Using Amazon CloudFront with Multi-Region Amazon S3 Origins

By Ben Bridts, AWS APN Ambassador at Cloudar

Link to full blog post: https://aws.amazon.com/blogs/apn/using-amazon-cloudfront-with-multi-region-amazon-s3-origins/

Appendix A: CloudFormation Templates

[Cloudfront.yaml]

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "Deploy CloudFront with Lambda at Edge to select S3 origin"

Parameters:
  OriginDns:
    Type: String
    Description: DNS record that will hold the origin S3 bucket to use

Conditions:
  WrongRegion: !Not [!Equals [!Ref 'AWS::Region', us-east-1]]

Resources:
  # This resource will only be created when trying to deploy in a wrong
region
  YouAreInTheWrongRegion:
    Type: "AWS::SSM::Parameter"
    Condition: WrongRegion
    Properties:
      # Leave name empty to force a fail
      Name: ''
      Type: String

  # We create a dummy S3 origin, that will be used if the X-DNS-ORIGIN
header is not set.
  # If we do this, we can leave the template with the real buckets 100%
seperate from this one
  # (We do need a bucket for the origin configuration)
  DummyBucket:
    Type: "AWS::S3::Bucket"

  OriginAccessIdentity:
    Type: "AWS::CloudFront::CloudFrontOriginAccessIdentity"
    Properties:
      CloudFrontOriginAccessIdentityConfig:
        Comment: !Sub "OAI created by ${AWS::StackName} in ${AWS::Region}"

  OriginDnsRole:
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
```

```
              Service:
                - lambda.amazonaws.com
                - edgelambda.amazonaws.com
        Version: '2012-10-17'
      Policies:
        - PolicyDocument:
            Statement:
              - Action:
                  - logs:CreateLogGroup
                  - logs:CreateLogStream
                  - logs:PutLogEvents
                Effect: Allow
                Resource: arn:aws:logs:*:*:*
            Version: '2012-10-17'
          PolicyName: write-logs
    Type: AWS::IAM::Role

  OriginDnsFunction:
    Type: AWS::Lambda::Function
    Properties:
      Code:
        ZipFile: |
          'use strict';
          const dns = require('dns');
          exports.handler = (event, context, callback) => {
              var request = event.Records[0].cf.request;
              if ('customHeaders' in request.origin.s3 && 'x-dns-origin' in
request.origin.s3.customHeaders) {
                  var hostname = request.origin.s3.customHeaders['x-dns-
origin'][0].value;
                  dns.resolveTxt(hostname, (err, records) => {
                      if (err) { throw err }
                      const domainName =
records[Math.floor(Math.random()*records.length)][0];
                      const labels = domainName.split('.');
                      if (labels.slice(-2).join('.') != 'amazonaws.com') {
                          throw "invalid domainName format";
                      }
                      const region = labels.slice(-3)[0];
                      // Set S3 origin to the values from the DNS record
                      request.origin.s3.region = region;
                      request.origin.s3.domainName = domainName;
                      request.headers['host'] = [{ key: 'host', value:
domainName }];
                      callback(null, request);
                  });
              } else {
                  // do nothing
                  callback(null, request);
              }
          };
      Description: !Sub "${AWS::StackName} - OriginDnsFunction"
      Handler: index.handler
      MemorySize: 128
      Role: !GetAtt "OriginDnsRole.Arn"
      Runtime: nodejs6.10
      Timeout: 1

  OriginDnsVersion1:
    Type: AWS::Lambda::Version
    Properties:
```

```
      FunctionName: !Ref OriginDnsFunction

  Distribution:
    Type: "AWS::CloudFront::Distribution"
    Properties:
      DistributionConfig:
        Comment: Multi-Region S3 CloudFront Distribution
        DefaultCacheBehavior:
          Compress: True
          # Lower default TTL to 0 seconds for easy testing
          DefaultTTL: 0
          ForwardedValues:
            QueryString: False
          LambdaFunctionAssociations:
            - EventType: origin-request
              LambdaFunctionARN: !Ref OriginDnsVersion1
          TargetOriginId: s3
          ViewerProtocolPolicy : redirect-to-https
        DefaultRootObject: index.html
        Enabled: True
        HttpVersion: http2
        IPV6Enabled: True
        Origins:
          # ${AWS::Region} isn't needed, but will prevent the 307 problem
in other regions
          - DomainName: !Sub
"${DummyBucket}.s3.${AWS::Region}.amazonaws.com"
            Id: s3
            OriginCustomHeaders:
              - HeaderName: X-DNS-ORIGIN
                HeaderValue: !Ref OriginDns
            S3OriginConfig:
              OriginAccessIdentity: !Sub "origin-access-
identity/cloudfront/${OriginAccessIdentity}"
        ViewerCertificate:
          CloudFrontDefaultCertificate: True

Outputs:
  CloudFrontDomain:
    Description: CloudFront Domain Name
    Value: !GetAtt Distribution.DomainName
  OaiS3CanonicalUserId:
    Description: Origin Access Identity S3CanonicalUserId
    Value: !GetAtt OriginAccessIdentity.S3CanonicalUserId
  OriginDnsValue:
    Description: Origin DNS Name
    # Value of the Parameter, but this makes it easy to find.
    Value: !Ref OriginDns
```