**Migration from Akamai CDN to AWS Cloudfront**

Introduction

This Document is about demonstrating the functionality of Cloud Front which is a CDN service like AKAMAI. We have assessed one of the site www-stg2.jetblue.com, we have gone ahead and mapped some of the functionality from AKAMAI to AWS CloudFront (CF).

AWS cloudFront is webservice that speeds up the distribution of web content by delivering the content from the nearest edge of the user due to which latency is reduced and performance is increased

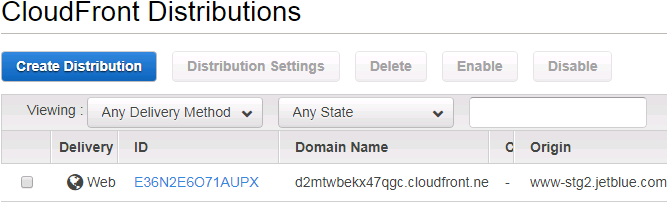
In this process we have identified several use cases and we have started working on them. We will go through each use case and explain the outcome. We have used CF as a CDN and Lamba@Edge as event trigger and automated the whole process using Terraform and also uploaded code in bitbucket.

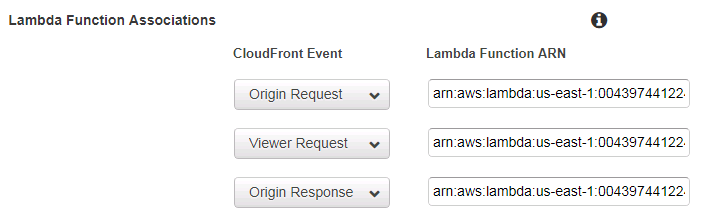
**USE-CASES**

**AWSCF3 : As a developer I need to setup the Cloudfront setup in AWS**

we have set up the coudfront in AWS with **www-stg2.jetblue.com** as a default origin.

we have triggered the lambda function on the different event triggers depending upon its functionality





**AWSCF4 : As a user I need to get the response based on the device it is requested from**

This user story demonstrates how the AWS cloudfront routes the pages based on the device request

Brief about it : When the user requests the URL it goes to AWS cloudfront distribution. AWS Cloudfront adds the whitelist headers ( **CloudFront-Is-Desktop-Viewer** and **CloudFront-Is-Mobile-Viewer**) and sets the value ( true / false depending upon the device from where the request is originated and the request is forwarded to lambda function, finally lambda function will extract the header values and based on the value it serves the appropriate page.

Lambda function event trigger is "**origin-request**"

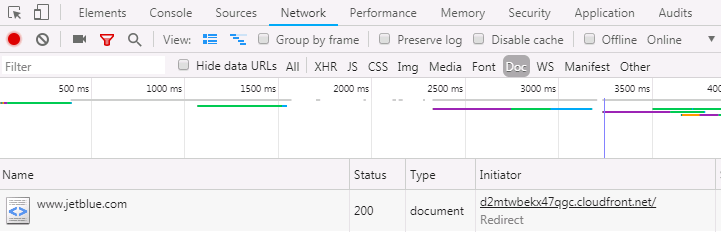
DEMO:

1.Inspect the browser set the mode to the desktop

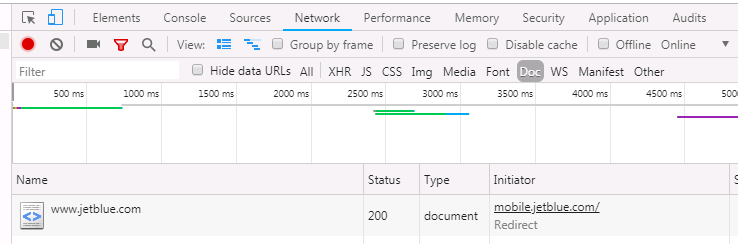
2.Hit aws Cloudfront domain "d2mtwbekx47qgc.cloudfront.net" and see inspect --> network --> doc

