Nickolas Kraus

Creating a Static Website using CloudFormation

February 2019 · 4 minutes

Creating static websites is a hobby of mine. This blog, in fact, was created using Hugo, a static site generator, and is hosted on AWS. The original article for creating a static website with Hugo and AWS can be found here. When I first detailed the steps for creating and hosting a static website on AWS, I used the AWS Management Console and AWS CLI. Both methods are slow and inefficient. In this article, I expedite the process using CloudFormation and a little bash scripting.

Overview

Hosting a static website on AWS makes use of the following resources:

» Amazon S3
» AWS Certificate Manager
» Amazon CloudFront
» Amazon Route 53

Prerequisites

First, you must purchase a domain name through Amazon. I plan to automate this process in the future, however, for the time being this can be done through the AWS Management Console.

Creating the CloudFormation template

The CloudFormation template is as follows:

template.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Static website
```

```
Parameters:
  DomainName:
    Description: Domain name of website
    Type: String

Resources:

  S3BucketLogs:
    Type: AWS::S3::Bucket
    DeletionPolicy: Delete
    Properties:
      AccessControl: LogDeliveryWrite
      BucketName: !Sub '${AWS::StackName}-logs'

  S3BucketRoot:
    Type: AWS::S3::Bucket
    DeletionPolicy: Delete
    Properties:
      AccessControl: PublicRead
      BucketName: !Sub '${AWS::StackName}-root'
      LoggingConfiguration:
        DestinationBucketName: !Ref S3BucketLogs
        LogFilePrefix: 'cdn/'
      WebsiteConfiguration:
        ErrorDocument: '404.html'
        IndexDocument: 'index.html'

  S3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref S3BucketRoot
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: 's3:GetObject'
            Principal: '*'
            Resource: !Sub '${S3BucketRoot.Arn}/*'

  CertificateManagerCertificate:
    Type: AWS::CertificateManager::Certificate
    Properties:
      DomainName: !Ref DomainName
      ValidationMethod: DNS

  CloudFrontDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Aliases:
          - !Ref DomainName
        CustomErrorResponses:
          - ErrorCachingMinTTL: 60
            ErrorCode: 404
            ResponseCode: 404
            ResponsePagePath: '/404.html'
        DefaultCacheBehavior:
          AllowedMethods:
            - GET
            - HEAD
          CachedMethods:
            - GET
            - HEAD
          Compress: true
          DefaultTTL: 86400
          ForwardedValues:
            Cookies:
              Forward: none
            QueryString: true
          MaxTTL: 31536000
          SmoothStreaming: false
          TargetOriginId: !Sub 'S3-${AWS::StackName}-root'
          ViewerProtocolPolicy: 'redirect-to-https'
        DefaultRootObject: 'index.html'
        Enabled: true
        HttpVersion: http2
        IPV6Enabled: true
        Logging:
          Bucket: !GetAtt S3BucketLogs.DomainName
          IncludeCookies: false
          Prefix: 'cdn/'
        Origins:
          - CustomOriginConfig:
              HTTPPort: 80
              HTTPSPort: 443
              OriginKeepaliveTimeout: 5
              OriginProtocolPolicy: 'https-only'
              OriginReadTimeout: 30
              OriginSSLProtocols:
                - TLSv1
                - TLSv1.1
                - TLSv1.2
            DomainName: !GetAtt S3BucketRoot.DomainName
            Id: !Sub 'S3-${AWS::StackName}-root'
        PriceClass: PriceClass_All
        ViewerCertificate:
          AcmCertificateArn: !Ref CertificateManagerCertificate
          MinimumProtocolVersion: TLSv1.1_2016
          SslSupportMethod: sni-only

  Route53RecordSetGroup:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      HostedZoneName: !Sub '${DomainName}.'
      RecordSets:
      - Name: !Ref DomainName
        Type: A
        AliasTarget:
          DNSName: !GetAtt CloudFrontDistribution.DomainName
          EvaluateTargetHealth: false
          HostedZoneId: Z2FDTNDATAQYW2
      - Name: !Sub 'www.${DomainName}'
        Type: A
        AliasTarget:
          DNSName: !GetAtt CloudFrontDistribution.DomainName
          EvaluateTargetHealth: false
          HostedZoneId: Z2FDTNDATAQYW2
```

To make this CloudFormation template more extensible, I pass in the domain name as a parameter via a `parameters.json` file.

`parameters.json`

```
[
  {
    "ParameterKey": "DomainName",
    "ParameterValue": "static-website.com"
  }
]
```

Validating and deploying the CloudFormation stack

```
$ aws cloudformation validate-template \
--template-body file://template.yaml
```

