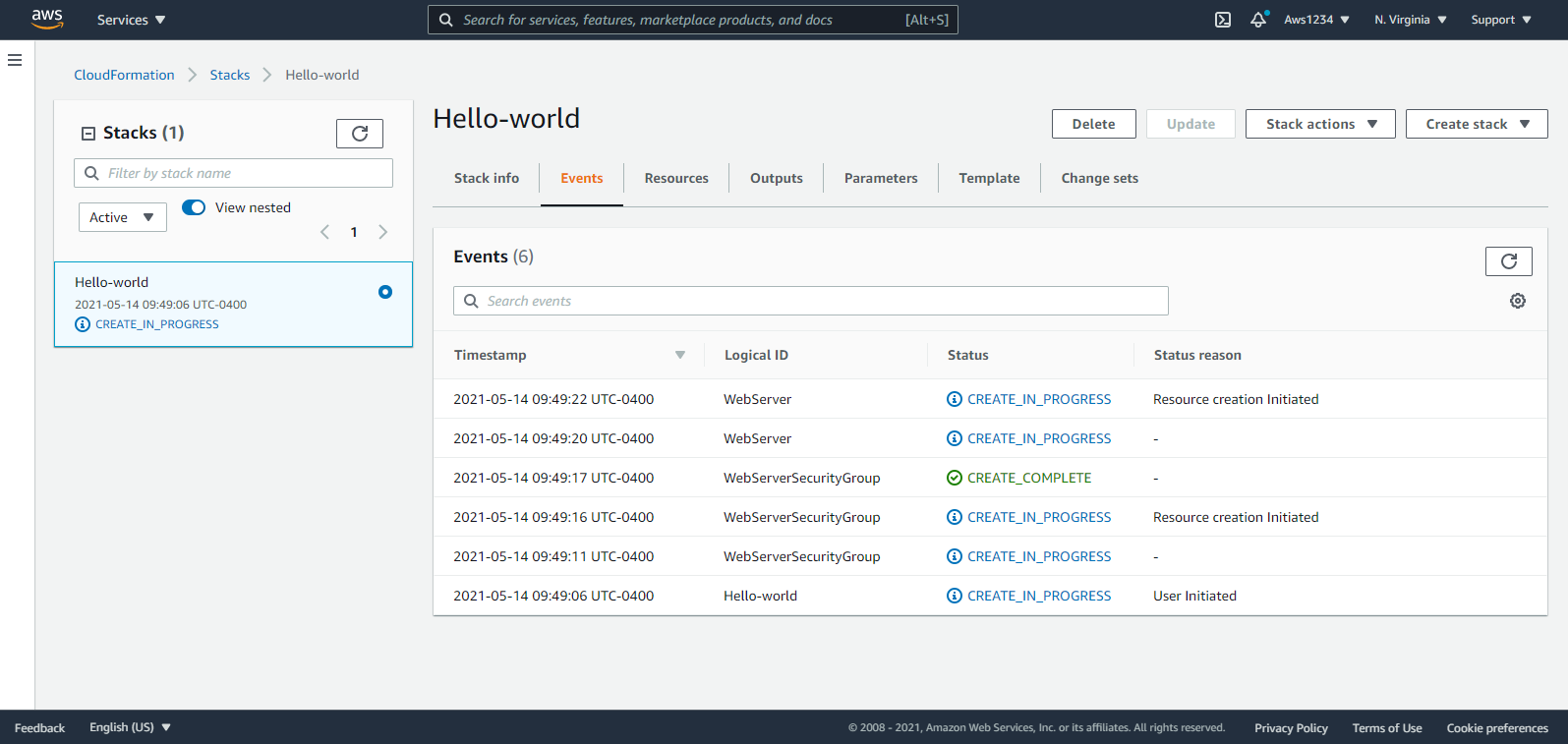
**Step3:** I specified the stack details by using stack name as Hello-World and Parameters.