3. Change device mode to mobile on the browser and repeat the step 2

When the request comes from Desktop ( Desktop mode activated)



When the request comes from mobile ( mobile mode activated)

****

**AWSCF5 : As a user I need to get the response based on the respective Geo Location**

This user story demonstrates how the AWS cloudfront serves the pages based on geolocation (Mexico (MX) and Jamaica (JM) )

Brief about it: When the user requests the URL it goes to AWS cloudfront distribution. AWS Cloudfront adds the whitelist headers ( **CloudFront-Country-Viewer**) and sets the value as country code depending upon the country from where the request is originated and the request is forwarded to lambda function, finally lambda function will extract the header values and based on the value it serves the appropriate page.

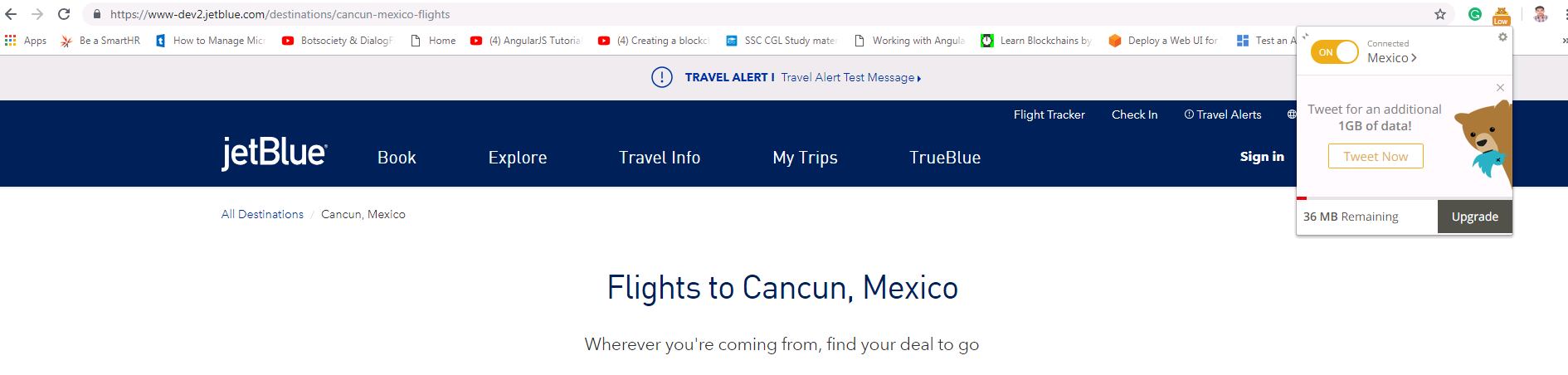
Lambda function event trigger is "**origin-request**"

DEMO: VPN - hit from mexico (MX) this will route to Mexico destination page

d2mtwbekx47qgc.cloudfront.net/country/flight

When request comes from Mexico ( VPN of Mexico Activated)

Note : Used a Chrome extension ( tunnel bear ) for VPN



**AWSCF6 : As a Developer I need to serve the Dynamic content from Custom Origin**

This user story demonstrates how the specific request are routed to custom Origin ( **dev.b6orgeng.net** )

Brief about it: This ticket calls the request from the users and if the request is among the specific URLs then it is routed to custom origin. Configuration of the custom origin is done in lambda function only.

Lambda function event trigger is "**origin-request**"

Custom origin: **dev.b6orgeng.net**

default origin: **www.stg2.jetblue.com**

Demo:

Akami rules in jetblue:

dev.b6orgeng.net/signin

dev.b6orgeng.net/travel-agents

AWS cloudfront

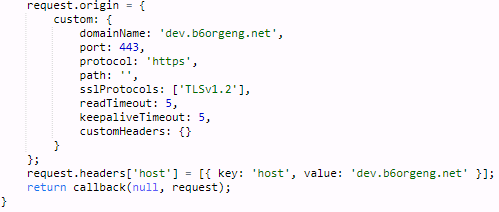
d2mtwbekx47qgc.cloudfront.net/signin

d2mtwbekx47qgc.cloudfront.net/travel-agents

List of URIs which are redirected to the Custom Origin



Lambda code snippet showing configuration of the Custom origin ( dev.b6orgeng.net )



**AWSCF7 : As a Developer I need to redirect HTTP requests to HTTPS**

This user story is to demonstrates how redirecting is happening and HTTP protocol is redirected to HTTPS protocol.