```
$ aws cloudformation create-stack \
--stack-name <stack-name> \
--template-body file://template.yaml \
--parameters file://parameters.json
```

Note: `create-stack` is used in order to pass in parameters as a file. The `deploy` command can be used with the addition of `cat`:

```
$ aws cloudformation deploy \
--stack-name <stack-name> \
--template-file template.yaml \
--parameter-overrides $(cat parameters.properties)
```

`parameters.properties`

```
DomainName=static-website.com
```

Validating a certificate with DNS

When you use the `AWS::CertificateManager::Certificate` resource in an AWS CloudFormation stack, the stack will remain in the `CREATE_IN_PROGRESS` state and any further stack operations will be delayed until you validate the certificate request. Certificate validation can be completed either by acting upon the instructions in the certificate validation email or by adding a CNAME record to your DNS configuration.

The **Status Reason** for your CloudFormation deploy will contain the following:

```
Content of DNS Record is: {Name: _x1.static-website.com.,Type: CNAME,Value: _x2.acm-validatic
```

Where `x1` and `x2` are random hexadecimal strings.

To automate DNS validation, you can use this script.

```
./dns-validation.sh $DOMAIN_NAME $STACK_NAME
```

Automation limitations with DNS validation

Since CloudFormation only outputs the **Name** and **Value** for the validation of the root domain name, any other subdomain that you wish to validate (ex. www), must be manually validated using the **Name** and **Value** given in the AWS Management Console.

If you want your website to be accessible via HTTPS on *both* the www subdomain and root domain, you will need to add an alternate name to the certificate and determine the **Name** and **Value** to validate the www subdomain manually:

```
CertificateManagerCertificate:
  Type: AWS::CertificateManager::Certificate
  Properties:
    DomainName: !Ref DomainName
    SubjectAlternativeNames:
      - !Sub 'www.${DomainName}'
    ValidationMethod: DNS
```

You will then be able to add the www subdomain to the CloudFront distribution:

```
CloudFrontDistribution:
  Type: AWS::CloudFront::Distribution
  Properties:
    DistributionConfig:
      Aliases:
        - !Ref DomainName
        - !Sub 'www.${DomainName}'
```

Note: DNS validation can be done manually via the AWS Management Console: **Certificate Manager** > **Create record in Route 53**.

Testing the static website

First, your static website needs to serve some content.

`index.html`

```
<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Upload `index.html` to the newly created S3 bucket:

```
aws s3 cp --acl "public-read" index.html s3://$S3_BUCKET_ROOT
```

Make a request to your static website:

```
$ http -v https://static-website.com
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: static-website.com
User-Agent: HTTPie/1.0.2



HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 17964
Connection: keep-alive
Content-Length: 61
Content-Type: text/html
Date: Sat, 23 Mar 2019 14:42:56 GMT
ETag: "bc72c661c6852e8ff3522a9484d45b41"
Last-Modified: Wed, 13 Feb 2019 20:51:42 GMT
Server: AmazonS3
Via: 1.1 53332bd6d55cfd374862eac4265e274a.cloudfront.net (CloudFront)
X-Amz-Cf-Id: CTwMePIQYYbJFjOQEOgXE-pCZKqGtiX0KbTlSgKZtiYUUgDxee8CzQ==
X-Cache: Hit from cloudfront

<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Conclusion

You now have your own static website hosted on AWS! Using a static site generator of your choosing, content can be added to your site using the following command:

```
aws s3 sync --acl "public-read" <path/to/content>/ s3://$S3_BUCKET_ROOT
```

To ensure that the cached content on the CloudFront distribution is invalidated when changes are made, use the following command:

```
aws cloudfront create-invalidation --distribution-id $CF_DISTRIBUTION_ID --paths "/*"
```

The code for this CloudFormation stack, as well as other CloudFormation templates can be found at NickolasHKraus/cloudformation-templates.

Limitations

Unfortunately, Amazon CloudFront does not return the default root object (ex. `index.html`) from a subfolder or subdirectory:

> *The default root object feature for CloudFront supports only the root of the origin that your distribution points to. CloudFront doesn't return default root objects in subdirectories.*

Amazon recommends that you can integrate Lambda@Edge with your distribution, however this is cumbersome and unnecessary. Simply create an Origin using the region-specific website endpoint of the S3 bucket:

```
bucket-name.s3-website-region.amazonaws.com
```

or

```
bucket-name.s3-website.region.amazonaws.com
```

It should be noted that Amazon S3 does not support HTTPS connections when configured as a website endpoint. You must specify **HTTP Only** as the Origin Protocol Policy for your CloudFront distribution.

For instructions on hosting a static website with Hugo, which uses subdirectory `index.html` files, see Hosting a Static Website with Hugo and CloudFormation.