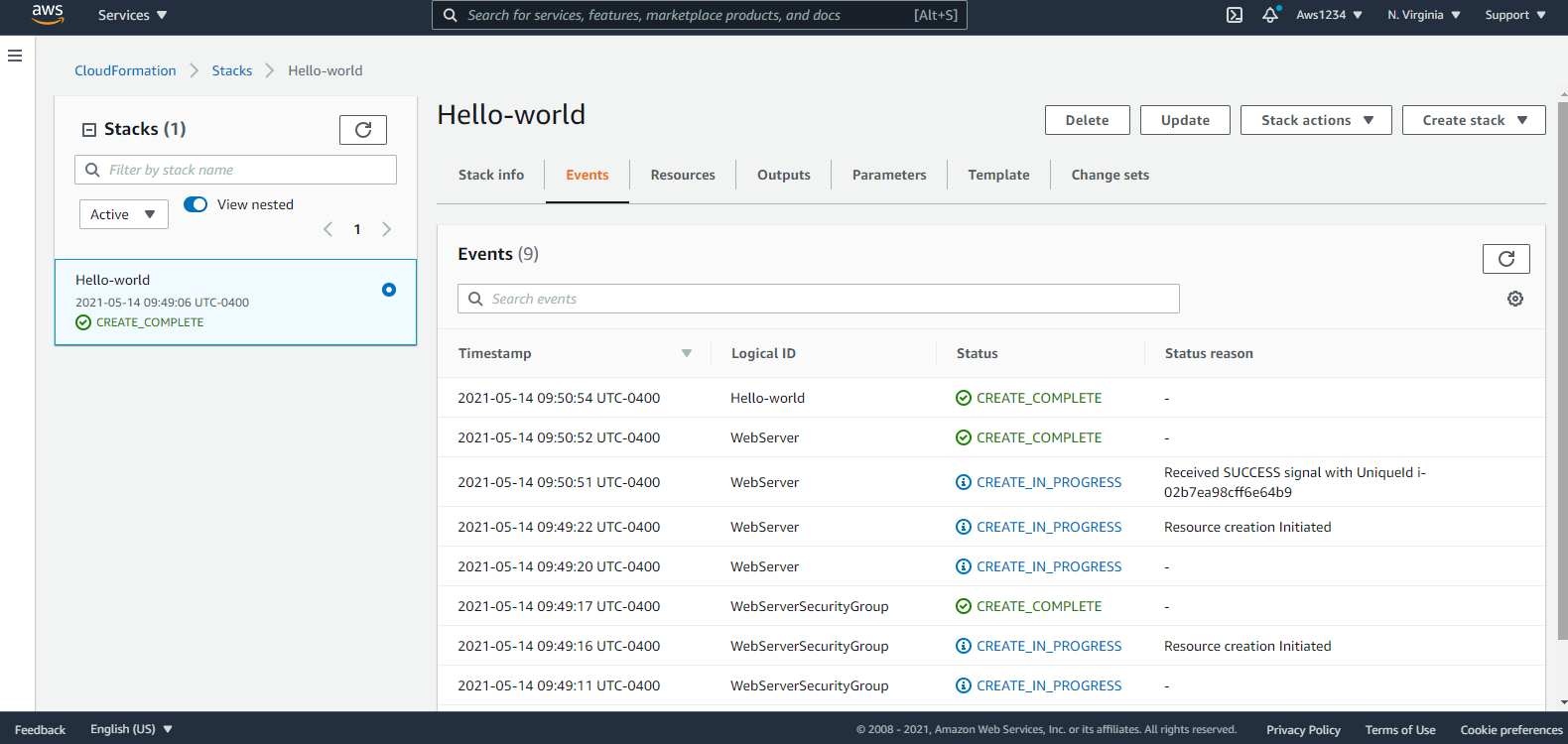


**Step4:** Weneed to review the Configured Stack Options and should be specify the tag name.



**Step5:** Once we done with review we can click on create stack then we can see that Hello-world stack status CREATE\_IN\_PROGRESS state.

  
  
**Step6:** we can notice that Hello-world stack is created and status changed to CREATE\_COMPLETE.



**Step7:** We can see that output as a **Hello-World** by visiting the DNS of the instance.