Brief about it : We have sourced that from "Redirection for Digital 2020" rules from AKAMAI luna dashboard and we have implemented the same redirection rules in AWS Cloudfront by using lambda function with redirection from HTTP to HTTPS.

We have mapped all the rules in lambda function based on the conditions.

Lambda function event trigger is "**viewer-request**"

Demo :

Requested URL :

http://d2mtwbekx47qgc.cloudfront.net/corporate-social-responsibility/

http://d2mtwbekx47qgc.cloudfront.net/baggage

<http://d2mtwbekx47qgc.cloudfront.net/congrats>

Response URL :

https://d2mtwbekx47qgc.cloudfront.net/corporate-social-responsibility/

https://d2mtwbekx47qgc.cloudfront.net/baggage

[https://d2mtwbekx47qgc.cloudfront.net/congrats](http://d2mtwbekx47qgc.cloudfront.net/congrats)

Requested URL:



Response URL



**AWSCF8 : As a Developer I need to add additional header in the response for security purpose**

This user story is to demonstrates how additional headers are added in the response for the security .

Brief about it: This ticket calls the request from the users and Cloudfront forwards the request to the Lambda function on origin-response as an event trigger which adds the headers in the response, which improves the security and privacy for a website and the response is securely delivered to the viewer.

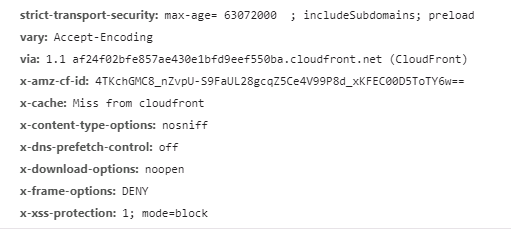
Lambda function event trigger is "**origin-response**"

DEMO: inspect the browser and go to network and doc

<http://d2mtwbekx47qgc.cloudfront.net/our-company/>

You can find the headers in the response header section

Additional headers added in the response by lambda function



**AWSCF9 : As a Developer I need to send the compressed Content in the response**

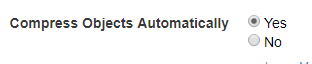
This user story is to demonstrates how the AWS CF serves the content in compressed form

Brief about it: This ticket calls the request from the users for any content and cloudfront will process the request and serves objects in compressed form in the gzip format.

DEMO: Inspect --> Network--> All-->size

d2mtwbekx47qgc.cloudfront.net/baggage

AWS Cloudfront comes with a inbuilt feature of compression, we have to only set the value to true so that all the web content delivered to the user is in compressed format



**AWSCF10 : As a Developer I need to rewrite the URL so that the end user finds it user friendly**

Need to rewrite the URL, so that the end user will have a user friendly url for access. In viewer request as an event trigger we have to map all the user friendly urls with the original URLs.

Lambda function event trigger is "**viewer-request**"

DEMO:

d2mtwbekx47qgc.cloudfront.net/csr/ (Original route is d2mtwbekx47qgc.cloudfront.net/corporate-social-responsibility/) based on the Redirection rule this will route to "/jetblue-for-good" path.

d2mtwbekx47qgc.cloudfront.net/csr/military/ (Original route is <http://d2mtwbekx47qgc.cloudfront.net/jetblue-for-good/military>) based on the Redirection rule this will route to "/jetblue-for-good/military/" path.

Requested URL :



OR



Response URL :

